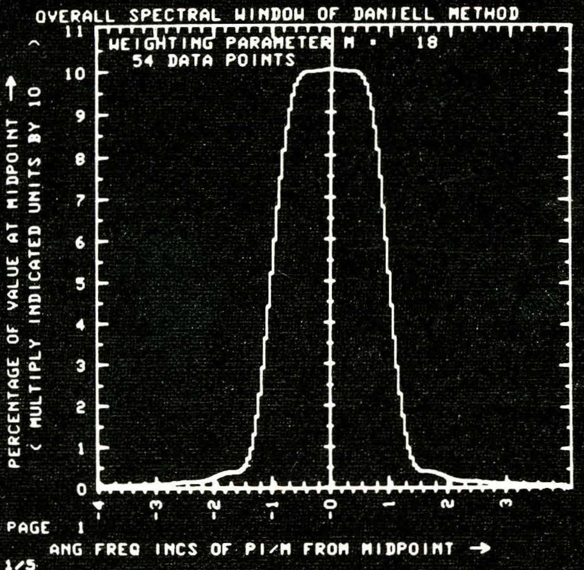
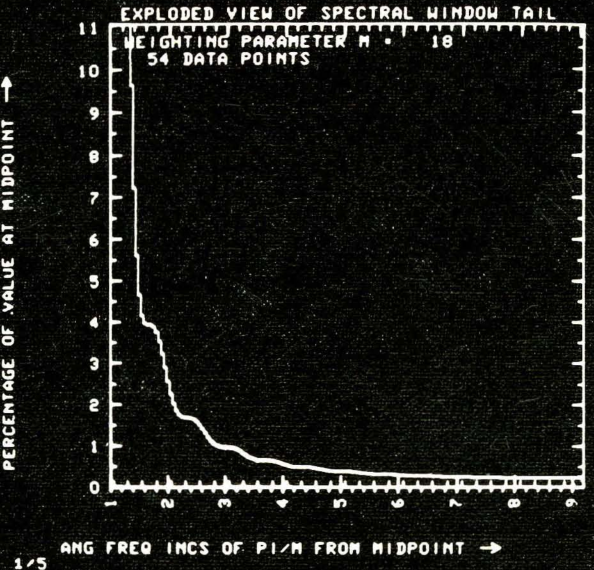
VOLUME I — A PROGRAM LIBRARY

Time-Series Computations in FORTRAN and FAP

SIMPSON





Time-Series Computations in FORTRAN and FAP

VOLUME I — A PROGRAM LIBRARY

Time-Series Computations in FORTRAN and FAP

VOLUME I — A PROGRAM LIBRARY

STEPHEN MILTON SIMPSON, JR.
Massachusetts Institute of Technology

ADDISON-WESLEY · *READING, MASSACHUSETTS*

ADDISON-WESLEY PUBLISHING COMPANY, INC.

Printed in the United States of America

Library of Congress Catalog Card No. 66-14667

ADDISON-WESLEY PUBLISHING COMPANY, INC.
Reading, Massachusetts · Palo Alto · London
New York · Dallas · Atlanta · Barrington, Illinois

ADDISON-WESLEY (CANADA) LIMITED
Don Mills, Ontario

To my mother, to my father,
and to Jackie.

Preface

In the fall of 1952 I joined, as a graduate student, a Massachusetts Institute of Technology project called the Geophysical Analysis Group, and so began a twelve-year effort in the application of digital computers to time-series problems. This project, the G.A.G., was organized by Professors G.P. Wadsworth and P.M. Hurley of M.I.T. and by Dr. Daniel Silverman of the Stanolind Oil and Gas Company. It assumed the task of attempting the realization of Norbert Wiener's time-series concepts on the Whirlwind I (WWI) Computer in the echo-sounding problems of seismic exploration for oil.

At the same time I developed a close friendship with my fellow student Enders A. Robinson, on whom the directorship of G.A.G. soon devolved. Robinson's efforts centered in the elucidation of theory and its translation to discrete notation, and my own work tended toward machine realization of theory, but we each made sufficient excursions into the other's domain to form a profitable research partnership. This pattern has persisted over the years.

The Geophysical Analysis Group is relevant for the reason that many of the programming concepts presented herein were seeded in the 16-bit registers of WWI for the seismic exploration problem. Digital prediction, both single and multiple, special digital filtering, spectral and correlation analysis, traveling spectral analysis, automatic processing systems for multirace seismograms, and many other operational concepts were developed and experimented with on WWI to an unprecedented degree. Besides myself and Robinson those involved with computation included Mark Smith, Howard Briscoe, William Walsh, Robert Bowman, Freeman Gilbert, Sven Treitel, Donald Grine, Kazi Haq, Donald Fink, Robert Wylie, Manuel Lopez-Linares, Richard Tooley, and Robert Sax. The ideas carried into industry and pursued there by students associated with G.A.G. have now ripened to the point of causing what amounts to a technological revolution in seismic interpretation.

In 1954 Robinson left, eventually to become Associate Professor of Mathematics at the University of Wisconsin, and I assumed directorship of G.A.G. until its termination in 1957, but frequent visits with each other kept alive our mutual interests.

With G.A.G.'s termination and the subsequent retirement of WWI, I was forced to the realization that my programming output might just as well have been expressed in vanishing ink—an experience which rankled long and which underlies our determination to develop stable programming and communicating techniques.

I took a year's leave of absence from my Assistant Professorship in the Department of Geology and Geophysics at M.I.T. and spent it in military applications of special-design general purpose computers with RCA. This work tended to keep me from recognizing the latent power of the then infant language of FORTRAN.

On returning to M.I.T. I kept my hand in programming on the IBM 704, but it was not until 1960, when I was asked by the Advanced Research Projects Agency to set up a project like G.A.G. but focused on the underground detection problem of VELA UNIFORM, that I became seriously involved with the new computers. I was fortunate in being able to attract Robinson to the project, as well as many gifted graduate students.

By this time FORTRAN had become well established, and, after some hesitancy, we began to use it, gradually evolving a sense of proportion in the mixture of FORTRAN and FAP programming. I find in this mixture that the whole is greater than the sum of its parts. For not only can we have the essential power of the individual languages, but they can supplement each other's weaknesses, as, for instance, they do when we use subroutine sandwiches of alternating language or use FAP programs to bolster FORTRAN's capabilities.

Once again this leaves me committed, albeit partially, to a machine language. But the situation is not as bad as it was ten years ago. In the first place, the ubiquity of the IBM 700 series machines suggests a national and international investment in specific hardware and software of considerable inertia. The time constant of change has lengthened to a point where we should be able to keep up with it without periodic wholesale abandonment of past results. Secondly, our program design, testing, and documentation techniques have matured to the point where machine language translation is not nearly as formidable a prospect as previously.

These considerations, the rapid advances which have been made in time-series computations, the growing requests we have had for the programs, and the general expanding interest in time series and in programming, have all encouraged me to pause and to pull together the myriad threads of our work into a single document representing, in first approximation, where time-series computations stand with respect to today's machines. Such has been my goal. However, this goal has proved too ambitious for a single volume, and we content ourselves in Volume I with a presentation of our subroutine library per se. Volume II will be devoted to the development of pertinent time-series theory from the computational viewpoint, to the consideration of computational applications in a realistic setting, and to discussion of programming technique.

Taken together, the first and second volumes of <u>Time-Series Computations in FORTRAN and FAP</u> may be considered an introduction to a new topic, namely, the realization of modern time-series theory on digital computers. Their principal intended audience is students of time series or communications engineering who wish to acquire advanced techniques of handling empirical time series with present-day computational equipment, especially on the IBM 709, 7090, or 7094. By "advanced" I refer both to the conceptual level of the techniques and to the professionalism of their realization.

But I would hope that this work, Volume I especially, should also prove of value to the general programming community. The majority of our programs are not specialized to the time-series area. What we have done is to fill the gap between basic FORTRAN statements and time-series operations with a complex of general-purpose black boxes that could be used to assist in the development of other areas of application. But even aside from functional utility, we hope that all computing groups faced with the problems of program exchange and communication will be interested in our experiments in communication formalisms.

The subroutine library constitutes the bulk of Volume I. It represents the distillation of years of effort of my co-workers and myself. Cost studies of programming systems of this size (about 40,000 registers) might predict a developmental price tag of about a quarter million dollars for this set. Consequently we have felt justified in devoting considerable time and effort to the development of techniques for communicating our results in the context of applied problems.

At the lowest level of communication, that is, the individual subroutine, we have tried to maintain high standards both of programming and of documentation. Toward the latter end, we have adhered to a program-writing format which might be called the self-documenting symbolic deck. In this format, the program card deck contains a program abstract and a detailed input-output specification, as well as illustrative and critical examples. The card deck is totally definitive of its own behavior.

The format was originally designed for input to an automatic debugging compiler which would read the examples, set up appropriate test programs, execute the test programs, and report back results. In the press of other business the compiler never proceeded beyond a rudimentary stage, but the format has remained and proved itself valuable in our own internal communications.

Furthermore, the format has proved itself many times over as a disciplining device for keeping programmers honest. It is a characteristic of the trade that programmers modify and remodify their decks. The juxtaposition of the documentation and the program proper in deck listings emphasizes documentation errors that result from such modifications, and the weeding out of these errors becomes a natural and integral part of the debugging process. Moreover, to a programmer, there is a great psychological difference between having to change a few comment cards and tracking down a secretary to make the same changes on a mimeograph master in order to run off an updated memorandum.

The self-documenting program deck is a black box with input-output terminals fully described. It is necessarily bulky, the description being generally several times the length of the program proper. For routine reference we turn to compressed summaries, the "program digests," which, by judicious choice of terminology, enable one familiar with the programs to refresh his memory of calling-sequence details needed while programming, with an absolute minimum of page turning. For general scanning of and access to the programs, we have sorted them by various functional and nonfunctional attributes. The other types of documentation in Volume I relate to subroutine library structure and are of more specialized interest to the system programmer.

But the study of n black boxes, each of which performs an isolated task in time-series analysis, does not give one a sense of the coherency of the subject, or of the methods of interconnecting the boxes in broad experimental applications. For such purposes we have designed the experimental programs to be presented in Volume II. Each of these programs represents a series of experimental studies in an interconnected area of time-series analysis, with some carry-over from one program to the next. They permit the reader to see essentially all of our subroutines used in an applied framework.

The applications chosen for illustration in Volume II range from elementary ones to problems the average student or research worker is unlikely to have encountered (especially multi-input processes). Since our theoretical development of time series is of rather limited scope, we have included appendixes on some of the less well-known topics covered in the experiments.

The experiments of Volume II are designed to be readable without knowledge of the basic machine language, FAP, and to require a minimum of experience with FORTRAN. The study of Volume II, especially in conjunction with practice on a computer which can accept the subprograms of Volume I, is probably the easiest way of acquiring familiarity with the techniques we have to offer.

It is an unfortunate fact that artificial but general languages like FORTRAN are, in themselves, incapable of expressing many of the critical time-series operations in truly efficient form. This situation may change, but probably not in the near future. To a large extent our subroutine library may be viewed as an interdependent collection of FORTRAN and FAP programs where the FORTRAN programs steer the FAP programs to the desired task. The higher-level FORTRAN programs will easily compile on machines outside the IBM 700 series family, but their required subordinates, the FAP workhorse programs, will not in general carry over without hand-coded translation.

For this reason, Volume II will present expositions of the more important logical processes used in the FAP subprograms to attain high-speed behavior, particularly in connection with correlation and spectral analysis. A knowledge of FAP is desirable but is not essential, since we lean considerably on ordinary flow charts for detailed relationships.

Other limitations of a formal nature inherent in FORTRAN II have led us to some programming effort in the twilight region between FORTRAN and FAP, that is, to the writing of FAP programs which utilize "forbidden" knowledge of the FORTRAN system in order to remove these limitations, and which we therefore label "system-expansion programs." Volume II includes a discussion of the techniques and problems involved in such programming and should prove of interest to serious students of programming.

In short, then, we have limited the first volume to the presentation of the subroutine library with subsidiary documentation designed for the working programmer, and we have reserved time-series and programming concepts for Volume II.

The "we" I use frequently above is not editorial, but includes my many coworkers, mostly graduate students, who have contributed to the subprogram collection and with whom it has been my pleasure to work. In this congenial and loosely structured group, considerations of programming technique and style were developed to refined levels. Although the authorship of the programs is given individually, I would like to emphasize the importance of the contributions of James Galbraith, Jon Claerbout, and most particularly Ralph Wiggins. Other students directly associated were William Ross, Cheh Pan, Carl Wunsch, and Roy Greenfield.

As for what theory we include in Volume II, much of it is pure review, but some of it has previously appeared only in project report form. I consider Robinson's solution, in the fall of 1962, of the multi-input iteration problem to be a significant achievement. Wiggins pursued and expanded the analysis from this base through the program-development stage, and in so doing was the first to demonstrate the computational feasibility of multi-input least squares.

But the work presented here has also depended on many others. The tireless and dedicated writing of test routines by Joseph Procito has been invaluable in the establishment of program reliability. In broader areas of service programming, analysis, data handling, desk calculating, etc., we also relied on Mrs. Irene Hawkins, Karl Gentili, my wife Jacqueline, Ervinia Irbin, Mrs. Susan Kannenberg, Allan Kessler, and Lloyd Kannenberg. Most of the card preparation and manuscript chores fell to Mrs. Elizabeth Studer, to my wife, and to Mrs. Wendy Tibbets, with assistance from Mrs. Elene Hershberg, Dauna Trop, Mrs. Myrna Kasser, Regina Lahteine, Mrs. Hazel White, and Mrs. Barbara Cullum.

The punched-card work involved in these two volumes is too elegant to be passed over without further comment. The conventions and forms that we now use regularly (not all of which appear in these volumes; for instance, the mathematics of Volume II was card-coded in the source manuscript) I consider to be significant experiments in a field — call it "punched-card typography" — of growing importance in printing. In large part these conventions are due to my wife, who has become our arbiter of formats and to whose sense of style and standards of excellence we are much indebted.

Over the years we have been favored with the most friendly cooperation of the machine operators and supervisors, starting in the early Whirlwind days with Robert A.J. Gildea (to whom I also owe many enjoyable hours of chess while waiting for the machine to come back) and Michael Solomita, and continuing with Anthony Sacco at the M.I.T. Computation Center and at the Cooperative Computing Laboratory at M.I.T., John Harmon and our long-term friend Michael Saxton of IBM, and more recently with Thomas Burhoe, Mason Fleming, and William Jarvis of IBM.

We owe much to the sponsors of both the G.A.G. project and the VELA UNIFORM project for the computing facilities these projects have afforded us in the development of time-series and computing concepts, and to Lincoln Laboratory, the M.I.T. Computation Center, and Geoscience Incorporated for the use of programs developed under their auspices.

Concerning editorial assistance, I am indebted to Robinson for critical review of the mathematical aspects of the manuscript and to Wiggins for his joint labors with my wife and myself in the editing of the programs.

It is indeed a pleasure for me to acknowledge the many contributions and accommodations from this small army of co-workers and associates.

Brookline, Massachusetts
November, 1965

S.M.S., Jr.

Contents

1
Introduction

The heart of this book is the presentation, in Section 10, of 267 programs which are rather widely applicable even though their development was motivated by problems in the field of time-series analysis. The reader may turn to any one of these programs and study it with understanding without a need for the material in the preceding sections, these sections being concerned with introductory and access information, and with tabulation of data abstracted from the programs. In particular, the present section is concerned with an overview of the programs as a set, and with general considerations of language, terminology, and programming philosophy.

The use of this book and of the programs presupposes some familiarity with the artificial computing language FORTRAN and, but to a lesser and nonessential extent with the machine language FAP. A reader with only FORTRAN background should be able to read through all of the textual material of the introductory sections in one sitting and lose very little from our occasional references to machine-language details. Thereafter he should have no trouble in locating programs of interest by means of the categorized lists of Section 3, or in utilizing, with the aid of Sections 4 and 5, the programs he has become familiar with. However, in his reading of the programs in Section 10, a person of this background will generally be limited to the FORTRAN programs, although if he has sufficient curiosity he will find that many of the machine-language programs are quite easy to follow with the aid of a machine manual (95 of these programs involve less than 50 machine-language instructions and constants, 31 less than 25).

Many of the tabulations to follow contain data on program storage lengths and binary card counts; these data are somewhat dependent on the particular system used to translate the symbolic decks into machine language. The statistics given pertain to FORTRAN II, Verson 2, IBM Modifications 1 through 27, further modified to accept the G format.*

---

*In reference to the monitor system, one might note that we have found it useful to modify the BSS loader to extend its limit on the maximum number of missing subroutines from 50 to 200. (This is accomplished by reassembling records 7, 8, and 9 of the FORTRAN Monitor System after appropriately redefining the symbol NMMSP.) Without this change large main programs referring to many library subroutines occasionally have their executions blocked. An alternative, if ad hoc, solution to this problem is to reduce the missing subroutines count by adding a number of the required routines to the input deck when the problem arises, rather than by obtaining them from the library.

GENERAL ASPECTS OF THE PROGRAM SET

To begin our general view of the program set we will comment briefly on the variety of functions performed. Examination of the categories of Section 3 will show a wide diversity in the computational topics embodied, these topics ranging from spectral analysis and discrete filtering (convolution) to matrix manipulation, to polynomial operations, to machine graphing, to technical matters of program administration, and to a number of other topics. Conversely there are some programs which essentially or actually duplicate the functions of others. There are several reasons for such redundancy. Sometimes the reason is to provide both FORTRAN and FAP versions, and sometimes it is to illustrate alternative programming techniques, but more often the reason is historical accident. The redundancies are preserved in our use because of references made in main programs not shown, and occasionally because of differences in taste, but other groups might find it profitable to trim down the collection. The tables of Sections 8 and 9 are useful for checking the consequences of contemplated program deletions.

Similarly, one will find considerable diversity both in program size, as measured by storage requirements* which vary from 1 register to 1499 registers with an average of 152, and in complexity, as measured by calling-sequence lengths, which vary from 0 to 22 with an average length of 4.5; by number of entry points, which varies from 1 to 18 around a mean of 1.5; and by the required number of pages of descriptive documentation, which ranges from 0.5 to 16 with an average of 1.4.

Necessarily we must also admit to some diversity in the technical quality (not in accuracy or utility) of the programs. In general, the quality will have a positive correlation with the date of the writing. Such quality problems as may exist are most often due to awkwardness of design or of expression, resulting in programs larger than necessary. However, the critical program loops are usually very efficient despite these factors.

Of the 267 programs, 90 are written in FORTRAN language, which is acceptable to most computers, and 177 are written in the FAP (FORTRAN Assembly Program) language, which is applicable only to the IBM 709, 7090, and 7094 computers.** The average length of the FAP programs (85 registers) is distinctly smaller than that of the FORTRAN programs (283 registers). All of the programs are subroutines in the general FORTRAN sense of the word. The FAP subroutines conform to the subroutine linkage requirements of FORTRAN II and consequently can be used by FORTRAN programmers who are unfamiliar with the FAP language.

It must not be assumed, however, that the 90 FORTRAN programs can be used immediately on computers other than the 709 series machines, or that they may be operated under FORTRAN IV. For the program library is strongly interconnected, and although each program is usable by the programmer as an apparently independent entity, many of them internally require the services of up to 16 other programs from the library. It turns out that only 23† of the FORTRAN programs either need no other programs from the library, or, if they do, need only programs which are themselves FORTRAN. Thus the library in present form is by and large specialized to operations under the FORTRAN II Monitor System of the IBM 709, 7090, or 7094.

The program changes needed to permit operation under FORTRAN IV on the IBM 709 series machines are minor compared to those necessitated by a change of computers, but are still more extensive than a specialist in such matters might guess from what has been said so far. The standard changes with regard to transfer of

*The numbers here are exclusive of lower-order programs that might be required. Including these, the range is 1 to 5106.

**Thirteen of the programs will work only on one or two of these three machines.

† Some of these 23 programs require the use of FORTRAN system routines.

control, to the direction of storage of subscripted arrays, and to the binary point of fixed-point numbers must be made for all FAP programs needed. But about 30 of the FAP programs depend on more than routine knowledge of FORTRAN II (some scan the calling program ahead of or behind the calling sequence, some have variable-length calling sequences, some refer to non-FORTRAN-callable system routines, and some utilize data left behind by the monitor). These programs require additional and more involved changes. Offsetting this complication, however, is the fact that a number of these unorthodox programs have the function of expanding the capabilities of FORTRAN II in ways now included as built-in features of FORTRAN IV, and consequently this number can be dropped entirely, providing suitable changes are made in programs referring to them. On the other hand, there will be cases where the possibility exists of choice between FORTRAN II and FORTRAN IV, and here one should balance the advantages of FORTRAN IV over FORTRAN II as bolstered by our system-expansion programs against the required changes in the particular programs needed.

In this volume, little direct help is provided for the problem of translation of the FAP programs for use on other machines. As a practical matter, however, it should be pointed out that many, perhaps most, of the FAP programs are of such elementary nature that their functional description and examples as given in Section 10 are all that coders for other machines will want. (They are more likely to consider many of the programs to be beneath their dignity.) The more involved FAP programs may require the services of an experienced translator. The second volume of this work will give numerical analysis discussions and flow charts of value in these cases.

The programs of Section 10 are alphabetically ordered by program name where the name of a program is, under ordinary circumstances, taken to be identical to the name of the entry point. For FAP programs with multiple entries the program name is taken from the first entry card in the deck. However, the alphabetized page headings of Section 10 do include all principal and secondary entries, in the latter case merely giving a reference to the associated principal entry. If the program will operate only on the 709, we append (709) to the name; if it will not work on the 709, (7090) is appended; and if it works only on the 7094, (7094) is appended. (None of the programs work only on the the 7090.) There are some programs of identical entry name (they always perform identical or practically identical functions), and these are distinguished by appending the serialization –II or –III.

TERMINOLOGY BACKGROUNDS

In the foregoing review we have been using a number of undefined terms, such as "program," "subroutine," "compiler," and "entry point," on the assumption that the reader is more or less familiar with them, at least in FORTRAN usage. We now would like to clarify usage for some of these terms. Unfortunately the attempt to capture their general meaning with precision leads one into more extensive discussions of topics concerning computer hardware and input-output devices than we wish to engage in, and we shall be satisfied with some of the salient definitional features of the broadest of these terms, namely "computer program."

As a trial definition, let us take the term computer program to mean in general, "the representation of a plan of activities which could be carried out by a computer, where the activities possess a logical completeness and integrity with respect to some motivating function." The program user is interested, in the first instance, in the nature of the motivating function and in those aspects of the plan which enable him to understand the program assumptions, or inputs, and the program results, or outputs. The technical substance of the term, however, is contained more in the natures of the "representation" and of the "plan," and in the interpretation of the phrase "could be carried out by a computer" than it is in the utilitarian aspect of the program.

The problems of meaning here, which are just beginning to be of concern in questions of law, are severe now and are likely to become worse with time. For the massive efforts going on in compiler development are continually expanding and diffusing the boundaries of meaning. To give an example, what is now commonly called a "FORTRAN program" could hardly have qualified as a computer program had it appeared back in 1950 (it would have been referred to probably as a form of algorithm), not because of the absence of a suitable computer but because there was no compiler at that time to give operational meaning to a FORTRAN program. At the present time it is easy to conceive of a compiler which will accept, say, building blueprints and generate programs to produce complete purchasing lists, construction schedules, etc. Is a blueprint then to be considered a computer program? Let us set this question aside for a moment.

What is happening is that compilers and input-output devices are being taught how to read and respond not only to specification languages, that is, representations of plans of activities especially invented for computers, but also to many such languages established prior to the development of modern computers. This educational process drives the perimeter of meaning for the term "computer program" outward so as to overlap accepted usage in older disciplines in which people are now seeking to tap the potential of the big machines.

These remarks point up a logical complication in our trial definition of the term "computer program," namely that this definition is clearly dependent on the meaning of the as yet undefined term "compiler," for a compiler, which may be classified briefly as a program-to-program translator, is itself a computer program. The definitional circle involved here can be broken by resorting to a recursive form of definition which uses the concept of a "machine-language program" to provide a semantic link to hardware.

Strictly speaking, we can define a machine-language program to be a representation of a plan of activities for a given computer, which is fully detailed in that it explicitly and individually specifies the desired initial physical state of every memory element in the computer which will participate in the activities. Thus for a binary computer the machine-language program might be a punched paper tape, where each potential punch position on the tape is equated by correspondence assumptions with an individual binary memory element. (Note , however, that an octal shorthand of the binary expressions on tape would not be a machine-language program in the strict sense, since such specification, while explicit, is not individual.) A loader (whose generalizations are "assemblers" or "compilers") is then a device or procedure aware of the correspondence assumptions and capable of forcing the physical states of the memory elements in question to correspond to the specifications of any given machine-language program.

The present usage of the term computer program is, then, more closely approximated by a "representation of a plan of computational activities which is either a machine-language program or else can be translated into a machine-language program by a computer responding to another machine-language program or by a succession of such translations." Thus one or more computers may be involved in the translation, and none of these is necessarily the same as the computer on which the original program is eventually executed. The translation programs are called assemblers or compilers.

It is useful to widen the meaning of "machine-language program" to include all programs written in a machine-dependent language under which the programmer has unrestricted and easy access to every capability of the machine. The term absolute machine-language program can be used to refer to the stricter usage when necessary. The more general machine language is essentially a symbolic shorthand notation for the absolute machine language. An assembler is then a translation program for machine-language programs in the wide sense.

Introduction

Our redefinition of computer programs evidently supports the affirmative position on the question raised earlier concerning blueprints, that is, Should blueprints be considered as computer programs once the compilers can handle them? Nevertheless a strong negative position can be developed. It stems from a critical analysis of what constitutes a "plan of activities for a computer."

The basic activity of a computer is computation, or calculation on numbers. The numbers calculated on are usually physically disjointed from the commands of calculation and are gathered in an area labeled data, while the commands are gathered in an area generally accepted as program. (Note that the FORTRAN language is designed in a way which formalizes this division and heightens the impression of absolute distinction between program and data.) Moreover, the numbers are frequently prepared on, say, a card deck and read into the data area by program commands prior to calculation. Such a card deck is not considered a program in any sense of the word.

But if the basic activity of a computer is calculation on numbers, the essential defining activity which distinguishes a computer from an overgrown desk calculator is calculation on the program itself. The proper understanding of this feature, abortively introduced by the precocious Babbage over a century ago, is often the most confusing obstacle a programmer must master in the study of his first machine. The confusion is due to the possibility, in fact the necessity, of the occurrences of ambiguity between program and data.* It is the transposition of this ambiguity to the level of compilers which makes our hypothetical question of blueprint classification truly a moot one.

Thus the card decks which are processed by programs may contain information other than numbers for calculation. In particular they may contain numbers and symbols which indicate to the program the user's desired specializations, selections, or sequencing among alternative computational capabilities built into the program. Such decks are no longer thought of as data decks but rather as control decks. The plan of activities begins to migrate from the program proper to the cards. In the limit the cards themselves can become a new program in their own right, and the processing program becomes a compiler.

When can one say that this limit has been reached? A useful measure to apply is the range of controls one can exercise with the card deck. If this range covers all or most of the actual machine capabilities,** as it does in FORTRAN, then the control deck may clearly be classed as a computer program. As this range narrows, the plan of activities must be said to reside more and more in the program which processes the cards.

A blueprint must be considered to be analogous to the control cards of the foregoing discussion. Clearly the range of controls possible is highly restrictive; there would probably be no way, except perhaps a highly artificial one, to request, for example, the sum of one hundred numbers. The true plan of activities for the computer is a combination of the blueprint and the compiler. In the light of the present discussion, the blueprint may be viewed as control information for this plan, or even as a plan of activities for the compiler, but not in itself as a true computer program.

---

*For an illustration in the present program set see subroutine PROCOR. PROCOR produces a computer program in response to an arbitrary array of numbers and may therefore be thought of as a specialized assembler whose input "program" is the number array.

**We are speaking through this discussion of "general purpose" digital computers, which we leave as an undefined concept. At present most of the major computers are sufficiently similar in respect to capability that each one can simulate the behavior of any of the others, as well as that of the prototype "Turing machine."

USAGES IN THE PRESENT VOLUME

For present purposes a "computer program" is a card deck (or any of its translations or transmissions produced by compilation, assembly, listing, card-to-tape loading, etc.) prepared according to the rules of FORTRAN II programming or of FAP programming for the 709, 7090, or 7094. Since these fully documented rules prescribe well-defined program entities, ticklish questions of shades of meaning do not arise. Nevertheless it is of some value to review here some highlights of subroutine notation, since all of our programs are of this type, as well as to discuss a few notational conventions of our own.

The expression computer routine is usually used to describe a program whose functional motivation, while possibly complete in itself, is somehow subsidiary to the principal computational thought under consideration. The "routine" may or may not be merely a segment embedded in a larger program. In any event, compiling and assembly systems provide formal rules by which principal and subsidiary computational thoughts may be linked with respect to program flow and information exchange, and routines written under these rules are known generically as subroutines. But there are exceptions and inconsistencies in usage. Thus in FORTRAN manuals the word "subprogram" tends to vie with "subroutine" for the generic title, since we see references both to "subprogram-type subroutines" and to "subroutine-type subprograms." This terminology problem, though not of great practical concern, is necessarily present in this volume since all the programs here are written so as to be FORTRAN-compatible. The ambiguity is relieved somewhat by adopting the following positions.

(1) "Subroutine" can refer either to the general class of subsidiary computations linked by formal rules to a larger computational scheme, or it may refer to a particular form of such linkage, the reference being apparent from context.

(2) When "subroutine" refers to a particular form, then it must refer to the functionally most general such form within the given class of forms.

Thus the specific form known in FORTRAN as the subroutine or (subroutine-type) subprogram may be considered most general in that its inputs and outputs are unrestricted in form, whereas the other subprogram types, known as functions, are restricted to having scalar-valued outputs and in some case inputs.*

From the practical point of view, however, our problem is merely to review the rules which distinguish among the three kinds of FORTRAN-style subroutines that appear in this book. The first of these is the ordinary subroutine subprogram, which is defined by the appearance at the beginning of the FORTRAN deck of a statement of the illustrative form**

```
SUBROUTINE SUB(A,B, . . . ,D)
```

where SUB refers to one to six alphanumeric characters starting alphabetically, but terminating with F only if less than four characters are involved, and where A,B,. . .,D

---

*Unfortunately even here we would have to yield to the technical argument that a FORTRAN function may have general outputs in addition to its scalar output (the function value), on which basis the FORTRAN function could be claimed as the most general subroutine type, although the design intent and description seem to center on the scalar output.

**One or more RETURN statements are usually included but are not mandatory. Similarly, in our FORTRAN Monitor System (FMS), the subroutine need not refer to all or even any of the names of its arguments.

is a list of nonsubscripted names, all different from SUB, which are the arguments of the subroutine (the list may be void, in which case the parentheses are suppressed), the names being those of variables or of other subroutine subprograms or FORTRAN functions.

This type of subroutine is referred to from another FORTRAN program by a statement such as

```
CALL SUB(E,F,...,H)
```

where E,F,...,H is a list of arguments each of which (1) would form a legal right-hand side to a non-Boolean arithmetic statement, or (2) would form a legal alphanumeric field in a format, or (3) is a name appearing on an F card in the calling program.

The arguments E,F,...,H should match A,B,...,D in mode (e.g., fixed point or floating point) and number. Moreover there must be understanding between the calling program and the subroutine concerning each argument which is a subscripted array. This is most easily achieved by making corresponding DIMENSION statements (identical except possibly for the variable name) in the two programs. But the DIMENSION statements do not necessarily have to agree, even with respect to number of subscripts (the same holds for variables equated by EQUIVALENCE statements). What is necessary is that the two programs reach an agreement based on the following rules governing the absolute machine location of a subscripted quantity:

```
LOC(A(I))      =  LOC(A(1)) - (I-1)
LOC(B(I,J))    =  LOC(B(1,1)) - (I-1) - (J-1)*IDIMEN
LOC(C(I,J,K))  =  LOC(C(1,1,1)) - (I-1) - (J-1)* IDIMEN
                                   - (K-1)*JDIMEN*IDIMEN
```

where LOC() symbolizes "absolute machine address of," and where we are assuming a DIMENSION statement of the form

```
DIMENSION A(IDIMEN),B(IDIMEN,JDIMEN),C(IDIMEN,JDIMEN,KDIMEN)
```

Note that the first equation above does not involve a dimension. Consequently it is frequently useful to have the subroutine first dimension all of its arrays as singly subscripted quantities (with dummy values of the dimension) and then access the elements using the above relations plus values of the dimensions given to it in the calling sequence. For example, if the calling program has

```
DIMENSION C(10,20,3)
    .
    .
    .
ID = 10
JD = 20
CALL SUB(C,ID,JD,...)
```

and the subroutine has

```
SUBROUTINE SUB(A,IDIMEN,JDIMEN,...)
DIMENSION A(1)
```

then the subroutine can acquire, for example, C(5,15,2) by the statements

```
L = 5 + 14*IDIMEN + JDIMEN*IDIMEN
X = A(L)
```

By using this type of scheme (not required in FORTRAN IV), it becomes unnecessary to recompile the subroutine for each calling program having different DIMENSION statements. We use it very frequently in the programs of Section 10.

The translation to FAP of the statement

```
CALL SUB(E,F,...,H)
```

is symbolically

```
TSX     $SUB,4
TSX     E,O
TSX     F,O
 .
 .
 .
TSX     H,O
```

where $SUB is a reference to the transfer list discussed below, and where E,F,...,H now stand for machine locations containing the corresponding arguments. For each argument which is an array in the calling program but which appears in the CALL statement with no subscripts, the location is assigned as though it had appeared with all of its subscripts set to value one.

The FORTRAN function (of which there are only two in the present set) is defined by the appearance at the beginning of the FORTRAN deck of something like

```
FUNCTION FNCTN(A,B,...,D)
```

and must include a RETURN statement preceded by an arithmetic statement of the form

```
FNCTN = ...
```

where FNCTN obeys the same naming rules as SUB above, and where A,B,...,D is similar to the same expression in the subroutine-subprogram case but must not be a void list.

The FORTRAN function is referred to from another program by an arithmetic statement such as

```
X = ...FNCTN(E,F,...,D)...
```

where the right-hand side of the equality is any legal FORTRAN expression which treats FNCTN(...) as a single number. The mode of this number is assumed determined by the function name according to FORTRAN naming conventions for variables E,F,...,H and A,B,...,D must match each other in the same manner as discussed above. The translation to FAP is the same as that of a subroutine subprogram, with $FNCTN, 4 replacing $SUB, 4, except that the statements immediately following the TSX H,O will assume the value of the function, i.e., the single number generated by the function, to be in the accumulator.

The third type of subroutine is the closed (or library) function, of which there are many examples in Section 10. This type must be hand coded with a structure such as

```
        ENTRY FNCTN
          .
          .
FNCTN   STO     A
          .
          .
        CLA     VALUE
        TRA     1,4
```

The reference to the closed function from a FORTRAN program is the same as a reference to a FORTRAN function, except for the following differences: F* is appended to the name, the function value is considered fixed-point if and only if the name of the function begins with X, and the arguments in the string E,F,. . . ,H may not be alphanumeric fields or names of subroutines. The information linkage is quite different and less uniform, however. The four statements in the table below, with their effective translations, illustrate the information linkage adequately.

```
X   =   FNCTNF(A)
CLA     A
TSX     $FNCTN,4
STO     X
```

```
X   =   FNCTNF(A,B)
LDQ     B
CLA     A
TSX     $FNCTN,4
STO     X
```

```
X   =   FNCTNF(A,B,C)
CLA     C
STO     32765     (DECIMAL)
LDQ     B
CLA     A
TSX     $FNCTN,4
STO     X
```

```
X   =   FNCTNF(A,B,C,D,E)
CLA     E
STO     32763     (DECIMAL)
CLA     D
STO     32764     (DECIMAL)
CLA     C
STO     32765     (DECIMAL)
LDQ     B
CLA     A
TSX     $FNCTN,4
STO     X
```

In addition to closed functions, it is of course also possible to hand-code subroutines and FORTRAN functions. The formal structure is similar to that for the closed function. Two examples are

```
        ENTRY   SUB
          .
          .
SUB       .
          .
          .
        TRA     N+1,4
```

```
        ENTRY FNCTN
          .
          .
FNCTN     .
          .
        CLA     VALUE
        TRA     N+1,4
```

*In the tabulations of this volume the terminal F is not considered to be part of the proper name of the function.

where N is the argument count of the pertinent statement in the calling program, and where the hand coding is done subject to the argument transmission conventions illustrated earlier.

In the listings of Section 10 the expressions FORTRAN subroutine and FAP subroutine under the "language" heading always refer to the subroutine subprogram, and the expression FAP function to closed functions. The two FORTRAN functions are so labeled.

Hand coding of subroutines, unlike FORTRAN coding, permits the bunching of many subroutines in one program deck (which is often useful if the subroutines perform similar operations). Thus

```
        ENTRY    SUB1
        ENTRY    SUB2
        ENTRY    FNCTN1
        ENTRY    FNCTN2
SUB1
          .
          .
          .
SUB2
          .
          .
          .
        TRA      5,4
FNCTN1  STO      A
        STQ      B
          .
          .
          .
        CLA      VALUE1
        TRA      1,4
FNCTN2
          .
          .
          .
        CLA      VALUE2
        TRA      6,4
```

might be a "single"program representing two subroutine subprograms, each of which has four arguments, one closed function of two arguments, and one FORTRAN function of five arguments, all four subroutines needing access to the same table of numbers.

This type of multiple-entry coding in FAP, appearing frequently in the library of Section 10, clouds the meaning of the term "program." From the standpoint of the calling program, each entry of a multiple-entry program is used as an independent subroutine; the calling program has no way of knowing that they are dependent. If a reference is made to just one of them, the loading program must nevertheless bring the entire bunch into the memory as a unit, since the physical deck cannot be divided. For example, the standard FORTRAN functions COS and SIN are separate entries to a single program and are always together in the machine if one of them is.

We might speak of logical programs and physical programs to clarify intention when necessary. For program-writing purposes, however, there is never any necessity to refer to other than logical programs. In any case, it has become customary in many circumstances to bypass the question by simply referring to entries or entry points. This terminology relates to the fundamental topic concerning "program" in the conception of the control hardware, namely, where to send control for the next job, and is neutral with respect to higher-level distinctions made by compilers. (Note that in the printed output of the compilation of a FORTRAN program one finds a list of required logical programs accurately entitled "entry points to subroutines not output from the library.")

For purposes of dealing with program decks and of general documentation, on the other hand, one must refer to a multiple-entried program as a unit. The manner of doing this is a matter of local convention. We have chosen to equate the name of each physical program with the name on its first entry card and to speak of that name as the principal entry, other entries being termed secondary entries.

When FORTRAN or FAP is processing a program, it forms a complete nonredundant list of all of the entry names referred to by the program. This list, which appears in BCD form in the first registers of the absolute relocatable binary deck produced by the translation, is called a transfer list or, as it is called more often in this book, a transfer vector. Each reference to a subroutine in the program body, that is, each TSX $SUB,4, becomes TSX A,4 where A is the register in the transfer list containing the name SUB. At execution time, the monitor system replaces the list of entry names with a corresponding list of Trap Transfer instructions whose address fields are the absolute machine locations assigned to the corresponding entries by the storage allocation logic for the particular execution. This scheme of routing all references to other entries through a single transfer vector helps minimize the relocation task of the loader.

Transfer vectors often contain entries whose names are illegal subroutine names (containing special characters) from the standpoint of usage by FORTRAN programs. These routines, requested by the compiler as needed to implement associated FORTRAN statements, are called non-FORTRAN-callable routines. They can be directly referred to, however, from FAP programs, and the reader will find a number of such references in our program set.

The program descriptions in Section 10 also use a more specialized notation; features of it are described in the following paragraphs.

A FORTRAN INTEGER, or FORTRAN-II INTEGER, or INTEGER is a fixed-point quantity with binary point assumed between bits 17 and 18, with bits 18 through 35 all zero, where the 36 bits are labeled S,1,2,. . . ,35.

A MACHINE-LANGUAGE INTEGER, abbreviated as MLI, has its binary point to the right of bit 35.

A triple-dot notation is often used to suppress symbolic subscripts in expressing lists of numbers. Thus

X(1 . . . 3) stands for (X(I), I = 1,3)

and

Y(1 . . . 3,1 . . . 2

or

Y(1,2,3,,1,2)

stands for ((Y(I,J),I = 1,3), J = 1,2)

The term VECTOR is used very commonly to refer to any singly subscripted FORTRAN variable, and its length is the highest subscript value of pertinence. A doubly subscripted variable is referred to as a MATRIX or 2-DIMENSIONAL ARRAY, and a triply subscripted variable as either a 3-DIMENSIONAL ARRAY or a MATRIX VECTOR, this last term implying that the first two subscripts define a two-dimensional array which, in the context of the computation, obeys laws of matrix algebra.

The abbreviations LSTHN, LSTHN =, GRTHN, and GRTHN = stand for the symbols $<$, $\leq$, $>$, and $\geq$ respectively.

Mathematical expressions appearing under ABSTRACT may deviate from FORTRAN conventions of naming and indexing. The emphasis here has been to produce expressions which are visually close to those of ordinary mathematics.

The numerical examples given involve some notation which should be fairly obvious. Fixed-point number lists should always be assumed to be FORTRAN-II integers unless preceded by OCT for octal or MLI for machine language integer. On the other hand, the representation of Hollerith data is not too satisfactory or consistent as given here. In most cases we use either

X(1 . . .) = 6H(something)

or

X(1 . . .) = 6Hsomething

to imply that the "something" is a string of Hollerith characters stored six to a register, that is, FORMAT(A6). However, in some cases the "something" may be split into groups of six characters separated by commas to conform to the representation of ordinary numerical lists. In deciding which is meant, the reader will have to use his judgment from the context.

There are some further notational discussions, which can be found in the introductions to Sections 4 and 10, and there is a pronunciation guide to the entry-point acronyms in Section 6.

PROGRAMMING PHILOSOPHY

The program set of Section 10 grew over a considerable period of time in a programming environment which possessed continuity of personnel and computers, coherency of computational purpose, total rapport between analysis and programming, relative freedom from a crisis atmosphere, and adequate financial support — an uncommon and fortuitous environment indeed, and one in which programming philosophy could be developed and realized. To a large extent the programs themselves are an adequate expression of such developments. This is particularly true of documentation and testing procedures, which are discussed in Section 10 as well as in the Preface, but more general design considerations may not be as self-evident.

Our most general explicit design tendencies have been to avoid writing "main" programs except when absolutely necessary, and, when it does become necessary, to pare down the functions of the main program so that it incorporates only the specializing and input-output aspects of the applied problem at hand. Thus each applied problem is subjected to analysis to determine (a) what aspects of it are expressible in terms of the existing program set, and (b) what remaining aspects might be of future value if expressed as subroutines to be added to the general collection. Everything else becomes a function of the main program, except that occasionally subroutines might be used here for certain purely technical reasons (for instance, to break down large programs into smaller blocks for reduction of compilation time during debugging.

Sometimes there are aspects of the main program's functions which for other technical reasons seem naturally to require subroutine usage (e.g., conputational patterns needed at numerous positions in the program). Such aspects are usually handled by methods internal to the main program, i.e., by arithmetic statement functions or by effective "internal subroutines" utilizing ASSIGNED or COMPUTED GO TO statements for linkage (we don't write main programs in FAP). This is done to help limit the indiscriminate growth of true subroutines and the attendant naming and documentation problems.

Computational aspects which are considered to be of future value are usually discussed by the responsible programmer with others in developing the detailed subroutine specifications. The basic choice of subroutine type has been almost invariably made between closed functions and subroutine subprograms.* Beyond this our programming group adheres to general conventions in calling-sequence design and terminology. These are detailed in Section 4. Input-output functions are not generally permitted to a subroutine unless they are its primary responsibility, in which case the external units involved are specified as arguments in the calling sequence, rather than assumed. Computational subroutines usually begin with an interlude for checking the legality of input-type arguments, and they refuse to perform badly requested computations, returning instead a diagnostic error flag as the only output. As a minimum we try to make the routines shock-proof with respect to the possibility of loss of control (loops, stops, unpredictable transfers). Special care is taken to ensure that the subroutine behaves reasonably when faced with conceptually legal but unusual, or limiting, or degenerate configurations of input arguments, so as not to create booby traps in applications broader than the specific one creating the need for the subroutine. Also, we try to see that such configurations appear in the testing programs.

But there are deeper problems of subroutine design which touch on questions not peculiar to the field of programming. By what process does one examine a complex of activities and abstract or invent useful subgroupings? With respect to the present program set we can pretty well sidestep the difficult part of this question, since the subgroupings are broadly based on corresponding and previously established ones of mathematical analysis. In particular, a program system meaningful with respect to a field of analysis would naturally tend to become a mapping of the operational structure of that field, and Volume II will expand on this topic for time-series analysis. The more difficult question still remains, however: What discriminates good program invention from bad within whatever freedom of decision prevails? Our only suggestion here is to recall the commonplace that good invention arises from the dissatisfaction of creative individuals familiar with both the cause of their irritation and the tools of the trade. The question itself is of clear importance in, say, the task of designing program-generating programs, but there is no need to pursue it in the present volume.

DESIGN FOR SPEED

We shall conclude this introductory section with a short discussion of one last consideration, namely that of computer time required. It has affected our programming strongly, since we have been dealing with many long empirical time-series and numerical filters. It has strongly biased our programming toward FAP over FORTRAN, and, in general, has decided the issues of tradeoff between speed and space in favor of the faster, if longer, programs.

In Section 3 there is a program category labeled FAST which contains a large number of entries. Study of the programs in this category will furnish details of the various programming techniques we have used to obtain speed. Volume II will provide further discussions of some of these techniques, but for the present we will only abstract some data relative to the 7094 on program speeds, in the three most important areas where our techniques have been significantly superior to elementary approaches: correlation or convolution, Fourier transformation, and solution of Toeplitz matrix equations.

---

*In retrospect, our tendency to avoid writing FORTRAN functions, based on no better reason than the fact that one can occasionally confuse them with subscripted variables, appears somewhat unfortunate, since we thereby denied ourselves a certain degree of flexibility. (Note, for example, that one may call a FORTRAN function as an ordinary subroutine subprogram in addition to using it as a numerical entity in an arithmetic statement.

Our key high-speed correlation or convolution program is PROCOR, whose writeup gives a reasonable idea of the techniques employed. Since PROCOR involves basically fixed-point arithmetic, a number of higher-level routines based on PROCOR have been written for ordinary correlation and convolution floating-point applications. They are QACORR, QXCORR, QXCOR1, and QCNVLV. Timing data for the autocorrelation program QACORR are adequate for illustration here. This program achieves nothing more complicated than does the following FORTRAN subroutine.

```
      SUBROUTINE FORAC(X, LX, MXLAG, ACOR)
      DIMENSION X(2), ACOR(2)
      JMAX    = MXLAG + 1
      DO 20     J = 1,JMAX
      SUM     = 0.0
      NMAX    = LX - J + 1
      DO 10     I = 1,NMAX
      K       = J + 1
10    SUM     = SUM + X(I)*X(K-1)
20    ACOR(J) = SUM
      RETURN
      END
```

Fig. 1 gives timing information on QACORR and on the above program for data lengths varying from about 20 to 10,000, showing that QACORR is inferior for very short data but possesses a speed advantage factor of ten or greater for the longer series (this factor jumps to around 17 for the 709 or 7090). Similar savings will be realized in cross correlation and convolution by the other programs using PROCOR.

The high-speed harmonic transform programs are based primarily on subroutine COSP. They are ASPECT, COSIS1, QFURRY, QIFURY, and XSPECT. (Subroutine FACTOR also uses COSP in finding minimum-phase transients from energy-density spectra.) The speed of COSP comes from careful looping logic on stored sinusoids. Speeds of ASPECT, which finds cosine transforms of symmetrical data (usually autocorrelations in our applications) and which uses folding and splitting logic in addition to using COSP, can be 10 to 100 times faster than those of elementary programs. Speed-run results for ASPECT are shown in Fig. 2 for various data lengths and frequency increments. The upswing of the curves in the lower portion of the figure results from the gradually dominating influence of the folding and splitting logic for long data.

Toeplitz matrices arise in many time-series problems, particularly in the determination of least-squares filters. These matrices are positive definite Hermitian with elements constant along any diagonal, so that if the matrix is n by n, there are only n independent elements rather than n^2. Recursion techniques exist for the solution of simultaneous equations involving these matrices, which require computational times proportional to n^2 rather than n^3 .* Subroutine WLLSFP or the coordinated pair RLSPR and RLSSR handles normal Toeplitz matrix equation problems. RLSPR2, FIRE2, MIPLS, MIFLS, and MISS are for use in multidimensional or multi-input problems when, for example, the matrix elements themselves become matrices. It is instructive to make a comparison between the computation times of ordinary simultaneous-equation programs not involving the special Toeplitz assumption and those of recursive programs. Such a comparison is made in Fig. 3, showing empirical times of WLLSFP and of the general utility subroutine SIMEQ. Analogous curves for the 7090 will show very similar relative behavior.

---

*An early reference is Levinson's Appendix to Interpolation, Extrapolation, and Prediction of Stationary Time Series, by N. Wiener, John Wiley & Sons.

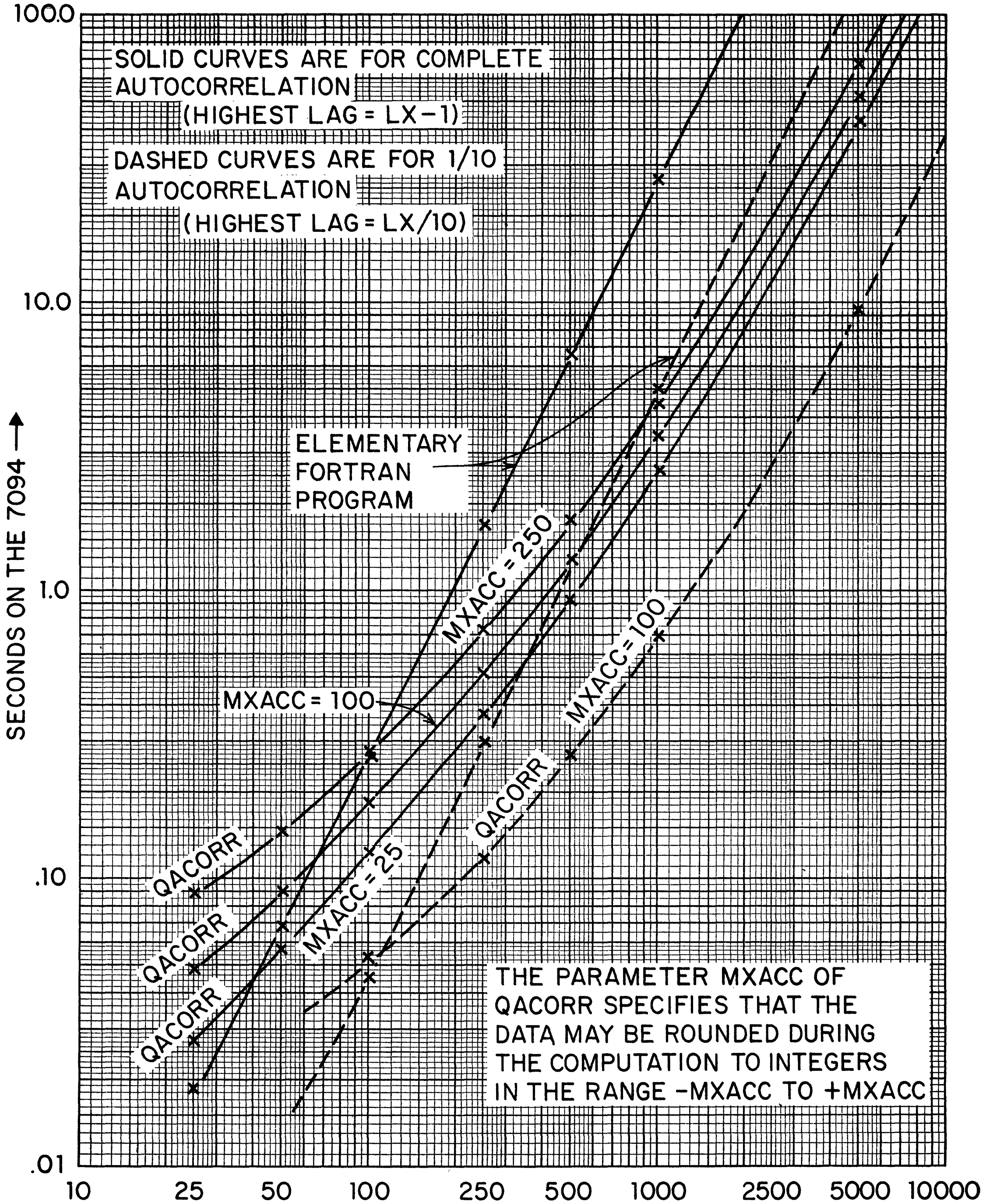


FIGURE 1. EMPIRICAL TIME CURVES FOR SUBROUTINE QACØRR AND FOR AN ELEMENTARY FORTRAN PROGRAM IN COMPUTING AUTO-CORRELATION FUNCTIONS.

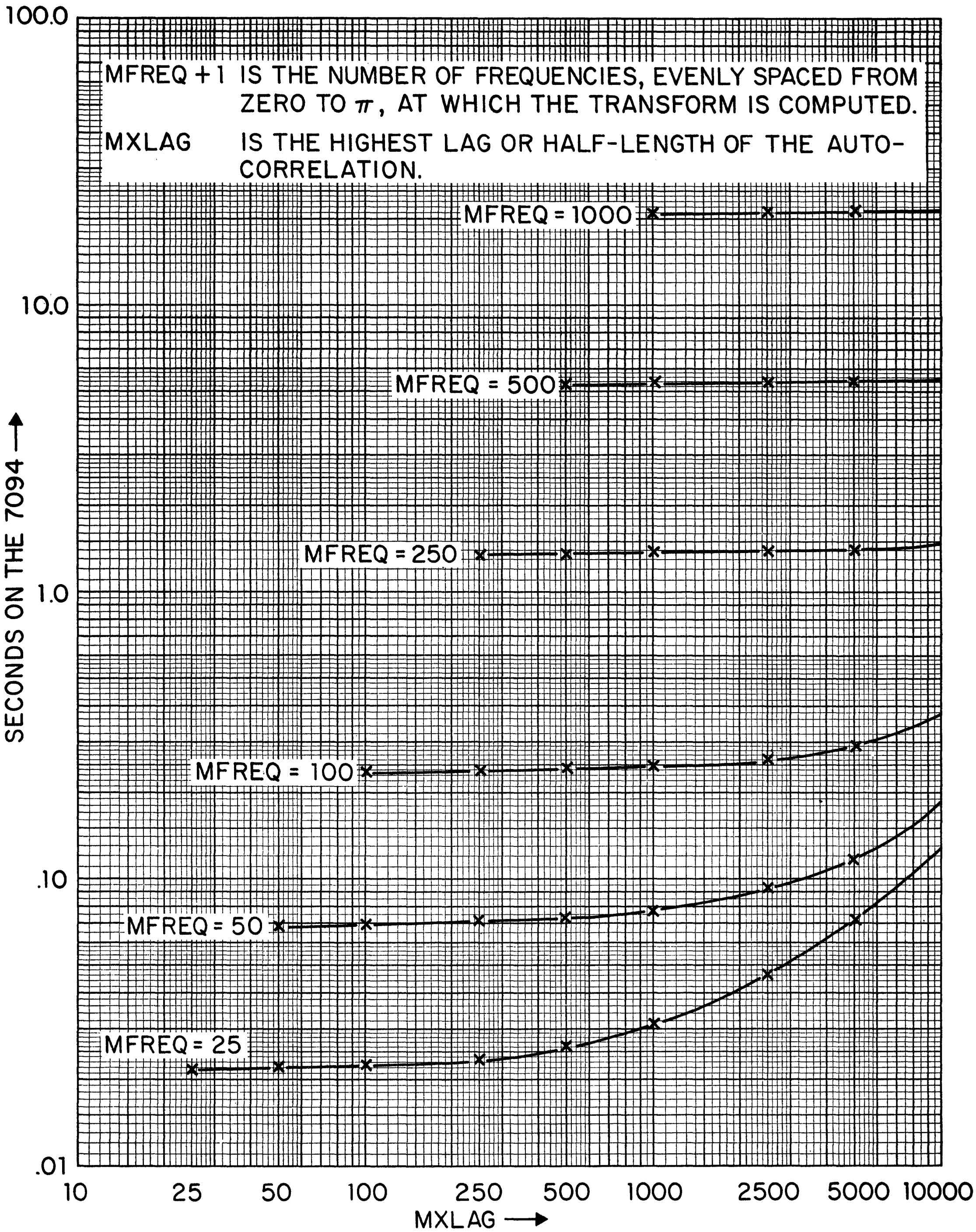


FIGURE 2. EMPIRICAL TIME CURVES FOR SUBROUTINE ASPECT IN COMPUTING COSINE TRANSFORMS OF AUTOCORRELATIONS OVER THE FULL FREQUENCY RANGE.

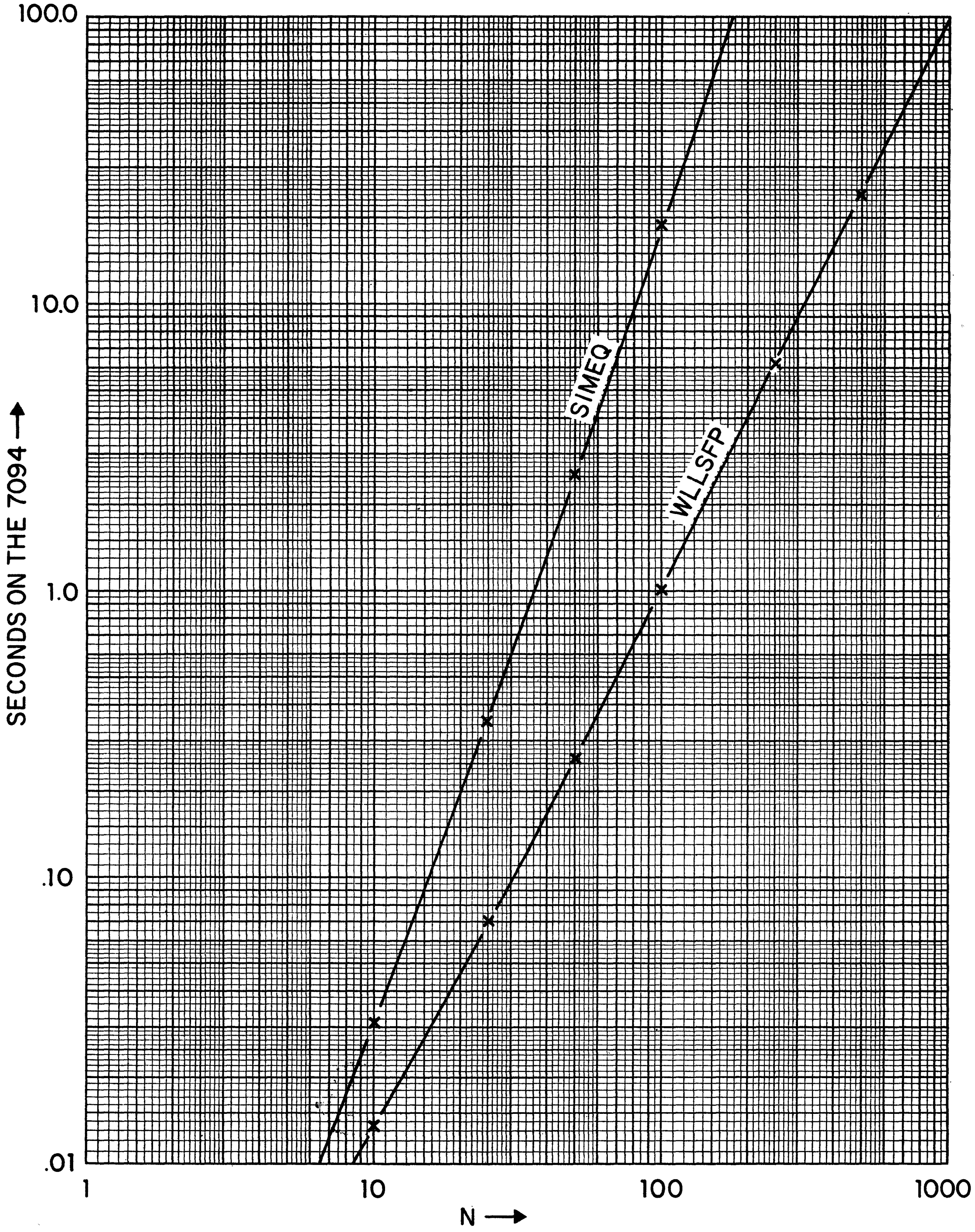


FIGURE 3. EMPIRICAL TIME CURVES FOR SUBROUTINES WLLSFP AND SIMEQ IN SOLVING THE MATRIX EQUATION AX = B FOR THE VECTOR X, WHERE A IS SQUARE TØEPLITZ OF DIMENSION N BY N.

REFERENCES

As mentioned earlier, the motivation behind the development of this program set was time-series applications. From 1952 to 1957 these applications concerned seismic exploration for oil and were pursued by the M.I.T. Geophysical Analysis Group of the M.I.T. Department of Geology and Geophysics. From 1960 to 1965 the work was continued by an M.I.T. VELA UNIFORM group of the same department on the problem of underground nuclear detection. The research reports of these two groups contain background material for the present volume and are available for reference.

The M.I.T. Geophysical Analysis Group Reports are on file at the M.I.T. Library, and also at VESIAC, the VELA Seismic Information Analysis Center at the Institute of Science and Technology of the University of Michigan. They are:

Robinson, E.A., and G.P. Wadsworth, *A prospectus on the applications of linear operators to seismology*, M.I.T. Geophysical Analysis Group Report 1, Cambridge, Mass., and VESIAC 1108, 1952, 61 pp.

Robinson, E.A., and G.P. Wadsworth, *Results of an autocorrelation and cross-correlation analysis of seismic records*, M.I.T., Geophysical Analysis Group Report 2, Cambridge, Mass., 1952, 40 pp.

Robinson, E.A., S.M. Simpson, Jr., M.K. Smith, and W.P. Walsh, *Case study of Henderson County seismic record*, M.I.T. Geophysical Analysis Group Report 3, Cambridge, Mass., and VESIAC 1100, 1953, 48 pp.

Robinson, E.A., S.M. Simpson, Jr., and M.K. Smith, *Linear operator study of a Texas Company seismic profile*, M.I.T. Geophysical Analysis Group Report 4, Cambridge, Mass., and VESIAC 1101, 1953, 181 pp.

Robinson, E.A., M.K. Smith, and S.M. Simpson, Jr., *On the theory and practice of linear operators in seismic analysis*, M.I.T. Geophysical Analysis Group Report 5, Cambridge, Mass., and VESIAC 1102, 1953, 95 pp.

Robinson, E.A., M.K. Smith, S.M. Simpson, Jr., D.E. Bowker, R. Bowman, H.W. Briscoe, J.F. Gilbert, S. Treitel, M.S. Turyn, and W.P. Walsh, *Further research on linear operators in seismic analysis*, M.I.T. Geophysical Analysis Group Report 6, Cambridge, Mass., and VESIAC 1103, 1954, 203 pp.

Robinson, E.A., *Predictive decomposition of time series with applications to seismic exploration*, Ph.D. Thesis, M.I.T., and M.I.T. Geophysical Analysis Group Report 7, Cambridge, Mass., and VESIAC 1104, 1954, 265 pp.

Simpson, S.M., Jr., *A multiple trace criterion for linear operator selection*, M.I.T. Geophysical Analysis Group Report 8, Cambridge, Mass., 1954, 30 pp.

Simpson, S.M., Jr., R. Bowman, D.R. Fink, J.F. Gilbert, D.R. Grine, M. Lopez-Linares, R.D. Tooley, S. Treitel, and R.W. Wylie, *Linear operators and seismic noise*, M.I.T. Geophysical Analysis Group Report 9, Cambridge, Mass., 1955, 281 pp.

Simpson, S.M., Jr., R. Bowman, D.R. Fink, D.R. Grine, M. Lopez-Linares, H. Posen, R.D. Tooley, S. Treitel, and R.W. Wylie, *Properties, origin, and treatment of certain types of seismic noise*, M.I.T. Geophysical Analysis Group Report 10, Cambridge, Mass., 1956, 211 pp.

Simpson, S.M., Jr., J.F. Gilbert, D.R. Grine, R.L. Sax, S. Treitel, and R.W. Wylie, The interrelation of the deterministic and probabilistic approaches to seismic problems, M.I.T. Geophysical Analysis Group Report 11, Cambridge, Mass., 1957, 136 pp.

The VELA UNIFORM reports [of Contract No. AF19-(604)-7378] are on file at the M.I.T. Library (where they are filed by their AFCRL numbers) and at VESIAC. They are also distributed by government organizations. Requests from government agencies and from Department of Defense contractors are handled by the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314. Others can apply to

Clearinghouse for Federal Scientific and
Technical Information (CFSTI)
Sills Building
5285 Port Royal Road
Springfield, Virginia 22151.

These reports (all unclassified) are:

Simpson, S.M., Jr., J. Claerbout, and J.N. Galbraith, Jr., Initial studies on underground nuclear detection with seismic data prepared by a novel digitization system, Report No. 1 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 61-863, 1961, 450 pp.

Robinson, E.A., S.M. Simpson, Jr., J. Claerbout, J.N. Galbraith, Jr., J. Clark, and W. Ross, Time series techniques applied to underground nuclear detection and further digitized seismic data, Report No. 2 of AF19(604)-7378, M.I.T. Cambridge, Mass., and AFCRL 62-262, December, 1961, 500 pp.

Robinson, E.A., S.M. Simpson, Jr., J. Claerbout, J.N. Galbraith, Jr., R.A. Wiggins, C. Pan, and J. Clark, Continued numerical studies on underground nuclear detection and further digitized seismic data, Report No. 3 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 62-879, June, 1962, 363 pp.

Simpson, S.M., Jr., Magnetic tape copies of M.I.T. Geophysics Program Set I, Report No. 4 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 63-282, December, 1962, 47 pp.

Claerbout, J.F., Digital filters and applications to seismic detection and discrimination, M.S. Thesis, M.I.T., and Report No. 5 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 63-604, February, 1963, 89 pp.

Galbraith, J.N., Jr., Computer studies of microseism statistics with applications to prediction and detection, Ph.D. Thesis, M.I.T., and Report No. 6 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 63-673, May, 1963, 283 pp.

Simpson, S.M., Jr., E.A. Robinson, R.A. Wiggins, and C.I. Wunsch, Studies in optimum filtering of single and multiple stochastic processes, Report No. 7 of AF19(604) 7378, M.I.T., Cambridge, Mass., and AFCRL 64-241, June, 1963, 140 pp.

Robinson, E.A., Seismic arrays for the detection of nuclear explosions, Report No. 8 of AF19(604)-7378, M.I.T., Cambridge, Mass., and AFCRL 64-855, June, 1964, 107 pp.

Wiggins, RA., On factoring the correlations of discrete multivariable stochastic processes, Ph. D. Thesis, M.I.T., and Report No. 9 of AF19(604) - 7378, M.I.T., Cambridge, Mass., and AFCRL 65-207, February, 1965, 196 pp.

Simpson, S.M., Jr., Magnetic tape copies of M.I.T. Geophysics Program Set II, Report No. 10 of AF19(604) - 7378, M.I.T., Cambridge, Mass., and AFCRL 65-306, March, 1965, 79 pp.

Simpson, S.M., Jr., R.W. Wiggins, and C. Pan, Sampling events from U.S.C. & G.S. Earthquake Cards, Report No. 11 (Final Report) of AF19(604) - 7378, M.I.T., Cambridge, Mass., and AFCRL 65 - 463, June, 1965, 70 pp.

2
Illustrative Usage of Programs

Examples of the use of the program as an isolated entity appear with each program listed in Section 10. Such examples are valuable, but often do not project a sense of the use of the program in an applied setting. Volume II of the present writing will give numerous illustrations of such usage in the time-series setting for which the program collection was developed. It is in keeping with the tenor of the present volume to present examples of usage in an applied but utilitarian setting. Such a setting is meaningful, because a good many of the programs of the collection fall under a utility classification in no way specialized to the field of time series.

The illustrations in the first set given are quite simple to scan and digest. A large number of the utility programs have truly elementary functions which are easily expressed by a few basic FORTRAN statements, the raison d' être of such programs being convenience, or speed, or both. The illustration for these programs is a sequence of isolated program usages paired with equivalent, basic FORTRAN sequences. In this fashion a large number of programs can be covered in a few pages. The selections here include all of the minor utility programs which have simple FORTRAN translations. The reader should be cautioned that the basic FORTRAN equivalents may not be exact in all variations of the sample usage, especially in cases of zero or negative-length vectors, or in cases where arguments in calling sequences are equated by FORTRAN equivalence statements not shown here. 2

The illustrations in the second set given in this section are simply listings of some of the testing programs we have used to verify the input-output behavior asserted in the program writeups of Section 10. In these test programs we have leaned heavily on the use of utility programs previously verified. The test programs must be studied with close reference to the program writeups of Section 10, since the test decks have no independent documentation. An examination of these listings will also bring out the general style we have evolved for writing test programs: this style may be of interest to persons with similar problems.

The third and last set of illustrations consists of three main programs which produced the timing data for Figs. 1, 2, and 3 of Section 1.

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN
CALL ABSVAL(X,I1,I2,Y,IANS) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=I1,I2 J=I-I1+1 10 Y(J)=ABSF(X(I))
CALL ADDK(C,X1,X2,...,XN) (XADDK FOR FIXED POINT)	X1=X1+C X2=X2+C (ETC) XN=XN+C
CALL ADDKS(C1,X1,Y1, C2,X2,Y2, 1 ..., CN,XN,YN) (XADDKS FOR FIXED POINT)	Y1=X1+C1 Y2=X2+C2 (ETC) YN=XN+CN
CALL AVRAGE(X,LX,XAVG) (XAVRGE OR XAVRGR FOR FIXED POINT)	SUM=0.0 DO 10 I=1,LX 10 SUM=SUM+X(I) XAVG=SUM/FLOATF(LX)
CALL BOOST(X,LX,C,Y) (XBOOST FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=X(I)+C
CALL CARIGE(ITAPE,NSPACE)	IF (NSPACE) 40,60,10 10 DO 20 I=1,NSPACE 20 WRITE OUTPUT TAPE ITAPE, 30 30 FORMAT(1H) GO TO 60 40 WRITE OUTPUT TAPE ITAPE,50 50 FORMAT(1H1) 60 CONTINUE
CALL CHOOSE(ZIFRST, X,X1,X2, ..., 1 Z,Z1,Z2)	IF (ZIFRST) 20,10,20 10 X=X1 (ETC) Z=Z1 GO TO 30 20 X=X2 (ETC) Z=Z2 30 CONTINUE
CALL CHSIGN(X,LX,Y) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=-X(I)
IF (CHUSETF(X,X1,X2,ZIFX1)) 1 40,50,60 (SAME PROGRAM FOR FIXED POINT)	IF (ZIFX1) 20,10,20 10 X=X1 GO TO 30 20 X=X2 30 IF (ZIFX1) 40,50,60
X=DELTAF(Y) (ARGUMENT MODE IMMATERIAL) (XDELTA FOR FIXED POINT)	IF (Y) 20,10,20 10 X=1.0 GO TO 30 20 X=0.0 30 CONTINUE

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN
CALL DIVIDE(X,LX,D,Y) (XDVIDE OR XDVIDR FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=X(I)/D
CALL DIVK(C,X1,...,XN) (XDIVK OR XDVRK FOR FIXED POINT)	X1=X1/C (ETC) XN=XN/C
CALL DIVKS(C1,X1,Y1,...,CN,XN,YN) (XDIVKS OR XDVRKS FOR FIXED POINT)	Y1=X1/C1 (ETC) YN=XN/CN
CALL DPRESS(X,LX,C,Y) (XDPRESS FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=X(I)-C
CALL DUBLL(X,LX) (DUBLX FOR FIXED POINT)	DO 10 I=1,LX 10 X(I)=2.0*X(I)
CALL EXCHVS(LXY,X,Y) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=1,LXY TEMP=X(I) X(I)=Y(I) 10 Y(I)=TEMP
CALL FDOT(LXY,X,Y,DOT)	DOT=0.0 DO 10 I=1,LXY 10 DOT=DOT+X(I)*Y(I)
CALL FDOTR(LXY,X,Y,DOTR)	DOTR=0.0 DO 10 I=1,LXY J=LXY+1-I 10 DOTR=DOTR+X(I)*Y(J)
CALL FIXV(X,LX,IX)	DO 10 I=1,LX 10 IX(I)=XFIXF(X(I))
CALL FIXVR(X,LX,IX)	DO 10 I=1,LX 10 IX(I)=XFIXF(X(I)+.5)
CALL FMTOUT(ITAPE,10H8H MESSAGE)	WRITE OUTPUT TAPE ITAPE,10 10 FORMAT(8H MESSAGE)
Y=GETX(X,IA,...,IX,IY,IZ) (IGETX FOR FIXED POINT)	I=IY(IZ) I=IX(I) (ETC) I=IA(I) Y=X(I)
IF (INDEXF(I,ICRTCL)) 10,20,30	I=I+1 IF (I-ICRTCL) 10,20,30
CALL HALVL(X,LX) (HALVX FOR FIXED POINT)	DO 10 I=1,LX 10 X(I)=X(I)/2.0

```
I-------------------------------------I---------------------------------------I
I                                     I                                       I
I            PROGRAM USAGE            I        EQUIVALENT BASIC FORTRAN       I
I                                     I                                       I
I-------------------------------------I---------------------------------------I
I  CALL INTSUM(X,LX,Y)                I      Y(1)=X(1)                        I
I                                     I      IF (LX-1)  30,30,10              I
I                                     I 10 DO 20  I=2,LX                      I
I                                     I 20 Y(I)=Y(I-1)+X(I)                   I
I  (XNTSUM FOR FIXED POINT)           I 30 CONTINUE                           I
I-------------------------------------I---------------------------------------I
I  CALL IXCARG(X,IX)                  I      COMMON  C                        I
I                                     I      IX=XLOCF(C)-XLOCF(X)+1           I
I-------------------------------------I---------------------------------------I
I  CALL LOC(X(I),ILOC)                I      ILOC=XLOCF(X)-I+1                I
I-------------------------------------I---------------------------------------I
I  CALL MOVE(LX,X,Y)                  I      IF (XLOCF(X)-XLOCF(Y))  10,50,30 I
I                                     I 10 DO 20  I=1,LX                      I
I                                     I 20 Y(I)=X(I)                          I
I                                     I      GO TO 50                         I
I                                     I 30 DO 40  I=1,LX                      I
I                                     I      J=LX+1-I                         I
I                                     I 40 Y(J)=X(J)                          I
I                                     I 50 CONTINUE                           I
I-------------------------------------I---------------------------------------I
I  CALL MULK(C,X1,...,XN)             I      X1=X1*C                          I
I                                     I        (ETC)                          I
I  (XMULK FOR FIXED POINT)            I      XN=XN*C                          I
I-------------------------------------I---------------------------------------I
I  CALL MULKS(C1,X1,Y1,...,CN,XN,YN)  I      Y1=C1*X1                         I
I                                     I        (ETC)                          I
I  (XMULKS FOR FIXED POINT)           I      YN=CN*XN                         I
I-------------------------------------I---------------------------------------I
I  CALL MULPLY(X,LX,C,Y)              I      DO 10  I=1,LX                    I
I  (XMLPLY FOR FIXED POINT)           I 10 Y(I)=C*X(I)                        I
I-------------------------------------I---------------------------------------I
I  A=NTHAF(J,A1,...,AN)               I      TEMP(1)=A1                       I
I                                     I        (ETC)                          I
I                                     I      TEMP(N)=AN                       I
I  (XNTHA FOR FIXED POINT)            I      A=TEMP(J)                        I
I-------------------------------------I---------------------------------------I
I  CALL PLURNS(A1,...,AN, ...,        I      CALL SUB(A1,...,AN)              I
I 1             Z1,...,ZN)            I        (ETC)                          I
I  CALL SUB(N)                        I      CALL SUB(Z1,...,ZN)              I
I-------------------------------------I---------------------------------------I
I  CALL REFLEC(X,LX,C,Y)              I      DO 10  I=1,LX                    I
I  (XRFLEC FOR FIXED POINT)           I 10 Y(I)=C-X(I)                        I
I-------------------------------------I---------------------------------------I
I  CALL REVER(X,LX,Y)                 I      IF (XLOCF(X)-XLOCF(Y))  10,20,10 I
I                                     I 10 N=LX                               I
I                                     I      GO TO 30                         I
I                                     I 20 N=(LX+1)/2                         I
I                                     I 30 DO 40  I=1,N                       I
I                                     I      J=LX+1-I                         I
I                                     I      TEMP=X(J)                        I
I                                     I      Y(J)=X(I)                        I
I  (SAME PROGRAM FOR FIXED POINT)     I 40 Y(I)=TEMP                          I
I-------------------------------------I---------------------------------------I
```

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN

`CALL REVERS(LX,X)`
 	
`(SAME PROGRAM FOR FIXED POINT)`
	`   N=(LX+1)/2`
	`   DO 10  I=1,N`
	`   J=LX+1-I`
	`   TEMP=X(J)`
	`   X(J)=X(I)`
	`10 X(I)=TEMP`

`Y=RNDF(X)`	`   Y=FLOATF(XFIXF(X+.5))`
`Y=RNDDNF(X)`	`   Y=FLOATF(XFIXF(X))`

`Y=RNDUPF(X)`	`   IF (X)  20,10,10`
	`10 Y=FLOATF(XFIXF(X+.99999999))`
	`   GO TO 30`
	`20 Y=FLOATF(XFIXF(X-.99999999))`
	`30 CONTINUE`

`CALL RNDV(X,LX,Y)`	`   DO 10  I=1,LX` `10 Y(I)=FLOATF(XFIXF(X(I)+.5))`
`CALL RNDVDN(X,LX,Y)`	`   DO 10  I=1,LX` `10 Y(I)=FLOATF(XFIXF(X(I))`

`CALL RNDVUP(X,LX,Y)`	`   DO 30  I=1,LX`
	`   IF (X(I))  20,10,10`
	`10 Y(I)=FLOATF(XFIXF(X(I)+.99999999))`
	`   GO TO 30`
	`20 Y(I)=FLOATF(XFIXF(X(I)-.99999999))`
	`30 CONTINUE`

`Y=SAMEF(IX)` `(XSAME FOR FIXED POINT)`	`   EQUIVALENCE  (Y,IY)` `   IY=IX`
`IF (SETAPTF(X,XNEW,FVALUE))` `1                     10,20,30`	`   X=XNEW` `   IF (FVALUE)  10,20,30`
`IF (SETESTF(X,XNEW,XCRTCL))` `1                     10,20,30`	`   X=XNEW` `   IF (XNEW-XCRTCL)  10,20,30`

`CALL SETK(C,X1,X2,...,XN)`
 	
`(SAME PROGRAM FOR FIXED POINT)`
	`   X1=C`
	`   X2=C`
	`     (ETC)`
	`   XN=C`

`B      P=777777712345`
`       CALL SETKP(C1,A,B,P,`
`     1            C2,D,E,F,G,P,`
`     2            C3,H)`
 	
`(SAME PROGRAM FOR FIXED POINT`
` OR MIXED MODES)`
	`   A=C1`
	`   B=C1`
	`   D=C2`
	`   E=C2`
	`   F=C2`
	`   G=C2`
	`   H=C3`

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN
CALL SETKS(C1,A,C2,B,...,CN,Z) (SAME PROGRAM FOR FIXED POINT OR MIXED MODES)	A=C1 B=C2 (ETC) Z=CN
CALL SETKV(C,LX,X) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=1,LX 10 X(I)=C
CALL SETKVS(C1,LX1,X1, ..., 1 CN,LXN,XN) (SAME PROGRAM FOR FIXED POINT OR MIXED MODES)	DO 10 I=1,LX1 10 X1(I)=C1 (ETC) DO 90 I=1,LXN 90 XN(I)=CN
CALL SETLIN(B,D,LX,X) (XSTLIN FOR FIXED POINT)	DO 10 I=1,LX 10 X(I)=B+D*FLOATF(I-1)
CALL SETLNS(B1,D1,LX1,X1, ..., 1 BN,DN,LXN,XN) (SAME PROGRAM FOR FIXED POINT OR MIXED MODES)	DO 10 I=1,LX1 10 X1(I)=B1+D1*FLOATF(I-1) (ETC) DO 90 I=1,LXN 90 XN(I)=BN+DN*FLOATF(I-1)
B P=777777712345 CALL SETVCP(X,C1,...,CL,P, 1 Y,D1,...,DM,P, 2 ..., 3 Z,G1,...,GN) (SAME PROGRAM FOR FIXED POINT OR MIXED MODES)	X(1)=C1 (ETC) X(L)=CL Y(1)=D1 (ETC) Y(M)=DM (ETC) Z(1)=G1 (ETC) Z(N)=GN
CALL SETVEC(X,C1,...,CN) (SAME PROGRAM FOR FIXED POINT)	X(1)=C1 (ETC) X(N)=CN
CALL SIFT(X,M,LY,Y) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=1,LY J=1+(I-1)*M 10 Y(I)=X(J)
CALL SQRDEV(X,C,LX,SSQ) (XSQDEV FOR FIXED POINT)	SSQ=0.0 DO 10 I=1,LX 10 SSQ=SSQ+(X(I)-C)*(X(I)-C)
CALL SQRDFR(X,Y,LXY,SSQ) (XSQDFR FOR FIXED POINT)	SSQ=0.0 DO 10 I=1,LXY 10 SSQ=SSQ+(X(I)-Y(I))**2
CALL SQROOT(X,LX,Y) (XSQRUT FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=SQRTF(X(I))

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN
CALL SQRSUM(X,LX,SSQ) (XSQSUM FOR FIXED POINT)	SSQ=0.0 DO 10 I=1,LX 10 SSQ=SSQ+X(I)*X(I)
CALL SQUARE(X,LX,Y) (XSQUAR FOR FIXED POINT)	DO 10 I=1,LX 10 Y(I)=X(I)*X(I)
Y=STEPCF(X) (ARGUMENT MODE IMMATERIAL) (XSTEPC FOR FIXED POINT OUTPUT)	Y=.5+SIGNF(.5,X)
Y=STEPLF(X) (ARGUMENT MODE IMMATERIAL) (XSTEPL FOR FIXED POINT OUTPUT)	IF (X) 20,10,10 10 Y=1.0 GO TO 30 20 Y=0.0 30 CONTINUE
Y=XSTEPRF(X) (ARGUMENT MODE IMMATERIAL) (XSTEPR FOR FIXED POINT OUTPUT)	IF (X) 20,20,10 10 Y=1.0 GO TO 30 20 Y=0.0 30 CONTINUE
CALL STZ(LX,X) (SAME PROGRAM FOR FIXED POINT)	DO 10 I=1,LX 10 X(I)=0.0
CALL STZS(LX1,X1,...,LXN,XN) (SAME PROGRAM FOR FIXED POINT OR MIXED MODES)	DO 10 I=1,LX1 10 X1(I)=0.0 (ETC) DO 90 I=1,LXN 90 XN(I)=0.0
CALL SUBK(C,X1,...,XN) (XSUBK FOR FIXED POINT)	X1=X1-C (ETC) XN=XN-C
CALL SUBKS(C1,X1,Y1,...,CN,XN,YN) (XSUBKS FOR FIXED POINT)	Y1=X1-C1 (ETC) YN=XN-CN
CALL SUM(X,LX,SUM) (XSUM FOR FIXED POINT)	SUM=0.0 DO 10 I=1,LX 10 SUM=SUM+X(I)
CALL SUMDEV(X,B,LX,SUMD) (XSMDEV FOR FIXED POINT)	SUMD=0.0 DO 10 I=1,LX 10 SUMD=SUMD+X(I)-B
CALL SUMDFR(X,Y,LXY,SUMD) (XSMDFR FOR FIXED POINT)	SUMD=0.0 DO 10 I=1,LXY 10 SUMD=SUMD+X(I)-Y(I)

PROGRAM USAGE	EQUIVALENT BASIC FORTRAN
IF (SWITCHF(ISENSE)) 10,10,20	IF (ISENSE) 10,10,30 30 IF (ISENSE-6) 40,40,10 40 GO TO (1,2,3,4,5,6), ISENSE 1 IF (SENSE SWITCH 1) 20,10 2 IF (SENSE SWITCH 2) 20,10 (ETC) 6 IF (SENSE SWITCH 6) 20,10
CALL VDOTV(X,Y,LXY,DIV,DOT)	DOT=0.0 DO 10 I=1,LXY 10 DOT=DOT+X(I)*Y(I) DOT=DOT/DIV
CALL VDVBYV(X,Y,LXY,Z) (XVDRBV OR XVDVBV FOR FIXED POINT)	DO 10 I=1,LXY 10 Z(I)=X(I)/Y(I)
CALL VECOUT(ITAPE,8H6H10F7.1, 1 X,I1,I2)	WRITE OUTPUT TAPE ITAPE,10, 1 (X(I),I=I1,I2) 10 FORMAT(10F7.1)
IF (VINDEXF(I,IC,IJ)) 10,20,30	I=I+IJ IF (I-IC) 10,20,30
CALL VMNUSV(X,Y,LXY,Z) (XVMNSV FOR FIXED POINT)	DO 10 I=1,LXY 10 Z(I)=X(I)-Y(I)
CALL VPLUSV(X,Y,LXY,Z) (XVPLSV FOR FIXED POINT)	DO 10 I=1,LXY 10 Z(I)=X(I)+Y(I)
CALL VTIMSV(X,Y,LXY,Z) (XVTMSV FOR FIXED POINT)	DO 10 I=1,LXY 10 Z(I)=X(I)*Y(I)
X=WHICHF(X1,X2,Y) (XWHICH FOR FIXED POINT)	IF (Y) 20,10,20 10 X=X1 GO TO 30 20 X=X2 30 CONTINUE
IF (XACTEQF(X,Y)) 10,20,30 (SAME PROGRAM FOR FIXED POINT ARGUMENTS)	IF (X-Y) 10,40,30 40 IF (X) 20,50,20 50 IF (SIGNF(1.,X)-SIGNF(1.,Y)) 1 10,20,30
IF (XLIMITF(X,XA,XB)) 10,20,30 (SAME PROGRAM FOR FIXED POINT ARGUMENTS)	IF (X-MAX1F(XA,XB)) 40,20,30 40 IF (X-MIN1F(XA,XB)) 10,20,20
CALL XLOCV(LOCV,X1,...,XN)	LOCV(1)=XLOCF(X1) (ETC) LOCV(N)=XLOCF(XN)
IF (XODZEF(I)) ---,10,20	IF (I-2*(I/2)) 20,10,20

SAMPLE TESTING PROGRAMS

```
*      TEST BLKSUM
*      XEQ
*      LIST8
*      LABEL
CTLKSUM
       DIMENSION X(9), S(4,4,4), LS(4,4), SPACE(10)
       ITEST=0
 7     ITEST=ITEST+1
       CALL VRSOUT(2,3,14H9H EXAMPLE ,I1,ITEST,ITEST)
       CALL SETVEC(X,2.,4.,6.,8.)
       CALL SETKVS(-9.,64,S, -9,16,LS, 2.0,1,DVSR)
       GO TO (1,2,3),ITEST
 1     DO 10  LX=1,4
       DO 10  L=1,LX
 10    CALL BLKSUM(X,LX,L,DVSR,S(1,L,LX),LS(L,LX))
       CALL VSOUT(2,3,S(1,1,1),6HS14141,5H4F7.1,1,16,
      1  S(1,1,2),6HS14142,5H4F7.1,1,16,  S(1,1,3),6HS14143,
      2  5H4F7.1,1,16, S(1,1,4),6HS14144,5H4F7.1,1,16,
      3  LS(1,1),6HLS1414,3H4I7,1,16)
       GO TO 7
 2     CALL BLKSUM(X,4,2,DVSR,X,LS)
       CALL VSOUT(2,3,X,1HX,5H4F7.1,1,4, LS,2HLS,2HI7,1,1)
       GO TO 7
 3     CALL BLKSUM(X,-1,2,1.0,S,LS)
       CALL BLKSUM(X, 3,0,1.0,S,LS)
       CALL BLKSUM(X, 3,4,1.0,S,LS)
       CALL BLKSUM(X, 3,2,0.0,S,LS)
       CALL VRSOUT(2,3,18H8H S,LS = ,F7.1,I7,SPACE,S,LS)
       CALL EXIT
       END
```

```
*      TEST CMPRA
*      XEQ
*      LIST8
*      LABEL
CTCMPRA
       GO TO 999
 10    CONTINUE
       Z = CMPRAF(X,Y)
       IZ=XCMPRAF(X,Y)
       FZ=CMPRFLF(X,Y)
       WRITE OUTPUT TAPE 2,20,J,X,Y,Z,X,Y,IZ,X,Y,FZ
 20    FORMAT(1H0I2,23H. ACOMP TEST -  ACOMPF(  O15,1H,  O15,4H) = G15.8/
      118X8HXACOMPF(G15.8,1H,G15.8,4H) = G15.8/18X8HFLCOMPF(G15.8,1H,
      2G15.8,4H) = G15.8)
 999   J=J+1
       GO TO (1,2,3,4,5,6,9999),J
```

(CONTINUED NEXT PAGE)

SAMPLE TESTING PROGRAMS

```
 1     CALL SETKS (1,X,1,Y)
       GO TO 10
 2     CALL SETKS (1,X,-1,Y)
       GO TO 10
 3     CALL SETKS (1.2345678,X,1.2345679,Y)
       GO TO 10
 4     CALL SETKS (6HABCDE1,X,6HABCDE2,Y)
       GO TO 10
 5     CALL SETKS (0,X,-0,Y)
       GO TO 10
 6     CALL SETKS (-50.,X,-51.,Y)
       GO TO 10
 9999  CALL EXIT
       END
```

```
*      TEST CRSVM
*      XEQ
*      LIST8
*      LABEL
CTCRSVM
       DIMENSION AA(1000),BB(1000),CC(1000),SPACE(1000)
       COMMON AA,BB,CC,SPACE
 10    J=J+1
       CALL VRSOUT (2,-1,20H1XI2,12H. CRSVM TEST,J,J)
       CALL RDATA  (4,2,IANS,SPACE,4HNRAC,NRAC,5HNCARB,NCARB,4HNCBC,NCBC,
      1 3HLAA,LAA,2HAA,AA,3HLBB,LBB,2HBB,BB,6HZFNBTR,ZFNBTR,6HIFSTLG,
      2 IFSTLG,3HLCC,LCC)
       CALL CSOUT (2,1,NRAC,4HNRAC,NCARB,5HNCARB,NCBC,4HNCBC,LAA,3HLAA,
      1 LBB,3HLBB,ZFNBTR,6HZFNBTR,IFSTLG,6HIFSTLG,LCC,3HLCC)
       CALL MOUT  (2,1,AA,2HAA,NRAC,NCARB,LAA)
       CALL MOUT  (2,1,BB,2HBB,NCARB,NCBC,LBB)
       CALL CRSVM (NRAC,NCARB,NCBC,LAA,AA,LBB,BB,ZFNBTR,IFSTLG,LCC,CC)
       CALL MOUT  (2,3,CC,2HCC,NRAC,NCBC,LCC)
       GO TO 10
       END
*      DATA
 NRAC=1 NCARB=2 NCBC=3 LAA=4 AA=1.,2.,3.,-2.,5.,-4.,1.,-1.
 LBB=2 BB=3.,2.,4.,3.,1.,-1.,-2.,-3.,-2.,2.,4.,-5. ZFNBTR=0.
 IFSTLG=-2 LCC=7 RETURN
 ZFNBTR=1. RETURN
```

SAMPLE TESTING PROGRAMS

```
*       TEST GETX
*       XEQ
*       LIST8
*       LABEL
CTGETX
        DIMENSION X(5),IX(5),I1(7),I2(3),C(10)
        CALL SETLIN (1.,1.,5,X)
        CALL SETVEC (IX,1,2,3,4,5)
        I1=4
        CALL VRSOUT (2,2,35HI3,26H. GETX,XGETX INPUTS - I1 = I3,C,1,I1)
        CALL VOUT   (2,1,X,1HX, 6H10F6.1,1,5)
        CALL VOUT   (2,1,IX,2HIX,4H10I6,1,5)
        X1= GETX (X,I1)
        IX1=IGETX(IX,I1)
        CALL VRSOUT (2,2,35H4X14HOUTPUTS - X = F6.2,4X4HIX = I6,C,X1,IX1)
        CALL SETVEC (I1,4,1,1,3,5,2,1)
        CALL SETVEC (I2,1,7,5)
        I3=3
        CALL VRSOUT (2,2,28HI3,22H. GETX,XGETX INPUTS - ,2,2)
        CALL VSOUT  (2,1,X,1HX,6H10F6.2,1,5,IX,2HIX,4H10I6,1,5,I1,2HI1,
       1  4H10I6,1,7,I2,2HI2,4H10I6,1,3,I3,2HI3,4H10I6,1,1)
        X1=GETX (X,I1,I2,I3)
        IX1=IGETX(IX,I1,I2,I3)
        CALL VRSOUT (2,2,35H4X14HOUTPUTS - X = F6.2,4X4HIX = I6,C,X1,IX1)
        CALL EXIT
        END
```

```
*       TEST INTHOL
*       XEQ
*       LIST8
*       LABEL
CTNTHOL
        DIMENSION  HOL(50), FMT(50), DATA(50)
        ITEST = 0
  7     ITEST = ITEST+1
        CALL SETKS(1,NHOL, 6H-53.31,HOL, 6H(F6.2),FMT, 1,NDATAD)
        CALL VRSOUT(2,3,14H9H EXAMPLE ,I1,ITEST,ITEST)
        GO TO (1,2,3),ITEST
  1     CALL INTHOL(NHOL,HOL,FMT,NDATAD,NDATAA,DATA)
        CALL VSOUT(2,3,NDATAA,6HNDATAA,2HI7,1,1,
       1             DATA,4HDATA,4HF9.2,1,1)
        GO TO 7
  2     NDATAD = 6
        GO TO 1
  3     CALL SETKS(2,NHOL, 3HXYZ,HOL, 6H  5 -9,HOL(2), 3,NDATAD)
        CALL INTHOL(NHOL,HOL,6HA6,2I3,NDATAD,NDATAA,DATA)
        CALL VSOUT(2,3,NDATAA,6HNDATAA,2HI7,1,1,
       1             DATA,4HDATA,9H1X,A3,2I7,1,3)
        CALL EXIT
        END
```

SAMPLE TESTING PROGRAMS

```
*     TEST LIMITS
*     XEQ
*     LIST8
*     LABEL
CTIMITS
      DIMENSION  S(3)
      CALL LIMITS(1,IANS, -0,-0,1, -0,+0,1, +0,-0,1, +0,+0,1,
     1  -0,-1,-0, -0,-1,+0, +0,-1,-0, +0,-1,+0, +0,+0,+0, +0,+0,-0,
     2  +0,-0,+0, +0,-0,-0, -0,+0,+0, -0,+0,-0, -0,-0,+0, -0,-0,-0)
      CALL VRSOUT(2, 3, 26H20H EXAMPLE 1.  IANS = ,I4, S, IANS)
      CALL LIMITS( 1,IANS1, 1.0,2.0,3.0)
      CALL LIMITS(21,IANS2, 3,1,4, 3.,1.,4., -3.,-4.,-1., 1,1,4, 1,2,3,
     1            4,1,4)
      CALL LIMITS(31,IANS3, 0.,0.,0., 1,1,1, -1,-1,-1, 3,1,2, 0,1,2)
      CALL VRSOUT(2, 3, 32H25H EXAMPLE 2.  IANS1...3 = ,3I4, S,
     1            IANS1, IANS2, IANS3)
      CALL LIMITS( 1,IANS1, 1.0,3.0,2.0)
      CALL LIMITS(21,IANS2, 3,4,1, 3.,4.,1., -3.,-1.,-4., 1,4,1, 1,3,2,
     1            4,4,1)
      CALL LIMITS(31,IANS3, 0.,0.,0., 1,1,1, -1,-1,-1, 3,2,1, 0,2,1)
      CALL VRSOUT(2, 3, 32H25H EXAMPLE 3.  IANS1...3 = ,3I4, S,
     1            IANS1, IANS2, IANS3)
      CALL EXIT
      END
```

```
*     TEST SHUFFL  -  NEEDS LOGICAL 9
*     XEQ
*     LIST8
*     LABEL
CTHUFFL
      DIMENSION  IRD(100),ISPACE(10),IXSHF1(10),IXSHF2(10)
      ITP=9
      CALL SETVEC(IRD,1,0,0,9,7,3,2,5,3,3,7,6,5,2,0,1,3,5,8,6,3,4,6,7,3,
     1                5,4,8,7,6,8,0,9,5,9,0,9,1,1,7,3,9,2,9,2,7,4,9,4,5,
     2                3,7,5,4,2,0,4,8,0,5,6,4,8,9,4,7,4,2,9,6,2,4,8,0,5,
     3                2,4,0,3,7,2,0,6,3,6,1,0,4,0,2,0,0,8,2,2,9,1,6,6,5)
      REWIND ITP
      WRITE OUTPUT TAPE ITP, 10, (IRD(I),I=1,100)
 10   FORMAT(50I1, 29X, 1H )
      REWIND ITP
      CALL SHUFFL(ITP,7,ISPACE,IXSHF1)
      CALL SHUFFL(ITP,10,ISPACE,IXSHF2)
      CALL VSOUT(2,5,IXSHF1,6HIXSHF1,8H20X,10I4,1,7,
     1              IXSHF2,6HIXSHF2,8H20X,10I4,1,10)
      REWIND ITP
      CALL EXIT
      END
```

SAMPLE TESTING PROGRAMS

```
*     TEST SIFT
*     XEQ
*     LIST8
*     LABEL
CTSIFT
      DIMENSION  X(50), XS1(50), XS2(50), XS3(50), XS4(50), XS5(50),
     1           XS6(50), FMT(2)
      CALL SETLIN(1.,1.,10,X)
      CALL SETK(-9.,XS5,XS6)
      CALL PLURNS(X,0,3,XS1, X,1,3,XS2, X,3,3,XS3, X,3,1,XS4,
     1           X,-1,3,XS5, X,1,0,XS6, X,5,2,X)
      CALL SIFT(4)
      CALL FMTOUT(2, 20H/////,11H EXAMPLE 1.)
      CALL SETVEC(FMT,6H(10X,1,6HOF5.1))
      CALL VSOUT(2,3, XS1,3HXS1,FMT,1,3, XS2,3HXS2,FMT,1,3,
     1  XS3,3HXS3,FMT,1,3, XS4,3HXS4,FMT,1,1, XS5,3HXS5,FMT,1,1,
     2  XS6,3HXS6,FMT,1,1, X,1HX,FMT,1,10)
      CALL EXIT
      END
```

```
*     TEST SIZEUP, SIZUPL
*     XEQ
*     LIST8
*     LABEL
CTIZEUP
      DIMENSION  X(10), INDEX1(10), INDEX2(10)
      ITEST = 0
 7    ITEST = ITEST+1
      LX = 5
      CALL SETVEC(X, 3.,-10.,-1.,2.,0.)
      GO TO (1,2,3),ITEST
 1    CALL SIZEUP(X,LX,INDEX1)
      CALL SIZUPL(X,LX,INDEX2)
      CALL VRSOUT(2,3,14H9H EXAMPLE ,I1,ITEST,ITEST)
      CALL VSOUT(2,3, INDEX1,6HINDEX1,3H5I5,1,5,
     1           INDEX2,6HINDEX2,3H5I5,1,5)
      GO TO 7
 2    CALL SETVEC(X ,1HX,1HA,1HC,1HN,1HA)
      GO TO 1
 3    CALL EXIT
      END
```

SAMPLE TESTING PROGRAMS

```
*       TEST TIMA2B
*       XEQ
*       LIST8
*       LABEL
CTTMA2B
        COMMON X,SPACE
        DIMENSION X(1001),SPACE(300)
B       XLXA=053400000000
        CALL SETKV (XLXA,1001,X)
        LOCB=XLOCF(X)
        CALL CLKON
 10     J=J+1
        CALL VRSOUT (2,2,21H1XI2,13H. TIMA2B TEST,J,J)
        CALL RDATA  (4,0,IANS,SPACE,4HNREG,NREG,6HZNDUMP,ZNDUMP,6HMINACC,
       1 MINACC)
        CALL CLOCK1(1,TIME)
        CALL TIMA2B (LOCB-NREG,LOCB,MINACC,SECS)
        CALL CLOCK1(2,TIME)
        CALL CSOUT (2,1,NREG,4HNREG,MINACC,6HMINACC,SECS,4HSECS,TIME,
       1 4HTIME)
        GO TO 10
        END
*       DATA
NREG=1000 MINACC=100 ZNDUMP=1. RETURN
NREG=100 ZNDUMP=0 RETURN
NREG=10 RETURN
NREG=2 RETURN
NREG=1 MINACC=100 RETURN
```

SAMPLE TIMING PROGRAMS

```
*     TIME TEST QACORR AND FORAC
*     XEQ
*     LIST8
*     LABEL
CTIMQAC
      DIMENSION  X(5000), SPACE(12000), ACOR(5000)
      COMMON     SPACE, X, ACOR
C
C OUTERMOST LOOP DECIDES WHETHER FULL AUTOCORRELATION OR
C 1/10 AUTOCORRELATION IS TO BE COMPUTED.  THE NEXT LEVEL LOOP
C SELECTS ONE OF 5 ACCURACY CONSTANTS FOR  QACORR.
C
      DO 100  IXFR=1,2
      FRCTN = NTHAF(IXFR, .10, 1.0)
      DO 100  IXA=1,5
      MXACC = XNTHAF(IXA, 25,50,100,250,500)
C
C THE INNERMOST LOOP SELECTS ONE OF 7 DATA LENGTHS, ACQUIRES THE DATA,
C TIMES THE CORRELATION PRODUCED BY  QACORR  AND THEN BY  FORAC,
C EXCEPT THAT OPERATION OF  FORAC  IS BYPASSED FOR DATA LENGTHS
C EXCEEDING 1000, AND FOR ACCURACY INDICES OTHER THAN 1 .
C
      DO 100  IXL=1,7
      LX    = XNTHAF(IXL, 25,50,100,250,500,1000,5000)
      MXLAG = XFIXF(FRCTN*FLOATF(LX)) - 1
      CALL GIVEX (LX, X)
      CALL TIMSUB(50, SECSQA)
      CALL QACORR(X, LX, MXACC, MXLAG, SPACE, ACOR, IANS)
      CALL VRSOUT(2, 2,
     1            49H34H LX, MXACC, MXLAG, IANS, SECSQA = , 4I6, F12.4,
     2            SPACE, LX, MXACC, MXLAG, IANS, SECSQA)
      IF (IXA-1)     100,70,100
 70   IF (LX-1000)   80,80,100
 80   CALL GIVEX (LX, X)
      CALL TIMSUB(50, SECSFA)
      CALL FORAC (X, LX, MXLAG, ACOR)
      CALL VRSOUT(2, 2, 36H21H LX, MXLAG, SECSFA = , 2I5, F12.4,
     1            SPACE, LX, MXLAG, SECSFA)
 100  CONTINUE
      CALL EXIT
      END
```

SAMPLE TIMING PROGRAMS

```
*     FORAC,  FORTRAN AUTOCORRELATION FOR COMPARISON WITH QACORR
*     LIST8
*     LABEL
CFORAC
      SUBROUTINE FORAC(X, LX, MXLAG, ACOR)
C
C TOKEN DIMENSIONS
C
      DIMENSION  X(2), ACOR(2)
      JMAX     = MXLAG + 1
      DO 20     J=1,JMAX
      SUM      = 0.0
      NMAX     = LX - J + 1
      DO 10     I=1,NMAX
      K        = J + I
 10   SUM      = SUM + X(I)*X(K-1)
 20   ACOR(J)  = SUM
      RETURN
      END
```

```
*     GIVEX,  PROVIDES A DATA VECTOR FOR QACORR TIME TESTS
*     LIST8
*     LABEL
CGIVEX
      SUBROUTINE GIVEX(LX, X)
C
C TOKEN DIMENSIONS
C
      DIMENSION  X(2)
C
C THE DATA VECTOR PROVIDED IS A MORE OR LESS WHITE LIGHT SERIES
C WITH VALUES IN THE RANGE  -1.0  TO  +1.0  .
C
      DO 10  I=1,LX
 10   X(I) = COSF(100.*FLOATF(I))
      RETURN
      END
```

SAMPLE TIMING PROGRAMS

```
*     TIME TEST ASPECT
*     XEQ
*     LIST8
*     LABEL
CTIMASP
      DIMENSION  ACOR(5001), SPECT(1001), SPACE(2010), COSTAB(1001)
      COMMON     ACOR, SPECT, SPACE, COSTAB
C
C INITIALIZE BY SETTING UP THE AUTOCORRELATION OF A SAW-TOOTH.
C
      CALL SETLIN(5001., -1., 5001, ACOR)
      CALL SQUARE(ACOR, 5001, ACOR)
C
C OUTER LOOP SELECTS ONE OF SIX FREQUENCY INCREMENT CONSTANTS,
C AND ESTABLISHES THE CORRESPONDING COSINE TABLE.
C
      DO 100  IXMFRQ=1,6
      MFREQ = XNTHAF(IXMFRQ, 25,50,100,250,500,1000)
      CALL COSTBL(MFREQ, COSTAB)
C
C INNER LOOP SELECTS ONE OF 8 CORRELATION LENGTHS AND PROCEEDS WITH
C THE TIMING, BUT HAS A BYPASS FOR CASES IN WHICH THE NO. OF
C FREQUENCIES EXCEEDS THE CORRELATION LENGTH.
C
      DO 100  IXMXLG=1,8
      MXLAG = XNTHAF(IXMXLG, 25,50,100,250,500,1000,2500,5000)
      IF (MXLAG-MFREQ)          100,70,70
 70   CALL TIMSUB(50, SECSAS)
      CALL ASPECT(ACOR, MXLAG, COSTAB, MFREQ, 0, MFREQ,
     1            1.0,  SPECT, SPACE,  DUMMY, ERR)
      CALL VRSOUT(2, 2,
     1            45H29H MXLAG, MFREQ, ERR, SECSAS = , 2I6, 2F12.4,
     2            SPACE, MXLAG, MFREQ, ERR, SECSAS)
 100  CONTINUE
      CALL EXIT
      END
```

SAMPLE TIMING PROGRAMS

```
*     TIME TEST WLLSFP AND SIMEQ
*     XEQ
*     LIST8
*     LABEL
CTIMWAS
      DIMENSION  X(502), R(502), G(500), A(501), C(1010), SPACE(1002),
     1           AA(10000), BB(101), E(101)
      COMMON     AA, SPACE, C
C
C INITIALIZE BY SETTING UP THE NORMALIZED AUTOCORRELATION OF A SAWTOOTH
C IN  R(1...501),  AND A LINEAR RIGHT HAND SIDE IN  G(1...500).
C (THE NORMALIZATION IS NECESSARY TO PREVENT OVERFLOW IN  SIMEQ.)
C THE VARIABLE NAMES ARE CHOSEN AS DEFINED BY  WLLSFP.
C
      CALL SETLIN(0.0, 1.0, 501, X)
      CALL WAC   (501, X, 501, R)
      CALL DIVIDE(R, 501, R, R)
      LR = 500
      CALL SETLIN(1.0, 1.0, 500, G)
      CALL DIVIDE(G, 500, G(500), G)
C
C LOOP SELECTS ONE OF 8 MATRIX SIZES,  LA,  RANGING FROM 3 TO 500,
C TIMES  WLLSFP  FOR THIS SIZE, AND THEN, PROVIDED  LA  DOESN'T
C EXCEED 100, TIMES  SIMEQ.
C
      DO 100  IXLA=1,8
      LA = XNTHAF(IXLA, 3,5,10,25,50,100,250,500)
      CALL TIMSUB(50, SECSWL)
      CALL WLLSFP(LR, R, G, LA, A, C)
      CALL VRSOUT(2, 2, 28H14H LA, SECSWL = , I5, F12.4,
     1            SPACE, LA, SECSWL)
      IF (LA-100)    80,80,100
C
C SINCE  SIMEQ  DESTROYS BOTH THE INPUT MATRIX AND THE RIGHT HAND SIDE,
C THESE INPUTS MUST BE ESTABLISHED FOLLOWING A  CALL INTMSB  STATEMENT
C AND PRIOR TO THE  CALL TIMSUB  STATEMENT.
C
 80   CALL INTMSB
      CALL REVER (R, 501, SPACE)
      CALL REVER (SPACE, 500, SPACE(502))
      DO 90  I=1,LA
      K = 502 - I
      J = 1 + (I-1)*LA
 90   CALL MOVE  (LA, SPACE(K), AA(J))
      D = 1.0
      CALL MOVE  (LA, G, BB)
      CALL TIMSUB(50, SECSIM)
      CALL SIMEQ (LA, LA, 1, AA, BB, D, E, ERR)
      CALL VRSOUT(2, 2, 34H19H LA, ERR, SECSIM = , I5, 2F12.4,
     1            SPACE, LA, ERR, SECSIM)
 100  CONTINUE
      CALL EXIT
      END
```

3
Program Categorizations

The usages presented in Section 2 are only samples, and highly specialized ones at that. For systematic access to programs of interest one needs an orderly indexing such as that provided by the general sortings discussed in this section. The characteristics on which these categorizations are based can be broadly divided into functional and nonfunctional ones.

The functions performed by the programs of Section 10 can be grouped into the following fifteen classes.

1. Administration
2. Input-output
3. Data transmission and access
4. Data-form changing
5. Data generation
6. Data inquiry
7. Elementary numerical functions
8. Miscellaneous numerical functions
9. Probability and statistics computations
10. Integration and differentiation
11. 2-D array and 3-D array operations
12. Polynomial computations
13. Correlation and convolution
14. Harmonic transformation
15. Miscellaneous spectral-analysis operations

In the following Summary of Functional Classifications, each of these classes is broken down into a number of subclasses or categories, according to which the programs are sorted in the bulk of this section. This summary thus delineates the scope of the programs and constitutes a starting point in a functionally oriented search of the library. It should be noted that there is some overlap in the category definitions. Moreover, programs with multiple functions may appear in two or more of the categories.

The remainder of the section is then devoted to program sortings based on nonfunctional characteristics such as authorship, language, linkage, and equipment uses, and on subjective qualities such as speed and utility. The categories used there are self-explanatory except, perhaps, for the term "FAP necessarily," by which we imply that it is either impossible or extremely awkward to express the function performed using only basic FORTRAN statements.

SUMMARY OF FUNCTIONAL CLASSIFICATIONS

1. ADMINISTRATIVE PROGRAMS

FOR CONTROL OF PROGRAM FLOW
FOR EXPANDING SYSTEM CAPABILITY
FOR UNORTHODOX SUBROUTINE USAGE
FOR INDEX LOGIC
FOR DOCUMENTING EXECUTIONS
FOR EQUIPMENT CONTROL
FOR PROGRAM TIMING
FOR ABSOLUTE MEMORY INFORMATION
FOR PROPER USE OF MISNAMED VARIABLES

2. INPUT-OUTPUT PROGRAMS

FOR BCD INPUT TO CORE
FOR BINARY INPUT TO CORE
FOR BCD OUTPUT FROM CORE
FOR BINARY OUTPUT FROM CORE
FOR GRAPHICAL OUTPUT FROM CORE
FOR FORMAT PURPOSES

3. DATA TRANSMISSION AND ACCESS PROGRAMS

FOR STORAGE-TO-STORAGE MOVEMENT
FOR STORAGE-TO-TAPE MOVEMENT
FOR TAPE-TO-STORAGE MOVEMENT
FOR TAPE-TO-TAPE MOVEMENT
FOR INFORMATION STORAGE
FOR INFORMATION RETRIEVAL

4. DATA FORM-CHANGING PROGRAMS

FOR CONVERTING DATA MODE
FOR PACKING DATA
FOR UNPACKING DATA
FOR SCALING DATA
FOR NORMALIZING DATA
FOR ROUNDING DATA
FOR SHIFTING DATA
FOR CHANGING DATA SPACING

5. DATA GENERATING PROGRAMS

FOR GENERATING HOLLERITH
FOR GENERATING RANDOM NUMBERS
FOR GENERATING SINUSOIDS
FOR GENERATING SCALARS
FOR GENERATING 1-D ARRAYS

6. DATA INQUIRY PROGRAMS

FOR FINDING EXTREMAL VALUES
FOR COMPARING DATA
FOR SEARCHING DATA
FOR SELECTING DATA
FOR ORDERING DATA
FOR CLASSIFYING DATA

7. ELEMENTARY NUMERICAL PROGRAMS

FOR ADDITION
FOR SUBTRACTION
FOR MULTIPLICATION
FOR DIVISION
FOR MODIFYING SIGN
FOR RAISING TO POWERS
FOR TAKING ROOTS
FOR TRIGONOMETRIC FUNCTIONS
FOR COLLAPSING VECTORS
FOR ROTATING VECTORS
FOR REVERSING VECTORS
FOR EXCHANGING VECTORS
FOR REFLECTING VECTORS

8. MISCELLANEOUS NUMERICAL PROGRAMS

FOR INTERPOLATION
FOR SAMPLE BASE CHANGING
FOR GENERATING SINUSOIDS
FOR TRIGONOMETRIC FUNCTIONS
FOR TREATING ODD AND EVEN PARTS
FOR FITTING EQUATIONS TO DATA
FOR CONTOURING
FOR DELTA AND STEP FUNCTIONS
FOR CONVERTING COMPLEX NUMBERS
FOR MOVING SUMMATION
FOR INVERTING FUNCTIONS
FOR DOT PRODUCTS

SUMMARY OF FUNCTIONAL CLASSIFICATIONS

9. PROBABILITY AND STATISTICS

FOR FINDING MOMENTS
FOR FINDING AVERAGES
FOR FINDING R.M.S. VALUES
FOR FINDING SUMS OF SQUARES
FOR FINDING SUMS OF DIFFERENCES
FOR GENERATING RANDOM NUMBERS
FOR RANDOMIZING DATA
FOR FINDING DISTRIBUTIONS
FOR PROBABILITY TRANSFORMATION
FOR CHI-SQUARE ANALYSIS
FOR DEPENDENCY TESTING
FOR NORMAL CURVE INTEGRATION

10. INTEGRATION AND DIFFERENTIATION PROGRAMS

FOR DEFINITE INTEGRATION
FOR INDEFINITE INTEGRATION
FOR DIFFERENTIATION
FOR INDEFINITE SUMMATION
FOR DIFFERENCING

11. 2-D ARRAY AND 3-D ARRAY PROGRAMS

FOR MATRIX MULTIPLICATION
FOR MATRIX INVERSION
FOR SOLVING MATRIX EQUATIONS
FOR DETERMINANT EVALUATION
FOR MATRIX TRANSPOSITION
FOR MATRIX FACTORIZATION
FOR 2-D ARRAY ROTATION
FOR INTERPOLATING 2-D ARRAY COLUMNS
FOR 2-D ARRAY DOT PRODUCTS
FOR 2-D ARRAY CORRELATION
FOR 2-D ARRAY FOURIER TRANSFORMATION
FOR SOLVING 2-D ARRAY EQUATIONS
FOR MATRIX VECTOR REVERSAL
FOR MATRIX VECTOR DOT PRODUCT
FOR MATRIX VECTOR CORRELATION
FOR SOLVING MATRIX VECTOR EQUATIONS

12. POLYNOMIAL PROGRAMS

FOR POLYNOMIAL EVALUATION
FOR FINDING POLYNOMIAL ROOTS
FOR POLYNOMIAL MULTIPLICATION
FOR POLYNOMIAL DIVISION
FOR POLYNOMIAL SQUARE ROOTS
FOR SYNTHESIZING POLYNOMIALS

13. CORRELATIONS AND CONVOLUTIONS

FOR AUTOCORRELATION
FOR CROSS-CORRELATION
FOR CONVOLUTION
FOR DOT PRODUCTS

14. HARMONIC TRANSFORMS

FOR COSINE TRANSFORMATION
FOR SINE TRANSFORMATION
FOR FOURIER TRANSFORMATION
FOR INVERSE FOURIER TRANSFORMATION

15. MISCELLANEOUS SPECTRAL ANALYSIS PROGRAMS

FOR DANIELL WEIGHTING
FOR SPECTRAL FACTORIZATION
FOR GENERATING NUMERICAL FILTERS
FOR CONVERTING TO AMPLITUDE AND PHASE
FOR CONVERTING TO REAL AND IMAGINARY
FOR SPECTRAL COMPARISONS
FOR GENERATING SINUSOIDS

The sorted lists which will follow below need some introduction with regard to format. First of all, the sortings have been made on the basis of names of principal entries, and the lists are alphabetically ordered with respect to these names. In the case of multiple-entry programs, the names of the secondary entries appear as a parenthetical list following each appearance of the principal entry name. However, a parenthetical list following a name is not necessarily a list of secondary entries; it may alternatively be a list of functionally related programs. For example, each appearance of the Fourier-transform program QFURRY is followed by a parenthetical reference to the inverse Fourier-transform program QIFURY, and conversely.

Secondly, it should be noted that we run into an occasional problem resulting from the fact that the present sortings are necessarily based on six-character names for the principal entries, whereas in the program listings of Section 10 we sometimes have appended serial numbers and/or computer numbers to distinguish between programs of identical principal entry names. The sortings have been made on the basis of effective names. The effective names are identical to the principal entry names in cases where no ambiguity can arise. Effective names for the exceptional cases are listed below.

Effective Name	True Name	Effective Name	True Name
CLOCK1	CLOCK1 (7090)	LINE	LINE (709)
CNVLV2	CONVLV-II	LINE90	LINE (7090)
DISPLA	DISPLA (709)	LINEH	LINEH (709)
DSPL 90	DISPLA (7090)	LINH90	LINEH (7090)
FRAME	FRAME (709)	LINEV	LINEV (709)
FRAM90	FRAME (7090)	LINV90	LINEV (7090)
FT24II	FT24 -II	MULK2	MULK -II
HST2	HSTPLT -II	SETK2	SETK -II
HST309	HSTPLT -III (709)	SETKS2	SETKS -II
HST390	HSTPLT -III (7090)	TIMA2B	TIMA2B (7094)

PROGRAMS SORTED BY FUNCTION

```
********************************
* 1. ADMINISTRATIVE PROGRAMS *
********************************

  FOR CONTROL OF PROGRAM FLOW
      INDEX  (CHUSET,  SETAPT,  SETEST,  VINDEX), SEVRAL  (PLURAL).

  FOR EXPANDING SYSTEM CAPABILITY
     FNDFMT,    GETX  ( IGETX), LOCATE  (   ARG,    CALL,   CALL2,  RETURN,
     SETSBV,   SETUP,   STORE,   WHERE,    XARG,  XINDEX,   XNAME,  XNARGS),
     ONLINE  ( (STH),   (STHD),   (STHM)), PLURNS,   RDATA,  REREAD  (ENDFIL,
     EOFSET,   (TSH),   (TSHM)), RPLFMT,    SAME  ( XSAME), SEVRAL  (PLURAL),
     VARARG.

  FOR UNORTHODOX SUBROUTINE USAGE
     LOCATE  (   ARG,    CALL,   CALL2,  RETURN,  SETSBV,   SETUP,   STORE,
      WHERE,    XARG,  XINDEX,   XNAME,  XNARGS), PLURNS,  SEVRAL  (PLURAL),
     VARARG.

  FOR INDEX LOGIC
     FASTRK,    GETX  ( IGETX),  INDEX  (CHUSET,  SETAPT,  SETEST,  VINDEX),
     LOCATE  (   ARG,    CALL,   CALL2,  RETURN,  SETSBV,   SETUP,   STORE,
      WHERE,    XARG,  XINDEX,   XNAME,  XNARGS).

  FOR DOCUMENTING EXECUTIONS
     DADECK,  LISTNG,  MEMUSE,   RDATA,  XLCOMN.

  FOR EQUIPMENT CONTROL
     CARIGE,   CLKON,   FRAME  (FRAM90),  FSKIP,  ONLINE  ( (STH),   (STHD),
     (STHM)), REREAD  (ENDFIL,  EOFSET,   (TSH),   (TSHM)),  RSKIP,  SETINO,
     SWITCH,  TRMINO,  ZEFBCD  (ZEFBIN).

  FOR PROGRAM TIMING
      CLKON,  CLOCK1,  TIMA2B,  TIMSUB  (INTMSB).

  FOR ABSOLUTE MEMORY INFORMATION
     IXCARG,     LOC,  MEMUSE,  XLCOMN,   XLOCV.

  FOR SUBROUTINE LIBRARY STUDY
    (NO ENTRIES FOR THIS CATEGORY)

  FOR PROPER USE OF MISNAMED VARIABLES
       SAME  ( XSAME).

****************************
* 2. INPUT-OUTPUT PROGRAMS *
****************************

  FOR BCD INPUT TO CORE
      RDATA,  REREAD  (ENDFIL,  EOFSET,   (TSH),   (TSHM)), ZEFBCD  (ZEFBIN).

  FOR BINARY INPUT TO CORE
     INDATA,  PACDAT,  ZEFBCD  (ZEFBIN).
```

PROGRAMS SORTED BY FUNCTION

FOR BCD OUTPUT FROM CORE
COLABL, CSOUT, CVSOUT, DISPLA (DSPL90), FMTOUT, MLI2A6, MOUT, MOUTAI, ONLINE ((STH), (STHD), (STHM)), PWMLIV, VECOUT, VOUT, VRSOUT, VSOUT.

FOR BINARY OUTPUT FROM CORE
OUDATA, WRTDAT.

FOR GRAPHICAL OUTPUT FROM CORE
CNTRDB, CNTROW, CONTUR, DISPLA (DSPL90), GRAPH, GRAPHX, HSTPLT (HST2, HST309, HST390), LINE (LINE90), LINEH (LINH90), LINEV (LINV90), PLOTVS, PLTVS1.

FOR FORMAT PURPOSES
COLABL, DSPFMT, FNDFMT, RPLFMT.

```
**************************
* 3. DATA TRANSMISSION   *
*    AND ACCESS PROGRAMS *
**************************
```

FOR STORAGE-TO-STORAGE MOVEMENT
EXCHVS, MOVE, MOVECS, MOVREV, MRVRS, MVBLOK.

FOR STORAGE-TO-TAPE MOVEMENT
GETRD1, OUDATA, WRTDAT.

FOR TAPE-TO-STORAGE MOVEMENT
INDATA, PACDAT.

FOR TAPE-TO-TAPE MOVEMENT
CPYFL2, DADECK.

FOR INFORMATION STORAGE
OUDATA, PAKN (UNPAKN), SETINO, TRMINO, WRTDAT.

FOR INFORMATION RETRIEVAL
GETX (IGETX), INDATA, LISTNG, NTHA (XNTHA), PACDAT, UNPAKN (PAKN).

```
**********************************
* 4. DATA FORM-CHANGING PROGRAMS *
**********************************
```

FOR CONVERTING DATA MODE
FIXV (FIXVR), FLOATM, FLOATV, FXDATA (FLDATA), HVTOIV (IVTOHV), INTHOL, ITOMLI, IVTOHV (HVTOIV), MLI2A6, XFIXM.

FOR PACKING DATA
PAKN (UNPAKN).

FOR UNPACKING DATA
UNPAKN (PAKN).

FOR SCALING DATA
FXDATA (FLDATA), MLISCL, SCPSCL, SMPSON.

FOR NORMALIZING DATA
FXDATA (FLDATA), NMZMG1, NRMVEC.

FOR ROUNDING DATA
FIXV (FIXVR), FXDATA (FLDATA), RND (RNDDN, RNDUP), RNDV (RNDVDN, RNDVUP), XDIV (XDIVR), XDVIDE (XDVIDR), XFIXM, XVDVBV (XDVRBV).

FOR SHIFTING DATA
HLADJ (HRADJ), ITOMLI, LSHFT (XLSHFT), SHFTR1, SHFTR2.

FOR CHANGING DATA SPACING
MOVREV.

```
******************************
* 5. DATA GENERATING PROGRAMS *
******************************
```

FOR GENERATING HOLLERITH
GENHOL, GETHOL, GNHOL2.

FOR GENERATING RANDOM NUMBERS
GETRD1.

FOR GENERATING SINUSOIDS
COSTBL (COSTBX, SINTBL, SINTBX), SEQSAC (NEXCOS, NEXSIN).

FOR GENERATING SCALARS
SETK (SETKS, SETVEC), SETK2, SETKP (SETVCP), SETKS2.

FOR GENERATING 1-D ARRAYS
SETK (SETKS, SETVEC), SETKP (SETVCP), SETKV, SETKVS, SETLIN (XSTLIN), SETLNS, STZ, STZS.

```
***************************
* 6. DATA INQUIRY PROGRAMS *
***************************
```

FOR FINDING EXTREMAL VALUES
MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM).

FOR COMPARING DATA
CMPARP (CMPARS), CMPARV (CMPARL), CMPRA (CMPRFL, XCMPRA), INDEX (CHUSET, SETAPT, SETEST, VINDEX), LIMITS, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), XACTEQ, XLIMIT.

FOR SEARCHING DATA
FASCN1, FASTRK, NXALRM, SEARCH, SRCH1.

FOR SELECTING DATA
CHOOSE, GETX (IGETX), NTHA (XNTHA), WHICH (XWHICH).

FOR ORDERING DATA
SIZEUP (SIZUPL).

FOR CLASSIFYING DATA
MONOCK, XOOZE.

```
*************************************
* 7. ELEMENTARY NUMERICAL PROGRAMS *
*************************************

   FOR ADDITION
        ADDK  ( ADDKS,    DIVK,   DIVKS,    MULK,   MULK2,    SUBK,   SUBKS,
       XADDK,  XADDKS,   XDIVK,  XDIVKS,   XDVRK,  XDVRKS,   XMULK,  XMULKS,
       XSUBK,  XSUBKS),  BOOST  (DPRESS,  XBOOST,  XDPRSS), FAPSUM,  NRMVEC,
         SUM  (  XSUM), VPLUSV  (VMNUSV,  XVMNSV,  XVPLSV).

   FOR SUBTRACTION
        ADDK  ( ADDKS,    DIVK,   DIVKS,    MULK,   MULK2,    SUBK,   SUBKS,
       XADDK,  XADDKS,   XDIVK,  XDIVKS,   XDVRK,  XDVRKS,   XMULK,  XMULKS,
       XSUBK,  XSUBKS),  BOOST  (DPRESS,  XBOOST,  XDPRSS),  REMAV,  VPLUSV
     (VMNUSV,  XVMNSV,  XVPLSV), XREMAV.

   FOR MULTIPLICATION
        ADDK  ( ADDKS,    DIVK,   DIVKS,    MULK,   MULK2,    SUBK,   SUBKS,
       XADDK,  XADDKS,   XDIVK,  XDIVKS,   XDVRK,  XDVRKS,   XMULK,  XMULKS,
       XSUBK,  XSUBKS),  DUBLX  ( DUBLL,   HALVL,   HALVX), MLISCL,   MULK2,
      MULPLY,  VTIMSV  (XVTMSV).

   FOR DIVISION
        ADDK  ( ADDKS,    DIVK,   DIVKS,    MULK,   MULK2,    SUBK,   SUBKS,
       XADDK,  XADDKS,   XDIVK,  XDIVKS,   XDVRK,  XDVRKS,   XMULK,  XMULKS,
       XSUBK,  XSUBKS), DIVIDE,   DUBLX  ( DUBLL,   HALVL,   HALVX), VDVBYV,
        XDIV  ( XDIVR), XDVIDE  (XDVIDR), XVDVBV  (XDVRBV).

   FOR MODIFYING SIGN
      ABSVAL,  CHPRTS  (RVPRTS), CHSIGN,  MOVREV.

   FOR RAISING TO POWERS
      MVSQAV,   POWER  (SMPRDV), SQRMLI,  SQUARE  (XSQUAR).

   FOR TAKING ROOTS
      SQROOT,  XSQRUT.

   FOR TRIGONOMETRIC FUNCTIONS
      ARCTAN,  SEQSAC  (NEXCOS,  NEXSIN).

   FOR COLLAPSING VECTORS
      COLAPS,  KOLAPS.

   FOR ROTATING VECTORS
      ROTAT1.

   FOR REVERSING VECTORS
      CHPRTS  (RVPRTS), MOVREV,   REVER,  REVERS.

   FOR EXCHANGING VECTORS
      EXCHVS.

   FOR REFLECTING VECTORS
      REFLEC  (XRFLEC).
```

PROGRAMS SORTED BY FUNCTION

```
****************************************
* 8. MISCELLANEOUS NUMERICAL PROGRAMS *
****************************************
```

FOR INTERPOLATION
ARBCOL, EXPAND, INTOPR, LINTR1, QINTR1.

FOR SAMPLE BASE CHANGING
EXPAND, NURINC, SIFT.

FOR GENERATING SINUSOIDS
COSTBL (COSTBX, SINTBL, SINTBX), SEQSAC (NEXCOS, NEXSIN).

FOR TRIGONOMETRIC FUNCTIONS
ARCTAN, SEQSAC (NEXCOS, NEXSIN).

FOR TREATING ODD AND EVEN PARTS
CHPRTS (RVPRTS), SPLIT (REFIT).

FOR FITTING EQUATIONS TO DATA
CUFIT1, INTOPR, LSLINE, PRBFIT, QUFIT1.

FOR CONTOURING
CNTRDB, CONTUR.

FOR DELTA AND STEP FUNCTIONS
DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR).

FOR CONVERTING COMPLEX NUMBERS
AMPHZ (REIM).

FOR MOVING SUMMATION
BLKSUM, MUVADD, MVINAV, MVNSUM, MVSQAV.

FOR INVERTING FUNCTIONS
IFNCTN.

FOR DOT PRODUCTS
DOTJ, FDOT (FDOTR), VDOTV.

```
**********************************
* 9. PROBABILITY AND STATISTICS *
**********************************
```

FOR FINDING MOMENTS
POWER (SMPRDV).

FOR FINDING AVERAGES
AVRAGE, MVINAV, MVSQAV, REMAV, TAMVL (TAMVR), XAVRGE (XAVRGR), XREMAV.

FOR FINDING R.M.S. VALUES
RMSDEV (RMSDAV).

FOR FINDING SUMS OF SQUARES
SQRDFR (SQRDEV), SQRSUM (XSQSUM), XSQDFR (XSQDEV).

FOR FINDING SUMS OF DIFFERENCES
SQRDFR (SQRDEV), SUMDFR (SUMDEV, XSMDEV, XSMDFR), XSQDFR (XSQDEV).

FOR GENERATING RANDOM NUMBERS
GETRD1.

FOR RANDOMIZING DATA
SHUFFL.

FOR FINDING DISTRIBUTIONS
FRQCT1, FRQCT2, POKCT1, PRBFIT, PROB2.

FOR PROBABILITY TRANSFORMATION
GRUP2, MPSEQ1, NOINT1 (NOINT2).

FOR CHI-SQUARE ANALYSIS
CHISQR, KIINT1.

FOR DEPENDENCY TESTING
MSCON1, POKCT1.

FOR NORMAL CURVE INTEGRATION
NOINT1 (NOINT2).

```
*********************************
* 10. INTEGRATION AND           *
*     DIFFERENTIATION PROGRAMS *
*********************************
```

FOR DEFINITE INTEGRATION
MVNTIN (MVNTNA), SMPSON, TINGL (TINGLA).

FOR INDEFINITE INTEGRATION
IDERIV (DERIVA), INTGRA (IINTGR), TAMVL (TAMVR).

FOR DIFFERENTIATION
DERIVA (IDERIV), IINTGR (INTGRA).

FOR INDEFINITE SUMMATION
INTSUM (DIFPRS, XNTSUM).

FOR DIFFERENCING
DIFPRS (INTSUM, XDFPRS).

```
*****************************************
* 11. 2-D ARRAY AND 3-D ARRAY PROGRAMS *
*****************************************
```

FOR MATRIX MULTIPLICATION
MATML1, MATML3.

FOR MATRIX INVERSION
MATINV, SIMEQ (DETRM).

FOR SOLVING MATRIX EQUATIONS
LSSS1, RLSPR, RLSSR, SIMEQ (DETRM), WLLSFP.

PROGRAMS SORTED BY FUNCTION

FOR DETERMINANT EVALUATION
SIMEQ (DETRM).

FOR MATRIX TRANSPOSITION
MATRA, MATRA1.

FOR MATRIX FACTORIZATION
MFACT.

FOR 2-D ARRAY ROTATION
ROAR2.

FOR INTERPOLATING 2-D ARRAY COLUMNS
ARBCOL.

FOR 2-D ARRAY DOT PRODUCTS
DOTP.

FOR 2-D ARRAY CORRELATIONS
SPCOR2.

FOR 2-D ARRAY FOURIER TRANSFORMATION
PLANSP.

FOR SOLVING 2-D ARRAY EQUATIONS
FIRE2, RLSPR2.

FOR MATRIX VECTOR REVERSAL
MRVRS.

FOR MATRIX VECTOR DOT PRODUCT
MDOT, MDOT3.

FOR MATRIX VECTOR CORRELATION
CRSVM.

FOR SOLVING MATRIX VECTOR EQUATIONS
MIFLS, MIPLS, MISS.

****
* 12. POLYNOMIAL PROGRAMS *
****

FOR POLYNOMIAL EVALUATION
FASCUB, IPLYEV, POLYEV.

FOR FINDING POLYNOMIAL ROOTS
MULLER.

FOR POLYNOMIAL MULTIPLICATION
CONVLV, CNVLV2.

FOR POLYNOMIAL DIVISION
POLYDV.

FOR POLYNOMIAL SQUARE ROOTS
PSQRT.

FOR SYNTHESIZING POLYNOMIALS
PLYSYN, POLYSN.

```
*************************************
* 13. CORRELATIONS AND CONVOLUTIONS *
*************************************
```

FOR AUTOCORRELATION
CROSS, CROST, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QACORR, QXCORR, WAC.

FOR CROSS-CORRELATION
CROSS, CROST, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QXCORR, QXCOR1.

FOR CONVOLUTION
CONVLV, CNVLV2, QCNVLV.

FOR DOT PRODUCTS
DOTJ, FDOT (FDOTR), VDOTV.

```
***************************
* 14. HARMONIC TRANSFORMS *
***************************
```

FOR COSINE TRANSFORMATION
ASPECT, ASPEC2, COSIS1, COSP (COSISP, SISP).

FOR SINE TRANSFORMATION
COSIS1, COSP (COSISP, SISP).

FOR FOURIER TRANSFORMATION
COSIS1, COSP (COSISP, SISP), FT24 (FT24II), QFURRY (QIFURY), XSPECT.

FOR INVERSE FOURIER TRANSFORMATION
QIFURY (QFURRY).

```
******************************
* 15. MISCELLANEOUS SPECTRAL *
*     ANALYSIS PROGRAMS      *
******************************
```

FOR DANIELL WEIGHTING
ADANL (ADANX, XDANL, XDANX).

FOR SPECTRAL FACTORIZATION
FACTOR.

FOR GENERATING NUMERICAL FILTERS
GNFLT1.

PROGRAMS SORTED BY FUNCTION

FOR CONVERTING TO AMPLITUDE AND PHASE
AMPHZ (REIM).

FOR CONVERTING TO REAL AND IMAGINARY
AMPHZ (REIM).

FOR SPECTRAL COMPARISONS
MXRARE.

FOR GENERATING SINUSOIDS
COSTBL (COSTBX, SINTBL, SINTBX), SEQSAC (NEXCOS, NEXSIN).

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

****
* AUTHORSHIP *
****

CLAERBOUT, JON F.
ADANL (ADANX, XDANL, XDANX), AMPHZ (REIM), CCNVLV, COSTBL (COSTBX, SINTBL, SINTBX), FAPSUM, FSKIP, INDATA, MAXSN (MAXAB, MINAB, MINSN), MOVE, OUDATA, PAKN (UNPAKN), POLYEV, PSQRT, SAME (XSAME), STZ, UNPAKN (PAKN), VARARG, WAC.

CLARK, JACQUELINE
COLAPS, KOLAPS.

GALBRAITH, JAMES N., JR.
CHISQR, FACTOR, FRQCT2, GRUP2, HSTPLT (HST2, HST309, HST390), LINEH (LINH90), LINH90, LINEV (LINV90), LINV90, MPSEQ1, MSCON1, PROB2, SMPSON, ZEFBCD (ZEFBIN).

GREENFIELD, ROY J.
PRBFIT.

HANSON, I.
MULLER.

M.I.T. COMPUTATION CENTER STAFF
FRAM90.

PAN, CHEH
FT24 (FT24II).

PROCITO, JOSEPH T.
QINTR1.

ROBINSON, ENDERS A.
PLYSYN.

SIMPSON, STEPHEN M., JR.
ABSVAL, ADDK (ADDKS, DIVK, DIVKS, MULK, MULK2, SUBK, SUBKS, XADDK, XADDKS, XDIVK, XDIVKS, XDVRK, XDVRKS, XMULK, XMULKS, XSUBK, XSUBKS), ARBCOL, ASPECT, ASPEC2, AVRAGE, BLKSUM, BOOST (DPRESS, XBOOST, XDPRSS), CARIGE, CHOOSE, CHPRTS (RVPRTS), CHSIGN, CLOCK1, CMPARP (CMPARS), CMPARV (CMPARL), CNTRDB, CNTROW, COLABL, CONTUR, COSP (COSISP, SISP), CUFIT1, CVSOUT, DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), DERIVA (IDERIV), DIFPRS (INTSUM, XDFPRS), DIVIDE, DSPFMT, DUBLX (DUBLL, HALVL, HALVX), EXCHVS, EXPAND, FASCN1, FASCUB, FASTRK, FIXV (FIXVR), FLOATM, FLOATV, FMTOUT, FNDFMT, FRQCT1, FXDATA (FLDATA), GETHOL, GETRD1, GNFLT1, GRAPH, GRAPHX, HLADJ (HRADJ), HVTOIV (IVTOHV), IDERIV (DERIVA), IFNCTN, IINTGR (INTGRA), INDEX (CHUSET, SETAPT, SETEST, VINDEX), INTGRA (IINTGR), INTOPR, INTSUM (DIFPRS, XNTSUM), ITOMLI, IVTOHV (HVTOIV), IXCARG, KIINT1, LIMITS, LINE (LINE90), LINE90, LINTR1, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), MAXSNM (MAXABM, MINABM, MINSNM), MEMUSE, MLISCL, MLI2A6, MONOCK,

(CONTINUED NEXT PAGE)

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

MOUTAI, MOVECS, MULK2, MULPLY, MUVADD, MVBLOK, MVINAV, MVNSUM, MVNTIN (MVNTNA), MVSQAV, MXRARE, NTHA (XNTHA), NURINC, NXALRM, PLOTVS, PLTVS1, PLURNS, POKCT1, POWER (SMPRDV), PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), PWMLIV, QACORR, QCNVLV, QFURRY (QIFURY), QIFURY (QFURRY), QUFIT1, QXCORR, REFLEC (XRFLEC), REMAV, REVER, RMSDEV (RMSDAV), RNDV (RNDVDN, RNDVUP), RPLFMT, SCPSCL, SEQSAC (NEXCOS, NEXSIN), SETINO, SETK (SETKS, SETVEC), SETK2, SETKP (SETVCP), SETKS2, SETKV, SETKVS, SETLIN (XSTLIN), SETLNS, SEVRAL (PLURAL), SHFTR1, SHUFFL, SIFT, SPLIT (REFIT), SQRDFR (SQRDEV), SQRMLI, SQROOT, SQRSUM (XSQSUM), SQUARE (XSQUAR), STZS, SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), SWITCH, TAMVL (TAMVR), TINGL (TINGLA), TRMINO, VDOTV, VDVBYV, VECOUT, VOUT, VPLUSV (VMNUSV, XVMNSV, XVPLSV), VRSOUT, VSOUT, VTIMSV (XVTMSV), WHICH (XWHICH), XACTEQ, XAVRGE (XAVRGR), XDIV (XDIVR), XDVIDE (XDVIDR), XFIXM, XLIMIT, XLOCV, XOOZE, XREMAV, XSPECT, XSQDFR (XSQDEV), XSQRUT, XVDVBV (XVDRBV).

WIGGINS, RALPH A.
ARCTAN, CLKON, CMPRA (CMPRFL, XCMPRA), COSIS1, CPYFL2, CROSS, CROST, CRSVM, CSOUT, DOTJ, DOTP, FDOT (FDOTR), FIRE2, FRAME (FRAM90), FT24II, GENHOL, GETX (IGETX), GNHOL2, HST2, HST309, HST390, INTHOL, IPLYEV, LISTNG, LOC, LSHFT (XLSHFT), LSLINE, LSSS1, MATINV, MATML1, MATML3, MATRA1, MDOT, MDOT3, MFACT, MIFLS, MIPLS, MISS, MOUT, MOVREV, MRVRS, NMZMG1, NRMVEC, ONLINE ((STH), (STHD), (STHM)), PACDAT, PLANSP, POLYSN, QXCOR1, RDATA, REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), REVERS, RLSPR, RLSPR2, RLSSR, RND (RNDDN, RNDUP), ROAR2, RSKIP, SEARCH, SPCOR2, SRCH1, WLLSFP, WRTDAT, XLCOMN.

CLAERBOUT, J.F., AND WIGGINS, R.A.
CNVLV2, POLYDV.

GALBRAITH, J.N., AND WIGGINS, R.A.
DADECK.

MIT LINCOLN LAB, MODIFIED BY GALBRAITH, J.N.
DISPLA (DSPL90).

WIGGINS, R.A., AND SIMPSON, S.M.
MATRA, SIZEUP (SIZUPL).

SIMPSON, S.M., AND GALBRAITH, J.N.
NOINT1 (NOINT2).

WIGGINS, R.A., AND CLARK, J.
ROTAT1.

SIMPSON, S.M., AND WIGGINS, R.A.
SHFTR2, TIMA2B, TIMSUB (INTMSB).

OLSZTYN, J.T., MODIFIED BY NEILL, A.M., AND BY WIGGINS, R.A.
SIMEQ (DETRM).

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

```
**********************
* LANGUAGE COMMENTS *
**********************

  FORTRAN
     CARIGE,  CHISQR,   CLKON,  CNTRDB,  CNTROW,  COLABL,  CONTUR,  CONVLV,
     COSIS1,   CROSS,   CROST,   CRSVM,  DADECK,    DOTP,   FIRE2,  FMTOUT,
     FRQCT1,  FT24II,  GETHOL,  GETRD1,  GNFLT1,   GRAPH,  GRAPHX,   GRUP2,
     INDATA,  IPLYEV,  IXCARG,  KIINT1,  LINTR1,  LISTNG,  LSLINE,   LSSS1,
     MATINV,  MATML3,    MDOT,   MDOT3,  MEMUSE,   MFACT,   MIFLS,   MIPLS,
       MISS,    MOUT,  MOUTAI,   MRVRS,  MSCON1,   MULK2,  MULLER,  MVINAV,
     MVSQAV,  MXRARE,  NRMVEC,  NXALRM,  OUDATA,  PLANSP,  PLOTVS,  PLTVS1,
     PLYSYN,  POKCT1,  POLYDV,  POLYEV,  POLYSN,  PRBFIT,   PROB2,   PSQRT,
     PWMLIV,  QACORR,  QCNVLV,  QFURRY  (QIFURY), QIFURY  (QFURRY), QINTR1,
     QXCORR,  QXCOR1,   RDATA,   RLSPR,  RLSPR2,   RLSSR,   ROAR2,  SETINO,
      SETK2,  SETKS2,  SHUFFL,  SMPSON,  SPCOR2,   SRCH1,  TRMINC,  VECOUT,
       VOUT,     WAC,  WLLSFP,  XSPECT.

  FAP BY OPTION
     ABSVAL,   ADANL  ( ADANX,   XDANL,   XDANX), AMPHZ  (  REIM), ARBCOL,
     ARCTAN,  ASPEC2,  AVRAGE,  BLKSUM,   BOOST  (DPRESS,  XBOOST,  XDPRSS),
     CHPRTS  (RVPRTS), CHSIGN,  CMPARV  (CMPARL),  CMPRA  (CMPRFL,  XCMPRA),
     COLAPS,  CNVLV2,    COSP  (COSISP,    SISP), COSTBL  (COSTBX,  SINTBL,
     SINTBX), CUFIT1,   DELTA  ( STEPC,   STEPL,   STEPR,  XDELTA,  XSTEPC,
     XSTEPL,  XSTEPR), DERIVA  (IDERIV), DIFPRS  (INTSUM,  XDFPRS), DIVIDE,
       DOTJ,  DSPFMT,   DUBLX  ( DUBLL,   HALVL,   HALVX), EXCHVS,  EXPAND,
     FACTOR,  FASCN1,  FASCUB,  FASTRK,    FDOT  ( FDOTR),    FIXV  ( FIXVR),
     FLOATM,  FLOATV,  FRQCT2,    FT24  (FT24II), IDERIV  (DERIVA), IFNCTN,
     IINTGR  (INTGRA),  INDEX  (CHUSET,  SETAPT,  SETEST,  VINDEX), INTHOL,
     INTGRA  (IINTGR), INTOPR,  INTSUM  (DIFPRS,  XNTSUM), KOLAPS,  MATML1,
      MATRA,  MATRA1,   MAXSN  ( MAXAB,   MINAB,   MINSN), MAXSNM  (MAXABM,
     MINABM,  MINSNM), MONOCK,    MOVE,  MOVREV,  MPSEQ1,  MULPLY,  MUVADD,
     MVBLOK,  MVNSUM,  MVNTIN  (MVNTNA), NMZMG1,  NOINT1  (NOINT2), NURINC,
     ONLINE  ( (STH),   (STHD),  (STHM)),   PAKN  (UNPAKN),  POWER  (SMPRDV),
     QUFIT1,  REFLEC  (XRFLEC),  REMAV,   REVER,  REVERS,  RMSDEV  (RMSDAV),
        RND  ( RNDDN,  RNDUP),    RNDV  (RNDVDN,  RNDVUP), ROTAT1,    SAME
    ( XSAME), SCPSCL,  SEARCH,  SEQSAC  (NEXCOS,  NEXSIN),  SETKV,  SETLIN
    (XSTLIN),   SIFT,   SIMEQ  ( DETRM), SIZEUP  (SIZUPL),  SPLIT  ( REFIT),
     SQRDFR  (SQRDEV), SQRMLI,  SQROOT,  SQRSUM  (XSQSUM), SQUARE  (XSQUAR),
        STZ,     SUM  (  XSUM), SUMDFR  (SUMDEV,  XSMDEV,  XSMDFR),  TAMVL
    ( TAMVR),  TINGL  (TINGLA),  VDOTV,  VDVBYV,  VPLUSV  (VMNUSV,  XVMNSV,
     XVPLSV), VTIMSV  (XVTMSV),  WHICH  (XWHICH), WRTDAT,  XAVRGE  (XAVRGR),
     XDVIDE  (XDVIDR), XREMAV,  XSQDFR  (XSQDEV), XSQRUT,  XVDVBV  (XVDRBV).

  FAP NECESSARILY
       ADDK  ( ADDKS,    DIVK,   DIVKS,    MULK,   MULK2,    SUBK,   SUBKS,
      XADDK,  XADDKS,   XDIVK,  XDIVKS,   XDVRK,  XDVRKS,   XMULK,  XMULKS,
      XSUBK,  XSUBKS), CHOOSE,  CLOCK1,  CMPARP  (CMPARS), CPYFL2,   CSOUT,
     CVSOUT,  DISPLA  (DSPL90), DSPL90,  FAPSUM,  FNDFMT,   FRAME  (FRAM90),
     FRAM90,   FSKIP,  FXDATA  (FLDATA), GENHOL,    GETX  ( IGETX), GNHOL2,
      HLADJ  ( HRADJ), HSTPLT  (  HST2,  HST309,  HST390),   HST2,  HST309,
     HST390,  HVTOIV  (IVTOHV), ITOMLI,  IVTOHV  (HVTOIV), LIMITS,    LINE
    (LINE90), LINE90,   LINEH  (LINH90), LINH90,   LINEV  (LINV90), LINV90,
        LOC,  LOCATE  (    ARG,   CALL,   CALL2,  RETURN,  SETSBV,   SETUP,
      STORE,   WHERE,    XARG,  XINDEX,   XNAME,  XNARGS),  LSHFT  (XLSHFT),
     MLISCL,  MLI2A6,  MOVECS,    NTHA  ( XNTHA), PACDAT,  PLURNS,  PROCOR
```

(CONTINUED NEXT PAGE)

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

(FASCOR, FASCR1, FASEPC, FASEP1), REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), RPLFMT, RSKIP, SETK (SETKS, SETVEC), SETKP (SETVCP), SETKVS, SETLNS, SEVRAL (PLURAL), SHFTR1, SHFTR2, STZS, SWITCH, TIMA2B, TIMSUB (INTMSB), UNPAKN (PAKN), VARARG, VRSOUT, VSOUT, XACTEQ, XDIV (XDIVR), XFIXM, XLCOMN, XLIMIT, XLOCV, XOOZE, ZEFBCD (ZEFBIN).

SUBROUTINE SUBPROGRAM

ABSVAL, ADANL (ADANX, XDANL, XDANX), ADDK (ADDKS, DIVK, DIVKS, MULK, MULK2, SUBK, SUBKS, XADDK, XADDKS, XDIVK, XDIVKS, XDVRK, XDVRKS, XMULK, XMULKS, XSUBK, XSUBKS), AMPHZ (REIM), ARBCOL, ASPECT, ASPEC2, AVRAGE, BLKSUM, BOOST (DPRESS, XBOOST, XDPRSS), CARIGE, CHISQR, CHOOSE, CHPRTS (RVPRTS), CHSIGN, CLKON, CLOCK1, CMPARP (CMPARS), CMPARV (CMPARL), CNTRDB, CNTRCW, COLABL, COLAPS, CONTUR, CONVLV, CNVLV2, COSIS1, COSP (COSISP, SISP), COSTBL (COSTBX, SINTBL, SINTBX), CPYFL2, CROSS, CROST, CRSVM, CSOUT, CUFIT1, CVSOUT, DADECK, DERIVA (IDERIV), DIFPRS (INTSUM, XDFPRS), DISPLA (DSPL90), DSPL90, DIVIDE, DOTJ, DOTP, DSPFMT, DUBLX (DUBLL, HALVL, HALVX), EXCHVS, EXPAND, FACTOR, FAPSUM, FASCN1, FASCUB, FASTRK, FDOT (FDOTR), FIRE2, FIXV (FIXVR), FLOATV, FMTOUT, FNDFMT, FRAME (FRAM90), FRAM90, FRQCT1, FRQCT2, FSKIP, FT24 (FT24II), FT24II, FXDATA (FLDATA), GENHOL, GETHOL, GETRD1, GNFLT1, GNHOL2, GRAPH, GRAPHX, GRUP2, HSTPLT (HST2, HST309, HST390), HST2, HST309, HST390, HVTOIV (IVTOHV), IDERIV (DERIVA), IFNCTN, IINTGR (INTGRA), INDATA, INTGRA (IINTGR), INTHOL, INTOPR, INTSUM (DIFPRS, XNTSUM), IPLYEV, ITOMLI, IVTOHV (HVTOIV), IXCARG, KIINT1, KOLAPS, LIMITS, LINE (LINE90), LINE90, LINEH (LINH90), LINH90), LINEV (LINV90), LINV90, LINTR1, LISTNG, LOC, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), LSLINE, LSSS1, MATINV, MATML1, MATML3, MATRA, MATRA1, MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM), MDOT, MDOT3, MEMUSE, MFACT, MIFLS, MIPLS, MISS, MLISCL, MLI2A6, MONOCK, MOUT, MOUTAI, MOVE, MOVECS, MOVREV, MPSEQ1, MRVRS, MSCON1, MULK2, MULLER, MULPLY, MUVADD, MVBLOK, MVINAV, MVNSUM, MVNTIN (MVNTNA), MVSQAV, MXRARE, NMZMG1, NOINT1 (NOINT2), NRMVEC, NURINC, NXALRM, ONLINE ((STH), (STHD), (STHM)), OUDATA, PACDAT, PAKN (UNPAKN), PLANSP, PLOTVS, PLTVS1, PLURNS, PLYSYN, POKCT1, POLYDV, POLYSN, POWER (SMPRDV), PRBFIT, PROB2, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), PSQRT, PWMLIV, QACORR, QCNVLV, QFURRY (QIFURY), QIFURY (QFURRY), QINTR1, QUFIT1, QXCORR, QXCOR1, RDATA, REFLEC (XRFLEC), REMAV, REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), REVER, REVERS, RLSPR, RLSPR2, RLSSR, RMSDEV (RMSDAV), RNDV (RNDVDN, RNDVUP), ROAR2, ROTAT1, RPLFMT, RSKIP, SCPSCL, SEARCH, SEQSAC (NEXCOS, NEXSIN), SETINO, SETK (SETKS, SETVEC), SETK2, SETKP (SETVCP), SETKS2, SETKV, SETKVS, SETLIN (XSTLIN), SETLNS, SEVRAL (PLURAL), SHFTR1, SHFTR2, SHUFFL, SIFT, SIMEQ (DETRM), SIZEUP (SIZUPL), SMPSON, SPCOR2, SPLIT (REFIT), SQRDFR (SQRDEV), SQRMLI, SQROOT, SQRSUM (XSQSUM), SQUARE (XSQUAR), SRCH1, STZ, STZS, SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), TAMVL (TAMVR), TIMA2B, TIMSUB (INTMSB), TINGL (TINGLA), TRMINO, UNPAKN (PAKN), VARARG, VDOTV, VDVBYV, VECOUT, VOUT, VPLUSV (VMNUSV, XVMNSV, XVPLSV), VRSOUT, VSOUT, VTIMSV (XVTMSV), WAC, WLLSFP, WRTDAT, XAVRGE (XAVRGR), XDVIDE (XDVIDR), XLIMIT, XLOCV, XREMAV, XSPECT, XSQDFR (XSQDEV), XSQRUT, XVDVBV (XVDRBV).

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

CLOSED FUNCTION
ARCTAN, CMPRA (CMPRFL, XCMPRA), DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), FLOATM, HLADJ (HRADJ), INDEX (CHUSET, SETAPT, SETEST, VINDEX), LSHFT (XLSHFT), NTHA (XNTHA), RND (RNDDN, RNDUP), SAME (XSAME), SWITCH, WHICH (XWHICH), XACTEQ, XDIV (XDIVR), XFIXM, XLCOMN, XLIMIT, XOOZE, ZEFBCD (ZEFBIN).

FORTRAN FUNCTION
GETX (IGETX).

MAIN PROGRAM
(NO ENTRIES FOR THIS CATEGORY)

```
*********************************
* COMMENTS ON LINKAGE, PROGRAM *
* AFFILIATIONS, AND STORAGE    *
*********************************
```

MULTIPLE ENTRIES
ADANL (ADANX, XDANL, XDANX), ADDK (ADDKS, DIVK, DIVKS, MULK, MULK2, SUBK, SUBKS, XADDK, XADDKS, XDIVK, XDIVKS, XDVRK, XDVRKS, XMULK, XMULKS, XSUBK, XSUBKS), AMPHZ (REIM), BOOST (DPRESS, XBOOST, XDPRSS), CHPRTS (RVPRTS), CMPARP (CMPARS), CMPARV (CMPARL), CMPRA (CMPRFL, XCMPRA), COSP (COSISP, SISP), COSTBL (COSTBX, SINTBL, SINTBX), DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), DIFPRS (INTSUM, XDFPRS), DUBLX (DUBLL, HALVL, HALVX), FDOT (FDOTR), FIXV (FIXVR), FXDATA (FLDATA), GETX (IGETX), HLADJ (HRADJ), INDEX (CHUSET, SETAPT, SETEST, VINDEX), INTSUM (DIFPRS, XNTSUM), LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), LSHFT (XLSHFT), MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM), MULPLY (XMLPLY), MVNTIN (MVNTNA), NOINT1 (NOINT2), NTHA (XNTHA), ONLINE ((STH), (STHD), (STHM)), POWER (SMPRDV), PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), REFLEC (XRFLEC), REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), RMSDEV (RMSDAV), RND (RNDDN, RNDUP), RNDV (RNDVDN, RNDVUP), SAME (XSAME), SEQSAC (NEXCOS, NEXSIN), SETK (SETKS, SETVEC), SETKP (SETVCP), SETLIN (XSTLIN), SEVRAL (PLURAL), SIMEQ (DETRM), SIZEUP (SIZUPL), SPLIT (REFIT), SQRDFR (SQRDEV), SQRSUM (XSQSUM), SQUARE (XSQUAR), SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), TAMVL (TAMVR), TIMSUB (INTMSB), TINGL (TINGLA), VPLUSV (VMNUSV, XVMNSV, XVPLSV), VTIMSV (XVTMSV), WHICH (XWHICH), XAVRGE (XAVRGR), XDIV (XDIVR), XDVIDE (XDVIDR), XSQDFR (XSQDEV), XVDVBV (XVDRBV), ZEFBCD (ZEFBIN).

NO ARGUMENTS
CLKON, DISPLA (DSPL90), FRAME (FRAM90), TIMSUB (INTMSB), REREAD (ENDFIL, EOFSET, (TSH), (TSHM)).

INVOLVES NON-STANDARD INFORMATION EXCHANGE
CLOCK1, DISPLA (DSPL90), DSPL90, GENHOL, INDEX (CHUSET, SETAPT, SETEST, VINDEX), LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), MEMUSE, ONLINE ((STH), (STHD), (STHM)), PLURNS, RDATA, REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), RPLFMT, SEQSAC (NEXCOS, NEXSIN), SEVRAL (PLURAL), TIMA2B, TIMSUB (INTMSB), VARARG, XLCOMN.

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

VARIABLE LENGTH CALLING SEQUENCE

ADDK (ADDKS, DIVK, DIVKS, MULK, MULK2, SUBK, SUBKS, XADDK, XADDKS, XDIVK, XDIVKS, XDVRK, XDVRKS, XMULK, XMULKS, XSUBK, XSUBKS), CHOOSE, CMPARP (CMPARS), CSOUT, CVSOUT, GETX (IGETX), INDATA, LIMITS, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), MOVECS, MULK2, NTHA (XNTHA), OUDATA, PLTVS1, PLURNS, RDATA, SETK (SETKS, SETVEC), SETK2, SETKP (SETVCP), SETKVS, SETLNS, SEVRAL (PLURAL), STZS, VRSOUT, VSOUT, XLOCV.

USES NO SUBROUTINES

ABSVAL, ADDK (ADDKS, DIVK, DIVKS, MULK, MULK2, SUBK, SUBKS, XADDK, XADDKS, XDIVK, XDIVKS, XDVRK, XDVRKS, XMULK, XMULKS, XSUBK, XSUBKS), AVRAGE, BLKSUM, BOOST (DPRESS, XBOOST, XDPRSS), CHISQR, CHOOSE, CHPRTS (RVPRTS), CHSIGN, CLOCK1, CMPARP (CMPARS), CMPARV (CMPARL), CMPRA (CMPRFL, XCMPRA), COLAPS, CONVLV, CNVLV2, COSP (COSISP, SISP), CUFIT1, DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), DERIVA (IDERIV), DIFPRS (INTSUM, XDFPRS), DIVIDE, DOTJ, DSPFMT, DUBLX (DUBLL, HALVX, HALVL), EXCHVS, FAPSUM, FASCN1, FASCUB, FASTRK, FDOT (FDOTR), FIXV (FIXVR), FLOATM, FLOATV, FRAME (FRAM90), FRAM90, FRQCT1, FRQCT2, FT24II, FXDATA (FLDATA), GETX (IGETX), GRUP2, HLADJ (HRADJ), HVTOIV (IVTOHV), IDERIV (DERIVA), IINTGR (INTGRA), INDEX (CHUSET, SETAPT, SETEST, VINDEX), INTGRA (IINTGR), INTOPR, INTSUM (DIFPRS, XNTSUM), ITOMLI, IVTOHV (HVTOIV), KOLAPS, LIMITS, LINE (LINE90), LINE90, LINEH (LINH90), LINH90, LINEV (LINV90), LINV90, LINTR1, LOC, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), LSHFT (XLSHFT), LSLINE, MATML1, MATRA, MATRA1, MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM), MLISCL, MLI2A6, MONOCK, MOVE, MOVREV, MPSEQ1, MSCON1, MULPLY, MUVADD, MVBLOK, MVINAV, MVNSUM, MVNTIN (MVNTNA), MVSQAV, NMZMG1, NTHA (XNTHA), NURINC, PLURNS, POLYEV, PROB2, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QUFIT1, REFLEC (XRFLEC), REMAV, REVER, REVERS, RND (RNDDN, RNDUP), ROTAT1, RPLFMT, SAME (XSAME), SCPSCL, SEARCH, SETK (SETKS, SETVEC), SETKV, SETKVS, SETLIN (XSTLIN), SHFTR1, SHFTR2, SIFT, SIMEQ (DETRM), SIZEUP (SIZUPL), SMPSON, SPLIT (REFIT), SQRDFR (SQRDEV), SQRMLI, SQRSUM (XSQSUM), SQUARE (XSQUAR), STZ, STZS, SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), SWITCH, TAMVL (TAMVR), TIMA2B, TINGL (TINGLA), UNPAKN (PAKN), VARARG, VDOTV, VDVBYV, VPLUSV (VMNUSV, XVMNSV, XVPLSV), VTIMSV (XVTMSV), WAC, WHICH (XWHICH), XACTEQ, XDIV (XDIVR), XFIXM, XLCOMN, XLIMIT, XLOCV, XOOZE, XSQDFR (XSQDEV).

DEPENDS ON NECESSARILY FAP SUBROUTINES

CLKON, CNTRDB, COLABL, CONTUR, CRSVM, DADECK, FMTOUT, GETRD1, GRAPH, GRAPHX, INDATA, INTHOL, LISTNG, MEMUSE, MOUTAI, OUDATA, PAKN, PLANSP, PLOTVS, PLTVS1, PWMLIV, QACORR, QCNVLV, QXCORR, QXCOR1, RDATA, SETINO, SHUFFL, SPCOR2, SRCH1, TRMINO, VECOUT, VOUT, XAVRGE (XAVRGR), XDVIDE (XDVIDR), XREMAV, XVDVBV (XVDRBV).

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

USES ONLY FORTRAN SYSTEM ROUTINES
ADANL (ADANX, XDANL, XDANX), ARCTAN, CARIGE, COSTBL (COSTBX, SINTBL, SINTBX), CPYFL2, DISPLA (DSPL90), FSKIP, GENHOL, GETRD1, GNFLT1, GNHOL2, IPLYEV, IXCARG, MULLER, MXRARE, ONLINE ((STH), (STHD), (STHM)), PACDAT, POWER, PRBFIT, PSQRT, REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), RMSDEV (RMSDAV), RSKIP, SEQSAC (NEXCOS, NEXSIN), SQROOT, WRTDAT, ZEFBCD (ZEFBIN).

LESS THAN 50 REGISTERS
ARCTAN, AVRAGE, BLKSUM, BOOST (DPRESS, XBOOST, XDPRSS), CARIGE, CHOOSE, CHSIGN, CLKON, CMPRA (CMPRFL, XCMPRA), CSOUT, DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), DIFPRS (INTSUM, XDFPRS), DIVIDE, DUBLX (DUBLL, HALVL, HALVX), EXCHVS, FAPSUM, FASTRK, FDOT (FDOTR), FIXV (FIXVR), FLOATM, FLOATV, FRAME (FRAM90), FRAM90, GENHOL, GETX (IGETX), HLADJ (HRADJ), HVTOIV (IVTOHV), IINTGR (INTGRA), INTGRA (IINTGR), INTSUM (DIFPRS, XNTSUM), ITOMLI, IXCARG, LIMITS, LINEH (LINH90), LINH90, LINEV (LINV90), LINV90, LOC, LSHFT (XLSHFT), MATRA1, MLISCL, MONOCK, MOVE, MOVECS, MULPLY, MVBLOK, NMZMG1, NTHA (XNTHA), REFLEC (XRFLEC), REMAV, REVER, REVERS, RND (RNDDN, RNDUP), RNDV (RNDVDN, RNDVUP), ROTAT1, RPLFMT, RSKIP, SAME (XSAME), SCPSCL, SEARCH, SETK (SETKS, SETVEC), SETKP (SETVCP), SETKV, SETKVS, SETLIN (XSTLIN), SETLNS, SIFT, SQRDFR (SQRDEV), SQROOT, SQRSUM (XSQSUM), SQUARE (XSQUAR), STZ, STZS, SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), SWITCH, TINGL (TINGLA), VARARG, VDOTV, VDVBYV, VPLUSV (VMNUSV, XVMNSV, XVPLSV), VRSOUT, VSOUT, VTIMSV (XVTMSV), WHICH (XWHICH), XACTEQ, XAVRGE (XAVRGR), XDIV (XDIVR), XDVIDE (XDVIDR), XFIXM, XLCOMN, XLIMIT, XLOCV, XOOZE, XREMAV, XSQDFR (XSQDEV), XSQRUT, XVDVBV (XVDRBV).

MORE THAN 500 REGISTERS
CNTRDB, CNTROW, CONTUR, COSP (COSISP, SISP), FT24 (FT24II), FT24II, GRAPH, INDATA, LISTNG, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), MIPLS, MULLER, PLANSP, PLTVS1, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QCNVLV, QXCOR1, RDATA, RLSPR2, XSPECT.

NEEDS SCRATCH AREA
ASPECT, CNTRDB, CNTROW, COLABL, CONTUR, COSIS1, CPYFL2, CVSOUT, FACTOR, GRAPH, GRAPHX, LISTNG, MATINV, MIFLS, MOUTAI, PLANSP, PLTVS1, PLYSYN, POLYSN, PRBFIT, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QACORR, QCNVLV, QFURRY (QIFURY), QIFURY (QFURRY), QXCORR, QXCOR1, RDATA, SHUFFL, SIMEQ (DETRM), SPCOR2, VRSOUT, WLLSFP, XSPECT.

USES G FORMAT
CSOUT, RDATA.

****
* EQUIPMENT DEALT WITH *
****

USES SWITCHES
CNTRDB, CONTUR, ONLINE ((STH), (STHD), (STHM)), PLOTVS, PLTVS1, SWITCH.

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

USES KEYS
(NO ENTRIES FOR THIS CATEGORY)

USES ONE TAPE
CARIGE, CNTRDB, COLABL, CONTUR, CSOUT, CVSOUT, FMTOUT, FSKIP, GETRD1, MEMUSE, MOUT, MOUTAI, ONLINE ((STH), (STHD), (STHM)), OUDATA, PACDAT, PLOTVS, PLTVS1, PWMLIV, REREAD (ENDFIL, EOFSET, (TSH), (TSHM)), RSKIP, SETINO, SHUFFL, TRMINO, VECOUT, VOUT, VRSOUT, VSOUT, WRTDAT, ZEFBCD (ZEFBIN).

USES TWO OR MORE TAPES
CPYFL2, DADECK, INDATA, LISTNG, RDATA.

USES SCOPE
DISPLA (DSPL90), FRAME (FRAM90), GRAPH, GRAPHX, HSTPLT (HST2, HST309, HST390), LINE (LINE90), LINEH (LINH90), LINEV (LINV90).

USES INTERVAL TIMER
CLKON, CLOCK1, TIMA2B, TIMSUB (INTMSB).

USES ON-LINE PRINTER
CLKON, CNTRDB, CONTUR, ONLINE ((STH), (STHD), (STHM)), PLOTVS, PLTVS1, PWMLIV.

USES OFF-LINE PRINTER
CARIGE, CNTRDB, COLABL, CONTUR, CSOUT, CVSOUT, FMTOUT, MEMUSE, MOUT, MOUTAI, PLOTVS, PLTVS1, PWMLIV, VECOUT, VOUT, VRSOUT, VSOUT.

709 ONLY
DISPLA (DSPL90), FRAME (FRAM90), LINE (LINE90), LINEH (LINH90), LINEV (LINV90).

7090 AND 7094 ONLY
DSPL90, FRAM90, LINE90, LINH90, LINV90.

```
***************************
* ANTITHETICAL SUBJECTIVE *
* JUDGEMENTS              *
***************************
```

MAJOR
ASPECT, CNTRDB, CNTROW, CONTUR, COSIS1, COSP (SISP, COSISP), CPYFL2, CRSVM, DOTP, FACTOR, FIRE2, GRAPH, IFNCTN, INDATA, LOCATE (ARG, CALL, CALL2, RETURN, SETSBV, SETUP, STORE, WHERE, XARG, XINDEX, XNAME, XNARGS), MATRA, MFACT, MIFLS, MIPLS, MISS, MSCON1, MULLER, MXRARE, OUDATA, PLANSP, PROCCR (FASCOR, FASCR1, FASEPC, FASEP1), QACORR, QCNVLV, QFURRY, QIFURY, QXCORR, QXCOR1, RDATA, RLSPR2, SEVRAL (PLURAL), SIMEQ (DETRM), SIZEUP (SIZUPL), SMPSON, SPCOR2, TIMA2B, TIMSUB (INTMSB), WLLSFP, XSPECT.

MINOR
ABSVAL, AVRAGE, BOOST (DPRESS, XBOOST, XDPRSS), CARIGE, CHSIGN, COLABL, DIVIDE, DUBLX (DUBLL, HALVL, HALVX), FIXV (FIXVR), FLOATM, INDEX (CHUSET, SETAPT, SETEST, VINDEX), IXCARG, MULPLY,

(CONTINUED NEXT PAGE)

PROGRAMS SORTED BY NON-FUNCTIONAL ATTRIBUTES

POWER (SMPRDV), REFLEC (XRFLEC), RMSDEV (RMSDAV), SQRDFR (SQRDEV), SQROOT, SQRSUM (XSQSUM), SQUARE (XSQUAR), SUM (XSUM), SUMDFR (SUMDEV, XSMDEV, XSMDFR), VDOTV, VDVBYV, VPLUSV (VMNUSV, XVMNSV, XVPLSV), VTIMSV (XVTMSV), XAVRGE (XAVRGR), XDIV (XDIVR), XDVIDE (XDVIDR), XREMAV, XSQDFR (XSQDEV), XSQRUT, XVDVBV (XVDRBV).

OFTEN USED

ADANL (ADANX, XDANL, XDANX), AMPHZ (REIM), ASPECT, CNVLV2, CONTUR, COSP (SISP, COSISP), COSTBL (COSTBX, SINTBL, SINTBX), DADECK, FMTOUT, FSKIP, FXDATA (FLDATA), GENHOL, HLADJ (HRADJ), HVTOIV (IVTOHV), INDATA, IVTOHV (HVTOIV), LIMITS, MATRA, MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM), MEMUSE, MOUTAI, MOVE, NTHA (XNTHA), OUDATA, PAKN (UNPAKN), PLOTVS, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), RDATA, REVER, REVERS, RND (RNDDN, RNDUP), RSKIP, SAME (XSAME), SETK (SETKS, SETVEC), SETKP (SETVCP), SETKV, SETKVS, SETLIN (XSTLIN), SIZEUP (SIZUPL), STZ, STZS, SWITCH, TIMSUB (INTMSB), UNPAKN (PAKN), VOUT, VRSOUT, VSOUT, WAC, WHICH (XWHICH), XACTEQ, XLCOMN, XLIMIT, ZEFBCD (ZEFBIN).

SELDOM USED

CMPARP (CMPARS), FASCN1, FASTRK, GETHOL, MXRARE, NXALRM, PWMLIV, REFLEC (XRFLEC), SETK2, SETKS2, SEVRAL (PLURAL), SQRMLI, SQROOT, XAVRGE (XAVRGR), XREMAV, XSQDFR (XSQDEV), XSQRUT, XVDVBV (XVDRBV).

FAST

ABSVAL, ARBCOL, ASPECT, BLKSUM, CHSIGN, CMPARV (CMPARL), COSIS1, COSP (COSISP, SISP), CPYFL2, CUFIT1, DELTA (STEPC, STEPL, STEPR, XDELTA, XSTEPC, XSTEPL, XSTEPR), DERIVA (IDERIV), DUBLX (DUBLL, HALVL, HALVX), EXCHVS, EXPAND, FACTOR, FAPSUM, FASCN1, FASCUB, FASTRK, FDOT (FDOTR), FIRE2, FT24 (FT24II), FT24II, HSTPLT (HST2, HST309, HST390), HST2, HST309, HST390, IDERIV (DERIVA), INTOPR, INTSUM (DIFPRS, XNTSUM), ITOMLI, IVTOHV (HVTOIV), LINE (LINE90), LINE90, LINEH (LINH90), LINH90, LINEV (LINV90), LINV90, MATRA, MATRA1, MAXSN (MAXAB, MINAB, MINSN), MAXSNM (MAXABM, MINABM, MINSNM), MIFLS, MIPLS, MISS, MONOCK, MOVE, MOVECS, MOVREV, MULLER, MUVADD, MVNSUM, MVNTIN (MVNTNA), NURINC, PLANSP, PROCOR (FASCOR, FASCR1, FASEPC, FASEP1), QACORR, QCNVLV, QFURRY (QIFURY), QIFURY (QFURRY), QUFIT1, QXCORR, QXCOR1, REVER, REVERS, RLSPR, RLSPR2, RLSSR, ROTAT1, SAME (XSAME), SEQSAC (NEXCOS, NEXSIN), SIFT, SIZEUP (SIZUPL), SPCOR2, STZ, STZS, TAMVL (TAMVR), TIMA2B, TIMSUB (INTMSB), TINGL (TINGLA), UNPAKN (PAKN), WHICH (XWHICH), WLLSFP, XACTEQ, XOOZE, XSPECT.

SLOW

DSPFMT, GNFLT1, MLI2A6, MULK2, PWMLIV, SRCH1.

4

Annotated Calling Sequences

For the working programmer the listings of this section, to which we apply the term annotated calling sequences, and the program digests of the next section have proved to be the most valuable condensed documentation we have evolved. Both of these forms are designed for rapid access to critical program details once an individual has obtained general knowledge of a program's function from a study of the complete listing as given in Section 10.

The annotated calling sequences consist of documentation alphabetically ordered by names of all entry points, with no distinction made between principal and secondary entries. Moreover we have not found it necessary to distinguish between programs of identical name, since in all such cases the calling sequences and functional properties are practically, if not identically, the same.

For a given entry the annotated calling sequence has four parts:

1. a short title,
2. the entry name,
3. a parenthetical list of symbols for the arguments of the entry,
4. an indicator of subprogram type (subroutine subprogram, closed function, or FORTRAN function.

Parts 1 and 3 are not necessarily identical to their counterparts in Section 10. This may be slightly confusing. The titles chosen here emphasize the mnemonic significance of the entry name and provide some stylistic uniformity. The symbol lists used to represent the calling sequences have been carefully chosen to convey in six or fewer characters maximum information about the nature of the arguments. These choices have been made within a fairly uniform notational framework in order to offset the parochial fashion in which individual authors assign argument names. A glossary of commonly used names and combining forms appears below. The meanings of many of the more specialized names can be worked out in the context of the title. For example, in

SRCH1(I1F2B, LV, V, VALUE, INDEX)

whose title is

SEARCH VECTOR FOR VALUE BEGINNING AT EITHER END

the vector searched is V(1. . .LV), the value searched for is VALUE, the index at which correspondence is found is INDEX, and I1F2B is a paremeter specifying search direction to be read "fixed point 1 if search forward, 2 if backward."

COMMONLY USED ARGUMENT NAMES AND COMBINING FORMS

Form	Interpretation
ACOR	Autocorrelation
ARG	Argument
C, C1, C2, . . .	Scalar constants
COSTAB	Cosine table
COSTR	Cosine transform
DAN	Refers to Daniell spectra
DATA	General (floating-point) vector
DUMMY	Argument not referred to
ERR or ERROR	Floating-point error indicator from subroutine (= 0.0 is normal condition)
FMT	Format
FOFIJ	Matrix FOFIJ(I,J)
FREQ, FRQ	Frequency
GZF . . .	Floating-point quantity with value greater than zero if . . .
HOL	Hollerith vector
I . . .	Fixed-point quantity with name . . .
IANS	Fixed-point "answer" from subroutine (IANS = 0 is the normal condition)
IDIMEN	User's dimension of the subscript I
IGZF . . .	Fixed-point quantity with value greater than zero if . . .
ILO, IHI	Low and high indices
ISENSE	Sense switch number
ISPACE	Scratch area
ITAPE, ITP . . .	Logical tape number
ITPIN, or ITPINP	System-input tape number
ITPOUT	System-output tape number
IX . . .	Index . . .
IZF . . .	Contraction of IZIF . . .
IZIF . . .	Fixed-point quantity with value zero if . . .
LAG	Correlation lag
LOCALL	Machine location of a CALL statement
LX, LY, L . . .	Length of vector X, vector Y, or vector . . .
MDAN	Daniell weighting parameter

MFREQ	Number of frequency intervals in the range 0 to π radians
MLI, MLI . . .	Machine-language integers, binary point to right of bit 35
MXACC	Maximum accuracy specification. Input vector will be scaled and fixed to integers of maximum magnitude = MXACC
N . . .	Fixed-point quantity with interpretation of "number of . . ."
NARGS	Number of arguments in a CALL statement
SINTAB	Sine table
SINTR	Sine transform
SPACE	Scratch area
SPECT	Spectrum
STOP	Constant = 777777712345(octal)
SUBRU	Name of subroutine in Hollerith
X, Y	General floating-point variable or vector
XCOR	Cross correlation
ZIF	Floating point quantity with value zero if . . .

Certain program design conventions, which we have adhered to rather closely, assist in the immediate interpretation of the annotated calling sequence. These conventions are:

1. The normal sequence of arguments in any call statement is

   pure input-type arguments (if any),

   followed by

   arguments which are both inputs and outputs (if any),

   followed by

   pure output-type arguments (if any).
2. The use of arguments which are both inputs and outputs is strongly discouraged, most particularly in the case of scalar arguments.
3. Wherever possible the programs are designed so as to permit the user to equate inputs and outputs if he wishes.
4. The use within a subroutine of "true" DIMENSION statements for arguments in a calling sequence is discouraged. Instead the user passes dimension information to the subroutine as explicit arguments.
5. In the case of vectors the normal argument subsequence is

   . . . , vector, length of vector, . . .

   (Note that SRCH1, which was used earlier in an illustration, involves an exception).

6. In the case of matrices the normal argument subsequence is

   . . . , matrix, length of column, length of row, dimension in calling program of column , . . .

   where a column is defined as the matrix values traversed when the first subscript varies over its domain of definition.

7. Error flags from subroutines (IANS, ERR, or ERROR) use the value zero to indicate the normal or no-error condition. There are no exceptions to this convention.

A word of caution is necessary with respect to some of these entries when one is acquiring the programs from the system subroutine library rather than from the input deck. The BSS loader scans only the first ten entries of each binary deck (those on the program card) of the library file in seeking to satisfy its missing-routines table. For programs with more than 10 entries, the eleventh and successive entries are invisible to the loader. Two programs in the present set fall in this category, LOCATE and ADDK. For LOCATE the "invisible" entries are STORE, XNARGS, and XNAME. For ADDK they are SUBKS, MULKS, DIVKS, XADDKS, XSUBKS, XMULKS, XDIVKS, and XDVRKS. However, no difficulty arises when the input deck refers to one or more of these entries provided that it also refers to one or more of the first ten entries (for LOCATE these are LOCATE, WHERE, CALL, CALL2, SETSBV, SETUP, RETURN, XINDEX, ARG, and XARG; for ADDK they are ADDK, SUBK, MULK, DIVK, XADDK, XSUBK, XMULK, XDIVK, XDVRK, and ADDKS) or that such reference develops from the routines picked up by the loader. Otherwise, to obtain execution the user must either modify his input deck to include this type of reference (i.e., a dummy CALL statement) or else add LOCATE or ADDK to his input deck.

AN 'F' IN COLUMN 'COL A' SIGNIFIES A 'FORTRAN FUNCTION' ROUTINE
AN 'F' IN COLUMN 'COL B' SIGNIFIES A 'CLOSED FUNCTION' ROUTINE
OTHERWISE THE ROUTINE IS A 'SUBROUTINE SUBPROGRAM'

PROGRAM TITLE	COL A	NAME	COL B	CALLING SEQUENCE
ABSOLUTE VALUE OF VECTOR		ABSVAL		(ANYVEC,ILO,IHI,ABSVEC,IANS)
MODIFY AUTOCORRELATION FOR DANIELL SPECTRUM, FLOATING		ADANL		(ACOR,MXLAG,MDAN,DACOR)
MODIFY AUTOCORRELATION FOR DANIELL SPECTRUM, FIXED		ADANX		(IACOR,MXLAG,MDAN,IDACOR)
ADD CONSTANT TO VARIABLES		ADDK		(C,X1,X2,...,XN)
ADD CONSTANTS TO VARIABLES		ADDKS		(C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)
AMPLITUDE AND PHASE FROM REAL AND IMAGINARY PARTS		AMPHZ		(REAL,XIMAJ,LRXI,AMP,PHZ,ZIFLIM)
INTERPOLATE ARBITRARY MATRIX COLUMN		ARBCOL		(FOFIJ,LI,LJ,IDIMEN,FJCOL,COL)
ARCTANGENT OF X AND Y COORDINATES		ARCTAN	F	(X,Y)
GET SUBROUTINE ARGUMENT		ARG	F	(LOCALL,NUMARG,IXVECT)
AUTO POWER SPECTRUM FROM CORRELATION		ASPECT		(ACOR,MXLAG,COSTAB,MFREQ,JMIN, JMAX,ZIFXD,SPECT,SPACE,ISCALE, ERR)
AUTO POWER SPECTRUM FROM CORRELATION		ASPEC2		(ACOR,MXLAG,FREQLO,FRQDEL, NFREQS,IERRLO,SPECT,IANS)
AVERAGE OF A VECTOR		AVRAGE		(X,LX,XAVG)
BLOCK SUMMATION WITH DIVISION		BLKSUM		(X,LX,LBLOK,DVSR,XBSMOD,LXBSOD)
BOOST VECTOR BY A CONSTANT		BOOST		(X,LX,XRIZE,XBUSTD)
PROXY CALL STATEMENT		CALL		(SUBRU,IANS,SPACER,ARG1,ARG2, ...,ARGN)
PROXY CALL STATEMENT USING SUBROUTINE VECTOR		CALL2		(SUBRUV,IANS)
OFF-LINE CARRIAGE SPACING		CARIGE		(ITAPE,NSPACE)
CHI-SQUARE FOR EQUI-PROBABLE CASE		CHISQR		(NCOUNT,ICOUNT,ISUMC,CHISQ,IANS)
CHOOSE BETWEEN TWO LISTS OF VARIABLES		CHOOSE		(ZIFRST, X,X1,X2, Y,Y1,Y2, ..., Z,Z1,Z2)
CHANGE EVEN AND ODD PARTS		CHPRTS		(SYM,ANT,N)
CHANGE SIGN BITS OF VECTOR		CHSIGN		(X,LX,XNEG)
CHOOSE ARGUMENT AND SET IT		CHUSET	F	(X,X1,X2,ZIFX1)
REQUEST OPERATOR TO TURN CLOCK ON		CLKON		
REAL TIME FROM CLOCK		CLOCK1		(JOB,TIME)
COMPARE VECTORS LOGICALLY		CMPARL		(X,Y,LXY,IGZFEQ)
COMPARE PAIRS OF VARIABLES		CMPARP		(IANS,X1,Y1,X2,Y2,...,XN,YN)
COMPARE A SET OF VARIABLES		CMPARS		(IANS,X1,X2,...,XN)
COMPARE VECTORS WHERE +0 = -0		CMPARV		(X,Y,LXY,IGZFEQ)
ALGEBRAICALLY COMPARE 2 VARIABLES		CMPRA	F	(X1,X2)
ALGEBRAICALLY COMPARE EXPONENT AND 22 MOST SIGNIFICANT BINARY BITS		CMPRFL	F	(X1,X2)
CONTOUR MATRIX IN DECIBELS		CNTRDB		(ITAPE,ISENSE,GZFAMP,VOFXY,LXV, LYV,LXDIM,VZERO,SPACE,IANS)

```
GENERATE HOLLERITH VECTOR FOR ONE
     OUTPUT ROW OF A CONTOUR PLOT..... CNTROW (VEC,LVEC,FXLO,FXHI,NCOLS,
                                              CHLVLS,NCHRS,DELEVL,VLEVL,
                                              SPACE,PLOTVC,IANS)
LABEL COLUMNS WITH VERTICAL INTEGERS.. COLABL (ITAPE,ICOLLO,NCOLLO,NCOLS,
                                              ISPACE)
COLLAPSE ONE-SIDED VECTOR............. COLAPS (X,LX,ZIFXD,XCOLAP,MCOLAP)
CONTOUR ARBITRARY RECTANGULAR
     SUBREGION OF A MATRIX............ CONTUR (ITAPE,ISENSE,VOFXY,LVX,LVY,
                                              LXDIM,FXLO,FXHI,NCOLS,NCOLLC,
                                              FYLO,FYHI,NROWS,ARGLO,ARGDEL,
                                              ZFAFXD,CHLVLS,NCHRS,DELEVL,
                                              VLEVL,SPACE,IANS)
COMPLETE CONVOLUTION OF TRANSIENTS
     (CONVLV AND CONVLV-II)........... CONVLV (LX,X,LY,Y,CONVXY)
COSINE AND SINE SPECTRUM.............. COSISP (SSX,ASX,SAX,AAX,MXLAG,COSTAB,
                                              SINTAB,MFREQ,JMIN,JMAX,ZIFXD,
                                              COSTR,SINTR)
COSINE AND/OR SINE SPECTRUM WITH
     SPLITTING........................ COSIS1 (I1C2S3,X,LX,COSTAB,SINTAB,
                                              MFREQ,JMIN,JMAX,COSTR,SINTR,
                                              ZIFSTO,SPACE,IANS)
COSINE SPECTRUM....................... COSP   (SSX,ASX,MXLAG,COSTAB,MFREQ,
                                              JMIN,JMAX,ZIFXD,COSTR)
COSINE TABLE GENERATION, FLOATING..... COSTBL (MFREQ,COSTAB)
COSINE TABLE GENERATION, FIXED........ COSTBX (MFREQ,ICOSTB)
COPY FILE - TAPE TO TAPE.............. CPYFL2 (ITPIN,ITPOUT,LRECMX,ZFEOFW,
                                              SPACE,IANS)
CROSSCORRELATION OF TRANSIENTS
     BEGINNING WITH ZERO LAG.......... CROSS  (LX,X,LY,Y,LC,C)
CROSSCORRELATION OF TRANSIENTS........ CROST  (LX,X,LY,Y,ILAG,LC,C)
CROSSCORRELATION OF TRANSIENT VECTORS
     OF MATRICES...................... CRSVM  (NRAC,NCARB,NCBC,LA,AA,LB,BB,
                                              ZIFNTR,ILAG,LC,CC)
VARIABLES OUTPUTED FIVE PER LINE...... CSOUT  (ITAPE,NSPACE,C1,C1NAME,
                                              C2,C2NAME,...,CN,CNNAME)
FIT CUBIC TO FOUR DATA VALUES......... CUFIT1 (FOFX,XLO,DELX,COEFS)
COLUMN VECTORS OUTPUTED BY NORMAL
     OR LITERAL FORMATS............... CVSOUT (ITAPE,NSPACE,FMTHED,FMTLIN,ILO,
                                              IHI,ARGLO,ARGDEL,SPACE,X1,X2,
                                              ...,XN)
COPY DATA-CARDS DECK ONTO OUTPUT
     TAPE............................. DADECK (ITPIN,ITPOUT)
UNIT DELTA FUNCTION................... DELTA F(ARG)
DERIVATIVE OF A VECTOR................ DERIVA (YOFX,LY,DELX,DYDX,YOFX1)
DETERMINANT OF MATRIX................. DETRM  (IDIMEN,IJSIZE,AOFIJ,STHEND,ERR)
DIFFERENCE A VECTOR BY ELEMENT PAIRS.. DIFPRS (X,LX,XPRSDF)
DISPLAY PRINT-TYPE OUTPUT ON SCOPE
     (DISPLA(709) AND DISPLA(7090))... DISPLA ...PRINT FMT,LIST...FMT FORMAT()
DIVIDE VECTOR BY A CONSTANT........... DIVIDE (X,LX,XDVSR,XDVDED)
DIVIDE VARIABLES BY A CONSTANT........ DIVK   (C,X1,X2,...,XN)
DIVIDE VARIABLES BY CONSTANTS......... DIVKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)
PSEUDO DO STATEMENT................... DO     SEVRAL(...,2HDO,NSUBS,I,ILO,
                                                     IHI,...)
DOT PRODUCT WITH JUMPED SUBSCRIPTS.... DOTJ   (LXY,JUMPX,X,JUMPY,Y,DOTXY,
                                              GZFADD,GZFSMD)
```

```
DISPLACED DOT PRODUCT OF
     2 DIMENSIONAL ARRAYS............. DOTP   (NRA,NCA,AA,NRB,NCB,BB,
                                              IRB,ICB,DOT,ORDER)
DEPRESS VECTOR BY A CONSTANT.......... DPRESS (X,LX,XSINK,XLWRD)
VARIABLE ORIGIN DISPLA FORMAT
     GENERATION....................... DSPFMT (CNTHOL,IORGX,IORGY,FMTEND,FMT)
DOUBLE VECTOR ELEMENTS, FLOATING...... DUBLL  (X,LX)
DOUBLE VECTOR ELEMENTS, FIXED......... DUBLX  (IX,LIX)
END-OF-FILE FLAG INDICATOR
     FOR REREAD......................F ENDFIL (ITAPE)
END-OF-FILE SET FOR REREAD............ EOFSET (ZIFTRN,EOF,ITAPE)
EXCHANGE TWO VECTORS.................. EXCHVS (LXY,X,Y)
EXPAND LENGTH OF VECTOR
     BY AN INTEGRAL FACTOR............ EXPAND (X,LX,MLPLYR,XPNDED,LXPNDD)
FACTORIZE ENERGY DENSITY SPECTRUM..... FACTOR (SPECT,LSPECT,LWAVE,WAVE,SPACE)
SUM WITH FAP ACL INSTRUCTION.......... FAPSUM (LX,X,ACLSUM)
FAST SCAN VECTOR FOR EXCESSIVE
    ELEMENT........................... FASCN1 (VECT,ILO,IHI,VALUE,IFIND,IANS)
FAST TRANSIENT CORRELATION............ FASCOR (Y,KMIN,KMAX,CORZER,ERROR)
FAST TRANSIENT CORRELATION SUMMED..... FASCR1 (Y,KMIN,KMAX,CORZER,ERROR)
FAST CUBIC EVALUATION ON UNIFORM
     GRID............................. FASCUB (COEFS,XLO,DELX,NF,FOFX)
FAST EQUI-PRODUCTS CORRELATION........ FASEPC (Y,KMIN,KMAX,CORZER,ERROR)
FAST EQUI-PRODUCTS CORRELATION SUMMED. FASEP1 (Y,KMIN,KMAX,CORZER,ERROR)
FAST TRACK THROUGH VECTOR OF INDICES.. FASTRK (IXVEC,IXSTRT,IXLOOK,MXTRAK,
                                              IANS)
FAST DOT PRODUCT...................... FDOT   (LXY,X,Y,DOTXY)
FAST DOT PRODUCT WITH ONE VECTOR
     REVERSED......................... FDOTR  (LXY,X,Y,DOTXYR)
FILTER BY RECURSION IN 2 DIMENSIONS... FIRE2  (NRA,NCAT,NCAN,AA,NRR,NCR,RR,
                                              NRG,GG,FF,CC)
FIX A FLOATING VECTOR................. FIXV   (X,LX,IX)
FIX A FLOATING VECTOR WITH ROUNDING... FIXVR  (X,LX,IX)
FLOAT AND SCALE MACHINE LANGUAGE
     INTEGERS......................... FLDATA (LX,X,SCALE)
FLOAT A MACHINE LANGUAGE INTEGER...... FLOATMF(INTEGR)
FLOAT A FIXED VECTOR.................. FLOATV (IX,LIX,X)
NORMAL OR LITERAL FORMAT OUTPUTED..... FMTOUT (ITAPE,FMT)
FIND COMMON INDEX OF NORMAL OR
     LITERAL FORMAT................... FNDFMT (FMT,IXCFMT)
ADVANCE FILM FRAME (FRAME(709) AND
     FRAME(7090))..................... FRAME
FREQUENCY COUNT OF INTEGERS........... FRQCT1 (IX,LIX,IXLO,IXHI,ICOUNT,IANS)
FREQUENCY COUNT IN RANGES............. FRQCT2 (X,LX,R,LR,ICOUNT,IANS)
FORWARD (OR BACK) SKIP TAPE FILES..... FSKIP  (ITAPE,NFILES)
FOURIER TRANSFORM 24 POINTS
     (FT24 AND FT24-II)............... FT24   (X,REAL,XIMAJ)
FIX AND SCALE DATA TO MACHINE LANGUAGE
     INTEGERS......................... FXDATA (LX,X,MXDATA,SCALE)
GENERATE OUTPUT-TYPE HOLLERITH........ GENHOL (HOL)...PRINT FMT,LIST...FMT
                                              FORMAT( )
GET HOLLERITH ARGUMENT................ GETHOL (ZIFMUV,HARG,HIFMUV,NCRS,IXCCM,
                                              ICOUNT)
GET RAND RANDOM DIGITS................ GETRD1 (ITAPE,NRD,IRD,IANS)
VARIABLE DEPTH INDEXING..............F GETX   (X,I1,I2,...,IN)
GENERATE SYMMETRIC FILTER............. GNFLT1 (AMSPEC,LSPEC,FLTR,IANS)
```

```
GENERATE OUTPUT-TYPE HOLLERITH........ GNHOL2 (DATA,NDATA,FMT,HOL,NCRS,IXCOM,
                                               INDEX)
SCOPE GRAPH OF VECTOR SETS............ GRAPH  (ISOL,IDOT,LVECS,TITLE,YUNITS,
                                               XUNITS,YTOP,YBOT,XMAX,XMIN,
                                               NOPPP,IPAGE,SPACE)
SUBROUTINE GRAPH EXPANDED............. GRAPHX (ISOL,IDOT,LVECS,TITLE,YUNITS,
                                               XUNITS,YTOP,YBOT,XMAX,XMIN,
                                               NOPPP,IPAGE,SPACE,NFRMSV)
FIND EQUALLY LIKELY GROUPINGS......... GRUP2  (PROB,LPROB,DELX,XLO,XLIMS,
                                               NGRUPS,IANS)
HALVE VECTOR ELEMENTS, FLOATING....... HALVL  (X,LX)
HALVE VECTOR ELEMENTS, FIXED.......... HALVX  (IX,LIX)
HOLLERITH LEFT ADJUST................. HLADJ F(HOL)
HOLLERITH RIGHT ADJUST................ HRADJ F(HOL)
HISTOGRAM PLOT ON SCOPE (HSTPLT,
     HSTPLT-II, HSTPLT-III(709),
     AND HSTPLT-III(7090)))........... HSTPLT (LNY,NY,ORG,NDELX,ZIFSOL,ZIFAXS,
                                               IFRSTB,ISKIPB)
HOLLERITH VECTOR TO INTEGER VECTOR.... HVTOIV (HV,LHV,IV)
INVERSE TO VECTOR DERIVATIVE.......... IDERIV (YOFX1,DYDX,DELX,LY,YOFX)
PSEUDO IF STATEMENT................... IF     SEVRAL(...,2HIF,X,NXNEG,NXZER,
                                                       NXPOS,...)
INVERSION OF MONOTONE FUNCTION........ IFNCTN (YOFX,LYOFX,XFIRST,XLAST,LXOFY,
                                               YLO,YHI,IERRLO,XOFY,IANS)
VARIABLE DEPTH INDEXING..............F IGETX  (IX,I1,I2,...,IN)
INVERSE TO VECTOR INTEGRAL............ IINTGR (YOFX1,YIGRTD,DELX,LY,YOFX,
                                               CIGRTN)
INPUT DATA FROM FILE AS GENERATED BY
     OUDATA........................... INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,
                                               6H AUXL1,AUXL1,6H AUXL2,AUXL2,
                                               ...,6H AUXLN,AUXLN)
INDEX BY UNITY AND COMPARE............ INDEX F(I,ICRTCL)
INTEGRAL OF A VECTOR.................. INTGRA (CIGRTN,YOFX,LY,DELX,YIGRTD,
                                               YOFX1)
INTERPRET HOLLERITH................... INTHOL (NHOL,HOL,FMT,NDATAD,NDATAA,
                                               DATA)
INITIALIZE SUBROUTINE TIMSUB.......... INTMSB
LINEAR INTERPOLATION OPERATOR
     FOR 1,2,3, OR 4 DATA VALUES...... INTOPR (NDATA,XLO,DELX,X,OPER)
INTEGRATED SUMMATION OF A VECTOR...... INTSUM (X,LX,XISUMD)
POLYNOMIAL EVALUATION FOR COMPLEX
     ARGUMENTS........................ IPLYEV (NCOFS,COFS,ZREAL,ZIMAJ,PREAL,
                                               PIMAJ)
INTEGER VECTOR TO MACHINE LANGUAGE
     INTEGER VECTOR................... ITOMLI (IV,LIV,MLIV,IANS)
INTEGER VECTOR TO HOLLERITH VECTOR.... IVTOHV (IV,LHV,HV)
INDEX WITH RESPECT TO COMMON OF
     ARGUMENT......................... IXCARG (ARG,IXCOM)
CHI-SQUARE TAIL INTEGRAL.............. KIINT1 (CHISQ,NDF,PROB,IANS)
COLLAPSE VECTOR ABOUT MIDPOINT........ KOLAPS (XMID,LXHAF,ZIFXD,LCHAF,CMID,
                                               ERR)
CHECK VARIABLES AGAINST THEIR LIMITS.. LIMITS (IANSX1,IANS, X1,X1A,X1B,
                                               X2,X2A,X2B, ..., XN,XNA,XNB)
ARBITRARY LINE ON SCOPE
     (LINE(709) AND LINE(7090))....... LINE   (X1,Y1,X2,Y2)
```

```
HORIZONTAL LINE ON SCOPE
     (LINEH(709) AND LINEH(7090))..... LINEH  (IXLEFT,IYLEFT,IXRITE,IDELX)
VERTICAL LINE ON SCOPE
     (LINEV(709) AND LINEV(7090))..... LINEV  (IXBOT,IYBOT,IYTOP,IDELY)
LINEAR INTERPOLATION.................. LINTR1 (X,XLO,DELX,YTABLE,LTABLE,YOFX)
LISTING OF AUXILIARY INFORMATION OF
     INDATA-OUDATA TAPE............... LISTNG (ITPFIL,ITPOUT,SPACE)
MACHINE LOCATION OF ARGUMENT.......... LOC    (ARG,IADARG)
LOCATE AND NAME A LIST OF SUBROUTINES. LOCATE (SUBRU1,SUBRU2,...,SUBRUN)
                                               CALL SUBR1... CALL SUBRN
LOGICAL SHIFT FUNCTION................ LSHFT F(NSHFT,X)
LEAST SQUARES LINE.................... LSLINE (Y,LY,XMIN,XMAX,CO,C1)
LEAST SQUARES SHAPER BY
     SIDEWAYS ITERATION............... LSSS1  (LPARF,PEO,ACOR,RSIDE,FLTR,
                                               ERRCOV)
MATRIX INVERSE........................ MATINV (LSQM,SQM,SQMINV,SPACE,ERR)
MATRIX MULTIPLY, SQUARE TIMES SQUARE.. MATML1 (LSQM,SQMA,SQMB,SQMAXB,NZFADD)
MATRIX MULTIPLY....................... MATML3 (N,M,L,ANBYM,BMBYL,NZFBTR,CNBYL,
                                               GZFADD)
TIGHT-PACKED MATRIX TRANSPOSE......... MATRA  (MATRX,NROWS,NCOLS,MATRXT)
SQUARE MATRIX TRANSPOSE............... MATRA1 (LSQM,SQM)
MAXIMUM ABSOLUTE VALUE OF VECTOR...... MAXAB  (LX,X,XMAXAB,INDEX)
MAXIMUM ABSOLUTE VALUE MATRIX
     ELEMENT.......................... MAXABM (FOFIJ,LI,LJ,IDIMEN,FMAXAB,
                                               IMAXAB,JMAXAB)
MAXIMUM SIGNED VALUE OF VECTOR........ MAXSN  (LX,X,XMAXSN,INDEX)
MAXIMUM SIGNED VALUE MATRIX ELEMENT... MAXSNM (FOFIJ,LI,LJ,IDIMEN,FMAXSN,
                                               IMAXSN,JMAXSN)
DOT PRODUCT OF VECTORS OF SQUARE
     MATRICES......................... MDOT   (NRCAB,LAB,AA,BB,DOT,MIFREV)
DOT PRODUCT OF VECTORS OF MATRICES.... MDOT3  (NRAD,NCARB,NCBD,LAB,AA,BB,
                                               ZIFNTR,DOT,MIFREV)
OUTPUT MEMORY USAGE DATA.............. MEMUSE (ITPOUT)
FACTORIZE A NON-SINGULAR MATRIX....... MFACT  (LSQM,SQM,SQMFAC)
MULTI-INPUT FILTER BY LEAST SQUARES... MIFLS  (NRC,LL,BB,RR,GG,FF,C)
MINIMUM ABSOLUTE VALUE OF VECTOR...... MINAB  (LX,X,XMINAB,INDEX)
MINIMUM ABSOLUTE VALUE MATRIX
     ELEMENT.......................... MINABM (FOFIJ,LI,LJ,IDIMEN,FMINAB,
                                               IMINAB,JMINAB)
MINIMUM SIGNED VALUE OF VECTOR........ MINSN  (LX,X,XMINSN,INDEX)
MINIMUM SIGNED VALUE MATRIX ELEMENT... MINSNM (FOFIJ,LI,LJ,IDIMEN,FMINSN,
                                               IMINSN,JMINSN)
MULTI-INPUT PREDICTOR-LEAST SQUARES... MIPLS  (NRC,LL,AA,BB,RR,C,ERR)
MULTI-INPUT SIDEWARDS ITERATION....... MISS   (NRC,LL,AA,BB,RR,GG,FF,C)
MACHINE LANGUAGE INTEGER VECTOR
     SCALING.......................... MLISCL (MLIV,LMLIV,ISCALE,MLIVSC,IANS)
MACHINE LANGUAGE INTEGER CONVERTED TO
     FORMAT(2A6)...................... MLI2A6 (MLI,MLIHOL,NCRS)
CHECK VECTOR FOR MONOTONE BEHAVIOUR... MONOCK (X,LX,ZFNDCR,IANSNG,IANS)
MATRIX OUTPUT IN G FORMAT............. MOUT   (ITAPE,NSPACE,X,XNAME,NRX,NCX,
                                               LX)
MATRIX PRINTED OUT AS INTEGERS........ MOUTAI (ITAPE,NSPACE,FOFIJ,FNAME,LI,LJ,
                                               IDIMEN,NDIGS,SCALE,SPACE)
MOVE VECTOR ANYWHERE.................. MOVE   (LX,X,Y)
MOVE VECTORS ANYWHERE................. MOVECS (L1,X1,Y1,L2,X2,Y2,...,LN,XN,YN)
```

```
MOVE, REVERSE, SPREAD, OR CHANGE
     SIGN OF VECTOR.................... MOVREV (LXY,IX,X,IYMIFR,Y,SIGN)
MAP FLOATING SEQUENCE TO INTEGERS..... MPSEQ1 (X,LX,XLMITS,NLMITS,IX,IXLO,
                                              IANS)
REVERSE VECTOR OF MATRICES............ MRVRS  (NRA,NCA,LA,AA)
MEAN SQUARE CONTINGENCY AND
     DEPENDENCY........................ MSCON1 (IJSIZE,POFIJ,CONTNG,DEPEND,
                                              IANS)
MULTIPLY VARIABLES BY A CONSTANT
     (MULK AND MULK-II)................ MULK   (C,X1,X2,...,XN)
MULTIPLY VARIABLES BY CONSTANTS....... MULKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)
POLYNOMIAL ROOTS BY MULLER'S METHOD... MULLER (COEF,IDEGRE,ROOTR,ROOTI)
MULTIPLY VECTOR BY A CONSTANT......... MULPLY (X,LX,XMLPLR,XMLPLD)
MOVING ADDITION OF FIXED VECTOR....... MUVADD (IV,ILO,IHI,LADD,MUVSUM,NSUMS,
                                              IANS)
MOVE A BLOCK.......................... MVBLOK (NMOVE,IASORS,IADEST)
MOVING AVERAGE OF VECTOR.............. MVINAV (X,LX,LAVHAF,XAV,IANS)
MOVING SUMMATION WITH DIVISION........ MVNSUM (X,LX,LSUM,DVSR,SUMOVD,LSUMOD)
MOVING TRAPEZOIDAL INTEGRAL........... MVNTIN (X,LX,DEL,LINT,XMI,LXMI)
MOVING ABSOLUTE TRAPEZOIDAL INTEGRAL.. MVNTNA (X,LX,DEL,LINT,XAMI,LXAMI)
MOVING SQUARE AVERAGE OF VECTOR....... MVSQAV (X,LX,LAVHAF,XSQAV,IANS)
MAXIMUM RATIO REGION OF TWO CUMULATIVE
     DISTRIBUTIONS..................... MXRARE (DN,DD,LD,DNFRAC,DDFRAC,MNREWI,
                                              RAMX,ILO,IHI,IANS)
NEXT COSINE VALUE..................... NEXCOSF(DUMMY)
NEXT SINE VALUE....................... NEXSINF(DUMMY)
NORMALIZE MAGNITUDES.................. NMZMG1 (LX,X,XMAX,SCALE)
NORMAL INTEGRAL UP TO X............... NOINT1 (X,PROBX)
EQUI-LIKELY RANGES OF NORMAL INTEGRAL. NOINT2 (XMEAN,XSD,NDIV,XDIV,IANS)
NORMALIZE AND BOOST A VECTOR.......... NRMVEC (ZIFRMS,SCALE,X,LX,XMEAN,
                                              XMAX,XNRM)
N-TH ARGUMENT BEYOND FIRST............ NTHA  F(N,A1,A2,...,AN,...)
NEW RANGE AND INCREMENT OF VECTOR..... NURINC (YOFX,LY,XLO,XHI,LYNU,XLONU,
                                              XHINU,IERR1,YOFXNU,IANS)
NEXT ALARM............................ NXALRM (JOB,MLIV,ILO,IHI,LEVEL,LTENSE,
                                              IBEGIN,IEND,ISUM,IANS)
ONLINE DUPLICATION OF OFFLINE OUTPUT.. ONLINE (ISENSE)
OUTPUT DATA AND AUXILIARY INFORMATION
     TO FILE TAPE...................... OUDATA (ITAPE,IRECNO,NOPTS,DATA,MODCOD,
                                              6H AUXL1,LAUXL1,AUXL1,...,
                                              6H AUXLN,LAUXLN,AUXLN)
READ EVERY N-TH WORD
     FROM BINARY TAPE.................. PACDAT (ITAPE,NWORDS,IFSTWD,IFOLD,
                                              DATA,LDATA,IANS)
PACK A FLOATING VECTOR, N WORDS PER
     REGISTER.......................... PAKN   (NWPR,LDATA,DATA,SCALE)
FAST TWO-DIMENSIONAL
     SPATIAL SPECTRUM.................. PLANSP (JOB,NRA,NCA,AA,MRS,JMAXR,MCS,
                                              JMAXC,SPT,SPACE1,SPACE2,IANS3)
PRINTER PLOT OF VECTORS, GENERAL...... PLOTVS (ITAPE,ISENSE,LOCYV,YSMBV,LYV,
                                              IXSTRV,NY,ARGLO,ARGDEL,ZFAFXD,
                                              FMTARG,NCOLS,YBOT,YTOP,HLINV,
                                              HLSMBV,NHL)
```

```
PRINTER PLOT OF VECTORS
     WITH AUTOMATIC SCALING........... PLTVS1 (ITAPE,ISENSE,ARGLO,ARGDEL,
                                               ZFAFXD,NCOLS,ZFZERS,RMSSEP,S,
                                               LX,ZFLIST,VMATRX,IDIMEN,NX)
                                                       OR
                                              (ITAPE,ISENSE,ARGLO,ARGDEL,
                                               ZFAFXD,NCOLS,ZFZERS,RMSSEP,S,
                                               LX,0.0,X1,X2,...,XN)
PLURALIZE A SUBROUTINE................ PLURAL (SUBROU,A1,A2,...,AN,
                                               B1,B2,...,BN,......)
PLURALIZE THE NEXT SUBROUTINE......... PLURNS (A1,A2,...,AN,B1,B2,...,BN,
                                               ......,Z1,Z2,...,ZN)
                                                       OR
                                              (A1,A2,...,ANA,STOP,B1,B2,...,
                                               BNB,STOP,.......,Z1,Z2,...,ZNZ)
POLYNOMIAL SYNTHESIZED FROM ITS
     COMPLEX ROOTS.................... PLYSYN (SCALES,RADII,DGREES,NROOTS,
                                               PLYCOS,NCOFS,SPACE)
POKER COUNT OF DIGIT SEQUENCE......... POKCT1 (IX,NHANDS,ICOUNT,IANS)
POLYNOMIAL DIVISION................... POLYDV (LDVSR,DVSR,LDVDD,DVDD,LQUOT,
                                               QUOT)
REAL POLYNOMIAL EVALUATION............ POLYEV (NCOFS,COFS,X,POFX)
POLYNOMIAL SYNTHESIS FROM REAL AND
     COMPLEX ROOTS.................... POLYSN (SCALE,NOZ,ZRE,ZIM,ZIFCOM,
                                               ZIFCNJ,LPOLY,POLY,SPACE)
RAISE VECTOR TO POSITIVE OR
     NEGATIVE INTEGRAL POWER.......... POWER  (X,LX,N,X2NTH)
PROBABILITY CURVE FITTED TO MOMENTS... PRBFIT (NMOMS,XMOMS,LX,X,POFX,SPACE,
                                               IANS)
SECOND ORDER PROBABILITY.............. PROB2  (IX,LIX,LAG,ICOUNT,PROB,IXHI,
                                               IANS)
WRITE PROGRAM FOR CORRELATION......... PROCOR (X,LX,MAXX,PROG1,PROG2,ERR)
POLYNOMIAL SQUARE ROOT................ PSQRT  (NCOFS,COFS,NCSQRR,CSQRR)
PRINT OR WRITE OUT A MACHINE
     LANGUAGE INTEGER VECTOR.......... PWMLIV (NWPL,ITAPE,MLIV,LMLIV,IANS)
QUICK AUTOCORRELATION................. QACORR (X,LX,MXACC,MXLAG,SPACE,ACOR,
                                               IANS)
QUICK CONVOLUTION..................... QCNVLV (X,LX,Y,LY,MXACC,LC,SPACE,C,
                                               IANS)
QUICK FOURIER TRANSFORM............... QFURRY (X,LX,IXZER,MFREQ,JMIN,JMAX,
                                               SPACE,FTREAL,FTIMAJ,IANS)
QUICK INVERSE FOURIER TRANSFORM....... QIFURY (FTREAL,FTIMAJ,MFREQ,LX,IXZER,
                                               SPACE,X,IANS)
QUADRATIC INTERPOLATION IN TABLE...... QINTR1 (X,XLO,DELX,TABLE,NTABLE,YOFX)
FIT QUADRATIC TO THREE DATA POINTS.... QUFIT1 (FOFX,XLO,DELX,COEFS)
QUICK CROSS CORRELATION............... QXCORR (X,Y,LXY,MXACC,MXLAG,SPACE,XCOR,
                                               IANS)
QUICK CROSSCORRELATION OF MLI VECTORS
     WITH VARIABLE LIMITS............. QXCOR1 (LX,X,LY,Y,MXACC,ILAG,NLAGS,
                                               CORR,ZIFSTO,LSPACE,SPACE,IANS)
READ DATA IN VARIABLE FORMAT.......... RDATA  (ITAPE,ITPCPY,IANS,SPACE,
                                               X1NAME,X1, X2NAME,X2, ... )
REFIT EVEN AND ODD PARTS.............. REFIT  (X,LX,ZIFXD,SYM,ANT)
REFLECT A VECTOR THROUGH A CONSTANT... REFLEC (X,LX,XMIROR,XIMAGE)
REAL AND IMAGINARY PARTS FROM
     AMPLITUDE AND PHASE.............. REIM   (AMP,PHZ,LAMPHZ,REAL,XIMAJ)
```

```
REMOVE AVERAGE OF A VECTOR ............ REMAV  (X,LX,XAVG,XNULD)
REREAD BCD RECORDS..................... REREAD
RETURN TO CALLING PROGRAM.............. RETURN (LOCALL,XR1,XR2)
REVERSE A VECTOR....................... REVER  (X,LX,XREVD)
REVERSE A VECTOR....................... REVERS (LX,X)
REALIZABLE LEAST SQUARES PREDICTION
     ERROR OPERATOR BY RECURSION...... RLSPR  (LPA,PEO,ACOR,ERRCOV)
REALIZABLE LEAST-SQUARES PREDICTOR BY
     RECURSION - 2-DIMENSIONAL........ RLSPR2 (NRA,NCAT,NCAN,AA,NRR,NCR,RR,
                                               CC,IANS)
REALIZABLE LEAST SQUARES SHAPER
     BY RECURSION..................... RLSSR  (LPARF,PEO,ACOR,RSIDE,FLTR,
                                               ERRCOV)
RMS DEVIATION FROM AVERAGE............ RMSDAV (X,LX,XAVG,RMSXMA)
RMS DEVIATION FROM BASE VALUE......... RMSDEV (X,LX,XBASE,RMSXMB)
ROUND FLOATING NUMBER................. RND   F(X)
ROUND A FLOATING NUMBER DOWN.......... RNDDN F(X)
ROUND A FLOATING NUMBER UP............ RNDUP F(X)
ROUND A FLOATING VECTOR............... RNDV   (X,LX,XR)
ROUND DOWN A FLOATING VECTOR.......... RNDVDN (X,LX,XR)
ROUND UP A FLOATING VECTOR............ RNDVUP (X,LX,XR)
ROTATE CENTRO-SYMMETRIC OR ANTI-
     SYMMETRIC 2-DIMENSIONAL ARRAY.... ROAR2  (I1SM1A,XA,N,M,XRA)
ROTATE A VECTOR....................... ROTAT1 (X,LX,NUP,ROTX)
REPLACE FORMAT IN NEXT INPUT OR
     OUTPUT STATEMENT................. RPLFMT (FMT,FMTNEW)
RECORD SKIPPING ON TAPE............... RSKIP  (NTAPE,NRECS,EOF)
REVERSE EVEN AND ODD PARTS............ RVPRTS (SYM,ANT,N)
SAME OUTPUT AS INPUT.................. SAME  F(IX)
SCOPE SCALING OF DATA................. SCPSCL (DATA,LDATA,YTOP,YBOT,CONVK,
                                               CONVL)
SEARCH VECTOR FOR VALUE............... SEARCH (LX,X,XWANT,INDEX)
INITIALIZE FOR SEQUENTIAL
     SINE AND COSINE VALUES........... SEQSAC (ARGLO,ARGDEL)
SET FIRST ARGUMENT AND PLACE
     THIRD IN ACCUMULATOR............. SETAPTF(X,XNEW,FVALUE)
SET ARGUMENT AND TEST ITS SIZE........ SETESTF(X,XNEW,XCRTCL)
SET UP AN INDATA-OUDATA TAPE FOR
     RECEIVING ADDITIONAL RECORDS..... SETINO (ITAPE,ZIFNEW,NRECS,IERR)
SET VARIABLES TO A CONSTANT
     (SETK AND SETK-II)............... SETK   (C,X1,X2,...,XN)
SUBROUTINE SETK PLURALIZED............ SETKP  (C1,X11,X12,...,X1N1,STOP,
                                               C2,X21,X22,...,X2N2,STOP,
                                               ......,CM,XM1,XM2,...,XMNM)
SET VARIABLES TO CONSTANTS
     (SETKS AND SETKS-II)............. SETKS  (C1,X1,C2,X2,...,CN,XN)
SET A CONSTANT VECTOR................. SETKV  (C,LX,X)
SET CONSTANT VECTORS.................. SETKVS (C1,L1,X1,C2,L2,X2,...,CN,LN,XN)
SET LINEAR VECTOR..................... SETLIN (BASE,DELTA,LX,X)
SET LINEAR VECTORS.................... SETLNS (BASE1,DELTA1,LX1,X1,BASE2,
                                               DELTA2,LX2,X2,...,BASEN,
                                               DELTAN,LXN,XN)
SET SUBROUTINE VECTOR................. SETSBV (SUBRU,SUBRUV,ARG1,ARG2,...,
                                               ARGN)
SET UP SUBROUTINE LINKAGE............. SETUP  (LOCALL,NARGS,XR1,XR2)
```

```
SUBROUTINE SETVEC PLURALIZED.......... SETVCP (X1,C11,C12,...,C1N1,STOP,
                                              X2,C21,C22,...,C2N2,STOP,
                                              ......,XM,CM1,CM2,...,CMNM)
SET VECTOR FROM LIST.................. SETVEC (X,C1,C2,...,CN)
OPERATE SEVERAL SUBROUTINES........... SEVRAL (SUBRUA,A1,A2,...,ANA,
                                              SUBRUB,B1,B2,...,BNB,......)
SHIFT VECTOR ELEMENTS RIGHT
      ARITHMETICALLY.................. SHFTR1 (NSHFTR,IX,LIX,IXSH,IANS)
SHIFT VECTOR ELEMENTS RIGHT LOGICALLY. SHFTR2 (NSHFTR,IX,LIX,IXSH,IANS)
SHUFFLE THE INTEGERS 1 TO N........... SHUFFL (ITPRD,NITEMS,ISPACE,IXSHUF)
SIFT OUT EQUALLY SPACED VALUES........ SIFT   (X,MESH,LXSFTD,XSFTD)
SOLVE SIMULTANEOUS EQUATIONS
      A(I,J)*X(J,K) = B(I,K).......... SIMEQ  (IDIMEN,IJSIZE,KSIZE,ATHENX,B,
                                              STHEND,SPACE,ERR)
SINE TABLE GENERATION, FLOATING....... SINTBL (MFREQ,SINTAB)
SINE TABLE GENERATION, FIXED.......... SINTBX (MFREQ,ISINTB)
SINE SPECTRUM......................... SISP   (SAX,AAX,MXLAG,SINTAB,MFREQ,
                                              JMIN,JMAX,ZIFXD,SINTR)
MAKE SIZE INDEX OF ALGEBRAIC VECTOR... SIZEUP (X,LX,INDEX)
MAKE SIZE INDEX OF LOGICAL VECTOR..... SIZUPL (X,LX,INDEX)
SUM INTEGRAL POWER OF DEVIATIONS OF
      VECTOR ELEMENTS FROM BASE....... SMPRDV (X,LX,N,XBASE,SXMB2N)
MULTI-PURPOSE SIMPSON'S RULE
      INTEGRAL........................ SMPSON (JOB,X,LX,DELX,XINT,IANS)
FAST 2-DIMENSIONAL SPATIAL
      CROSSCORRELATION................ SPCOR2 (NRX,NCX,XX,NRY,NCY,YY,MXACC,
                                              ILGR,NRZ,ILGC,INC,NCZ,ZZ,
                                              SPACE,IANS)
SPLIT INTO EVEN AND ODD PARTS......... SPLIT  (X,LX,ZIFXD,SYM,ANT)
SUM OF SQUARE DEVIATIONS OF VECTOR
      FROM BASE....................... SQRDEV (X,XBASE,LX,SSQXMB)
SUM OF SQUARED VECTOR DIFFERENCES..... SQRDFR (X,Y,LXY,SSQXMY)
SQUARE MACHINE LANGUAGE INTEGER
      VECTOR.......................... SQRMLI (MLIVEC,ILO,IHI,MLISQR,IANS)
SQUARE ROOT OF A VECTOR............... SQROOT (X,LX,XSQRTD)
SQUARE SUM OF VECTOR ELEMENTS......... SQRSUM (X,LX,SUMSQX)
SQUARE VECTOR ELEMENTS................ SQUARE (X,LX,XSQRD)
SEARCH VECTOR FOR VALUE BEGINNING
      AT EITHER END................... SRCH1  (I1F2B,LV,V,VALUE,INDEX)
UNIT STEP FUNCTION, CENTERED
      BETWEEN PLUS AND MINUS ZERO..... STEPC F(ARG)
UNIT STEP FUNCTION, TO LEFT OF ZERO... STEPL F(ARG)
UNIT STEP FUNCTION, TO RIGHT OF ZERO.. STEPR F(ARG)
STORAGE-TO-TAPE HOLLERITH............. (STH)  -NOT AVAILABLE BY FORTRAN CALLS-
STORAGE-TO-TAPE HOLLERITH DEBUG....... (STHD) -NOT AVAILABLE BY FORTRAN CALLS-
STORAGE-TO-TAPE HOLLERITH MONITOR..... (STHM) -NOT AVAILABLE BY FORTRAN CALLS-
STORE SUBROUTINE ARGUMENT............. STORE  (ARGU,LOCALL,NUMARG,IXVECT)
STORE ZEROES IN A VECTOR.............. STZ    (LX,X)
STORE ZEROES IN A LIST OF VECTORS..... STZS   (LX1,X1,LX2,X2,...,LXN,XN)
SUBTRACT CONSTANT FROM VARIABLES...... SUBK   (C,X1,X2,...,XN)
SUBTRACT CONSTANTS FROM VARIABLES..... SUBKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)
SUM VECTOR ELEMENTS................... SUM    (X,LX,SUMX)
SUM OF DEVIATIONS FROM BASE........... SUMDEV (X,XBASE,LX,SUMXMB)
SUM OF VECTOR DIFFERENCES............. SUMDFR (X,Y,LXY,SUMXMY)
TEST SPECIFIED SENSE SWITCH........... SWITCHF(ISENSE)
TRIANGULAR AVERAGE MOVING LEFT END.... TAMVL  (X,LX,LAVG,AVGL)
```

```
TRIANGULAR AVERAGE MOVING RIGHT END... TAMVR  (X,LX,LAVG,AVGR)
REAL TIME BETWEEN 2 MACHINE LOCATIONS. TIMA2B (LOCA,LOCB,MINACC,SECS)
REAL TIME OF NEXT SUBROUTINE.......... TIMSUB (MINACC,SECS)
DEFINITE TRAPEZOIDAL INTEGRAL......... TINGL  (YOFX,LX,DELX,TING)
DEFINITE TRAPEZOIDAL INTEGRAL
     OF ABSOLUTE VALUE................ TINGLA (YOFX,LX,DELX,TINGA)
TERMINATE AN INDATA-OUDATA TAPE....... TRMINO (ITAPE,NBAKUP)
TAPE-TO-STORAGE HOLLERITH............. (TSH)  -NOT AVAILABLE BY FORTRAN CALLS-
TAPE-TO-STORAGE HOLLERITH MONITOR..... (TSHM) -NOT AVAILABLE BY FORTRAN CALLS-
UNPACK, N WORDS PER REGISTER.......... UNPAKN (NWPR,LUDATA,DATA,SCALE)
SET FOR VARIABLE ARGUMENT COUNT....... VARARG (LOCS)
VECTOR DOTTED WITH VECTOR............. VDOTV  (X,Y,LXY,DVSR,XDYODV)
VECTOR DIVIDED BY VECTOR.............. VDVBYV (X,Y,LXY,XDVBYY)
VECTOR OUTPUTED BY NORMAL OR
     LITERAL FORMAT................... VECOUT (ITAPE,FMT,X,ILO,IHI)
INDEX BY VARIABLE AND COMPARE......... VINDEXF(I,ICRTCL,IJUMP)
VECTOR MINUS VECTOR................... VMNUSV (X,Y,LXY,XMNUSY)
VECTOR OUTPUTED WITH LABEL BY
     NORMAL OR LITERAL FORMAT......... VOUT   (ITAPE,NSPACE,X,XNAME,XFMT,ILO,
                                               IHI)
VECTOR PLUS VECTOR.................... VPLUSV (X,Y,LXY,XPLUSY)
VARIABLES OUTPUTED BY NORMAL
     OR LITERAL FORMAT................ VRSOUT (ITAPE,NSPACE,FMT,SPACE,X1,X2,
                                               ...,XN)
VECTORS OUTPUTED WITH LABELS BY
     NORMAL OR LITERAL FORMATS........ VSOUT  (ITAPE,NSPACE,X1,X1NAME,X1FMT,
                                               ILO1,IHI1,X2,X2NAME,X2FMT,ILO2,
                                               IHI2,...,XN,XNNAME,XNFMT,ILCN,
                                               IHIN)
VECTOR TIMES VECTOR................... VTIMSV (X,Y,LXY,XTIMSY)
WIENER AUTOCORRELATION................ WAC    (LX,X,LACOR,ACOR)
FIND WHERE SUBROUTINE IS.............. WHERE  (SUBRU,IANS,LOC,NARGS)
CHOOSE WHICH OF TWO ARGUMENTS TO USE.. WHICH F(X1,X2,ZIFX1)
WIENER-LEVINSON LEAST-SQUARES FILTER
     OR PREDICTOR..................... WLLSFP (MXLAG,ACOR,RSIDE,LFILTR,FILTR,
                                               AUXSEQ)
WRITE BINARY RECORD ON TAPE........... WRTDAT (ITAPE,DATA,LDATA,IANS)
EXACT EQUALITY TEST INCLUDING
     SIGN BIT......................... XACTEQF(X,Y)
FIXED ADD CONSTANT TO VARIABLES....... XADDK  (IC,IX1,IX2,...,IXN)
FIXED ADD CONSTANTS TO VARIABLES...... XADDKS (IC1,IX1,IY1,IC2,IX2,IY2,...,
                                               ICN,IXN,IYN)
GET FIXED SUBROUTINE ARGUMENT......... XARG  F(LOCALL,NUMARG,IXVECT)
FIXED AVERAGE OF A VECTOR............. XAVRGE (IX,LIX,IXAVG)
FIXED AVERAGE WITH ROUNDING OF A
     VECTOR........................... XAVRGR (IX,LIX,IXAVG)
FIXED BOOST VECTOR BY A CONSTANT...... XBOOST (IX,LIX,IXRIZE,IXBSTD)
ALGEBRAICALLY COMPARE 2 VARIABLES..... XCMPRAF(X1,X2)
MODIFY CROSS CORRELATION FOR DANIELL
     SPECTRUM, FLOATING............... XDANL  (XCORZ,MXLAG,MDAN,DXCORZ)
MODIFY CROSS CORRELATION FOR DANIELL
     SPECTRUM, FIXED.................. XDANX  (IXCORZ,MXLAG,MDAN,DIXCRZ)
FIXED UNIT DELTA FUNCTION............. XDELTAF(ARG)
FIXED DIFFERENCE A VECTOR BY
     ELEMENT PAIRS.................... XDFPRS (IX,LIX,IXPRSD)
FIXED DIVISION........................ XDIV  F(NUMERA,IDENOM)
FIXED DIVIDE VARIABLES BY A CONSTANT.. XDIVK  (IC,IX1,IX2,...,IXN)
```

```
FIXED DIVIDE VARIABLES BY CONSTANTS... XDIVKS (IC1,IX1,IY1,IC2,IX2,IY2,...,
                                              ICN,IXN,IYN)
FIXED DIVISION WITH ROUNDING.......... XDIVR F(NUMERA,IDENOM)
FIXED DEPRESS VECTOR BY A CONSTANT.... XDPRSS (IX,LIX,IXSINK,IXLWRD)
FIXED DIVIDE VECTOR BY A CONSTANT..... XDVIDE (IX,LIX,IXDVSR,IXDVDD)
FIXED DIVIDE VECTOR BY CONSTANT WITH
     ROUNDING......................... XDVIDR (IX,LIX,IXDVSR,IXDVDD)
FIXED DIVIDE AND ROUND VARIABLES
     BY A CONSTANT.................... XDVRK  (IC,IX1,IX2,...,IXN)
FIXED DIVIDE AND ROUND VARIABLES
     BY CONSTANTS..................... XDVRKS (IC1,IX1,IY1,IC2,IX2,IY2,...,
                                              ICN,IXN,IYN)
FIX FLOATING TO MACHINE LANGUAGE
     INTEGER.......................... XFIXM F(ZFTRNC,FLTG)
INDEX WITH RESPECT TO COMMON OF
     SUBROUTINE ARGUMENT.............. XINDEXF(LOCALL,NUMARG)
LENGTH OF COMMON AVAILABLE OR USED.... XLCOMNF(ZIFACT)
FIXED LIMIT CHECKING FUNCTION......... XLIMITF(X,XA,XB)
FIXED VECTOR FROM APPLYING XLOC
     FUNCTION TO LIST OF ARGUMENTS.... XLOCV  (LOCV,X1,X2,...,XN)
LOGICAL SHIFT FUNCTION................ XLSHFTF(NSHFT,IX)
FIXED MULTIPLY VECTOR BY A CONSTANT... XMLPLY (IX,LIX,IXMPLR,IXMPLD)
FIXED MULTIPLY VARIABLES BY A
     CONSTANT......................... XMULK  (IC,IX1,IX2,...,IXN)
FIXED MULTIPLY VARIABLES BY
     CONSTANTS........................ XMULKS (IC1,IX1,IY1,IC2,IX2,IY2,...,
                                              ICN,IXN,IYN)
COMPARE HOLLERITH NAMES............... XNAME F(HNAME1,HNAME2)
FIND NUMBER OF SUBROUTINE ARGUMENTS... XNARGSF(LOCALL)
FIXED N-TH ARGUMENT BEYOND FIRST...... XNTHA F(N,IA1,IA2,...,IAN,...)
FIXED INTEGRATED SUMMATION
     OF A VECTOR...................... XNTSUM (IX,LIX,IXISMD)
FIXED ONE IF ODD, ZERO IF EVEN........ XOOZE F(INTGER)
FIXED REMOVE AVERAGE OF A VECTOR...... XREMAV (IX,LIX,IXAVG,IXNULD)
FIXED REFLECT A VECTOR THROUGH
     A CONSTANT....................... XRFLEC (IX,LIX,IXMIRR,IXIMGE)
SAME OUTPUT AS INPUT.................. XSAME F(X)
FIXED SUM OF DEVIATIONS FROM BASE..... XSMDEV (IX,IXBASE,LIX,ISMXMB)
FIXED SUM OF VECTOR DIFFERENCES....... XSMDFR (IX,IY,LXY,ISMXMY)
CROSS POWER SPECTRUM FROM CROSS
     CORRELATION...................... XSPECT (XCORZ,MXLAG,COSTAB,SINTAB,
                                              MFREQ,JMIN,JMAX,CSPEC,SSPEC,
                                              SPACE,ERR)
FIXED SUM OF SQUARE DEVIATION OF
     VECTOR FROM BASE................. XSQDEV (IX,IXBASE,LIX,ISSXMB)
FIXED SUM OF SQUARED VECTOR
     DIFFERENCES...................... XSQDFR (IX,IY,LXY,ISSXMY)
FIXED SQUARE ROOT OF A VECTOR......... XSQRUT (IX,LIX,IXSQRT)
FIXED SQUARE SUM VECTOR ELEMENTS...... XSQSUM (IX,LIX,ISMSQX)
FIXED SQUARE A VECTOR................. XSQUAR (IX,LIX,IXSQRD)
FIXED UNIT STEP FUNCTION, CENTERED
     BETWEEN PLUS AND MINUS ZERO...... XSTEPCF(ARG)
FIXED UNIT STEP FUNCTION,
     TO LEFT OF ZERO.................. XSTEPLF(ARG)
FIXED UNIT STEP FUNCTION,
     TO RIGHT OF ZERO................. XSTEPRF(ARG)
```

```
FIXED SET LINEAR VECTOR................ XSTLIN (IBASE,IDELTA,LIX,IX)
FIXED SUBTRACT CONSTANT FROM
     VARIABLES......................... XSUBK  (IC,IX1,IX2,...,IXN)
FIXED SUBTRACT CONSTANTS FROM
     VARIABLES......................... XSUBKS (IC1,IX1,IY1,IC2,IX2,IY2,...,
                                               ICN,IXN,IYN)
FIXED SUM VECTOR ELEMENTS.............. XSUM   (IX,LIX,ISUMX)
FIXED VECTOR DIVIDED, WITH ROUNDING,
     BY VECTOR......................... XVDRBV (IX,IY,LXY,IXDRBY)
FIXED VECTOR DIVIDED BY VECTOR......... XVDVBV (IX,IY,LXY,IXDVBY)
FIXED VECTOR MINUS VECTOR.............. XVMNSV (IX,IY,LXY,IXMNSY)
FIXED VECTOR PLUS VECTOR............... XVPLSV (IX,IY,LXY,IXPLSY)
FIXED VECTOR TIMES VECTOR.............. XVTMSV (IX,IY,LXY,IXTMSY)
FIXED CHOOSE WHICH OF
     TWO ARGUMENTS TO USE.............. XWHICHF(IX1,IX2,ZIFIX1)
ZERO IF END-OF-FILE, BCD TAPE.......... ZEFBCDF(ITAPE)
ZERO IF END-OF-FILE, BINARY TAPE....... ZEFBINF(ITAPE)
```

5
Program Digests

Experience has shown that the annotated calling sequences of the previous section supply perhaps 50 to 75 per cent of the information needs of the working programmer once he has become generally familiar with the program set. Practically all of his remaining needs are provided by the digest of the present section.

The "program digests" listed here are highly compact statements of input-output functional specifications, augmented by data on language, storage requirements, entry names, and transfer vectors. They do not include timing information. The ordering is again alphabetic by entry name; identically named entries have separate digests when there are functional differences.

The digests pivot around the calling sequences, and the reader will note that their representations here differ somewhat from those of the previous section. The argument names used in this section are identical to those chosen by the authors as shown in the program listings of Section 10.

It is possible to use these digests as introductory abstracts of the program functions, but the compact notations and numerical details involved make reading difficult. The abstracts given in the program listings are much more suitable for this purpose, even if less convenient for scanning.

```
     AN 'F' PRECEDING THE LEFT PARENTHESIS OF THE CALLING SEQUENCE
     SIGNIFIES A 'CLOSED FUNCTION' ROUTINE.


ABSVAL (ANYVEC,ILO,IHI,ABSVEC,IANS)                          FAP,  50 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS ABSVEC(1...IHI-ILO+1) = MAGNITUDE OF ANYVEC(ILO...IHI).
      EQUIV(ABSVEC,ANYVEC(ILO)) OK.  SETS IANS=0 IF OK, =-1 IF ILO LSTHN 1 OR
      IF IHI LSTHN ILO.

ADANL  (AA,N,M,DAA)                                          FAP, 183 REGISTERS
          OTHER ENTRIES - XDANL,ADANX,XDANX.  TRANSFER VECTOR - SIN.
      SETS DAA(1...N+1) = DA(0...N) WHERE DA(L) = (M/(L*PI))*A(L)*SIN(L*PI/M)
      AND A(0...N) IS FURNISHED IN AA(1...N+1), WHERE N MUST BE GRTHN= 0,
      M GRTHN 0.  EQUIV(DAA,AA) OK.

ADANX  (IAA,N,M,IDAA)                          FAP, SECONDARY ENTRY OF ADANL
      SAME FUNCTION AS ADANL EXCEPT INPUTS, IAA(1...N+1), AND OUTPUTS,
      IDAA(1...N+1), ARE FIXED POINT.

ADDK   (C,X1,X2,...,XN)                                      FAP, 114 REGISTERS
          OTHER ENTRIES - SUBK,MULK,DIVK,XADDK,XSUBK,XMULK,XDIVK,XDVRK,ADDKS,
          SUBKS,MULKS,DIVKS,XADDKS,XSUBKS,XMULKS,XDIVKS,XDVRKS.  NO TRANSFER
          VECTOR.
      SETS  X1=X1+C, X2=X2+C, ..., XN=XN+C.  EQUIV(ANY ARGUMENTS) OK, BUT
      INITIAL VALUE OF  C  IS ALWAYS THE ADDEND.  STRAIGHT RETURN IF N=0.

ADDKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)         FAP, SECONDARY ENTRY OF ADDK
      SETS  Y1=X1+C1,  Y2=X2+C2,  ...,  YN=XN+CN.  EQUIV(ANY TWO ARGUMENTS)
      OK BUT MAY CHANGE INPUTS  CJ  OR  XJ.  PROCESSING IS LEFT TO RIGHT.
      STRAIGHT RETURN IF  N=0.

AMPHZ  (RE,XIM,LR,AMP,PHZ,R)                                 FAP, 149 REGISTERS
          OTHER ENTRY - REIM.  TRANSFER VECTOR - ATAN,SQRT,RND,COS,SIN.
      SETS AMP(1...LR) = AMPLITUDE, PHZ(1...LR) = PHASE IN RADIANS OF REAL,
      IMAGINARY PARTS RE(1...LR), XIM(1...LR).  R=0. GIVES PHZ FROM +PI TO
      -PI, NOT=0. GIVES PHZ CONTINUOUS.  EQUIV(RE,AMP),(XIM,PHZ) OK.

ARBCOL (FOFIJ,LI,LJ,IDIMEN,FJCOL,COL)                       FAP,  129 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - INTOPR.
      SETS  COL(1...LI)  BY CUBIC INTERPOLATION BETWEEN THE FOUR COLUMNS
      FOFIJ(1...LI,K)  K=J-1,J,J+1,J+2  OF THE MATRIX  FOFIJ(1...LI,1...LJ),
      WHERE  J = FJCOL  ROUNDED DOWN TO NEAREST INTEGER, EXCEPT THAT QUADRATIC
      OR LINEAR INTERPOLATION IS EMPLOYED IF NECESSARY TO AVOID USE OF  K
      VALUES  LSTHN  1  OR  GRTHN  LJ.  LI  AND  LJ  MUST EXCEED ZERO,  FJCOL
      MUST BE  GRTHN=  1.0  AND  LSTHN=  FLOATF(LJ+1),  AND CALLER MUST USE
      DIMENSION  FOFIJ(IDIMEN,IGNORED)  WITH  IDIMEN  GRTHN=  LI.  STRAIGHT
      RETURN WITH NO OUTPUT FOR ILLEGAL  LI,  LJ,  IDIMEN,  OR  FJCOL.

ARCTANF(X,Y)                                                FAP,   29 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - ATAN.
      HAS VALUE  ANGLE  (IN RADIANS)  (-3.14159265  LSTHN  ANGLE  LSTHN=
      3.14159265)  CORRESPONDING TO THE POINT  (X,Y) .
```

```
ARG   F(LOCALL,NUMARG,IXVECT)                    FAP, SECONDARY ENTRY OF LOCATE
      GIVES ELEMENT NO. IXVECT OF THE VECTOR WHICH IS ARGUMENT NO. NUMARG
      OF THE CALL STATEMENT AT MACHINE ADDRESS LOCALL.

ASPECT (ACOR,N,COSTAB,M,JMIN,JMAX,TYPE,SPECT,SPACE,        FAP, 278 REGISTERS
        ISCALE,ERR)
           NO OTHER ENTRIES.  TRANSFER VECTOR - COLAPS,COSP,DUBLX,DUBLL,
           SPLIT,RVPRTS.
      SETS SPECT(1...JMAX-JMIN+1) = SP(JMIN...JMAX) WHERE SP(J) = AC(0) +
      2*SUM(FROM I=1 TO N) OF (AC(I)*COS(I*J*PI/M))  WHERE AC(0...N) FURNISHED
      IN ACOR(1...N+1).  M,N EXCEED ZERO, AND 0 LSTHN= JMIN LSTHN JMAX
      LSTHN= M.  TYPE =0. FOR ACOR FXD, NOT=0. FOR ACOR FLTG. SPACE(1...2*M+1)
      IS SCRATCH IN CASE M LSTHN= N.  ISCALE IS OUTPUT SCALE FACTOR FXD PT
      CASE ONLY.  EQUIV(ACOR,SPACE) OK BUT DESTROYS ACOR.  SETS ERR=0. IF OK,
      =1. IF N, M, JMIN OR JMAX ILLEGAL.

ASPEC2 (ACOR,MXLAG,FREQLO,FRQDEL,                           FAP,   74 REGISTERS
        NFREQS,IERRLO,SPECT,IANS)
           NO OTHER ENTRY.  TRANSFER VECTOR - SEQSAC,NEXCOS.
      SETS  SPECT(J) = AC(0) + 2*SUM (FROM I=1 TO MXLAG) OF
      (AC(I)*COS(I*W(J))) FOR J = 1...NFREQS,  WHERE  W(J)=FREQLO+(J-1)*FRQDEL
      RADIANS, AND WHERE  AC(0...MXLAG)  FURNISHED IN  ACOR(1...MXLAG+1).
      REQUIRE  MXLAG  GRTHN=  0,  NFREQS  GRTHN=  1 .  SETS  IANS=0  IF OK,
      =IERRLO  IF  MXLAG  ILLEGAL,  =IERRLO+1  IF  NFREQS  ILLEGAL.

AVRAGE (X,LX,XAVG)                                           FAP,  24 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  XAVG = (1/LX) * SUM (FROM I= 1 TO LX) OF X(I).  STRAIGHT RETURN IF
      LX LSTHN 1.

BLKSUM (X,LX,LBLOK,DVSR,XBSMOD,LXBSOD)                      FAP,   49 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  LXBSOD = LX/LBLOK  ROUNDED DOWN AND SETS  XBSMOD(I)  =
      (1/DVSR) * (SUM  (FROM J=(I-1)*LBLOK+1 TO I*LBLOK)  OF  X(J) )
      FOR  I=1,2,...,LXBSOD.  EQUIVALENCE  (X,XBSMOD)  OK.  STRAIGHT RETURN
      WITH NO OUTPUT IF  LX  OR  LBLOK  LSTHN  1,  IF  LBLOK  GRTHN  LX,  OR
      IF  DVSR = 0.0 .

BOOST  (X,LX,XRIZE,XBUSTD)                                   FAP,  34 REGISTERS
           OTHER ENTRIES - XBOOST,DPRESS,XDPRSS.  NO TRANSFER VECTOR.
      SETS  XBUSTD(1...LX) = X(1...LX)+XRIZE.  EQUIV(X,XBUSTD) OK, AND
      EQUIV(XRIZE, SOME X(I)) OK, BUT INITIAL VALUE OF  XRIZE  IS ALWAYS THE
      ADDEND.  STRAIGHT RETURN IF LX LSTHN 1.

CALL   (SUBRU,IANS,SPACER,ARG1,ARG2,...,ARGN)   FAP, SECONDARY ENTRY OF LOCATE
      IS SAME AS CALL SUBR(ARG1...ARGN) WHERE SUBRU IS PROXY NAME FOR
      SUBR AND SPACER IS DUMMY.  SETS IANS=0 IF ALL OK, =-1,...,-4 IF
      SUBROUTINE NOT FOUND (SEE DETAILS UNDER ENTRY WHERE).

CALL2  (SUBRUV,IANS)                             FAP, SECONDARY ENTRY OF LOCATE
      IS EQUIVALENT TO CALL SUBR(ARG1...ARGN) IF SUBRUV WAS FORMED BY
      CALL SETSBV(SUBRU,SUBRUV,ARG1...ARGN) WHERE SUBRU IS PROXY NAME OF
      SUBR.  SETS IANS=0 IF ALL OK, =-1,...,-4 IF SUBROUTINE NOT FOUND (SEE
      DETAILS UNDER ENTRY WHERE).
```

```
CARIGE (ITAPE,NSPACE)                                          FORTRAN,  47 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - (STH),(FIL).
      CAUSES NSPACE SPACES TO BE PRINTED FROM LOGICAL TAPE ITPOUT PROVIDED
      NSPACE GRTHN= 0.  IF NSPACE LSTHN= -1 IT CAUSES 1 PAGE RESTORE.

CHISQR (NBLOCS,ICOUNT,N,CHISQ,IANS)                            FORTRAN, 105 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS CHISQ = SUM(FROM I=1 TO NBLOCS) OF ((1/ECNT)*(ICOUNT(I)-ECNT)**2),
      WHERE ECNT=N/NBLOCS, GIVEN N = SUM OF ICOUNT(I).  SETS IANS=0 IF OK,
      =1 OR =2 IF ILLEGAL NBLOCS OR N.

CHOOSE (ZIFRST, X,X1,X2, Y,Y1,Y2, ..., Z,Z1,Z2)                   FAP,   17 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      IF  ZIFRST=0,  SETS  X=X1, Y=Y1, ..., Z=Z1.  IF  ZIFRST  NOT=  0,  SETS
      X=X2, Y=Y2, ..., Z=Z2.  MODES OF ARGUMENTS IMMATERIAL.

CHPRTS (SYM,ANT,N)                                                FAP,   76 REGISTERS
           OTHER ENTRY - RVPRTS.  NO TRANSFER VECTOR.
      REVERSES SYM(1...LS) AND REVERSES ANT(1...LA) CHANGING SIGNS, WHERE
      LS=LA=N/2 IF N EVEN,  LS=(N+1)/2 LA=(N-1)/2 IF N ODD.  STRAIGHT EXIT IF
      N LSTHN= 1.

CHSIGN (X,LX,XNEG)                                                FAP,   18 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS XNEG(1...LX) = -X(1...LX).  EQUIV(X,XNEG) OK.  STRAIGHT RETURN
      IF  LX  LSTHN 1.

CHUSETF(X,X1,X2,ZIFX1)                                FAP, SECONDARY ENTRY OF INDEX
      PUTS  X1  (IF  ZIFX1 = 0.0)  OR  X2  (IF  ZIFX1 NOT= 0.0)  INTO MACHINE
      LOCATION CONTAINING  X,  THEN SETS ACCUMULATOR  =  ZIFX1,  WHERE  MODES
      OF ARGUMENTS IMMATERIAL.

CLKON                                                          FORTRAN,  46 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - CLOCK1,(SPH),(FIL).
      IF THE INTERVAL TIMER IS ON, CONTROL RETURNS IMMEDIATELY.  IF NOT, CLKCN
      PRINTS THE ON-LINE MESSAGE  QUOTE OPERATOR PLEASE TURN INTERVAL TIMER
      ON UNQUOTE  UNTIL THE TIMER IS TURNED ON.

CLOCK1 (JOB,TIME)                                         FAP (7090),  57 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      FOR JOB=0 CLOCK1 CHECKS TO SEE IF INTERVAL TIMER IS RUNNING, AND LEAVES
      JOB=0 IF RUNNING, SETS JOB=-1 IF NOT RUNNING.  FOR JOB=1 CLOCK1
      REMEMBERS PRESENT SETTING OF TIMER.  FOR JOB=2 (OR 3) CLOCK1 SETS TIME =
      NO. SECONDS, FLTG, (OR NO. CLOCK COUNTS, FIXED) SINCE LAST CALL OF
      CLOCK1 WITH JOB=1.

CMPARL (V1,V2,LV,IANS)                                FAP, SECONDARY ENTRY OF CMPARV
      SETS IANS = +1 IF V1(I)=V2(I) FOR ALL I=1...LV (36 BIT COMPARISON IS
      MADE IN WHICH +0 IS CONSIDERED NOT = -0), OR IANS = -K IF V1(K)
      NOT = V2(K)  (COMPARISON ORDER IS 1,LV,LV-1,...,2), OR IANS=0
      IF LV LSTHN 1.

CMPARP (IANS,X1,Y1,X2,Y2,...,XN,YN)                               FAP,   53 REGISTERS
           OTHER ENTRY - CMPARS.  NO TRANSFER VECTOR.
      SETS IANS=0 IF X1=Y1 AND X2=Y2 AND ... AND XN=YN, WHERE +0=-0.  SETS
      IANS=K IF XK NOT= YK, WHERE K IS LOWEST SUCH INDEX.
```

```
CMPARS (IANS,X1,X2,...,XN)                           FAP, SECONDARY ENTRY OF CMPARP
      SETS IANS=0 IF X1=X2=...=XN, WHERE +0=-0.  SETS IANS=K IF XK NOT= XK+1
      WHERE K IS LOWEST SUCH INDEX.

CMPARV (V1,V2,LV,IANS)                                           FAP,  50 REGISTERS
           OTHER ENTRY - CMPARL.  NO TRANSFER VECTOR.
      SETS IANS = +1 IF V1(I)=V2(I) FOR ALL I=1...LX, WHERE
      +0 IS CONSIDERED = -0, OR IANS = -K IF V1(K) NOT = V2(K)
      (COMPARISON ORDER IS 1,LV,LV-1,...,2), OR IANS=0 IF
      LV LSTHN 1.  MODE OF V1 AND V2 ARBITRARY

CMPRA F(X1,X2)                                                   FAP,   18 REGISTERS
           OTHER ENTRIES - XCMPRA,CMPRFL.  NO TRANSFER VECTOR.
      HAS VALUE  =  0  IF  X1  AND  X2  ARE IDENTICAL INCLUDING SIGN BIT,
      VALUE  =  1  IF  X1  IS ALGEBRAICALLY GRTHN  X2,  VALUE  =  -1  IF
      X1  IS ALGEBRAICALLY LSTHN  X2  WHERE  +0  GRTHN  -0  AND MODES OF  X1
      AND  X2  IMMATERIAL.

CMPRFLF(X1,X2)                                        FAP, SECONDARY ENTRY OF CMPRA
      HAS VALUE  =  0  IF  TX1  AND  TX2  ARE IDENTICAL INCLUDING SIGN BIT,
      VALUE  =  1  IF  TX1  IS ALGEBRAICALLY GRTHN  TX2,  VALUE  =  -1  IF
      TX1  IS ALGEBRAICALLY LSTHN  TX2  WHERE  TX1  AND  TX2  REPRESENT THE
      30 MOST SIGNIFICANT BINARY BITS OF  X1  AND  X2  RESPECTIVELY,
      +0  GRTHN  -0,  AND MODES OF  X1  AND  X2  IMMATERIAL.

CNTRDB (ITAPE,ISENSE,GZFAMP,VOFXY,LXV,                      FORTRAN,  550 REGISTERS
        LYV,LXDIM,VZERO,SPACE,IANS)
           NO OTHER ENTRIES.  TRANSFER VECTOR - SETVEC,CONTUR,SAME,LOG,EXP,
           (STH),(FIL).
      FORMS 12-INCH (121 COLUMNS) BY 24-INCH (145 ROWS) CONTOUR PLOT ON
      LOGICAL  ITAPE  FROM MATRIX  VOFXY(1...LXV,1...LYV),  FOR WHICH USER HAS
      DIMENSION  VOFXY(LXDIM,IGNORD),  VOFXY(1...LXV,1)  BECOMING FIRST
      OUTPUT ROW AND  V(1...LXV,LYV)  LAST.  BUILT IN CONTOUR LEVELS ARE
      PRINTED OUT.  PLOT IS MADE OF  20*LOG(VOFXY/VZERO)  IF  GZFAMP  GRTHN
      0.,  OF  10*LOG(VOFXY/VZERO)  IF  GZFAMP = 0.,  OR OF  VOFXY  IF  GZFAMP
      LSTHN  0.  IF  ISENSE = 1...6,  ON-LINE MONITORING OF PLOT OCCURS WHILE
      SENSE SWITCH  ISENSE  IS DEPRESSED. SPACE(1...204+LXV+XMAXOF(4,484/LXV))
      NEEDED FOR SCRATCH.  LXV  AND  LYV  MUST EXCEED  1,  LXDIM  GRTHN=  LXV,
      AND  VZERO  NOT=  0  IF  GZFAMP  GRTHN=  0.  SETS  IANS = 0  IF OK,
      = -1,-2,-3,-4  IF  LXV, LYV, LXDIM,  OR  VZERO  ILLEGAL,  = -100+K  IF
      CONTUR  FLAGS ERROR WITH ITS  IANS = K.

CNTROW (VEC,LVEC,FXLO,FXHI,NCOLS,CHLVLS,NCHRS,              FORTRAN,  802 REGISTERS
        DELEVL,VLEVL,SPACE,PLOTVC,IANS)
           NO OTHER ENTRIES.  TRANSFER VECTOR - CUFIT1,QUFIT1,FASCUB,RND,
           RNDDN,RNDUP.
      SETS  PLOTVC(1...NCOLS)  WITH BLANKS, WITH CHARACTERS SELECTED FROM
      CHLVLS(1...NCHRS),  AND POSSIBLY WITH  *  OR  $  CHARACTERS, TO INDICATE
      APPROXIMATE POSITIONS OF SPECIFIED LEVELS OF VALUES OF A SUBSECTION OF
      VEC(1...LVEC),  THE SUBSECTION BEING SYMBOLIZED BY  VEC(FXLO...FXHI)
      WHERE  FXLO,  FXHI  MAY BE FRACTIONAL.  CUBIC INTERPOLATION IS USED.  IF
      DELEVL=0.  THEN  VLEVL(1...NCHRS)  SPECIFIES LEVELS CORRESPONDING TO THE
      1A6  FORMAT CHARACTERS IN  CHLVLS.  IF  DELEVL  GRTHN  0.,  VLEVL  IS
      SIMPLE VARIABLE, CONTOUR LEVELS ARE  VLEVL  PLUS OR MINUS MULTIPLES OF
      DELEVL,  AND ASSOCIATION OF  CHLVLS(1...NCHRS)  WITH  VLEVL,
```

(CONTINUED NEXT PAGE)

VLEVL+DELEVL, ... IS PERIODIC. * IS USED TO INDICATE 2 LEVELS CROWDING 1 COLUMN, $ FOR 3 OR MORE. REQUIRE LVEC GRTHN= 2, FXLO GRTHN= 1.0, FXHI GRTHN FXLO AND LSTHN= FLOATF(LVEC), NCOLS GRTHN= 2, NCHRS GRTHN= 1, DELEVL GRTHN= 0., VLEVL(I+1) GRTHN VLEVL(I) FOR CASE DELEVL = 0., AND SPACE(1...2+MAXOF(4,4*NCOLS/L)) BE AVAILABLE FOR SCRATCH WHERE L = FXHI ROUNDED UP - FXLO ROUNDED DOWN. SETS IANS = 0 IF OK, = -1,-2,...,-7, IF LVEC,FXLO,FXHI,NCOLS,NCHRS, DELEVL,VLEVL ILLEGAL.

COLABL (ITAPE,ICOLLO,NCOLLO,NCOLS,ISPACE) FORTRAN, 185 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - GENHOL,(SPH),(FIL),(STH).
PRINTS THREE LINES ON LOGICAL ITAPE USING COLUMNS ICOLLO THRU ICOLLO+NCOLS-1, COLUMN ICOLLO DISPLAYING THE 3-DIGIT INTEGER NCOLLO, COLUMN ICOLLO+1 DISPLAYING NCOLLO+1, ETC. REQUIRE ISPACE(1...NCOLS) FOR SCRATCH, ALL INPUTS GRTHN= 1 EXCEPT EXCEPT NCOLLO GRTHN= 0, AND ITAPE LSTHN= 20 . ONLY CHECK ITAPE GIVING STRAIGHT RETURN IF ILLEGAL.

COLAPS (X,N,TYPE,XC,M) FAP, 50 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XC(1...M) FROM X(1...N), WHERE XC(I) = X(I)+X(I+M)+X(I+2M) +... IF N EXCEEDS M, XC(1...N) = X(1...N) AND XC(N+1...M) = 0 IF N LSTHN= M. TYPE=0. MEANS X FXD., NOT=0. IF FLTG.

CONTUR (ITAPE,ISENSE,VOFXY,LVX,LVY,LXDIM,FXLO, FXHI,NCOLS,NCOLLO,FYLO,FYHI,NROWS,ARGLO, ARGDEL,ZFAFXD,CHLVLS,NCHRS,DELEVL,VLEVL, SPACE,IANS) FORTRAN, 587 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - RNDDN,RNDUP,COLABL,ARBCOL, CNTROW,XSAME,(STH),(FIL),(SPH),SWITCH.
STARTS WITH PAGE RESTORE AND FORMS A CONTOUR PLOT ON LOGICAL ITAPE (VALUE 1 TO 20) OCCUPYING (EXCLUSIVE OF LABELLING) NCOLS (2 TO 119) COLUMNS AND NROWS (GRTHN= 2) ROWS, OF AN ARBITRARY RECTANGULAR SUBSET OF MATRIX VOFXY(1...LVX,1...LVY) (WITH LVX, LVY GRTHN= 2) FOR USER HAS DIMENSION VOFXY(LXDIM,IGNORD) (WITH LXDIM GRTHN= LVX), WHERE THE SUBSET IS SYMBOLIZED BY VOFXY(FXLO...FXHI,FYLO...FYHI), FXLO,FXHI,FYLO, AND FYHI BEING NOT-NECESSARILY-INTEGRAL INDICES SATISFYING 1.0 LSTHN= FXLO LSTHN FXHI LSTHN= FLOATF(LVX) AND 1.0 LSTHN= FYLO LSTHN FYHI LSTHN= FLOATF(LVY), AND WHERE THE FIRST OUTPUT ROW IS FOR VOFXY(FXLO...FXHI,FYLO). CUBIC INTERPOLATION IS USED IN FINDING POSITIONS OF CONTOUR LEVELS. COLUMNS ARE LABELLED FROM NCOLLO (VALUE 0 TO 1000-NCOLS) TO NCOLLO+NCOLS-1 . ROWS ARE LABELLED ARGLO,ARGLO+ARGDEL,... WHERE ZFAFXD = 0 OR NOT= 0 INDICATES ARGLO, ARGDEL FIXED OR FLOATING RESPECTIVELY. CHLVLS(1...NCHRS) (WITH NCHRS GRTHN= 1) ARE FORMAT(1A6) CHARACTERS TO USE FOR CORRESPONDING CONTOUR LEVELS. IF DELEVL = 0., THEN VLEVL(1...NCHRS) (MUST BE MONOTONELY INCREASING) ARE THE LEVELS. IF DELEVL GRTHN 0. (MUST NOT BE LSTHN 0.), THEN VLEVL IS SIMPLE VARIABLE, CONTOUR LEVELS ARE VLEVL PLUS OR MINUS ALL INTEGRAL MULTIPLES OF DELEVL, AND ASSOCIATION OF CHLVLS(1...NCHRS) WITH VLEVL,VLEVL+DELEVL,... IS PERIODIC. * IS USED TO INDICATE 2 LEVELS CROWDING ONE PRINT POSITION, $ FOR 3 OR MORE. REQUIRE SPACE(1...L+NCOLS+3+XMAXOF(4,4*NCOLS/L)) FOR SCRATCH WHERE L = FXHI ROUNDED UP - FXLO ROUNDED DOWN. SETS IANS = 0 IF OK, = -1,-2,...,-9, -10,-105,-106,-107 FOR ILLEGAL ITAPE,LVX,LVY,LXDIM,FXLO,FXHI,NCOLS, FYLO,FYHI,NROWS,NCHRS,DELEVL,VLEVL.

```
CONVLV (LX,XX,LY,YY,CC)                                  FORTRAN, 96 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS CC(1...LX+LY-1) = C(0...LX+LY-2) WHERE C(I) = SUM(FROM J=0 TO LX-1)
      OF (X(J)*Y(I-J))  GIVEN X(0...LX-1) IN XX(1...LX) AND Y(0...LY-1) IN
      YY(1...LY), AND ASSUMING Y(K)=0 FOR K OUTSIDE RANGE 0...LY-1.  STRAIGHT
      RETURN IF LX OR LY LSTHN= 0.  EQUIV(XX,YY) OK.

CONVLV (LX,XX,LY,YY,CC) -II                                   FAP,  56 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SAME FUNCTION AS FORTRAN VERSION OF CONVLV.

COSISP (SSX,ASX,SAX,AAX,L,COSTAB,SINTAB,M,          FAP, SECONDARY ENTRY OF COSP
        JMIN,JMAX,TYPE,COSTR,SINTR)
      SETS  COSTR(I)  AND  SINTR(I),  I=1...JMAX-JMIN+1, IN SAME WAY THAT
            CALL COSP (SSX,ASX,L,COSTAB,M,JMIN,JMAX,TYPE,COSTR)
            CALL SISP (SAX,AAX,L,SINTAB,M,JMIN,JMAX,TYPE,SINTR)
      WOULD SET THEM.  EQUIV (SSX,ASX,SAX,AAX)  OK.

COSIS1 (JOB,X,LX,COSTAB,SINTAB,MFREQ,JMIN,                FORTRAN, 406 REGISTERS
        JMAX,COSTR,SINTR,ZIFSTO,SPACE,IANS)
           NO OTHER ENTRIES.  TRANSFER VECTOR - IXCARG,SPLIT,MOVREV,CHPRTS,
           COSP,SISP,COSISP.
      SETS COSTR(I),  I=1...JMAX-JMIN+1,  IN SAME WAY THAT
            CALL COSP(X, X, LX-1, COSTAB, MFREQ, JMIN, JMAX, 1., COSTR)
      WOULD SET IT IF  JOB=1  OR  =3 .  SETS  SINTR(I),
      I=1...JMAX-JMIN+1,  IN SAME WAY THAT
            CALL SISP(X, X, LX-1, SINTAB, MFREQ, JMIN, JMAX, 1., SINTR)
      WOULD SET IT IF  JOB=2  OR  =3 .  IF COSINE OR SINE TRANSFORM NOT
      WANTED, ARGUMENTS ASSOCIATED WITH IT ARE DUMMIES.  LX  MUST BE ODD.
      ZIFSTO=0.  IMPLIES STORE  COSTR  AND/OR  SINTR,  NOT= 0.  IMPLIES ADD
      VALUES INTO OUTPUT AREAS.  SPACE(1...LX+3)  IS SCRATCH.  EQUIVALENCE
      (X,SPACE)  OK.  SETS  IANS = 0  IF NO ILLEGAL INPUTS,  =  ARGUMENT
      NUMBER IF IT IS ILLEGAL.

COSP   (SSX,ASX,L,COSTAB,M,JMIN,JMAX,TYPE,COSTR)              FAP, 504 REGISTERS
           OTHER ENTRIES - SISP,COSISP.  NO TRANSFER VECTOR.
      SETS COSTR(1...JMAX-JMIN+1) = CT(JMIN...JMAX) WHERE CT(J) = SUM (FROM
      I=0 TO L) OF (X(I)*COS(I*J*(PI/M))) AND X(0...L) = SSX(1...L+1) FOR
      J EVEN, = ASX(1...L+1) FOR J ODD.  COSTAB(1...M+1) IS INPUT TABLE
      CONTAINING COS(I*PI/M)  I=0...M .  TYPE = 0.0 SIGNIFIES SSX, ASX AND
      COSTAB FXD.PT., NOT = 0.0 SIGNIFIES FLTG.PT.  EQUIV(SSX,ASX) OK.
      IF M NEGATIVE, ITS MAGNITUDE IS USED AND CT(...) IS ADDED INTO
      COSTR(...) RATHER THAN STORED INTO IT.  STRAIGHT RETURN IF L LSTHN= 0,
      OR M=0, OR JMIN LSTHN 0, OR JMAX LSTHN= JMIN OR GRTHN M.

COSTBL (N,COSTAB)                                             FAP, 121 REGISTERS
           OTHER ENTRIES - SINTBL,COSTBX,SINTBX.  TRANSFER VECTOR - COS,SIN.
      SETS COSTAB(1...N+1) = COS(I*PI/N)  I=0...N.  STRAIGHT RETURN IF
      N LSTHN= 0.

COSTBX (N,ICOSTB)                                   FAP, SECONDARY ENTRY OF COSTBL
      SETS ICOSTB(1...N+1) = COS(I*PI/N)  I=0...N, WHERE ICOSTB IS FXD.PT.
      (BINARY PT BETWEEN SIGN AND BIT 1), AND 1.0 = OCT 377777777777. STRAIGHT
      RETURN IF N LSTHN= 0.
```

CPYFL2 (ITPIN,ITPOUT,LRECMX,ZFEOFW,SPACE,IANS) FAP, 178 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - (IOS),(TCO),(WRS),(RCH),(TRC),(ETT),(WEF),(BSR),(RDS).
COPIES ONE FILE (BINARY OR BCD) OF RECORDS OF LENGTH LSTHN= LRECMX FROM LOGICAL TAPE ITPIN TO LOGICAL TAPE ITPOUT. IF ZFEOFW = 0, THE END-OF-FILE MARK IS ALSO COPIED, OTHERWISE NO END-OF-FILE MARK IS PLACED ON ITPOUT. SPACE(1...2*LRECMX) NEEDED FOR SCRATCH. IF RECORDS ARE LONGER THAN LRECMX, THEY ARE TRUNCATED. SOME TYPICAL FORTRAN-II RECORD LENGTHS ARE BCD CARDS - 14 WORDS, BCD OUTPUT RECORDS - 22 WORDS, PACKED BCD OUTPUT RECORDS - 66 WORDS, BINARY CARDS - 27 WORDS, BINARY OUTPUT RECORDS - 256 WORDS. SETS IANS = 0 IF ALL OK, = 1, 2, OR 3 IF PERMANENT REDUNDANCY ON ITPIN, ITPOUT, OR BOTH, = 4, 5, ..., 15 IF END TAPE AND ALSO POSSIBLE REDUNDANCIES ENCOUNTERED ON ONE OR BOTH UNITS. SEE WRITEUP.

CROSS (LX,X,LY,Y,LC,C) FORTRAN, 107 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FDOT,STZ.
SETS C(1...LC) = XCOR(0...LC-1) WHERE XCOR(K) = SUM (FROM I=1 TO LX) OF (X(I)*Y(I-K)) WHERE Y IS TAKEN TO BE ZERO OUTSIDE ITS RANGE. ROUTINE RETURNS WITH NO COMPUTATION IF LX, LY, LC LSTHN 1 . EQUIVALENCE (X,Y) OK.

CROST (LX,X,LY,Y,ILAG,LC,C) FORTRAN, 134 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - CROSS,REVERS.
SETS C(1...LC) = XCOR(ILAG,...,ILAG+LC-1) WHERE XCOR(K) = SUM (FROM I=1 TO LX) OF (X(I)*Y(I-K)), WHERE Y IS TAKEN = 0.0 OUTSIDE ITS RANGE. ROUTINE RETURNS WITH NO COMPUTATIONS IF LX, LY, LC LSTHN 1 . EQUIVALENCE (X,Y) OK.

CRSVM (NRAC,NCARB,NCBC,LA,AA,LB,BB,ZIFNTR,ILAG,LC,CC) FORTRAN, 327 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - MDOT3,STZ,SETKS.
SETS CC(1...NRAC*NRBC*LC) = C(ILAG...ILAG+LC-1) WHERE C(K) = SUM (FROM I=1 TO LC) OF MATRIX PRODUCT OF A(I) AND B(I-K), WHERE C(I) IS THE NRAC X NCBC MATRIX STORED BY COLUMNS BEGINNING AT CC(1+NRAC*NCBC*(I-1)), A(I) IS THE NRAC X NCARB MATRIX STORED BY COLUMNS BEGINNING AT AA(1+NRAC*NCARB*(I-1)), AND B(I) IS THE NRACB X NCBC MATRIX STORED BY COLUMNS, IF ZIFNTR=0., OR BY ROWS, IF ZIFNTR NOT= 0., BEGINNING AT BB(1+NRACB*NCBC*(I-1)). B(I) IS TAKEN TO BE 0.0 OUTSIDE ITS RANGE. NO COMPUTATIONS ARE MADE (CC MAY BE SET TO ZERO) IF NRAC,NCARB,NCBC,LA,LB,LC LSTHN 1 . EQUIVALENCE (AA,BB) OK.

CSOUT (ITAPE,NSPACE,C1,C1NAME,C2,C2NAME,...,CN,CNNAME) FAP, 49 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - CARIGE,HRADJ,(STH),(FIL).
OUTPUTS C1NAME,C1, C2NAME,C2, ..., CNNAME,CN ON LOGICAL OUTPUT TAPE ITAPE ACCORDING TO THE FORMAT (5(2X, A6, 3H = , G14.7)) PRECEDED BY NSPACE SPACES (OR PAGE RESTORE IF NSPACE LSTHN 0).

CUFIT1 (FOFX,XLO,DELX,COEFS) FAP, 158 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS COEFS(1...4) = C0,C1,C2,C3 SUCH THAT G(X) = C0 + C1*X + C2*X**2 + C3*X**3 SATISFIES G(XLO) = FOFX(1), G(XLO+DELX) = FOFX(2), G(XLO+2*DELX) = FOFX(3), G(XLO+3*DELX) = FOFX(4) . STRAIGHT RETURN WITH NO OUTPUT IF DELX = 0.

```
CVSOUT (ITAPE,NSPACE,FMTHED,FMTLIN,ILO,IHI,                  FAP,  84 REGISTERS
        ARGLO,ARGDEL,SPACE,X1,X2,...,XN)
          NO OTHER ENTRIES.  TRANSFER VECTOR - CARIGE,FMTOUT,VECOUT.
     OUTPUTS N VECTOR RANGES,  X1(ILO...IHI),X2(ILO...IHI),...,XN(ILO...IHI),
     IN COLUMN FORMAT, INSERTING FIRST COLUMN  ARG = ARGLO,ARGLO+ARGDEL,
     ARGLO+2*ARGDEL,...,  ONTO LOGICAL TAPE ITAPE WITH NSPACE INITIAL SPACES
     (OR PAGE RESTORE IF NSPACE LESS THAN 0).  FMTHED(I) IS A NORMLIT FORMAT
     VECTOR (AS DEFINED BELOW) FOR HEADING THE COLUMNS AND FMTLIN(I) IS A
     NORMLIT FORMAT VECTOR FOR PRINTING THE SUCCESSIVE LINES (MUST INCLUDE
     PRINTING OF ARG, ALWAYS FLOATING).  SPACE(1...N+1) NEEDED FOR SCRATCH.
          DEFINITION - A NORMLIT FORMAT VECTOR IS EITHER
             A) A NORMAL FORMAT VECTOR
          OR B) LITERAL HOLLERITH IN A CALLING SEQUENCE WHOSE CHARACTERS
                (READING CONTINUOUSLY FROM LEFT TO RIGHT) ARE THE DESIRED
                FORMAT STRIPPED OF THE ENCLOSING PARENTHESES.  THE FIRST AND
                SECOND CHARACTERS MUST NOT BE QUOTE ( UNQUOTE OR QUOTE )
                UNQUOTE RESPECTIVELY. (TWO BLANKS FOLLOWED BY ( WOULD BE OK.)

DADECK (ITPIN,ITPOUT)                                     FORTRAN,  100 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - EOFSET,RSKIP,(TSH),(RTN),
          (STH),(FIL).
     COPIES SUCCESSIVE CARD IMAGES (COLUMNS  1  THRU  80)  FROM LOGICAL TAPE
     ITPIN  ONTO LOGICAL TAPE  ITPOUT  (COLUMNS  2  THRU  81)  UNTIL
     END-OF-FILE REACHED ON  ITPIN.  THEN BACKSPACES  ITPIN  TO ORIGINAL
     POSITION.

DELTA F(ARG)                                                  FAP,   17 REGISTERS
          OTHER ENTRIES - XDELTA,STEPR,XSTEPR,STEPL,XSTEPL,STEPC,XSTEPC.  NO
          TRANSFER VECTOR.
     HAS VALUE  =  1.0  IF  ARG  (ANY MODE)  =  ZERO.  OTHERWISE HAS VALUE
     =  0.0  .

DERIVA (YOFX,LY,DELX,DYDX,YOFX1)                              FAP,  61 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS DYDX(1) = (YOFX(2)-YOFX(1))/DELX, AND (IF LY GRTHN 2)
     DYDX(K) = (YOFX(K+1)-YOFX(K-1)/(2.0*DELX) FOR K=2...LY-1,
     AND DYDX(LY) = (YOFX(LY)-YOFX(LY-1))/DELX, AND YOFX1=YOFX(1).
     EQUIV(DYDX,YOFX) OK.  STRAIGHT RETURN IF LY LSTHN 2 OR
     DELX = 0.  (BUT MAY BE NEGATIVE).  FUNCTION IS EXACT INVERSE
     TO THAT OF IDERIV.

DETRM  (N,LN,A,D,ERR)                            FAP, SECONDARY ENTRY OF  SIMEQ
     SETS  D = CONSTANT*DETERMINANT OF MATRIX  A(I,J)  I,J=1...LN WHERE N
     IS USERS DIMENSION OF I  (2 LSTHN= LN LSTHN=N), AND CONSTANT = INPUT
     VALUE OF D.  SETS D = 0.0  IF A SINGULAR.  SETS ERR = 0.0  IF OK,
     NON SINGULAR, =1.0 IF OVER OR UNDERFLOW, = 2.0  IF SINGULAR.  A(I,J)
     DESTROYED.

DIFPRS (X,LX,XPRSDF)                                          FAP,  30 REGISTERS
          OTHER ENTRY - XDFPRS.  NO TRANSFER VECTOR.
     SETS  XPRSDF(1)=X(1),  XPRSDF(I)=X(I)-X(I-1)  FOR  I=2...LX.
     EQUIV(XPRSDF,X) OK.  STRAIGHT RETURN IF  LX  LSTHN 1.
```

```
DISPLA                                                   FAP (709) , 220 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - (IOH).
                                                         FAP (7090), 219 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - (IOH),FRAME.
     THE SEQUENCE CALL DISPLA - PRINT FMT, LIST - FMT FORMAT (NHMCDX,Y,
     FMTEND) FUNCTIONS LIKE PRINT FMT, LIST - FORMAT(FMTEND), WHERE FMT =
     STATEMENT NO. OR VARIABLE NAME CONTAINING THE FORMAT.  N = CHARACTER
     COUNT FROM M TO FMTEND.  C = B OR S FOR BIG OR SMALL CHARACTERS
     (BIG CHAR = 20*28 (36 ACROSS SCOPE), SMALL = 15*21 (48 ACROSS SCOPE)),
     D = H OR V FOR HORIZONTAL OR VERTICAL DISPLAY.  X,Y = 2 INTEGERS FOR
     SCOPE COORDINATES OF LOWER LEFT CORNER OF FIRST CHARACTER.  M = 2
     MEANS SET FOR NEW CDX,Y, AND SINGLE SPACING.  M=1 SAME AS M=2 BUT
     CHANGE FRAME FIRST.  M = + MEANS USE PREVIOUS M = 1 OR 2 MODE (CDX,Y NOT
     PRESENT).  M = O(ZERO) SAME AS = + BUT DOUBLE SPACE.  M = (BLANK)
     SAME AS M = + BUT SINGLE SPACE.  NHMCDX,Y, MUST BE TIGHT PACKED.

DIVIDE (X,LX,XDVSR,XDVDED)                                      FAP,  23 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  XDVDED(1...LX) = X(1...LX)/XDVSR.  EQUIV(X,XDVDED) OK, AND
     EQUIV(XDVSR, SOME X(I)) OK, BUT INITIAL VALUE OF  XDVSR  IS ALWAYS THE
     DIVISOR.  STRAIGHT RETURN IF XDVSR=0.0, OR LX LSTHN 1.

DIVK   (C,X1,X2,...,XN)                          FAP, SECONDARY ENTRY OF ADDK
     SETS  X1=X1/C, X2=X2/C, ..., XN=XN/C.  EQUIV(ANY ARGUMENTS) OK, BUT
     INITIAL VALUE OF  C  IS ALWAYS THE DIVISOR.  STRAIGHT RETURN IF C=0.0,
     OR N=0.

DIVKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)          FAP, SECONDARY ENTRY OF ADDK
     SETS Y1=X1/C1, Y2=X2/C2, ..., YN=XN/CN.  EQUIV(ANY TWO ARGUMENTS)
     OK BUT MAY CHANGE INPUTS  CJ  OR  XJ.  PROCESSING IS LEFT TO RIGHT.
     YJ IS NOT COMPUTED IF  CJ=0  AT COMPUTATION TIME.
     STRAIGHT RETURN IF  N=0.

DO      (NSUBS,I,ILO,IHI)                          FAP, PSEUDO ENTRY OF SEVRAL
     USAGE IS CALL SEVRAL (...,2HDO,NSUBS,I,ILO,IHI,...).  FUNCTION
     IS SIMILAR TO THE FORTRAN STATEMENT DO NSUBS I=ILO,IHI WHEN
     NSUBS (MUST EXCEED ZERO) IS THE NO. OF SUBROUTINES (IMMEDIATELY
     FOLLOWING THE 2HDO SEQUENCE) IN THE DO LOOP.  ILO MAY BE NEGATIVE, OR
     ZERO.  LOOPS WITHIN LOOPS EXCLUDED.  PSEUDO IF STATEMENT IN LOOP
     EXCLUDED.

DOTJ   (LXY,IDX,X,IDY,Y,DOT,ADD,ORDER)                          FAP,  59 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS DOT=DOTP IF ADD LSTHN= 0, =DOT+DOTP IF ADD GRTHN 0, WHERE DOTP =
     X(1)*Y(1)+X(1+IDX)*Y(1+IDY)+X(1+2*IDX)*Y(1+2*IDY)+...+X(1+(LXY-1)*IDX)*
     Y(1+(LXY-1)*IDY)  IF ORDER GRTHN 0, AND DOTP = X(1)*Y(1+(LXY-1)*IDY)+
     ...+X(1+(LXY-1)*IDX)*Y(1)  IF ORDER LSTHN= 0.  IDX MUST BE GRTHN= 0,
     IDY MUST BE GRTHN= 1 .

DOTP   (NRA,NCA,AA,NRB,NCB,BB,IRB,ICB,DOT,ORDER)         FORTRAN,  264 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - DOTJ.
     SETS  DOT  =  SUM  (FROM I=1 TO NRA)  OF  SUM  (FROM J=1 TO NCA)  OF
     ( AA(I1+(J1-1)*NRA) * BB(I+IRB+(J+ICB-1)*NRB) )  WHERE IF  ORDER=1.,
```

(CONTINUED NEXT PAGE)

I1=I, J1=J, IF ORDER=2., I1=NRA-1+1, J1=J, IF ORDER=-1., I1=I, J1=NCA-J+1, AND IF ORDER=-2., I1=NRA-I+1, J1=NCA-J+1 . BB IS TAKEN AS 0. OUTSIDE ITS RANGE. AA IS AN NRA BY NCA ARRAY STORED CLOSELY SPACED BY COLUMNS, BB IS AN NRB BY NCB ARRAY STORED BY COLUMNS. DOT=0. IF NRA, NCA, NRB, NCB LSTHN 1 . EQUIVALENCE (AA,BB) OK.

DPRESS (X,LX,XSINK,XLWRD) FAP, SECONDARY ENTRY OF BOOST
SETS XLWRD(1...LX) = X(1...LX)-XSINK. EQUIV(X,XLWRD) OK, AND EQUIV(XSINK, SOME X(I)) OK, BUT INITIAL VALUE OF XSINK IS ALWAYS THE SUBTRAHEND. STRAIGHT RETURN IF LX LSTHN 1.

DSPFMT (CNTHOL,IORGX,IORGY,FMTEND,FMT) FAP, 194 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS FMT(1,2,...) = FORMAT SUITABLE FOR SUBROUTINE DISPLA, WITH DESIRED SCOPE ORIGIN X = IORGIN, Y=IORGY, WHERE CNTHOL = DESIRED CONTROL CHARACTERS M, C, D OF DISPLA IN FORMAT (1A3), AND FMTEND(1,0,-1,...) IS LITERAL HOLLERITH ARGUMENT GIVING FORMAT FOR PRINTING LIST (EXCLUDES EXTREMAL PARENTHESES).

DUBLL (X,LX) FAP, SECONDARY ENTRY OF DUBLX
SETS X(1...LX) = 2.0*X(1...LX). MAGNITUDE OF LX IS USED AND LX=0 TREATED AS LX=1 .

DUBLX (IX,LX) FAP, 45 REGISTERS
OTHER ENTRIES - DUBLL,HALVX,HALVL. NO TRANSFER VECTOR.
SETS IX(1...LX) = 2*IX(1...LX). MAGNITUDE OF LX IS USED AND LX=0 TREATED AS LX=1 .

ENDFIL (ITAPE) (FORTRAN FUNCTION) FAP, SECONDARY ENTRY OF REREAD
CHECKS AN INTERNAL FLAG OF REREAD. IF EOFSET WAS CALLED WITH ZIFTRN=1. AND IF AN END-OF-FILE WAS ENCOUNTERED, ENDFILF(ITAPE)=1. AND ITAPE = LOGICAL TAPE NUMBER THAT THE END-OF-FILE WAS ENCOUNTERED ON. OTHERWISE ENDFILF(ITAPE)=0. THE FLAG IS RESET AFTER EACH USE OF ENDFIL.

EOFSET (ZIFTRN,EOF,ITAPE) FAP, SECONDARY ENTRY OF REREAD
INSTRUCTS REREAD ON THE ACTION IT SHOULD TAKE IF AN END-OF-FILE IS ENCOUNTERED WHILE READING. IF ZIFTRN=-1. REREAD WILL CALL EXIT, IF =0. REREAD WILL RETURN CONTROL TO THE FIRST STATEMENT FOLLOWING THIS 'CALL EOFSET' STATEMENT WITH EOF=1. AND ITAPE = LOGICAL TAPE UNIT THAT THE END-OF-FILE WAS ENCOUNTERED ON, IF =1. REREAD WILL SET AN INTERNAL FLAG (THAT MAY BE CHECKED BY FUNCTION ENDFIL) AND INTERPRETS THE END-OF-FILE AS A RECORD OF BLANKS. EOF=0., ON NORMAL RETURN FROM EOFSET.

EXCHVS (LXY,X,Y) FAP, 22 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS Y(1...LXY) = INPUT VALUES OF X(1...LXY), AND X(1...LXY) = INPUT VALUES OF Y(1...LXY). EQUIV(X,Y) OK. STRAIGHT RETURN IF LXY LSTHN 1.

```
EXPAND (X,LX,MLPLYR,XPNDED,LXPNDD)                              FAP,  189 REGISTERS
         NO OTHER ENTRIES.  TRANSFER VECTOR - INTOPR.
     SETS  LXPNDD = (LX-1)*MLPLYR+1,  SETS  XPNDED(1,1+MXPLYR,1+2*MLPLYR,....,
     LXPNDD) = X(1...LX),  AND SETS THE INTERMEDIATE VALUES (FOR THE CASE
     MLPLYR  GRTHN=  2)  OF  XPNDED  BY CUBIC INTERPOLATION (REDUCED TO
     QUADRATIC AT THE ENDS OR TO LINEAR IF  LX = 2).  STRAIGHT RETURN WITH NO
     OUTPUT IF  LX  LSTHN=  0,  OR IF  LX  GRTHN=  2  BUT  MLPLYR  LSTHN=
     0 .  IF  LX = 1  MLPLYR  IS  IGNORED.

FACTOR (SPECT,N,L,WAVE,SPACE)                                   FAP,  308 REGISTERS
         NO OTHER ENTRIES.  TRANSFER VECTOR - MAXAB,COSTBL,COSP,LOG,EXP.
     SETS WAVE(1...L) = MINIMUM PHASE WAVELET WITH GIVEN ENERGY DENSITY
     SPECTRUM, SPECT(1...N), CORRESPONDING TO FREQUENCY RANGE 0 TO PI (ZERO
     LSTHN L LSTHN= N).  SPACE(1...3*L+N+1) NEEDED FOR SCRATCH.
     EQUIV(WAVE,SPECT) OK.

FAPSUM (LD,DATA,SUMCK)                                           FAP,  14 REGISTERS
         NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS SUMCK = LOGICAL SUM OF DATA(1...LD) (USING ACL INSTRUCTION)

FASCN1 (VECT,ILO,IHI,VALUE,IFIND,IANS)                           FAP, 107 REGISTERS
         NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SCANS VECT(ILO...IHI) LOOKING FOR FIRST ELEMENT IF ANY WHICH IS GRTHN=
     VALUE.  SETS IANS=0 IF NONE, SETS IFIND AND IANS=1 IF VECT(IFIND)
     GRTHN=VALUE.  SETS IANS=-2 OR -3 IF ILLEGAL ILO OR IHI (1 LSTHN=
     ILO LSTH=IHI).  VECT AND VALUE EITHER BOTH FLTG. PT. OR BOTH FXD. PT.

FASCOR (Y,KMIN,KMAX,CORZER,ERROR)                   FAP, SECONDARY ENTRY OF PROCOR
     SETS CORZER(-KMN+1...KMAX+1) = XCOR(-KMN...KMAX) WHERE XCOR(K) = SUM
     (FROM I=1 TO LX)  OF (X(I)*Y(I+K))  I.E. ZERO LAG GOES IN CORZER(1)
     AND WHERE  1) X AND LX WERE THE ARGUMENTS OF A PRIOR CALL PROCOR
     STATEMENT,  2) KMN=MAGNITUDE OF KMIN,  3) -LX LSTHN KMIN LSTHN= 0
     LSTHN= KMAX  LSTHN LX,  4) Y IS TAKEN TO BE = 0 OUTSIDE RANGE
     1...LX, AND  5) Y,X, AND CORZER ARE MACHINE LANGUAGE INTEGERS.  SETS
     ERROR = 0  IF OK,  = 1.0  IF NO PREVIOUS CALL PROCOR, = 2.0  IF
     ILLEGAL KMIN OR KMAX, = 3.0  IF OVERFLOW OCCURS.

FASCR1 (Y,KMIN,KMAX,CORZER,ERROR)                   FAP, SECONDARY ENTRY OF PROCOR
     FUNCTIONS IDENTICALLY TO SUBROUTINE FASCOR EXCEPT THAT THE
     CORRELATION IS ADDED INTO THE OUTPUT AREA RATHER THAN BEING STORED INTO
     IT.

FASCUB (COEFS,XLO,DELX,NF,FOFX)                                 FAP,  141 REGISTERS
         NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  FOFX(1...NF) = F(XLO), F(XLO+DELX), ..., F(XLO+(NF-1)*DELX)  WHERE
     F(X) = C0 + C1*X + C2*X**2 + C3*X**3  WHERE  C0,C1,C2,C3  GIVEN BY
     COEFS(1...4).  STRAIGHT RETURN WITH NO OUTPUT IF  LX  LSTHN=  0 .

FASEPC (Y,KMIN,KMAX,CORZER,ERROR)                   FAP, SECONDARY ENTRY OF PROCOR
     FUNCTIONS IDENTICALLY TO SUBROUTINE FASCOR EXCEPT THAT IT DOES NOT
     MAKE THE TRANSIENT ASSUMPTION ABOUT Y(I), I.E. IT GIVES EQUI-PRODUCTS
     CORRELATION.

FASEP1 (Y,KMIN,KMAX,CORZER,ERROR)                   FAP, SECONDARY ENTRY OF PROCOR
     FUNCTIONS IDENTICALLY TO SUBROUTINE FASEPC EXCEPT THAT THE CORRELATION
     IS ADDED INTO THE OUTPUT AREA RATHER THAN BEING STORED INTO IT.
```

```
FASTRK (IXVEC,IXSTRT,IXLOOK,MXTRAK,IANS)                   FAP,   26 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     EXAMINES IXVEC(IXSTRT),IXVEC(IXVEC(IXSTRT)),IXVEC(IXVEC(IXVEC(IXSTRT))),
     ...,ETC.,...  UNTIL WHICHEVER OF THE FOLLOWING OCCURS FIRST,  A) IT
     FINDS AN ELEMENT  IXVEC(K) = IXLOOK,  B) IT FINDS A ZERO ELEMENT IN
     IXVEC,  OR  C)  MXTRAK  EXAMINATIONS ARE COMPLETED WITHOUT ENCOUNTERING
     A)  OR  B).  SETS  IANS = K,0, OR -1  FOR CASE  A),  B),  OR  C).
     REQUIRE  IXVEC(I)  GRTHN=  0  AND  IXSTRT,IXLOOK,MXTRAK  GRTHN=  1,  BUT
     THESE REQUIREMENTS NOT CHECKED.

FDOT   (LXY,X,Y,ANS)                                        FAP,   40 REGISTERS
          OTHER ENTRY - FDOTR.  NO TRANSFER VECTOR.
     SETS ANS = X(1)*Y(1)+X(2)*Y(2)+...+X(LXY)*Y(LXY), WHERE LXY GRTHN= 1 .

FDOTR  (LXY,X,Y,ANS)                              FAP, SECONDARY ENTRY OF FDOT
     SETS ANS = X(1)*Y(LXY)+X(2)*Y(LXY-1)+...+X(LXY)*Y(1) WHERE LXY
     GRTHN= 1 .

FIRE2  (NRA,NCAT,NCAN,AA,NRR,NCR,RR,NRG,GG,FF,C)        FORTRAN, 271 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - IXCARG,STZ,DOTP,MATML3,DOTJ.
     SETS  FF(1...NRA*NCAN)  =  F(1...NRA,1...NCAN)  WHERE  SUM  (FROM
     I=1  TO  NRA)  OF  SUM  (FROM  J=1  TO  NCAN)  OF
     (F(I,J)*R(I-K,J-L))  =  G(K,L)  FOR  K=1...NRA,  L=1...NCAN,  GIVEN
     F(1...NRA,1...NCAN-1).  RR(1...NRR*NCR)  =  R(-NRR/2...NRR/2,0...NCR-1)
     WHERE  NRR  MUST BE ODD.  GG(1...NRG)  =  G(-NRG/2...NRG/2,NCAN).
     AA(1...NRA*NCAT*NRA)  AND  CC(1...4*NRA*NRA)  ARE THE OUTPUTS OF
     SUBROUTINE  RLSPR2.  NCAN  MUST BE  LSTHN=  NCAT.

FIXV   (X,LX,IXFIXD)                                        FAP,   35 REGISTERS
          OTHER ENTRY - FIXVR.  NO TRANSFER VECTOR.
     SETS  IXFIXD(1...LX)  FROM  X(1...LX),  WHERE  IXFIXD(I) = XFIXF(X(I)),
     WHERE X(I) IS TRUNCATED BEFORE FIXING.  EQUIV(IXFIXD,X) OK.  STRAIGHT
     RETURN IF LX LSTHN= 0 .

FIXVR  (X,LX,IXFIXD)                              FAP, SECONDARY ENTRY OF FIXV
     IDENTICAL TO  FIXV  EXCEPT  X(I)  IS ROUNDED BEFORE FIXING.

FLDATA (LX,X,SCALE)                             FAP, SECONDARY ENTRY OF FXDATA
     SETS  X(1...LX) = (FLTG.PT. FORM OF  X(1...LX))/SCALE, WHERE  X  ON
     INPUT ARE CONSIDERED 35-BIT-PLUS-SIGN INTEGERS, AND SCALE IS A NON-ZERO
     FLTG. NO.  STRAIGHT RETURN IF LX LSTHN= 0 OR IF SCALE = 0.

FLOATMF(INTEGR)                                             FAP,   25 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     FUNCTION CONVERTS INTEGR TO FLTG.PT., WHERE INTEGR IS ANY
     35-BIT-PLUS-SIGN  INTEGER.

FLOATV (IX,LIX,XFLOTD)                                      FAP,   22 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  XFLOTD(1...LIX)  FROM  IX(1...LIX),  WHERE  XFLOTD(I) =
     FLOATF(IX(I)).  EQUIV(XFLOTD,IX) OK.  STRAIGHT RETURN IF LIX LSTHN= 0 .
```

```
FMTOUT (ITAPE,FMT)                                        FORTRAN,  51 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - FNDFMT,RPLFMT,(STH),(FIL).
      OPERATION IS EQUIVALENT TO   WRITE OUTPUT TAPE ITAPE,FMT  , WHERE FMT(I)
      IS A NORMLIT FORMAT VECTOR, AS DEFINED ABOVE IN CVSOUT.

FNDFMT (FMT,IXCFMT)                                           FAP,  88 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - REVER.
      ON INPUT FMT(I) IS A NORMLIT FORMAT VECTOR AS DEFINED IN CVSOUT ABOVE.
      IF FMT IS NORMAL, IXCFMT IS SET = TO INDEX WITH RESPECT TO COMMON OF
      FMT(1), AND NO OTHER OUTPUT.  IF FMT(I) IS LITERAL THEN IT IS REVERSED
      IN PLACE, WITH ENCLOSING PARENTHESES ADDED TO MAKE IT A LEGAL FORMAT AND
      IXCFMT IS SET = INDEX WITH RESPECT TO COMMON OF THE RESULTING FORMAT
      VECTOR.  SUCCESSIVE CALLS OF FNDFMT WITH LITERAL FORMAT WORK PROPERLY
      WITHOUT LEADING TO RE-REVERSAL.

FRAME                                                 FAP (709) , 4 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
                                                      FAP (7090), 9 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      ADVANCES FILM IN SCOPE CAMERA BY ONE FRAME.  SEPARATE VERSIONS
      FOR 709, 7090.

FRQCT1 (IX,NX,IXLO,IXHI,ICT,IANS)                        FORTRAN, 117 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  ICT(1...IXHI-IXLO+1) = IC(IXLO...IXHI) WHERE IC(J) = NO. OF
      ELEMENTS OF IX(1...NX) WHICH HAVE VALUE = J.  IXLO LSTHN= ALL IX(I)
      LSTHN= IXHI. IANS=0  IF OK, = 1 OR 2 IF ILLEGAL NX OR IXLO.

FRQCT2 (X,LX,B,LB,ICOUNT,IANS)                               FAP, 117 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS ICOUNT(1...LB+1) WHERE ICOUNT(J) = NO. OF VALUES IN X(1...LX)
      SUCH THAT B(J-1) LSTHN= X LSTHN B(J), GIVEN MONOTONELY INCREASING
      VECTOR  B(1...LB), WHERE B(0) AND B(LB+1) ARE INFERRED TO BE - AND +
      INFINITY.  IANS = 0 IF OK, = 1,2, OR 3 IF ILLEGAL LX, ILLEGAL LB,
      OR SOMETHING WEIRD.  X MAY BE ANY MODE.

FSKIP  (ITAPE,NFILES)                                       FAP,   50 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - (IOS),(RDS),(BSR),(TCO),
          (TEF),(TRC).
      SPACES FORWARD NFILES FILES ON TAPE (BACKWARDS IF NFILES NEGATIVE),
      LEAVING TAPE AT END-OF-FILE-MARK EDGE FURTHEST FROM LOAD POINT.
      IF NFILES=0 TAPE NOT MOVED.  IF TAPE IS PART WAY THRU A FILE IT
      COUNTS AS 1 FILE.

FT24   (D,A,B)                                               FAP, 777 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - FXDATA,FLDATA.
      SETS A(1...13) = CT(0...12) AND B(1...13) = ST(0...12)  WHERE CT(J) =
      SUM(FROM I=0 TO 23) OF (X(I)*COS(I*J*PI/12)), ST(J) = SAME SUM WHERE
      SIN(...) REPLACES COS(...), AND X(0...23) IS GIVEN IN D(1...24).  D,A,B
      ARE FLOATING BUT COMPUTATIONS CARRIED OUT FXD PT TO ACCURACY OF 1 PART
      IN 10,000.

FT24   (DD,AA,BB) - II                                   FORTRAN, 818 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      OPERATES IDENTICALLY TO FAP VERSION OF FT24 EXCEPT COMPUTATION IS
      CARRIED OUT FLOATING POINT (THE FIXING PROCESS IS OMITTED).
```

FXDATA (LX,X,MXDATA,SCALE) FAP, 102 REGISTERS
OTHER ENTRY - FLDATA. NO TRANSFER VECTOR.
SETS X(1...LX) = SCALED AND FIXED FORM OF X(1...LX), THE X(I) BEING CONVERTED WITH ROUNDING TO MACHINE-LANGUAGE-INTEGERS WITH MAXIMUM MAGNITUDE = MXDATA (GIVEN AS FORTRAN INTEGER). ALSO SETS SCALE = FLOATF(MXDATA)/XMAX WHERE XMAX = MAX MAGNITUDE OF ORIGINAL X(I), BUT SETS SCALE = -1. IF LX LSTHN= 0 OR MXDATA LSTHN= 0, SCALE = -2. IF XMAX IS ZERO.

GENHOL (HOL) FAP, 48 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - (IOH).
USAGE IS - CALL GENHOL(HOL) - PRINT FMT, LIST - FMT FORMAT ( ) .
GENHOL SETS HOL(1...N) = HOLLERITH EQUIVALENT (FORMAT(NA6)) TO LINE(S) WHICH WOULD HAVE BEEN PRINTED BY THE PRINT STATEMENT (WHICH WILL BE BYPASSED ON RETURN). N WILL = (5 + TOTAL CHARACTER COUNT)/6 .

GETHOL (JOB,HARG,HOL,NCRS,IXCOM,ICOUNT) FORTRAN, 169 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - XLOC,REVERS.
HARG(1,0,-1...-LHOL+2) IS LHOL REGISTERS OF INPUT LITERAL HOLLERITH WITH FENCE AT HARG(-LHOL+1) . INITIALLY FENCE = OCT 777777777777, BUT IF THIS SAME CALL STATEMENT HAS BEEN OPERATED BEFORE WITH JOB NOT=0 FENCE WILL READ OCT 777777777776. SUPPOSE JOB NOT=0. THEN, IF FENCE = ALL 7'S, GETHOL REVERSES STORAGE OF HARG(1...-LHOL+2), SETS HARG(-LHOL+1) = OCT 777777777776, SETS NCRS = 6*LHOL, SETS IXCOM = INDEX WITH RESPECT TO COMMON OF HARG(-LHOL+2), AND INCREMENTS ICOUNT BY 1 . IF FENCE = OCT 777777777776, SAME OUTPUTS EXCEPT NO REVERSAL OF HARG(1...-LHOL+2). SUPPOSE JOB=0. THEN IF FENCE = OCT 777777777777, GETHOL SETS HOL(1...LHOL) = HARG(1,0,..., -LHOL+2), SETS NCRS = 6*LHOL, SETS IXCOM = INDEX WITH RESPECT TO COMMON OF HOL(1), ICOUNT NOT MODIFIED. IF FENCE = OCT 777777777776, SAME OUTPUTS EXCEPT HOL(1...LHOL) = HARG(-LHOL+2,...,0,1). IN ANY CASE ERROR RETURN WITH NCRS = -1 IF LHOL EXCEEDS 106.

GETRD1 (ITAPE,NX,IX,IANS) FORTRAN, 229 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - (TSH),(RTN).
SETS IX(1...NX) = NEXT NX (EXCEEDS 0) DIGITS FROM RANDOM DIGITS BCD TAPE (EACH CARD FORMAT(50I1)) MOUNTED ON LOGICAL TAPE NO. ITAPE. SETS IANS = 0 IF OK, = -1 OR +2 IF ILLEGAL ITAPE OR NX. NEVER REWINDS ITAPE.

GETX (X,I1,I2,...,IN) (FORTRAN FUNCTION) FAP, 31 REGISTERS
OTHER ENTRY - IGETX. NO TRANSFER VECTOR.
SETS Y = GETX(X,I1,I2,...,IN) WHICH IS EQUIVALENT TO THE LIST OF FORTRAN STATEMENTS JNM1 = INM1(IN), ..., J2 = I2(J3), J1 = I1(J2), Y = X(J1). EQUIVALENCE ANY IN OK.

GNFLT1 (AMSPEC,LSPEC,FLTR,IANS) FORTRAN, 232 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - COS.
SETS FLTR(1...2*LSPEC-1) = SYMMETRICAL (ABOUT FLTR(LSPEC)) COEFFICIENTS WHOSE AMPLITUDE SPECTRUM MATCHES SPECTRUM AMSPEC(1...LSPEC) GIVEN AT EQUALLY SPACED FREQUENCIES FROM 0 TO PI. FLTR FORMED FROM TUKEY-HAMMING ORTHONORMAL SET. SETS IANS = 0 IF OK, = -1 FOR ILLEGAL AMSPEC (ALL ZERO), = -2 FOR ILLEGAL LSPEC (OUTSIDE RANGE 3 TO 1001).

```
GNHOL2 (DATA,NDATA,FMT,HOL,NCRS,IXCOM,INDEX)                     FAP,  74 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - (IOH),(FIL).
     FMT(1,0,...,-M+2)  IS  M  REGISTERS OF INPUT LITERAL HOLLERITH
     REPRESENTING A FORMAT BY WHICH DATA(1...NDATA) IS TO BE INTERPRETED
     (NDATA MAY = 0) .  GNHOL2 SETS  HOL(1...NCRS/6) = NCRS HOLLERITH
     CHARACTERS RESULTING FROM FMT AND DATA, SETS NCRS = 6*NO. WORDS
     IN HOL, SETS IXCOM = INDEX WITH RESPECT TO COMMON OF HOL(1), AND
     INCREMENTS INDEX BY 1 .

GRAPH  (ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,              FORTRAN, 1499 REGISTERS
        YBOT,XMAX,XMIN,NOPPP,IPAGE,SPACE)
          NO OTHER ENTRIES.  TRANSFER VECTOR - DISPLA,(SPH),(FIL),LINE,LOG,
          EXP(2,XFIXM,FLOATM,DSPFMT,FRAME,XLOC,MVBLOK,SCPSCL,HSTPLT.
     PLOTS THE ELEMENTS OF AN ARBITRARY NO. OF FLTG. PT. VECTORS (ALL OF
     SAME LENGTH N) EQUALLY SPACED ACROSS AN ARBITRARY NO. OF FRAMES
     (CONTROLLED BY NOPPP = NO. POINTS/PAGE, LAST POINT OF ONE FRAME BEING
     REPEATED AS FIRST POINT OF NEXT, 3 LSTHN = NOPPP LSTHN = 401).
     ISOL(1...NS) = VECTOR OF MACHINE LOCATIONS OF VECTORS TO BE PLOTTED
     IN SOLID MODE WITH ISOL(NS+1) = 0, SIMILARLY IDOT(1...ND+1) SPECIFIES
     VECTORS FOR DOTTED MODE (NS+ND MUST EXCEED ZERO).  SUCCESSIVE FRAMES
     SERIALIZED FROM IPAGE WHICH IS LEFT 1 GREATER THAN LAST INDEX USED.
     SPACE(1...N) USED FOR SCRATCH.  YTOP AND YBOT DEFINE TOP AND BOTTOM OF
     PLOTTING AREA (SAME UNITS AS VECTORS, YTOP GRTHN YBOT).  XMAX AND
     XMIN ARE ARBITRARY COORDINATES ASSOCIATED WITH NTH AND FIRST VECTOR
     ELEMENTS (XMAX GRTHN XMIN).  PLOTS ARE SUPPLIED WITH LABELLED AXES AND
     CONVENIENT CHECK MARKS IN USER UNITS.  TITLE(1...8) = 48 HOLLERITH
     FOR PAGE HEADING.  YUNITS(1...6) AND XUNITS(1...6) = 36 HOLLERITH EACH
     FOR LABELLING VERTICAL AND HORIZONTAL AXES.  ALTERNATIVELY THE 48
     HOLLERITH FOR HEADING CAN BE SET IN TITLE(1,0,-1,...,-7)  WITH
     TITLE(1) = 6H$$$$$$ AS FLAG, USING HOLLERITH FIELD IN CALLING
     SEQUENCE.  SIMILARLY FOR YUNITS, XUNITS.  SPACE(1) IS SET = 0.0 IF OK,
     = 1.0  IF ILLEGAL N, NOPPP, YTOP, XMAX, OR NO. OF VECTORS (ALSO
     COMMENT MADE ON SCOPE).  PLOTTING STYLE CONTROLLED BY SUBROUTINE HSTPLT
     OF WHICH THERE ARE SEVERAL ALTERNATIVE VERSIONS.  GRAPH DOES NOT CHNAGE
     FRAMES BEFORE PLOTTING ITS FIRST PAGE OR AFTER ITS LAST PAGE.

GRAPHX (ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,YBOT,          FORTRAN, 123 REGISTERS
        XMAX,XMIN,NOPPP,IPAGE,SPACE,NFRMZV)
          NO OTHER ENTRIES.  TRANSFER VECTOR - GRAPH,FRAME.
     FUNCTIONALLY EQUIVALENT TO CALL GRAPH (ISOL,...,SPACE) EXCEPT THAT
     PLOTS ARE EXPANDED OVER NFRMZV (EXCEEDS ZERO) FRAMES IN VERTICAL
     DIRECTION, YTOP NOW REFERING TO UPPER EDGE OF TOP ROW OF FRAMES, YBOT
     TO LOWER EDGE OF BOTTOM ROW OF FRAMES, AND THAT SPACE(2) SET = 2.0
     IF NFRMZV ILLEGAL.

GRUP2  (P,NDELX,DELX,XLO,YLIM,NWANT,IANS)                   FORTRAN, 201 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     GIVEN  P(1...NDELX) = HISTOGRAM TYPE PROBABILITY SUCH THAT
     P(I) = PROBABILITY DENSITY FOR VARIATE  X  IN RANGE  XLO+(I-1)*DELX
     TO  XLO+I*DELX,  (WITH SUM (FROM I=1 TO NDELX) OF (P(I)*DELX)  REQUIRED
     TO = 1.0), THEN GRUP2 SETS  XLIM(1...NWANT+1)  SUCH THAT INTEGRAL OF
     P(X) FROM XLIM(I) TO XLIM(I+1) EQUALS 1/NWANT, WITH XLIM(1) = XLO
     AND XLIM(NWANT+1) = XLO+DELX*NWANT, AND SETS IANS = 0  IF OK,
     = -1, -2, -3, OR -4 IF ILLEGAL NDELX (LSTHN 2), DELX (LSTHN= 0.),
     NWANT (LSTHN 2) OR SOMETHING WEIRD.
```

```
HALVL  (X,LX)                                      FAP, SECONDARY ENTRY OF DUBLX
     SETS X(1...LX) = 1/2 OF INPUT X(1...LX).  MAGNITUDE OF LX IS USED AND
     LX=0 TREATED AS LX=1 .

HALVX  (IX,LX)                                     FAP, SECONDARY ENTRY OF DUBLX
     SETS IX(1...LX) = 1/2 OF INPUT IX(1...LX).  MAGNITUDE OF LX IS USED AND
     LX=0 TREATED AS LX=1 .

HLADJ F(HOL)                                                 FAP,  46 REGISTERS
          OTHER ENTRY - HRADJ.  NO TRANSFER VECTOR.
     USAGE, HOLADJ=HLADJF(HOL), SETS HOLADJ = LEFT ADJUSTED FORM OF HOL
     TREATED AS 6 BCD CHARACTERS (SPACES ROTATED TO RIGHT END).

HRADJ F(HOL)                                      FAP, SECONDARY ENTRY OF HLADJ
     USAGE, HOLADJ=HRADJF(HOL), SETS HOLADJ = RIGHT ADJUSTED FORM OF HOL
     TREATED AS 6 BCD CHARACTERS (SPACES ROTATED TO LEFT END).

HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)             FAP, 145 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - LINEH,LINEV.
     PLOTS  NY(1...LNY) GIVEN IN SCOPE UNITS (0 TO 1023), EACH POINT
     PLOTTED AS HORIZONTAL BAR OF SCOPE LENGTH NDELX/128 EXCEPT BARS FOR
     END POINTS HALF AS LONG, WHERE LEFT  X  COORDINATE OF FIRST BAR
     GIVEN (FLTG.PT. SCOPE UNITS) IN  ORG(1), AND WITH ENDS OF SUCCESSIVE
     BARS CONNECTED BY VERTICAL BARS, ALL BARS BEING SOLID OR DOTTED
     AS  DOT = 0. OR NOT = 0. ALSO OPTIONALLY (YES IF AXES = 0., NO IF NOT)
     PLOTS SOLID HORIZONTAL AXIS FROM (X,Y)=(ORG(1),ORG(2)) TO (X,Y) =
     (ORG(3),ORG(2)) WITH VERTICAL CHECK MARKS AT MIDDLES OF BARS FOR
     NY(IFRSTB), NY(IFRSTB+ISKIPB), NY(IFRSTB+2*ISKIPB),... WHERE
     IFRSTB, ISKIPB GRTHN= 1 .

HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB) - II        FAP, 188 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - LINEH,LINEV.
     FUNCTIONS SIMILARLY TO HSTPLT BUT PLOTS NY(1...LNY) AS VERTICAL
     LINES FROM Y ORIGIN = NY(1) (REMEMBERED FROM FIRST CALL OF
     HSTPLT-II WITH AXIS = 0).  ORG(1...3), NDELX, DOT, AXIS HAVE
     SAME MEANING AS HSTPLT, BUT IFRSTB, ISKIPB IGNORED.

HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB) - III FAP(709), 256 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - LINEH.
                                                       FAP(7090), 258 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - LINEH.
     FUNCTIONS SIMILARLY TO HSTPLT BUT PLOTS NY(1...LNY) AS DARK POINTS
     WITH LIGHTER CUBIC CURVES INTERPOLATED BETWEEN POINTS.

HVTOIV (HV,LHV,IV)                                           FAP,  39 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  IV(1...6*LHV)  AS SPREAD OUT FORM OF  HV(1...LHV)  ASSUMED TO
     BE IN FORMAT (LHVA6), SO THAT EACH IV(I) IS INTEGER IN RANGE 0 TO 63.
     FUNCTION IS EXACT INVERSE OF SUBROUTINE IVTOHV.

IDERIV (YOFX1,DYDX,DELX,LY,YOFX)                             FAP,  54 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS YOFX(1) = YOFX1, YOFX(2) = DELX*DYDX(1) + YOFX(1),
     AND (IF LY GRTHN 2) YOFX(K) = 2*DELX*DYDX(K-1) + YOFX(K-2)
     FOR K = 3,4,...,LY.  EQUIV(YOFX,DYDX) OK.  STRAIGHT RETURN
     IF LY LSTHN 2 OR DELX=0.  IS EXACT INVERSE OPERATION TO THAT OF
     SUBROUTINE DERIVA.
```

```
IF     (X,NXNEG,NXZER,NXPOS)                          FAP, PSEUDO ENTRY OF SEVRAL
      USAGE IS CALL SEVRAL (...,2HIF,X,NXNEG,NXZER,NXPOS,...).
      FUNCTION IS SIMILAR TO FORTRAN STATEMENT IF(X) NXNEG,NXZER,NXPOS
      WHERE NX IS THE INDEX OF A SUBROUTINE RELATIVE TO THE IF SEQUENCE
      (NX NEGATIVE FOR PRIOR SUBROUTINES).  LOOPS WITHIN LOOPS MAY BE BUILT
      WITH PSEUDO IFS.

IFNCTN (YOFX,LYOFX,XFIRST,XLAST,LXOFY,                        FAP,  208 REGISTERS
        YLO,YHI,IERRLO,XOFY,IANS)
          NO OTHER ENTRIES.  TRANSFER VECTOR - MONOCK,REVER.
      SETS  XOFY(1...LXOFY)  SUCH THAT  Y(XOFY(1))=YLO, Y(XOFY(2))=YLO+DELY,
      Y(XOFY(3))=YLO+2*DELY, ..., Y(XOFY(LXOFY))=YHI,  WHERE  DELY =
      (YHI-YLO)/LXOFY-1)  AND WHERE THE FUNCTION  Y(X)  IS DEFINED BY STRAIGHT
      LINE SEGMENTS BETWEEN THE VALUES  Y(XFIRST)=YOFX(1), Y(XFIRST+DELX)=
      YOFX(2), Y(XFIRST+2*DELX)=YOFX(3), ..., Y(XLAST)=YOFX(LYOFX)  WITH
      DELX = (XLAST-XFIRST)/(LYOFX-1),  WHERE  YOFX(1...LYOFX)  MUST BE EITHER
      MONOTONE NON-INCREASING OR MONOTONE NON-DECREASING.  IF THE INVERSE
      FUNCTION  X(Y)  HAS A VERTICAL RISE OR DROP AT A REQUIRED  Y  VALUE THE
      MIDPOINT IS SELECTED FOR  XOFY.  REQUIRE  LYOFX  GRTHN=  2,  LXOFY
      GRTHN=  1,  XFIRST  NOT=  XLAST,  AND, IF  YMAX = MAX(YOFX(1...LYOFX)),
      YMIN = MIN(YOFX(1...LYOFX)),  YMIN  LSTHN=  YLO  LSTHN  YMAX  IF  LXOFY
      GRTHN=  2  BUT  YMIN  LSTHN=  YLO  LSTHN=  YMAX  IF  LXOFY = 1,  AND
      YLO  LSTHN  YHI  LSTHN=  YMAX  IF  LXOFY  GRTHN=  2 .  SETS  IANS = 0
      IF OK,  = IERRLO+K, K=0,1,3,4,5, OR 6  IF  YOFX,LYOFX,XLAST,LXOFY,YLO,
      OR  YHI  ILLEGAL.

IGETX  (IX,I1,I2,...,IN)       (FORTRAN FUNCTION) FAP, SECONDARY ENTRY OF GETX
      PERFORMS SAME FUNCTION AS GETX.

IINTGR (YOFX1,YIGRTD,DELX,LY,YOFX,CIGRTN)                     FAP,  49 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS YOFX(1) = YOFX1, AND (IF LY GRTHN 1) YOFX(K) = (2/DELX)*
      (YIGRTD(K) - YIGRTD(K-1)) - YOFX(K-1) FOR K=2...LY.
      EQUIV(YOFX,YIGRTD) OK.  STRAIGHT RETURN IF LY LSTHN 1 OR DELX=0.  (BUT
      DELX MAY BE NEGATIVE).  FUNCTION IS EXACT INVERSE TO THAT OF SUBROUTINE
      INTGRA.

INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,6HNAUXL1,            FORTRAN, 896 REGISTERS
        AUXL1,...,6HNAUXLN,AUXLN)
          NO OTHER ENTRIES.  TRANSFER VECTOR - VARARG,FSKIP,(TSB),(RLR),
          FAPSUM,LOC,MVBLOK,XSAME,(SPH),(FIL),(STH),UNPAKN.
      SEARCHES LOGICAL TAPE ITAPE (ASSUMED TO HAVE BEEN CREATED BY
      SUBROUTINE OUDATA) FOR RECORD NO.  IRECNO (IRECNO ANY MODE), EXCEPT
      THAT IF IRECNO = 0 IT MERELY GOES AFTER NEXT RECORD ON TAPE AND THEN
      SETS IRECNO = RECORD NO. FOUND.  IF THE RECORD IS FOUND (ALWAYS IF
      IRECNO = 0) AND IF ON INPUT NOPTS IS GRTHN = 0, INDATA SETS
      DATA(1...LREC) = RETRIEVED DATA (AS ORIGINALLY FED TO OUDATA),
      SETS NOPTS = LREC WHERE LREC HAS BEEN OBTAINED FROM THE TAPE, AND THEN
      PROCEEDS TO PROCESS ARGUMENTS BEYOND ERR (IF ANY).  IF HOWEVER NOPTS
      WERE LSTHN 0 ON INPUT THE SETTING OF DATA(1...LREC) IS OMITTED.
      (NOTE THAT DATA SHOULD BE DIMENSIONED TO LARGEST OF LREC+1 OR
      ABOUT 200.)  IF RECORD NO. NOT FOUND CONTROL RETURNS TO CALLING PROGRAM.
      THE N PAIRS OF ARGUMENTS BEYOND ERR ARE OPTIONAL.  (BUT  N  MUST NOT
      EXCEED 25).  FIRST OF EACH PAIR IS 6 HOLLERITH (1A6) NAMING DESIRED
      AUXILIARY INFORMATION AS ORIGINALLY FED TO OUDATA FOR THIS RECORD,
      AND THE SECOND IS STORAGE LOCATION FOR THE RETRIEVED INFO (MUST BE
```

(CONTINUED NEXT PAGE)

DIMENSIONED AS ORIGINALLY FED TO OUDATA), BUT THE ORDERING AND TOTAL NO. OF AUXILIARY REQUESTS NEEDN'T MATCH THOSE OF THE ORIGINAL OUDATA CALL. INDATA SETS AUXL1(1...),...,AUXLN(1...) ACCORDINGLY, EXCEPT OMISSIONS WILL OCCUR IF NAME CAN'T BE FOUND ON TAPE. SETS ERR = 0. IF ALL OK, = 1. IF 1 OR MORE AULX REQUESTS NOT FILLABLE, =2. IF SUMCK ERROR ON TAPE (DOESN'T STOP FILLING REQUESTS) = 3. IF TOO MANY DIFFERENT VALUES OF ITAPE HAVE OCCURRED (LIMIT PRESENTLY = 2), = 4. IF IRECNO NOT FOUND, = 5. IF ILLEGAL NO. ARGUMENTS IN CALL STATEMENT (MUST BE ODD), = 6. IF THERE ARE AN EXCESSIVE NO. OF RECORDS ON THE TAPE (PRESENT LIMIT = 200). ALSO ON-LINE ERROR PRINT OCCURS.

INDEX F(I,ICRTCL) FAP, 50 REGISTERS

OTHER ENTRIES - VINDEX,SETEST,SETAPT,CHUSET. NO TRANSFER VECTOR. ADDS 1 TO MACHINE LOCATION CONTAINING I THEN SETS ACCUMULATOR = -1.0 IF NEW I LSTHN ICRTCL, = 0.0 IF NEW I = ICRTCL, = +1.0 IF NEW I GRTHN ICRTCL, WHERE +0 AND -0 TREATED AS EQUAL. NOT RELATED TO XINDEX FUNCTION.

INTGRA (CIGRTN,YOFX,LY,DELX,YIGRTD,YOFX1) FAP, 47 REGISTERS

NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS YIGRTD(1) = CIGRTN, AND (IF LY GRTHN 1)
YIGRTD(K) = YIGRTD(K-1) + DELX*(YOFX(K) + YOFX(K-1))/2.0
FOR K = 2...LY, AND YOFX1 = YOFX(1). EQUIV(YIGRTD,YOFX) OK.
STRAIGHT RETURN IF LY LSTHN 1 OR DELX = 0. (MAY BE NEGATIVE).
FUNCTION IS EXACT INVERSE TO THAT OF IINTGR.

INTHOL (NHOL,HOL,FMT,NDATAD,NDATAA,DATA) FAP, 72 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - FNDFMT,(IOH),(RTN). INTERPRETS A VECTOR OF HOLLERITH WORDS HOL(1...NHOL) (NHOL GRTHN= 1) ACCORDING TO A FORMAT FMT(1...) TO ATTEMPT TO FIND NDATAD (GRTHN= 1) DATA VALUES. THE DATA VALUES FOUND WHILE MAKING ONE SCAN OF HOL AND FMT ARE STORED IN DATA(1...NDATAA).

INTMSB FAP, SECONDARY ENTRY OF TIMSUB

SEE ABSTRACT OF TIMSUB BELOW.

INTOPR (NDATA,XLO,DELX,X,OPER) FAP, 111 REGISTERS

NO OTHER ENTRIES. NO TRANSFER VECTOR. SETS OPER(1...NDATA) SUCH THAT, FOR ANY DATA VALUES D(1...NDATA), THE SUM (FROM I = 1 TO NDATA) OF OPER(I)*D(I) WILL EQUAL P(X), WHERE P IS THE EXACT FITTING POLYNOMIAL OF DEGREE NDATA-1 SATISFYING P(XLO+(I-1)*DELX) =D(I) FOR I=1...NDATA. REQUIRE NDATA = 1,2,3, OR 4 AND DELX NOT= 0.0 . STRAIGHT RETURN WITH NO OUTPUT FOR ILLEGAL NDATA OR DELX.

INTSUM (X,LX,XISUMD) FAP, 27 REGISTERS

OTHER ENTRY - XNTSUM. NO TRANSFER VECTOR.
SETS XISUMD(I) = SUM (FROM J = 1 TO I) OF X(I), I=1...LX.
EQUIV(XISUMD,X) OK. STRAIGHT RETURN IF LX LSTHN 1.

IPLYEV (LA,A,X,Y,EVR,EVI) FORTRAN, 98 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - (IFMP).
IF Z=X+I*Y (I=SQRT(-1)), IPLYEV SETS EVR AND EVI WHERE
EV = EVR+I*EVI = SUM (FROM K = 1 TO LA) OF (A(K)*Z(TO THE K-1)).
LA MUST BE GRTHN= 2 .

ITOMLI (IV,LIV,MLIV,IANS) FAP, 37 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS MLIV(1...LIV) = IV(1...LIV) SHIFTED RIGHT 18 PLACES ARITHMETICALLY. EQUIV(IV,MLIV) OK. SETS IANS = 0 IF OK, = -1 IF LIV LSTHN 1 .

IVTOHV (IV,LHV,HV) FAP, 70 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS HV(1...LHV), BY PACKING IV(1...6*LHV) 6 AT A TIME USING ONLY BITS 12-17 OF IV(I). IV(1,2,...,6) GOES TO HV(1) (BITS S 1-5, 6-11, ..., 30-35), ETC. STRAIGHT RETURN IF LHV LSTHN 1 . TURNS OFF AC OVERFLOW INDICATOR. FUNCTION IS EXACT INVERSE TO SUBROUTINE HVTOIV. EQUIV (IV,HV) OK.

IXCARG (ARG,IXCOM) FORTRAN, 35 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - XLOC.
SETS IXCOM = INDEX WITH RESPECT TO COMMON OF ARG.

KIINT1 (CHISQ,NDF,PROB,IANS) FORTRAN, 191 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - SQRT,EXP(3,NOINT1.
SETS PROB = PROBABILITY THAT CHI-SQUARE WILL EXCEED CHISQ FOR NDF DEGREES OF FREEDOM. SETS IANS = 0 IF OK, = 1 IF CHISQ LSTHN 0., = 2 IF NDF LSTHN 1 .

KOLAPS (XMID,M,TYPE,L,CMID,ERR) FAP, 100 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS CMID(-L+1,...,L+1) = C(-L,...,+L) WHERE C(I) = X(I) + X(I+2*L) + X(I-2*L) + X(I+4*L) + X(I-4*L) + ... FOR I = -(L-1) ... L-1 AND C(-L), C(L) = ONE HALF OF ABOVE EXPRESSION, WHERE THE X SERIES X(-M...+M) IS GIVEN IN XMID(-M+1...M+1) . TYPE = 0.0 SIGNIFIES X IS FXD.PT., NOT = 0. SIGNIFIES FLTG.PT. SETS ERR = 0. IF OK, = 1.0 IF L LSTHN 1 OR M LSTHN 0, = 2. IF OVERFLOW OCCURS. (L MAY EXCEED M) EQUIV(CMID,XMID) OK.

LIMITS (IANSX1,IANS, X1,X1A,X1B, X2,X2A,X2B, ..., XN,XNA,XNB) FAP, 44 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS IANS=0 IF XJLO LSTHN= XJ LSTHN= XJHI FOR J=1...N WHERE J IS TRIPLET INDEX AND XJLO=MIN(XJA,XJB) XJHI=MAX(XJA,XJB), BUT SETS IANS=IANSX1+K-1 IF XK FAILS TO LIE IN CLOSED RANGE XKLO TO XKHI WHERE K IS THE LOWEST SUCH INDEX. MODES OF ARGUMENTS IMMATERIAL. PLUS AND MINUS ZERO ARE TREATED EQUAL IN THE COMPARISONS. N SHOULD EXCEED ZERO AND ARGUMENT COUNT BE 2+3*N (OTHERWISE ILLEGAL RETURN).

LINE (X1,Y1,X2,Y2) FAP (709) , 91 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
FAP (7090), 95 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
PLOTS STRAIGHT LINE ON SCOPE FROM (X1,Y1) TO (X2,Y2) WHERE X1,Y1,X2,Y2 ARE IN FLTG.PT. SCOPE UNITS (0. TO 1023.). SEPARATION BETWEEN INDIV. PTS. ON LINE WILL LIE BETWEEN 1.414 AND 2.0 SCOPE UNITS. PLOTTING OMITTED IF X1,Y1,X2, OR Y2 ILLEGAL.

```
LINEH  (NXLEFT,NYLEFT,NXRITE,NDELX)                        FAP (709) ,   34 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
                                                           FAP (7090),   35 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      PLOTS STRAIGHT HORIZONTAL LINE ON SCOPE FROM (NXLEFT,NYLEFT) TO
      (NXRITE,NYLEFT) WHERE ALL ARGUMENTS IN FXD.PT. SCOPE UNITS (SHOULD BE
      0 TO 1023, ARE TREATED MODULO 1024), AND WHERE SEPARATION BETWEEN
      INDIVIDUAL POINTS WILL BE NDELX UNITS.  NXRITE SHOULD EXCEED NXLEFT AND
      DELX EXCEED 0.

LINEV  (NXBOT,NYBOT,NYTOP,NDELY)                           FAP (709) ,   34 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
                                                           FAP (7090),   35 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      PLOTS STRAIGHT VERTICAL LINE ON SCOPE FROM (NXBOT,NYBOT) TO (NXBOT,
      NYTOP) WITH NDELY UNITS BETWEEN INDIVIDUAL POINTS, WHERE ALL
      ARGUMENTS ARE IN FXD.PT. SCOPE UNITS (SHOULD BE 0 TO 1023, ARE TREATED
      MODULO 1024).  NYTOP SHOULD EXCEED NYBOT AND NDELY EXCEED 0 .

LINTR1 (X,XLO,DELX,TABLE,NTABLE,YOFX)                          FORTRAN, 96 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      GIVEN  TABLE(1...NTABLE) CORRESPONDING TO ARGUMENT VALUES XLO,XLO+DELX,
      ...,XLO+(NTABLE-1)*DELX, LINTR1 SETS YOFX = LINEARLY INTERPOLATED
      VALUE FROM THE TABLE CORRESPONDING TO ARGUMENT VALUE = X .
      X MUST LIE IN RANGE OF TABLE ARGUMENTS, DELX MUST EXCEED 0., AND
      NTABLE EXCEED 1 .

LISTNG (ITAPE,JTAPE,DATA)                                     FORTRAN, 755 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - (RWT),(STH),(FIL),(TSB),(RLR),
           FAPSUM,SAME,XSAME,(SPH),FSKIP,SHFTR2.
      MAKES A LISTING ON OUTPUT TAPE JTAPE OF THE RECORD NOS. AND
      AUXILIARY INFO FROM THE INDATA-OUDATA TYPE TAPE ON LOGICAL ITAPE
      AND CHECKS SUMCHECKS.  ITAPE IS LEFT REWOUND BUT JTAPE IS NEVER
      MOVED BACKWARDS.  DATA(1...MAX) IS USED FOR SCRATCH WHERE MAX = 1+LENGTH
      OF LONGEST RECORD ON ITAPE.

LOC    (VAR,IADD)                                                  FAP,   4 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS IADD = MACHINE ADDRESS OF VAR.

LOCATE (SUBRU1,SUBRU2,...,SUBRUN)                                FAP, 512 REGISTERS
           OTHER ENTRIES - WHERE,CALL,CALL2,SETSBV,SETUP,RETURN,XINDEX,ARG,
           XARG,STORE,XNARGS,XNAME.  NO TRANSFER VECTOR.
      USAGE IS  CALL LOCATE(SUBRU1,SUBRU2,...,SUBRUN)
                CALL SUBR1(ARG11,ARG12,...,ARG1M1)
                      ETC.
                CALL SUBRN(ARGN1,ARGN2,...,ARGNMN)
      THEN LOCATE ESTABLISHES A 1-1 EQUIVALENCE BETWEEN THE REAL
      SUBROUTINE NAMES  SUBR1,...,SUBRN  AND THE PROXY NAMES
      SUBRU1,...,SUBRUN  FOR USE IN LATER CALL CALL, CALL CALL2, OR CALL WHERE
      STATEMENTS.  CONTROL RETURNS BEYOND CALL SUBRN STATEMENT.  THE ARGUMENT
      LISTS (ARG11 ETC.) ARE OPTIONAL (SUBROUTINE WHERE MAY USE THEM LATER).
      MAX NO. OF CALL LOCATE STATEMENTS IS 14.
```

LSHFT F(N,X) FAP, 12 REGISTERS
OTHER ENTRY - XLSHFT. NO TRANSFER VECTOR.
LOGICALLY SHIFTS X N BINARY PLACES (SHIFT IS LEFT IF N NEGATIVE).

LSLINE (YY,LY,XMIN,XMAX,CO,C1) FORTRAN, 117 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS CO AND C1 SUCH THAT (Y(XMIN)-CO-C1*XMIN)**2+...+
(Y(XMAX)-CO-C1*XMAX)**2 IS MINIMUM, GIVEN YY(1)=Y(XMIN),
YY(2)=Y(XMIN+DX),...,YY(LY)=Y(XMAX), WHERE DX=(XMAX-XMIN)/(LY-1).
LY MUST EXCEED 1 .

LSSS1 (L,A,R,G,F,ALPHA) FORTRAN, 122 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FDOT.
SOLVES THE EQUATION SUM (FROM I=1 TO L) OF F(L-I+1)*R(I-K+1) =
G(L-K+2) GIVEN THE SOLUTION OF SUM (FROM I=1 TO L) OF
F(L-I+1)*R(I-K+1) = G(L-K+1) WHERE R(1...L) IS ONE SIDE OF AN
AUTOCORRELATION VECTOR (R(1) IS THE CENTER TERM), A(1...L) IS THE
LEAST SQUARE PREDICTION ERROR OPERATOR FOR R, G(1...L+1) IS A SECTICN
OF A CROSSCORRELATION VECTOR, AND ALPHA IS THE EXPECTED ERROR
CORRESPONDING TO A. L MUST BE GRTHN= 2 .

MATINV (NRA,A,AINV,SPACE,ERR) FORTRAN, 90 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - SIMEQ
SETS AINV(1...NRA*NRA) = MATRIX INVERSE OF A(1...NRA*NRA).
EQUIVALENCE (A,AINV) OK. ERR = 0. IF ALL OK, = 1. IF OVERFLOW
OCCURS, = 2. IF A IS SINGULAR. SPACE(1...(NRA+1)*NRA) IS SCRATCH.
NRA MUST BE GRTHN 0, BUT IS NOT CHECKED.

MATML1 (NRABC,AA,BB,CC,ZIFSTO) FAP, 61 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS CC = CM WHERE CM = MATRIX PRODUCT OF AA AND BB IF
ZIFSTO = 0. SETS CC = CC+CM IF ZIFSTO NOT= 0. AA(1...NRABC*NRABC),
BB(1...NRABC*NRABC), AND CC(1...NRABC*NRABC) ARE NRABC BY NRABC
MATRICES ALL STORED BY EITHER ROWS OR COLUMNS. EQUIVALENCE (AA,BB)
OK. NRABC MUST BE GRTHN= 1 BUT THIS IS NOT CHECKED.

MATML3 (NRAC,NCARB,NCBC,AA,BB,ZIFNTR,CC,ZIFSTO) FORTRAN, 120 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - DOTJ.
SETS CC = CM WHERE CM = MATRIX PRODUCT OF AA AND BB IF
ZIFSTO = 0. SETS CC = CC+CM IF ZIFSTO NOT= 0. AA(1...NRAC*NCARB)
IS AN NRAC BY NCARB MATRIX STORED BY COLUMNS. BB(1...NCARB*NCBC)
IS AN NCARB BY NCBC MATRIX STORED BY COLUMNS, IF ZIFNTR = 0., OR
BY ROWS, IF ZIFNTR NOT= 0. CC(1...NRAC*NCBC) IS AN NRAC BY NCBC
MATRIX STORED BY COLUMNS. ROUTINE RETURNS (CC MAY BE SET TO ZERO) IF
NRAC, NCARB, NCBC ARE LSTHN= 1 . EQUIVALENCE (AA,BB) OK.

MATRA (A,N,M,ATRAN) FAP, 92 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS ATRAN(1...N*M) = TIGHT PACKED TRANSPOSE OF THE TIGHT PACKED MATRIX
A(1...M*N) WHERE A (STORED BY COLUMNS) HAS N ROWS AND M COLUMNS. BIT 35
IS SET = 0 THROUGHOUT ATRAN. EQUIV(ATRAN,A) IS OK. N AND M MUST BE
GRTHN= 1 .

```
MATRA1 (NRCA,AA)                                                   FAP, 42 REGISTERS
         NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      STORES THE TRANSPOSE OF THE SQUARE MATRIX  AA(1...NRCA*NRCA)  ONTO
      ITSELF.  NRCA  MUST BE GRTHN=  1  (NOT CHECKED).

MAXAB  (LX,X,XMAX2,I)                                FAP, SECONDARY ENTRY OF MAXSN
      SETS  XMAX2  AND  I,  GIVEN  X(1...LX), SUCH THAT  XMAX2=X(I)
      WHERE MAGNITUDE OF  X(I)  IS GREATEST MAGNITUDE OF  X(1...LX).
      (NOTE XMAX2 MAY BE NEGATIVE.)  LX MUST EXCEED 0 .  X MAY BE ANY MODE.

MAXABM (FOFIJ,LI,LJ,IDIMEN,                         FAP, SECONDARY ENTRY OF MAXSNM
        FMAXAB,IMAXAB,JMAXAB)
      SETS  FMAXAB, IMAXAB, JMAXAB  SUCH THAT  FMAXAB = FOFIJ(IMAXAB,JMAXAB)
      SATISFIES  MAGNITUDE(FMAXAB)  GRTHN=  FOFIJ(I,J)  FOR  I=1...LI,
      J=1...LJ  WHERE USER HAS  DIMENSION  FOFIJ(IDIMEN,IGNORD).  FOFIJ  MAY
      BE FIXED OR FLOATING.  LI  AND  LJ  MUST EXCEED ZERO, AND  IDIMEN
      GRTHN=  LI  (NOT CHECKED).

MAXSN  (LX,X,XMAX1,I)                                              FAP,  54 REGISTERS
         OTHER ENTRIES - MINSN,MAXAB,MINAB.  NO TRANSFER VECTOR.
      SETS  XMAX1  AND  I, GIVEN  X(1...LX), SUCH THAT  XMAX1=X(I) IS
      GRTHN= ALL OTHER X(1...LX).  LX MUST EXCEED 0 .  X MAY BE ANY MODE.

MAXSNM (FOFIJ,LI,LJ,IDIMEN,FMAXSN,IMAXSN,JMAXSN)                 FAP,   61 REGISTERS
         OTHER ENTRIES - MINSNM,MAXABM,MINABM.  NO TRANSFER VECTOR.
      SETS  FMAXSN, IMAXSN, JMAXSN  SUCH THAT  FMAXSN = FOFIJ(IMAXSN,JMAXSN)  I
      IS  GRTHN=  FOFIJ(I,J)  FOR  I=1...LI,  J=1...LJ  WHERE  CALLER HAS
      DIMENSION  FOFIJ(IDIMEN,IGNORD).  FOFIJ  MAY BE FIXED OR FLOATING.  LI
      AND  LJ  MUST EXCEED ZERO,  AND  IDIMEN  GRTHN=  LI  (NOT CHECKED).

MDOT   (NRCAB,LAB,AA,BB,DOT,MIFREV)                           FORTRAN, 109 REGISTERS
         NO OTHER ENTRIES.  TRANSFER VECTOR - MATML1.
      SETS  DOTM  =  MATRIX PRODUCTS  A(1)*B(1) + ... + A(LAB)*B(LAB)  IF
      MIFREV  GRTHN=  0,  OR  =  MATRIX PRODUCTS  A(1)*B(LAB) + ...
      + A(LAB)*B(1)  IF  MIFREV  LSTHN  0 .  A(I)  REPRESENTS THE  NRCAB  BY
      NRCAB  MATRIX STORED BEGINNING AT  AA(1+NRCAB*NRCAB*(I-1)),  B(I)
      REPRESENTS THE  NRCAB  BY  NRCAB  MATRIX STORED BEGINNING AT
      BB(1+NRCAB*NRCAB*(I-1)),  AND  DOTM  REPRESETNS THE  NRCAB  BY  NRCAB
      MATRIX  DOT(1...NRCAB*NRCAB).  EQUIVALENCE  (AA,BB)  OK.  NRCAB,  LAB
      MUST BE GRTHN=  1 .

MDOT3  (NRAD,NCARB,NCBD,LAB,AA,BB,ZIFNTR,DOT,MIFREV)    FORTRAN,  122 REGISTERS
         NO OTHER ENTRIES.  TRANSFER VECTOR - MATML3.
      SETS  DOTM  =  MATRIX PRODUCTS  A(1)*B(1) + ... + A(LAB)*B(LAB)  IF
      MIFREV  GRTHN=  0,  OR  =  MATRIX PRODUCTS  A(1)*B(LAB) + ...
      + A(LAB)*B(1)  IF  MIFREV  LSTHN  0 .  A(I)  REPRESENTS THE  NRAD  BY
      NCARB  MATRIX STORED BY COLUMNS BEGINNING AT  AA(1+NRAD*NCARB*(I-1)),
      B(I)  REPRESENTS THE  NCARB  BY  NCBD  MATRIX STORED BY COLUMNS, IF
      ZIFNTR = 0.,  OR BY ROWS, IF  ZIFNTR NOT= 0.,  BEGINNING AT
      BB(1+NCARB*NCBD*(I-1)),  AND  DOTM  REPRESENTS THE  NRAD  BY  NCBD
      MATRIX  DOT(1...NRAD*NCBD).  EQUIVALENCE  (AA,BB)  OK.  NRAD,  NCARB,
      NCBD,  LAB  MUST BE GRTHN=  1  (NOT CHECKED).
```

MEMUSE (ITPOUT) FORTRAN, 71 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - XLCOMN,(STH),(FIL).
PRINTS ONE LINE (COLUMNS 2 THRU 96) ON LOGICAL TAPE ITPOUT GIVING (IN DECIMAL) PROGRAM STORAGE, DIMENSIONED COMMON STORAGE, AND AVAILABLE COMMON STORAGE. DOES NOT CHECK ITPOUT.

MFACT (NRA,AA,AFACT) FORTRAN, 187 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - DOTJ,STZ,SQRT.
FINDS THE UPPER TRIANGULAR MATRIX AFACT(1...NRA*NRA) SUCH THAT THE MATRIX PRODUCT (AFACT * AFACT TRANSPOSE) = AA. NRA MUST BE GRTHN= 1 .

MIFLS (NRC,LL,BB,RR,GG,FF,C) FORTRAN, 276 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - MATML3,MOVREV.
SOLVES THE EQUATION SUM (FROM I=1 TO LL) OF F(LL-I+1)*R(I-J+1) = G(LL-J+1) FOR J=1...LL GIVEN THE SOLUTION FOR THE LOWER ORDER EQUATION SUM (FROM I=1 TO LL-1) OF F1(LL-I+1)*R(I-J+1) = G(LL-J) FOR J=1...LL-1 AND THE OTHER INPUTS. BB, RR, GG, AND FF ARE ALL NRC BY NRC MATRIX VALUED VECTORS OF LENGTH LL. BB IS THE LEAST-SQUARE OPTIMUM HINDSIGHT OPERATOR FOR THE AUTOCORRELATION RR (SEE MIPLS). RR(1+NRC*NRC*(I-1)) = R(I) IS THE AUTOCORRELATION OF AN NRC BY M WAVELET (OR STATIONARY TIME SERIES). R(1) REPRESENTS THE MIDDLE TERM. GG(1+NRC*NRC*(I-1)) = G(I) IS PART OF A CROSS CORRELATION SERIES. FF(1...NRC*NRC*(LL-1)) = F1(1...LL-1) ON ENTRANCE. FF(1...NRC*NRC*LL) = F(1...LL) ON RETURN. C(1...6*NRC*NRC) CONTAINS SOME EXPECTED ERROR MATRICES (GIVEN BY MIPLS) AND SCRATCH SPACE. NRC AND LL MUST BE GRTHN= 1 .

MINAB (LX,X,XMIN2,I) FAP, SECONDARY ENTRY OF MAXSN
SETS XMIN2 AND I, GIVEN X(1...LX), SUCH THAT XMIN2 = X(I) WHERE MAGNITUDE OF X(I) IS SMALLEST MAGNITUDE OF X(1...LX). (NOTE XMIN2 MAY BE NEGATIVE.) LX MUST EXCEED 0 . X MAY BE ANY MODE.

MINABM (FOFIJ,LI,LJ,IDIMEN, FMINAB,IMINAB,JMINAB) FAP, SECONDARY ENTRY OF MAXSNM
SETS FMINAB, IMINAB, JMINAB SUCH THAT FMINAB = FOFIJ(IMINAB,JMINAB) SATISFIES MAGNITUDE(FMINAB) LSTHN= FOFIJ(I,J) FOR I=1...LI, J=1...LJ WHERE USER HAS DIMENSION FOFIJ(IDIMEN,IGNORD). FOFIJ MAY BE FIXED OR FLOATING. REQUIRE LI AND LJ GRTHN= 1 AND IDIMEN GRTHN= LI (NOT CHECKED).

MINSN (LX,X,XMIN1,I) FAP, SECONDARY ENTRY OF MAXSN
SETS XMIN1 AND I, GIVEN X(1...LX), SUCH THAT XMIN1=X(I) IS LSTHN= ALL OTHER X(1...LX). LX MUST EXCEED 0 . X MAY BE ANY MODE.

MINSNM (FOFIJ,LI,LJ,IDIMEN, FMINSN,IMINSN,JMINSN) FAP, SECONDARY ENTRY OF MAXSNM
SETS FMINSN, IMINSN, JMINSN SUCH THAT FMINSN = FOFIJ(IMINSN,JMINSN) LSTHN= FOFIJ(I,J) FOR I=1...LI, J=1...LJ WHERE USER HAS DIMENSION FOFIJ(IDIMEN,IGNORD). FOFIJ MAY BE FIXED OR FLOATING. REQUIRE LI AND LJ GRTHN= 1 AND IDIMEN GRTHN= LI (NOT CHECKED).

```
MIPLS  (NRC,LL,AA,BB,RR,C,ERR)                          FORTRAN, 571 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - IXCARG,MATINV,MATML3,MATRA,
          MDOT3,MOVREV,STZ.
     SOLVES THE EQUATIONS  SUM  (FROM I=1 TO LL)  OF  A(LL-I+1)*R(I-J+1)
     =  0.  AND  SUM  (FROM I=1 TO LL)  OF  B(I)*R(I-J+1)  =  0.  FOR
     J=1...LL  GIVEN THE SOLUTIONS TO THE EQUATIONS  SUM  (FROM I=1 TO LL-1)
     OF  A1(LL-I)*R(I-J+1)  =  0.  AND  SUM  (FROM I=1 TO LL-1)  OF
     B1(I)*R(I-J+1)  =  0.  FOR  J=1...LL-1,  AND THE EXPECTED ERRORS
     FOR  A1  AND  B1  (SEE BELOW).  AA(1...NRC*NRC*LL)  =  A(1...LL),
     BB(1...NRC*NRC*LL)  =  B(1...LL),  RR(1...NRC*NRC*LL)  =  R(1...LL),
     AND  C(1...4*NRC*NRC)  ARE VECTORS OF  NRC  BY  NRC  MATRICES.
     AA(1...NRC*NRC*(LL-1))  =  A1(1...LL-1)  IS THE LEAST-SQUARE OPTIMUM
     PREDICTION OPERATOR, AND  BB(1...NRC*NRC*(LL-1))  =  B1(1...LL-1)
     IS THE LEAST-SQUARE OPTIMUM HINDSIGHT OPERATOR FOR THE AUTOCORRELATION
     RR  (NOTE THAT  R(1)  IS THE CENTER TERM).  LET  C(1...4*NRC*NRC)  =
     CM(1...4),  THEN  CM(1)  CONTAINS THE EXPECTED ERROR FOR  AA,  CM(2)
     CONTAINS THE EXPECTED ERROR FOR  BB,  CM(3)  CONTAINS  CM(1) INVERSE,
     AND  CM(4)  CONTAINS  CM(2) INVERSE.  C(1+4*NRC*NRC...NRC+5*NRC*NRC)  IS
     SCRATCH.  LL  MUST BE GRTHN=  0,  NRA  MUST BE GRTHN=  1 .  ERR = 0.  IF
     ALL OK,  = 1.  IF  CM(1)  OR  CM(2)  SINGULAR (THEORETICALLY
     IMPOSSIBLE),  = 2.  IF OVERFLOW OCCURS WHILE INVERTING  CM(1)  OR
     CM(2),  = 3.  IF  LL  LSTHN  0 .  LL  BUMPED UP BY  1 .  (THE WRITEUP
     ASSUMES THE NEW VALUE OF  LL  AS THE LIMITS).

MISS   (NRC,LL,AA,BB,RR,GG,FF,C)                       FORTRAN,  335 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - MATML3,MDOT3,MOVREV.
     SOLVES THE EQUATION  SUM  (FROM I=1 TO LL)  OF  F(LL-I+1)*R(I-J+1)  =
     G(LL-J+1)  GIVEN THE SOLUTION  SUM  (FROM I=1 TO LL)  OF
     F(LL-I+1)*R(I-J+1)  =  G(LL-J)  FOR  J=1...LL  AND THE OUTPUTS OF  MIPLS
     AA,  BB,  AND  C.  SEE THE WRITEUP OF  MIPLS  FOR AN EXPLANATION OF  F,
     R,  G,  NRC,  LL,  AA,  BB,  RR,  GG,  FF,  AND  C.
     C(4*NRC*NRC+1...6*NRC*NRC)  IS SCRATCH.

MLISCL (MLIV,LMLIV,ISCALE,MLIVSC,IANS)                      FAP,  47 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  MLIVSC(1...LMLIV) = ISCALE*MLIV(1...LMLIV) ASSUMING ISCALE
     IS FORTRAN INTEGER AND MLIV IS MACHINE LANGUAGE INTEGER VECTOR.
     EQUIV(MLIVSC,MLIV) OK.  SETS IANS = 0 IF ALL OK, = -1 IF LMLIV LSTHN 1,
     = - 2 IF OVERFLOW OCCURS.

MLI2A6 (MLI,MLIHOL,NCRS)                                    FAP, 128 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  MLIHOL(1...2) = 12 HOLLERITH (FORMAT(2A6)) REPRESENTING MLI
     CONSIDERED AS A MACHINE LANGUAGE INTEGER.  THE 12 HOLLERITH ARE
     RIGHT ADJUSTED WITH LEADING ZEROES AND PLUS SIGN SUPPRESSED.  SETS
     NCRS = NO. NON-BLANK HOLLERITH (INCL. MINUS SIGN IF PRESENT).

MONOCK (X,LX,ZFNDCR,IANSNG,IANS)                           FAP,   48 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  IANS = 0  IF  LX = 1  OR IF  X(1...LX)  SATISFIES TEST  X(I)
     GRTHN=  X(I-1)  I=2...LX  IN THE CASE  ZFNDCR = 0.0,  OR SATISFIES TEST
     X(I)  LSTHN=  X(I-1)  I=2...LX  IN THE CASE  ZFNDCR  NOT=  0.0,  BUT
     SETS  IANS = IANSNG  IF TEST MADE AND FAILS.  PLUS AND MINUS ZERO
     TREATED EQUAL.  STRAIGHT RETURN WITH NO OUTPUT IF  LX  LSTHN=  0 .
```

```
MOUT   (ITAPE,NSPACE,X,XNAME,NRX,NCX,LX)                    FORTRAN,  130 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - CARIGE,(FIL),(STH).
      WRITES A VECTOR OF CLOSELY SPACED MATRICES  X(1...NRX,1...NCX,1...LX)
      ON LOGICAL TAPE  ITAPE  USING A FIXED  G  FORMAT WITH  5  VALUES PER
      LINE.  THE ARRAY IS PRECEEDED BY  NSPACE  BLANK SPACES (PAGE IS RESTORED
      IF  NSPACE  IS NEGATIVE) AND A LABEL CONSTRUCTED FROM THE  6  HOLLERITH
      CHARACTERS IN  XNAME.

MOUTAI (ITAPE,NSPACE,FOFIJ,FNAME,LI,LJ,                     FORTRAN,  357 REGISTERS
        IDIMEN,NDIGS,SCALE,SPACE)
          NO OTHER ENTRIES.  TRANSFER VECTOR - CARIGE,GNHOL2,MAXABM,RND,MCVE,
          MULPLY,FIXVR,SAME,LOG,EXP(2,(FIL),(STH).
      CREATES  NSPACE  BLANK LINES ON LOGICAL  ITAPE  (OR PAGE RESTORE IF
      NSPACE  LSTHN  0)  FOLLOWED BY A HEADING LINE INVOLVING SIX HOLLERITH
      FROM  FNAME  (FORMAT(1A6))  AND DESCRIBING SCALING USED.  THEN PRINTS
      A SCALED AND FIXED FORM OF  FOFIJ(1...LI,.1..LJ),  WHERE USER HAS
      DIMENSION  FOFIJ(IDIMEN,IGNORD),  COLUMNS OF FOFIJ  (I.E., FIXED  J
      VALUES) BEING PRINTED ALONG OUTPUT ROWS EACH OF WHICH IS LABELLED WITH
      ITS  J  VALUE AND  FOFIJ  BEING FLOATING POINT UNLESS  SCALE = 0.0,
      WHERE THE USER CONTROLS THE FIELD WIDTH  =  NDIGS+1  (THE NUMBER OF
      WORDS PER LINE BECOMES  60,40,30,25, OR 20  ACCORDING AS  NDIGS = 1,2,3,
      4, OR 5),  AND THE SCALING BY  SCALE.  IF  SCALE = 0.0  FOFIJ  IS
      ASSUMED ALREADY FIXED POINT COMPATIBLE WITH  NDIGS.  IF  SCALE  GRTHN
      0.0  THE OUTPUT INTEGERS WILL BE  (SCALE*FOFIJ)  ROUNDED TO NEAREST
      INTEGERS.  IF  SCALE = -1.0,  MOUTAI  SCALES BY THAT POWER OF TEN WHICH
      WILL GIVE  10**(NDIGS-1)  LSTHN=  MAXMAG  LSTHN  10**NDIGS  WHERE
      MAXMAG  IS THE LARGEST OUTPUT MAGNITUDE.  IF  SCALE = -2.0  MOUTAI
      SCALES SO THAT  MAXMAG = 10**(NDIGS-1).  ORIGINAL MATRIX  FOFIJ  LEFT
      UNDISTURBED.  SPACE(1...LI+1)  MUST BE AVAILABLE FOR SCRATCH.  ITAPE,
      LI,LJ,  SHOULD EXCEED  0,  ITAPE  LSTHN=  20,  IDIMEN  GRTHN=  LI,  AND
      NDIGS = 1,2,3,4, OR 5,  BUT NONE OF THESE ARE CHECKED.

MOVE   (N,SOURCE,DEST)                                          FAP,  32 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  DEST(1...N) = SOURCE(1...N).  ALL TYPES OF OVERLAP OF VECTORS
      SOURCE AND DEST ARE PERMITTED.  VECTORS CAN BE ANY MODE.  STRAIGHT
      RETURN IF  N  LSTHN 1 .

MOVECS (LXY1,X1,Y1,...,LXYN,XN,YN)                              FAP,  24 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - MOVE.
      SETS  Y1(1...LXY1)=X1(1...LXY1),  ...,  YN(1...LXYN)=XN(1...LXYN).
      EQUIV(XJ,YK) OK FOR ANY  J,K.  YK  UNDISTURBED IF  LXYK  LSTHN 1.
      VECTORS MOVED IN SAME ORDER AS THEY APPEAR IN CALLING SEQUENCE.

MOVREV (LXY,IX,X,IYMIFR,Y,SIGN)                                  FAP, 74 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      MOVES  X(1), X(1+IX), ..., X(1+(LXY-1)*IX)  TO  Y(1), Y(1+IY),...,
      Y(1+(LXY-1)*IY)  WHERE  IY  =  ABSOLUTE VALUE OF  IYMIFR.  IF  IYMIFR
      LSTHN  0,  THE STORAGE ORDER IS REVERSED WHILE MOVING.  IF  SIGN  LSTHN=
      0.,  THE SIGN IS CHANGED WHILE MOVING.  ROUTINE RETURNS IF  LXY  LSTHN
      1,  IX  LSTHN  0 .  OVERLAP MAY OCCUR ONLY IF  IX  =  IYMIFR  =  1 .
```

* MPSEQ1 TO MULPLY *
****

```
MPSEQ1 (X,LX,B,LB,IX,IXLO,IANS)                              FAP, 110 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  IX(1...LX)  FROM  X(1...LX),  B(1...LB)  AND  IXLO  AS FOLLOWS -
     SETS IX(I) TO EQUAL                          CONDITION ON X(I)
        IXLO                                           X(I)   LSTHN  B(1)
        IXLO                     B(1)     LSTHN=   X(I)   LSTHN  B(2)
        IXLO+1                   B(2)     LSTHN=   X(I)   LSTHN  B(3)
                   ETC.
        IXLO+LB-2                B(LB-1)  LSTHN=   X(I)   LSTHN  B(LB)
        IXLO+LB-2                B(LB)    LSTHN=   X(I)
     ASSUMING LX, LB GRTHN= 1 AND B(J) GRTHN B(J-1).  X AND B CAN
     BE ANY MODE AS LONG AS THEY ARE SAME MODE.  SETS  IANS = 0 IF ALL OK,
     = -1, -2 OR -3 IF ILLEGAL LX, LB OR WEIRD ERROR.

MRVRS  (NRA,NCA,LA,AA)                                   FORTRAN,  61 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - REVERS.
     REVERSES THE STORAGE ORDER OF THE VECTOR  AA(1...NRA*NCA*LA)  OF
     NRA  BY  NCA  MATRICES.  NRA,  NCA,  AND  LA  MUST BE GRTHN=  1 .

MSCON1 (NORDER,P,PHI,DEPEND,IANS)                        FORTRAN, 238 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  PHI = MEAN SQUARE CONTINGENCY AND DEPEND = DEPENDENCY MEASURE,
     GIVEN SECOND PROBABILITY MATRIX  P(1...NORDER,1...NORDER) WITH
     DIMENSION P(25,25) NORMALIZED SO SUM (OVER I AND J) OF P(I,J) = 1.0,
     AND CONSTRAINED SO  SUM (OVER  I  OR OVER  J) OF  P(I,J)  IS NOT
     = 0. FOR ANY  J  OR  I.  SETS IANS = 0 IF ALL OK, = -1 IF NORDER
     OUTSIDE RANGE  1...25,  = -2 IF ILLEGAL  P  MATRIX.

MULK   (C,X1,X2,...,XN)                           FAP, SECONDARY ENTRY OF ADDK
     SETS  X1=X1*C, X2=X2*C, ..., XN=XN*C.  EQUIV(ANY ARGUMENTS) OK, BUT
     INITIAL VALUE OF  C  IS ALWAYS THE MULTIPLIER.  STRAIGHT RETURN IF N=0.

MULK   (C,X1,X2,...,XN) - II                             FORTRAN,  76 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - SETUP,ARG,STORE,RETURN.
     SAME FUNCTION AS FAP VERSION OF MULK

MULKS  (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)           FAP, SECONDARY ENTRY OF ADDK
     SETS  Y1=X1*C1,  Y2=X2*C2,  ...,  YN=XN*CN.  EQUIV(ANY TWO ARGUMENTS)
     OK BUT MAY CHANGE INPUTS  CJ  OR  XJ.  PROCESSING IS LEFT TO RIGHT.
     STRAIGHT RETURN IF  N=0.

MULLER (COE,N1,ROOTR,ROOTI)                              FORTRAN,  757 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - SQRT.
     FINDS THE REAL AND COMPLEX ROOTS  ROOTR(1...N1),  ROOTI(1...N1)  FOR A
     REAL VALUED POLYNOMIAL  COE(1...N1+1).  N1  GRTHN=  1 .

MULPLY (X,LX,XMLPLR,XMLPLD)                                 FAP,  34 REGISTERS
          OTHER ENTRY - XMLPLY.  NO TRANSFER VECTOR.
     SETS  XMLPLD(1...LX) = X(1...LX)*XMLPLR.  EQUIV(X,XMLPLD) OK, AND
     EQUIV(XMLPLR, SOME X(I)) OK, BUT INITIAL VALUE OF  XMLPLR  IS ALWAYS
     THE MULTIPLIER.  STRAIGHT RETURN IF LX LSTHN 1.
```

```
MUVADD (IV,ILO,IHI,LADD,MUVSUM,NSUMS,IANS)                FAP, 129 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  MUVSUM(1...NSUMS)  AND SETS  NSUMS = IHI-ILO+2-LADD, WHERE
        MUVSUM(1) = IV(ILO)+IV(ILO+1)+...+IV(ILO+LADD-1)
        MUVSUM(2) = IV(ILO+1)+IV(ILO+2)+...+IV(ILO+LADD)
               ETC.
        MUVSUM(NSUMS)=IV(IHI-LADD+1)+...+IV(IHI-1)+IV(IHI)
      SUBJECT TO  0 LSTHN ILO LSTHN=IHI, AND LADD GRTHN 0 .  SETS
      IANS = 0 IF ALL OK, = 1 IF LADD EXCEEDS IHI-ILO+1 (OTHERWISE IT
      TREATS THIS CASE AS THOUGH LADD = IHI-ILO+1), = -1 IF ILO, IHI OR
      LADD ILLEGAL, = -2 IF OVERFLOW (ALL SUMS COMPUTED ANYWAY).

MVBLOK (NN,ISORCE,IDEST)                                  FAP,  19 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  DEST(1...NN) = SORCE(1...NN) GIVEN ISORCE = MACHINE ADDRESS
      OF SORCE(1), IDEST = MACHINE ADDRESS OF DEST(1), AND NN.  THE VECTORS
      SORCE AND DEST MAY OVERLAP ONLY IF ISORCE EXCEEDS IDEST.

MVINAV (REC,LREC,K,RECAV,IANS)                        FORTRAN, 221 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  RECAV(1...LREC) WHERE  RECAV(I) = (1/(2K+1))*SUM (FROM J = I-K
      TO I+K) OF REC(J), WHERE COMPUTATIONS MADE AS THOUGH REC(J) WERE ZERO
      OUTSIDE 1...LREC.  LREC MUST EXCEED 0, K IS GRTHN= 0, AND (2*K+1) MUST
      BE LSTHN LREC (UNLESS K=0).  SETS  IANS = 0 IF ALL OK, = -2 OR -3 FOR
      ILLEGAL LREC OR K.

MVNSUM (X,LX,LSUM,DVSR,SUMOVD,LSUMOD)                     FAP,   71 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  LSUMOD = LX-LSUM+1  AND  SUMOVD(I)  =  (1/DVSR) * (SUM  (FROM
      J=I TO I+LSUM-1)  OF  X(J) )  FOR  I=1,2,...,LSUMOD.  EQUIVALENCE
      (X,SUMOVD)  OK.  STRAIGHT RETURN WITH NO OUTPUT IF  LX  OR  LSUM
      LSTHN  1,  OR IF  LSUM  GRTHN  LX.

MVNTIN (X,LX,DEL,LINT,XMI,LXMI)                           FAP,   88 REGISTERS
          OTHER ENTRY - MVNTNA.  NO TRANSFER VECTOR.
      SETS  LXMI = LX-LINT+1,  AND SETS  XMI(I)  =  DEL * ( X(I)/2.0
      + (SUM  (FROM J=I+1 TO I+LINT-2)  OF  X(J) ) + X(I+LINT-1)/2.0 )  FOR
      I=1...LXMI,  EXCEPT STRAIGHT RETURN WITH NO OUTPUTS IF  LX  OR  LINT
      LSTHN  2,  OR IF  LINT  GRTHN  LX.  EQUIVALENCE  (X,XMI)  AND
      EQUIVALENCE  (LX,LXMI)  PERMITTED.

MVNTNA (X,LX,DEL,LINT,XAMI,LXAMI)             FAP, SECONDARY ENTRY OF MVNTIN
      SETS  LXAMI = LX-LINT+1,  AND SETS  XAMI(I)  =  DEL * ( XA(I)/2.0
      + (SUM  (FROM J=I+1 TO I+LINT-2)  OF  XA(J) ) + XA(I+LINT-1)/2.0 )
      FOR  I=1...LXAMI,  WHERE  XA(I)  =  ABSOLUTE VALUE OF  X(I),  EXCEPT
      STRAIGHT RETURN WITH NO OUTPUT IF  LX  OR  LINT  LSTHN  2,  OR IF  LINT
      GRTHN  LX.  EQUIVALENCE  (X,XAMI)  AND  EQUIVALENCE  (LX,LXAMI)
      PERMITTED.

MVSQAV (REC,LREC,K,RECAV,IANS)                        FORTRAN, 236 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  RECAV(1...LREC) WHERE  RECAV(I) = (1/(2K+1))*SUM (FROM J=I-K
      TO I+K) OF REC(J)*REC(J), WHERE COMPUTATIONS MADE AS THOUGH REC(J)
      WERE ZERO OUTSIDE 1...LREC.  LREC MUST EXCEED 0, K IS GRTHN = 0 AND
      (2*K+1) MUST BE LSTHN LREC (UNLESS K=0).  SETS  IANS = 0 IF ALL OK, = -2
      OR -3 FOR ILLEGAL LREC OR K.
```

```
MXRARE (DN,DD,LD,DNFRAC,DDFRAC,MNREWI,RAMX,            FORTRAN, 302 REGISTERS
        ILO,IHI,IANS)
          NO OTHER ENTRIES.  TRANSFER VECTOR - EXP(2.
     GIVEN  DN(1...LD),  DD(1...LD)  WHERE  DN  AND  DD  SATISFY  D(I+1)
     GRTHN = D(I)  AND  D(LD)  GRTHN D(1), MXRARE SETS  RAMX, ILO, IHI
     WHERE RAMX = (DN(IHI)-DN(ILO))/(DD(IHI)-DD(ILO)) AND ILO, IHI ARE
     CHOSEN TO MAXIMIZE RAMX, SUBJECT TO THREE CONSTRAINTS  1) (DN(IHI)-
     DN(ILO)/(DN(LD)-DN(1))  MUST BE GRTHN= DNFRAC,  2) (DD(IHI)-DD(ILO))/
     (DD(LD)-DD(1))  MUST BE GRTHN= DDFRAC,  AND  3) (IHI-ILO) MUST BE GRTHN=
     MNREWI.  DNFRAC AND DDFRAC MUST LIE IN CLOSED RANGE 0. TO 1.0,
     MNREWI IS GRTHN 0 AND LSTHN LD.  IN CASE OF ZERO DENOMINATORS, 0/0 IS
     TAKEN = 0  AND K/0 IS TAKEN = 10 EXP 35 (AND CHOSEN AS MAXIMUM).
     SETS  IANS = 0 IF ALL OK, = -1, -2,... OR -6 FOR ILLEGAL DN,DD,LD,
     DNFRAC, DDFRAC OR MNREWI, = 1 IF A 0/0 RATIO FOUND, = 2 IF A K/0 RATIO
     FOUND (SUPERCEDES IANS = 1 CASE).

NEXCOSF(DUMMY)                                  FAP, SECONDARY ENTRY OF SEQSAC
     HAS VALUE  =  COS(ARGLO+(NTIMES-1)*ARGDEL)  WHERE  NTIMES  =  NUMBER OF
     TIMES  NEXCOSF  HAS BEEN USED PRIOR TO PRESENT USE AND SUBSEQUENT TO THE
     LAST  CALL SEQSAC(ARGLO,ARGDEL)  STATEMENT.  DUMMY IS IGNORED.

NEXSINF(DUMMY)                                  FAP, SECONDARY ENTRY OF SEQSAC
     HAS VALUE  =  SIN(ARGLO+(NTIMES-1)*ARGDEL)  WHERE  NTIMES  =  NUMBER OF
     TIMES  NEXSINF  HAS BEEN USED PRIOR TO PRESENT USE AND SUBSEQUENT TO THE
     LAST  CALL SEQSAC(ARGLO,ARGDEL)  STATEMENT.  DUMMY  IS IGNORED.

NMZMG1 (LX,X,XMAX,SCALE)                                  FAP,  34 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  X(1...LX) = C*X(1...LX)  AND SETS SCALE = 1/C, WHERE C =
     XMAX/(MAGNITUDE OF THAT X(I) INPUT VALUE WHICH HAS LARGEST MAGNITUDE).
     IF XMAX=0. IT SETS SCALE=0. AND X(1...LX)=0.  LX SHOULD EXCEED 0 .

NOINT1 (X,PROB)                                           FAP, 369 REGISTERS
          OTHER ENTRY - NOINT2.  TRANSFER VECTOR - LINTR1.
     SETS  PROB = (1/SQRT(2PI))*INTEGRAL (FROM MINUS INFINITY TO X)
     OF (EXP(-(X**2)/2)DX).

NOINT2 (XMEAN,XSD,NDIV,XDIV,IANS)               FAP, SECONDARY ENTRY OF NOINT1
     SUPPOSE  P(X)  IS UNIT AREA NORMAL DISTRIBUTION WITH MEAN XMEAN AND
     STANDARD DEVIATION XSD.  THE NOINT2 SETS XDIV(1...NDIV-1)  SO THAT
     THE INTEGRAL OF  P(X)  FROM  XDIV(I)  TO  XDIV(I+1)  IS CONSTANT
     (= 1/NDIV)  FOR I = 0,1,...,NDIV-1  WHERE  XDIV(0)  AND  XDIV(NDIV)  ARE
     IMPLIED TO BE - AND + INFINITY RESPECTIVELY.  SETS  IANS = 0 IF ALL
     OK, = 1 OR 2 FOR ILLEGAL XSD (LSTHN= 0.) OR NDIV (LSTHN 2).

NRMVEC (ZIFRMS,SCALE,X,LX,XMEAN,XMAX,XNRM)            FORTRAN,  111 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - MAXAB,SQRT.
     SETS  XNRM(I)  =  X(I)*SCALE/XMAX + XMEAN  FOR I=1...LX  WHERE  XMAX  =
     1/LX  TIMES THE SQUARE ROOT OF  SUM  (FROM I=1 TO LX)  OF  X(I)*X(I)
     IF  ZIFRMS = 0.,  OR  =  ABSOLUTE VALUE OF MAXIMUM OF  X(I)  I=1...LX
     IF  ZIFRMS NOT= 0.  LX  MUST BE  GRTHN=  1 .  EQUIVALENCE  (X,XNRM)  OK.
```

```
NTHA   F(N,A1,A2,...,AN,...)                                    FAP,    11 REGISTERS
             OTHER ENTRY - XNTHA.  NO TRANSFER VECTOR.
       HAS VALUE  =  AN  WHERE  AN  =  N-TH ARGUMENT FOLLOWING  N,  EXCEPT
       VALUE  =  N  IF  N  LSTHN=  0  AND  VALUE IS UNPREDICTABLE IF  N+1
       EXCEEDS ARGUMENT COUNT.

NURINC (YOFX,LY,XLO,XHI,LYNU,XLONU,                             FAP,   121 REGISTERS
         XHINU,IERR1,YOFXNU,IANS)
             NO OTHER ENTRIES.  NO TRANSFER VECTOR.
       SETS  YOFXNU(1...LYNU)  =  VALUES LINEARLY INTERPOLATED FROM
       YOFX(1...LX)  WHERE  YOFX(I)  CORRESPONDS TO  Y(XLO+(I-1)*DELX)  WITH
       DELX = (XHI-XLO)/(LY-1)  AND WHERE  YOFXNU(I)  CORRESPONDS TO
       Y(XLONU+(I-1)*DELXNU)  WHERE  DELXNU = (XHINU-XLONU)/(LYNU-1).  REQUIRE
       LY  GRTHN=  2,  XHI  GRTHN  XLO,  LYNU  GRTHN=  1,  XLO  LSTHN=  XLONU
       LSTHN=  XHI,  AND (ONLY IF  LYNU  GRTHN=  2)  XLONU  LSTHN  XHINU
       LSTHN=  XHI.  SETS  IANS=0  IF ALL OK,  = IERR1+K-1  K = 2,4,5,6, OR 7
       IF  LY,XHI,LYNU,XLONU, OR XHINU  ILLEGAL.

NXALRM (JOB,MLIV,ILO,IHI,LEVEL,LTENSE,                     FORTRAN, 243 REGISTERS
         IBGIN,IEND,ISUM,IANS)
             NO OTHER ENTRIES.  TRANSFER VECTOR - FASCN1.
       SCANS MLIV(ILO...IHI)  LOOKING FOR A BLOCK OF AT LEAST  LTENSE
       CONTIGUOUS ELEMENTS ALL OF WHICH ARE GRTHN= LEVEL (MLIV AND LEVEL
       ARE FXD.PT. WITH ARB. BINARY POINT POSITION).  IF JOB = 0 AND IF
       BLOCK IS FOUND NXALRM SETS IBGIN AND IEND SO THAT  MLIV(IBGIN...IEND)
       DEFINES THE BLOCK, AND SETS ISUM = SUM OF BLOCK ELEMENTS (OVERFLOW
       IGNORED).  IF JOB = 1 THE SETTING OF IEND AND ISUM IS SUPPRESSED.
       SETS IANS = 0 IF NO BLOCK FOUND (IN THIS CASE IBGIN, IEND, ISUM ARE
       SET = 0), = 1 IF FOUND AND SPECIFIED, = 2 IF POSSIBLE BLOCK WAS
       STARTING BUT RAN OFF END OF MLIV BEFORE TRUE IEND LOCATED (IN THIS
       CASE IEND SET = IHI, ISUM = SUM FROM IBGIN TO IHI), = -1 IF ILLEGAL
       ILO (LSTHN 1), IHI (LSTHN ILO) OR LTENSE (LSTHN 1), = -99 IF
       UNEXPECTED ERROR FROM FASCN1.

ONLINE (ISENSE)                                                 FAP,   134 REGISTERS
             OTHER ENTRIES - (STH),(STHD),(STHM).  TRANSFER VECTOR - (FIL),
             (IOH),(RCH),(SPH),(TES),(WER),(WRS),(WTC).
       CAUSES ALL MATERIAL THAT IS WRITTEN ON AN OUTPUT TAPE TO BE PRINTED
       ONLINE  (1)  IF  ONLINE  HAS BEEN CALLED,  AND  (2)  IF SENSE SWITCH
       ISENSE  IS DOWN.

OUDATA (ITAPE,IRECNO,NOPTS,DATA,MODCOD,                    FORTRAN, 495 REGISTERS
        6HNAUXL1,LAUXL1,AUXL1,...,6HNAUXLN,LAUXLN,AUXLN)
             NO OTHER ENTRIES.  TRANSFER VECTOR - VARARG,LOC,MVBLOK,FAPSUM,PAKN,
             (STB),(WLR),(EFT).
       WRITES ONE BINARY FILE ON LOGICAL TAPE ITAPE FOR FUTURE RETRIEVAL BY
       SUBROUTINE INDATA.  FILE WILL CONTAIN DATA(1...NOPTS)  PACKED
       MODCOD WORDS/REGISTER (MODCOD LIES IN CLOSED RANGE 1...18) , THE
       RECORD NO. IRECNO (ANY MODE), AND SOME CONTROL INFORMATION.  THE
       ARGUMENT TRIPLETS BEYOND MODCOD WHICH SPECIFY AUXILIARY INFO ARE
       OPTIONAL.  IF PRESENT THE FILE WILL ALSO CONTAIN THE N VECTORS
       AUXL1(1...LAUXL1)...AUXLN(1...LAUXLN)  AND THEIR NAMES (INDICATED HERE
       AS HOLLERITH) 6HNAUXL1,...,6HNAUXLN.  NOPTS MUST EXCEED 0 .
```

(CONTINUED NEXT PAGE)

DATA(1...NOPTS) MUST BE FLTG.PT. ONLY IF MODCOD EXCEEDS 1 (IN WHICH CASE DATA(1...NOPTS) IS DESTROYED ON OUTPUT). THE AUXIL INFO IS ANY MODE (IT IS NEVER PACKED) SUBJECT TO CONSTRAINTS 1) N LSTHN 31, AND 2) SUM OF LAUXL VALUES LSTHN= 198-2*N .

PACDAT (ITAPE,NWORDS,IFSTWD,IFOLD,DATA,LDATA,IANS) FAP, 152 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - (IOS),(TCO),(RDS),(RCH),(ETT).

READS EVERY IFOLD-TH WORD, BEGINNING WITH THE IFSTWD WORD, FROM A BINARY RECORD ON LOGICAL TAPE ITAPE UNTIL A TOTAL OF NWORDS WORDS ARE READ. THE WORDS ARE STORED IN FORTRAN ORDER IN DATA(1...LDATA) WHERE LDATA IS THE ACTUAL NUMBER OF WORDS READ. SETS IANS = 0 IF ALL OK, = 1 IF AN END-OF-FILE MARK IS ENCOUNTERED, = 2 IF A REDUNDANCY IS ENCOUNTERED, = -1 IF ITAPE LSTHN 1, = -2 IF NWORDS LSTHN 1, = -3 IF IFSTWD LSTHN 1, = -4 IF IFOLD LSTHN 1 . IF ONE RECORD IS SHORTER THAN IFSTWD+(NWORDS-1)*IFOLD, MORE RECORDS WILL BE READ UNTIL THE DESIRED NUMBER OF WORDS IS FOUND. HOWEVER PHASING ERRORS MAY OCCUR AT THE RECORD GAPS. THE TAPE IS LEFT POSITIONED AFTER THE LAST RECORD READ, AFTER THE END-OF-FILE MARK IF IANS = 1, OR AFTER THE RECORD CONTAINING A REDUNDANCY IF IANS = 2 .

PAKN (N,LD,D,SCALE) FAP, 78 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - FXDATA.

SETS D(1...(LD+N-1)/N) AS SCALED, ROUNDED, FIXED, AND PACKED FORM OF FLTG.PT. INPUT D(1...LD). PACKING IS N WORDS PER REGISTER (RIGHT TO LEFT) WHERE 1 LSTHN= N LSTHN= 18 . SETS SCALE = VALUE BY WHICH DATA(I) MULTIPLIED BEFORE FIXING. IF LD=1, DATA(1...LD) AND SCALE UNDISTURBED. PAKN AND UNPAKN ARE (APPROXIMATE) INVERSES.

PLANSP (JOB,NRA,NCA,AA,MRS,JMAXR,MCS, JMAXC,SPT,SPACE1,SPACE2,IANS) FORTRAN, 1169 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - SETKS,LIMITS,IXCARG,CHOOSE, XOOZE,MOVREV,STZ,ROAR2,XADDKS,KOLAPS,COSTBL,SINTBL,XADDK,COSIS1, MATRA.

SETS SPT(1...(2*JMAXR+1)*(JMAXC+1)) = SP(-JMAXR...JMAXR,0...JMAXC) WHERE SP(I,J) = SUM (FROM X=-XL TO XL) OF SUM (FROM Y=-YL TO YL) OF (A(X,Y) * COS(I*X*PI/MRS + J*Y*PI/MCS)) IF JOB = 1, OR WHERE SP(I,J) = SUM (FROM X=-XL TO XL) OF SUM (FROM Y=-YL TO YL) OF (A(X,Y) * SIN(I*X*PI/MRS + J*Y*PI/MCS)) IF JOB = -1 GIVEN AA(1...NRA*NCA) = A(-XL...XL,YM...YL) WHERE YM = 0 IF NCA ODD, = .5 IF NCA EVEN. SETS IANS(1) = 0 IF ALL OK, = 1 IF JOB GRTHN 1 OR LSTHN -1, = 2 IF NRA LSTHN 1, = 3 IF NCA LSTHN 1, = 4 IF MRS LSTHN 1, = 5 IF MCS LSTHN 1, = 6 IF JMAXR LSTHN 1 OR GRTHN MRS, = 7 IF JMAXC LSTHN 1 OR GRTHN MCS. SETS IANS(2) = LSP1 AND IANS(3) = LSP2 WHERE SPACE1(1...LSP1) AND SPACE2(1...LSP2) ARE NEEDED FOR SCRATCH. IF JMAXR = MRS = JMAXC = MCS = NRA/2 = NCA/2 = M THEN LSP1 LSTHN= 8*M*M+9*M+5 AND LSP2 LSTHN= 2*M*M+3*M+1 (SEE ABSTRACT FOR DETAILED DEFINITION). EQUIVALENCE (AA,SPACE1), (SPT,SPACE2) ALLOWED. IF JOB = 0 NO COMPUTATIONS ARE MADE AND ONLY IANS(1...3) IS RETURNED AS AN OUTPUT.

PLOTVS (ITAPE,ISENSE,LOCYV,YSMBV,LYV,IXSTRV,NY, ARGLO,ARGDEL,ZFAFXD,FMTARG,NCOLS,YBOT, YTOP,HLINV,HLSMBV,NHL) FORTRAN, 494 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - RND,SETKS,SETKV,SETVEC,(FIL), (SPH),(STH),SWITCH.

CREATES, ON LOGICAL ITAPE (SUPPRESSED IF ITAPE LSTHN= 0) WITH ON-LINE MONITORING OPTION (WHILE SENSE SWITCH ISENSE IS DEPRESSED PROVIDED ISENSE=1...6) OR DEFINITE ON-LINE OUTPUT (FOR ISENSE GRTHN= 7), A PLOT (THE PLOTTING FIELD OCCUPYING NCOLS COLUMNS BEGINNING IMMEDIATELY AFTER THE LAST COLUMN USED ACCORDING TO FMTARG WHICH IS A 1A6 FORMAT WITHOUT PARENTHESES FOR PRINTING ROW LABELS ARGLO, ARGLO+ARGDEL, ARGLO+2*ARGDEL, ... BEGINNING IN COLUMN 2 WHERE ZFAFXD = 0.0 IMPLIES ARGLO, ARGDEL ARE FIXED, NOT= 0.0 IMPLIES FLOATING) OF THE NY VECTORS Y1(1...LYV(1)), Y2(1...LYV(2)), ..., YNY(1...LYV(NY)) WHOSE MACHINE ADDRESSES ARE GIVEN IN LOCYV(1...NY), WHERE EACH ELEMENT OF YSMBV(1...NY) GIVES A 1A1 HOLLERITH CHARACTER TO USE FOR THE CORRESPONDING VECTOR, WHERE IXSTRV(1...NY) GIVES THE OUTPUT ROW INDEX (GRTHN= 1) AT WHICH THE PLOTTING OF THE CORRESPONDING VECTOR IS TO START (LAST ROW INDEX USED IS MAX OVER I OF (LYV(I)+IXSTRV(I)-1)), WHERE YBOT AND YTOP ARE VALUES OF Y TO BE ASSOCIATED WITH FIRST AND LAST COLUMNS OF PLOTTING FIELD RESPECTIVELY (YTOP LSTHN YBOT IS OK), VALUES OF Y BEING IGNORED IF THEY FALL OUTSIDE THESE LIMITS, AND WHERE AN * IS USED EACH TIME TWO OR MORE VECTORS INTERSECT. IF NHL = 0 JOB IS DONE. OTHERWISE HLINV(1...NHL) GIVES Y VALUES AT WHICH HORIZONTAL LINES (WHEN VIEWED WITH PAGE COLUMNS HORIZONTAL) ARE TO BE DRAWN WITH CORRESPONDING 1A6 CHARACTERS IN HLSMBV(1...NHL), VECTOR CHARACTERS TAKING PRECEDENCE OVER HORIZONTAL LINE CHARACTERS IN CASES OF INTERSECTION. REQUIREMENTS NY GRTHN= 1, LYV(I) GRTHN= 1, (NCOLS+1 + NO. COLUMNS IMPLIED IN FMTARG) GRTHN= 132, YTOP NOT= YBOT, IXSTRV(I) GRTHN= 1, AND LEGITIMACY OF LOCYV, YSMBV, HLSMBV VECTORS ARE NOT CHECKED.

PLTVS1 (ITAPE,ISENSE,ARGLO,ARGDEL,ZFAFXD,NCOLS, ZFZERS,RMSSEP,S,LX,ZFLIST,VMATRX,IDIMEN,NX) FORTRAN, 817 REGISTERS

NO OTHER ENTRIES. TRANSFER VECTOR - BOOST,DPRESS,MAXSN,MINSN, MULPLY,PLOTVS,RMSDEV,SETKS,SETKVS,SETVEC,VARARG,XLOC,XSAME,XSTLIN, (FIL),(STH).

CREATES, ON LOGICAL ITAPE (SUPPRESSED IF ITAPE LSTHN= 0) WITH ON-LINE MONITORING OPTION (WHILE SENSE SWITCH ISENSE IS DEPRESSED PROVIDED ISENSE=1...6) OR DEFINITE ON-LINE OUTPUT (FOR ISENSE GRTHN= 7), A PLOT (THE PLOTTING FIELD OCCUPIES NCOLS COLUMNS BEGINNING IN COLUMN 6 IF ZFAFXD = 0.0 OR COLUMN 14 IF ZFAFXD NOT= 0.0, COLUMNS 2,3,...5 OR 13 CONTAINING ROW LABELS ARGLO, ARGLO+ARGDEL, ARGLO+2*ARGDEL, ... IN FORMAT(I4) OR (E12.5) ACCORDING AS ZFAFXD = 0.0 OR NOT= 0.0) OF THE NX VECTORS X1(1...LX), X2(1...LX), ..., XNX(1...LX) WHICH, IF ZFLIST NOT= 0.0, ARE SUPPLIED IN THE MATRIX VMATRX(1...LX,1,,,NX) WHERE USER HAS DIMENSION VMATRX(IDIMEN,IGNORD), BUT WHICH, IF ZFLIST = 0.0, ARE SUPPLIED BY A LIST OF ARGUMENTS X1,X2,...,XNX WHICH SUPPLANT THE ARGUMENTS VMATRX,IDIMEN,NX IN THE CALLING SEQUENCE, WHERE THE FIRST NX CHARACTERS OF THE LIST 1,2,...,9,A,B,...,Z ARE CHOSEN FOR THE CORRESPONDING VECTORS, AND WHERE BEFORE PLOTTING, THE VECTORS ARE ALL SCALED TO UNIT RMS VALUE, (J-1)*RMSSEP IS SUBTRACTED FROM THE J-TH VECTOR, AND THE PLOTTING FIELD IS ADJUSTED SO LARGEST AND SMALLEST X WILL COVER NCOLS COLUMNS, EXCEPT THAT THE PLOTTING OF EACH VECTOR WHICH IS IDENTICALLY

(CONTINUED NEXT PAGE)

```
      ZERO IS SUPPRESSED (TO ALLOW USER-CONTROLLED SPACING BY UNITS OF
      RMSSEP).  THE TRUE MAXIMA AND MINIMA OF EACH VECTOR, AS WELL AS ITS
      CHARACTER, ARE PRINTED IN A TABLE PRIOR TO THE GRAPHICAL PLOT (WHICH IS
      PRECEDED BY A PAGE RESTORE).  AFTER PLOTTING, THE VECTORS ARE RESCALED
      TO THEIR ORIGINAL VALUES.  IF  ZFZERS NOT= 0.0  THE JOB IS DONE.  FOR
      ZFZERS = 0.0  THE PLOT ALSO CONTAINS HORIZONTAL LINES (VIEWED WITH PAGE
      COLUMNS HORIZONTAL) COMPOSED OF PERIOD CHARACTERS INDICATING THE ZERO
      LEVELS FOR EACH OF THE VECTORS.  S(1...300)  REQUIRED FOR SCRATCH.
      NO CHECKING IS MADE ON THE VARIOUS REQUIREMENTS,  RMSSEP  GRTHN=  0.0,
      LX  GRTHN=  1,  IDIMEN  GRTHN=  LX,  NX  GRTHN=  1  BUT  LSTHN=  35,
      AND  NCOLS  NOT TOO LARGE FOR PRINTER,  AND  ITAPE  NOT EXCESSIVE.

PLURAL (SUBROU,A1,A2,...,AN,B1,B2,...,BN,        FAP, SECONDARY ENTRY OF SEVRAL
        ......,Z1,Z2,...,ZN)
      FUNCTION IS EQUIVALENT TO
                            CALL SUBRU(A2,A2,...,AN)
                            CALL SUBRU(B1,B2,...,BN)
                                 ETC
                            CALL SUBRU(Z1,Z2,...,ZN)
      WHERE SUBRU HAS BEEN LOCATED UNDER THE PROXY NAME SUBROU BY A PRIOR
      CALL LOCATE STATEMENT.  SUBRU MUST NOT USE DATA BEYOND THE END OF ITS
      CALLING SEQUENCE.

PLURNS (A1,A2,...,AN,B1,B2,...,BN,......,Z1,Z2,...,ZN)      FAP,  73 REGISTERS
    OR
       (A1,A2,...,ANA,STOP,B1,B2,...,BNB,STOP,
        ......,Z1,Z2,...,ZNZ)
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      CALL PLURNS(FIRST ARGUMENT STRING ABOVE) IMMEDIATELY FOLLOWED BY
      CALL SUBRU(N), WHERE N = NORMAL, NON-ZERO ARGUMENT COUNT OF SUBRU, IS
      EQUIVALENT TO
          CALL SUBRU(A1,A2,...,AN)
          CALL SUBRU(B1,B2,...,BN)
              ETC
          CALL SUBRU(Z1,Z2,...,ZN).
      CALL PLURNS(SECOND ARGUMENT STRING ABOVE) WHERE STOP = OCT777777712345
      IMMEDIATELY FOLLOWED BY CALL SUBRU(0) OR CALL SUBRU IS EQUIVALENT TO
          CALL SUBRU(A1,A2,...,ANA)
          CALL SUBRU(B1,B2,...,BNB)
              ETC
          CALL SUBRU(Z1,Z2,...,ZNZ).
      ANY OF THE ARGUMENT COUNTS NA,NB,...,NZ MAY BE ZERO.  LIMITATION - NONE
      OF THE ARGUMENTS IN ONE ARGUMENT GROUP MAY BE EXPRESSIONS INVOLVING
      OUTPUTS OF PREVIOUS ARGUMENT GROUPS EXCEPT FOR PURE EQUIVALENCES.

PLYSYN (SCALES,RADII,DGREES,NROOTS,PLYCOS,              FORTRAN, 170 REGISTERS
        NCOFS,SPACE)
          NO OTHER ENTRIES.  TRANSFER VECTOR - COS,CONVLV.
      SETS  NCOFS = M+2N+1  AND  PLYCOS(1...NCOFS) WHERE PLYCOS ARE
      POLYNOMIAL COEFFICIENTS DETERMINED TO HAVE PRESPECIFIED ROOTS (M REAL,
      N  COMPLEX CONJUGATE PAIRS, N+M = NROOTS) AND WEIGHTING FACTORS.
      ROOTS GIVEN BY  RADII(1...NROOTS)  AND DGREES(1...NROOTS)  WITH WEIGHTS
      SCALES(1...NROOTS).  ROOT CONSIDERED REAL ONLY IF DGREES = 0.0 OR
      EXACT MULTIPLE OF 180., OTHERWISE COMPLEX CONJUGATE INFERRED.
      NROOTS MUST EXCEED ZERO AND SPACE(1...NCOFS) NEEDED FOR SCRATCH.
```

```
POKCT1 (IX,NHANDS,ICT,IANS)                            FORTRAN, 219 REGISTERS
        NO OTHER ENTRIES.  TRANSFER VECTOR - FRQCT1.
     TREATS  IX(1...5*NHANDS), WHERE 0 LSTHN = IX(I) LSTHN = 9, AS
     NHANDS POKER HANDS (NON OVERLAPPING GROUPS OF 5) AND SETS  ICT(1...8)
     = FREQUENCIES OF DIFFERENT TYPE HANDS,  WHERE ICT(1)= NO. BUSTS,
     ICT(2) = NO. PAIRS,  ICT(3) = NO. 2-PAIRS,  ICT(4) = NO. 3-OF-KINDS,
     ICT(5) = NO. FULL HOUSES, ICT(6) = NO. STRAIGHTS, ICT(7) = NO. 4-OF-
     KINDS, AND ICT(8) = NO. 5-OF-KINDS.  SETS  IANS = 0 IF ALL OK, = 1
     IF NHANDS LSTHN 1, = 3 IF ERROR RETURN FROM FRQCT1.  THE APRIORI
     PROBABILITIES ASSOCIATED WITH ICT(1...8) ARE  .2952, .5040, .1080,
     .0720, .0090, .0072, .0045, .0001 .

POLYDV (N,DVS,M,DVD,L,Q)                               FORTRAN, 130 REGISTERS
        NO OTHER ENTRIES.  TRANSFER VECTOR - MOVE,STZ.
     SETS  Q(1...L) = FIRST  L  COEFFICIENTS OF QUOTIENT OF POLYNOMIAL
     DVD(1)+DVD(2)*X+...+DVD(M)*X**(M-1) DIVIDED BY POLYNOMIAL
     DVS(1)+DVS(2)*X+...+DVS(N)*X**(N-1),  WHERE  M, N, L MUST BE GRTHN= 1,
     AND DVD(1) NOT = 0.

POLYEV (N,C,X,A)                                       FORTRAN,  54 REGISTERS
        NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  A = C(1)+C(2)*X+...+C(N)*X**(N-1),  WHERE N GRTHN= 1 .

POLYSN (SCALE,NOZ,ZRE,ZIM,ZIFCOM,                     FORTRAN,  256 REGISTERS
       ZIFCNJ,LPOLY,POLY,SPACE)
        NO OTHER ENTRIES.  TRANSFER VECTOR - SQRT,CONVLV,MOVE,COS.
     SETS  LPOLY  AND  POLY(1...LPOLY)  WHERE  POLY  ARE THE POLYNOMIAL
     COEFFICIENTS DETERMINED TO HAVE THE PRESPECIFIED ROOTS  ZRE(1...NOZ)
     AND  ZIM(1...NOZ)  (NOZ  GRTHN=  1).  IF  ZIFCOM=0.,  ZRE  AND  ZIM
     CONTAIN THE REAL AND IMAGINARY PARTS OF THE ROOTS.  IF  ZIFCOM NOT= 0.,
     ZRE  AND  ZIM  CONTAIN THE MAGNITUDE AND THE ARGUMENT IN DEGREES OF THE
     ROOTS.  IF  ZIFCNJ=0.  POLYSN  INSERTS A COMPLEX CONJUGATE FOR EACH OF
     THE  M  NON-REAL ROOTS AND  LPOLY = M+NOZ+1 .  IF  ZIFCNJ NOT= 0.,  THEN
     POLYSN  ASSUMES THAT THE COMPLEX CONJUGATE PAIRS OCCUR IN THE LIST AND
     LPOLY = NOZ+1 .  THE POLYNOMIAL IS MULTIPLIED BY  SCALE AFTER IT IS
     COMPUTED.  IF  SCALE=0.  THE POLYNOMIAL IS SET SO THAT  POLY(1) = 1.
     TEMPORARY  SPACE(1...2*NOZ)  IS NEEDED.

POWER  (X,LX,N,X2NTH)                                      FAP,  50 REGISTERS
        OTHER ENTRY - SMPRDV.  TRANSFER VECTOR - EXP(2.
     SETS X2NTH(I) = X(I)**N FOR I=1...LX, WHERE N IS ARBITRARY.
     EQUIV(X2NTH,X) OK.  STRAIGHT RETURN IF LX LSTHN 1.

PRBFIT (NOR,XMOM,NOUT,X,F,PHI,IANS)                    FORTRAN, 373 REGISTERS
        NO OTHER ENTRIES.  TRANSFER VECTOR - SQRT,EXP(2,EXP.
     LET  F(X)  BE A UNIT AREA, ZERO MEAN DISTRIBUTION FUNCTION, GENERATED BY
     AN EDGEWORTH SERIES, WHOSE HIGHER MOMENTS UP TO ORDER NOR (LSTHN = 6)
     ARE GIVEN BY  XMOM(2,3,...,NOR).  THEN PRBFIT SETS  F(1...NOUT) =
     VALUES OF F(X) FOR X = X(1...NOUT).  PHI(1...NOUT) USED FOR SCRATCH.
     SETS  IANS = 0 IF ALL OK, = 1 FOR ILLEGAL NOR (LSTHN 2 OR GRTHN 6).
```

PROB2 (IX,LX,N,IP,P,IXHI,IANS)   FORTRAN, 229 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
IX(1...LX) IS GIVEN INTEGERS WITH VALUES FROM 1 TO IXHI (LSTHN= 25).
THEN PROB2 SETS P(1...IXHI,1...IXHI) AND IP(1...IXHI,1...IXHI)
(DIMENSION P(25,25),IP(25,25)), WHERE P(M,L) = PROBABILITY THAT (IX(K)=
M AND IX(K+N)=L), P(M,L) BEING NORMALIZED SO SUM OVER M, L = 1.0, AND
WHERE IP(M,L) = NO. TIMES IT OCCURS THAT (IX(K)= M AND IX(K+N)=L), THE
COUNTS BEING MADE OVER ALL K, K+N PAIRS SUCH THAT BOTH K AND K+N LIE
IN CLOSED RANGE 1...LX. N MAY BE NEGATIVE BUT MAGNITUDE (N) LSTHN LX.
SETS IANS = 0 IF ALL OK, = -1, -2, -3 OR -6 IF ILLEGAL IX VALUE,
LX, N, OR IXHI, = 3 IF OK BUT N = 0 (P AND IP ARE DIAGONAL).
EQUIV(P,IP) OK (COUNT MATRIX IP DESTROYED).

PROCOR (X,LX,MAXX,PROG1,PROG2,ERR)   FAP, 770 REGISTERS
OTHER ENTRIES - FASCOR,FASEPC,FASCR1,FASEP1. NO TRANSFER VECTOR.
X(1...LX) ARE MACHINE LANGUAGE INTEGERS WITH MAGNITUDES LSTHN= MAXX
(1 TO 1000). THEN PROCOR WRITES AN OBJECT PROGRAM WHICH WILL COMPUTE
CORRELATIONS OF X(1...LX) WITH OTHER SERIES, (THE OBJECT PROGRAM TO
BE OPERATED BY OTHER ENTRIES OF PROCOR). PROG1 AND PROG2 DEFINE A
STORAGE AREA FOR THE OBJECT PROGRAM OF AT LEAST LX+10*(MAXX+1)+1
REGISTERS, PROG1 BEING LOWER ABSOLUTE MACHINE ADDRESS, I.E.
XLOCF(PROG2)-XLOCF(PROG1) MUST BE GRTHN= LX+10*(MAXX+1). SETS ERR = 0.0
IF OK, =1.,2.,3., OR 4. IF STORAGE AREA TOO SMALL, IF LX LSTHN 1, IF
SOME X(I) MAGNITUDE EXCEEDS MAXX, OR IF MAXX ILLEGAL.

PSQRT (N,C,M,A)   FORTRAN, 155 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - SQRT.
SETS A(1...M) = FIRST M COEFFICIENTS OF THE POWER SERIES
A(1)+A(2)*X+A(3)*X**2+... WHOSE SQUARE IS A GIVEN POLYNOMIAL C(1)+
C(2)*X+...+C(N)*X**(N-1). N AND M MUST EXCEED 0 AND C(1) MUST EXCEED
0.0 .

PWMLIV (JOB,ITAPE,MLIV,LMLIV,IANS)   FORTRAN, 300 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - MLI2A6,(STH),(FIL),(SPH).
OUTPUTS MLIV(1...LMLIV) AS MACHINE LANGUAGE INTEGERS, ONTO LOGICAL TAPE
ITAPE (VALUE 1...12) IF JOB GRTHN ZERO, THROUGH ONLINE PRINTER (IGNORING
ITAPE) IF JOB LSTHN ZERO, WHERE MAGNITUDE(JOB) = DESIRED NO. OF
WORDS/LINE (GRTHN= 1, LSTHN= 10). FIELD WIDTH OF EACH WORD IS 12.
SETS IANS = 0 IF ALL OK, = -1, -2, OR -4, FOR ILLEGAL JOB, ITAPE, OR
LMLIV (MUST EXCEED 0).

QACORR (X,LX,MXACC,MXLAG,SPACE,ACOR,IANS)   FORTRAN, 207 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FXDATA,PROCOR,FASCOR,FLDATA.
SETS ACOR(1...MXLAG+1) = AC(0...MXLAG) WHERE AC(L) = (1/LX)*SUM
(FROM I=1 TO I=LX-L) OF X(I)*X(I+L). COMPUTATIONS ARE APPROXIMATE.
X(1...LX) IS CONVERTED TO INTEGER SEQUENCE WITH MAXIMUM MAGNITUDE =
MXACC (1 TO 1000) DURING COMPUTATIONS, BUT REFLOATED AFTERWARDS (HENCE
LEFT MORE OR LESS MODIFIED). SPACE(1...LX+10*(MXACC+1)+1) IS
SCRATCH AREA. SETS IANS = 0 IF ALL OK, = -2 IF LX LSTHN 1 OR GRTHN
10000, = -3 IF MXACC ILLEGAL, = -4 IF MXLAG NEGATIVE, = -98 OR -99 IF
WEIRD ERROR RETURN FROM PROCOR OR FASCOR OCCURS.

```
QCNVLV (XX,LXX,YY,LYY,MXACC,LCC,                        FORTRAN, 569 REGISTERS
       SPACE,CC,IANS)
          NO OTHER ENTRIES.  TRANSFER VECTOR - XLOC,FXDATA,PROCOR,FASCOR,
          FASEPC,FLDATA.
     SUPPOSE  XX(1...LXX) CONTAINS  X(0...LX=LXX-1) AND  YY(1...LYY)
     CONTAINS  Y(0...LY=LYY-1).  THEN QCNVLV SETS  CC(1...LCC) =
     C(0...LC=LCC-1) WHERE  C(J) = SUM (FROM  I=0 TO LX) OF X(I)*Y(J-I),
     WHERE  Y(K)  IS TREATED = 0. FOR K OUTSIDE RANGE 0 TO LY. COMPUTATIONS
     ARE APPROXIMATE.  XX(1...LXX) AND  YY(1...LYY) ARE CONVERTED TO INTEGER
     SEQUENCES WITH MAXIMUM MAGNITUDE = MXACC (1 TO 1000) DURING
     COMPUTATIONS, BUT ARE REFLOATED AFTERWARDS (HENCE LEFT MORE OR LESS
     MODIFIED).  SPACE(1...LMIN+10*(MXACC+1)+1) IS SCRATCH AREA, WHERE
     LMIN = MINIMUM (LXX,LYY). SETS IANS = 0 IF ALL OK, = -2 IF LXX LSTHN 1,
     = -3 IF YY PARTIALLY OVERLAPS  XX (HOWEVER EQUIV(XX,YY) IS OK),
     = -4 IF LYY LSTHN 1 OR LMIN EXCEEDS 10000, = -5 IF MXACC ILLEGAL,
     = -6 IF LCC LSTHN 1 (LCC MAY EXCEED LXX+LYY), = -99 IF WEIRD ERROR
     RETURN OCCURS FROM PROCOR, FASCOR OR FASEPC.

QFURRY (X,LX,IXZER,M,JMIN,JMAX,SPACE,                   FORTRAN, 244 REGISTERS
       CSP,SSP,IANS)
          NO OTHER ENTRIES.  TRANSFER VECTOR - STZ,MOVE,COSTBL,SINTBL,XSPECT.
     SETS  CSP(1...JMAX-JMIN+1) = CS(JMIN...JMAX) AND  SETS  SSP(1...JMAX-
     JMIN+1) = SS(JMIN...JMAX) WHERE 0 LSTHN= JMIN LSTHN JMAX LSTHN= M, AND
     WHERE  CS(J) = SUM (FROM I=L TO N) OF (XT(I)*COS(I*J*PI/M)) AND SS(J) =
     SAME SUM WITH SIN REPLACING COS, AND WHERE  L=1-IXZER  N=LX-IXZER
     (IXZER IS ARBITRARY, MAY EXCEED LX) AND THE  XT  SERIES IS GIVEN
     BY  X(1...LX) = XT(L...N).  SPACE(1...LSPACE) NEEDED FOR SCRATCH
     WHERE  LSPACE=2*(M+K)+6  WITH  K = MAGNITUDE OF  L  OR OF  N  WHICHEVER
     GREATER.  SETS  IANS = 0  IF ALL OK,  = -1  IF  LX  LSTHN  1,  = -2  IF
     M  LSTHN  1,  = -3  IF  JMAX  OR  JMIN  ILLEGAL.

QIFURY (FTREAL,FTIMAJ,MFREQ,LX,IXZER,SPACE,X,IANS)      FORTRAN, 280 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - COSTBL,SINTBL,COSISP,XLOC.
     SETS  X(I) = (1/2*M) * ( SUM (FROM J = -M TO M) OF
     S(J)*COS(J*(I-IXZER)*(PI/M)) + A(J)*SIN(J*(I-IXZER)*(PI/M)) )
     FOR I=1...LX  GIVEN  FTREAL(1...MFREQ+1)  AND  FTIMAJ(1...MFREQ+1),
     WHERE  M=MFREQ,  PI=3.14159265,  S(0,1,...,M-1) = FTREAL(1...M),
     S(M)=FTREAL(M+1)/2,  S(-1,...,-M)=S(1...M),  A(0...M)=FTIMAJ(1...M+1),
     AND  A(-1,...,-M)=-A(1...M).  EQUIV(X, FTREAL OR FTIMAJ) OK.
     SPACE(1...4*(M+1))  NEEDED FOR SCRATCH.  SETS  IANS=0  IF ALL OK,  = -1
     OR -2  FOR ILLEGAL  MFREQ (LSTHN 1)  OR ILLEGAL  LX (LSTHN 1).  FUNCTICN
     IS INVERSE TO THAT OF QFURRY.

QINTR1 (X,XLO,DELX,TABLE,NTABLE,YOFX)                   FORTRAN,  229 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - RNDUP,QUFIT1.
     SETS  YOFX  =  VALUE QUADRATICALLY INTERPOLATED FROM THREE TABLE VALUES
     OF  TABLE(1...NTABLE)  CLOSEST IN CORRESPONDENCE WITH THE ARGUMENT  X
     WHERE  TABLE(1,2,...)  CORRESPOND TO  XLO,XLO+DELX,...  WITH  DELX
     GRTHN  0.0,  EXCEPT LINEAR INTERPOLATION IS USED FOR  NTABLE = 2 .
     HOWEVER, SETS  YOFX  =  0.0  IF  X  OUTSIDE LIMITS  XLO  TO
     XLO+(NDATA-1)*DELX.  STRAIGHT RETURN WITH NO OUTPUT IF  NTABLE  LSTHN=
     1  OR IF  DELX  LSTHN=  0.0  .
```

```
QUFIT1 (FOFX,XLO,DELX,COEFS)                                  FAP,    79 REGISTERS
        NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  COEFS(1...3)  SUCH THAT THE QUADRATIC  F(X)  =  COEFS(1)
     + COEFS(2)*X + COEFS(3)*X**2  SATISFIES  F(XLO)=FOFX(1), F(XLO+DELX)
     =FOFX(2), F(XLO+2*DELX)=FOFX(3).  HIGH SPEED IF  XLO=-1.0  AND
     DELX=1.0 .  STILL FASTER IF  DELX  IS SET  = 0.0  IN WHICH CASE
     COMPUTATIONS MADE AS THOUGH  XLO=-1.0  AND  DELX=1.0  REGARDLESS OF
     ACTUAL  XLO  VALUE.

QXCORR (X,Y,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)             FORTRAN, 283 REGISTERS
        NO OTHER ENTRIES.  TRANSFER VECTOR - XLOC,FXDATA,PROCOR,FASCOR,
        FLDATA.
     ASSUME  X(1...LXY) AND  Y(1...LXY) ARE NOT EQUIV. THEN QXCORR
     SETS  XCOR(1...2*MXLAG+1) = XC(-MXLAG...MXLAG) WHERE  XC(L) =
     (1/LXY)*SUM (FROM I = 1 TO LXY) OF X(I)*Y(I+L), WHERE Y(K) IS TREATED
     = 0. FOR K OUTSIDE RANGE 1 TO LXY.  IF EQUIV(X,Y) EXISTS THEN
     QXCORR SETS  XCOR(1...MXLAG+1) = XC(0...MXLAG), I.E. ONE SIDE OF
     AUTOCORRELATION.  COMPUTATIONS ARE APPROXIMATE.  X(1...LXY) AND
     Y(1...LXY) ARE CONVERTED TO INTEGER SEQUENCES WITH MAXIMUM MAGNITUDE
     = MXACC (1 TO 1000) DURING COMPUTATIONS, BUT ARE REFLOATED AFTERWARDS
     (HENCE LEFT MORE OR LESS MODIFIED).  SPACE(1...LXY+10*(MXACC+1)+1)
     USED FOR SCRATCH.  SETS IANS = 0 IF ALL OK, = -2 IF Y PARTIALLY
     OVERLAPS X (EQUIV(X,Y) IS OK), = -3 IF LXY LSTHN 1, = -4 IF MXACC
     ILLEGAL, = -5 IF MXLAG NEGATIVE (MXLAG MAY EXCEED LXY), = -98 OR -99
     IF WEIRD ERROR RETURN OCCURS FROM PROCOR OR FASCOR.

QXCOR1 (LX,X,LY,Y,MXACC,ILAG,NLAGS,                      FORTRAN, 502 REGISTERS
      CORR,ZIFSTO,LSPACE,SPACE,IANS)
        NO OTHER ENTRIES.  TRANSFER VECTOR - SETKS,IXCARG,LIMITS,STZ,
        REVERS,PROCOR,FASCR1,FASCP1.
     SETS  CORR(1...NLAGS)  =  C(ILAG...ILAG+NLAGS-1)  WHERE  C(L)  =  SUM
     (FROM  I=1  TO  LX+LY)  OF  (X(I+L)*Y(I)),  WHERE  X  AND  Y  ARE
     TAKEN TO BE ZERO OUTSIDE THE RANGE OF DEFINITION.  X  AND  Y  ARE MLI
     VECTORS WITH LARGEST ABSOLUTE VALUE  LSTHN=  MXACC.  1  LSTHN=  MXACC
     LSTHN=  1000 .  NLAGS  MUST BE  GRTHN=  1 .  ZIFSTO = 0.  IMPLIES STORE
     OUTPUT WITHOUT ADDING,  NOT= 0.  IMPLIES ADD CORRELATION INTO THE OUTPUT
     AREA.  SPACE(1...LSPACE)  IS COMPUTATION SPACE.  EQUIVALENCE  (X,Y)  OK.
     IANS = 0  IF NO TROUBLE,  = 1  IF  LX  LSTHN  1,  = 2  IF  LY  LSTHN  1,
     = 3  IF  MXACC  LSTHN  1  OR  GRTHN  1000,  = 4  IF  NLAGS  LSTHN  1,
     = 5  IF  LSPACE  LSTHN  MIN(LX,LY) + 1 + 10*(MXACC+1),  = 24  IF A VALUE
     OF  X  OR  Y  ILLEGAL,  = 33  IF OVERFLOW OCCURS.

RDATA  (ITAPE,ITPCPY,IANS,SPACE,                         FORTRAN,  645 REGISTERS
     X1NAME,X1, X2NAME,X2, ... )
        NO OTHER ENTRIES.  TRANSFER VECTOR - ARG,CMPRA,HVTOIV,INTHOL,
        IVTOHV,IXCARG,RETURN,SETUP,STORE,(FIL),(RTN),(STH),(TSH).
     READS DATA CARDS IN FLEXIBLE FORMAT FROM LOGICAL TAPE  ITAPE.  CARDS ARE
     COPIED VERBATIM ON OUTPUT TAPE  ITPCPY  UNLESS  ITPCPY = 0 .  TEMPORARY
     SPACE(1...110)  NEEDED.  X1NAME  GIVES HOLLERITH NAME FOR STORAGE
     LOCATION  X1,  ETC.  RDATA  SCANS A CARD FOR A HOLLERITH NAME WHICH IT
     MATCHES WITH THE  XNAMES.  WHEN  XNNAME  IS FOUND, IT STORES THAT DATA
     FOLLOWING  XNNAME  ON THE CARD IN THE  XN  VECTOR.  THE DATA MAY BE IN
     4  FORMS.  (1)  AN INDEX VALUE  IN,  ENCLOSED IN PARENTHESES,
     INDICATING THE POSITION  XN(IN)  THAT THE NEXT WORD IS TO BE STORED IN.
     IF NO INDEX IS GIVEN ONE IS ASSUMED.  (2)  FIXED OR FLOATING NUMBERS
```
(CONTINUED NEXT PAGE)

THAT ARE INTERPRETED IN G FORMAT. (3) 12 OCTAL DIGITS FOLLOWED BY AN 'O' THAT ARE INTERPRETED IN O12 FORMAT. OR (4) N HOLLERITH CHARACTERS PRECEDED BY 'NH'. ANY NUMBER OF FIELDS MAY BE PLACED ON A CARD. RDATA CONTINUES READING CARDS UNTIL IT ENCOUNTERS THE WORD 'RETURN'. IANS=0 IF ALL OK, =-1 IF CALLED WITH THE WRONG NUMBER OF ARGUMENTS, = A POSITIVE COUNT OF UNINTERPRETABLE FIELDS IF THESE ARE ENCOUNTERED.

REFIT (X,LX,TYPE,SYM,ANT) FAP, SECONDARY ENTRY TO SPLIT
GIVEN SYM(1...LS) AND ANT(1...LA) WHERE LS+LA=LX AND, IF LX ODD LS=(LX+1)/2 LA=(LX-1)/2, OR IF LX EVEN LS=LA=LX/2, REFIT SETS X(1...LX), WHERE IF LX ODD X(I)=(SYM(LS+1-I)-ANT(LS-I))/2 FOR I=1...LS-1, X(LS)=SYM(1), X(I)=(SYM(I-LA)+ANT(I-LA-1))/2 FOR I=LS+1...LX, AND WHERE IF LX EVEN X(I)=(SYM(LS+1-I)-ANT(LS+1-I))/2 FOR I=1...LS AND X(I)=(SYM(I-LS)+ANT(I-LS))/2 FOR I=LS+1...LX. TYPE = 0.0 SIGNIFIES SYM, ANT AS FXD.PT. AND NOT = 0. SIGNIFIES FLTG.PT. (X WILL BE SAME MODE). LX SHOULD EXCEED 0 . EQUIV(SYM,X) OK ONLY IF ALSO HAVE EQUIV(ANT,X(LS+1)).

REFLEC (X,LX,XMIROR,XIMAGE) FAP, 28 REGISTERS
OTHER ENTRY - XRFLEC. NO TRANSFER VECTOR.
SETS XIMAGE(1...LX)=XMIROR-X(1...LX). EQUIV(XIMAGE,X) AND (XMIROR, ANY X(I)) OK, BUT INPUT XMIROR VALUE ALWAYS USED IN SUBTRACTION. STRAIGHT RETURN IF LX LSTHN 1.

REIM (AMP,PHZ,LR,RE,XIM) FAP, SECONDARY ENTRY TO AMPHZ
SETS RE(1...LR) AND XIM(1...LR) WHERE RE(J)=AMP(J)*COS(PHZ(J)) AND XIM(J)=AMP(J)*SIN(PHZ(J)). LR MUST EXCEED 0 . PHZ IS IN RADIANS.

REMAV (X,LX,XAVG,XNULD) FAP, 36 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XAVG = (1/LX)*(SUM(FROM I=1 TO LX) OF X(I)), AND XNULD(I) = X(I)-XAVG FOR I=1...LX. EQUIV(X,XNULD) OK. STRAIGHT RETURN IF LX LSTHN 1.

REREAD FAP, 114 REGISTERS
OTHER ENTRIES - EOFSET,ENDFIL,(TSH),(TSHM). TRANSFER VECTOR - (IOH),(RDS),(RDC),(RCH),(TCO),(TEF),EXIT,(RER).
CAUSES THE NEXT 'READ INPUT TAPE' STATEMENT TO REINTERPRET THE LAST CARD READ. SUCH STATEMENTS SHOULD INTERPRET ONLY ONE CARD.

RETURN (LOCALL,XR1,XR2) FAP, SECONDARY ENTRY TO LOCATE
RETURN SENDS CONTROL TO THE FORTRAN STATEMENT JUST FOLLOWING THE FORTRAN CALL STATEMENT WHOSE MACHINE ADDRESS IS LOCALL, AFTER RESTORING INDEX REGISTERS 1 AND 2 FROM XR1 AND XR2 . LOCALL, XR1 AND XR2 SHOULD HAVE BEEN SET UP FROM A PRIOR CALL SETUP STATEMENT.

REVER (X,LX,XREVD) FAP, 30 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XREVD(1...LX)=X(LX...1) EQUIV(XREVD,X) OK. STRAIGHT RETURN IF LX LSTHN 1 . X(I) IS ANY MODE.

REVERS (LX,X) FAP, 29 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS X(1...LX) = X(LX...1) WHERE X IS ANY MODE AND LX MUST EXCEED 0 .

RLSPR (L,A,R,ALPHA) FORTRAN, 142 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FDOTR.
SOLVES THE EQUATION SUM (FROM I=1 TO L) OF A(L-I+1)*R(I-J+1) = 0. FOR J=1...L GIVEN THE SOLUTION SUM (FROM I=1 TO L1) OF A(L-I)*R(I-J+1) = 0. FOR J=1...L1 . L1 = L-1 IS THE VALUE OF L ON INPUT. L IS THEN BUMPED UP BY 1 ON RETURN. R(1...L) IS ONE SIDE OF AN AUTOCORRELATION VECTOR (R(1) IS CENTER TERM). ALPHA = SUM (FROM I=1 TO L) OF A(I)*R(I). L MUST BE GRTHN= 0 ON INPUT.

RLSPR2 (NRA,NCAT,NCAN,AA,NRR,NCR,RR,CC,IANS) FORTRAN, 700 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - IXCARG,STZ,MOVREV,DOTP,MATML3, DOTJ,SIMEQ.
SETS AA(1...NRA*NCAT*NRA) = A(1...NRA,1...NCAT,1...NRA) WHERE SUM (FROM I=1 TO NRA) OF SUM (FROM J=0 TO NCAN+1) OF A(I,J,K)*R(I-M,J+N-1)) = 0. FOR K=1...NRA, M=1...NRA, N=1...NCAN+1 . SETS CC(1...NRA*NRA) = C(1...NRA,1...NRA) INVERSE, WHERE C(L,K) = SUM (FROM I=1 TO NRA) OF SUM (FROM J=0 TO NCAN) OF (A(I,J,K)*R(I-L,L)), GIVEN THE A AND C ARRAYS FROM THE LAST CALL OF RLSPR2. IT IS SELF-INITIATING IF NCAN=0, THEN EACH CALL BUMPS NCAN UP ONE. RR(1...NRR*NCR) = R(-NRR/2...NRR/2,0...NCR-1) IS AN AUTOCORRELATION ARRAY. CC(1...2*NRA*NRA+NRA) IS COMPUTATION SPACE. IANS=0 NORMALLY, =1 IF NCAN GRTHN NCAT, =2 IF NCAN LSTHN 0, =3 IF OVERFLOW OCCURS WHILE INVERTING A MATRIX.

RLSSR (L,A,R,G,F,ALPHA) FORTRAN, 82 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FDOTR.
SOLVES THE EQUATION SUM (FROM I=1 TO L) OF F(L-I+1)*R(I-J+1) = G(L-J+1) FOR J=1...L GIVEN THE SOLUTION OF SUM (FROM I=1 TO L-1) OF F(L-I)*R(I-J+1) = G(L-J) FOR J=1...L-1, AND A(1...L) AND ALPHA AS GIVEN BY RLSPR.

RMSDAV (X,LX,XAVG,RMSXMA) FAP, SECONDARY ENTRY OF RMSDEV
SETS XAVG = (SUM (FROM I=1 TO LS) OF X(I))/LX AND RMSXMA = SQUARE ROOT((SUM (FROM I=1 TO LX) OF (X(I)-XAVG)SQUARED)/LX), BUT STRAIGHT RETURN WITH NO OUTPUT IF LX LSTHN 1 .

RMSDEV (X,LX,XBASE,RMSXMB) FAP, 50 REGISTERS
OTHER ENTRY - RMSDAV. TRANSFER VECTOR - SQRT.
SETS RMSXMB = SQUARE ROOT((SUM (FROM I=1 TO LX) OF (X(I)-XBASE)SQUARED)/LX), EXCEPT STRAIGHT RETURN WITH NO OUTPUT IF LX LXTHN 1 .

RND F(Y) FAP, 15 REGISTERS
OTHER ENTRIES - RNDUP,RNDDN. NO TRANSFER VECTOR.
FUNCTION WHICH ROUNDS Y TO NEAREST FLOATING POINT INTEGER. ROUNDING IS UP IF FRACTIONAL PART GRTHN= .500000000, DOWN OTHERWISE.

```
RNDDN F(Y)                                      FAP, SECONDARY ENTRY OF RND
      FUNCTION WHICH ROUNDS  Y  DOWN TO GREATEST FLOATING POINT INTEGER
      WHICH IS LSTHN= Y .

RNDUP F(Y)                                      FAP, SECONDARY ENTRY OF RND
      FUNCTION WHICH ROUNDS  Y  UP TO SMALLEST FLOATING POINT INTEGER
      WHICH IS GRTHN= Y .

RNDV   (X,LX,XR)                                         FAP,  34 REGISTERS
           OTHER ENTRIES - RNDVDN,RNDVUP.  TRANSFER VECTOR - RND,RNDUP,RNDDN.
      SETS  XR(1...LX)=X(1...LX)  ROUNDED TO NEAREST FLTG. PT. INTEGER (ROUNDS
      UP FOR FRACTION = 0.5).  EQUIV(X,RX) OK.  STRAIGHT RETURN IF LX LSTHN 1.

RNDVDN (X,LX,XR)                                FAP, SECONDARY ENTRY OF RNDV
      SETS  XR(1...LX)=X(1...LX)  ROUNDED DOWN TO NEAREST FLTG. PT. INTEGER
      (I.E.,  1.7 GOES TO 1.0,  -1.7 TO -1.0).  EQUIV(XR,X) OK.  STRAIGHT
      RETURN IF  LX  LSTHN 1.

RNDVUP (X,LX,XR)                                FAP, SECONDARY ENTRY OF RNDV
      SETS  XR(1...LX),X(1...LX)  ROUNDED UP TO NEAREST FLTG. PT. INTEGER
      (I.E.,  1.0 GOES TO 1.0,  1.1 TO 2.0,  -1.1 TO -2.0).  EQUIV(XR,X) OK.
      STRAIGHT RETURN IF  LX  LSTHN 1.

ROAR2  (JOB,XA,N,M,XRA)                            FORTRAN,  174 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - MATRA,MOVREV,REVERS.
      SETS  XRA(1...(M+M+1)*(N+1))  =  X(-M...M,0...N)  GIVEN
      XA(1...(N+N+1)*(M+1))  =  X(-N...N,0...M)  UNDER THE ASSUMPTION THAT
      X  IS CENTRO-SYMMETRIC IF  JOB=1,  OR CENTRO-ANTISYMMETRIC IF  JOB=-1 .
      EQUIVALENCE  (XA,XRA)  ALLOWED.

ROTAT1 (X,NX,NUP,ROTX)                                  FAP,   46 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  ROTX(1...NX) WHERE  ROTX(I)=X((I-NUP)MODULO NX), WHERE  NX
      EXCEEDS 0, NUP IS ARBITRARY, AND  X(1...NX) CAN BE ANY MODE.
      EQUIV(X,ROTX) IS OK.

RPLFMT (FMT,FMTNEW)                                      FAP,  17 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SOMEWHERE FOLLOWING CALL RPLFMT THERE MUST APPEAR AN INPUT OR OUTPUT
      STATEMENT USING THE FORMAT FMT.  THIS STATEMENT IS FOUND AND THE FORMAT
      FMTNEW SUBSTITUTED FOR FMT.

RSKIP  (NTAPE,NRECS,EOF)                                 FAP,  37 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - (IOS),(TRC),(TCO),(TEF),(RDS),
           (BSR).
      SKIPS  NRECS PHYSICAL RECORDS FORWARD ON LOGICAL TAPE NTAPE
      (BACKWARDS IF NRECS NEGATIVE AND NO ACTION IF NRECS = 0).  SET  EOF =
      0.0  NORMALLY BUT = 1.0 IF FOUND END-OF-FILE IN SKIPPING FORWARD
      (NO CHECK FOR END-OF-FILE MADE FOR BACKSKIPPING).

RVPRTS (SYM,ANT,N)                            FAP, SECONDARY ENTRY OF CHPRTS
      SETS  SYM(1...LS) = SYM(LS...1)  AND  ANT(1...LA) = ANT(LA...1)
      WHERE LS=LA=N/2 IF N EVEN, LS=(N+1)/2 LA=(N-1)/2 IF N ODD.
      N  MUST EXCEED 0 .  MODES OF SYM AND ANT ARBITRARY.
```

```
SAME  F(IX1)                                                FAP,    1 REGISTER
          OTHER ENTRY - XSAME.  NO TRANSFER VECTOR.
      FUNCTION DOES NOTHING BUT SUPPLY FLOATING POINT LABEL FOR ITS
      ARGUMENT WHICH IS ANY MODE.

SCPSCL (SPACE,NOPTP,YTOP,YBOT,CONVK,CONVL)                  FAP,   33 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      (SPECIAL SUBROUTINE OF GRAPH.)  SETS  SPACE(1...NOPTS) AS FORTRAN
      INTEGERS WHERE  SPACE(I) = XFIXF(CONVK+CONVL*X(I))  WHERE  X(I) =
      MAX1F(MIN1F(SPACE(I),YTOP),YBOT), AND WHERE NOPTP EXCEEDS 0,
      YTOP EXCEEDS YBOT.

SEARCH (LV,VECTOR,XNUM,INDEX)                               FAP,   25 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SEARCHES  VECTOR(1...LV)  LOOKING FOR ELEMENT IDENTICALLY = XNUM.
      IF ANY EXIST SETS INDEX = LOWEST VALUE FOR WHICH  VECTOR(INDEX) = XNUM.
      IF NOT SETS  INDEX = 0 .  MODES OF VECTOR, XNUM ARBITRARY.  LV MUST
      EXCEED  0  (IF = 0  INDEX IS SET = 0).

SEQSAC (ARGLO,ARGDEL)                                       FAP,   94 REGISTERS
          OTHER ENTRIES - NEXCOS,NEXSIN.  TRANSFER VECTOR - COS,SIN.
      NO VISIBLE OUTPUTS.  SETS ENTRIES  NEXCOS  AND  NEXSIN  SO THAT THEIR
      OUTPUTS ON SUBSEQUENT USES WILL BE FOR ARGUMENT VALUES INCREMENTED BY
      ARGDEL  WHERE  ARGLO  AND  ARGDEL  ARE IN RADIANS.

SETAPTF(X,XNEW,FVALUE)                          FAP, SECONDARY ENTRY OF INDEX
      PUTS  XNEW  IN MACHINE LOCATION CONTAINING  X,  THEN SETS ACCUMULATOR
      EQUAL  FVALUE.  MODES IMMATERIAL BUT VALUE MISNAMED IF  FVALUE  IS
      FIXED POINT.

SETESTF(X,XNEW,XCRTCL)                          FAP, SECONDARY ENTRY OF INDEX
      PUTS  XNEW  IN MACHINE LOCATION CONTAINING  X,  THEN SETS ACCUMULATOR  =
      -1.0  IF  XNEW  LSTHN  XCRTCL,  =  0.0  IF  XNEW  =  XCRTCL,  =  +1.0
      IF  XNEW  GRTHN  XCRTCL,  WHERE MODE OF ARGUMENTS IMMATERIAL AND WHERE
      PLUS AND MINUS ZERO TREATED AS EQUAL.

SETINO (ITAPE,ZIFNEW,NRECS,ERR)                        FORTRAN,   84 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - XLIMIT,FSKIP,(RWT),(TSB),
          (RLR).
      SETS  NRECS=0  AND REWINDS LOGICAL TAPE  ITAPE.  THEN RETURNS IF
      ZIFNEW=0.0 .  IF  ZIFNEW NOT= 0.0  ASSUMES TAPE CONTAINS INDATA-OUDATA
      FORMAT RECORDS AND USES  INDATA  TO SPACE TO THE ZERO RECORD NUMBER
      RECORD, LEAVING TAPE POSITIONED TO REWRITE THAT RECORD AND SETTING
      NRECS  =  RECORD COUNT PRIOR TO THAT RECORD.  SETS  ERR=7.0  IF  ITAPE
      NOT IN CLOSED RANGE  1...20  AND  INDATA  SETS  ERR=1.0,2.0,...,6.0  IF
      OTHER TROUBLE.

SETK   (C,X1,X2,...,XN)                                     FAP,   37 REGISTERS
          OTHER ENTRIES - SETKS,SETVEC.  NO TRANSFER VECTOR.
      SETS X1 = X2 = ... = XN = C WHERE C IS ANY MODE.  N MUST EXCEED 0 .

SETK   (C,X1,X2,...,XN) - II                          FORTRAN, 63 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - SETUP,STORE,RETURN.
      SAME FUNCTION AS FAP VERSION OF SETK.
```

```
SETKP  (C1,X11,X12,...,X1N1,STOP,C2,X21,X22,...,              FAP,  40 REGISTERS
        X2N2,STOP,......,CM,XM1,XM2,...,XMNM)
          OTHER ENTRY - SETVCP.  TRANSFER VECTOR - SETK,SETVEC.
      CALL SETKP(ABOVE ARGUMENTS) WHERE STOP = OCT777777712345 IS EQUIVALENT
      TO
           CALL SETK(C1,X11,X12,...,X1N1)
           CALL SETK(C2,X21,X22,...,X2N2)
               ETC
           CALL SETK(CM,XM1,XM2,...,XMNM).

SETKS  (C1,X1,C2,X2,...,CN,XN)                  FAP, SECONDARY ENTRY OF SETK
      SETS X1=C1, X2=C2,..., XN=CN IN THAT ORDER.  WHERE
      C1,C2,..., ARE ANY MODES.  EQUIV(CM,XL) OK FOR ANY M,L PAIR.

SETKS  (C1,X1,C2,X2,...,CN,XN) - II                   FORTRAN,  91 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - SETUP,ARG,STORE,RETURN.
      SAME FUNCTION AS FAP VERSION OF SETKS

SETKV  (C,LX,X)                                               FAP,  15 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  X(1...LX) = C,  WHERE  C  IS ANY MODE.  EQUIV(C,SOME X(I)) OK,
      BUT INITIAL VALUE OF  C  IS ALWAYS THE QUANTITY STORED.  STRAIGHT RETURN
      IF  LX  LSTHN 1.

SETKVS (C1,L1,X1,C2,L2,X2,...,CN,LN,XN)                       FAP,  25 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  X1(1...L1)=C1,  X2(1...L2)=C2, ..., XN(1...LN)=CN  IN
      THAT ORDER, WHERE C1,C2,... ARE ANY MODE.  IF ANY LX LSTHN= 0,
      CORRESPONDING  X  NOT MODIFIED.  EQUIV (ANY TWO ARGUMENTS) OK.

SETLIN (BASE,DELTA,LX,X)                                      FAP,  27 REGISTERS
          OTHER ENTRY - XSTLIN.  NO TRANSFER VECTOR.
      SETS  X(I)=BASE+(I-1)*DELTA,  I=1...LX.  EQUIV(BASE,DELTA,ANY X(I)) OK,
      INPUT VALUES OF  BASE  AND  DELTA  ALWAYS USED.  STRAIGHT RETURN IF
      LX  LSTHN 1.

SETLNS (BASE1,DELTA1,LX1,X1,BASE2,DELTA2,                     FAP,  39 REGISTESR
        LX2,X2,...,BASEN,CELTAN,LXN,XN)
          NO OTHER ENTRIES.  TRANSFER VECTOR - SETLIN,XSTLIN.
      CALL SETLNS(ABOVE ARGUMENTS) IS EQUIVALENT TO
           CALL SETLIN(BASE1,DELTA1,LX1,X1)
           CALL SETLIN(BASE2,DELTA2,LX2,X2)
               ETC
           CALL SETLIN(BASEN,DELTAN,LXN,XN)
      EXCEPT THAT FOR EACH DELTA WHICH, INTERPRETED AS FIXED POINT, IS LSTHN=
      10000 OR WHICH HAS BIT 9 = 0 SUBROUTINE XSTLIN IS USED IN PLACE OF
      SETLIN.

SETSBV (SUBRU,SUBRUV,ARG1,ARG2,...,ARGN)        FAP, SECONDARY ENTRY OF LOCATE
      SETS  SUBRUV(1...N+4) = SUBROUTINE VECTOR AS REQUIRED BY A CALL CALL2
      STATEMENT, WHERE  N  MAY = 0 .  SETS  SUBRUV(1) = SUBRU = SUBROUTINE
      PROXY NAME, (2) = N, (3) = OCT 777777777777, (4) = IXARG1,...,  (N+3) =
      IXARGN, (N+4) = OCT 777777777777, WHERE IXARG = INDEX WITH RESPECT TO
      COMMON BLOCK OF ARG.  SUBRUV IS A MIXED MODE VECTOR AS SHOWN.
```

```
SETUP  (LOCALL,NARGS,XR1,XR2)                      FAP, SECONDARY ENTRY OF LOCATE
      CALL SETUP IS USED AS FIRST INSTRUCTION OF A SUBROUTINE.  SETS LOCALL =
      MACHINE ADDRESS OF CALL STATEMENT CALLING THE SUBROUTINE, SETS
      NARGS = NO. OF ARGUMENTS IN THAT CALL STATEMENT, SETS  XR1 AND XR2
      (DECREMENTS) = INDEX REGISTERS 1 AND 2 .

SETVCP (X1,C11,C12,...,C1N1,STOP,                  FAP, SECONDARY ENTRY OF SETKP
        X2,C21,C22,...,C2N2,STOP,
        ......,XM,CM1,CM2,...,CMNM)
      CALL SETVCP(ABOVE ARGUMENTS) WHERE STOP = OCT777777712345 IS EQUIVALENT
      TO
           CALL SETVEC(X1,C11,C12,...,C1N1)
           CALL SETVEC(X2,C21,C22,...,C2N2)
               ETC
           CALL SETVEC(XM,CM1,CM2,...,CMNM).

SETVEC (X,C1,C2,...,CN)                            FAP, SECONDARY ENTRY OF SETK
      SETS  X(1...N) = C1,C2,...,CN  WHERE  C1,C2,...  ARE ANY MODE.

SEVRAL (SUBRUA,A1,A2,...,ANA,SUBRUB,B1,B2,...,BNB,          FAP, 416 REGISTERS
        ......,SUBRUZ,Z1,Z2,...,ZNZ)
           OTHER ENTRIES - PLURAL,DO,IF.  TRANSFER VECTOR - LOCATE,WHERE.
      THE ABOVE CALL SEVRAL STATEMENT ASSUMES THE SUBROUTINES SUBRA...SUBRZ
      WITH PROXY NAMES SUBRUA...SUBRUZ HAVE BEEN PREVIOUSLY LOCATED BY A
      CALL LOCATE STATEMENT, IN WHICH THE ARGUMENT LISTS ARE OPTIONAL, BUT IF
      PRESENT MUST BE CORRECT IN NUMBER.  THE FUNCTION IS EQUIVALENT TO
                            CALL SUBRA(A1,A2,...,ANA)
                            CALL SUBRB(B1,B2,...,BNB)
                                ETC
                            CALL SUBRZ(Z1,Z2,...,ZNZ)
      NONE OF SUBRA...SUBRZ MAY USE DATA BEYOND THE END OF THEIR
      CALLING SEQUENCES.  THE PSEUDO ENTRIES DO AND IF MAY BE USED
      AS SUBROUTINES TO BE OPERATED AND DO NOT NEED TO BE LOCATED.  PLURAL
      MAY NOT BE OPERATED BY SEVRAL.

SHFTR1 (NSHFT,IV,LIV,IVSH,IANS)                             FAP,  70 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  IVSH(1...LIV) FROM  IV(1...LIV) WHERE  IVSH(I) = IV(I) SHIFTED
      RIGHT ARITHMETICALLY  N  BITS (LEFT IF N NEGATIVE, NO SHIFT IF N = 0)
      WHERE N = NSHFT (MODULO 36).  SETS  IANS = 0 IF ALL OK, = +1 IF
      OVERFLOW (ON NEG NSHFT, BUT SHIFTING COMPLETED), = -3 IF LIV
      LSTHN 1 .  EQUIV(IVSH,IV)  OK.

SHFTR2 (NSHFT,IV,LIV,IVSH,IANS)                             FAP,  72 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      SETS  IVSH(1...LIV) FROM  IV(1...LIV) WHERE  IVSH(I) = IV(I)
      SHIFTED RIGHT LOGICALLY  N  BITS (LEFT IF  N  NEGATIVE, NO SHIFT IF
      N = 0) WHERE  N = NSHFT (MODULO 36).  SETS  IANS = 0  IF ALL OK,
      = +1 IF OVERFLOW (ON NEG NSHFT, BUT SHIFTING COMPLETED), = -3
      IF LIV LSTHN 1 .  EQUIV(IVSH,IV) OK.

SHUFFL (ITPRD,NITEMS,ISPACE,IXSHUF)                    FORTRAN,  101 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - GETRD1,SEARCH,SIZEUP.
```

(CONTINUED NEXT PAGE)

SETS IXSHUF(1...NITEMS) AS A RANDOM ORDERING OF THE INTEGERS 1...NITEMS INDEPENDENT FROM PREVIOUS ORDERINGS, IF ANY, FORMED BY PRIOR CALLS OF SHUFFL WITHIN THE PRESENT EXECUTION. ASSUMES LOGICAL TAPE ITPRD CONTAINS RAND RANDOM DIGITS BCD CARDS, USES 5*NITEMS NEW RANDOM DIGITS FOR EACH CALL (SUPPLIED BY GETRD1), AND NEVER REWINDS ITPRD. NEEDS ISPACE(1...NITEMS) FOR SCRATCH. DOES NOT CHECK LEGALITY OF ITPRD BUT GIVES STRAIGHT RETURN WITH NO OUTPUT IF NITEMS LSTHN 1 .

SIFT (X,MESH,LXSFTD,XSFTD) FAP, 30 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XSFTD(1...LXSFTD) = X(1,1+MESH,1+2*MESH,...,1+(LXSFTD-1)*MESH) WHERE X IS ANY MODE. REQUIRE LXSFTD GRTHN= 1, MESH GRTHN= 0 . STRAIGHT RETURN WITH NO OUTPUT IF EITHER ILLEGAL. EQUIV(X,XSFTD) OK.

SIMEQ (N,LN,LM,A,B,D,E,ERR) FAP, 441 REGISTERS
OTHER ENTRY - DETRM. NO TRANSFER VECTOR.
SOLVES MATRIX EQUATION AX=B FOR X, WHERE A(1...LN,1...LN) IS DIMENSIONED A(N, ARBITRARY) WITH LN LSTHN= N, B(1...LN,1...LM) IS DIMENSIONED B(N,N1) WITH LM LSTHN= N1 LSTHN= N BUT LM MAY EXCEED LN, AND THE OUTPUT X(1...LN,1...LM) HAS SAME DIMENSIONS AS B BUT REPLACES THE A MATRIX. D ON INPUT IS SCALE TO MULTIPLY DETERMINANT BY, ON OUTPUT D = SCALED VERSION OF DETERMINANT OF A (WILL = 0. IF A SINGULAR). B IS DESTROYED. E(1...LN) MUST BE AVAILABLE FOR SCRATCH. SETS ERR = 0. IF ALL OK, = 1. IF UNDERFLOW OR OVERFLOW, = 2. IF A IS SINGULAR.

SINTBL (N,SINTAB) FAP, SECONDARY ENTRY OF COSTBL
SETS SINTAB(1...N+1) = S(0...N) WHERE S(I) = SIN(I*PI/N). STRAIGHT RETURN IF N LSTHN= 0 .

SINTBX (N,ISINTB) FAP, SECONDARY ENTRY OF COSTBL
SETS ISINTB(1...N+1) = IS(0...N) WHERE IS(I) = SIN(I*PI/N) AND IS FXD.PT. WITH BINARY PT. BETWEEN SIGN AND BIT 1 AND 1.0 IS SET = OCT 377777777777. STRAIGHT RETURN IF N LSTHN= 0 .

SISP (SAX,AAX,L,SINTAB,M, FAP, SECONDARY ENTRY OF COSP
JMIN,JMAX,TYPE,SINTR)
SETS SINTR(1...JMAX-JMIN+1) = ST(JMIN...JMAX) WHERE ST(J) = SUM (FROM I = 0 TO L) OF (X(I)*SIN(I*J*PI/M)), WHERE X(0...L) = SAX(1...L+1) FOR J ODD, = AAX(1...L+1) FOR J EVEN, GIVEN THE TABLE SINTAB(1...M+1) = S(0...M) WITH S(I) = SIN(I*PI/M). TYPE =0.0 SPECIFIES SAX, AAX, AND COSTAB TO BE FXD.PT., TYPE NOT = 0.0 DESIGNATES EVERYTHING FLTG.PT. EQUIV(SAX,AAX) OK. IF M NEGATIVE, ITS MAGNITUDE IS USED AND ST(...) IS ADDED INTO SINTR(...) RATHER THAN STORED INTO IT. STRAIGHT RETURN IF L LSTHN= 0, OR M=0, OR JMIN LSTHN= 0, OR JMAX LSTHN= JMIN OR GRTHN M.

SIZEUP (X,LX,INDEX) FAP, 136 REGISTERS
OTHER ENTRY - SIZUPL. NO TRANSFER VECTOR.
SETS INDEX(1...LX) FROM X(1...LX) SUCH THAT X(INDEX(I+1)) IS ALGEBRAICALLY GRTHN= X(INDEX(I)) WHERE X IS ANY MODE. STRAIGHT RETURN IF LX LSTHN 1 . EQUAL VALUES OF X ARE NOT NECESSARILY IN THE SAME ORDER AS THEY OCCURRED IN X.

SIZUPL (X,LX,INDEX) FAP, SECONDARY ENTRY OF SIZEUP
SETS INDEX(1...LX) FROM X(1...LX) SUCH THAT X(INDEX(I+1)) IS
LOGICALLY GRTHN= X(INDEX(I)) WHERE X IS ANY MODE. STRAIGHT RETURN
IF LX LSTHN 1 . EQUAL VALUES OF X ARE NOT NECESSARILY IN THE
SAME ORDER AS THEY OCCURRED IN X.

SMPRDV (X,LX,N,XBASE,SXMB2N) FAP, SECONDARY ENTRY OF POWER
SETS SXMB2N = SUM (FROM I=1 TO LX) OF (X(I)-XBASE)**N, WHERE
N IS ARBITRARY. STRAIGHT RETURN IF LX LSTHN 1.

SMPSON (JOB,X,LX,DELX,XINT,IANS) FORTRAN, 317 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
FOR JOB=0 SETS XINT = SIMPSON'S RULE INTEGRAL OF X(1...LX) WITH
INCREMENT DELX, TRAPEZOIDAL RULE BEING USED BETWEEN X(LX-1) AND
X(LX) IF LX IS EVEN. FOR JOB GRTHN 0 DOES SAME AS JOB=0 BUT
LEAVES X(1...LX) SCALED BY WEIGHTING COEFFICIENTS DELX*(1/3,4/3,2/3,
4/3,...,4/3,1/3) IF LX ODD OR BY DELX*(1/3,4/3,2/3,...,4/3,5/6,1/2)
IF LX EVEN. FOR JOB LSTHN 0 MERELY REMOVES ABOVE SCALING FROM
X(1...LX). REQUIRE DELX NOT= 0.0 (NOT CHECKED) AND LX GRTHN= 3
(STRAIGHT RETURN WITH NO OUTPUT IF ILLEGAL).

SPCOR2 (NRX,NCX,XX,NRY,NCY,YY,MXACC,ILGR, FORTRAN, 291 REGISTERS
NRZ,ILGC,INC,NCZ,ZZ,SPACE,IANS)
NO OTHER ENTRIES. TRANSFER VECTOR - XLOC,STZ,FXDATA,QXCOR1,FLDATA.
SETS ZZ(1...NRZ*NCZ) = Z(ILGR...ILGR+NRZ-1,ILGC...ILGC+NCZ-1) WHERE
Z(I,J) = SUM (FROM K=1 TO NCX) OF SUM (FROM L=1 TO NRX) OF
(X(K+I-1,L+J1)*Y(K,L) WHERE J1 = ILGC,ILGC+INC,...,ILGC+(NCZ-1)*INC,
AND X(1...NRX,1...NCX) = XX(1...NRX*NCX), Y(1...NRY,1...NCY) =
YY(1...NRY*NCY). COMPUTATIONS ARE APPROXIMATE. XX AND YY ARE
CONVERTED TO INTEGER SEQUENCES WITH MAXIMUM MAGNITUDE = MXACC
(1 TO 1000) DURING COMPUTATIONS, BUT ARE REFLOATED AFTERWARDS.
SPACE(1...MIN(NRX,NRY)+10*(MXACC+1)+1) USED FOR SCRATCH. SETS IANS
= 0 IF ALL OK, = ARGUMENT NUMBER IF ONE IS ILLEGAL.

SPLIT (X,LX,TYPE,SYM,ANT) FAP, 224 REGISTERS
OTHER ENTRY - REFIT. NO TRANSFER VECTOR.
SETS SYM(1...LS) AND ANT(1...LA) FROM X(1...LX), WHERE LS = LA =
LX/2 FOR LX EVEN, LS = (LX+1)/2 AND LA = (LX-1)/2 FOR LX ODD, AND WHERE
FOR LX EVEN SYM(I) = X(LS+I)+X(LS+1-I) AND ANT(I) = X(LA+I)-
X(LA+1-I), BUT WHERE FOR LX ODD SYM(1) = X(LS) SYM(I) = X(LS-1+I)
+X(LS+1-I) FOR I = 2...LS AND ANT(I) = X(LS+I)-X(LS-I). TYPE = 0.
SIGNIFIES X IS FXD.PT. (SYM AND ANT WILL HAVE SAME BINARY PT.),
TYPE NOT = 0.0 SIGNIFIES SYM, ANT, X FLTG.PT. ANT IS OUTPUT ONLY IF
LA GRTHN 0 . STRAIGHT RETURN IF LX LSTHN= 0 . EQUIV(SYM,X) OK ONLY IF
EQUIV(ANT,X(LS+1)) ALSO HOLDS.

SQRDEV (X,XBASE,LX,SSQXMB) FAP, SECONDARY ENTRY OF SQRDFR
SETS SSQXMB = SUM (FROM I = 1 TO LX) OF (X(I)-XBASE)SQUARED.
STRAIGHT RETURN IF LX LSTHN 1.

SQRDFR (X,Y,LXY,SSQXMY) FAP, 36 REGISTERS
OTHER ENTRY - SQRDEV. NO TRANSFER VECTOR.
SETS SSQXMY = SUM (FROM I = 1 TO LX) OF (X(I)-Y(I))SQUARED.
STRAIGHT RETURN IF LX LSTHN 1.

SQRMLI (MLIVEC,ILO,IHI,MLISQR,IANS) FAP, 55 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS MLISQR(1...IHI-ILO+1) = SQUARES OF MLIVEC(ILO...IHI)
ASSUMING MLIVEC ARE MACHINE LANGUAGE INTEGERS, WHERE 1 LSTHN= ILO
LSTHN= IHI. SETS IANS = 0 IF ALL OK, = -1 IF ILLEGAL ILO OR IHI,
= -2 IF ONE OF THE SQUARES OVERFLOWS (IMMEDIATE RETURN IN THIS CASE).

SQROOT (X,LX,XSQRTD) FAP, 24 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - SQRT.
SETS XSQRTD(1...LX) = SQUARE ROOT (MAGNITUDE(X(1...LX))).
EQUIV(XSQRTD,X) OK. STRAIGHT RETURN IF LX LSTHN 1.

SQRSUM (X,LX,SUMSQX) FAP, 36 REGISTERS
OTHER ENTRY - XSQSUM. NO TRANSFER VECTOR.
SETS SUMSQX = SUM (FROM I = 1 TO LX) OF X(I)*X(I). STRAIGHT RETURN
IF LX LSTHN 1 .

SQUARE (X,LX,XSQRD) FAP, 32 REGISTERS
OTHER ENTRY - XSQUAR. NO TRANSFER VECTOR
SETS XSQRD(1...LX) = X(1...LX) SQUARED. EQUIV(X,XSQRD) OK. STRAIGHT
RETURN IF LX LSTHN 1.

SRCH1 (JOB,LV,V,VALUE,INDEX) FORTRAN, 93 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - XACTEQ.
SEARCHES V(1...LV) FOR VALUE SO THAT VALUE = V(INDEX).
IF JOB = 1 SEARCHING BEGINS AT V(1), IF JOB = 2 SEARCHING
BEGINS AT V(LV). LV MUST BE GRTHN= 1 .

STEPC F(ARG) FAP, SECONDARY ENTRY OF DELTA
HAS VALUE 1.0 OR 0.0 ACCORDING AS SIGN BIT OF ARG IS PLUS OR
MINUS.

STEPL F(ARG) FAP, SECONDARY ENTRY OF DELTA
HAS VALUE 1.0 IF ARG (ANY MODE) HAS VALUE GRTHN= MINUS ZERO.
OTHERWISE HAS VALUE = 0.0 .

STEPR F(ARG) FAP, SECONDARY ENTRY OF DELTA
HAS VALUE = 1.0 IF ARG (ANY MODE) EXCEEDS ZERO.
OTHERWISE HAS VALUE = 0.0 .

(STH) FAP, SECONDARY ENTRY OF ONLINE
SERVES SAME FUNCTION AS STANDARD FORTRAN (STH) SUBROUTINE.

(STHD) FAP, SECONDARY ENTRY OF ONLINE
SERVES SAME FUNCTION AS STANDARD FORTRAN (STHD) SUBROUTINE.

(STHM) FAP, SECONDARY ENTRY OF ONLINE
SERVES SAME FUNCTION AS STANDARD FORTRAN (STHM) SUBROUTINE.

STORE (ARGU,LOCALL,NUMARG,IXVECT) FAP, SECONDARY ENTRY OF LOCATE
STORES THE VALUE ARGU (ANY MODE) AS ELEMENT NO. IXVECT OF THE VECTOR
WHICH IS ARGUMENT NO. NUMARG OF THE CALL STATEMENT AT MACHINE ADDRESS
LOCALL. LOCALL SHOULD BE NON-NEG, NUMARG MUST EXCEED 0 BUT IXVECT
IS UNRESTRAINED.

```
STZ     (LX,X)                                              FAP,  14 REGISTERS
             NO OTHER ENTRIES.  NO TRANSFER VECTOR.
        SETS  X(1...LX) = ZERO,  WHERE  X  IS ANY MODE.  STRAIGHT RETURN IF
        LX LSTHN 1.

STZS    (LX1,X1,LX2,X2,...,LXN,XN)                          FAP,  24 REGISTERS
             NO OTHER ENTRIES.  NO TRANSFER VECTOR.
        SETS ALL ELEMENTS OF THE VECTORS X1(1...LX1),X2(1...LX2),...,XN(1...LXN)
        EQUAL ZERO (MODE ARBITRARY), EXCEPT BYPASSES EACH VECTOR X
        FOR WHICH LX LSTHN 1.  N SHOULD EXCEED ZERO.

SUBK    (C,X1,X2,...,XN)                          FAP, SECONDARY ENTRY OF ADDK
        SETS  X1=X1-C, X2=X2-C, ..., XN=XN-C.  EQUIV(ANY ARGUMENTS) OK,
        BUT INITIAL VALUE OF   C  IS ALWAYS THE SUBTRAHEND.  STRAIGHT RETURN IF
        N=0.

SUBKS   (C1,X1,Y1,C2,X2,Y2,...,CN,XN,YN)          FAP, SECONDARY ENTRY OF ADDK
        SETS  Y1=X1-C1,  Y2=X2-C2,  ...,  YN=XN-CN.  EQUIV(ANY TWO ARGUMENTS)
        OK BUT MAY CHANGE INPUTS  CJ  OR  XJ.  PROCESSING IS LEFT TO RIGHT.
        STRAIGHT RETURN IF  N=0.

SUM     (X,LX,SUMX)                                         FAP,  23 REGISTERS
             OTHER ENTRY - XSUM.  NO TRANSFER VECTOR.
        SETS  SUMX = SUM (FROM I= 1 TO LX) OF X(I).  STRAIGHT RETURN
        IF LX LSTHN 1.

SUMDEV (X,XBASE,LX,SUMXMB)                      FAP, SECONDARY ENTRY OF SUMDFR
        SETS  SUMXMB = SUM (FROM I = 1 TO LX) OF (X(I)-XBASE).  STRAIGHT RETURN
        IF  LX  LSTHN 1.

SUMDFR (X,Y,LXY,SUMXMY)                                     FAP,  44 REGISTERS
             OTHER ENTRIES - XSMDFR,SUMDEV,XSMDEV.  NO TRANSFER VECTOR.
        SETS  SUMXMY = SUM (FROM I = 1 TO LX) OF (X(I)-Y(I)).   STRAIGHT RETURN
        IF  LX  LSTHN 1.

SWITCHF(ISENSE)                                             FAP,   15 REGISTERS
             NO OTHER ENTRIES.  NO TRANSFER VECTOR.
        HAS VALUE  =  1.0  IF SIMULTANEOUSLY  ISENSE  IS IN THE CLOSED RANGE
        1...6  AND THE CORRESPONDING SENSE SWITCH IS DEPRESSED (ON).  OTHERWISE
        HAS VALUE  =  0.0 .

TAMVL  (X,LX,LAVG,AVGL)                                     FAP,   63 REGISTERS
             OTHER ENTRY - TAMVR.  NO TRANSFER VECTOR.
        SETS  AVGL(I)  =  (1/(LX-I+1)) * (SUM  (FROM J=I TO LX)  OF  X(J) )
        FOR  I=1...LAVG.  STRAIGHT RETURN WITH NO OUTPUT IF  LX  OR  LAVG
        LSTHN  1,  OR IF  LAVG  GRTHN  LX.

TAMVR  (X,LX,LAVG,AVGR)                          FAP, SECONDARY ENTRY OF  TAMVL
        SETS  AVGR(I)  =  (1/(LX-I+1)) * (SUM  (FROM J=I TO LX-I+1)  OF  X(J) )
        FOR  I=1...LAVG.  STRAIGHT RETURN WITH NO OUTPUT IF  LX  OR  LAVG
        LSTHN  1,  OR IF  LAVG  GRTHN  LX.
```

TIMA2B (LOCA,LOCB,MINACC,SECS)   FAP (7094), 124 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
MACHINE ADDRESS LOCA CONTAINS A PROGRAM WHICH WHEN DONE SENDS CONTROL TO LOCB AND WHICH MAY BE REPETITIVELY OPERATED. TIMA2B SETS SECS = TIME IN SECONDS (TO ACCURACY OF 1 PART IN MINACC PARTS) OF 1 OPERATION OF THE PROGRAM EXCLUDING TIME OF OPERATION AT LOCB. ASSUMES INTERVAL TIMER IS ON.

TIMSUB (MINACC,SECS)   FAP, 229 REGISTERS
OTHER ENTRY - INTMSB. TRANSFER VECTOR - TIMA2B.
CALL TIMSUB IS IMMEDIATELY FOLLOWED BY CALL SUBRU(A,B,...,Z) OR BY AN X=SOMEF(...) TYPE STATEMENT WHERE SUBRU OR SOMEF MAY BE OPERATED REPETITIVELY. TIMSUB SETS SECS = TIME IN SECONDS (TO AN ACCURACY OF 1 PART IN MINACC PARTS) THAT ONE OPERATION OF SUBRU OR SOMEF REQUIRES. IF SUBRU OR SOMEF MAY NOT BE OPERATED REPETITIVELY WITHOUT REGENERATING ITS INPUTS, THE INPUT SETUP SEQUENCE SHOULD IMMEDIATELY PRECEDE THE CALL TIMSUB(MINACC,SECS) STATEMENT AND IMMEDIATELY PRECEDING THE INPUT SETUP SEQUENCE SHOULD APPEAR A CALL INTMSB STATEMENT. ASSUMES INTERVAL TIMER IS ON.

TINGL (YOFX,LY,DELX,TING)   FAP, 43 REGISTERS
OTHER ENTRY - TINGLA. NO TRANSFER VECTOR.
SETS TING = TRAPEZOIDAL INTEGRAL OF YOFX(1...LY) WITH INCREMENT DELX (MAY BE NEGATIVE) BUT STRAIGHT RETURN WITH NO OUTPUT IF LY LSTHN 2 .

TINGLA (YOFX,LY,DELX,TINGA)   FAP, SECONDARY ENTRY OF TINGL
SETS TINGA = TRAPEZOIDAL INTEGRAL OF MAGNITUDES OF YOFX(1...LY) WITH INCREMENT DELX BUT STRAIGHT RETURN WITH NO OUTPUT IF LY LSTHN 2 . TINGA WILL BE NEGATIVE IF DELX IS.

TRMINO (ITAPE,NBAKUP)   FORTRAN, 67 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - XLIMIT,OUDATA,FSKIP,(RWT).
CREATES, VIA OUDATA, A ZERO RECORD NUMBER DUMMY RECORD ON LOGICAL TAPE ITAPE, THEN LEAVES TAPE NBAKUP FILES CLOSER TO LOAD POINT THAN ITS POSITION AT INSTANT OF CALL TRMINO STATEMENT, EXCEPT REWINDS IF NBAKUP LSTHN 0 . NBAKUP = 0 LEAVES TAPE READY TO READ DUMMY RECORD. REQUIRE ITAPE IN CLOSED RANGE 1...20 OTHERWISE STRAIGHT RETURN WITH NO OUTPUT.

(TSH)   FAP, SECONDARY ENTRY OF REREAD
SERVES SAME FUNCTION AS STANDARD FORTRAN (TSH) SUBROUTINE.

(TSHM)   FAP, SECONDARY ENTRY OF REREAD
SERVES SAME FUNCTION AS STANDARD FORTRAN (TSHM) SUBROUTINE.

UNPAKN (N,LD,D,SCALE)   FAP, 78 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS D(1...LD) AS UNPACKED, FLOATED AND RESCALED FORM OF PACKED INPUT D(1...(LD+N-1)/N), THE PACKED INPUT HAVING ORIGINALLY BEEN FORMED BY A CALL PAKN(N,LD,D,SCALE) STATEMENT. D IS UNCHANGED IF N = 1 . UNPAKN IS APPROXIMATE INVERSE TO PAKN. TO RECOVER FXD.PT. INTEGERS WHICH WERE FLOATED AND PACKED BY PAKN, USE UNPAKN FOLLOWED BY ROUNDING AND FIXING LOOP.

VARARG (LOCS) FAP, 44 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
USED AT VERY BEGINNING OF A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE AS FOLLOWS - CALL VARARG(LOCS) - GO TO 20 - 10 RETURN - WHERE STATEMENT 20 BEGINS THE COMPUTATIONS WHICH TERMINATE WITH A GO TO 10 STATEMENT. IN THIS USAGE VARARG SETS LOCS(1...N+1) = XLOCF(ARG1),...,XLOCF(ARGN), 0 WHERE ARGJ = JTH ARGUMENT OF CALL STATEMENT WITH N TOTAL ARGUMENTS, AND MODIFIES RETURN STATEMENT AT 10 FOR PROPER LINKAGE.

VDOTV (X,Y,LXY,DVSR,XDYODV) FAP, 25 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XDYODV = (1/DVSR) * (SUM (I=1 TO LXY) OF X(I)*Y(I)) PROVIDED DVSR NOT= 0.0 . IF DVSR = 0.0, SETS XDYODV = 1.0 AND SETS DVSR = SUM (I=1 TO LXY) OF X(I)*Y(I). STRAIGHT RETURN WITH NO OUTPUT IF LXY LSTHN 1 .

VDVBYV (X,Y,LXY,XDVBYY) FAP, 22 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
SETS XDVBYY(1...LXY) FROM X(1...LXY) AND Y(1...LXY), WHERE XDVBYY(I) = X(I)/Y(I). EQUIV(XDVBYY, X OR Y) OK. DIVISION BY ZERO NOT TESTED FOR BY VDVBYV. STRAIGHT RETURN IF LXY LSTHN 1 .

VECOUT (ITAPE,FMT,X,ILO,IHI) FORTRAN, 66 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - FNDFMT,RPLFMT,(STH),(FIL).
OUTPUTS THE VECTOR RANGE X(ILO...IHI) ONTO LOGICAL TAPE ITAPE ACCORDING TO FMT(I), WHERE FMT(I) IS A NORMLIT FORMAT VECTOR AS DEFINED ABOVE IN CVSOUT. REQUIREMENT THAT 1 LSTHN= ILO LSTHN= IHI NOT CHECKED BY VECOUT.

VINDEXF(I,ICRTCL,IJUMP) FAP, SECONDARY ENTRY OF INDEX
ADDS IJUMP TO MACHINE LOCATION CONTAINING I, THEN SETS ACCUMULATOR = -1.0 IF NEW I LSTHN ICRTCL, = 0.0 IF NEW I = ICRTCL, = +1.0 IF NEW I GRTHN ICRTCL, WHERE +0 AND -0 TREATED AS EQUAL.

VMNUSV (X,Y,LXY,XMNUSY) FAP, SECONDARY ENTRY OF VPLUSV
SETS XMNUSY(1...LXY) FROM X(1...LXY) AND Y(1...LXY), WHERE XMNUSY(I) = X(I)-Y(I). EQUIV(XMNUSY, X OR Y) OK. STRAIGHT RETURN IF LXY LSTHN 1 .

VOUT (ITAPE,NSPACE,X,XNAME,XFMT,ILO,IHI) FORTRAN, 104 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - CARIGE,HRADJ,(STH),(FIL), VECOUT.
OUTPUTS VECTOR RANGE X(ILO,...,IHI) ONTO LOGICAL TAPE ITAPE ACCORDING TO XFMT(I), WHERE XFMT(I) IS A NORMLIT FORMAT VECTOR AS DEFINED IN CVSOUT ABOVE, PRECEDED BY 1) NSPACE SPACES (OR A PAGE RESTORE IF NSPACE LSTHN 0), AND 2) A HEADING LINE OF FORM XNAME(ILO,ILO+1,...,IHI) = , WHERE XNAME IS 6 OR LESS HOLLERITH CHARACTERS. IHI MUST BE GRTHN= ILO. (IF =, THE HEADING IS XNAME(ILO).) ILO MUST EXCEED ZERO.

VPLUSV (X,Y,LXY,XPLUSY) FAP, 34 REGISTERS
OTHER ENTRIES - XVPLSV,VMNUSV,XVMNSV. NO TRANSFER VECTOR.
SETS XPLUSY(1...LXY) FROM X(1...LXY) AND Y(1...LXY), WHERE XPLUSY(I) = X(I)+Y(I). EQUIV(XPLUSY, X OR Y) OK. STRAIGHT RETURN IF LXY LSTHN 1 .

```
VRSOUT (ITAPE,NSPACE,FMT,SPACE,X1,X2,...,XN)                FAP,  47 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - CARIGE,VECOUT.
     OUTPUTS QUANTITIES X1,X2,...,XN ONTO LOGICAL TAPE ITAPE ACCORDING TO
     FORMAT FMT(I), WHERE FMT(I) IS A NORMLIT FORMAT VECTOR AS DEFINED IN
     CVSOUT ABOVE, PRECEDED BY NSPACE SPACES (OR PAGE RESTORE IF NSPACE IS
     LSTHN 0).  SPACE(1...N) USED FOR SCRATCH.  EQUIV(SPACE,X1) OK IF N=1.

VSOUT  (ITAPE,NSPACE,X1,X1NAME,X1FMT,ILO1,IHI1,              FAP,  37 REGISTERS
        X2,X2NAME,X2FMT,ILO2,IHI2,...,XN,XNNAME,
        XNFMT,ILON,IHIN)
          NO OTHER ENTRIES.  TRANSFER VECTOR - VOUT.
     CALL VSOUT(ABOVE ARGUMENTS) IS EQUIVALENT TO
          CALL VOUT(ITAPE,NSPACE,X1,X1NAME,X1FMT,ILO1,IHI1)
          CALL VOUT(ITAPE,NSPACE,X2,X2NAME,X2FMT,ILO2,IHI2)
              ETC
          CALL VOUT(ITAPE,NSPACE,XN,XNNAME,XNFMT,ILON,IHIN).

VTIMSV (X,Y,LXY,XTIMSY)                                      FAP,  34 REGISTERS
          OTHER ENTRY - XVTMSV.  NO TRANSFER VECTOR.
     SETS  XTIMSY(1...LXY)  FROM  X(1...LXY)  AND  Y(1...LXY),  WHERE
     XTIMSY(I) = X(I)*Y(I).  EQUIV(XTIMSY, X OR Y) OK.  STRAIGHT RETURN
     IF LXY LSTHN 1 .

WAC    (LY,Y,LA,A)                                       FORTRAN, 107 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  A(1...LA) = AC(0...LA-1) WHERE  AC(L) = SUM (FROM J = 1 TO LY)
     OF (Y(J)*Y(J+L))  WHERE Y(K) TREATED = ZERO FOR K GRTHN LY.  LY AND
     LA MUST EXCEED 0, LA MAY EXCEED LY.

WHERE  (SUBRU,IANS,LOC,NARGS)                 FAP, SECONDARY ENTRY OF LOCATE
     SUBRU IS PROXY NAME OF SUBROUTINE TO BE FOUND THROUGH TABLES
     ESTABLISHED BY PRIOR CALL LOCATE STATEMENT(S).  IF FOUND  WHERE  SETS
     LOC = MACHINE ADDRESS OF ENTRY POINT OF SUBROUTINE WITH PROXY NAME
     SUBRU (ASSUME REAL NAME IS SUBR) AND SETS  NARGS = NO. ARGUMENTS
     IN THE CALL SUBR STATEMENT FOLLOWING THE DEFINITIVE CALL LOCATE.  LOC
     AND NARGS UNDISTURBED IF NOT FOUND.  SETS  IANS = 0 IF FOUND, LSTHN 0
     IF NOT.  IANS = -1 IF TABLES OK, = -2 IF SUBRU FOUND IN A CALL LOCATE
     BUT ASSOCIATED CALL LIST TOO SHORT, = -3 IF NO CALL LOCATE YET MADE,
     = -4 IF EXCESSIVE NO. OF CALL LOCATES.

WHICH F(X1,X2,ZIFX1)                                         FAP,   4 REGISTERS
          OTHER ENTRY - XWHICH.  NO TRANSFER VECTOR.
     HAS VALUE  =  X1  IF  ZIFX1=0.0,  VALUE  =  X2  IF  ZIFX1 NOT= 0.0  .

WLLSFP (LR,R,G,LA,A,C)                                   FORTRAN, 217 REGISTERS
          NO OTHER ENTRIES.  TRANSFER VECTOR - FDOTR,FDOT,MOVE.
     SOLVES FOLLOWING TOEPLITZ MATRIX EQUATION,  SUM (FROM N=0 TO M)
     OF (AA(N)*RR(K-N) = GG(K)  K=0...M, AND FINDS  AA(0...N) GIVEN
     GG(0...M)  AND ANY  RR(0...M,M+1) WITH IMPLIED SYMMETRY  RR(-I)=RR(I)
     FOR WHICH THE (M+1)*(M+1) TOEPLITZ FORM  R(K-N) IS POSITIVE
     DEFINITE.  SUPPOSE  LA  IS POSITIVE (MUST EXCEED 1).  THEN  M  IS
     TAKEN = LA-1 AND WLLSFP SETS  A(1...LA) = AA(0...M)  TAKING
     RR(0...M) FROM R(1...M+1...LR+1) AND GG(0...M) FROM G(1...M+1...LR)
     WHERE  LR  GRTHN= LA,  USING  C(1...2*LR) AS SCRATCH (WILL CONTAIN
     LEVINSON AUXIL SEQUENCE  CC(0...M) PLUS OTHER STUFF).  NOW SUPPOSE
```

(CONTINUED NEXT PAGE)

LA IS NEGATIVE (LSTHN= -2). WLLSFP ASSUMES THAT THIS IS REPEAT CALL
WITH DESIRE TO EXTEND PREVIOUS SOLUTION WITH M=LLA-1 (LLA=
MAGNITUDE(LA)) TO NEW M = LR-1 AND THAT A(1...LLA) AND C(1...LLA)
ARE UNDISTURBED FROM THAT CALL. SOLUTION WILL BE AS BEFORE WITH RESULTS
SET IN A(1...LAA) AND LA SET = LLA.

WRTDAT (ITAPE,DATA,LDATA,IANS)   FAP,   77 REGISTERS
NO OTHER ENTRIES. TRANSFER VECTOR - (IOS),(TCO),(WRS),(RCH),(TRC),
(ETT).
WRITES A BINARY RECORD OF LENGTH LDATA ON LOGICAL TAPE ITAPE FROM
THE FORTRAN-II VECTOR DATA(1...LDATA). SETS IANS = 0 IF ALL OK,
= 2 IF A REDUNDANCY IS ENCOUNTERED, = 3 IF AN END TAPE MARK IS
ENCOUNTERED, = -1 IF ITAPE LSTHN 1 OR GRTHN 20, = -2 IF
LDATA LSTHN 1 .

XACTEQF(X,Y)   FAP,   11 REGISTERS
NO OTHER ENTRIES. NO TRANSFER VECTOR.
HAS VALUE = 0 IF X AND Y ARE IDENTICAL INCLUDING SIGN BIT,
VALUE = 1 IF X GRTHN Y, VALUE = -1 IF X LSTHN Y WHERE
+0 GRTHN -0 AND MODES OF X, Y IMMATERIAL.

XADDK (IC,IX1,IX2,...,IXN)   FAP, SECONDARY ENTRY OF ADDK
SETS IX1=IX1+IC, IX2=IX2+IC, ..., IXN=IXN+IC. EQUIV(ANY ARGUMENTS) IS
OK, BUT INITIAL VALUE OF IC IS ALWAYS THE ADDEND. STRAIGHT RETURN IF
N=0.

XADDKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN) FAP, SECONDARY ENTRY OF ADDK
SETS IY1=IX1+IC1, IY2=IX2+IC2, ..., IYN=IXN+ICN. EQUIV(ANY TWO
ARGUMENTS) OK BUT MAY CHANGE INPUTS ICJ OR IXJ. PROCESSING IS LEFT
TO RIGHT. STRAIGHT RETURN IF N=0.

XARG F(LOCALL,NUMARG,IXVECT)   FAP, SECONDARY ENTRY OF LOCATE
FUNCTION WHICH IS IDENTICAL TO ARGF BUT GIVES FIXED POINT LABEL
TO RESULT.

XAVRGE (IX,LIX,IXAVG)   FAP,   34 REGISTERS
OTHER ENTRY - XAVRGR. TRANSFER VECTOR - XDIV,XDIVR.
SETS IXAVG = (1/LIX) * SUM (FROM I= 1 TO LIX) OF (IX(I), AS TRUNCATED
FORTRAN II INTEGER. OVERFLOW WILL NOT OCCUR. STRAIGHT RETURN IF
LIX LSTHN 1.

XAVRGR (IX,LIX,IXAVG)   FAP, SECONDARY ENTRY OF XAVRGE
SAME AS XAVRGE EXCEPT OUTPUT IS ROUNDED NOT TRUNCATED.

XBOOST (IX,LIX,IXRIZE,IXBSTD)   FAP, SECONDARY ENTRY OF BOOST
SETS IXBSTD(1...LIX) = IX(1...LIX)+IXRIZE. EQUIV(IX,IXBSTD) OK, AND
EQUIV(IXRIZE, SOME IX(I)) OK, BUT INITIAL VALUE OF IXRIZE IS ALWAYS
THE ADDEND. STRAIGHT RETURN IF LIX LSTHN 1.

XCMPRAF(X1,X2)   FAP, SECONDARY ENTRY OF CMPRA
HAS VALUE = 0 IF X1 AND X2 ARE IDENTICAL INCLUDING SIGN BIT,
VALUE = 1 IF X1 IS ALGEBRAICALLY GRTHN X2, VALUE = -1 IF X1
IS ALGEBRAICALLY LSTHN X2 WHERE +0 GRTHN -0 AND MODES OF X1
AND X2 IMMATERIAL.

```
XDANL  (XX,N,M,DXX)                                FAP, SECONDARY ENTRY OF ADANL
      SETS  DXX(-N+1...N+1) = DX(-N...N) WHERE  DX(L) = X(L)*((M/L*PI)*
      SIN(L*PI/M)), GIVEN  XX(-N+1...N+1) = X(-N...N) WITH  N  GRTHN = 0
      AND  M  GRTHN  0 .  EQUIV (DXX,XX) OK.

XDANX  (IXX,N,M,IDXX)                              FAP, SECONDARY ENTRY OF ADANL
      SAME FUNCTION AS SUBROUTINE XDANL EXCEPT THAT INPUT IXX AND OUTPUT
      IDXX ARE FXD.PT.  EQUIV (IDXX,IXX) OK.

XDELTAF(ARG)                                       FAP, SECONDARY ENTRY OF DELTA
      HAS VALUE  =  1  IF  ARG  (ANY MODE)  =  ZERO.  OTHERWISE HAS VALUE  =
      0  .

XDFPRS (IX,LIX,IXPRSD)                             FAP, SECONDARY ENTRY OF DIFPRS
      SETS  IXPRSD(1)=X(1),  IXPRSD(I)=IX(I)-IX(I-1)  FOR  I=2...LIX.
      EQUIV(IXPRSD,IX) OK.  STRAIGHT RETURN IF  LIX  LSTHN 1.

XDIV  F(NUMERA,IDENOM)                                       FAP,  27 REGISTERS
           OTHER ENTRY - XDIVR.  NO TRANSFER VECTOR.
      FUNCTION WHOSE VALUE IS  NUMERA/IDENOM  TRUNCATED TO FORTRAN II INTEGER.
      STRAIGHT RETURN IF IDENOM = ZERO.

XDIVK  (IC,IX1,IX2,...,IXN)                         FAP, SECONDARY ENTRY OF ADDK
      SETS  IX1=IX1/IC, IX2=IX2/IC, ..., IXN=IXN/IC,  AS TRUNCATED FORTRAN II
      INTEGERS.  EQUIV(ANY ARGUMENTS) OK, BUT INITIAL VALUE OF  IC  IS ALWAYS
      THE DIVISOR.  STRAIGHT RETURN IF IC=0, OR N=0.

XDIVKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)  FAP, SECONDARY ENTRY OF ADDK
      SETS  IY1=IX1/IC1,  IY2=IX2/IC2,  ...,  IYN=IXN/ICN,  AS TRUNCATED
      FORTRAN-II INTEGERS.  EQUIV(ANY TWO ARGUMENTS) OK BUT MAY CHANGE INPUTS
      ICJ  OR  IXJ.  PROCESSING IS LEFT TO RIGHT.  IYJ  IS NOT COMPUTED IF
      ICJ=0 AT COMPUTATION TIME.  STRAIGHT RETURN IF N=0.

XDIVR F(NUMERA,IDENOM)                              FAP, SECONDARY ENTRY OF XDIV
      SAME AS XDIV FUNCTION EXCEPT OUTPUT IS ROUNDED, NOT TRUNCATED.

XDPRSS (IX,LIX,IXSINK,IXLWRD)                      FAP, SECONDARY ENTRY OF BOOST
      SETS  IXLWRD(1...LIX) = IX(1...LIX)-IXSINK.  EQUIV(IX,IXLWRD) OK, AND
      EQUIV(IXSINK, SOME IX(I)) OK, BUT INITIAL VALUE OF  IXSINK  IS ALWAYS
      THE SUBTRAHEND.  STRAIGHT RETURN IF  LIX LSTHN 1.

XDVIDE (IX,LIX,IXDVSR,IXDVDD)                                FAP,  33 REGISTERS
           OTHER ENTRY - XDVIDR.  TRANSFER VECTOR - XDIV,XDIVR.
      SETS IXDVDD(1...LIX) = IX(1...LIX)/IXDVSR  AS TRUNCATED FORTRAN II
      INTEGERS.  EQUIV(IX,IXDVDD) OK, AND EQUIV(IXDVSR, SOME IX(I)) OK, BUT
      INITIAL VALUE OF  IXDVSR  IS ALWAYS THE DIVISOR.  STRAIGHT RETURN IF
      IXDVSR=0, OR LIX LSTHN 1.

XDVIDR (IX,LIX,IXDVSR,IXDVDD)                      FAP, SECONDARY ENTRY OF XDVIDE
      SAME AS  XDVIDE  BUT OUTPUT ROUNDED, NOT TRUNCATED.

XDVRK  (IC,IX1,IX2,...,IXN)                         FAP, SECONDARY ENTRY OF ADDK
      SAME AS  XDIVK  EXCEPT OUTPUT ROUNDED, NOT TRUNCATED.

XDVRKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)  FAP, SECONDARY ENTRY OF ADDK
      SAME AS XDIVKS, EXCEPT OUTPUT ROUNDED, NOT TRUNCATED.
```

```
XFIXM F(JOB,FLTG)                                            FAP,  31 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     FUNCTION CONVERTS FLTG TO MACHINE LANGUAGE INTEGER.  IF JOB = 0 FLTG IS
     TRUNCATED TO INTEGER, IF JOB NOT = 0 FLTG IS ROUNDED TO INTEGER.
     MAGNITUDE OF FLTG SHOULD BE LSTHN= 2**27-1., IF BIGGER THE RESULT WILL
     BE CLIPPED TO THIS MAGNITUDE.

XINDEXF(LOCALL,NUMARG)                            FAP, SECONDARY ENTRY OF LOCATE
     FUNCTION PRODUCES INDEX WITH RESPECT TO COMMON OF ARGUMENT NO. NUMARG
     OF THE CALL STATEMENT AT MACHINE ADDRESS LOCALL, WHERE NUMARG GRTHN= 1 .

XLCOMNF(ZIFACT)                                              FAP,   14 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     GIVES THE LENGTH OF  COMMON  SPACE AVAILABLE BEYOND THE LAST STORED
     ROUTINE IF  ZIFACT=0.,  OR THE TOTAL LENGTH OF  COMMON  SPACE
     DIMENSIONED BY THE ROUTINES IF  ZIFACT NOT= 0.

XLIMITF(X,XA,XB)                                             FAP,   25 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     HAS VALUE  =  0  IF  XLO  LSTHN=  X  LSTHN=  XHI  WHERE  XLO =
     MIN(XA,XB)  AND  XHI = MAX(XA,XB),  VALUE  =  +1  IF  X  GRTHN  XHI,
     VALUE  =  -1  IF  X  LSTHN  XLO,  WHERE  +0  IS CONSIDERED  =  -0  IN
     THE COMPARISONS MADE, AND MODE OF ARGUMENTS IMMATERIAL.

XLSHFTF(N,X)                                       FAP, SECONDARY ENTRY OF LSHFT
     PERFORMS SAME FUNCTION AS LSHFT.

XLOCV  (LOCV,X1,X2,...,XN)                                   FAP,   24 REGISTERS
          NO OTHER ENTRIES.  NO TRANSFER VECTOR.
     SETS  LOCV(J)  =  MACHINE ADDRESS (AS FORTRAN-II FIXED POINT INTEGER) OF
     XJ,  FOR  J=1...N  WHERE  N  GRTHN=  1 .

XMLPLY (IX,LIX,IXMPLR,IXMPLD)                     FAP, SECONDARY ENTRY OF MULPLY
     SETS IXMPLD(1...LIX) = IX(1...LIX)*IXMPLR.  EQUIV(IX,IXMPLD) OK, AND
     EQUIV(IXMPLR, SOME IX(I)) OK, BUT INITIAL VALUE OF  IXMPLR  IS ALWAYS
     THE MULTIPLIER.  OVERFLOW DANGER NOT CHECKED.  STRAIGHT RETURN IF
     LIX LSTHN 1.

XMULK  (IC,IX1,IX2,...,IXN)                          FAP, SECONDARY ENTRY OF ADDK
     SETS  IX1=IX1*IC, IX2=IX2*IC, ..., IXN=IXN*IC.  EQUIV(ANY ARGUMENTS) OK,
     BUT INITIAL VALUE OF  IC  IS ALWAYS THE MULTIPLIER.  OVERFLOW DANGER NCT
     CHECKED.  STRAIGHT RETURN IF N=0.

XMULKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)  FAP, SECONDARY ENTRY OF ADDK
     SETS  IY1=IX1*IC1,  IY2=IX2*IC2,  ...,  IYN=IXN*ICN.  EQUIV(ANY TWO
     ARGUMENTS) OK BUT MAY CHANGE INPUTS  ICJ  OR  IXJ.  PROCESSING IS LEFT
     TO RIGHT.  OVERFLOW POSSIBLE, NOT TESTED FOR.  STRAIGHT RETURN IF N=0.

XNAME F(HNAME1,HNAME2)                            FAP, SECONDARY ENTRY OF LOCATE
     FUNCTION HAS VALUE = +0 IF HNAME1 AND HNAME2 (BOTH FORMAT(A6))
     ARE THE SAME HOLLERITH DISREGARDING LEADING SPACES, = -1 IF THEY DIFFER.

XNARGSF(LOCALL)                                   FAP, SECONDARY ENTRY OF LOCATE
     FUNCTION HAS VALUE = NO. ARGUMENTS ASSOCIATED WITH THE CALL
     STATEMENT AT MACHINE ADDRESS LOCALL, EXCEPT VALUE = -1 IF LOCALL
     NOT THE ADDRESS OF A CALL STATEMENT (I.E. NOT A  TSX  X,4)
```

```
XNTHA F(N,IA1,IA2,...,IAN,...)                        FAP, SECONDARY ENTRY OF NTHA
      HAS VALUE  =  IAN  WHERE  IAN  =  N-TH  ARGUMENT FOLLOWING  N,  EXCEPT
      VALUE  =  N  IF  N  LSTHN=  0  AND VALUE IS UNPREDICTABLE IF  N+1
      EXCEEDS ARGUMENT COUNT.

XNTSUM (IX,LIX,IXISMD)                                FAP, SECONDARY ENTRY OF INTSUM
      SETS  IXISMD(I)  =  SUM (FROM J = 1 TO I) OF  IX(I),  I=1...LIX.
      EQUIV(IXISMD,IX) OK.  POSSIBLE OVERFLOW NOT CHECKED FOR.  STRAIGHT
      RETURN IF  LIX  LSTHN 1.

XOOZE F(INT)                                                     FAP,    4 REGISTERS
           NO OTHER ENTRIES.  NO TRANSFER VECTOR.
      HAS VALUE  =  +1  IF  INT  IS AN ODD FORTRAN-II INTEGER,  VALUE  =  0
      IF  INT  IS EVEN, WHERE SIGN OF  INT  IS IMMATERIAL.

XREMAV (IX,LIX,IXAVG,IXNULD)                                     FAP,  31 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - XAVRGR.
      SETS IXAVG = (1/LIX)*(SUM(FROM I=1 TO LIX) OF IX(I)) ROUNDED
      TO FORTRAN-II INTEGER, AND SETS IXNULD(I)=IX(I)-IXAVG  I=1...LIX.
      EQUIV(IX,IXNULD) OK.  NO DANGER OF OVERFLOW IN COMPUTING IXAVG.
      STRAIGHT RETURN IF LIX LSTHN 1.

XRFLEC (IX,LIX,IXMIRR,IXIMGE)                         FAP, SECONDARY ENTRY OF REFLEC
      SETS  IXIMGE(1...LIX)=IXMIRR-IX(1...LIX).  EQUIV(IXIMGE,IX) AND (IXMIRR,
      ANY IX(I)) OK, BUT INPUT  IXMIRR  VALUE ALWAYS USED IN SUBTRACTION.
      STRAIGHT RETURN IF  LIX  LSTHN 1.

XSAME F(X)                                              FAP, SECONDARY ENTRY OF SAME
      FUNCTION DOES NOTHING BUT SUPPLY FIXED POINT LABEL FOR ITS ARGUMENT
      WHICH IS ANY MODE.

XSMDEV (IX,IXBASE,LIX,ISMXMB)                         FAP, SECONDARY ENTRY OF SUMDFR
      SETS  ISMXMB = SUM (FROM I = 1 TO LIX) OF (IX(I)-IXBASE).  POSSIBLE
      OVERFLOW NOT CHECKED FOR.  STRAIGHT RETURN IF  LIX  LSTHN 1.

XSMDFR (IX,IY,LXY,ISMXMY)                             FAP, SECONDARY ENTRY OF SUMDFR
      SETS  ISMXMY = SUM (FROM I = 1 TO LIX) OF (IX(I)-IY(I)).   POSSIBLE
      OVERFLOW NOT CHECKED FOR.  STRAIGHT RETURN IF  LIX  LSTHN 1.

XSPECT (XCOR,N,COSTAB,SINTAB,M,JMIN,JMAX,                   FORTRAN, 523 REGISTERS
        CSP,SSP,SPACE,ERR)
           NO OTHER ENTRIES.  TRANSFER VECTOR - SPLIT,COSISP,REFIT,XLOC,
           KOLAPS,CHPRTS.
      SETS  CSP(1...JMAX-JMIN+1) = CS(JMIN...JMAX) AND SETS  SSP(1...JMAX-
      JMIN+1) = SS(JMIN...JMAX), WHERE CS(J) = SUM (FROM I = -N TO N) OF
      (XC(I)*COS(I*J*PI/M)) AND  SS(J) = SAME SUM WITH SIN REPLACING COS,
      GIVEN XCOR(-N+1...N+1) = XC(-N...N), GIVEN 0 LSTHN= JMIN LSTHN JMAX
      LSTHN= M,  AND GIVEN COSTAB(1...M+1) = COS(I*PI/M) FOR I=0...M
      AND  SINTAB(1...M+1) = SIN(I*PI/M) FOR I=0...M .  SPACE IS
      SCRATCH AREA.  IF  M  GRTHN  N  NO SCRATCH AREA IS NEEDED AND SPACE
      NOT USED.  IF  M  LSTHN= N, 2*M+4 REGISTERS REQUIRED WHICH WILL BE
      TAKEN AS SPACE(1...2*M+4) IF USER HAS NOT MADE PERMISSABLE  EQUIV
      (SPACE,XCOR).  IF EQUIV(SPACE,XCOR) HAS BEEN MADE THE  2*M+4 SCRATCH ARE
      TAKEN FROM  XCOR(-M+1...M+4) WHICH WILL REQUIRE  3 REGISTERS
      BEYOND  XCOR(N+1) IN THE CASE M = N . SETS ERR = 0.0 IF ALL OK,
      = 1.0 IF N (MUST EXCEED 0), M, JMIN OR JMAX ILLEGAL.
```

```
XSQDEV (IX,IXBASE,LIX,ISSXMB)                     FAP, SECONDARY ENTRY OF XSQDFR
      SETS  ISSXMB = SUM (FROM I = 1 TO LIX) OF (IX(I)-IXBASE)SQUARED.
      POSSIBLE OVERFLOW NOT CHECKED.  STRAIGHT RETURN IF LIX LSTHN 1 .

XSQDFR (IX,IY,LXY,ISSXMY)                                   FAP,  37 REGISTERS
           OTHER ENTRY - XSQDEV.  NO TRANSFER VECTOR.
      SETS  ISSXMY = SUM (FROM I = 1 TO LIX) OF (IX(I)-IY(I)SQUARED.
      POSSIBLE OVERFLOW NOT CHECKED FOR.  STRAIGHT RETURN IF  LIX  LSTHN 1.

XSQRUT (IX,LIX,IXSQRT)                                      FAP,  37 REGISTERS
           NO OTHER ENTRIES.  TRANSFER VECTOR - FIXVR,SQRT.
      SETS  IXSQRT(1...LIX) = SQUARE ROOT (MAGNITUDE(IX(1...LIX))), ROUNDED TO
      NEAREST FORTRAN II INTEGER.  EQUIV(IXSQRT,IX) OK.  STRAIGHT RETURN IF
      LIX  LSTHN 1.

XSQSUM (IX,LIX,ISMSQX)                            FAP, SECONDARY ENTRY OF SQRSUM
      SETS  ISMSQX = SUM (FROM I= 1 TO LIX) OF IX(I)*IX(I).  OVERFLOW DANGER
      NOT CHECKED.  STRAIGHT RETURN IF LIX LSTHN 1.

XSQUAR (IX,LIX,IXSQRD)                            FAP, SECONDARY ENTRY OF SQUARE
      SETS  IXSQRD(1...LIX) = IX(1...LIX) SQUARED.  EQUIV(IX,IXSQRD) OK.
      OVERFLOW DANGER NOT CHECKED.  STRAIGHT RETURN IF LIX LSTHN 1.

XSTEPCF(ARG)                                      FAP, SECONDARY ENTRY OF DELTA
      HAS VALUE  1  OR  0  ACCORDING AS SIGN BIT OF  ARG  IS PLUS OR MINUS.

XSTEPLF(ARG)                                      FAP, SECONDARY ENTRY OF DELTA
      HAS VALUE  =  1  IF  ARG  (ANY MODE) IS  GRTHN=  MINUS ZERO.  OTHERWISE
      HAS VALUE  =  0  .

XSTEPRF(ARG)                                      FAP, SECONDARY ENTRY OF DELTA
      HAS VALUE  =  1  IF  ARG  (ANY MODE) EXCEEDS ZERO.  OTHERWISE HAS
      VALUE  0  .

XSTLIN (IBASE,IDELTA,LIX,IX)                      FAP, SECONDARY ENTRY OF SETLIN
      SETS  IX(I)=IBASE+(I-1)*IDELTA,  I=1...LIX.  EQUIV(IBASE,IDELTA, ANY
      IX(I)) OK, INPUT VALUES OF  IBASE  AND  IDELTA  ALWAYS USED.  STRAIGHT
      RETURN IF LIX LSTHN 1 .

XSUBK  (IC,IX1,IX2,...,IXN)                         FAP, SECONDARY ENTRY OF ADDK
      SETS  IX1=IX1-IC, IX2=IX2-IC, ..., IXN=IXN-IC.  EQUIV(ANY ARGUMENTS) OK,
      BUT INITIAL VALUE OF  IC  IS ALWAYS THE SUBTRAHEND.  STRAIGHT RETURN IF
      N=0.

XSUBKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)  FAP, SECONDARY ENTRY OF ADDK
      SETS  IY1=IX1-IC1,  IY2=IX2-IC2,  ...,  IYN=IXN-ICN.  EQUIV(ANY TWO
      ARGUMENTS) OK BUT MAY CHANGE INPUTS  ICJ  OR  IXJ.  PROCESSING IS LEFT
      TO RIGHT.  STRAIGHT RETURN IF  N=0.

XSUM   (IX,LIX,ISUMIX)                            FAP, SECONDARY ENTRY OF SUM
      SETS ISUMIX = SUM (FROM I = 1 TO LIX) OF IX(I).  OVERFLOW DANGER NOT
      CHECKED.  STRAIGHT RETURN IF LIX LSTHN 1.

XVDRBV (IX,IY,LXY,IXDVBY)                         FAP, SECONDARY ENTRY OF XVDVBV
      IDENTICAL TO  XVDVBV  EXCEPT RESULTS ROUNDED, NOT TRUNCATED.
```

```
XVDVBV (IX,IY,LXY,IXDVBY)                                         FAP,  34 REGISTERS
          OTHER ENTRY - XVDRBV.  TRANSFER VECTOR - XDIV,XDIVR.
      SETS  IXDVBY(1...LXY)  FROM  IX(1...LXY)  AND  IY(1...LXY),  WHERE
      IXDVBY(I) = IX(I)/IY(I),  TRUNCATED TO FORTRAN-II INTEGERS.
      EQUIV(IXDVBY, IX OR IY) OK.  STRAIGHT RETURN IF LXY LSTHN 1 .

XVMNSV (IX,IY,LXY,IXMNSY)                         FAP, SECONDARY ENTRY OF VPLUSV
      SETS  IXMNSY(1...LXY)  FROM  IX(1...LXY)  AND  IY(1...LXY),  WHERE
      IXMNSY(I)=IX(I)-IY(I).  EQUIV(IXMNSY, IX OR IY) OK.  STRAIGHT RETURN
      IF LXY LSTHN 1 .

XVPLSV (IX,IY,LXY,IXPLSY)                         FAP, SECONDARY ENTRY OF VPLUSV
      SETS  IXPLSY(1...LXY)  FROM  IX(1...LXY)  AND  IY(1...LXY), WHERE
      IXPLSY(I)=IX(I)+IY(I).  EQUIV(IXPLSY, IX OR IY) OK.  STRAIGHT RETURN
      IF LXY LSTHN 1 .

XVTMSV (IX,IY,LXY,IXTMSY)                         FAP, SECONDARY ENTRY OF VTIMSV
      SETS  IXTMSY(1...LXY)  FROM  IX(1...LXY)  AND  IY(1...LXY),  WHERE
      IXTMSY(I) = IX(I)*IY(I).  EQUIV(IXTMSY, IX OR IY) OK.  NO OVERFLOW
      CHECK MADE.  STRAIGHT RETURN IF LXY LSTHN 1 .

XWHICHF(IX1,IX2,ZIFIX1)                           FAP, SECONDARY ENTRY OF WHICH
      HAS VALUE  =  IX1  IF  ZIFX1=0.0,  VALUE  =  IX2  IF  ZIFIX1 NOT= 0.0  .

ZEFBCDF(ITAPE)                                              FAP,   54 REGISTERS
          OTHER ENTRY - ZEFBIN.  TRANSFER VECTOR - (IOS),(RDS),(RCH),(TCO),
          (TEF),(TRC),(BSR).
      FUNCTION HAS VALUE  =  0.0  IF NEXT RECORD ON LOGICAL TAPE NUMBER  ITAPE
      IS AN  END-OF-FILE  RECORD  (BCD  MODE),  =  1.0  IF NOT  END-OF-FILE,
      =  -1.0  IF REDUNDANCY  (10  READ ATTEMPTS MADE).  TAPE IS LEFT UNMOVED.

ZEFBINF(ITAPE)                                    FAP, SECONDARY ENTRY OF ZEFBCD
      FUNCTION HAS VALUE  =  0.0  IF NEXT RECORD ON LOGICAL TAPE NUMBER  ITAPE
      IS AN  END-OF-FILE  RECORD (BINARY MODE),  =  1.0  IF NOT  END-OF-FILE,
      =  -1.0  IF REDUNDANCY  (10  READ ATTEMPTS MADE).  TAPE IS LEFT UNMOVED.
```

6

Program Statistics

The program statistics tabulation below provides an alphabetically ordered listing of all entries, with their secondary entries, transfer vectors, storage requirements, acceptance dates* of symbolic decks, symbolic deck-card counts, binary deck-card counts, authors, programming language, and entry-name pronounciations. All numbers given in the tables are decimal. The symbol M is used for machine language (i.e., FAP) and F for FORTRAN. Authors are coded by initials as follows.

AMN	Arcadio M. Niell
CP	Cheh Pan
EAR	Enders A. Robinson
IH	Ira Hanson
JC	Jacqueline Clark
JFC	Jon F. Claerbout
JNG	James N. Galbraith, Jr.
JTO	J.T. Olsztyn
JTP	Joseph T. Procito, Jr.
MIT	MIT Lincoln Laboratory or Computation Center Staff
RAW	Ralph A. Wiggins
RJG	Roy J. Greenfield
SMS	Stephen M. Simpson, Jr.

The pronounciations given approximate the conversational usage of our programming group. A letter followed by a period indicates a syllable pronounced as in alphabetic recitation of the letter. An unsyllabized word always receives ordinary pronounciation. A stress mark following such a word indicates that the whole word, rather than its last syllable, is accented.

---

*See the discussion at the beginning of Section 10 for the meaning of this term.

PROGRAM STATISTICS

****
* ABSVAL TO ARBCOL *
****

ENTRY NAME	SECONDARY ENTRIES	TRANSFER VECTOR	STORAGE	DATE OF SYMBOLIC DECK	SYMBOLIC CARD COUNT	BINARY CARD COUNT	AUTHOR	LANGUAGE	ENTRY NAME PRONUNCIATION
ABSVAL			50	9/29/64	117	4	SMS	M	ABZ' VAL
ADANL			183	9/29/64	336	11	JFC	M	A.' DAN L.'
	XDANL	SIN							
	ADANX								
	XDANX								
ADANX (SEE ADANL)									A.' DAN X.'
ADDK			114	9/29/64	366	8	SMS	M	ADD' K.
	SUBK								
	MULK								
	DIVK								
	XADDK								
	XSUBK								
	XMULK								
	XDIVK								
	XDVRK								
	ADDKS								
	SUBKS								
	MULKS								
	DIVKS								
	XADDKS								
	XSUBKS								
	XMULKS								
	XDIVKS								
	XDVRKS								
ADDKS (SEE ADDK)									ADD KAYZ'
AMPHZ			149	10/ 1/64	251	10	JFC	M	AMP' PHASE
	REIM	ATAN							
		SQRT							
		RND							
		COS							
		SIN							
ARBCOL			129	9/ 9/64	271	8	SMS	M	ARB' KAHL
		INTOPR							

* ARCTAN TO CMPARP *

ARCTAN		29	9/ 4/64	92	3	RAW	M	ARC' TAN
	ATAN							
ARG	(SEE LOCATE)							ARG
ASPECT		278	9/29/64	536	15	SMS	M	ASPECT
	COLAPS							
	COSP							
	DUBLX							
	DUBLL							
	SPLIT							
	RVPRTS							
ASPEC2		74	3/15/65	206	5	SMS	M	AS' PEK 2'
	SEQSAC							
	NEXCOS							
AVRAGE		24	9/29/64	79	3	SMS	M	AV' REDGE
BLKSUM		49	9/ 4/64	169	4	SMS	M	BLOCK' SUM
BOOST		34	9/29/64	147	3	SMS	M	BOOST
XBOOST								
DPRESS								
XDPRSS								
CALL	(SEE LOCATE)							CALL
CALL2	(SEE LOCATE)							CALL' 2'
CARIGE		47	9/29/64	98	4	SMS	F	CARRIAGE
	(STH)							
	(FIL)							
CHISQR		105	9/29/64	85	6	JNG	F	KAI' SKER
CHOOSE		17	9/ 4/64	84	2	SMS	M	CHOOSE
CHPRTS		76	9/29/64	149	5	SMS	M	CHEH PARTS'
RVPRTS								
CHSIGN		18	9/29/64	78	2	SMS	M	CHEH SINE'
CHUSET	(SEE INDEX)							CHU' SET
CLKON		46	9/29/64	42	4	RAW	F	CLOCK' ON
	CLOCK1							
	(SPH)							
	(FIL)							
CLOCK1 (7090)		57	3/15/65	148	4	SMS	M	CLOCK' 1'
CMPARL	(SEE CMPARV)							KUM PAR' L.'
CMPARP		53	9/29/64	151	4	SMS	M	KUM PAR' P.'
CMPARS								

CMPARS (SEE CMPARP)								KUM PAR' S.'
CMPARV		50	9/ 4/64	156	4	SMS	M	KUM PAR' V.'
CMPARL								
CMPRA		18	9/ 4/64	104	2	RAW	M	KUM' PRUH
XCMPRA								
CMPRFL								
CMPRFL (SEE CMPRA)								KUM' PRUH FUL
CNTRDB		550	9/ 9/64	251	27	SMS	F	CONTOUR' D. B.
	SETVEC							
	LOG							
	CONTUR							
	EXP							
	SAME							
	(STH)							
	(FIL)							
CNTROW		802	9/ 9/64	521	39	SMS	F	CONTOUR' ROW
	RNDDN							
	RNDUP							
	QUFIT1							
	CUFIT1							
	FASCUB							
	RND							
COLABL		185	9/ 4/64	124	10	SMS	F	KAH' LAH BUL
	GENHOL							
	(SPH)							
	(FIL)							
	(STH)							
COLAPS		50	9/29/64	128	4	JC	M	COLLAPSE
CONTUR		587	9/ 9/64	642	29	SMS	F	CONTOUR
	RNDDN							
	RNDUP							
	(STH)							
	(FIL)							
	COLABL							
	ARBCOL							
	CNTROW							
	SWITCH							
	(SPH)							
	XSAME							
CONVLV		96	9/29/64	99	6	JFC	F	CONVOLVE
CONVLV-II		56	10/ 2/64	149	4	JFC+ RAW	M	CONVOLVE' DASH 2'
COSISP (SEE COSP)								KOH' SISP

****

* COSIS1 TO CVSOUT *
****

COSIS1		406	9/10/64	264	21	RAW	F	KOH' SIS 1'
	IXCARG							
	SPLIT							
	MOVREV							
	CHPRTS							
	COSP							
	SISP							
	COSISP							
COSP		504	9/29/64	878	27	SMS	M	KAHSP
SISP								
COSISP								
COSTBL		121	9/29/64	200	8	JFC	M	KAHSS' TUH BUL
SINTBL	COS							
COSTBX	SIN							
SINTBX								
COSTBX (SEE COSTBL)								KAHSS' TUB X.'
CPYFL2		178	9/ 9/64	304	10	RAW	M	COPY' FILE 2'
	(IOS)							
	(TCO)							
	(WRS)							
	(RCH)							
	(TRC)							
	(ETT)							
	(WEF)							
	(BSR)							
	(RDS)							
CROSS		107	9/29/64	87	7	RAW	F	CROSS
	STZ							
	FDOT							
CROST		134	9/29/64	99	8	RAW	F	CROSS' T.'
	CROSS							
	REVERS							
CRSVM		327	9/10/64	220	17	RAW	F	C.' R. S. V. M.'
	SETKS							
	MDOT3							
	STZ							
CSOUT		49	9/ 4/64	127	4	RAW	M	SEEZ' OUT
	CARIGE							
	(STH)							
	HRADJ							
	(FIL)							
CUFIT1		158	9/ 4/64	326	9	SMS	M	Q.' FIT 1'
CVSOUT		84	9/29/64	221	6	SMS	M	C.' VEEZ' OUT
	CARIGE							

	FMTOUT							
	VECOUT							
DADECK		100	9/ 4/64	70	6	JNG+	F	DAY' DECK
	EOFSET					RAW		
	(TSH)							
	(RTN)							
	(STH)							
	(FIL)							
	RSKIP							
DELTA		17	9/ 4/64	141	2	SMS	M	DELTA
XDELTA								
STEPR								
XSTEPR								
STEPL								
XSTEPL								
STEPC								
XSTEPC								
DERIVA		61	9/29/64	160	5	SMS	M	DEH RIV' UH
DETRM (SEE SIMEQ)								D. TERM'
DIFPRS		30	9/29/64	118	3	SMS	M	DIF' PERZ
XDFPRS								
DISPLA (709)		220	9/29/64	474	12	MIT	M	DISPLAY
	(IOH)							
DISPLA (7090)		219	9/ 4/64	481	13	MIT	M	DISPLAY
	(IOH)							
	FRAME							
DIVIDE		23	9/29/64	88	3	SMS	M	DIVIDE
DIVK (SEE ADDK)								DIV' K.
DIVKS (SEE ADDK)								DIV KAYZ'
''DO'' (SEE SEVRAL)								DO
DOTJ		59	10/ 2/64	143	4	RAW	M	DOT' J.'
DOTP		264	9/29/64	147	14	RAW	F	DOT' P.'
	DOTJ							
DPRESS (SEE BOOST)								DEPRESS
DSPFMT		194	9/29/64	313	11	SMS	M	DISP' FUM ET
DUBLL (SEE DUBLX)								DOUBLE' L.'
DUBLX		45	9/29/64	129	4	SMS	M	DOUBLE' X.'
DUBLL								

* DUBLX TO FLOATM *

HALVX								
HALVL								
ENDFIL (SEE REREAD)								END' FILE
EOFSET (SEE REREAD)								E.' O. F.' SET
EXCHVS		22	9/29/64	84	3	SMS	M	EX' CHEH VEEZ
EXPAND		189	9/ 4/64	380	11	SMS	M	EXPAND
	INTOPR							
FACTOR		308	9/ 8/64	489	17	JNG	M	FACTOR
	MAXAB							
	LOG							
	COSTBL							
	COSP							
	EXP							
FAPSUM		14	9/29/64	66	2	JFC	M	FAP' SUM
FASCN1		107	9/29/64	199	7	SMS	M	FASS' SCAN 1'
FASCOR (SEE PROCOR)								FASS' CORE
FASCR1 (SEE PROCOR)								FASS' KER 1'
FASCUB		141	9/ 4/64	260	9	SMS	M	FASS' CUBE
FASEPC (SEE PROCOR)								FASS' E. P. C.'
FASEP1 (SEE PROCOR)								FASS' E. P. 1'
FASTRK		26	9/ 8/64	119	3	SMS	M	FASS' TRACK
FDOT		40	9/ 4/64	101	3	RAW	M	F.' DOT
FDOTR								
FDOTR (SEE FDOT)								F.' DOT R.'
FIRE2		271	9/ 8/64	152	14	RAW	F	FIRE 2'
	IXCARG							
	STZ							
	DOTP							
	MATML3							
	DOTJ							
FIXV		35	9/29/64	105	3	SMS	M	FIX' V.'
FIXVR								
FIXVR (SEE FIXV)								FIX' V. R.'
FLDATA (SEE FXDATA)								FLOW' DATA
FLOATM		25	9/29/64	91	3	SMS	M	FLOAT' M.'

FLOATV		22	9/29/64	81	3	SMS	M	FLOAT' V.'
FMTOUT		51	9/29/64	71	4	SMS	F	FUMT' OUT
	FNDFMT							
	RPLFMT							
	(STH)							
	(FIL)							
FNDFMT		88	9/29/64	203	6	SMS	M	FIND' FUMT
	REVER							
FRAME (709)		4	9/29/64	34	2	RAW	M	FRAME
FRAME (7090)		9	9/ 4/64	47	2	MIT	M	FRAME
FRQCT1		117	9/29/64	95	7	SMS	F	FREE' COUNT 1'
FRQCT2		117	9/29/64	212	7	JNG	M	FREE' COUNT 2'
FSKIP		50	9/ 4/64	104	4	JFC	M	F.' SKIP
	(IOS)							
	(RDS)							
	(BSR)							
	(TCO)							
	(TEF)							
	(TRC)							
FT24		777	9/29/64	848	40	CP	M	F.'T. 24'
	FXDATA							
	FLDATA							
FT24 -II		818	9/29/64	147	39	RAW	F	F.'T. 24' DASH 2'
FXDATA		102	10/ 1/64	248	7	SMS	M	FIX' DATA
FLDATA								
GENHOL		48	3/15/65	145	4	RAW	M	JEN' HAHL
	(IOH)							
GETHOL		169	9/29/64	176	9	SMS	F	GET' HAHL
	XLOC							
	REVERS							
GETRD1		229	10/ 1/64	173	10	SMS	F	GET' R. D. 1'
	(TSH)							
	(RTN)							
GETX		31	9/ 4/64	128	3	RAW	M	GET' X.
IGETX								
GNFLT1		232	9/29/64	164	12	SMS	F	JEN' FILT 1'
	COS							
GNHOL2		74	9/29/64	158	5	RAW	M	JEN' HAHL 2'
	(IOH)							

	(FIL)							
GRAPH		1499	9/29/64	1103	72	SMS	F	GRAPH
	DISPLA							
	(SPH)							
	(FIL)							
	LINE							
	LOG							
	EXP(2							
	XFIXM							
	FLOATM							
	DSPFMT							
	FRAME							
	XLOC							
	MVBLOK							
	SCPSCL							
	HSTPLT							
GRAPHX		123	9/29/64	154	7	SMS	F	GRAPH' X.'
	GRAPH							
	FRAME							
GRUP2		201	10/ 1/64	141	11	JNG	F	GROUP' 2'
HALVL (SEE DUBLX)								HALVE' L.'
HALVX (SEE DUBLX)								HALVE' X.'
HLADJ		46	9/29/64	111	4	SMS	M	H. L.' ADJUS
HRADJ								
HRADJ (SEE HLADJ)								H. R.' ADJUS
HSTPLT		145	9/29/64	346	9	JNG	M	HIST'PLOT
	LINEH							
	LINEV							
HSTPLT-II		188	9/29/64	336	11	RAW	M	HIST'PLOT DASH 2'
	LINEH							
	LINEV							
HSTPLT-III (709)		256	9/29/64	438	14	RAW	M	HIST'PLOT DASH 3'
	LINEH							
HSTPLT-III (7090)		258	9/ 8/64	446	14	RAW	M	HIST'PLOT DASH 3'
	LINEH							
HVTOIV		39	9/29/64	110	3	SMS	M	H.' V. TO I.' V.
IDERIV		54	9/29/64	149	4	SMS	M	I.' DEH RIV
''IF'' (SEE SEVRAL)								IF
IFNCTN		208	9/ 4/64	444	12	SMS	M	I.' FUNCTION
	MONOCK							
	REVER							

IGETX (SEE GETX)								I.' GET X.
IINTGR		49	9/29/64	157	4	SMS	M	I.' INT' GR
INDATA		896	10/ 1/64	489	32	JFC	F	IN' DATA
	VARARG							
	FSKIP							
	(TSB)							
	(RLR)							
	FAPSUM							
	LOC							
	MVBLOK							
	XSAME							
	(SPH)							
	(FIL)							
	(STH)							
	UNPAKN							
INDEX		50	9/ 4/64	270	4	SMS	M	INDEX
VINDEX								
SETEST								
SETAPT								
CHUSET								
INTGRA		47	9/29/64	175	4	SMS	M	INT' GRAH
INTHOL		72	9/ 9/64	156	5	RAW	M	INT' HAHL
	FNDFMT							
	(IOH)							
	(RTN)							
INTMSB (SEE TIMSUB).								IN' TUM SUB
INTOPR		111	9/ 4/64	251	7	SMS	M	INT' AH PER
INTSUM		27	9/29/64	110	3	SMS	M	INT' SUM
XNTSUM								
IPLYEV		98	10/ 2/64	84	6	RAW	F	I.' POLLY' EV
	(IFMP)							
ITOMLI		37	9/29/64	98	3	SMS	M	IT' UM LEE
IVTOHV		70	3/15/65	148	5	SMS	M	I.' V. TO H.' V.
IXCARG		35	9/29/64	67	3	SMS	F	IX' KARG
	XLOC							
KIINT1		191	9/29/64	129	10	SMS	F	KAI' INT 1'
	SQRT							
	EXP(3							
	NOINT1							

KOLAPS		100	9/29/64	219	6	JC	M	KOH' LAPSE
LIMITS		44	9/ 8/64	162	4	SMS	M	LIMITS
LINE (709)		91	9/29/64	193	6	SMS	M	LINE
LINE (7090)		95	9/ 4/64	208	6	SMS	M	LINE
LINEH (709)		34	9/29/64	158	3	JNG	M	LINE' H.'
LINEH (7090)		35	9/ 4/64	168	3	JNG	M	LINE' H.'
LINEV (709)		34	9/29/64	161	3	JNG	M	LINE' V.'
LINEV (7090)		35	9/ 4/64	169	3	JNG	M	LINE' V.'
LINTR1		96	9/29/64	93	6	SMS	F	LIN' TER 1'
LISTNG		755	9/29/64	221	38	RAW	F	LISTING
	(RWT)							
	(STH)							
	(FIL)							
	(TSB)							
	(RLR)							
	FAPSUM							
	SAME							
	XSAME							
	(SPH)							
	FSKIP							
	SHFTR2							
LOC		4	9/29/64	54	2	RAW	M	LOKE
LOCATE		512	3/15/65	2008	28	SMS	M	LOCATE
WHERE								
CALL								
CALL2								
SETSBV								
SETUP								
RETURN								
XINDEX								
ARG								
XARG								
STORE								
XNARGS								
XNAME								
LSHFT		12	9/29/64	72	2	RAW	M	L.' SHIFT
XLSHFT								
LSLINE		117	10/ 1/64	82	7	RAW	F	L. S. LINE'
LSSS1		122	9/29/64	116	7	RAW	F	L.' S. S. S. 1'
	FDOT							

MATINV		90	9/29/64	79	6	RAW	F	MAT' INV
	SIMEQ							
MATML1		61	9/29/64	137	5	RAW	M	MAT' MUL 1'
MATML3		120	9/29/64	105	7	RAW	F	MAT' MUL 3'
	DOTJ							
MATRA		92	9/29/64	177	6	RAW+	M	MAY' TRAH
						SMS		
MATRA1		42	9/29/64	95	4	RAW	M	MAY' TRAH 1'
MAXAB (SEE MAXSN)								MAX' AB
MAXABM (SEE MAXSNM)								MAX' UH BIM
MAXSN		54	9/29/64	170	5	JFC	M	MAX' SIN
MINSN								
MAXAB								
MINAB								
MAXSNM		61	9/ 4/64	247	5	SMS	M	MAX' SNIM
MINSNM								
MAXABM								
MINABM								
MDOT		109	9/29/64	94	7	RAW	F	M.' DOT
	MATML1							
MDOT3		122	9/29/64	120	7	RAW	F	M.' DOT 3'
	MATML3							
MEMUSE		71	9/ 4/64	69	5	SMS	F	MEM' YEWSS
	XLCOMN							
	(STH)							
	(FIL)							
MFACT		187	9/29/64	103	10	RAW	F	M.' FACT
	STZ							
	DOTJ							
	SQRT							
MIFLS		276	9/ 8/64	167	14	RAW	F	MIFFLES
	MOVREV							
	MATML3							
MINAB (SEE MAXSN)								MIN' AB
MINABM (SEE MAXSNM)								MIN' UH BIM
MINSN (SEE MAXSN)								MIN' SIN
MINSNM (SEE MAXSNM)								MIN' SNIM

MIPLS		571	9/29/64	254	28	RAW	F	MIPPLES
	IXCARG							
	MATINV							
	MATML3							
	MATRA							
	MDOT3							
	MOVREV							
	STZ							
MISS		335	10/ 5/64	150	17	RAW	F	MISS
	MOVREV							
	MATML3							
	MDOT3							
MLISCL		47	9/29/64	115	4	SMS	M	MUH LIS' KUHL
MLI2A6		128	9/29/64	218	8	SMS	M	MILLIE' TO A. 6'
MONOCK		48	9/ 4/64	165	4	SMS	M	MAHN' AHK
MOUT		130	9/ 8/64	101	8	RAW	F	MOUT
	CARIGE							
	(STH)							
	(FIL)							
MOUTAI		357	9/ 4/64	295	18	SMS	F	MOUT' TAI
	EXP(2							
	CARIGE							
	GNHOL2							
	MAXABM							
	LOG							
	RND							
	(STH)							
	(FIL)							
	SAME							
	MOVE							
	MULPLY							
	FIXVR							
MOVE		32	9/29/64	92	3	JFC	M	MOVE
MOVECS		24	9/29/64	106	3	SMS	M	MOO' VEX
	MOVE							
MOVREV		74	9/29/64	156	5	RAW	M	MOVE' REV
MPSEQ1		110	9/29/64	197	7	JNG	M	MAP' SEEK 1'
MRVRS		61	9/29/64	67	4	RAW	F	MER' VERZ
	REVERS							
MSCON1		238	9/29/64	108	11	JNG	F	MISS' KAHN 1'
MULK	(SEE ADDK)							MUL' K.

* MULK TO NXALRM *

MULK -II		76	9/29/64	78	5	SMS	F	MUL' K. DASH 2'
	SETUP							
	ARG							
	STORE							
	RETURN							
MULKS (SEE ADDK)								MUL KAYZ'
MULLER		757	9/ 9/64	232	36	IH	F	MULLER
	SQRT							
MULPLY		34	9/29/64	114	3	SMS	M	MUL' PLEE
XMLPLY								
MUVADD		129	9/29/64	245	8	SMS	M	MOVE' ADD
MVBLOK		19	9/29/64	83	2	SMS	M	MOVE' BLOCK
MVINAV		221	9/29/64	116	12	SMS	F	MOVING' AV
MVNSUM		71	9/ 4/64	202	5	SMS	M	MOVING' SUM
MVNTIN		88	9/ 4/64	234	6	SMS	M	MOVING TEEN'
MVNTNA								
MVNTNA (SEE MVNTIN)								MOVING TEEN' UH
MVSQAV		236	9/29/64	116	13	SMS	F	MUHV' SKAV
MXRARE		302	9/29/64	250	16	SMS	F	MAX' RARE
	EXP(2							
NEXCOS (SEE SEQSAC)								NEX' KOHSS
NEXSIN (SEE SEQSAC)								NEX' SINE
NMZMG1		34	9/29/64	97	3	RAW	M	N.'M.Z.'M.G.' 1'
NOINT1		369	9/29/64	375	20	SMS+	M	NOINT' 1'
NOINT2	LINTR1					JNG		
NOINT2 (SEE NOINT1)								NOINT' 2'
NRMVEC		111	9/29/64	100	7	RAW	F	NORM' VEK
	SQRT							
	MAXAB							
NTHA		11	10/ 6/64	93	2	SMS	M	ENTH' UH
XNTHA								
NURINC		121	9/ 4/64	327	8	SMS	M	NEW' RINK
NXALRM		243	9/29/64	178	13	SMS	F	NEX' ALARM
	FASCN1							

```
************************      PROGRAM STATISTICS      ************************
*  ONLINE  TO  PLOTVS  *                              *  ONLINE  TO  PLOTVS  *
************************                              ************************


ONLINE    .        .  134 .  4/14/65 .  191 .    8 . RAW . M . ON' LINE
   (STH)  . (IOH)  .      .          .      .      .     .   .
   (STHM) . (WER)  .      .          .      .      .     .   .
   (STHD) . (TES)  .      .          .      .      .     .   .
          . (WRS)  .      .          .      .      .     .   .
          . (WTC)  .      .          .      .      .     .   .
          . (RCH)  .      .          .      .      .     .   .
          . (FIL)  .      .          .      .      .     .   .
          . (SPH)  .      .          .      .      .     .   .
          .        .      .          .      .      .     .   .
OUDATA    .        .  495 .  3/15/65 .  269 .   11 . JFC . F . OW' DATA
          . VARARG .      .          .      .      .     .   .
          . LOC    .      .          .      .      .     .   .
          . MVBLOK .      .          .      .      .     .   .
          . FAPSUM .      .          .      .      .     .   .
          . PAKN   .      .          .      .      .     .   .
          . (STB)  .      .          .      .      .     .   .
          . (WLR)  .      .          .      .      .     .   .
          . (EFT)  .      .          .      .      .     .   .
          .        .      .          .      .      .     .   .
PACDAT    .        .  152 .  9/ 9/64 .  259 .    9 . RAW . M . PACK' DAT
          . (IOS)  .      .          .      .      .     .   .
          . (TCO)  .      .          .      .      .     .   .
          . (RDS)  .      .          .      .      .     .   .
          . (RCH)  .      .          .      .      .     .   .
          . (ETT)  .      .          .      .      .     .   .
          .        .      .          .      .      .     .   .
PAKN      .        .   78 .  9/29/64 .  147 .    5 . JFC . M . PACK' N.'
          . FXDATA .      .          .      .      .     .   .
          .        .      .          .      .      .     .   .
PLANSP    .        . 1169 .  9/ 9/64 .  383 .   56 . RAW . F . PLANSP
          . SETKS  .      .          .      .      .     .   .
          . LIMITS .      .          .      .      .     .   .
          . IXCARG .      .          .      .      .     .   .
          . CHOOSE .      .          .      .      .     .   .
          . XOOZE  .      .          .      .      .     .   .
          . MOVREV .      .          .      .      .     .   .
          . STZ    .      .          .      .      .     .   .
          . ROAR2  .      .          .      .      .     .   .
          . XADDKS .      .          .      .      .     .   .
          . KOLAPS .      .          .      .      .     .   .
          . COSTBL .      .          .      .      .     .   .
          . SINTBL .      .          .      .      .     .   .
          . XADDK  .      .          .      .      .     .   .
          . COSIS1 .      .          .      .      .     .   .
          . MATRA  .      .          .      .      .     .   .
          .        .      .          .      .      .     .   .
PLOTVS    .        .  494 .  9/ 4/64 .  261 .   18 . SMS . F . PLOT' VEEZ'
          . SETVEC .      .          .      .      .     .   .
          . SETKS  .      .          .      .      .     .   .
          . SETKV  .      .          .      .      .     .   .
          . (STH)  .      .          .      .      .     .   .
          . (FIL)  .      .          .      .      .     .   .
          . SWITCH .      .          .      .      .     .   .
          . (SPH)  .      .          .      .      .     .   .
          . RND    .      .          .      .      .     .   .
```

PLTVS1		817	9/ 4/64	393	40	SMS	F	PLOT' VEEZ 1'
	VARARG							
	SETKS							
	SETVEC							
	SETKVS							
	XSTLIN							
	XLOC							
	XSAME							
	RMSDEV							
	(STH)							
	(FIL)							
	MAXSN							
	MINSN							
	MULPLY							
	BOOST							
	PLOTVS							
	DPRESS							
PLURAL (SEE SEVRAL)								PLURAL
PLURNS		73	9/29/64	247	5	SMS	M	PLURNS
PLYSYN		172	10/ 5/64	162	10	EAR	F	PLEE' SIN
	COS							
	CONVLV							
POKCT1		219	9/29/64	134	11	SMS	F	POH' COUNT 1'
	FRQCT1							
POLYDV		130	9/ 9/64	102	7	JFC+	F	POLLY' DIV
	MOVE					RAW		
	STZ							
POLYEV		54	9/29/64	62	4	JFC	F	POLLY' EV
POLYSN		256	9/ 8/64	167	14	RAW	F	POLLY' SIN
	SQRT							
	COS							
	CONVLV							
	MOVE							
POWER		50	9/29/64	130	4	SMS	M	POWER
SMPRDV	EXP(2							
PRBFIT		373	9/29/64	187	16	RJG	F	PRAHB' FIT
	SQRT							
	EXP(2							
	EXP							
PROB2		229	10/ 6/64	175	12	JNG	F	PRAHB' 2'
PROCOR		770	9/29/64	1499	40	SMS	M	PROH' CORE
FASCOR								
FASEPC								

* PROCOR TO QXCOR1 *

FASCR1								
FASEP1								
PSQRT		155	10/ 5/64	91	9	JFC	F	P.' SQUIRT
	SQRT							
PWMLIV		300	9/29/64	142	15	SMS	F	PWIM' LIV
	MLI2A6							
	(STH)							
	(FIL)							
	(SPH)							
QACORR		207	9/29/64	184	11	SMS	F	KACK' CORE
	FXDATA							
	PROCOR							
	FASCOR							
	FLDATA							
QCNVLV		569	9/29/64	294	27	SMS	F	Q.' CONVOLVE
	XLOC							
	FXDATA							
	PROCOR							
	FASCOR							
	FASEPC							
	FLDATA							
QFURRY		244	9/29/64	181	13	SMS	F	Q.' FURRY
	STZ							
	MOVE							
	COSTBL							
	SINTBL							
	XSPECT							
QIFURY		280	9/29/64	206	14	SMS	F	Q. I.' FURRY
	COSTBL							
	SINTBL							
	COSISP							
	XLOC							
QINTR1		229	9/ 4/64	192	12	JTP	F	KINT' ER 1'
	RNDUP							
	QUFIT1							
QUFIT1		79	9/ 4/64	200	5	SMS	M	Q.' FIT 1'
QXCORR		283	9/29/64	249	15	SMS	F	KIX' CORE
	XLOC							
	FXDATA							
	PROCOR							
	FASCOR							
	FLDATA							
QXCOR1		502	3/15/65	198	25	RAW	F	KIX' CORE 1'
	SETKS							
	IXCARG							

	LIMITS							
	STZ							
	REVERS							
	PROCOR							
	FASCR1							
	FASEP1							
RDATA		645	3/15/65	396	31	RAW	F	R.' DATA
	SETUP							
	RETURN							
	IXCARG							
	(TSH)							
	(RTN)							
	(STH)							
	(FIL)							
	HVTOIV							
	IVTOHV							
	CMPRA							
	ARG							
	INTHOL							
	STORE							
REFIT (SEE SPLIT)								REE' FIT
REFLEC		28	9/29/64	108	3	SMS	M	REE' FLEK
XRFLEC								
REIM (SEE AMPHZ)								REE' IM
REMAV		36	9/29/64	106	3	SMS	M	REM' AV
REREAD		114	9/ 9/64	283	7	RAW	M	REE' READ
EOFSET	(IOH)							
ENDFIL	(RDS)							
(TSH)	(RDC)							
(TSHM)	(RCH)							
	(TCO)							
	(TEF)							
	EXIT							
	(RER)							
RETURN (SEE LOCATE)								RETURN
REVER		30	9/29/64	98	3	SMS	M	REV' ER
REVERS		29	9/29/64	77	3	RAW	M	REVERSE
RLSPR		142	10/ 5/64	121	8	RAW	F	R.' LISPER
	FDOTR							
RLSPR2		700	9/ 9/64	281	34	RAW	F	R.' LISPER 2'
	IXCARG							
	STZ							
	MOVREV							
	DOTP							

	MATML3							
	DOTJ							
	SIMEQ							
RLSSR		82	9/29/64	115	5	RAW	F	R.' LESSER
	FDOTR							
RMSDAV (SEE RMSDEV)								R. M. S. DAV'
RMSDEV		50	9/ 4/64	160	4	SMS	M	R. M. S. DEEV'
RMSDAV	SQRT							
RND		15	9/29/64	79	2	RAW	M	ROUND
RNDUP								
RNDDN								
RNDDN (SEE RND)								ROUND' DOWN'
RNDUP (SEE RND)								ROUND' UP'
RNDV		34	9/29/64	118	3	SMS	M	ROUND' V.'
RNDVUP	RND							
RNDVDN	RNDUP							
	RNDDN							
RNDVDN (SEE RNDV)								ROUND' V. DOWN'
RNDVUP (SEE RNDV)								ROUND' V. UP'
ROAR2		174	9/10/64	114	9	RAW	F	ROAR' 2'
	MATRA							
	MOVREV							
	REVERS							
ROTAT1		46	9/ 4/64	110	4	RAW+	M	ROTATE' 1'
						JC		
RPLFMT		17	9/29/64	85	2	SMS	M	RIPPLE' FUMT
RSKIP		37	9/29/64	90	3	RAW	M	R.' SKIP
	(IOS)							
	(TRC)							
	(TCO)							
	(TEF)							
	(RDS)							
	(BSR)							
RVPRTS (SEE CHPRTS)								REV' PARTS
SAME		1	9/29/64	40	2	JFC	M	SAME
XSAME								
SCPSCL		33	9/29/64	111	3	SMS	M	SKUP' SCALE
SEARCH		25	9/29/64	95	3	RAW	M	SEARCH

SEQSAC		94	9/ 8/64	278	6	SMS	M	SEEK' SACK
NEXCOS	COS							
NEXSIN	SIN							
SETAPT (SEE INDEX)								SET APT'
SETEST (SEE INDEX)								SEH TEST'
SETINO		84	9/ 8/64	92	6	SMS	F	SEH TEE' NOH
	XLIMIT							
	(RWT)							
	(TSB)							
	(RLR)							
	FSKIP							
SETK		37	9/29/64	190	3	SMS	M	SET' K.'
SETKS								
SETVEC								
SETK -II		63	9/29/64	73	4	SMS	F	SET' K. DASH 2'
	SETUP							
	STORE							
	RETURN							
SETKP		40	9/29/64	124	3	SMS	M	SET' K. P.'
SETVCP	SETK							
	SETVEC							
SETKS (SEE SETK)								SET' KAYZ'
SETKS -II		91	9/29/64	86	6	SMS	F	SET' KAYZ DASH 2'
	SETUP							
	ARG							
	STORE							
	RETURN							
SETKV		15	9/29/64	75	2	SMS	M	SET' K. V.'
SETKVS		25	9/29/64	106	3	SMS	M	SET' K. VEEZ'
SETLIN		27	9/29/64	95	3	SMS	M	SET' LIN
XSTLIN								
SETLNS		39	9/29/64	124	3	SMS	M	SET' LINZ
	SETLIN							
	XSTLIN							
SETSBV (SEE LOCATE)								SET' SUB V.'
SETUP (SEE LOCATE)								SET' UP
SETVCP (SEE SETKP)								SET' V.C.P.'
SETVEC (SEE SETK)								SET' VEK

SEVRAL		416	9/29/64	949	22	SMS	M	SEV' RUL
PLURAL	LOCATE							
''DO''	WHERE							
''IF''								
SHFTR1		70	9/29/64	158	5	SMS	M	SHIFT' ER 1'
SHFTR2		72	9/29/64	163	5	SMS+	M	SHIFT' ER 2'
						RAW		
SHUFFL		101	9/ 8/64	125	6	SMS	F	SHUFFLE
	GETRD1							
	SEARCH							
	SIZEUP							
SIFT		30	9/ 4/64	118	3	SMS	M	SIFT
SIMEQ		441	9/ 9/64	642	24	JTO+	M	SIME' E. Q.
DETRM						AMN+		
						RAW		
SINTBL (SEE COSTBL)								SINE' TUB L.'
SINTBX (SEE COSTBL)								SINE' TUB X.'
SISP (SEE COSP)								SISP
SIZEUP		136	3/15/65	247	8	RAW+	M	SIZE' UP
SIZUPL						SMS		
SIZUPL (SEE SIZEUP)								SIZE' UP L.'
SMPRDV (SEE POWER)								SUM' PER DEEV
SMPSON		317	9/ 4/64	197	17	JNG	F	SMIP' SON
SPCOR2		291	9/ 8/64	181	15	RAW	F	SPUH CORE' 2'
	XLOC							
	STZ							
	FXDATA							
	QXCOR1							
	FLDATA							
SPLIT		224	9/29/64	395	13	SMS	M	SPLIT
REFIT								
SQRDEV (SEE SQRDFR)								SKUR DEEV'
SQRDFR		36	9/29/64	111	3	SMS	M	SKUR DIFFER'
SQRDEV								
SQRMLI		55	9/29/64	128	4	SMS	M	SQUIRM' LEE
SQROOT		24	9/29/64	83	3	SMS	M	SKUH ROOT'
	SQRT							

SQRSUM			36	9/29/64	107	3	SMS	M	SKUR SUM'
XSQSUM									
SQUARE			32	9/29/64	111	3	SMS	M	SQUARE
XSQUAR									
SRCH1			93	9/ 8/64	93	6	RAW	F	SEARCH' 1'
	XACTEQ								
STEPC	(SEE DELTA)								STEP' C'
STEPL	(SEE DELTA)								STEP' L.'
STEPR	(SEE DELTA)								STEP' R.'
(STH)	(SEE ONLINE)								S.' T. H.'
(STHD)	(SEE ONLINE)								S.' T. H. D.'
(STHM)	(SEE ONLINE)								S.' T. H. M.'
STORE	(SEE LOCATE)								STORE
STZ			14	9/29/64	60	2	JFC	M	S. T. Z.'
STZS			24	9/29/64	97	3	SMS	M	S. T. ZEEZ'
SUBK	(SEE ADDK)								SUB' K.
SUBKS	(SEE ADDK)								SUB KAYZ'
SUM			23	9/29/64	92	3	SMS	M	SUM
XSUM									
SUMDEV	(SEE SUMDFR)								SUM DEEV'
SUMDFR			44	9/29/64	156	4	SMS	M	SUM DIFFER'
XSMDFR									
SUMDEV									
XSMDEV									
SWITCH			15	9/ 4/64	84	2	SMS	M	SWITCH
TAMVL			63	9/ 4/64	189	5	SMS	M	TAM' VUL
TAMVR									
TAMVR	(SEE TAMVL)								TAM' VER
TIMA2B	(7094)		124	9/ 9/64	258	8	SMS+ RAW	M	TIME' A.' TO B.'
TIMSUB			229	9/ 8/64	450	13	SMS+ RAW	M	TIME' SUB
INTMSB	TIMA2B								
TINGL			43	9/ 8/64	147	4	SMS	M	TINGLE
TINGLA									

TINGLA (SEE TINGL)								TING' GLAH
TRMINO		67	9/ 4/64	77	5	SMS	F	TUR MEEN' OH
	XLIMIT							
	OUDATA							
	FSKIP							
	(RWT)							
(TSH) (SEE REREAD)								T.' S. H.'
(TSHM) (SEE REREAD)								T.' S. H.' M.'
UNPAKN		78	9/ 9/64	150	5	JFC	M	UNPACK' N.'
VARARG		44	9/29/64	132	4	JFC	M	VAR' ARG
VDOTV		25	9/ 4/64	121	3	SMS	M	V.' DOT V.'
VDVBYV		22	9/29/64	90	3	SMS	M	V.' D. V.' BY V.'
VECOUT		66	9/29/64	91	5	SMS	F	VEK' OUT
	FMDFMT							
	RPLFMT							
	(STH)							
	(FIL)							
VINDEX (SEE INDEX)								V.' INDEX'
VMNUSV (SEE VPLUSV)								V.' MINUS V.'
VOUT		104	9/29/64	111	7	SMS	F	V.' OUT
	CARIGE							
	HRADJ							
	(STH)							
	(FIL)							
	VECOUT							
VPLUSV		34	9/29/64	127	3	SMS	M	V.' PLUS V.'
XVPLSV								
VMNUSV								
XVMNSV								
VRSOUT		47	9/29/64	138	4	SMS	M	VERZ' OUT
	CARIGE							
	VECOUT							
VSOUT		37	9/29/64	125	3	SMS	M	VEEZ' OUT
	VOUT							
VTIMSV		34	9/29/64	112	3	SMS	M	V.' TIMES V.'
XVTMSV								
WAC		107	9/29/64	83	6	JFC	F	WACK

WHERE (SEE LOCATE).								WHERE
WHICH XWHICH		4	9/ 4/64	77	2	SMS	M	WHICH
WLLSFP	FDOTR FDOT MOVE	217	10/ 6/64	264	11	RAW	F	WILL' USS FUHP
WRTDAT	(IOS) (TCO) (WRS) (RCH) (TRC) (ETT)	77	9/ 8/64	126	5	RAW	M	WRITE' DAT
XACTEQ		11	9/ 4/64	76	2	SMS	M	EXACT' E. Q.
XADDK (SEE ADDK)								X. ADD' K.
XADDKS (SEE ADDK)								X. ADD KAYZ'
XARG (SEE LOCATE).								X. ARG'
XAVRGE XAVRGR	XDIV XDIVR	34	9/29/64	104	3	SMS	M	X. AV' REDGE
XAVRGR (SEE XAVRGE).								X. AV' REDGE R.'
XBOOST (SEE BOOST)								X. BOOST'
XCMPRA (SEE CMPRA)								X. KUM' PRA
XDANL (SEE ADANL)								X.' DAN L.'
XDANX (SEE ADANL)								X.' DAN X.'
XDELTA (SEE DELTA)								X. DELTA'
XDFPRS (SEE DIFPRS).								X. DIFF' PURZ
XDIV XDIVR		27	9/29/64	109	3	SMS	M	X. DIV'
XDIVK (SEE ADDK)								X. DIV' K.
XDIVKS (SEE ADDK)								X. DIV KAYZ'
XDIVR (SEE XDIV)								X. DIV' ER
XDPRSS (SEE BOOST)								X. DEPRESS'

XDVIDE		33	9/29/64	105	3	SMS	M	X. DIVIDE'
XDVIDR	XDIV							
	XDIVR							
XDVIDR (SEE XDVIDE)								X. DIVIDE R.'
XDVRK (SEE ADDK)								X. DIV' ER K.'
XDVRKS (SEE ADDK)								X. DIV' ER KAYZ'
XFIXM		31	9/29/64	98	3	SMS	M	X. FIX' UM
XINDEX (SEE LOCATE)								X. INDEX'
XLCOMN		14	9/ 4/64	76	2	RAW	M	X.' L. COMMON'
XLIMIT		25	9/ 4/64	101	3	SMS	M	X. LIMIT'
XLOCV		24	9/ 4/64	100	3	SMS	M	X. LCKE V.'
XLSHFT (SEE LSHFT)								X. L.' SHIFT
XMLPLY (SEE MULPLY)								X. MUL' PLEE
XMULK (SEE ADDK)								X. MUL' K.
XMULKS (SEE ADDK)								X. MUL KAYZ'
XNAME (SEE LOCATE)								X. NAME'
XNARGS (SEE LOCATE)								X. NARGS'
XNTHA (SEE NTHA)								X. ENTH' UH
XNTSUM (SEE INTSUM)								X. INT' SUM
XOOZE		4	9/ 4/64	61	2	SMS	M	X. OOZE'
XREMAV		31	9/29/64	112	3	SMS	M	X. REH MAV'
	XAVRGR							
XRFLEC (SEE REFLEC)								X. REE FLEK'
XSAME (SEE SAME)								X. SAME'
XSMDEV (SEE SUMDFR)								X. SUM DEEV'
XSMDFR (SEE SUMDFR)								X. SUM DIFFER'
XSPECT		523	9/29/64	239	26	SMS	F	X.' SPECT
	SPLIT							
	COSISP							
	REFIT							
	XLOC							
	KOLAPS							
	CHPRTS							

XSQDEV (SEE XSQDFR)								X. SKUH DEEV'
XSQDFR		37	9/29/64	113	3	SMS	M	X. SKUH DIFFER'
XSQDEV								
XSQRUT		37	9/29/64	103	3	SMS	M	X. SKUH ROOT'
	FIXVR							
	SQRT							
XSQSUM (SEE SQRSUM)								X. SKUH SUM'
XSQUAR (SEE SQUARE)								X. SKWAHR'
XSTEPC (SEE DELTA)								X. STEP C.'
XSTEPL (SEE DELTA)								X. STEP L.'
XSTEPR (SEE DELTA)								X. STEP R.'
XSTLIN (SEE SETLIN)								X. SET' LIN
XSUBK (SEE ADDK)								X. SUB' K.
XSUBKS (SEE ADDK)								X. SUB KAYZ'
XSUM (SEE SUM)								X. SUM'
XVDRBV (SEE XVDVBV)								X.V.' D.R.B.V.'
XVDVBV		34	9/29/64	109	3	SMS	M	X.V.' D.V.' B.V.'
XVDRBV	XDIV							
	XDIVR							
XVMNSV (SEE VPLUSV)								X. V.' MINUS V.'
XVPLSV (SEE VPLUSV)								X. V.' PLUS V.'
XVTMSV (SEE VTIMSV)								X. V.' TIMES V.'
XWHICH (SEE WHICH)								X. WHICH'
ZEFBCD		54	9/ 8/64	129	4	JNG	M	ZEFF' B. C. D.'
ZEFBIN	(IOS)							
	(RDS)							
	(RCH)							
	(TCO)							
	(TEF)							
	(TRC)							
	(BSR)							
ZEFBIN (SEE ZEFBCD)								ZEFF' BIN'

7
A One-Pass Subroutine Library

A subroutine library in the FORTRAN Monitor System (FMS) is a magnetic tape file containing binary mode images of the column binary subprogram decks forming the library. It contains no table of contents, and this fact gives rise to the first* problem of library design, namely the problem of arranging routines in one-pass order. This term refers to the behavior of the FMS loading program, which, at the beginning of each execution, passes continuously through the library file gathering all subprograms required by, but missing from, the input deck, plus all additional subprograms required by those gathered. If the library is so arranged that for every program the loader can pick up, all of the lower-level programs are physically located deeper in the file, then all programs required for any execution are retrievable by the loader in one pass, or less, of the library. Otherwise the loader must return to the beginning of the file and start searching again.

For example, if the library contains program A which requires program B, and B which requires program C, and C which has no requirements, then the order A,B,C is one-pass, whereas the order C,B,A is three-pass (in the event that the input deck refers only to program A). In this example note that if program C requires program A, one-passness can be realized only by using redundant copies, i.e., A,B,C,A. In a library the size of the present one, a single pass takes half a minute or more. Consequently the one-pass property is economically important.

The second problem of library design is strategic arrangement, within the one-pass constraint, designed to minimize the average (over many executions) distance that the loader must penetrate the library file before its search is ended. The controlling factors in this problem are the natures and frequencies of expected input deck requirements, the logical relationships between the programs in the library, and the physical lengths of these programs. As a general guide short programs and often used programs should appear early, seldom used and longer programs late in the library. However, since this rule will often be in conflict with the one-pass constraint, one resorts to sprinkling redundant copies of key programs throughout the library so as to expand the arrangement possibilities within the constraint. The redundancy must be limited, however, since by lengthening the entire library it tends to cancel its own benefits.

The main portion of this section is made up of listings of a one-pass library, composed from the programs of Section 10 plus the standard FORTRAN System routines

*Assuming that the more basic problem of completness is satisfied, i.e., no program in the library calls on any program not in the library.

(including double precision and complex arithmetic routines) and arranged for minimizing average search time with respect to our usage experience. Where Section 10 has more than one program of the same name the versions selected for the library are as follows.

On library		Excluded	
CONVLV-II		CONVLV	
DISPLA	(7090)	DISPLA	(709)
FRAME	(7090)	FRAME	(709)
FT24	-II	FT24	
HSTPLT		HSTPLT-II, HSTPLT-III (709), HSTPLT-III (7090)	
LINE	(7090)	LINE	(709)
LINEH	(7090)	LINEH	(709)
LINEV	(7090)	LINEV	(709)
ADDK	(which has MULK as a secondary entry)	MULK	-II
SETK		SETK	-II
SETK	(which has SETKS as a secondary entry)	SETKS	-II

Consequently, the library is designed for the 7090 or 7094. The 7090 programs work on the 7094 and vice versa, except that TIMA2B (7094) must be modified as indicated in the listing of Section 10 to give correct results on the 7090.

The modifications required by an adaptation of the library to the 709 consist of swapping the 709 and 7090 programs as in the above list and deleting the following programs (for which we have no 709 versions):

CLOCK 1(7090), CLKON, TIMSUB, TIMA2B (7094).

The library has 402 principal entries, of which 99 are redundant copies. The first table below lists these 402 entries in the order of their occurence in the library (their storage requirements and binary card counts are also given). Following this table is another giving an alphabetized ordering of the 402 principal entries with their corresponding index positions within the library.

The following rule will enable one to distinguish FORTRAN System routines. A principle entry name is that of a FORTRAN System routine if either of the following is true.

1. its first character is a left parenthesis, or
2. it is from the following list of 27 routines.

ATAN	CHAIN	COS	DATAN	DEXP	DEXP(3
DINT	DLOG	DMOD	DSIN	DSQRT	DUMP
EXIT	EXP	EXP(1	EXP(2	EXP(3	IABS
IEXP	IEXP(2	ILOG	ISIN	ISQRT	LOG
SQRT	TANH	XLOC			

SUBROUTINE LIBRARY PRINCIPAL ENTRIES, STORAGE LENGTHS, BINARY CARD COUNTS

1.	(FPT)	41	4	56.	XACTEQ	11	2	111.	CARIGE	47	4
2.	(IOH)	1016	52	57.	ADDK	114	8	112.	MAXSNM	61	5
3.	(IOS)	87	7	58.	CHOOSE	17	2	113.	GNHOL2	74	5
4.	(EXEM)	458	24	59.	CMPRA	18	2	114.	CPYFL2	178	10
5.	(IOU)	24	3	60.	DELTA	17	2	115.	EXCHVS	22	3
6.	DUMP	177	7	61.	INDEX	50	4	116.	MULPLY	34	3
7.	EXIT	17	2	62.	LIMITS	44	4	117.	PLURNS	73	5
8.	(TES)	1	2	63.	LSHFT	12	2	118.	SRCH1	93	6
9.	ONLINE	134	8	64.	NTHA	11	2	119.	XACTEQ	11	2
10.	CSOUT	49	4	65.	SETKP	40	3	120.	AMPHZ	149	10
11.	CVSOUT	84	6	66.	SETK	37	3	121.	COS	105	7
12.	FMTOUT	51	4	67.	SETKV	15	2	122.	ATAN	77	5
13.	REREAD	114	7	68.	SETKVS	25	3	123.	GETHOL	169	9
14.	VRSOUT	47	4	69.	SETLNS	39	3	124.	REVERS	29	3
15.	VSOUT	37	3	70.	SETLIN	27	3	125.	MEMUSE	71	5
16.	VOUT	104	7	71.	SWITCH	15	2	126.	XLCOMN	14	2
17.	CARIGE	47	4	72.	WHICH	4	2	127.	(STH)	83	5
18.	VECOUT	66	5	73.	XLIMIT	25	3	128.	(WER)	57	4
19.	RPLFMT	17	2	74.	XOOZE	4	2	129.	CROST	134	8
20.	FNDFMT	88	6	75.	DOTJ	59	4	130.	CROSS	107	7
21.	REVER	30	3	76.	FIXV	35	3	131.	STZ	14	2
22.	HLADJ	46	4	77.	FSKIP	50	4	132.	LSSS1	122	7
23.	(STH)	83	5	78.	MATRA	92	6	133.	PLTVS1	817	40
24.	(STB)	53	4	79.	MAXSN	54	5	134.	MULPLY	34	3
25.	(WER)	57	4	80.	XLCOMN	14	2	135.	MAXSN	54	5
26.	(TSB)	66	5	81.	(SLI)	13	2	136.	SETLIN	27	3
27.	(RER)	37	3	82.	(SLO)	13	2	137.	SETKVS	25	3
28.	(IOB)	570	6	83.	MOUT	130	8	138.	PLOTVS	494	18
29.	(BST)	28	3	84.	CARIGE	47	4	139.	SWITCH	15	2
30.	(CSH)	125	8	85.	(STH)	83	5	140.	SETKV	15	2
31.	(EFT)	7	2	86.	(WER)	57	4	141.	SETK	37	3
32.	(RWT)	7	2	87.	GETX	31	3	142.	RND	15	2
33.	(SCH)	96	6	88.	TIMSUB	229	13	143.	(SPH)	183	11
34.	IXCARG	35	3	89.	TIMA2B	124	8	144.	BOOST	34	3
35.	XLOC	12	2	90.	LOCATE	512	28	145.	RMSDEV	50	4
36.	CLKON	46	4	91.	RDATA	645	31	146.	SQRT	44	4
37.	CLOCK1	57	4	92.	CMPRA	18	2	147.	RLSPR	142	8
38.	(SPH)	183	11	93.	IXCARG	35	3	148.	RLSSR	82	5
39.	RND	15	2	94.	XLOC	12	2	149.	SHUFFL	101	6
40.	SAME	1	2	95.	REREAD	114	7	150.	SEARCH	25	3
41.	ARCTAN	29	3	96.	(RER)	37	3	151.	GETRD1	229	10
42.	ATAN	77	5	97.	INTHOL	72	5	152.	REREAD	114	7
43.	COSTBL	121	8	98.	FNDFMT	88	6	153.	(RER)	37	3
44.	COS	105	7	99.	REVER	30	3	154.	SIZEUP	136	8
45.	EXP	52	4	100.	HVTOIV	39	3	155.	CMPARP	53	4
46.	EXP(1	35	3	101.	IVTOHV	70	5	156.	FDOT	40	3
47.	EXP(2	38	3	102.	DADECK	100	6	157.	IPLYEV	98	6
48.	EXP(3	93	6	103.	RSKIP	37	3	158.	(IFMP)	136	8
49.	LOG	53	4	104.	MOUTAI	357	18	159.	QACORR	207	11
50.	SQRT	44	4	105.	FIXV	35	3	160.	QCNVLV	569	27
51.	TANH	86	6	106.	MOVE	32	3	161.	QXCORR	283	15
52.	MOVE	32	3	107.	LOG	53	4	162.	SPCOR2	291	15
53.	MOVREV	74	5	108.	EXP(2	38	3	163.	QXCOR1	502	25
54.	REVERS	29	3	109.	SAME	1	2	164.	REVERS	29	3
55.	STZ	14	2	110.	RND	15	2	165.	IXCARG	35	3

SUBROUTINE LIBRARY PRINCIPAL ENTRIES, STORAGE LENGTHS, BINARY CARD COUNTS

I	166.	XLOC	12	2	I	221.	TAMVL	63	5	I	276.	MATML3	120	7	I
I	167.	PROCOR	770	40	I	222.	TINGL	43	4	I	277.	DOTP	264	14	I
I	168.	ROTAT1	46	4	I	223.	VDOTV	25	3	I	278.	DOTJ	59	4	I
I	169.	STZS	24	3	I	224.	VDVBYV	22	3	I	279.	SIMEQ	441	24	I
I	170.	ZEFBCD	54	4	I	225.	VPLUSV	34	3	I	280.	IDERIV	54	4	I
I	171.	FXDATA	102	7	I	226.	VTIMSV	34	3	I	281.	MFACT	187	10	I
I	172.	IFNCTN	208	12	I	227.	WLLSFP	216	11	I	282.	DOTJ	59	4	I
I	173.	REVER	30	3	I	228.	FDOT	40	3	I	283.	STZ	14	2	I
I	174.	MONOCK	48	4	I	229.	XDVIDE	33	3	I	284.	SQRT	44	4	I
I	175.	POLYDV	130	7	I	230.	XREMAV	31	3	I	285.	NURINC	121	8	I
I	176.	STZ	14	2	I	231.	XAVRGE	34	3	I	286.	SIFT	30	3	I
I	177.	POLYEV	54	4	I	232.	XVDVBV	34	3	I	287.	CNTRDB	550	27	I
I	178.	POLYSN	256	14	I	233.	XDIV	27	3	I	288.	SETK	37	3	I
I	179.	MOVE	32	3	I	234.	XLOCV	24	3	I	289.	EXP	52	4	I
I	180.	ABSVAL	50	4	I	235.	XSQDFR	37	3	I	290.	CONTUR	587	29	I
I	181.	ADANL	183	11	I	236.	XSQRUT	37	3	I	291.	SWITCH	15	2	I
I	182.	AVRAGE	24	3	I	237.	FIXV	35	3	I	292.	COLABL	185	10	I
I	183.	BLKSUM	49	4	I	238.	ASPECT	278	15	I	293.	CNTROW	802	39	I
I	184.	CHSIGN	18	2	I	239.	COLAPS	50	4	I	294.	ARBCOL	129	8	I
I	185.	CMPARV	50	4	I	240.	DUBLX	45	4	I	295.	CUFIT1	158	9	I
I	186.	CONVLV	56	4	I	241.	FACTOR	308	17	I	296.	FASCUB	141	9	I
I	187.	DERIVA	61	5	I	242.	MAXSN	54	5	I	297.	GENHOL	48	4	I
I	188.	DIFPRS	30	3	I	243.	LOG	53	4	I	298.	EXPAND	189	11	I
I	189.	DIVIDE	23	3	I	244.	EXP	52	4	I	299.	INTOPR	111	7	I
I	190.	FLOATV	22	3	I	245.	PLANSP	1169	56	I	300.	QINTR1	229	12	I
I	191.	IINTGR	49	4	I	246.	XOOZE	4	2	I	301.	RND	15	2	I
I	192.	INTGRA	47	4	I	247.	SETK	37	3	I	302.	QUFIT1	79	5	I
I	193.	INTSUM	27	3	I	248.	LIMITS	44	4	I	303.	LISTNG	755	38	I
I	194.	ITOMLI	37	3	I	249.	CHOOSE	17	2	I	304.	FSKIP	50	4	I
I	195.	MDOT	109	7	I	250.	ADDK	114	8	I	305.	(SPH)	183	11	I
I	196.	MATML1	61	5	I	251.	COSIS1	406	21	I	306.	(RWT)	7	2	I
I	197.	MLISCL	47	4	I	252.	ROAR2	174	9	I	307.	(TSB)	66	5	I
I	198.	MOVECS	24	3	I	253.	REVERS	29	3	I	308.	(IOB)	570	6	I
I	199.	MPSEQ1	110	7	I	254.	MOVREV	74	5	I	309.	SHFTR2	72	5	I
I	200.	MRVRS	61	4	I	255.	QFURRY	244	13	I	310.	TRMINO	67	5	I
I	201.	MUVADD	129	8	I	256.	MOVE	32	3	I	311.	OUDATA	495	11	I
I	202.	MVINAV	221	12	I	257.	XSPECT	523	26	I	312.	(EFT)	7	2	I
I	203.	MVNSUM	71	5	I	258.	SPLIT	224	13	I	313.	(STB)	53	4	I
I	204.	MVNTIN	88	6	I	259.	KOLAPS	100	6	I	314.	PAKN	78	5	I
I	205.	MVSQAV	236	13	I	260.	CHPRTS	76	5	I	315.	FXDATA	102	7	I
I	206.	NMZMG1	34	3	I	261.	QIFURY	280	14	I	316.	ASPEC2	74	5	I
I	207.	NRMVEC	111	7	I	262.	COSTBL	121	8	I	317.	SEQSAC	94	6	I
I	208.	MAXSN	54	5	I	263.	COS	105	7	I	318.	INDATA	896	32	I
I	209.	POWER	50	4	I	264.	COSP	504	27	I	319.	SAME	1	2	I
I	210.	REFLEC	28	3	I	265.	CRSVM	327	17	I	320.	UNPAKN	78	5	I
I	211.	REMAV	36	3	I	266.	FIRE2	271	14	I	321.	SETINO	84	6	I
I	212.	RNDV	34	3	I	267.	MIFLS	276	14	I	322.	FSKIP	50	4	I
I	213.	SHFTR1	70	5	I	268.	MIPLS	571	28	I	323.	XLIMIT	25	3	I
I	214.	SQRDFR	36	3	I	269.	MATRA	92	6	I	324.	(RWT)	7	2	I
I	215.	SQRMLI	55	4	I	270.	MATINV	90	6	I	325.	(TSB)	66	5	I
I	216.	SQROOT	24	3	I	271.	MISS	335	17	I	326.	(IOB)	570	6	I
I	217.	SQRSUM	36	3	I	272.	MDOT3	122	7	I	327.	(RER)	37	3	I
I	218.	SQUARE	32	3	I	273.	RLSPR2	700	34	I	328.	FAPSUM	14	2	I
I	219.	SUM	23	3	I	274.	MOVREV	74	5	I	329.	KIINT1	191	10	I
I	220.	SUMDFR	44	4	I	275.	IXCARG	35	3	I	330.	EXP(3	93	6	I

SUBROUTINE LIBRARY PRINCIPAL ENTRIES, STORAGE LENGTHS, BINARY CARD COUNTS

I	331.	NOINT1	369	20	I	386.	ILOG	190	11	I
I	332.	LINTR1	96	6	I	387.	ISIN	184	11	I
I	333.	LOC	4	2	I	388.	ISQRT	88	6	I
I	334.	MVBLOK	19	2	I	389.	(DFAD)	80	5	I
I	335.	VARARG	44	4	I	390.	DATAN	440	24	I
I	336.	GRAPHX	123	7	I	391.	DEXP(3	34	3	I
I	337.	GRAPH	1499	72	I	392.	DEXP	153	9	I
I	338.	MVBLOK	19	2	I	393.	DLOG	273	15	I
I	339.	LOG	53	4	I	394.	DINT	10	2	I
I	340.	EXP(2	38	3	I	395.	DMOD	48	4	I
I	341.	XLOC	12	2	I	396.	DSIN	222	13	I
I	342.	DISPLA	219	13	I	397.	DSQRT	66	5	I
I	343.	DSPFMT	194	11	I	398.	PLYSYN	172	10	I
I	344.	FLOATM	25	3	I	399.	CONVLV	56	4	I
I	345.	FRAME	9	2	I	400.	COS	105	7	I
I	346.	HSTPLT	145	9	I	401.	PSQRT	155	9	I
I	347.	LINE	95	6	I	402.	SQRT	44	4	I
I	348.	LINEH	35	3	I					
I	349.	LINEV	35	3	I					
I	350.	SCPSCL	33	3	I					
I	351.	XFIXM	31	3	I					
I	352.	MULLER	757	36	I					
I	353.	CHISQR	105	6	I					
I	354.	FASTRK	26	3	I					
I	355.	FRQCT2	117	7	I					
I	356.	GNFLT1	232	12	I					
I	357.	GRUP2	201	11	I					
I	358.	LSLINE	117	7	I					
I	359.	MATRA1	42	4	I					
I	360.	MSCON1	238	11	I					
I	361.	PACDAT	152	9	I					
I	362.	POKCT1	219	11	I					
I	363.	FRQCT1	117	7	I					
I	364.	PRBFIT	373	16	I					
I	365.	EXP	52	4	I					
I	366.	PROB2	229	12	I					
I	367.	WAC	107	6	I					
I	368.	WRTDAT	77	5	I					
I	369.	PWMLIV	300	15	I					
I	370.	(SPH)	183	11	I					
I	371.	(STH)	83	5	I					
I	372.	(WER)	57	4	I					
I	373.	MLI2A6	128	8	I					
I	374.	SMPSON	317	17	I					
I	375.	FT24	818	39	I					
I	376.	MXRARE	302	16	I					
I	377.	EXP(2	38	3	I					
I	378.	NXALRM	243	13	I					
I	379.	FASCN1	107	7	I					
I	380.	SEVRAL	416	22	I					
I	381.	LOCATE	512	28	I					
I	382.	CHAIN	179	10	I					
I	383.	IABS	21	3	I					
I	384.	IEXP	157	9	I					
I	385.	IEXP(2	161	9	I					

SUBROUTINE LIBRARY PRINCIPAL ENTRIES ALPHABETIZED, WITH ENTRY INDICES

(BST)	29	BOOST	144	DSPFMT	343	IABS	383	MFACT	281
(CSH)	30	CARIGE	17	DSQRT	397	IDERIV	280	MIFLS	267
(DFAD)	389	CARIGE	111	DUBLX	240	IEXP	384	MIPLS	268
(EFT)	312	CARIGE	84	DUMP	6	IEXP(2	385	MISS	271
(EFT)	31	CHAIN	382	EXCHVS	115	IFNCTN	172	MLISCL	197
(EXEM)	4	CHISQR	353	EXIT	7	IINTGR	191	MLI2A6	373
(FPT)	1	CHOOSE	249	EXP	45	ILOG	386	MONCCK	174
(IFMP)	158	CHOOSE	58	EXP	289	INDATA	318	MOUT	83
(IOB)	326	CHPRTS	260	EXP	365	INDEX	61	MOUTAI	104
(IOB)	308	CHSIGN	184	EXP	244	INTGRA	192	MOVE	52
(IOB)	28	CLKON	36	EXP(1	46	INTHOL	97	MOVE	106
(IOH)	2	CLOCK1	37	EXP(2	340	INTOPR	299	MOVE	179
(IOS)	3	CMPARP	155	EXP(2	377	INTSUM	193	MOVE	256
(IOU)	5	CMPARV	185	EXP(2	108	IPLYEV	157	MOVECS	198
(RER)	96	CMPRA	92	EXP(2	47	ISIN	387	MOVREV	274
(RER)	153	CMPRA	59	EXP(3	330	ISQRT	388	MOVREV	53
(RER)	327	CNTRDB	287	EXP(3	48	ITOMLI	194	MOVREV	254
(RER)	27	CNTROW	293	EXPAND	298	IVTOHV	101	MPSEQ1	199
(RWT)	32	COLABL	292	FACTOR	241	IXCARG	275	MRVRS	200
(RWT)	306	COLAPS	239	FAPSUM	328	IXCARG	165	MSCCN1	360
(RWT)	324	CONTUR	290	FASCN1	379	IXCARG	93	MULLER	352
(SCH)	33	CONVLV	399	FASCUB	296	IXCARG	34	MULPLY	134
(SLI)	81	CONVLV	186	FASTRK	354	KIINT1	329	MULPLY	116
(SLO)	82	COS	263	FDOT	156	KOLAPS	259	MUVADD	201
(SPH)	38	COS	121	FDOT	228	LIMITS	62	MVBLOK	338
(SPH)	370	COS	400	FIRE2	266	LIMITS	248	MVBLOK	334
(SPH)	305	COS	44	FIXV	105	LINE	347	MVINAV	202
(SPH)	143	COSIS1	251	FIXV	76	LINEH	348	MVNSUM	203
(STB)	24	COSP	264	FIXV	237	LINEV	349	MVNTIN	204
(STB)	313	COSTBL	43	FLOATM	344	LINTR1	332	MVSQAV	205
(STH)	371	COSTBL	262	FLOATV	190	LISTNG	303	MXRARE	376
(STH)	127	CPYFL2	114	FMTOUT	12	LOC	333	NMZMG1	206
(STH)	85	CROSS	130	FNDFMT	98	LOCATE	90	NOINT1	331
(STH)	23	CROST	129	FNDFMT	20	LOCATE	381	NRMVEC	207
(TES)	8	CRSVM	265	FRAME	345	LOG	243	NTHA	64
(TSB)	307	CSOUT	10	FRQCT1	363	LOG	107	NURINC	285
(TSB)	26	CUFIT1	295	FRQCT2	355	LOG	49	NXALRM	378
(TSB)	325	CVSOUT	11	FSKIP	322	LOG	339	ONLINE	9
(WER)	128	DADECK	102	FSKIP	77	LSHFT	63	OUDATA	311
(WER)	372	DATAN	390	FSKIP	304	LSLINE	358	PACDAT	361
(WER)	25	DELTA	60	FT24	375	LSSS1	132	PAKN	314
(WER)	86	DERIVA	187	FXDATA	315	MATINV	270	PLANSP	245
ABSVAL	180	DEXP	392	FXDATA	171	MATML1	196	PLOTVS	138
ADANL	181	DEXP(3	391	GENHOL	297	MATML3	276	PLTVS1	133
ADDK	250	DIFPRS	188	GETHOL	123	MATRA	269	PLURNS	117
ADDK	57	DINT	394	GETRD1	151	MATRA	78	PLYSYN	398
AMPHZ	120	DISPLA	342	GETX	87	MATRA1	359	POKCT1	362
ARBCOL	294	DIVIDE	189	GNFLT1	356	MAXSN	135	POLYDV	175
ARCTAN	41	DLOG	393	GNHOL2	113	MAXSN	242	POLYEV	177
ASPECT	238	DMOD	395	GRAPH	337	MAXSN	79	POLYSN	178
ASPEC2	316	DOTJ	75	GRAPHX	336	MAXSN	208	POWER	209
ATAN	42	DOTJ	278	GRUP2	357	MAXSNM	112	PRBFIT	364
ATAN	122	DOTJ	282	HLADJ	22	MDOT	195	PROB2	366
AVRAGE	182	DOTP	277	HSTPLT	346	MDOT3	272	PROCOR	167
BLKSUM	183	DSIN	396	HVTOIV	100	MEMUSE	125	PSQRT	401

SUBROUTINE LIBRARY PRINCIPAL ENTRIES ALPHABETIZED, WITH ENTRY INDICES

8
Cross-Reference Table for the One-Pass Library

Certain useful tabulations concerning linkage relationships are possible with respect to a complete and self-consistent library such as that given in Section 7 which are not possible with respect to the program set of Section 10. (The program set of Section 10 is incomplete because it excludes FORTRAN System routines, and it is ambiguous from the standpoint of linkage because name duplication exists.)

The linkage environment of an isolated program in a library complex has two directions. In one (upward), the program is called upon by a specific set of higher-level programs; in the other (downward) it requires a certain set of lower-level routines for the performance of its functions. In the present section we present a tabulation of the first layer of higher-level routines which call a given entry. Section 9 gives the complete environment in the other direction (the first layer of lower routines being essentially synonymous with the transfer vector).

The tabulations of the routines are alphabetically ordered by entry names. All entries, both principal and secondary, are included. The terminologies used in connection with names of secondary entries, with names of principal entries which have no secondary entries, and with names of principal entries which do have secondary entries all differ slightly. For secondary entries the format is

A or B is called by C, D, . . .

or

A or B is not called by any programs in this set

where

A is the name of a secondary entry,

B is the name of its associated principal entry,

C, D, . . . is a list of all principal entries which contain the name A in their transfer vectors

For principal entries which have no secondary entries the format is

A is called directly by C, D, . . .

or

A is not called by any programs in this set

where C, D, . . . is as above.

Cross-Reference Table for the One-Pass Library

For principal entries which have secondary entries two statements are made, one concerning the use of the principal entry name itself, in format

A is called directly by C, D, . . .

or

A   A is not called directly by any programs in this set

and the other concerning the secondary entries, in format

is called indirectly by X, Y, . . .

or

is not called directly or indirectly by any programs in this set,

where

X, Y, . . . is a complete list of all principal entry names, each of whose transfer vectors contains one or more of the names of the secondary entries of A.

8

```
(BSR) , OF  (IOS), IS CALLED BY  (BST), (EXEM),  (RER),  (WER), CPYFL2,  FSKIP,
                                 RSKIP, ZEFBCD.

(BST)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(BUF) , OF  (IOB), IS CALLED BY  (TSB).

(CSH)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(DFAD) IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

(DFDP), OF (DFAD), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(DFMP), OF (DFAD), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(DFSB), OF (DFAD), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(EFT)  IS CALLED DIRECTLY BY    OUDATA.

(ETT) , OF  (IOS), IS CALLED BY  (WER), CPYFL2, PACDAT, WRTDAT.

(EXB) , OF  (IOB), IS CALLED BY  (STB),  (TSB).

(EXE) , OF (EXEM), IS CALLED BY  (CSH),  (FPT),  (IOH),  (IOS),  (RER),  (TSB),
                                 (WER),   DEXP,   DLOG,  DSQRT.

(EXEM) IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY   (CSH),  (FPT),  (IOH),  (IOS),  (RER),  (TSB),
                                 (WER),   DEXP,   DLOG,  DSQRT.

(FIL) , OF  (IOH), IS CALLED BY CARIGE,  CLKON, CNTRDB, COLABL, CONTUR,  CSOUT,
                                DADECK, FMTOUT, GNHOL2,  GRAPH, INDATA, LISTNG,
                                MEMUSE,   MOUT, MOUTAI, ONLINE, PLOTVS, PLTVS1,
                                PWMLIV,  RDATA, VECOUT,   VOUT.

(FPT)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(IFDP), OF (IFMP), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(IFMP) IS CALLED DIRECTLY BY    IPLYEV.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

(IOB)  IS CALLED DIRECTLY BY     (STB),  (TSB).
       IS CALLED INDIRECTLY BY   (STB),  (TSB).

(IOH)  IS CALLED DIRECTLY BY     (CSH),  (FPT),  (SCH),  (SPH),  (STH), DISPLA,
                                GENHOL, GNHOL2, INTHOL, ONLINE, REREAD.
       IS CALLED INDIRECTLY BY  CARIGE,  CLKON, CNTRDB, COLABL, CONTUR,  CSOUT,
                                DADECK, FMTOUT, GETRD1, GNHOL2,  GRAPH, INDATA,
                                INTHOL, LISTNG, MEMUSE,   MOUT, MOUTAI, ONLINE,
                                PLOTVS, PLTVS1, PWMLIV,  RDATA, VECOUT,   VOUT.
```

```
(IOS)  IS CALLED DIRECTLY BY     (BST),  (EFT), (EXEM),  (IOB),  (IOH),  (RWT),
                                CPYFL2,  FSKIP, PACDAT,  RSKIP, WRTDAT, ZEFBCD.
       IS CALLED INDIRECTLY BY   (BST),  (CSH),  (EFT), (EXEM),  (RER),  (RWT),
                                 (SCH),  (SPH),  (STB),  (STH),  (TSB),  (WER),
                                CPYFL2,  FSKIP, ONLINE, PACDAT, REREAD,  RSKIP,
                                WRTDAT, ZEFBCD.

(IOU)  IS CALLED DIRECTLY BY     (IOS).

(RCH) , OF  (IOS), IS CALLED BY  (BST),  (CSH),  (RER),  (SCH),  (SPH),  (STB),
                                 (STH),  (TSB),  (WER), CPYFL2, ONLINE, PACDAT,
                                REREAD, WRTDAT, ZEFBCD.

(RDC) , OF  (RER), IS CALLED BY  (TSB), REREAD.

(RDS) , OF  (IOS), IS CALLED BY  (BST),  (CSH),  (RER),  (TSB), CPYFL2,  FSKIP,
                                PACDAT, REREAD,  RSKIP, ZEFBCD.

(RER)  IS CALLED DIRECTLY BY     (TSB), REREAD.
       IS CALLED INDIRECTLY BY   (TSB), REREAD.

(REW) , OF  (IOS), IS CALLED BY  (RWT),  (WER).

(RLR) , OF  (TSB), IS CALLED BY INDATA, LISTNG, SETINO.

(RTN) , OF  (IOH), IS CALLED BY DADECK, GETRD1, INTHOL,  RDATA.

(RWT)  IS CALLED DIRECTLY BY    LISTNG, SETINO, TRMINO.

(SCH)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(SET) , OF  (IOB), IS CALLED BY  (TSB).

(SLI)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(SLO)  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(SPH)  IS CALLED DIRECTLY BY     CLKON, COLABL, CONTUR,  GRAPH, INDATA, LISTNG,
                                ONLINE, PLOTVS, PWMLIV.

(STB)  IS CALLED DIRECTLY BY    OUDATA.
       IS CALLED INDIRECTLY BY  OUDATA.

(STH)  IS CALLED DIRECTLY BY    CARIGE, CNTRDB, COLABL, CONTUR,  CSOUT, DADECK,
                                FMTOUT, INDATA, LISTNG, MEMUSE,   MOUT, MOUTAI,
                                PLOTVS, PLTVS1, PWMLIV,  RDATA, VECOUT,   VOUT.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

(STHD), OF  (STH), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(STHM), OF  (STH), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
(TCO) , OF  (IOS), IS CALLED BY  (BST),  (CSH),  (RER),  (SCH),  (SPH),  (WER),
                                 CPYFL2,  FSKIP, PACDAT, REREAD,  RSKIP, WRTDAT,
                                 ZEFBCD.

(TEF) , OF  (IOS), IS CALLED BY  (BST),  (CSH),  (RER),  FSKIP, REREAD,  RSKIP,
                                 ZEFBCD.

(TES)  IS CALLED DIRECTLY BY      (IOS),  (STB),  (STH),  (WER),  CHAIN,   DUMP,
                                   EXIT, ONLINE.

(TRC) , OF  (IOS), IS CALLED BY  (BST),  (RER),  (WER), CPYFL2,  FSKIP,  RSKIP,
                                 WRTDAT, ZEFBCD.

(TSB)  IS CALLED DIRECTLY BY     INDATA, LISTNG, SETINO.
       IS CALLED INDIRECTLY BY   INDATA, LISTNG, SETINO.

(TSH) , OF REREAD, IS CALLED BY DADECK, GETRD1,  RDATA.

(TSHM), OF REREAD, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

(WEF) , OF  (IOS), IS CALLED BY  (EFT),  (WER), CPYFL2.

(WER)  IS CALLED DIRECTLY BY     (STB),  (STH), ONLINE.
       IS CALLED INDIRECTLY BY   (STB),  (STH), ONLINE.

(WLR) , OF  (STB), IS CALLED BY OUDATA.

(WRS) , OF  (IOS), IS CALLED BY  (SCH),  (SPH),  (STB),  (STH),  (WER), CPYFL2,
                                 ONLINE, WRTDAT.

(WTC) , OF  (WER), IS CALLED BY  (STB),  (STH), ONLINE.

ABSVAL IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ADANL  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

ADANX , OF  ADANL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ADDK   IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  PLANSP.

ADDKS , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

AMPHZ  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

ARBCOL IS CALLED DIRECTLY BY     CONTUR.

ARCTAN IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ARG    , OF LOCATE, IS CALLED BY  RDATA.

ASPECT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ASPEC2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
ATAN   IS CALLED DIRECTLY BY     AMPHZ, ARCTAN.

AVRAGE IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

BLKSUM IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

BOOST  IS CALLED DIRECTLY BY    PLTVS1.
       IS CALLED INDIRECTLY BY  PLTVS1.

CALL  , OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CALL2 , OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CARIGE IS CALLED DIRECTLY BY     CSOUT, CVSOUT,   MOUT, MOUTAI,   VOUT, VRSOUT.

CHAIN  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CHISQR IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CHOOSE IS CALLED DIRECTLY BY    PLANSP.

CHPRTS IS CALLED DIRECTLY BY    COSIS1, XSPECT.
       IS CALLED INDIRECTLY BY  ASPECT.

CHSIGN IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CHUSET, OF  INDEX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CLKON  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CLOCK1 IS CALLED DIRECTLY BY     CLKON.

CMPARL, OF CMPARV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CMPARP IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

CMPARS, OF CMPARP, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CMPARV IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

CMPRA  IS CALLED DIRECTLY BY     RDATA.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

CMPRFL, OF  CMPRA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CNTRDB IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CNTROW IS CALLED DIRECTLY BY    CONTUR.

COLABL IS CALLED DIRECTLY BY    CONTUR.

COLAPS IS CALLED DIRECTLY BY    ASPECT.

CONTUR IS CALLED DIRECTLY BY    CNTRDB.

CONVLV IS CALLED DIRECTLY BY    PLYSYN, POLYSN.
```

```
COS    IS CALLED DIRECTLY BY      AMPHZ, COSTBL, GNFLT1, PLYSYN, POLYSN, SEQSAC.
       IS CALLED INDIRECTLY BY    ADANL,  AMPHZ, COSTBL, SEQSAC.

COSISP, OF   COSP, IS CALLED BY COSIS1, QIFURY, XSPECT.

COSIS1 IS CALLED DIRECTLY BY     PLANSP.

COSP   IS CALLED DIRECTLY BY     ASPECT, COSIS1, FACTOR.
       IS CALLED INDIRECTLY BY   COSIS1, QIFURY, XSPECT.

COSTBL IS CALLED DIRECTLY BY     FACTOR, PLANSP, QFURRY, QIFURY.
       IS CALLED INDIRECTLY BY   PLANSP, QFURRY, QIFURY.

COSTBX, OF COSTBL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CPYFL2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CROSS  IS CALLED DIRECTLY BY      CROST.

CROST  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CRSVM  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CSOUT  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

CUFIT1 IS CALLED DIRECTLY BY     CNTROW.

CVSOUT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DADECK IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DATAN  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

DATAN2, OF  DATAN, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DCOS  , OF   DSIN, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DELTA  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

DERIVA IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DETRM , OF  SIMEQ, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DEXP   IS CALLED DIRECTLY BY     DEXP(3.

DEXP(2, OF IEXP(2, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DEXP(3 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DIFPRS IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

DINT   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DISPLA IS CALLED DIRECTLY BY      GRAPH.

DIVIDE IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
DIVK  , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DIVKS , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DLOG   IS CALLED DIRECTLY BY    DEXP(3.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

DLOG10, OF   DLOG, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DMOD   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DOTJ   IS CALLED DIRECTLY BY      DOTP,  FIRE2, MATML3,  MFACT, RLSPR2.

DOTP   IS CALLED DIRECTLY BY     FIRE2, RLSPR2.

DPRESS, OF  BOOST, IS CALLED BY PLTVS1.

DSIN   IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

DSPFMT IS CALLED DIRECTLY BY     GRAPH.

DSQRT  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

DUBLL , OF  DUBLX, IS CALLED BY ASPECT.

DUBLX  IS CALLED DIRECTLY BY    ASPECT.
       IS CALLED INDIRECTLY BY  ASPECT.

DUMP   IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  (EXEM).

ENDFIL, OF REREAD, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

EOFSET, OF REREAD, IS CALLED BY DADECK.

EXCHVS IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

EXEDMP, OF (EXEM), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

EXIT   IS CALLED DIRECTLY BY     CHAIN,   DUMP, REREAD.

EXP    IS CALLED DIRECTLY BY    CNTRDB, FACTOR, PRBFIT.

EXP(1  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

EXP(2  IS CALLED DIRECTLY BY     GRAPH, MOUTAI, MXRARE,  POWER, PRBFIT.

EXP(3  IS CALLED DIRECTLY BY    KIINT1.

EXPAND IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FACTOR IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FAPSUM IS CALLED DIRECTLY BY    INDATA, LISTNG, OUDATA.

FASCN1 IS CALLED DIRECTLY BY    NXALRM.
```

```
FASCOR, OF PROCOR, IS CALLED BY QACORR, QCNVLV, QXCORR.

FASCR1, OF PROCOR, IS CALLED BY QXCOR1.

FASCUB IS CALLED DIRECTLY BY    CNTROW.

FASEPC, OF PROCOR, IS CALLED BY QCNVLV.

FASEP1, OF PROCOR, IS CALLED BY QXCOR1.

FASTRK IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FDOT   IS CALLED DIRECTLY BY     CROSS,  LSSS1, WLLSFP.
       IS CALLED INDIRECTLY BY   RLSPR,  RLSSR, WLLSFP.

FDOTR , OF   FDOT, IS CALLED BY  RLSPR,  RLSSR, WLLSFP.

FIRE2  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FIXV   IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  MOUTAI, XSQRUT.

FIXVR , OF   FIXV, IS CALLED BY MOUTAI, XSQRUT.

FLDATA, OF FXDATA, IS CALLED BY QACORR, QCNVLV, QXCORR, SPCOR2.

FLOATM IS CALLED DIRECTLY BY     GRAPH.

FLOATV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FMTOUT IS CALLED DIRECTLY BY    CVSOUT.

FNDFMT IS CALLED DIRECTLY BY    FMTOUT, INTHOL, VECOUT.

FRAME  IS CALLED DIRECTLY BY    DISPLA,  GRAPH, GRAPHX.

FRQCT1 IS CALLED DIRECTLY BY    POKCT1.

FRQCT2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FSKIP  IS CALLED DIRECTLY BY    INDATA, LISTNG, SETINO, TRMINO.

FT24   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

FXDATA IS CALLED DIRECTLY BY      PAKN, QACORR, QCNVLV, QXCORR, SPCOR2.
       IS CALLED INDIRECTLY BY  QACORR, QCNVLV, QXCORR, SPCOR2.

GENHOL IS CALLED DIRECTLY BY    COLABL.

GETHOL IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

GETRD1 IS CALLED DIRECTLY BY    SHUFFL.

GETX   IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

GNFLT1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

GNHOL2 IS CALLED DIRECTLY BY MOUTAI.

GRAPH IS CALLED DIRECTLY BY GRAPHX.

GRAPHX IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

GRUP2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

HALVL , OF DUBLX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

HALVX , OF DUBLX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

HLADJ IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
IS CALLED INDIRECTLY BY CSOUT, VOUT.

HRADJ , OF HLADJ, IS CALLED BY CSOUT, VOUT.

HSTPLT IS CALLED DIRECTLY BY GRAPH.

HVTOIV IS CALLED DIRECTLY BY RDATA.

IABS IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ICOS , OF ISIN, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IDERIV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IEXP IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IEXP(2 IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

IFNCTN IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IGETX , OF GETX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IINTGR IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ILOG IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

INDATA IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

INDEX IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

INTGRA IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

INTHOL IS CALLED DIRECTLY BY RDATA.

INTMSB, OF TIMSUB, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

INTOPR IS CALLED DIRECTLY BY ARBCOL, EXPAND.

INTSUM IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

IOER , OF (EXEM), IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IPLYEV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

```
ISIN   IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

ISQRT  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ITOMLI IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

IVTOHV IS CALLED DIRECTLY BY     RDATA.

IXCARG IS CALLED DIRECTLY BY    COSIS1,  FIRE2,  MIPLS, PLANSP, QXCOR1,  RDATA,
                                RLSPR2.

KIINT1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

KOLAPS IS CALLED DIRECTLY BY    PLANSP, XSPECT.

LIMITS IS CALLED DIRECTLY BY    PLANSP, QXCOR1.

LINE   IS CALLED DIRECTLY BY     GRAPH.

LINEH  IS CALLED DIRECTLY BY    HSTPLT.

LINEV  IS CALLED DIRECTLY BY    HSTPLT.

LINTR1 IS CALLED DIRECTLY BY    NOINT1.

LISTNG IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

LOC    IS CALLED DIRECTLY BY    INDATA, OUDATA.

LOCATE IS CALLED DIRECTLY BY    SEVRAL.
       IS CALLED INDIRECTLY BY   RDATA, SEVRAL.

LOG    IS CALLED DIRECTLY BY    CNTRDB, FACTOR,  GRAPH, MOUTAI.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

LOG10 , OF    LOG, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

LSHFT  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

LSLINE IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

LSSS1  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MATINV IS CALLED DIRECTLY BY     MIPLS.

MATML1 IS CALLED DIRECTLY BY      MDOT.

MATML3 IS CALLED DIRECTLY BY     FIRE2,  MDOT3,  MIFLS,  MIPLS,   MISS, RLSPR2.

MATRA  IS CALLED DIRECTLY BY     MIPLS, PLANSP,  ROAR2.

MATRA1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MAXAB , OF  MAXSN, IS CALLED BY FACTOR, NRMVEC.

MAXABM, OF MAXSNM, IS CALLED BY MOUTAI.
```

```
MAXSN  IS CALLED DIRECTLY BY    PLTVS1.
       IS CALLED INDIRECTLY BY  FACTOR, NRMVEC, PLTVS1.

MAXSNM IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  MOUTAI.

MDOT   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MDOT3  IS CALLED DIRECTLY BY     CRSVM,  MIPLS,   MISS.

MEMUSE IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MFACT  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MIFLS  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MINAB , OF  MAXSN, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MINABM, OF MAXSNM, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MINSN , OF  MAXSN, IS CALLED BY PLTVS1.

MINSNM, OF MAXSNM, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MIPLS  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MISS   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MLISCL IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MLI2A6 IS CALLED DIRECTLY BY    PWMLIV.

MONOCK IS CALLED DIRECTLY BY    IFNCTN.

MOUT   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MOUTAI IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MOVE   IS CALLED DIRECTLY BY    MOUTAI, MOVECS, POLYDV, POLYSN, QFURRY, WLLSFP.

MOVECS IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MOVREV IS CALLED DIRECTLY BY    COSIS1,  MIFLS,  MIPLS,   MISS, PLANSP, RLSPR2,
                                 ROAR2.

MPSEQ1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MRVRS  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MSCON1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MULK  , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MULKS , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MULLER IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
MULPLY IS CALLED DIRECTLY BY    MOUTAI, PLTVS1.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

MUVADD IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MVBLOK IS CALLED DIRECTLY BY     GRAPH, INDATA, OUDATA.

MVINAV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MVNSUM IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MVNTIN IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

MVNTNA, OF MVNTIN, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MVSQAV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

MXRARE IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NEXCOS, OF SEQSAC, IS CALLED BY ASPEC2.

NEXSIN, OF SEQSAC, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NMZMG1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NOINT1 IS CALLED DIRECTLY BY    KIINT1.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

NOINT2, OF NOINT1, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NRMVEC IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NTHA   IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

NURINC IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

NXALRM IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ONLINE IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  CARIGE, CNTRDB, COLABL, CONTUR,  CSOUT, DADECK,
                                FMTOUT, INDATA, LISTNG, MEMUSE,   MOUT, MOUTAI,
                                PLOTVS, PLTVS1, PWMLIV,  RDATA, VECOUT,   VOUT.

OUDATA IS CALLED DIRECTLY BY    TRMINO.

PACDAT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PAKN   IS CALLED DIRECTLY BY    OUDATA.

PDUMP , OF   DUMP, IS CALLED BY (EXEM).

PLANSP IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PLOTVS IS CALLED DIRECTLY BY    PLTVS1.

PLTVS1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
PLURAL, OF SEVRAL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PLURNS IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PLYSYN IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

POKCT1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

POLYDV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

POLYEV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

POLYSN IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

POWER  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

PRBFIT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PROB2  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PROCOR IS CALLED DIRECTLY BY    QACORR, QCNVLV, QXCORR, QXCOR1.
       IS CALLED INDIRECTLY BY  QACORR, QCNVLV, QXCORR, QXCOR1.

PSQRT  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

PWMLIV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QACORR IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QCNVLV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QFURRY IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QIFURY IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QINTR1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QUFIT1 IS CALLED DIRECTLY BY    CNTROW, QINTR1.

QXCORR IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

QXCOR1 IS CALLED DIRECTLY BY    SPCOR2.

RDATA  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

REFIT , OF  SPLIT, IS CALLED BY XSPECT.

REFLEC IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

REIM  , OF  AMPHZ, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

REMAV  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

REREAD IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
       IS CALLED INDIRECTLY BY  DADECK, GETRD1,  RDATA.
```

```
RETURN, OF LOCATE, IS CALLED BY  RDATA.

REVER  IS CALLED DIRECTLY BY    FNDFMT, IFNCTN.

REVERS IS CALLED DIRECTLY BY     CROST, GETHOL,  MRVRS, QXCOR1,  ROAR2.

RLSPR  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RLSPR2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RLSSR  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RMSDAV, OF RMSDEV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RMSDEV IS CALLED DIRECTLY BY    PLTVS1.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

RND    IS CALLED DIRECTLY BY     AMPHZ, CNTROW, MOUTAI, PLOTVS,   RNDV.
       IS CALLED INDIRECTLY BY  CNTROW, CONTUR, QINTR1,   RNDV.

RNDDN , OF    RND, IS CALLED BY CNTROW, CONTUR,   RNDV.

RNDUP , OF    RND, IS CALLED BY CNTROW, CONTUR, QINTR1,   RNDV.

RNDV   IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

RNDVDN, OF   RNDV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RNDVUP, OF   RNDV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ROAR2  IS CALLED DIRECTLY BY    PLANSP.

ROTAT1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

RPLFMT IS CALLED DIRECTLY BY    FMTOUT, VECOUT.

RSKIP  IS CALLED DIRECTLY BY    DADECK.

RVPRTS, OF CHPRTS, IS CALLED BY ASPECT.

SAME   IS CALLED DIRECTLY BY    CNTRDB, LISTNG, MOUTAI.
       IS CALLED INDIRECTLY BY  CONTUR, INDATA, LISTNG, PLTVS1.

SCPSCL IS CALLED DIRECTLY BY     GRAPH.

SEARCH IS CALLED DIRECTLY BY    SHUFFL.

SEQSAC IS CALLED DIRECTLY BY    ASPEC2.
       IS CALLED INDIRECTLY BY  ASPEC2.

SETAPT, OF  INDEX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SETEST, OF  INDEX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SETINO IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
SETK   IS CALLED DIRECTLY BY     SETKP.
       IS CALLED INDIRECTLY BY  CNTRDB,  CRSVM, PLANSP, PLOTVS, PLTVS1, QXCOR1,
                                  SETKP.

SETKP  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SETKS , OF   SETK, IS CALLED BY  CRSVM, PLANSP, PLOTVS, PLTVS1, QXCOR1.

SETKV  IS CALLED DIRECTLY BY    PLOTVS.

SETKVS IS CALLED DIRECTLY BY    PLTVS1.

SETLIN IS CALLED DIRECTLY BY    SETLNS.
       IS CALLED INDIRECTLY BY  PLTVS1, SETLNS.

SETLNS IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SETSBV, OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SETUP , OF LOCATE, IS CALLED BY  RDATA.

SETVCP, OF  SETKP, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SETVEC, OF   SETK, IS CALLED BY CNTRDB, PLOTVS, PLTVS1,  SETKP.

SEVRAL IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SHFTR1 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SHFTR2 IS CALLED DIRECTLY BY    LISTNG.

SHUFFL IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SIFT   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SIMEQ  IS CALLED DIRECTLY BY    MATINV, RLSPR2.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SIN    , OF    COS, IS CALLED BY  ADANL,  AMPHZ, COSTBL, SEQSAC.

SINTBL, OF COSTBL, IS CALLED BY PLANSP, QFURRY, QIFURY.

SINTBX, OF COSTBL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SISP  , OF   COSP, IS CALLED BY COSIS1.

SIZEUP IS CALLED DIRECTLY BY    SHUFFL.
       IS NOT CALLED INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SIZUPL, OF SIZEUP, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SMPRDV, OF  POWER, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SMPSON IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SPCOR2 IS NOT CALLED BY ANY PROGRAMS IN THIS SET.
```

```
SPLIT  IS CALLED DIRECTLY BY    ASPECT, COSIS1, XSPECT.
       IS CALLED INDIRECTLY BY  XSPECT.

SQRDEV, OF SQRDFR, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SQRDFR IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SQRMLI IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SQROOT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SQRSUM IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SQRT   IS CALLED DIRECTLY BY     AMPHZ,   IABS, KIINT1,  MFACT, MULLER, NRMVEC,
                                POLYSN, PRBFIT,  PSQRT, RMSDEV, SQROOT, XSQRUT.

SQUARE IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SRCH1  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

STEPC , OF  DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

STEPL , OF  DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

STEPR , OF  DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

STORE , OF LOCATE, IS CALLED BY  RDATA.

STZ    IS CALLED DIRECTLY BY     CROSS,  CRSVM,  FIRE2,  MFACT,  MIPLS, PLANSP,
                                POLYDV, QFURRY, QXCOR1, RLSPR2, SPCOR2.

STZS   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SUBK   , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SUBKS , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SUM    IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SUMDEV, OF SUMDFR, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

SUMDFR IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

SWITCH IS CALLED DIRECTLY BY    CONTUR, PLOTVS.

TAMVL  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

TAMVR , OF  TAMVL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

TANH   IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

TIMA2B IS CALLED DIRECTLY BY    TIMSUB.

TIMSUB IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

TINGL  IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.
```

TINGLA, OF TINGL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

TRMINO IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

UNPAKN IS CALLED DIRECTLY BY INDATA.

VARARG IS CALLED DIRECTLY BY INDATA, OUDATA, PLTVS1.

VDOTV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VDVBYV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VECOUT IS CALLED DIRECTLY BY CVSOUT, VOUT, VRSOUT.

VINDEX, OF INDEX, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VMNUSV, OF VPLUSV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VOUT IS CALLED DIRECTLY BY VSOUT.

VPLUSV IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

VRSOUT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VSOUT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

VTIMSV IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

WAC IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

WHERE , OF LOCATE, IS CALLED BY SEVRAL.

WHICH IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

WLLSFP IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

WRTDAT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XACTEQ IS CALLED DIRECTLY BY SRCH1.

XADDK , OF ADDK, IS CALLED BY PLANSP.

XADDKS, OF ADDK, IS CALLED BY PLANSP.

XARG , OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XAVRGE IS NOT CALLED DIRECTLY BY ANY PROGRAMS IN THIS SET.
IS CALLED INDIRECTLY BY XREMAV.

XAVRGR, OF XAVRGE, IS CALLED BY XREMAV.

XBOOST, OF BOOST, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XCMPRA, OF CMPRA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDANL , OF ADANL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

```
XDANX , OF  ADANL, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDELTA, OF  DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDFPRS, OF DIFPRS, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDIV   IS CALLED DIRECTLY BY    XAVRGE, XDVIDE, XVDVBV.
       IS CALLED INDIRECTLY BY  XAVRGE, XDVIDE, XVDVBV.

XDIVK , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDIVKS, OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDIVR , OF   XDIV, IS CALLED BY XAVRGE, XDVIDE, XVDVBV.

XDPRSS, OF  BOOST, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDVIDE IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

XDVIDR, OF XDVIDE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDVRK , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XDVRKS, OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XFIXM  IS CALLED DIRECTLY BY     GRAPH.

XINDEX, OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XLCOMN IS CALLED DIRECTLY BY    MEMUSE.

XLIMIT IS CALLED DIRECTLY BY    SETINO, TRMINO.

XLOC   IS CALLED DIRECTLY BY    GETHOL,  GRAPH, IXCARG, PLTVS1, QCNVLV, QIFURY,
                                QXCORR, SPCOR2, XSPECT.

XLOCV  IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XLSHFT, OF  LSHFT, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XMLPLY, OF MULPLY, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XMULK , OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XMULKS, OF   ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XNAME , OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XNARGS, OF LOCATE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XNTHA , OF   NTHA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XNTSUM, OF INTSUM, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XOOZE  IS CALLED DIRECTLY BY    PLANSP.
```

XREMAV IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XRFLEC, OF REFLEC, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSAME , OF SAME, IS CALLED BY CONTUR, INDATA, LISTNG, PLTVS1.

XSMDEV, OF SUMDFR, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSMDFR, OF SUMDFR, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSPECT IS CALLED DIRECTLY BY QFURRY.

XSQDEV, OF XSQDFR, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSQDFR IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

XSQRUT IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSQSUM, OF SQRSUM, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSQUAR, OF SQUARE, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSTEPC, OF DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSTEPL, OF DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSTEPR, OF DELTA, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSTLIN, OF SETLIN, IS CALLED BY PLTVS1, SETLNS.

XSUBK , OF ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSUBKS, OF ADDK, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XSUM , OF SUM, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XVDRBV, OF XVDVBV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XVDVBV IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

XVMNSV, OF VPLUSV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XVPLSV, OF VPLUSV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XVTMSV, OF VTIMSV, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

XWHICH, OF WHICH, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

ZEFBCD IS NOT CALLED DIRECTLY OR INDIRECTLY BY ANY PROGRAMS IN THIS SET.

ZEFBIN, OF ZEFBCD, IS NOT CALLED BY ANY PROGRAMS IN THIS SET.

9
Subroutine Rosters For the One-Pass Library

The cross-reference table of the preceding section is principally of value in work on the library itself, since (in conjunction with the tables of Section 7) one can use it to determine the effects of repositioning or deleting individual programs. Of greater utility to the working programmer are data on the lower-level environment of a given program. The "subroutine rosters" of the present section provide this type of data for the subroutine library defined in Section 7.

We define the subroutine roster of a given program as a list of all programs needed to make the given program operative. For a programmer who does not use a library tape but has access to a drawer of binary decks, the roster is a list of all the additional binary decks he must collect to make possible the execution of a given program. In any event, it is desirable that (1) such lists be expressed in terms of principal entry names, and that (2) each such list be subdivided into FORTRAN System routines and non-FORTRAN-System routines. [Note that because of (1) the names in the roster may not coincide with the names in the transfer vectors of the program and in its lower-level programs.]

The roster tables that follow are designed to fill these two prescriptions. In addition, they give memory-storage requirements (in decimal) of the given program, of each principal entry in its roster, and of the whole set. The tables are organized alphabetically by all entry names in the library (for secondary entries no rosters are given, only references to their principal entries). The phrase

NEEDS FSRS -

is used to introduce a subroster of FORTRAN System routines, and the phrase

NEEDS SRS -

is used to head a subroster of subroutines (or functions) not in the FORTRAN System. The absence of a transfer vector is denoted by the expression

NEEDS NO LOWER ROUTINES

```
(BSR)   (SECONDARY ENTRY OF (IOS) )

(BST)                                                  PROGRAM PROPER....   28
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ...........................   764
                                                        STORAGE TOTAL....  792

(BUF)   (SECONDARY ENTRY OF (IOB) )

(CSH)                                                  PROGRAM PROPER....  125
  NEEDS FSRS -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (TES)    1,   DUMP  177,   EXIT   17 ................ 1780
                                                        STORAGE TOTAL.... 1905

(DFAD)  NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   80

(DFDP)  (SECONDARY ENTRY OF (DFAD))

(DFMP)  (SECONDARY ENTRY OF (DFAD))

(DFSB)  (SECONDARY ENTRY OF (DFAD))

(EFT)                                                  PROGRAM PROPER....    7
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ...........................   764
                                                        STORAGE TOTAL....  771

(ETT)   (SECONDARY ENTRY OF (IOS) )

(EXB)   (SECONDARY ENTRY OF (IOB) )

(EXE)   (SECONDARY ENTRY OF (EXEM))

(EXEM)                                                 PROGRAM PROPER....  458
  NEEDS FSRS -      (IOS)   87,  (IOU)   24,  (TES)    1,   DUMP  177,
                     EXIT   17 ......................................   306
                                                        STORAGE TOTAL....  764

(FIL)   (SECONDARY ENTRY OF (IOH) )

(FPT)                                                  PROGRAM PROPER....   41
  NEEDS FSRS -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (TES)    1,   DUMP  177,   EXIT   17 ................ 1780
                                                        STORAGE TOTAL.... 1821

(IFDP)  (SECONDARY ENTRY OF (IFMP))

(IFMP)  NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....  136

(IOB)                                                  PROGRAM PROPER....  570
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ...........................   764
                                                        STORAGE TOTAL.... 1334
```

```
************************       SUBROUTINE ROSTERS       ************************
*  (IOH)   TO  (STB)   *                                *  (IOH)   TO  (STB)   *
************************                                ************************


(IOH)                                                     PROGRAM PROPER.... 1016
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ................................  764
                                                          STORAGE TOTAL.... 1780

(IOS)                                                     PROGRAM PROPER....   87
  NEEDS FSRS -     (EXEM)  458,  (IOU)   24,  (TES)    1,  DUMP  177,
                     EXIT   17 ..............................................  677
                                                          STORAGE TOTAL....  764

(IOU)    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   24

(RCH)    (SECONDARY ENTRY OF (IOS) )

(RDC)    (SECONDARY ENTRY OF (RER) )

(RDS)    (SECONDARY ENTRY OF (IOS) )

(RER)                                                     PROGRAM PROPER....   37
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ................................  764
                                                          STORAGE TOTAL....  801

(REW)    (SECONDARY ENTRY OF (IOS) )

(RLR)    (SECONDARY ENTRY OF (TSB) )

(RTN)    (SECONDARY ENTRY OF (IOH) )

(RWT)                                                     PROGRAM PROPER....    7
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ................................  764
                                                          STORAGE TOTAL....  771

(SCH)                                                     PROGRAM PROPER....   96
  NEEDS FSRS -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (TES)    1,   DUMP  177,   EXIT   17 .................. 1780
                                                          STORAGE TOTAL.... 1876

(SET)    (SECONDARY ENTRY OF (IOB) )

(SLI)    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   13

(SLO)    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   13

(SPH)                                                     PROGRAM PROPER....  183
  NEEDS FSRS -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (TES)    1,   DUMP  177,   EXIT   17 .................. 1780
                                                          STORAGE TOTAL.... 1963

(STB)                                                     PROGRAM PROPER....   53
  NEEDS FSRS -     (EXEM)  458,  (IOB)  570,  (IOS)   87,  (IOU)   24,
                    (TES)    1,  (WER)   57,   DUMP  177,   EXIT   17 .... 1391
                                                          STORAGE TOTAL.... 1444
```

```
************************        SUBROUTINE ROSTERS        ************************
*  (STH)   TO  AMPHZ   *                                   *  (STH)   TO  AMPHZ   *
************************                                   ************************


(STH)                                                      PROGRAM PROPER....   83
  NEEDS FSRS -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (TES)    1,  (WER)   57,   DUMP  177,   EXIT   17 .... 1837
                                                            STORAGE TOTAL.... 1920

(STHD)  (SECONDARY ENTRY OF (STH) )

(STHM)  (SECONDARY ENTRY OF (STH) )

(TCO)   (SECONDARY ENTRY OF (IOS) )

(TEF)   (SECONDARY ENTRY OF (IOS) )

(TES)   NEEDS NO LOWER ROUTINES                             STORAGE TOTAL....    1

(TRC)   (SECONDARY ENTRY OF (IOS) )

(TSB)                                                      PROGRAM PROPER....   66
  NEEDS FSRS -     (EXEM)  458,  (IOB)  570,  (IOS)   87,  (IOU)   24,
                    (RER)   37,  (TES)    1,   DUMP  177,   EXIT   17 .... 1371
                                                            STORAGE TOTAL.... 1437

(TSH)   (SECONDARY ENTRY OF REREAD)

(TSHM)  (SECONDARY ENTRY OF REREAD)

(WEF)   (SECONDARY ENTRY OF (IOS) )

(WER)                                                      PROGRAM PROPER....   57
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 ...........................    764
                                                            STORAGE TOTAL....  821

(WLR)   (SECONDARY ENTRY OF (STB) )

(WRS)   (SECONDARY ENTRY OF (IOS) )

(WTC)   (SECONDARY ENTRY OF (WER) )

ABSVAL  NEEDS NO LOWER ROUTINES                             STORAGE TOTAL....   50

ADANL                                                      PROGRAM PROPER....  183
  NEEDS FSRS -       COS  105 .......................................    105
                                                            STORAGE TOTAL....  288

ADANX   (SECONDARY ENTRY OF ADANL )

ADDK    NEEDS NO LOWER ROUTINES                             STORAGE TOTAL....  114

ADDKS   (SECONDARY ENTRY OF ADDK  )

AMPHZ                                                      PROGRAM PROPER....  149
  NEEDS SRS  -       RND   15 .......................................     15
  AND  FSRS  -      ATAN   77,    COS  105,    SQRT   44 ...............    226
                                                            STORAGE TOTAL....  390
```

```
ARBCOL                                                  PROGRAM PROPER....  129
  NEEDS SRS  -    INTOPR  111 ..........................................  111
                                                         STORAGE TOTAL....  240

ARCTAN                                                  PROGRAM PROPER....   29
  NEEDS FSRS -      ATAN   77 ..........................................   77
                                                         STORAGE TOTAL....  106

ARG     (SECONDARY ENTRY OF LOCATE)

ASPECT                                                  PROGRAM PROPER....  278
  NEEDS SRS  -    CHPRTS   76, COLAPS   50,   COSP  504,  DUBLX   45,
                   SPLIT  224 ..........................................  899
                                                         STORAGE TOTAL.... 1177

ASPEC2                                                  PROGRAM PROPER....   74
  NEEDS SRS  -    SEQSAC   94 ..........................................   94
  AND  FSRS  -       COS  105 ..........................................  105
                                                         STORAGE TOTAL....  273

ATAN    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   77

AVRAGE  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   24

BLKSUM  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   49

BOOST   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   34

CALL    (SECONDARY ENTRY OF LOCATE)

CALL2   (SECONDARY ENTRY OF LOCATE)

CARIGE                                                  PROGRAM PROPER....   47
  NEEDS SRS  -    ONLINE  134 ..........................................  134
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 .......................................... 2020
                                                         STORAGE TOTAL.... 2201

CHAIN                                                   PROGRAM PROPER....  179
  NEEDS FSRS -     (TES)    1,   EXIT   17 .............................   18
                                                         STORAGE TOTAL....  197

CHISQR  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  105

CHOOSE  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   17

CHPRTS  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   76

CHSIGN  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   18

CHUSET  (SECONDARY ENTRY OF INDEX )
```

```
CLKON                                                    PROGRAM PROPER....   46
  NEEDS SRS  -    CLOCK1   57 ...........................................   57
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,   DUMP  177,   EXIT   17 .... 1963
                                                          STORAGE TOTAL.... 2066

CLOCK1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   57

CMPARL  (SECONDARY ENTRY OF CMPARV)

CMPARP  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   53

CMPARS  (SECONDARY ENTRY OF CMPARP)

CMPARV  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   50

CMPRA   NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   18

CMPRFL  (SECONDARY ENTRY OF CMPRA )

CNTRDB                                                   PROGRAM PROPER....  550
  NEEDS SRS  -    ARBCOL  129, CNTROW  802, COLABL  185, CONTUR  587,
                  CUFIT1  158, FASCUB  141, GENHOL   48, INTOPR  111,
                  ONLINE  134, QUFIT1   79,    RND   15,   SAME    1,
                    SETK   37, SWITCH   15 ............................. 2442
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17,    EXP   52,    LOG   53 ............... 2125
                                                          STORAGE TOTAL.... 5117

CNTROW                                                   PROGRAM PROPER....  802
  NEEDS SRS  -    CUFIT1  158, FASCUB  141, QUFIT1   79,    RND   15 ....  393
                                                          STORAGE TOTAL.... 1195

COLABL                                                   PROGRAM PROPER....  185
  NEEDS SRS  -    GENHOL   48, ONLINE  134 .............................  182
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ........................................... 2020
                                                          STORAGE TOTAL.... 2387

COLAPS  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   50

CONTUR                                                   PROGRAM PROPER....  587
  NEEDS SRS  -    ARBCOL  129, CNTROW  802, COLABL  185, CUFIT1  158,
                  FASCUB  141, GENHOL   48, INTOPR  111, ONLINE  134,
                  QUFIT1   79,    RND   15,   SAME    1, SWITCH   15 .... 1818
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ........................................... 2020
                                                          STORAGE TOTAL.... 4425

CONVLV  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   56

COS     NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  105
```

```
COSISP  (SECONDARY ENTRY OF COSP  )

COSIS1                                                  PROGRAM PROPER....  406
  NEEDS SRS  -     CHPRTS   76,   COSP  504, IXCARG   35, MOVREV   74,
                    SPLIT  224 ..........................................  913
  AND  FSRS  -      XLOC   12 ..........................................   12
                                                         STORAGE TOTAL.... 1331

COSP    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  504

COSTBL                                                  PROGRAM PROPER....  121
  NEEDS FSRS -        COS  105 ..........................................  105
                                                         STORAGE TOTAL....  226

COSTBX  (SECONDARY ENTRY OF COSTBL)

CPYFL2                                                  PROGRAM PROPER....  178
  NEEDS FSRS -     (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                     DUMP  177,   EXIT   17 .............................  764
                                                         STORAGE TOTAL....  942

CROSS                                                   PROGRAM PROPER....  107
  NEEDS SRS  -       FDOT   40,    STZ   14 .............................   54
                                                         STORAGE TOTAL....  161

CROST                                                   PROGRAM PROPER....  134
  NEEDS SRS  -      CROSS  107,   FDOT   40, REVERS   29,    STZ   14 ....  190
                                                         STORAGE TOTAL....  324

CRSVM                                                   PROGRAM PROPER....  327
  NEEDS SRS  -       DOTJ   59, MATML3  120,  MDOT3  122,   SETK   37,
                      STZ   14 ..........................................  352
                                                         STORAGE TOTAL....  679

CSOUT                                                   PROGRAM PROPER....   49
  NEEDS SRS  -     CARIGE   47,  HLADJ   46, ONLINE  134 ................  227
  AND  FSRS  -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                     EXIT   17 .......................................... 2020
                                                         STORAGE TOTAL.... 2296

CUFIT1  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  158

CVSOUT                                                  PROGRAM PROPER....   84
  NEEDS SRS  -     CARIGE   47, FMTOUT   51, FNDFMT   88, ONLINE  134,
                    REVER   30, RPLFMT   17, VECOUT   66 ................  433
  AND  FSRS  -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                     EXIT   17 .......................................... 2020
                                                         STORAGE TOTAL.... 2537
```

```
DOTJ    NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   59

DOTP                                                            PROGRAM PROPER....  264
  NEEDS SRS  -      DOTJ  59 ........................................    59
                                                                 STORAGE TOTAL....  323

DPRESS  (SECONDARY ENTRY OF BOOST )

DSIN    NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....  222

DSPFMT  NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....  194

DSQRT                                                           PROGRAM PROPER....   66
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 ..............................   764
                                                                 STORAGE TOTAL....  830

DUBLL   (SECONDARY ENTRY OF DUBLX )

DUBLX   NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   45

DUMP                                                            PROGRAM PROPER....  177
  NEEDS FSRS -     (TES)    1,   EXIT   17 ..............................    18
                                                                 STORAGE TOTAL....  195

ENDFIL  (SECONDARY ENTRY OF REREAD)

EOFSET  (SECONDARY ENTRY OF REREAD)

EXCHVS  NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   22

EXEDMP  (SECONDARY ENTRY OF (EXEM))

EXIT                                                            PROGRAM PROPER....   17
  NEEDS FSRS -     (TES)    1 ........................................     1
                                                                 STORAGE TOTAL....   18

EXP     NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   52

EXP(1   NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   35

EXP(2   NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   38

EXP(3   NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   93

EXPAND                                                          PROGRAM PROPER....  189
  NEEDS SRS  -    INTOPR  111 ........................................   111
                                                                 STORAGE TOTAL....  300

FACTOR                                                          PROGRAM PROPER....  308
  NEEDS SRS  -      COSP  504, COSTBL  121,  MAXSN   54 ................   679
  AND  FSRS  -       COS  105,    EXP   52,    LOG   53 ................   210
                                                                 STORAGE TOTAL.... 1197

FAPSUM  NEEDS NO LOWER ROUTINES                                  STORAGE TOTAL....   14
```

```
DADECK                                                  PROGRAM PROPER....  100
  NEEDS SRS  -    ONLINE  134, REREAD  114,  RSKIP   37 ................  285
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (RER)   37,  (SPH)  183,  (TES)    1,  (WER)   57,
                    DUMP  177,   EXIT   17 ............................. 2057
                                                        STORAGE TOTAL.... 2442

DATAN   NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....  440

DATAN2  (SECONDARY ENTRY OF DATAN )

DCOS    (SECONDARY ENTRY OF DSIN  )

DELTA   NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   17

DERIVA  NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   61

DETRM   (SECONDARY ENTRY OF SIMEQ )

DEXP                                                    PROGRAM PROPER....  153
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 .............................  764
                                                        STORAGE TOTAL....  917

DEXP(2  (SECONDARY ENTRY OF IEXP(2)

DEXP(3                                                  PROGRAM PROPER....   34
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DEXP  153,   DLOG  273,   DUMP  177,   EXIT   17 .... 1190
                                                        STORAGE TOTAL.... 1224

DIFPRS  NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   30

DINT    NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   10

DISPLA                                                  PROGRAM PROPER....  219
  NEEDS SRS  -     FRAME    9 ......................................    9
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (TES)    1,   DUMP  177,   EXIT   17 ................ 1780
                                                        STORAGE TOTAL.... 2008

DIVIDE  NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   23

DIVK    (SECONDARY ENTRY OF ADDK  )

DIVKS   (SECONDARY ENTRY OF ADDK  )

DLOG                                                    PROGRAM PROPER....  273
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 .............................  764
                                                        STORAGE TOTAL.... 1037

DLOG10  (SECONDARY ENTRY OF DLOG  )

DMOD    NEEDS NO LOWER ROUTINES                         STORAGE TOTAL....   48
```

```
FASCN1  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  107

FASCOR  (SECONDARY ENTRY OF PROCOR)

FASCR1  (SECONDARY ENTRY OF PROCOR)

FASCUB  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  141

FASEPC  (SECONDARY ENTRY OF PROCOR)

FASEP1  (SECONDARY ENTRY OF PROCOR)

FASTRK  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   26

FDOT    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   40

FDOTR   (SECONDARY ENTRY OF FDOT  )

FIRE2                                                   PROGRAM PROPER....  271
  NEEDS SRS  -     DOTJ   59,  DOTP  264, IXCARG   35, MATML3  120,
                    STZ   14 .........................................  492
  AND  FSRS  -     XLOC   12 .........................................   12
                                                         STORAGE TOTAL....  775

FIXV    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   35

FIXVR   (SECONDARY ENTRY OF FIXV  )

FLDATA  (SECONDARY ENTRY OF FXDATA)

FLOATM  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   25

FLOATV  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   22

FMTOUT                                                  PROGRAM PROPER....   51
  NEEDS SRS  -   FNDFMT   88, ONLINE  134,  REVER   30, RPLFMT   17 ....  269
  AND  FSRS  -   (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                  (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                   EXIT   17 ......................................... 2020
                                                         STORAGE TOTAL.... 2340

FNDFMT                                                  PROGRAM PROPER....   88
  NEEDS SRS  -    REVER   30 .........................................   30
                                                         STORAGE TOTAL....  118

FRAME   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....    9

FRQCT1  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  117

FRQCT2  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  117

FSKIP                                                   PROGRAM PROPER....   50
  NEEDS FSRS -   (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                   DUMP  177,   EXIT   17 ............................  764
                                                         STORAGE TOTAL....  814
```

```
FT24    NEEDS NO LOWER ROUTINES                               STORAGE TOTAL....  818

FXDATA  NEEDS NO LOWER ROUTINES                               STORAGE TOTAL....  102

GENHOL                                                       PROGRAM PROPER....   48
  NEEDS FSRS -      (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                     (TES)    1,   DUMP  177,   EXIT   17 .................. 1780
                                                              STORAGE TOTAL.... 1828

GETHOL                                                       PROGRAM PROPER....  169
  NEEDS SRS   -     REVERS   29 ..........................................   29
  AND  FSRS   -       XLOC   12 ..........................................   12
                                                              STORAGE TOTAL....  210

GETRD1                                                       PROGRAM PROPER....  229
  NEEDS SRS   -     REREAD  114 ..........................................  114
  AND  FSRS   -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                     (RER)   37,  (TES)    1,   DUMP  177,   EXIT   17 .... 1817
                                                              STORAGE TOTAL.... 2160

GETX    NEEDS NO LOWER ROUTINES                               STORAGE TOTAL....   31

GNFLT1                                                       PROGRAM PROPER....  232
  NEEDS FSRS -         COS  105 ..........................................  105
                                                              STORAGE TOTAL....  337

GNHOL2                                                       PROGRAM PROPER....   74
  NEEDS FSRS -      (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                     (TES)    1,   DUMP  177,   EXIT   17 .................. 1780
                                                              STORAGE TOTAL.... 1854

GRAPH                                                        PROGRAM PROPER.... 1499
  NEEDS SRS   -     DISPLA  219, DSPFMT  194, FLOATM   25,  FRAME    9,
                    HSTPLT  145,   LINE   95,  LINEH   35,  LINEV   35,
                    MVBLOK   19, SCPSCL   33,  XFIXM   31 ..................  840
  AND  FSRS   -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                     (SPH)  183,  (TES)    1,   DUMP  177,   EXIT   17,
                     EXP(2   38,    LOG   53,   XLOC   12 .................. 2066
                                                              STORAGE TOTAL.... 4405

GRAPHX                                                       PROGRAM PROPER....  123
  NEEDS SRS   -     DISPLA  219, DSPFMT  194, FLOATM   25,  FRAME    9,
                     GRAPH 1499, HSTPLT  145,   LINE   95,  LINEH   35,
                     LINEV   35, MVBLOK   19, SCPSCL   33,  XFIXM   31 .... 2339
  AND  FSRS   -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                     (SPH)  183,  (TES)    1,   DUMP  177,   EXIT   17,
                     EXP(2   38,    LOG   53,   XLOC   12 .................. 2066
                                                              STORAGE TOTAL.... 4528

GRUP2   NEEDS NO LOWER ROUTINES                               STORAGE TOTAL....  201

HALVL   (SECONDARY ENTRY OF DUBLX )

HALVX   (SECONDARY ENTRY OF DUBLX )

HLADJ   NEEDS NO LOWER ROUTINES                               STORAGE TOTAL....   46
```

```
HRADJ   (SECONDARY ENTRY OF HLADJ )

HSTPLT                                                  PROGRAM PROPER....  145
  NEEDS SRS  -     LINEH   35,  LINEV   35 ..............................    70
                                                         STORAGE TOTAL....  215

HVTOIV  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   39

IABS                                                    PROGRAM PROPER....   21
  NEEDS FSRS -      SQRT   44 ...........................................    44
                                                         STORAGE TOTAL....   65

ICOS    (SECONDARY ENTRY OF ISIN  )

IDERIV  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   54

IEXP    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  157

IEXP(2  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  161

IFNCTN                                                  PROGRAM PROPER....  208
  NEEDS SRS  -    MONOCK   48,  REVER   30 ..............................    78
                                                         STORAGE TOTAL....  286

IGETX   (SECONDARY ENTRY OF GETX  )

IINTGR  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   49

ILOG    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  190

INDATA                                                  PROGRAM PROPER....  896
  NEEDS SRS  -    FAPSUM   14,  FSKIP   50,    LOC    4, MVBLOK   19,
                  ONLINE  134,   SAME    1, UNPAKN   78, VARARG   44 ....   344
  AND  FSRS  -     (EXEM)  458,  (IOB)  570,  (IOH) 1016,  (IOS)   87,
                   (IOU)   24,  (RER)   37,  (SPH)  183,  (TES)    1,
                   (TSB)   66,  (WER)   57,   DUMP  177,   EXIT   17 .... 2693
                                                         STORAGE TOTAL.... 3933

INDEX   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   50

INTGRA  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   47

INTHOL                                                  PROGRAM PROPER....   72
  NEEDS SRS  -    FNDFMT   88,  REVER   30 ..............................   118
  AND  FSRS  -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (TES)    1,   DUMP  177,   EXIT   17 ................ 1780
                                                         STORAGE TOTAL.... 1970

INTMSB  (SECONDARY ENTRY OF TIMSUB)

INTOPR  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  111

INTSUM  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   27

IOER    (SECONDARY ENTRY OF (EXEM))
```

```
IPLYEV                                                  PROGRAM PROPER....   98
  NEEDS FSRS -    (IFMP)  136 ..........................................   136
                                                         STORAGE TOTAL....  234

ISIN    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  184

ISQRT   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   88

ITOMLI  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   37

IVTOHV  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   70

IXCARG                                                  PROGRAM PROPER....   35
  NEEDS FSRS -      XLOC   12 ..........................................    12
                                                         STORAGE TOTAL....   47

KIINT1                                                  PROGRAM PROPER....  191
  NEEDS SRS  -    LINTR1   96, NOINT1  369 .............................   465
  AND  FSRS  -     EXP(3   93,   SQRT   44 .............................   137
                                                         STORAGE TOTAL....  793

KOLAPS  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  100

LIMITS  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   44

LINE    NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   95

LINEH   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   35

LINEV   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   35

LINTR1  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   96

LISTNG                                                  PROGRAM PROPER....  755
  NEEDS SRS  -    FAPSUM   14,  FSKIP   50, ONLINE  134,   SAME    1,
                  SHFTR2   72 ..........................................   271
  AND  FSRS  -    (EXEM)  458,  (IOB)  570,  (IOH) 1016,  (IOS)   87,
                   (IOU)   24,  (RER)   37,  (RWT)    7,  (SPH)  183,
                   (TES)    1,  (TSB)   66,  (WER)   57,   DUMP  177,
                    EXIT   17 ..........................................  2700
                                                         STORAGE TOTAL.... 3726

LOC     NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....    4

LOCATE  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  512

LOG     NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   53

LOG10   (SECONDARY ENTRY OF LOG    )

LSHFT   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   12

LSLINE  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....  117
```

```
LSSS1                                                    PROGRAM PROPER....  122
  NEEDS SRS  -      FDOT   40 ..........................................     40
                                                          STORAGE TOTAL....  162

MATINV                                                   PROGRAM PROPER....   90
  NEEDS SRS  -     SIMEQ  441 ..........................................    441
                                                          STORAGE TOTAL....  531

MATML1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   61

MATML3                                                   PROGRAM PROPER....  120
  NEEDS SRS  -      DOTJ   59 ..........................................     59
                                                          STORAGE TOTAL....  179

MATRA   NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   92

MATRA1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   42

MAXAB   (SECONDARY ENTRY OF MAXSN )

MAXABM  (SECONDARY ENTRY OF MAXSNM)

MAXSN   NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   54

MAXSNM  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   61

MDOT                                                     PROGRAM PROPER....  109
  NEEDS SRS  -    MATML1   61 ..........................................     61
                                                          STORAGE TOTAL....  170

MDOT3                                                    PROGRAM PROPER....  122
  NEEDS SRS  -      DOTJ   59, MATML3  120 .............................    179
                                                          STORAGE TOTAL....  301

MEMUSE                                                   PROGRAM PROPER....   71
  NEEDS SRS  -    ONLINE  134, XLCOMN   14 .............................    148
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 .......................................... 2020
                                                          STORAGE TOTAL.... 2239

MFACT                                                    PROGRAM PROPER....  187
  NEEDS SRS  -      DOTJ   59,     STZ   14 .............................     73
  AND  FSRS  -      SQRT   44 ..........................................     44
                                                          STORAGE TOTAL....  304

MIFLS                                                    PROGRAM PROPER....  276
  NEEDS SRS  -      DOTJ   59, MATML3  120, MOVREV   74 ................    253
                                                          STORAGE TOTAL....  529

MINAB   (SECONDARY ENTRY OF MAXSN )

MINABM  (SECONDARY ENTRY OF MAXSNM)

MINSN   (SECONDARY ENTRY OF MAXSN )
```

```
MINSNM  (SECONDARY ENTRY OF MAXSNM)

MIPLS                                                    PROGRAM PROPER....  571
  NEEDS SRS  -      DOTJ   59, IXCARG   35, MATINV   90, MATML3  120,
                   MATRA   92,  MDOT3  122, MOVREV   74,  SIMEQ  441,
                     STZ   14 ........................................ 1047
  AND  FSRS  -      XLOC   12 ........................................   12
                                                          STORAGE TOTAL.... 1630

MISS                                                     PROGRAM PROPER....  335
  NEEDS SRS  -      DOTJ   59, MATML3  120,  MDOT3  122, MOVREV   74 ....  375
                                                          STORAGE TOTAL....  710

MLISCL  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   47

MLI2A6  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  128

MONOCK  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   48

MOUT                                                     PROGRAM PROPER....  130
  NEEDS SRS  -    CARIGE   47, ONLINE  134 ............................  181
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ........................................ 2020
                                                          STORAGE TOTAL.... 2331

MOUTAI                                                   PROGRAM PROPER....  357
  NEEDS SRS  -    CARIGE   47,   FIXV   35, GNHOL2   74, MAXSNM   61,
                    MOVE   32, MULPLY   34, ONLINE  134,    RND   15,
                    SAME    1 ........................................  433
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17,  EXP(2   38,    LOG   53 ................. 2111
                                                          STORAGE TOTAL.... 2901

MOVE    NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   32

MOVECS                                                   PROGRAM PROPER....   24
  NEEDS SRS  -      MOVE   32 ........................................   32
                                                          STORAGE TOTAL....   56

MOVREV  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   74

MPSEQ1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  110

MRVRS                                                    PROGRAM PROPER....   61
  NEEDS SRS  -    REVERS   29 ........................................   29
                                                          STORAGE TOTAL....   90

MSCON1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  238

MULK    (SECONDARY ENTRY OF ADDK  )

MULKS   (SECONDARY ENTRY OF ADDK  )
```

```
MULLER                                               PROGRAM PROPER....  757
  NEEDS FSRS -      SQRT   44 ...........................................   44
                                                      STORAGE TOTAL....  801

MULPLY  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   34

MUVADD  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....  129

MVBLOK  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   19

MVINAV  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....  221

MVNSUM  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   71

MVNTIN  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   88

MVNTNA  (SECONDARY ENTRY OF MVNTIN)

MVSQAV  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....  236

MXRARE                                               PROGRAM PROPER....  302
  NEEDS FSRS -     EXP(2   38 ...........................................   38
                                                      STORAGE TOTAL....  340

NEXCOS  (SECONDARY ENTRY OF SEQSAC)

NEXSIN  (SECONDARY ENTRY OF SEQSAC)

NMZMG1  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   34

NOINT1                                               PROGRAM PROPER....  369
  NEEDS SRS  -    LINTR1   96 ...........................................   96
                                                      STORAGE TOTAL....  465

NOINT2  (SECONDARY ENTRY OF NOINT1)

NRMVEC                                               PROGRAM PROPER....  111
  NEEDS SRS  -     MAXSN   54 ...........................................   54
  AND  FSRS  -      SQRT   44 ...........................................   44
                                                      STORAGE TOTAL....  209

NTHA    NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   11

NURINC  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....  121

NXALRM                                               PROGRAM PROPER....  243
  NEEDS SRS  -    FASCN1  107 ...........................................  107
                                                      STORAGE TOTAL....  350

ONLINE                                               PROGRAM PROPER....  134
  NEEDS FSRS -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ........................................... 2020
                                                      STORAGE TOTAL.... 2154
```

```
************************        SUBROUTINE ROSTERS         ************************
*  OUDATA  TO  PLYSYN  *                                   *  OUDATA  TO  PLYSYN  *
************************                                   ************************


OUDATA                                                     PROGRAM PROPER....  495
  NEEDS SRS  -    FAPSUM   14, FXDATA  102,    LOC    4, MVBLOK   19,
                    PAKN   78, VARARG   44 ............................  261
  AND  FSRS  -    (EFT)     7, (EXEM)  458,  (IOB)  570,  (IOS)   87,
                  (IOU)    24,  (STB)   53,  (TES)    1,  (WER)   57,
                    DUMP  177,   EXIT   17 ............................ 1451
                                                             STORAGE TOTAL.... 2207

PACDAT                                                     PROGRAM PROPER....  152
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 ............................  764
                                                             STORAGE TOTAL....  916

PAKN                                                       PROGRAM PROPER....   78
  NEEDS SRS  -    FXDATA  102 .........................................  102
                                                             STORAGE TOTAL....  180

PDUMP   (SECONDARY ENTRY OF DUMP  )

PLANSP                                                     PROGRAM PROPER.... 1169
  NEEDS SRS  -      ADDK  114, CHOOSE   17, CHPRTS   76, COSIS1  406,
                    COSP  504, COSTBL  121, IXCARG   35, KOLAPS  100,
                  LIMITS   44,  MATRA   92, MOVREV   74, REVERS   29,
                   ROAR2  174,   SETK   37,  SPLIT  224,    STZ   14,
                   XOOZE    4 ......................................... 2065
  AND  FSRS  -       COS  105,   XLOC   12 ............................  117
                                                             STORAGE TOTAL.... 3351

PLOTVS                                                     PROGRAM PROPER....  494
  NEEDS SRS  -    ONLINE  134,    RND   15,   SETK   37,  SETKV   15,
                  SWITCH   15 .........................................  216
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ......................................... 2020
                                                             STORAGE TOTAL.... 2730

PLTVS1                                                     PROGRAM PROPER....  817
  NEEDS SRS  -     BOOST   34,  MAXSN   54, MULPLY   34, ONLINE  134,
                  PLOTVS  494, RMSDEV   50,    RND   15,   SAME    1,
                    SETK   37,  SETKV   15, SETKVS   25, SETLIN   27,
                  SWITCH   15, VARARG   44 ............................  979
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17,   SQRT   44,   XLOC   12 ................ 2076
                                                             STORAGE TOTAL.... 3872

PLURAL  (SECONDARY ENTRY OF SEVRAL)

PLURNS  NEEDS NO LOWER ROUTINES                              STORAGE TOTAL....   73

PLYSYN                                                     PROGRAM PROPER....  172
  NEEDS SRS  -    CONVLV   56 .........................................   56
  AND  FSRS  -       COS  105 .........................................  105
                                                             STORAGE TOTAL....  333
```

```
POKCT1                                                       PROGRAM PROPER....  219
  NEEDS SRS  -    FRQCT1  117 ..........................................  117
                                                             STORAGE TOTAL....  336

POLYDV                                                       PROGRAM PROPER....  130
  NEEDS SRS  -      MOVE   32,     STZ   14 ..............................   46
                                                             STORAGE TOTAL....  176

POLYEV  NEEDS NO LOWER ROUTINES                              STORAGE TOTAL....   54

POLYSN                                                       PROGRAM PROPER....  256
  NEEDS SRS  -    CONVLV   56,   MOVE   32 ..............................   88
  AND  FSRS  -       COS  105,   SQRT   44 ..............................  149
                                                             STORAGE TOTAL....  493

POWER                                                        PROGRAM PROPER....   50
  NEEDS FSRS -     EXP(2   38 ..........................................   38
                                                             STORAGE TOTAL....   88

PRBFIT                                                       PROGRAM PROPER....  373
  NEEDS FSRS -       EXP   52,  EXP(2   38,    SQRT   44 .................  134
                                                             STORAGE TOTAL....  507

PROB2   NEEDS NO LOWER ROUTINES                              STORAGE TOTAL....  229

PROCOR  NEEDS NO LOWER ROUTINES                              STORAGE TOTAL....  770

PSQRT                                                        PROGRAM PROPER....  155
  NEEDS FSRS -      SQRT   44 ..........................................   44
                                                             STORAGE TOTAL....  199

PWMLIV                                                       PROGRAM PROPER....  300
  NEEDS SRS  -    MLI2A6  128, ONLINE  134 ..............................  262
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 .......................................... 2020
                                                             STORAGE TOTAL.... 2582

QACORR                                                       PROGRAM PROPER....  207
  NEEDS SRS  -    FXDATA  102, PROCOR  770 ..............................  872
                                                             STORAGE TOTAL.... 1079

QCNVLV                                                       PROGRAM PROPER....  569
  NEEDS SRS  -    FXDATA  102, PROCOR  770 ..............................  872
  AND  FSRS  -      XLOC   12 ..........................................   12
                                                             STORAGE TOTAL.... 1453

QFURRY                                                       PROGRAM PROPER....  244
  NEEDS SRS  -    CHPRTS   76,   COSP  504, COSTBL  121, KOLAPS  100,
                    MOVE   32,  SPLIT  224,    STZ   14, XSPECT  523 .... 1594
  AND  FSRS  -       COS  105,   XLOC   12 ..............................  117
                                                             STORAGE TOTAL.... 1955
```

```
QIFURY                                                  PROGRAM PROPER....  280
  NEEDS SRS  -      COSP  504, COSTBL  121 ..............................  625
  AND  FSRS  -       COS  105,   XLOC   12 ..............................  117
                                                         STORAGE TOTAL.... 1022

QINTR1                                                  PROGRAM PROPER....  229
  NEEDS SRS  -    QUFIT1   79,    RND   15 ..............................   94
                                                         STORAGE TOTAL....  323

QUFIT1  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   79

QXCORR                                                  PROGRAM PROPER....  283
  NEEDS SRS  -    FXDATA  102, PROCOR  770 ..............................  872
  AND  FSRS  -      XLOC   12 ........................................   12
                                                         STORAGE TOTAL.... 1167

QXCOR1                                                  PROGRAM PROPER....  502
  NEEDS SRS  -    IXCARG   35, LIMITS   44, PROCOR  770, REVERS   29,
                    SETK   37,    STZ   14 ..............................  929
  AND  FSRS  -      XLOC   12 ........................................   12
                                                         STORAGE TOTAL.... 1443

RDATA                                                   PROGRAM PROPER....  645
  NEEDS SRS  -     CMPRA   18, FNDFMT   88, HVTOIV   39, INTHOL   72,
                  IVTOHV   70, IXCARG   35, LOCATE  512, ONLINE  134,
                  REREAD  114,  REVER   30 .............................. 1112
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (RER)   37,  (SPH)  183,  (TES)    1,  (WER)   57,
                    DUMP  177,   EXIT   17,   XLOC   12 ................. 2069
                                                         STORAGE TOTAL.... 3826

REFIT   (SECONDARY ENTRY OF SPLIT )

REFLEC  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   28

REIM    (SECONDARY ENTRY OF AMPHZ )

REMAV   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   36

REREAD                                                  PROGRAM PROPER....  114
  NEEDS FSRS -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (RER)   37,  (TES)    1,   DUMP  177,   EXIT   17 .... 1817
                                                         STORAGE TOTAL.... 1931

RETURN  (SECONDARY ENTRY OF LOCATE)

REVER   NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   30

REVERS  NEEDS NO LOWER ROUTINES                          STORAGE TOTAL....   29

RLSPR                                                   PROGRAM PROPER....  142
  NEEDS SRS  -      FDOT   40 ........................................   40
                                                         STORAGE TOTAL....  182
```

```
RLSPR2                                                  PROGRAM PROPER....  700
  NEEDS SRS  -      DOTJ   59,   DOTP  264, IXCARG   35, MATML3  120,
                  MOVREV   74,  SIMEQ  441,    STZ   14 ................. 1007
  AND  FSRS  -      XLOC   12 ......................................     12
                                                          STORAGE TOTAL.... 1719

RLSSR                                                   PROGRAM PROPER....   82
  NEEDS SRS  -      FDOT   40 ......................................     40
                                                          STORAGE TOTAL....  122

RMSDAV  (SECONDARY ENTRY OF RMSDEV)

RMSDEV                                                  PROGRAM PROPER....   50
  NEEDS FSRS -      SQRT   44 ......................................     44
                                                          STORAGE TOTAL....   94

RND     NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   15

RNDDN   (SECONDARY ENTRY OF RND    )

RNDUP   (SECONDARY ENTRY OF RND    )

RNDV                                                    PROGRAM PROPER....   34
  NEEDS SRS  -       RND   15 ......................................     15
                                                          STORAGE TOTAL....   49

RNDVDN  (SECONDARY ENTRY OF RNDV   )

RNDVUP  (SECONDARY ENTRY OF RNDV   )

ROAR2                                                   PROGRAM PROPER....  174
  NEEDS SRS  -     MATRA   92, MOVREV   74, REVERS   29 ................  195
                                                          STORAGE TOTAL....  369

ROTAT1  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   46

RPLFMT  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   17

RSKIP                                                   PROGRAM PROPER....   37
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 ............................  764
                                                          STORAGE TOTAL....  801

RVPRTS  (SECONDARY ENTRY OF CHPRTS)

SAME    NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....    1

SCPSCL  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   33

SEARCH  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   25

SEQSAC                                                  PROGRAM PROPER....   94
  NEEDS FSRS -       COS  105 ......................................    105
                                                          STORAGE TOTAL....  199

SETAPT  (SECONDARY ENTRY OF INDEX )
```

```
SETEST  (SECONDARY ENTRY OF INDEX )

SETINO                                                         PROGRAM PROPER....   84
  NEEDS SRS  -     FSKIP   50, XLIMIT   25 .............................   75
  AND  FSRS  -    (EXEM)  458,  (IOB)  570,  (IOS)   87,  (IOU)   24,
                   (RER)   37,  (RWT)    7,  (TES)    1,  (TSB)   66,
                    DUMP  177,   EXIT   17 ............................. 1444
                                                                STORAGE TOTAL.... 1603

SETK    NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   37

SETKP                                                          PROGRAM PROPER....   40
  NEEDS SRS  -      SETK   37 ..........................................   37
                                                                STORAGE TOTAL....   77

SETKS   (SECONDARY ENTRY OF SETK  )

SETKV   NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   15

SETKVS  NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   25

SETLIN  NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   27

SETLNS                                                         PROGRAM PROPER....   39
  NEEDS SRS  -    SETLIN   27 ..........................................   27
                                                                STORAGE TOTAL....   66

SETSBV  (SECONDARY ENTRY OF LOCATE)

SETUP   (SECONDARY ENTRY OF LOCATE)

SETVCP  (SECONDARY ENTRY OF SETKP )

SETVEC  (SECONDARY ENTRY OF SETK  )

SEVRAL                                                         PROGRAM PROPER....  416
  NEEDS SRS  -    LOCATE  512 ..........................................  512
                                                                STORAGE TOTAL....  928

SHFTR1  NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   70

SHFTR2  NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   72

SHUFFL                                                         PROGRAM PROPER....  101
  NEEDS SRS  -    GETRD1  229, REREAD  114, SEARCH   25, SIZEUP  136 ....  504
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (RER)   37,  (TES)    1,   DUMP  177,   EXIT   17 .... 1817
                                                                STORAGE TOTAL.... 2422

SIFT    NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....   30

SIMEQ   NEEDS NO LOWER ROUTINES                                 STORAGE TOTAL....  441

SIN     (SECONDARY ENTRY OF COS   )

SINTBL  (SECONDARY ENTRY OF COSTBL)
```

```
SINTBX  (SECONDARY ENTRY OF COSTBL)

SISP    (SECONDARY ENTRY OF COSP   )

SIZEUP  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  136

SIZUPL  (SECONDARY ENTRY OF SIZEUP)

SMPRDV  (SECONDARY ENTRY OF POWER  )

SMPSON  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  317

SPCOR2                                                   PROGRAM PROPER....  291
  NEEDS SRS   -    FXDATA  102, IXCARG   35, LIMITS   44, PROCOR  770,
                   QXCOR1  502, REVERS   29,   SETK   37,     STZ   14 .... 1533
  AND  FSRS   -      XLOC   12 ......................................     12
                                                          STORAGE TOTAL.... 1836

SPLIT   NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....  224

SQRDEV  (SECONDARY ENTRY OF SQRDFR)

SQRDFR  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   36

SQRMLI  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   55

SQROOT                                                   PROGRAM PROPER....   24
  NEEDS FSRS -       SQRT   44 ......................................     44
                                                          STORAGE TOTAL....   68

SQRSUM  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   36

SQRT    NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   44

SQUARE  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   32

SRCH1                                                    PROGRAM PROPER....   93
  NEEDS SRS   -    XACTEQ   11 ......................................     11
                                                          STORAGE TOTAL....  104

STEPC   (SECONDARY ENTRY OF DELTA  )

STEPL   (SECONDARY ENTRY OF DELTA  )

STEPR   (SECONDARY ENTRY OF DELTA  )

STORE   (SECONDARY ENTRY OF LOCATE)

STZ     NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   14

STZS    NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   24

SUBK    (SECONDARY ENTRY OF ADDK   )

SUBKS   (SECONDARY ENTRY OF ADDK   )
```

```
SUM      NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    23

SUMDEV   (SECONDARY ENTRY OF SUMDFR)

SUMDFR   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    44

SWITCH   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    15

TAMVL    NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    63

TAMVR    (SECONDARY ENTRY OF TAMVL )

TANH     NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    86

TIMA2B   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....   124

TIMSUB                                                     PROGRAM PROPER....   229
  NEEDS SRS  -     TIMA2B  124 ......................................   124
                                                            STORAGE TOTAL....   353

TINGL    NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    43

TINGLA   (SECONDARY ENTRY OF TINGL )

TRMINO                                                     PROGRAM PROPER....    67
  NEEDS SRS  -     FAPSUM   14,  FSKIP   50, FXDATA  102,    LOC    4,
                   MVBLOK   19, OUDATA  495,   PAKN   78, VARARG   44,
                   XLIMIT   25 ......................................   831
  AND  FSRS  -      (EFT)    7, (EXEM)  458,  (IOB)  570,  (IOS)   87,
                    (IOU)   24,  (RWT)    7,  (STB)   53,  (TES)    1,
                    (WER)   57,   DUMP  177,   EXIT   17 ................  1458
                                                            STORAGE TOTAL....  2356

UNPAKN   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    78

VARARG   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    44

VDOTV    NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    25

VDVBYV   NEEDS NO LOWER ROUTINES                            STORAGE TOTAL....    22

VECOUT                                                     PROGRAM PROPER....    66
  NEEDS SRS  -     FNDFMT   88, ONLINE  134,  REVER   30, RPLFMT   17 ....   269
  AND  FSRS  -     (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                    (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                     EXIT   17 ......................................  2020
                                                            STORAGE TOTAL....  2355

VINDEX   (SECONDARY ENTRY OF INDEX )

VMNUSV   (SECONDARY ENTRY OF VPLUSV)
```

```
VOUT                                                  PROGRAM PROPER....  104
  NEEDS SRS  -    CARIGE   47, FNDFMT   88,  HLADJ   46, ONLINE  134,
                   REVER   30, RPLFMT   17, VECOUT   66 .................  428
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ....................................... 2020
                                                       STORAGE TOTAL.... 2552

VPLUSV  NEEDS NO LOWER ROUTINES                        STORAGE TOTAL....   34

VRSOUT                                                PROGRAM PROPER....   47
  NEEDS SRS  -    CARIGE   47, FNDFMT   88, ONLINE  134,  REVER   30,
                  RPLFMT   17, VECOUT   66 ..........................  382
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ....................................... 2020
                                                       STORAGE TOTAL.... 2449

VSOUT                                                 PROGRAM PROPER....   37
  NEEDS SRS  -    CARIGE   47, FNDFMT   88,  HLADJ   46, ONLINE  134,
                   REVER   30, RPLFMT   17, VECOUT   66,   VOUT  104 ....  532
  AND  FSRS  -    (EXEM)  458,  (IOH) 1016,  (IOS)   87,  (IOU)   24,
                   (SPH)  183,  (TES)    1,  (WER)   57,   DUMP  177,
                    EXIT   17 ....................................... 2020
                                                       STORAGE TOTAL.... 2589

VTIMSV  NEEDS NO LOWER ROUTINES                        STORAGE TOTAL....   34

WAC     NEEDS NO LOWER ROUTINES                        STORAGE TOTAL....  107

WHERE   (SECONDARY ENTRY OF LOCATE)

WHICH   NEEDS NO LOWER ROUTINES                        STORAGE TOTAL....    4

WLLSFP                                                PROGRAM PROPER....  216
  NEEDS SRS  -      FOOT   40,   MOVE   32 ...........................   72
                                                       STORAGE TOTAL....  288

WRTDAT                                                PROGRAM PROPER....   77
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                    DUMP  177,   EXIT   17 ...........................  764
                                                       STORAGE TOTAL....  841

XACTEQ  NEEDS NO LOWER ROUTINES                        STORAGE TOTAL....   11

XADDK   (SECONDARY ENTRY OF ADDK  )

XADDKS  (SECONDARY ENTRY OF ADDK  )

XARG    (SECONDARY ENTRY OF LOCATE)

XAVRGE                                                PROGRAM PROPER....   34
  NEEDS SRS  -      XDIV   27 .......................................   27
                                                       STORAGE TOTAL....   61

XAVRGR  (SECONDARY ENTRY OF XAVRGE)

XBOOST  (SECONDARY ENTRY OF BOOST )
```

```
XCMPRA  (SECONDARY ENTRY OF CMPRA )

XDANL   (SECONDARY ENTRY OF ADANL )

XDANX   (SECONDARY ENTRY OF ADANL )

XDELTA  (SECONDARY ENTRY OF DELTA )

XDFPRS  (SECONDARY ENTRY OF DIFPRS)

XDIV    NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   27

XDIVK   (SECONDARY ENTRY OF ADDK  )

XDIVKS  (SECONDARY ENTRY OF ADDK  )

XDIVR   (SECONDARY ENTRY OF XDIV  )

XDPRSS  (SECONDARY ENTRY OF BOOST )

XDVIDE                                               PROGRAM PROPER....   33
  NEEDS SRS  -     XDIV   27 ....................................     27
                                                      STORAGE TOTAL....   60

XDVIDR  (SECONDARY ENTRY OF XDVIDE)

XDVRK   (SECONDARY ENTRY OF ADDK  )

XDVRKS  (SECONDARY ENTRY OF ADDK  )

XFIXM   NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   31

XINDEX  (SECONDARY ENTRY OF LOCATE)

XLCOMN  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   14

XLIMIT  NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   25

XLOC    NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   12

XLOCV   NEEDS NO LOWER ROUTINES                       STORAGE TOTAL....   24

XLSHFT  (SECONDARY ENTRY OF LSHFT )

XMLPLY  (SECONDARY ENTRY OF MULPLY)

XMULK   (SECONDARY ENTRY OF ADDK  )

XMULKS  (SECONDARY ENTRY OF ADDK  )

XNAME   (SECONDARY ENTRY OF LOCATE)

XNARGS  (SECONDARY ENTRY OF LOCATE)

XNTHA   (SECONDARY ENTRY OF NTHA  )
```

```
XNTSUM  (SECONDARY ENTRY OF INTSUM)

XOOZE   NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....    4

XREMAV                                                   PROGRAM PROPER....   31
  NEEDS SRS  -    XAVRGE   34,   XDIV   27 .............................   61
                                                          STORAGE TOTAL....   92

XRFLEC  (SECONDARY ENTRY OF REFLEC)

XSAME   (SECONDARY ENTRY OF SAME  )

XSMDEV  (SECONDARY ENTRY OF SUMDFR)

XSMDFR  (SECONDARY ENTRY OF SUMDFR)

XSPECT                                                   PROGRAM PROPER....  523
  NEEDS SRS  -    CHPRTS   76,   COSP  504, KOLAPS  100,  SPLIT  224 ....  904
  AND  FSRS  -      XLOC   12 ......................................   12
                                                          STORAGE TOTAL.... 1439

XSQDEV  (SECONDARY ENTRY OF XSQDFR)

XSQDFR  NEEDS NO LOWER ROUTINES                           STORAGE TOTAL....   37

XSQRUT                                                   PROGRAM PROPER....   37
  NEEDS SRS  -      FIXV   35 ......................................   35
  AND  FSRS  -      SQRT   44 ......................................   44
                                                          STORAGE TOTAL....  116

XSQSUM  (SECONDARY ENTRY OF SQRSUM)

XSQUAR  (SECONDARY ENTRY OF SQUARE)

XSTEPC  (SECONDARY ENTRY OF DELTA )

XSTEPL  (SECONDARY ENTRY OF DELTA )

XSTEPR  (SECONDARY ENTRY OF DELTA )

XSTLIN  (SECONDARY ENTRY OF SETLIN)

XSUBK   (SECONDARY ENTRY OF ADDK  )

XSUBKS  (SECONDARY ENTRY OF ADDK  )

XSUM    (SECONDARY ENTRY OF SUM   )

XVDRBV  (SECONDARY ENTRY OF XVDVBV)

XVDVBV                                                   PROGRAM PROPER....   34
  NEEDS SRS  -      XDIV   27 ......................................   27
                                                          STORAGE TOTAL....   61

XVMNSV  (SECONDARY ENTRY OF VPLUSV)
```

```
XVPLSV  (SECONDARY ENTRY OF VPLUSV)

XVTMSV  (SECONDARY ENTRY OF VTIMSV)

XWHICH  (SECONDARY ENTRY OF WHICH )

ZEFBCD                                                    PROGRAM PROPER....    54
  NEEDS FSRS -    (EXEM)  458,  (IOS)   87,  (IOU)   24,  (TES)    1,
                   DUMP  177,   EXIT   17 ............................  764
                                                           STORAGE TOTAL....  818

ZEFBIN  (SECONDARY ENTRY OF ZEFBCD)
```

10
Complete Program Listings

The remainder of this volume is devoted to listings of the symbolic card decks of the program library. The reproductions shown here have been made by a photo-offset process from IBM 1401 printings of magnetic tapes produced by a formating program whose inputs were master tapes (their development is discussed below) containing the symbolic decks. The function of the formating program was merely to paginate the source decks and provide headings for dictionary-style access. The program decks on the master symbolic tape are serialized (in columns 76 through 79) in a manner evident from the listings of this section, and the normal page divisions are made every 75 cards. Some of the pages, however, contain fewer than 75 cards. This does not imply the accidental appearance of blank cards on the source tape but rather is a side effect of page division rules used by the formating program to avoid a splitting of photograph inserts (as occur in GRAPH, LINE, HSTPLT, etc.) between pages. (The obscured portions of the cards used as spacers for the photographic inserts are blank except for instructions concerning the size of prints to be inserted.)

The visual distinctions in the listings between 1's and I's and between 0's and O's are occasionally troublesome, but one gets adept in these discriminations after a while. As a general rule we try to avoid the use of names composed of letter-number mixtures except where the number is terminal. Even in these cases, if the context demands a name with a terminal zero we often substitute the character Z.

The majority of cards in the symbolic decks are devoted to program description. The general card format adhered to has been as shown in Table 1 below, with simple modifications in the cases of multiple-entry programs.

Although the format details will be apparent from the cards, a few comments are in order. Note that the program name always appears starting in column 2 of the third card of each deck. There is some inconsistency in the abstracts—the older programs adjust comments into column 15, newer ones into column 16. Under "equipment" many of the older programs state "709 or 7090." These programs also work on the 7094. Storage requirements are given in decimal. For FORTRAN programs these requirements will depend to some extent on the compiling system used. The numbers given here are storage lengths as compiled by the system described in Section 1. In those cases where the examples consist of pairs of descriptions of inputs and outputs, one is to infer that the sample CALL statement, as written, is executed following the establishment of the inputs. For a discussion of conventions used in designing calling sequences turn to Section 4. For sample test programs used to prove the examples see Section 2. Section 1 also includes further discussion of notational conventions.

TABLE 1

PROLOGUE CARDS

ABSTRACT

- Discussion of program function
- Language comments
- Equipment needed
- Storage required
- Speed
- Author and date

Usage

- Transfer vector
- Sample FORTRAN usage statement
- Input descriptions in terms of the sample statement
- Output descriptions in terms of the sample statement
- Examples of usage

PROGRAM PROPER

END CARD

The master symbolic tapes which are the source of the present listings were generated under a program-development scheme roughly as follows.

1. Authors write their own complete programs, including examples.
2. A test program is written (preferably by a second programmer) to carry out the examples.
3. The program is debugged to the author's satisfaction.
4. The symbolic decks (program and tester) are added to a batch of programs awaiting "acceptance processing."
5. Acceptance processing, which is carried out when the batch reaches reasonable size, comprises the following steps:

   (a) The batch is listed and subjected to format and grammatical editing.
   (b) The program decks are loaded onto tape and serialized onto a second tape.
   (c) The serialized tape is punched to give serialized decks, and compiled to give binary decks.
   (d) The serialized decks are collated with the test decks, a "definitive execution" is carried out, and the test results are carefully rechecked.
   (e) When all troubles are corrected, the binary program decks from (d) are compared with those from (c) and then added to the subroutine library, and the serialized tape from (b) is added by a merging program to the previously developed master symbolic tapes.

Our experience has been that programs accepted by this process have a high probability of working as expected.

```
*      ABSVAL (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0116
*      FAP
*ABSVAL
       COUNT   100
       LBL     ABSVAL
       ENTRY   ABSVAL (ANYVEC,ILO,IHI,ABSVEC,IANS)
*
*                   ----ABSTRACT----
*
* TITLE - ABSVAL
*       FAST ABSOLUTE VALUE OF A VECTOR
*
*             ABSVAL FORMS A VECTOR EQUAL TO THE MAGNITUDE OF A
*             SPECIFIED RANGE OF ANOTHER VECTOR. INPUT VECTOR MAY
*             BE FIXED POINT OR FLOATING POINT.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 50 REGISTERS
* SPEED     - TAKES 74 + 6*N MACHINE CYCLES ON THE 7090, WHERE
*                 N = NO. ELEMENTS IN SPECIFIED RANGE
* AUTHOR    - S.M. SIMPSON JR,  JUNE 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL ABSVAL(ANYVEC,ILO,IHI,ABSVEC,IANS)
*
* INPUTS
*
*    ANYVEC(I) I=ILO,...,IHI IS THE RANGE (ANYVEC IS FIXED OR FLTG).
*
*    ILO       MUST EXCEED 0.
*
*    IHI       MUST EQUAL OR EXCEED ILO.
*
* OUTPUTS
*
*    ABSVEC(I) I=1,2,...,(IHI-ILO+1) CONTAINS
*              MAGNITUDE(ANYVEC(ILO,...,IHI)).
*              EQUIVALENCE  (ANYVEC,ABSVEC)  IS PERMITTED.
*
*    IANS      = 0 MEANS JOB DONE.
*              =-1 MEANS ILLEGAL ILO OR IHI.
*
* EXAMPLES
*
* 1. INPUTS  - ANYVEC(1...10) = -1.0,-2.0,-3.0,...  ILO=3 IHI=7
*    OUTPUTS - IANS=0  ABSVEC(1...5)=3.0,4.0,5.0,6.0,7.0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ILO=IHI=2
*    OUTPUTS - IANS=0, ABSVEC(1)=2.0
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=2
*    OUTPUTS - IANS=-1
*
       HTR     0
       BCI     1,ABSVAL
ABSVAL SXA     EXIT,1
       SXD     ABSVAL-2,4
       CLA     2,4            A(A(ILO))
       STA     GET2
       CLA     3,4            A(A(IHI))
       STA     GET3
       CLA     5,4            A(A(IANS))
       STA     PUT5
* SET UP CONSTANTS ILO, IHI, LVECT AND CHECK THEM
* SET IANS FOR ILLEGAL INPUT.
       CLS     K1
       STO     IANS
  GET2 CLA     **             A(ILO)
       ARS     18
```

****
*   ABSVAL   *
****
(PAGE 2)

```
          STO   ILO
          TMI   LEAVE
          TZE   LEAVE
     GET3 CLA   **             A(IHI)
          ARS   18
          STO   IHI
          TMI   LEAVE
          TZE   LEAVE
          SUB   ILO
          ADD   K1
          STO   LVECT
          TMI   LEAVE
          TZE   LEAVE
          STZ   IANS
     * SET LOOP UP.
          CLA   1,4            A(A(ANYVEC))
          SUB   ILO
          ADD   K2
          STA   CAL
          CLA   4,4            A(A(ABSVEC))
          ADD   K1
          STA   STO
          LXA   LVECT,1
     * LOOP.
      CAL CAL   **,1           A(ANYVEC)-ILO+2
      STO STO   **,1           A(ABSVEC)+1
          TIX   CAL,1,1
     * STORE IANS AND LEAVE.
    LEAVE CLA   IANS
          ALS   18
     PUT5 STO   **             A(IANS)
     EXIT AXT   **,1
          TRA   6,4
     * CONSTANTS
       K1 PZE   1
       K2 PZE   2
     * VARIABLES
      ILO PZE   **
      IHI PZE   **
     IANS PZE   **             0 OR -1
    LVECT PZE   **             IHI-ILO+1
          END
```

```
*       ADANL (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0335
*       FAP
*ADANL
        COUNT   300
        LBL     ADANL
        ENTRY   ADANL (AA,N,M,DAA)
        ENTRY   XDANL (XX,N,M,DXX)
        ENTRY   ADANX (IAA,N,M,IDAA)
        ENTRY   XDANX (IXX,N,M,IDXX)
*
*                    ----ABSTRACT----
*
* TITLE - ADANL WITH SECONDARY ENTRY POINTS XDANL, ADANX, XDANX
*         MODIFY AUTO- OR CROSS CORRELATIONS FOR DANIELL SPECTRA
*
*               ADANL WEIGHTS A ONE-SIDED, FLOATING POINT AUTOCORRELATION
*               FUNCTION, A(L) L=0...N, BY A SIN(Y)/Y TYPE CURVE TO
*               PRODUCE A FLOATING POINT OUTPUT DA(L)
*
*                                         M          L*PI
*                      DA(L)  =  A(L) * ( ---- * SIN(----) )
*                                        L*PI          M
*
*                        FOR  L  = 0,1,...,N
*                      WHERE  M AND N ARE INPUT PARAMETERS
*                             PI = 3.14159265
*
*               XDANL WEIGHTS A TWO-SIDED, FLOATING POINT CROSS-
*               CORRELATION FUNCTION, X(L) L= -N...0...N, BY A SIN(Y)/Y
*               TYPE CURVE TO PRODUCE A FLOATING POINT OUTPUT DX(L)
*
*                                         M          L*PI
*                      DX(L)  =  X(L) * ( ---- * SIN(----) )
*                                        L*PI          M
*
*                        FOR  L  = -N,-N+1,...,N
*                      WHERE  M AND N ARE INPUT PARAMETERS
*                             PI = 3.14159265
*
*               ADANX IS IDENTICAL TO ADANL EXCEPT THAT THE INPUTS AND
*               OUTPUTS ARE FIXED POINT VECTORS.
*
*               XDANX IS IDENTICAL TO XDANL EXCEPT THAT THE INPUTS AND
*               OUTPUTS ARE FIXED POINT VECTORS.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 183 CELLS
* SPEED      - (APPROX)         709                 7090
*               FLOATING PT -  6M +  .9N        1.2M + .18N     MILLISECS
*               FIXED POINT -  6M + 1.6N        1.2M + .325N    MILLISECS
* AUTHOR     - J. CLAERBOUT
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - SIN
*
* FORTRAN USAGE OF ADANL
*     CALL ADANL (AA,N,M,DAA)
*
* INPUTS
*    AA(I)      I=1,2,...,N+1  CONTAINS THE AUTOCORRELATION   A(0,...,N)
*                            WHERE  AA(I) = AC(I-1)
*
*    N          MUST BE NON NEGATIVE
*
*    M          IS THE DANIELL WEIGHTING PARAMETER
*               IS A NON-ZERO INTEGER
*
* OUTPUTS
*    DAA(I)     I=1...N+1    CONTAINS THE WEIGHTED AUTOCORRELATION
*                            DA(0...N)  AS DEFINED IN ABSTRACT
*                            WHERE  DAA(I) = DA(I-1)
```

```
*               EQUIVALENCE (DAA,AA) IS PERMITTED
*
* FORTRAN USAGE OF XDANL
*     CALL XDANL(XX,N,M,DXX)
*
* INPUTS
*    XX(I)      I= -N+1,-N+2,...,N+1  CONTAINS THE CROSS-CORRELATION
*                           X(-N...N)  WHERE  XX(I) = X(I-1)
*
*    N          SAME AS FOR ADANL
*    M          SAME AS FOR ADANL
*
* OUTPUTS
*    DXX(I)     I= -N+1,-N+2,...,N+1  CONTAINS THE WEIGHTED CROSS-
*                         CORRELATION  DX(-N...N) AS DEFINED IN ABSTRACT
*                         WHERE  DXX(I) = DX(I-1)
*               EQUIVALENCE (XX,DXX) IS PERMITTED
*
* FORTRAN USAGE OF ADANX
*     CALL ADANX(IAA,N,M,IDAA)
*
* INPUTS
*    IAA(I)     SAME MEANING AS FOR ADANL EXCEPT THAT THE AUTOCORRELATION
*                   ELEMENTS ARE FIXED POINT QUANTITIES. THE POSITION OF
*                   THE BINARY POINT IS IMMATERIAL BUT THE DATA MUST NOT
*                   OCCUPY BITS 1 THROUGH 8 .
*
*    N          SAME MEANING AS FOR ADANL
*    M          SAME MEANING AS FOR ADANL
*
* OUTPUTS
*    IDAA(I)    SAME MEANING AS FOR ADANL EXCEPT THE DATA IS FIXED POINT
*                   WITH BINARY POINT SAME AS THAT OF IAA.
*
* FORTRAN USAGE OF XDANX
*     CALL XDANX(IXX,N,M,IDXX)
*
* INPUTS
*     IXX(I)    SAME MEANING AS FOR XDANL EXCEPT THAT THE DATA IS FIXED
*                   POINT AND MUST NOT OCCUPY BITS 1 THROUGH 8.
*
*    N          SAME MEANING AS FOR XDANL
*    M          SAME MEANING AS FOR XDANL
*
* OUTPUTS
*    IDXX(I)    SAME MEANING AS FOR XDANL EXCEPT IDXX IS FIXED POINT.
*
* EXAMPLES
*
* 1. GENERAL BEHAVIOR ON ELEMENTARY CORRELATIONS
*    INPUTS   - AA(1...4)=1.0,1.0,1.0,1.0  IAA(1...4)=500,500,500,500
*               XX(1...7)=1.0,1.0,...,1.0  IXX(1...7)=500,500,...,500
*               (NOTE - BIT 9 IS THE MOST SIGNIFICANT BIT OCCUPIED
*                BY IAA OR IXX WITH THESE DEFINITIONS)
*               N = 3   M = 2
*    USAGE    -        CALL ADANL(AA,N,M,DAA)
*                      CALL XDANL(XX(4),N,M,DXX(4))
*                      CALL ADANX(IAA,N,M,IDAA)
*                      CALL XDANX(IXX(4),N,M,IDXX(4))
*    OUTPUTS - DAA(1...4)=1.0,.636620,0.0,-.212207
*               DXX(1...7)=-0.212207,0.0,.636620,1.0,.636620,0.0,-.212207
*               IDAA(1...4)=500,318,0,-106
*               IDXX(1...7)=-106,0,318,500,318,0,-106
*
* 2. EQUATING OUTPUTS WITH INPUTS
*    INPUTS   - SAME AS EXAMPLE 1.
*    USAGE    -        CALL ADANL(AA,N,M,AA)
*                      CALL XDANL(XX(4),N,M,XX(4))
*                      CALL ADANX(IAA,N,M,IAA)
*                      CALL XDANX(IXX(4),N,M,IXX(4))
*    OUTPUTS - AA(1...4)=DAA(1...4) OF EXAMPLE 1.
*               XX(1...7)=DXX(1...7) OF EXAMPLE 1.
*               IAA(1...4)=IDAA(1...4) OF EXAMPLE 1.
```

```
*               IXX(1...7)=IDXX(1...7) OF EXAMPLE 1.
*
*
* PROGRAM FOLLOWS BELOW
*
        HTR     0
        BCI     1,ADANL
 ADANL  SXA     SV1,1
        TSX     MOVA,1         MOVE DATA TO OUTPUT FIELD
        CLA     *
        STO     AORX           AORX=0 IF CROSS
        TRA     LXL
 XDANL  SXA     SV1,1
        TSX     MOVX,1         MOVE DATA
        STZ     AORX           AORX=0 IF CROSS
        CLA     *
  LXL   STO     XORL           XORD=0 IF FIXED
        SXD     ADANL-2,4
        TRA     SETUP          SKIP FLOATING
 ADANX  SXA     SV1,1
        TSX     MOVA,1         MOVE DATA
        CLA     *              AORX=0 IF CROSS
        STO     AORX
        TRA     XXL
 XDANX  SXA     SV1,1
        TSX     MOVX,1         MOVE DATA
        STZ     AORX           AORX=0 IF CROSS
  XXL   STZ     XORL           XORL=0 IF FIXED
        SXD     ADANL-2,4
        TRA     FLOAT
 MOVA   CLA     M8
        STA     TAX
        TRA     MAX
 MOVX   CLA     M7
        STA     TAX
  MAX   CLA     1,4
        STA     M1
        STA     M3
        CLA     4,4
        STA     MM2
        STA     M4
        CLA*    2,4
        STD     M5
        SXA     SV2,2
        SXA     MOVOV,1
        AXT     0,2
        AXC     0,1
  M1    CLA     **,2           (**=CC)
 MM2    STO     **,2           (**=DDCC)
  TAX   TRA     **             (**=M3 OR M6)
  M3    CLA     **,1           (**=CC)
  M4    STO     **,1           (**=DDCC)
        TXI     *+1,1,-1
  M6    TXI     *+1,2,1
  M5    TXL     M1,2,**        (**=N)
 MOVOV  AXT     **,1           (**=IR1)
        TRA     1,1
  M7    PZE     M3
  M8    PZE     M6
*FLOAT THE INPUT DATA
 FLOAT  CLA     4,4
        STA     FL1
        STA     FL2
        CLA*    2,4
        STD     FL4
        STD     R4
        AXT     1,1
        AXC     1,2
  FL1   CLA     **,1           **=R
        ORA     =0233000000000
        FAD     =0233000000000
        STO*    FL1
        ZET     AORX
        TRA     FL3            AUTO
  FL2   CLA     **,2           CROSS       **=R
```

```
        ORA    =0233000000000
        FAD    =0233000000000
        STO*   FL2
        TXI    *+1,2,-1
 FL3    TXI    *+1,1,1
 FL4    TXL    FL1,1,**       **=N
*SET UP FOR WEIGHTING LOOP
 SETUP  CLA*   3,4            =M
        ARS    17
        STO    TWOM
        ARS    1
        ORA    =0233000000000
        FAD    =0233000000000
        FDP    =3.14159265
        STQ    MOVPI
        CLA    =1.
        FDP    MOVPI
        STQ    PIOVM
        CLA    4,4
        STA    R
        STA    R1
        STA    R2
        CLA*   2,4
        STD    ND             N IN DECR
        ARS    18
        STO    N
        CLA*   3,4
        ADD*   3,4
        STD    M2
        CLA    M2
        CAS    ND
        CLA    ND
        STD    MIN
        STD    MIN
        STZ    ARG
        AXC    0,2
        AXT    0,1
*BEGIN WEIGHTING LOOP
 NXWVL  TXI    *+1,2,-1
        TXI    *+1,1,1
 MIN    TXL    *+2,1,**       **=MIN(2M,N)
        TRA    SMDONE         SMOOTHING DONE
* FORM SIN(PI*I/M)
        CLA    ARG
        FAD    PIOVM
        STO    ARG
        TSX    $SIN,4
        FDP    PIOVM
        STQ    IWT            =(M/PI)SIN(PI*I/M)
        PXA    0,1            PUT I IN AC
        STO    X2MPI          I+MULTIPLE OF 2*M
 MORE   ORA    =0233000000000
        FAD    =0233000000000
        STO    L2MPI          I+MULTIPLE OF 2*M
        LXA    X2MPI,4        I+MULTIPLE OF 2*M
        CLA    IWT            =(M/PI)SIN(PI*I/M)
        FDP    L2MPI          I+MULTIPLE OF 2*M
        STQ    TEMP
 R      FMP    **,4           **=DATA LOCATION
        STO*   *-1
        ZET    AORX
        TRA    LP             AUTO COR
        LAC    X2MPI,4        CROSS COR
        LDQ    TEMP
        FMP*   R
        STO*   R
*       INCREMENT    X2MPI BY 2M
 LP     CLA    X2MPI
        ADD    TWOM
        STO    X2MPI
        CAS    N              TEST IF I PLUS SOME
        TRA    NXWVL          MULTIPLE OF 2*M IS
        TRA    MORE           GREATER THAN N
        TRA    MORE
SMDONE  ZET    XORL
```

```
        TRA    SV1           DONT FIX DATA
        AXT    1,1           FIX DATA
        AXC    1,2
R1      CLA    **,1
        UFA    =0233000000000
        LRS
        ANA    =0777777777
        LLS
        STO*   R1
        ZET    AORX
        TRA    R3            AUTO
 R2     CLA    **,2
        UFA    =0233000000000
        LRS
        ANA    =0777777777
        LLS
        STO*   R2
        TXI    *+1,2,-1
R3      TXI    *+1,1,1
R4      TXL    R1,1,**       **=N
SV1     AXT    **,1
SV2     AXT    **,2
        LXD    ADANL-2,4
        TRA    5,4
KD1     PZE    ,,1
AORX                         =0 IF CROSS
XORL                         =0 IF FIXED
MOVPI                        M/PI
PIOVM                        PI/M
N                            STORES N IN ADDR
X2MPI                        FIXED I+MULTIPLE OF 2*M
L2MPI                        FLTG I+MULTIPLE OF 2*M
IWT                          =(M/PI)SIN(PI*I/M)
TWOM                         2*M IN ADDRESS
TEMP
M2                           STORE 2M IN DECR
ND                           N IN DECR
ARG                          ARGUMENT OF SINE
        END
```

```
************************
*     ADANX            *
************************
 REFER TO
     ADANL
```



```
************************
*     ADANX            *
************************
 REFER TO
     ADANL
```

```
*       ADDK    (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0365
*       FAP
*ADDK
        COUNT   250
        LBL     ADDK
        ENTRY   ADDK   ( C, X1, X2,..., XN)
        ENTRY   SUBK   ( C, X1, X2,..., XN)
        ENTRY   MULK   ( C, X1, X2,..., XN)
        ENTRY   DIVK   ( C, X1, X2,..., XN)
        ENTRY   XADDK  (IC,IX1,IX2,...,IXN)
        ENTRY   XSUBK  (IC,IX1,IX2,...,IXN)
        ENTRY   XMULK  (IC,IX1,IX2,...,IXN)
        ENTRY   XDIVK  (IC,IX1,IX2,...,IXN)
        ENTRY   XDVRK  (IC,IX1,IX2,...,IXN)
        ENTRY   ADDKS  ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
        ENTRY   SUBKS  ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
        ENTRY   MULKS  ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
        ENTRY   DIVKS  ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
        ENTRY   XADDKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
        ENTRY   XSUBKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
        ENTRY   XMULKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
        ENTRY   XDIVKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
        ENTRY   XDVRKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*
*                       ----ABSTRACT----
*
* TITLE - ADDK  WITH SECONDARY ENTRIES   SUBK,    MULK,    DIVK,
*                                XADDK,  XSUBK,  XMULK,  XDIVK, XDVRK,
*                                ADDKS,  SUBKS,  MULKS,  DIVKS,
*                               XADDKS, XSUBKS, XMULKS, XDIVKS, XDVRKS
*
*         MODIFY A SET OF VARIABLES BY A CONSTANT OR BY CONSTANTS
*
*               ADDK AND ITS OTHER ENTRIES ARE VARIABLE LENGTH CALLING
*               SEQUENCE SUBROUTINES.  FOR THE FIRST NINE ENTRIES THE
*               FIRST ARGUMENT IS TAKEN AS A CONSTANT BY WHICH THE
*               REMAINING ARGUMENTS ARE TO BE MODIFIED.  THE MODIFICATION
*               DEPENDS ON THE ENTRY USED AS FOLLOWS
*
*                FLOATING     FIXED        FUNCTION
*                ARGUMENTS  ARGUMENTS
*
*                  ADDK       XADDK    ADDS THE CONSTANT
*                  SUBK       XSUBK    SUBTRACTS THE CONSTANT
*                  MULK       XMULK    MULTIPLIES BY THE CONSTANT
*                  DIVK       XDIVK    DIVIDES BY THE CONSTANT
*                   --        XDVRK    DIVIDES BY THE CONSTANT WITH
*                                         ROUNDING INSTEAD OF TRUNCATION
*
*               THE LAST NINE ENTRIES ASSUME THAT THE NUMBER OF ARGU-
*               MENTS IS A MULTIPLE OF THREE, AND THAT WITHIN EACH
*               TRIPLET OF THREE ARGUMENTS THE FIRST IS A CONSTANT BY
*               WHICH THE SECOND IS TO BE MODIFIED WITH THE RESULT
*               STORED IN THE THIRD ARGUMENT.  THE NAMES OF THE LAST
*               NINE ENTRIES (THE PLURAL ENTRIES) ARE DERIVED FROM THOSE
*               OF THE FIRST NINE (THE SINGULAR ENTRIES) BY ADDING
*               THE LETTER S.  THE MODIFICATION ASSOCIATED WITH A
*               PLURAL ENTRY IS THE SAME AS THAT OF ITS SINGULAR
*               COUNTERPART.
*
*               THE ORDER OF PROCESSING IS TOWARDS HIGHER ARGUMENTS.
*
*               THE DIVISION ENTRIES SKIP OVER AN ATTEMPT TO DIVIDE BY
*               ZERO WITHOUT TURNING ON ANY INDICATORS, BUT NO OTHER
*               TESTS FOR POSSIBLE OVERFLOW ARE MADE.
*
*               FOR THE PLURAL ENTRIES, AN ILLEGAL RETURN RESULTS
*               FROM AN ARGUMENT COUNT WHICH IS NOT A MULTIPLE OF 3.
*
*               THERE IS NO LIMIT ON THE NUMBER OF ARGUMENTS PERMITTED.
*
*               THERE ARE NO RESTRAINTS ON ARGUMENT EQUIVALENCES.
*               HOWEVER NO OUTPUT (THIRD) ARGUMENT MAY BE INVOLVED AS A
*               SUBSCRIPT OF, OR IN A DEFINING EXPRESSION FOR, A
*               SUBSEQUENT INPUT ARGUMENT, OTHER THAN BY A PURE
```

```
*              EQUIVALENCE.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 114 REGISTERS
* SPEED      - K1 + K2*N  MACHINE CYCLES, WHERE N = NO. MODIFICATIONS,
*              AND K1  LIES BETWEEN  33 AND 44
*                  K2  LIES BETWEEN  22 AND 59,  DEPENDING ON ENTRY
*                        AND ON COMPUTER.
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL  ADDK  ( C, X1, X2,..., XN)
*     CALL  SUBK  ( C, X1, X2,..., XN)
*     CALL  MULK  ( C, X1, X2,..., XN)
*     CALL  DIVK  ( C, X1, X2,..., XN)
*     CALL XADDK  (IC,IX1,IX2,...,IXN)
*     CALL XSUBK  (IC,IX1,IX2,...,IXN)
*     CALL XMULK  (IC,IX1,IX2,...,IXN)
*     CALL XDIVK  (IC,IX1,IX2,...,IXN)
*     CALL XDVRK  (IC,IX1,IX2,...,IXN)
*     CALL  ADDKS ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
*     CALL  SUBKS ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
*     CALL  MULKS ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
*     CALL  DIVKS ( C1, X1, Y1, C2, X2, Y2,..., CN, XN, YN)
*     CALL XADDKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*     CALL XSUBKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*     CALL XMULKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*     CALL XDIVKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*     CALL XDVRKS (IC1,IX1,IY1,IC2,IX2,IY2,...,ICN,IXN,IYN)
*
*        IN THE ABOVE EXPRESSIONS, THE LETTER N MAY HAVE ANY VALUE
*        EXCEEDING ZERO.
*
*
* INPUTS
*
*    C          IS A FLTG CONSTANT FOR MODIFYING X1,X2,...XN, FOR THE
*                  FLOATING SINGULAR ENTRIES.
*
*    X1, X2,...,XN  ARE  THE FLOATING VARIABLES TO BE MODIFIED, FOR
*                  ALL FLOATING ENTRIES.
*                  THEY ARE ALSO OUTPUTS FOR SINGULAR FLOATING ENTRIES
*
*              EQUIVALENCE(C,SOME XJ)  IS PERMITTED.  THE INITIAL
*                  VALUE OF C WILL ALWAYS BE USED FOR MODIFICATION.
*
*    C1, C2,...,CN  ARE THE FLOATING CONSTANTS USED, FOR THE FLOATING
*                  PLURAL ENTRIES, TO MODIFY X1,X2,...,XN RESPECTIVELY.
*
*              EQUIVALENCE(CJ,XL) IS PERMITTED FOR ANY J,L PAIR.
*
*    IC                IS  THE FIXED PT. ANALOG  OF  C
*    IX1, IX2,...,IXN  ARE THE FIXED PT. ANALOGS OF  X1,...,XN
*    IC1, IC2,...,ICN  ARE THE FIXED PT. ANALOGS OF  C1,...,CN
*
*
* OUTPUTS
*
*    X1, X2,..., XN  ARE OUTPUTS FOR ENTRIES ADDK, SUBK, MULK, DIVK
*                  ADDK  GIVES   X1 = X1+C,...,XN = XN+C
*                  SUBK  GIVES   X1 = X1-C,...,XN = XN-C
*                  MULK  GIVES   X1 = X1*C,...,XN = XN*C
*                  DIVK  GIVES   X1 = X1/C,...,XN = XN/C
*
*    IX1,...,IXN  ARE SIMILAR OUTPUTS FOR  XADDK, XSUBK, XMULK, XDIVK,
*                  AND XDVRK, WHERE XDIVK TRUNCATES, XDVRK ROUNDS.
*
*    Y1, Y2,...,YN  ARE OUTPUTS FOR ENTRIES  ADDKS, SUBKS, MULKS, DIVKS
*                  ADDKS  GIVES   Y1=X1+C1, Y2=X2+C2,..., YN=XN+CN
```

```
*                 SUBKS  GIVES   Y1=X1-C1,      ...      , YN=XN-CN
*                 MULKS  GIVES   Y1=X1*C1,      ...        YN=XN*CN
*                 DIVKS  GIVES   Y1=X1/C1,      ...      , YN=XN/CN
*
*              EQUIVALENCE  (CJ,XL),(CJ,YL),(XJ,YL)  IS PERMITTED
*                 FOR ALL J,L PAIRS.  THE VALUES OF THE OPERANDS
*                 USED DURING A MODIFICATION ARE THEIR PRESENT VALUES
*                 AND NOT NECESSARILY THEIR INITIAL VALUES.
*
*              THE ENTRY DIVK BYPASSES THE COMPUTATION OF EACH YJ
*                 OUTPUT FOR WHICH THE CORRESPONDING CJ HAS VALUE
*                 ZERO AT THE TIME OF MODIFICATION.
*
*              THE COMPUTATIONAL ORDER IS Y1, Y2,...,YN.
*
*    IY1, IY2,...,IYN  ARE SIMILAR OUTPUTS FOR  XADDKS, XSUBKS, XMULKS,
*                 XDIVKS, AND XDVRKS.
*
* 1. EXAMPLES OF THE SINGULAR ENTRIES
*
*    INPUTS  - A1, A2, A3 = 1., 2., 3.   B1, B2, B3 = 1., 2., 3.
*              C1, C2, C3 = 1., 2., 3.   D1, D2, D3 = 1., 2., 3.
*              IA1,IA2,IA3 = 1, 2, 3     IB1,IB2,IB3 = 1, 2, 3
*              IC1,IC2,IC3 = 1, 2, 3     ID1,ID2,ID3 = 1, 2, 3
*              IE1,IE2,IE3 = 1, 2, 3     X = 1.0
*
*    USAGE   -        CALL ADDK  (2., A1, A2, A3)
*                     CALL XADDK (2, IA1,IA2,IA3)
*                     CALL SUBK  (2., B1, B2, B3)
*                     CALL XSUBK (2, IB1,IB2,IB3)
*                     CALL MULK  (2., C1, C2, C3)
*                     CALL XMULK (2, IC1,IC2,IC3)
*                     CALL DIVK  (2., D1, D2, D3)
*                     CALL XDIVK (2, ID1,ID2,ID3)
*                     CALL XDVRK (2, IE1,IE2,IE3)
*                     CALL ADDK  (2., X)
*
*    OUTPUTS - A1, A2, A3 = 3., 4., 5.   IA1, IA2, IA3 = 3, 4, 5
*              B1, B2, B3 =-1., 0., 1.   IB1, IB2, IB3 =-1, 0, 1
*              C1, C2, C3 = 2., 4., 6.   IC1, IC2, IC3 = 2, 4, 6
*              D1, D2, D3 = .5, 1., 1.5  ID1, ID2, ID3 = 0, 1, 1
*              IE1, IE2, IE3 = 1, 1, 2   X = 3.
*
* 2. EXAMPLES OF THE PLURAL ENTRIES
*
*    INPUTS  - SAME AS EXAMPLE 1
*
*    USAGE   -        CALL  ADDKS( 1., A1, Y1,  4., A2, Y2)
*                     CALL  SUBKS( 2., A1, Z1,  3., A2, Z2)
*                     CALL  MULKS( 3., A1, U1,  2., A2, U2)
*                     CALL  DIVKS( 4., A1, V1,  1., A2, V2)
*                     CALL XADDKS( 1 ,IA1,IY)
*                     CALL XSUBKS( 2 ,IA1,IZ)
*                     CALL XMULKS( 3 ,IA1,IU)
*                     CALL XDIVKS( 2 ,IA3,IV)
*                     CALL XDVRKS( 2 ,IA3,IW)
*
*    OUTPUTS - Y1,Y2  =  2.,6.      Z1,Z2  = -1.,-1.
*              U1,U2  =  3.,4.      V1,V2  = .25,2.0
*              IY = 2,   IZ = -1,   IU = 3,   IV = 1,   IW = 2
*
* PROGRAM FOLLOWS BELOWS
*
*
* NO TRANSFER VECTOR
       HTR     0            XR4
       BCI     1,ADDK
* PRINCIPAL ENTRY.  ADDK(C,X1,X2,...,XN) = ADDK(ARGSK)
 ADDK  STZ     ZIFK
* SECOND ENTRY.  ADDKS(C1,X1,Y1,...,CN,XN,YN) = ADDKS(ARGSKS)
 ADDKS CLA     TRA1
       TRA     SETUP
* THIRD ENTRY.  XADDK(IC,IX1,IX2,...,IXN) = XADDK(XARGSK)
 XADDK STZ     ZIFK
* FOURTH ENTRY.  XADDKS(IC1,IX1,IY1,...,ICN,IXN,IYN) = XADDKS(XARGSKS)
```

```
XADDKS CLA     TRA2
       TRA     SETUP
* FIFTH ENTRY.  SUBK(ARGSK)
 SUBK  STZ     ZIFK
* SIXTH ENTRY.  SUBKS(ARGSKS)
 SUBKS CLA     TRA3
       TRA     SETUP
* SEVENTH ENTRY.  XSUBK(XARGSK)
 XSUBK STZ     ZIFK
* EIGHTH ENTRY.  XSUBKS(XARGSKS)
XSUBKS CLA     TRA4
       TRA     SETUP
* NINTH ENTRY.  MULK(ARGSK)
 MULK  STZ     ZIFK
* TENTH ENTRY.  MULKS(ARGSKS)
 MULKS CLA     TRA5
       TRA     SETUP
* ELEVENTH ENTRY.  XMULK(XARGSK)
 XMULK STZ     ZIFK
* TWELFTH ENTRY.  XMULKS(XARGSKS)
XMULKS CLA     TRA6
       TRA     SETUP
* THIRTEENTH ENTRY.  DIVK(ARGSK)
 DIVK  STZ     ZIFK
* FOURTEENTH ENTRY.  DIVKS(ARGSKS)
 DIVKS CLA     TRA7
       TRA     SETUP
* FIFTEENTH ENTRY.  XDIVK(XARGSK)
 XDIVK STZ     ZIFK
* SIXTEENTH ENTRY.  XDIVKS(XARGSKS)
XDIVKS CLA     XCA
       TRA     SETVRY
* SEVENTEENTH ENTRY.  XDVRK(XARGSK)
 XDVRK STZ     ZIFK
* EIGHTEENTH ENTRY.  XDVRKS(XARGSKS)
XDVRKS CLA     RND
SETVRY STO     VARY
       CLA     TRA8
* SET BRANCH AT MODIFY.  THEN CHECK SINGULAR OR PLURAL
 SETUP SXD     ADDK-2,4
       STA     MODIFY
       ZET     ZIFK
       TRA     PLURAL
* SET UP FOR SINGULAR ENTRIES
       CLA*    1,4            C OR IC
       STO     CONST
       CLA     SING1          (PZE  GETX,0,1)
       LDQ     SING2          (PZE     1,0,-1)
       TXI     STA,4,-1       (SET TO PICK UP X1 FIRST)
* SET UP FOR PLURAL ENTRIES
PLURAL CLA     PLUR1          (PZE  GETC,0,2)
       LDQ     PLUR2          (PZE     3,0,-3)
 STA   STA     GETXOC
       ARS     18
       STA     GETX
       XCA
       STA     STORE
       STD     BACK
* ACQUIRE NEXT POSSIBLE TSX  X,0  AND CHECK IF IT IS.
GETSXZ CAL     1,4            A(TSX  X1,0) SINGULAR,  A(TSX  C1,0) PLURAL
       ANA     MASK           KNOCK OUT ADDRESS
       LAS     TSXZ
       TRA     LEAVE
GETXOC TRA     **             ** = GETX (SINGULAR),  = GETC (PLURAL)
* EXIT AT END OF ARGUMENT STRING.
 LEAVE SXA     ZIFK,4       RESTORE ZIFK TO NON-ZERO (PLURAL INDICATION)
       TRA     1,4
* STORE NEXT C OR IC. GET NEXT X OR IX IN AC.
* BRANCH TO MODIFY.
 GETC  CLA*    1,4            C1,C2,...
       STO     CONST
 GETX  CLA*    **,4           ** = 1 (SINGULAR),  = 2 (PLURAL)
MODIFY TRA     **             ** = MOD1,MOD2,...,MOD8
* MODIFICATION 1.  ADDK OR ADDKS
 MOD1  FAD     CONST
```

```
 * STORE RESULT.  GO BACK FOR MORE
  STORE STO*    **,4           ** = 1 (SINGULAR),   = 3 (PLURAL)
  BACK  TXI     GETSXZ,4,**    ** =-1 (SINGULAR),   =-3 (PLURAL)
 * MODIFICATION 2.  XADDK OR XADDKS
  MOD2  ADD     CONST
        TRA     STORE
 * MODIFICATION 3.  SUBK OR SUBKS
  MOD3  FSB     CONST
        TRA     STORE
 * MODIFICATION 4.  XSUBK OR XSUBKS
  MOD4  SUB     CONST
        TRA     STORE
 * MODIFICATION 5.  MULK OR MULKS
  MOD5  XCA
        FMP     CONST
        TRA     STORE
 * MODIFICATION 6.  XMULK OR XMULKS
  MOD6  XCA
        MPY     CONST
        ALS     17
        TRA     STORE
 * MODIFICATION 7.  DIVK OR DIVKS
  MOD7  NZT     CONST          BYPASS FOR CONST = 0.
        TRA     BACK
        FDP     CONST
        XCA
        TRA     STORE
 * MODIFICATION 8.  XDIVK, XDIVKS, XDVRK, OR XDVRKS
  MOD8  NZT     CONST
        TRA     BACK
        LRS     35
        DVP     CONST
 VARY   NOP                    = XCA   OR   TRA ROUND
  ALS18 ALS     18
        TRA     STORE
 * ROUNDING INSERT, COMPARES TWICE THE REMAINDER AGAINST DENOMINATOR.
  ROUND SSP
        ALS     1
        SBM     CONST
        CLM                    PREPARE FOR ROUNDING DOWN
        TMI     RXCA
        CLA     KRND           PREPARE FOR ROUNDING UP
  RXCA  XCA
        RND
        TRA     ALS18
 * CONSTANTS, TEMPORARIES
  TRA1  TRA     MOD1
  TRA2  TRA     MOD2
  TRA3  TRA     MOD3
  TRA4  TRA     MOD4
  TRA5  TRA     MOD5
  TRA6  TRA     MOD6
  TRA7  TRA     MOD7
  TRA8  TRA     MOD8
  TSXZ  TSX     0,0
  MASK  OCT     777777700000
  XCA   XCA
  RND   TRA     ROUND
  KRND  OCT     200000000000
  SING1 PZE     GETX,0,1
  SING2 PZE     1,0,32767
  PLUR1 PZE     GETC,0,2
  PLUR2 PZE     3,0,32765
  CONST PZE     **,**,**       = C OR IC,  CL OR ICL   L = 1,...,N
  ZIFK  PZE     1              SET = 0 FOR SINGULAR
        END
```

```
************************
*    ADDKS             *
************************
 REFER TO
     ADDK
```



```
************************
*    ADDKS             *
************************
 REFER TO
     ADDK
```

```
*       AMPHZ (SUBROUTINE)                10/1/64   LAST CARD IN DECK IS NO. 0250
*       FAP
*AMPHZ
        COUNT   280
        LBL     AMPHZ
        ENTRY   AMPHZ (RE,XIM,LR,AMP,PHZ,R)
        ENTRY   REIM (AMP,PHZ,LR,RE,XIM)
*
*                   ----ABSTRACT----
*
* TITLE - AMPHZ , WITH SECONDARY ENTRY POINT REIM
*       AMPLITUDE AND PHASE FROM REAL AND IMAGINARY, OR REVERSE
*
*             AMPHZ COMPUTES AN AMPLITUDE (AMP) AND PHASE (PHZ) VECTOR
*             FROM THE VECTORS OF THE REAL (RE) AND IMAGINARY (XIM)
*             PARTS.   THUS IF
*
*                            Z(J) = RE(J)+I*XIM(J)
*
*             WHERE  I = (-1)**.5
*             THEN
*
*                            AMP(J) = (RE(J)**2+XIM(J)**2)**.5
*                            PHZ(J) = ARCTAN(XIM(J)/RE(J))
*                                       (WITH PROPER QUADRANT CHOICE)
*                                   = 0.0  IF XIM=RE=0.0
*
*             PHZ(J) IS COMPUTED SUCH THAT
*                            -PI LSTHN PHZ(J) LSTHN= PI
*             THEN, IF DESIRED, A MULTIPLE OF 2PI IS ADDED TO PHZ(J) SO
*             AS TO MINIMIZE THE DIFFERENCE BETWEEN PHZ(J) AND PHZ(J-1).
*             FOR THE LATTER CONDITION, PHZ(1) WILL BE BETWEEN
*             + AND - PI.  THIS PROCESS GIVES POINTS ON THE TRUE
*             CONTINUOUS PHASE CURVE PROVIDED THE TRUE PHASE JUMPS ARE
*             LESS THAN MAGNITUDE PI.
*             PI = 3.14159265.
*
*             REIM REVERSES THE ABOVE PROCESS.  IT CALCULATES THE REAL
*             AND IMAGINARY VECTORS FROM THE AMPLITUDE AND PHASE
*             VECTORS.
*
*                            RE(J) = AMP(J)*COS(PHZ(J))
*                           XIM(J) = AMP(J)*SIN(PHZ(J))
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 149 REGISTERS
* SPEED     - ABOUT  .00050*LR  SECONDS ON THE 7094 MOD 1 FOR BOTH
*               AMPHZ AND REIM, WHERE LR IS LENGTH OF THE VECTORS.
* AUTHOR    - J.F. CLAERBOUT
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - RND
*       AND FORTRAN SYSTEM ROUTINES - ATAN,SQRT,COS,SIN
*
* FORTRAN USAGE OF AMPHZ
*     CALL AMPHZ (RE,XIM,LR,AMP,PHZ,R)
*
* INPUTS
*
*    RE(I)      I=1...LR IS FLOATING POINT VECTOR OF REAL VALUES.
*
*    XIM(I)     I=1...LR IS FLOATING POINT VECTOR OF IMAGINARY VALUES.
*
*    LR         IS FORTRAN II INTEGER.
*               MUST EXCEED 0
*
*    R          =0 INDICATES PHASE IS BETWEEN + AND - PI.
*               NOT = 0 INDICATES THAT THE PHASE IS TO BE A CONTINUOUS
*                 FUNCTION FOR WHICH THE FIRST VALUE IS BETWEEN + AND -PI
*
* OUTPUTS
*
*    AMP(I)     I=1...LR IS FLOATING POINT VECTOR OF THE AMPLITUDES.
```

```
*              EQUIVALENCE WITH RE IS ALLOWED.
*
*   PHZ(I)     I=1...LR IS FLOATING POINT VECTOR OF THE PHASES.
*              IF AMP(J)=0. THEN PHZ(J)=0.
*              EQUIVALENCE WITH XIM IS ALLOWED.
*
* FORTRAN USAGE OF REIM
*    CALL REIM (AMP,PHZ,LR,RE,XIM)
*
* INPUTS AND OUTPUTS HAVE THE SAME DEFINITIONS AS FOR AMPHZ
*   EXCEPT  AMP, PHZ, AND LR ARE INPUTS,
*           RE, AND XIM ARE OUTPUTS.
*
* EXAMPLES OF AMPHZ
*
* 1. INPUTS  - RE(1...7)  = 1.,3.,0.,-3.,-1.,-3.,0.  LR = 7  R = 0
*              XIM(1...7) = 0.,4.,1.,+4., 0.,-4.,-1.
*   OUTPUTS - AMP(1...7) = 1.,5.,1.,5.,1.,5.,1.
*              PHZ(1...7) = 0.,0.9273,1.5708,2.2143,3.1416,-2.2143,-1.57
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  R=1.
*   OUTPUTS - AMP(1...7) = 1.,5.,1.,5.,1.,5.,1.
*              PHZ(1...7) = 0.,0.9273,1.5708,2.2143,3.1416,4.0689,4.7124
*
        BSS     1
        BCI     1,AMPHZ
AMPHZ   SXD     *-2,4
        SXA     SV1,1
        CLA*    6,4
        STO     TSTCN
        CLA*    3,4           GET
        STD     A10          N
        CLA     1,4           RE
        ADD     =1
        STA     A2
        CLA     2,4           IM
        ADD     =1
        STA     A1
        CLA     4,4           AMP
        ADD     =1
        STA     A4
        CLA     5,4           PHZ
        ADD     =1
        STA     A5            THIS PHZ
        ADD     =1
        STA     A6            PREVIOUS PHZ
        STO     FIRST         STORE SOME NON-ZERO QUANTITY
        AXT     1,1           SET FOR LOOP OF LENGTH N
PIPI    ZET*    A2            IF REAL PART IS ZERO
        TRA     A1            ISN*T
        CLA     =1.57079633   SET EQUAL +PI/2 OR -PI/2
        TRA     A3
A1      CLA     **,1          IMAG
A2      FDP     **,1          REAL
        XCA
        SSP
        TSX     $ATAN,4
*       THE FOUR QUADRANT AMBIGUITY IS ELEGANTLY RESOLVED IN THE NEXT 5
*       INSTRUCTIONS
        LDQ*    A2            RIGHT OR LEFT HALF PLANE
        TQP     A3            RIGHT
        FSB     =3.14159265   LEFT
A3      LDQ*    A1
        LLS     0             UPPER OR LOWER HALF PLANE
        STO     PHZ
        LDQ*    A1          COMPUTE AMPLITUDE
        FMP*    A1
        STO     AMP
        LDQ*    A2
        FMP*    A2
        FAD     AMP
        TSX     $SQRT,4
A4      STO     **,1          STO AMPL
        TNZ     *+2           IF AMP=0.
        STZ     PHZ           SET PHZ=0.
```

```
         ZET    TSTCN         REPRESENT PHASE PI TO -PI
         TRA    CONT           OR CONTINUOUSLY
   A45   STZ    FIRST
         CLA    PHZ            STORE PHASE
   A5    STO    **,1
   A9    TXI    *+1,1,1        REPEAT LOOP
   A10   TXL    PIPI,1,**      **=N
   A11   LXD    AMPHZ-2,4
  SV1    AXT    **,1
         TRA    7,4
   CONT  ZET    FIRST
         TRA    A45            WE DONT HAVE A FIRST VALUE YET
  *       GET M=(PREV PHASE)/(2 * PI)  ROUNDED TO INTEGER
   A6    CLA    **,1      GET PREV PHASE
         FDP    =6.2831853
         XCA
         TSX    $RND,4
         XCA
         FMP    =6.2831853      2*PI*M = C(AC)
         STO    TWOPIM
  *      FORM   SS=ABSF(PHZ+2.*PI*M-PHZPRV)
  *      FORM   SM=ABSF(PHZ+2.*PI*M-PHZPRV-2.*PI)
  *      FORM   SP=ABSF(PHZ+2.*PI*M-PHZPRV+2.*PI)
         FAD    PHZ
         FSB*   A6
         STO    SS         SS STILL NEEDS ABS VALUE
         FSB    =6.2831853
         SSP
         STO    SM          GOT SM
         CLA    SS
         FAD    =6.2831853
         SSP
         STO    SP      GOT SP
         CAL    SS
         STO    SS          GOT SS
  *      FORM   PHZTRIAL=PHZ+2.*PI*M
         CLA    PHZ
         FAD    TWOPIM
         STO    PHZ
  *      WHIC IS SMALLER, SS,SP, OR SM
  *      IF SS, THEN PHASE = PHZTRIAL
  *      IF SM, THEN PHASE = PHZTRIAL - 2 PI
  *      IF SP, THEN PHASE = PHZTRIAL + 2 PI
         CLA    SS
         SUB    SM
         TPL    A7   TRA IF SS GREATER SM
         CLA    SS          SS SMALLER SM
         SUB    SP
         TMI    A5-1      SS SMALLEST, STORE PHASE
   A65   CLA    PHZ           SP SMALLEST
         FAD    =6.2831853
         TRA    A5          STORE CORRECT PHASE
   A7    CLA    SM          SM SMALLER SS
         SUB    SP
         TPL    A65         SP SMALLEST
         CLA    PHZ         SM SMALLEST
         FSB    =6.2831853
         TRA    A5      STORE CORRECT PHASE
   TSTCN PZE
  TWOPIM PZE
   SS    PZE
   SM    PZE
   SP    PZE
   FIRST PZE
   AMP   PZE
   PHZ   PZE
   REIM  SXD    AMPHZ-2,4
         SXA    R5,1
         CLA    1,4            AMP
         ADD    =1
         STA    R2
         CLA    2,4            PHASE
         ADD    =1
         STA    R1
         CLA    4,4            RE
```

```
        ADD     =1
        STA     R3
        CLA     5,4
        ADD     =1
        STA     R4            IM
        CLA*    3,4           GET N
        PDX     ,1            STORE IN IR1
R2      CLA     **,1
        STO     AMP
R1      CLA     **,1
        STO     PHZ
        TSX     $COS,4
        XCA
        FMP     AMP
R3      STO     **,1
        CLA     PHZ
        TSX     $SIN,4
        XCA
        FMP     AMP
R4      STO     **,1
        TIX     R2,1,1
R5      AXT     **,1
        LXD     AMPHZ-2,4
        TRA     6,4
ORF     OCT     233000000000
        END
```

```
*      ARBCOL (SUBROUTINE)                9/9/64   LAST CARD IN DECK IS NO. 0270
*      FAP
*ARBCOL
       COUNT   200
       LBL     ARBCOL
       ENTRY   ARBCOL (FOFIJ,LI,LJ,IDIMEN,FJCOL,COL)
*
*
*                      ----ABSTRACT----
*
* TITLE - ARBCOL
*        FIND A MATRIX COLUMN WITH ARBITRARY INDEX BY INTERPOLATION
*
*              ARBCOL IS GIVEN A MATRIX AND A FLOATING POINT NUMBER
*              (GENERALLY NOT A WHOLE NUMBER) REPRESENTING A DESIRED
*              COLUMN NUMBER IN THE MATRIX.  THE FOUR COLUMNS WHICH
*              ARE CLOSEST IN NUMBER TO THE DESIRED COLUMN NUMBER ARE
*              SUBJECTED TO CUBIC INTERPOLATION TO YIELD THE
*              INTERPOLATED COLUMN.
*
*              ARBCOL REDUCES THE DEGREE OF INTERPOLATION IN THE
*              CASES OF MATRICES WITH ONLY 3, 2, OR 1 COLUMNS.
*
*              THE PROCEDURE USED IS TO FIND THE PROPER INTERPOLATION
*              OPERATOR FOR THE GIVEN COLUMN NUMBER AND THEN APPLY IT
*              IN A HIGH SPEED LOOP ON THE ROW INDEX.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE    - 129 REGISTERS
* SPEED      - TAKES ABOUT  530 + 90*N  MACHINES CYCLES ON THE 7090,
*                 WHERE  N  =  NO. ROWS IN THE MATRIX.
* AUTHOR     - S.M. SIMPSON, MARCH 1964
*
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - INTOPR
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL ARBCOL(FOFIJ,LI,LJ,IDIMEN,FJCOL,COL)
*
* INPUTS
*
*    FOFIJ(I,J)  I=1...LI, J=1...LJ  IS A MATRIX OF FLOATING POINT
*                   ELEMENTS.
*
*    LI          MUST EXCEED ZERO
*
*    LJ          MUST EXCEED ZERO
*
*    IDIMEN      IS THE DIMENSION, IN THE CALLING PROGRAM, OF THE
*                   INDEX I OF FOFIJ(I,J)
*                MUST BE GRTHN= LI
*
*    FJCOL       IS THE FLOATING POINT COLUMN NUMBER FOR WHICH AN
*                   INTERPOLATED COLUMN IS DESIRED
*                MUST BE GRTHN= 1.0, AND LSTHN FLOATF(LJ+1)
*
* OUTPUTS        STRAIGHT RETURN WITH NO OUTPUTS IF LI, LJ, IDIMEN, OR
*                FJCOL IS ILLEGAL
*
*    COL(I)      I=1...LI  IS THE INTERPOLATED COLUMN
*
*
* EXAMPLES
*
* 1.  THIS EXAMPLE INTERPOLATES ALL HALF-INDEX AND FULL-INDEX COLUMNS
*      IN A 1-COLUMN, A 2-COLUMN,..., AND A 5-COLUMN MATRIX. IT ALSO
*      SHOWS THAT NO INTERPOLATION RESULTS FOR ILLEGAL FJCOL VALUES.
*
*    INPUTS  - FOFIJ(1,2,3,,1,2,3,4,5) =
*                   0.,0.,0.,, 0.,1.,2.,, 0.,2.,4.,, 0.,3.,6.,, 0.,4.,8.
```

```
*              LI=3   IDIMEN=10   COL(1...3,,1...9,,1...5) = -99.,...
*
*   USAGE   -        DIMENSION FOFIJ(10,5),COL(3,9,5)
*                    DO 10    LJ=1,5
*                    DO 10    J=1,9
*                    FJCOL = (FLOATF(J+1))/2.0
*               10   CALL ARBCOL(FOFIJ,LI,LJ,IDIMEN,FJCOL,COL(1,J,LJ))
*
*   OUTPUTS - COL(1...3,1,LJ) = 0., 0., 0.   FOR LJ=1...5
*             COL(1...3,2,1)  = 0., 0., 0.
*             COL(1...3,2,LJ) = 0., .5, 1.   FOR LJ=2...5
*             COL(1...3,3,LJ) = 0., 1., 2.   FOR LJ=2...5
*             COL(1...3,4,LJ) = 0.,1.5, 3.   FOR LJ=2...5
*             COL(1...3,5,LJ) = 0., 2., 4.   FOR LJ=3...5
*             COL(1...3,6,LJ) = 0.,2.5, 5.   FOR LJ=3...5
*             COL(1...3,7,LJ) = 0., 3., 6.   FOR LJ=4,5
*             COL(1...3,8,LJ) = 0.,3.5, 7.   FOR LJ,4,5
*             COL(1...3,9,5)  = 0., 4., 8.
*             COL(1...3,J,LJ) = -99.,-99.,-99.  WHENEVER J GRTHN 2*LJ
*
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS INTOPR ONLY
       HTR      0             XR1
       HTR      0             XR4
       BCI      1,ARBCOL
*
* ONLY ENTRY.  ARBCOL(FOFIJ,LI,LJ,IDIMEN,FJCOL,COL)
*
ARBCOL SXD      ARBCOL-2,4
       SXD      ARBCOL-3,1
*
* CHECK LI, LJ AND IDIMEN
*
       CLA*     2,4           LI
       TMI      LEAVE
       TZE      LEAVE
       PDX      0,1           (FOR LOOP AT STZ)
       CLA*     3,4           LJ
       TMI      LEAVE
       TZE      LEAVE
       CLA*     4,4           IDIMEN
       SUB*     2,4           MINUS LI
       TMI      LEAVE
       ADD*     2,4
       ARS      18
       STO      IDIM
*
* FIND JCOL = FJCOL ROUNDED DOWN EXCEPT IN THE CASE THAT
*                  FJCOL = FLOATF(LJ) AND LJ EXCEEDS 1
*   IN WHICH CASE SET JCOL = LJ-1
*
       CLA*     5,4           FJCOL
       UFA      K233
       LRS      0
       ANA      KDECR
       LLS      0
       ALS      18
       CAS*     3,4
       TRA      LEAVE         EXCEEDS LJ
       TRA      LJCK          EQUALS LJ
       TMI      LEAVE
       TZE      LEAVE
       TRA      JCOK
 LJCK  SUB      KD1
       TNZ      JCOK
       ADD      KD1           (EQUALS LJ EQUALS 1)
*
* THEN FORM  X = FJCOL - FLOATF(JCOL)
*
JCOK   STO      JCOL
       LRS      18
       ORA      K233
       FAD      K233
```

```
       CHS
       FAD*    5,4
       STO     X
*
* NOW SOME ADDRESS SETUPS
*
       CLA     JCOL
       SUB     KD2
       XCA
       MPY     IDIM
       LLS     17
       CHS
       ADD     K1
       ADD     1,4            A(FOFIJ)-(JCOL-2)*IDIMEN+1
       STA     LDQ1
       SUB     IDIM
       STA     LDQ2
       SUB     IDIM
       STA     LDQ3
       SUB     IDIM
       STA     LDQ4
       CLA     6,4            A(COL)
       ADD     K1
       STA     STORE
*
* SET UP AS THOUGH NDATA=4 THEN TEST JCOL
*
       CLA     FMP1
       STA     TSXOP
       CLS     K1L
       STO     XLO
       CLA     KD4
       STO     NDATA
       CLA     JCOL
       SUB     KD1
       TZE     JCOL1
*
* SETTINGS FOR JCOL EXCEEDING 1 ARE ALL MADE UNLESS JCOL+1=LJ
*
       ADD     KD2            JCOL+1
       SUB*    3,4            COMPARE WITH LJ
       TZE     NDATA3         SAME, CHANGE NDATA TO 3
       TRA     GETOP          SMALLER, ALL SETTINGS OK
*
* SETTINGS IF JCOL=1
*
 JCOL1 CLA     FMP2           A(FMP2) = OPER2
       STA     TSXOP
       STZ     XLO            XLO=0.0
       CLA*    3,4            TRIAL SET NDATA
       STO     NDATA          =LJ (OK IF LJ=1 OR 2)
       SUB     KD3            THEN TEST
       TMI     STZOP          (NEGATIVE IF LJ=1 OR 2)
*
* CHANGE NDATA TO 3 FOR INTERPOLATING NEAR RIGHTMOST COLUMN.
*
NDATA3 CLA     KD3
       STO     NDATA
*
* CLEAR OPER1,3,4, IF NDATA IS NOT 4
*
 STZOP STZ     OPER1
       STZ     OPER3
       STZ     OPER4
*
* GO GET THE OPERATOR
*
 GETOP TSX     $INTOPR,4
       TSX     NDATA,0              1       2       3       3       4
       TSX     XLO,0              0.0     0.0     0.0    -1.0    -1.0
       TSX     K1L,0          DELX
       TSX     X,0
 TSXOP TSX     **,0           ** =  OPER2   OPER2   OPER2   OPER1   OPER1
       LXD     ARBCOL-2,4
*
```

```
* LOOP FORMING INTERPOLATES,  XR1=LI,LI-1,...,1
*
 STZ   STZ     TEMP
 LDQ1  LDQ     **,1          ** = A(FOFIJ)-(JCOL-2)*IDIMEN+1
 FMP1  FMP     OPER1
       FAD     TEMP
       STO     TEMP
 LDQ2  LDQ     **,1          ** = DITTO LDQ1 MINUS IDIMEN
 FMP2  FMP     OPER2
       FAD     TEMP
       STO     TFMP
 LDQ3  LDQ     **,1          ** = DITTO LDQ2 MINUS IDIMEN
       FMP     OPER3
       FAD     TEMP
       STO     TEMP
 LDQ4  LDQ     **,1          ** = DITTO LDQ3 MINUS IDIMEN
       FMP     OPER4
       FAD     TEMP
 STORE STO     **,1          ** = A(COL)+1
       TIX     STZ,1,1
*
* EXIT
*
 LEAVE LXD     ARBCOL-3,1
       TRA     7,4
*
* CONSTANTS, VARIABLES
*
 K1    PZE     1
 KD1   PZE     0,0,1
 KD2   PZE     0,0,2
 KD3   PZE     0,0,3
 KD4   PZE     0,0,4
 K233  OCT     233000000000
 KDECR OCT     000000377777
 K1L   DEC     1.0
 IDIM  PZE     **
 JCOL  PZE     0,0,**
 NDATA PZE     0,0,**        1, 2, 3, OR 4
 XLO   PZE     **,**,**      NORMALLY = -1.0  (MAY BE 0.0)
 X     PZE     **,**,**      EQUALS FJCOL-FLOATF(JCOL)
 TEMP  PZE     **,**,**
 OPER4 PZE     **,**,**      MULTIPLIES COLUMN NO. JCOL+2
 OPER3 PZE     **,**,**      MULTIPLIES COLUMN NO. JCOL+1
 OPER2 PZE     **,**,**      MULTIPLIES COLUMN NO. JCOL
 OPER1 PZE     **,**,**      MULTIPLIES COLUMN NO. JCOL-1
       END
```

```
*      ARCTAN (FUNCTION)                9/4/64   LAST CARD IN DECK IS NO. 0091
*      FAP
*ARCTAN
       COUNT    30
       LBL      ARCTAN
       ENTRY    ARCTAN F(X,Y)
*
*
*                    ----ABSTRACT----
*
* TITLE - ARCTAN
*        ARCTANGENT FUNCTION
*
*              ARCTAN FINDS THE ANGLE IN RADIANS ASSOCIATED WITH AN
*              X AND Y COORDINATE SUCH THAT
*
*                 -3.14159265  LSTHN  ANGLE  LSTHN=  3.14159265
*
*
* LANGUAGE    - FAP FUNCTION (FORTRAN II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 29 REGISTERS
* SPEED       - ABOUT 250 MACHINE CYCLES ON 7090.
* AUTHOR      - R.A. WIGGINS     MARCH 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - ATAN
*
* FORTRAN USAGE
*     ANGLE = ARCTANF(X,Y)
*
*
* INPUTS
*
*    X         IS THE ABSCISSA OF THE POINT
*
*    Y         IS THE ORDINATE OF THE POINT
*
*
* OUTPUTS
*
*    ANGLE     IS THE ANGLE IN RADIANS FROM THE POSITIVE X-AXIS TO THE
*                POINT,  =  ARCTANGENT OF  Y/X .
*
*
* EXAMPLES
*
* 1. USAGE   -        ANGLE1 = ARCTANF ( -2.,  0.)
*                     ANGLE2 = ARCTANF ( -2.,  1.)
*                     ANGLE3 = ARCTANF (  0.,  1.)
*                     ANGLE4 = ARCTANF (  2.,  1.)
*                     ANGLE5 = ARCTANF (  2.,  0.)
*                     ANGLE6 = ARCTANF (  2., -1.)
*                     ANGLE7 = ARCTANF (  0., -1.)
*                     ANGLE8 = ARCTANF ( -2., -1.)
*    OUTPUTS - ANGLE1 = 3.1416   ANGLE2 = 2.6779   ANGLE3 = 1.5708
*              ANGLE4 = 0.4636   ANGLE5 = 0.       ANGLE6 =-0.4636
*              ANGLE7 =-1.5708   ANGLE8 =-2.6779
*
*
* PROGRAM FOLLOWS BELOW
*
 XR4   HPR      0
       BCI      1,ARCTAN
ARCTAN SXD      XR4,4
       STO      X1
       TZE      A
       TMI      A1
       STZ      CRRCT
       TRA      A3
 A     CLA      =1.57079632
       TRA      A1+1
```

```
************************
*    ARCTAN          *
************************
(PAGE  2)
```


```
      A1     CLA     =3.14159265
             TQP     A2
             SSM
      A2     STO     CRRCT
             PXD     ,0
             NZT     X1
             TRA     ADD
      A3     XCA
             FDP     X1
             XCA
             TSX     $ATAN,4
      ADD    FAD     CRRCT
             LXD     XR4,4
             TRA     1,4
      CRRCT  PZE     0
      X1     PZE     0
             END
```

```
************************
*    ARG               *
************************
 REFER TO
     LOCATE
```



```
************************
*    ARG               *
************************
 REFER TO
     LOCATE
```

```
*       ASPECT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0535
*       FAP
*ASPECT
        COUNT   560
        LBL     ASPECT
        ENTRY   ASPECT (ACOR,N,COSTAB,M,JMIN,JMAX,TYPE,SPECT,SPACE,
*                       ISCALE,ERR)
*
*                    ----ABSTRACT----
*
* TITLE - ASPECT
*        FAST COSINE TRANSFORMS OF ONE-SIDED AUTOCORRELATIONS
*
*             ASPECT PRODUCES A HI-SPEED POWER- OR ENERGY-DENSITY
*             SPECTRUM (OR PORTION THEREOF) FROM AN N-LAG AUTOCORREL-
*             ATION FUNCTION, AC(I) I=0,1,...,N, ACCORDING TO
*
*                                    N
*                SP(J)  =  AC(0) + 2*SUM ( AC(I)*COS(I*J*(PI/M)) )
*                                   I=1
*
*                   FOR J = JMIN,JMIN+1,...,JMAX
*               WHERE
*                       PI = 3.14159265
*                       N,M,JMIN AND JMAX ARE INPUT PARAMETERS
*                       COS(J*(PI/M)) J=0,1,...,M IS AN INPUT TABLE
*                       0 LSTHN= JMIN LSTHN JMAX LSTHN= M
*
*             SPEED IS ATTAINED BY
*                  1. (FOR M LSTHN=N)
*                     - COLLAPSING AC(I) INTO THE RANGE 0 TO 2M
*                     - SPLITTING THE COLLAPSED
*                         CORRELATION INTO ODD AND EVEN PARTS AND
*                         SUBPARTS (ONLY 2 OF THESE 4 ARE USED)
*                  2. USING THE HIGH-SPEED LOOPING LOGIC OF SUBROUTINE
*                     COSP TO PERFORM THE TRANSFORMS OF THE SHORTENED
*                     PARTS (LENGTH = M/2)
*
*             THE AUTOCORRELATION MAY BE FLOATING POINT OR FIXED
*             (COMPUTATIONS SLIGHTLY FASTER FOR FIXED POINT)
*
*             2*M+1 TEMPORARY REGISTERS ARE NEEDED UNLESS USER IS
*             WILLING TO SACRIFICE THE AUTOCORRELATION FOR THIS PURPOSE
*             (TEMPORARIES NOT REQUIRED FOR M GRTHN N)
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 278 REGISTERS
* SPEED     - FIXED PT, M LSTHN= N -  17*M*(JMAX-JMIN+1)  MACH. CYCLES
*             FLTG. PT, M LSTHN= N -  19*M*(JMAX-JMIN+1)  MACH. CYCLES
*             (FOR M GRTHN N SUBSTITUTE 2N FOR M IN ABOVE FORMULAS)
* AUTHOR    - S.M. SIMPSON JR, OCT, 1961
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   COLAPS, COSP, DUBLX, DUBLL,
*                                       SPLIT, RVPRTS
*      AND FORTRAN SYSTEM ROUTINES -    NONE
*
* FORTRAN USAGE
*     CALL ASPECT(ACOR,N,COSTAB,M,JMIN,JMAX,TYPE,SPECT,SPACE,ISCALE,ERR)
*
* INPUTS
*
*    ACOR(I)    I=1...N+1  CONTAINS AC(J) J=0,1,...,N
*                   ACOR IS FIXED OR FLTG AS SPECIFIED BY TYPE
*
*    N          MUST EXCEED ZERO
*
*    COSTAB(I) I=1...M+1  CONTAINS COS(J*PI/M)  J=0,1,...,M
*                 COSTAB IS FIXED OR FLTG AS SPECIFIED BY TYPE
*                 IF FIXED PT IT IS ASSUMED THAT THE BINARY POINT IS
*                 BETWEEN THE SIGN BIT AND BIT 1 SO THAT VALUES =+1. AND
*                 -1. SHOULD BE ENTERED AS OCT 377777777777 AND
*                 OCT 777777777777 RESPECTIVELY. THE BINARY POINT OF
```

```
      *                ACOR IS IMMATERIAL BUT ACCURACY IS GREATER FOR FEWER
      *                LEADING ZEROES.
      *
      *    M         MUST EXCEED ZERO
      *
      *    JMIN      DEFINES LOWEST MULTIPLE OF FUNDAMENTAL DESIRED
      *              MUST BE GRTHN= 0 AND LSTHN JMAX
      *
      *    JMAX      DEFINES HIGHEST MULTIPLE OF FUNDAMENTAL DESIRED
      *              MUST BE GRTHN JMIN AND LSTHN= M
      *
      *    TYPE      = 0.0 SIGNIFIES ACOR AND COSTAB ARE FIXED POINT
      *              NOT = 0.0 MEANS ACOR AND COSTAB ARE FLTG. POINT
      *
      *    SPACE(I)  I=1...2*M+1  MUST BE AVAILABLE FOR TEMPORARY USE IF
      *                   M IS LSTHN= N. SPACE(I) NOT USED FOR M GRTHN N.
      *              EQUIVALENCE(SPACE,ACOR) IS PERMITTED (AC(I) WILL BE LOST)
      *
      * OUTPUTS
      *
      *    SPECT(I)  I=1...JMAX-JMIN+1 WILL CONTAIN SP(J) J=JMIN...JMAX AS
      *                 DEFINED IN ABSTRACT. (IT IS FIXED OR FLOATING
      *                 ACCORDING TO TYPE)
      *
      *    ISCALE    IS NOT USED FOR FLOATING POINT DATA
      *              IS A SCALE FACTOR FOR FIXED POINT RESULTS, DETERMINED
      *                 BY ASPECT SO AS TO AVOID OVERFLOW.
      *                 =0 MEANS BINARY POINT OF SP(J) SAME AS AC(J)
      *                 NOT = 0 MEANS BINARY POINT OF SP(J) IS ISCALE BITS
      *                         TO THE RIGHT OF BINARY POINT OF AC(J)
      *
      *    ERR       = 0.0 NORMAL
      *              = 1.0 IF N,M,JMIN OR JMAX IS ILLEGAL
      *
      * EXAMPLES
      *
      * 1. COMPLETE SPECTRUM, NOT TRYING TO SAVE SPACE, FIXED OR FLOATING
      *    INPUTS  - ACOR(1...4) = 2.,2.,3.,4.  IACOR(1...4) = 200,200,300,400
      *              COSTAB(1...3) = 1.0,0.0,-1.0    N=3   M=2
      *              COSTBL(1...3)=OCT377777777777,000000000000,777777777777
      *              JMIN = 0 ,    JMAX = 2
      *    USAGE   -       CALL ASPECT(ACOR,N,COSTAB,M,JMIN,JMAX,1.0,SPECT,
      *                                SPACE,DUMMY,ERR1)
      *                    CALL ASPECT(IACOR,N,COSTBL,M,JMIN,JMAX,0.,ISPECT,
      *                                SPACE,ISCALE,ERR2)
      *    OUTPUTS - ERR1 = ERR2 = 0.
      *              SPECT(1...3)=20.,-4.,-4.  ISPECT(1...3)=2000,-400,-400
      *              ISCALE = 0
      *
      * 2. USE OF SPACE SAVING FEATURE
      *    INPUTS  - SAME AS EXAMPLE 1.
      *    USAGE   -       CALL ASPECT(ACOR,N,COSTAB,M,JMIN,JMAX,1.0,SPECT,
      *                                ACOR,DUMMY,ERR)
      *                    CALL ASPECT(IACOR,N,COSTBL,M,JMIN,JMAX,0.0,ISPECT,
      *                                IACOR,ISCALE,ERR)
      *    OUTPUTS - SAME AS EXAMPLE 1. (BUT ACOR AND IACOR ARE DESTROYED)
      *
      * 3. PARTIAL SPECTRUM
      *    INPUTS  - SAME AS EXAMPLE 1. EXCEPT JMIN=1
      *    USAGE   - SAME AS EXAMPLE 1.
      *    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT  SPECT(1...2)=-4.,-4.
      *                                         ISPECT(1...2)=-400,-400
      *
      * 4. FINER GRAINED SPECTRUM , M GRTHN N , FLTG PT
      *    INPUTS  - SAME AS EXAMPLE 1. EXCEPT
      *                COSTAB(1...5)=1.0,.70711,0.0,-.70711,-1.0   M=4
      *    USAGE   - SAME AS FIRST CALL IN EXAMPLE 1.
      *    OUTPUTS - ERR=0.  SPECT(1...5) = 20.,-.82844,-4.,4.82844,-4.
      *
      * 5. FIXED POINT CASE INVOLVING SCALING
      *    INPUTS  - SAME AS EXAMPLE 1. EXCEPT IACOR(1...4) =
      *                   20000,20000,30000,40000
      *    USAGE   - SAME AS SECOND CALL IN EXAMPLE 1.
      *    OUTPUTS - ERR2=0.  ISPECT=100000,-20000,-20000  ISCALE=1  (I.E.
```

```
      *               (SPECT VALUES SHOULD BE DOUBLED FOR TRUE SCALE)
      *
      * 6. ERROR EXITS (WITH NO COMPUTATION)
      *    USAGE   -       CALL ASPECT(ACOR,-1,COSTAB,3,0,3,1.0,SPECT,ACOR,
      *                               DUMMY,ERR1)
      *                    CALL ASPECT(ACOR,2,COSTAB,0,0,3,1.0,SPECT,ACOR,
      *                               DUMMY,ERR2)
      *                    CALL ASPECT(ACOR,2,COSTAB,3,-1,3,1.0,SPECT,ACOR,
      *                               DUMMY,ERR3)
      *                    CALL ASPECT(ACOR,2,COSTAB,3,0,4,1.0,SPECT,ACOR,
      *                               DUMMY,ERR4)
      *                    CALL ASPECT(ACOR,2,COSTAB,3,2,2,1.0,SPECT,ACOR,
      *                               DUMMY,ERR5)
      *    OUTPUTS - ERR1=1. (ILLEGAL N)  ERR2=1. (ILLEGAL M)
      *              ERR3=1. (ILLEGAL JMIN)  ERR4=1. (ILLEGAL JMAX)
      *              ERR5=1. (ILLEGAL JMAX)
      *
      *
      * PROGRAM FOLLOWS BELOW
      *         NOTATION DIFFERENCES IN PROGRAM NOTES ARE
      *              AACC=ACOR
      *              SCALE=ISCALE
      *
              HTR      0
              BCI      1,ASPECT
       ASPECT SXD      *-2,4
              SXA      LV+1,1
              SXA      LV+2,2
       *MAKE PARTIAL ARGUMENT MAP (CHECKING N,M,JMIN)
         A1   CLA      1,4            AACC
              STA      T1
              CLA*     2,4            N
              TMI      A2A
              TZE      A2A
              STD      T2
              CLA      3,4            COSTAB
              STA      T3
              CLA*     4,4            M
              TMI      A2A
              TZE      A2A
              STD      T4
              CLA*     5,4            JMIN
              TMI      A2A
              STD      T5
              CLA*     6,4            JMAX
              STD      T6
              CLA*     7,4            CONTENTS OF TYPE
              STO      T7
              CLA      8,4            SPECT
              STA      T8
              CLA      9,4            SPACE
              STA      T9
              CLA      10,4           SCALE
              STA      T10
       *CHECK LEGALITIES, JMIN LESS THAN JMAX LESS THAN OR=M
         A2   CLA      T6             JMAX
              CAS      T5             JMIN
              TRA      A2B            OK
              NOP                     NO GOOD
         A2A  CLA      KL1            NO GOOD
              STO*     11,4           SET ERR INDICATOR
              TRA      LV             EXIT
         A2B  CAS      T4             M
              TRA      A2A            NO GOOD
              NOP                     OK
              STZ*     11,4           OK SET ERR=0.0
       *IF OK SET UP CONSTANTS
         A3   CLA      T4             M
              ADD      KD1            M+1
              STD      T15
              ADD      T4             2M+1
              STD      T16
              SUB      KD1            2M
              STD      T17
              CLA      T2             N
```

```
         ADD    KD1          N+1
         STD    T18
         CLA    T6           JMAX
         SUB    T5           JMAX-JMIN
         ADD    KD1          JMAX-JMIN+1
         STD    T19
         CLA    T4           M/2 FOR M EVEN
         ARS    1            SET P=
         STD    T14          (M-1)/2 FOR M ODD
*NOW ADDRESSES
  A4     CLA    T1           AACC
         STA    A6
         STA    A7
         STA    A8
         STA    A9
         STA    A23
         STA    A26
         STA    A27
         STA    A28
         STA    A29
         STA    A30
         STA    A31
         STA    A34
         STA    A35
         STA    A37
         STA    A38
         ADD    K1           AACC+1
         STA    A11
         STA    A19
         STA    A21
         CLA    T10          SCALE
         STA    A18
         CLA    T9           SPACE
         STA    A24
         STA    A44
         STA    A48
         STA    A49
         STA    A50
         STA    A52
         STA    A53
         STA    A55
         STA    A56
         ALS    18
         SUB    T17          SPACE-2M
         ARS    18
         STA    A40
         ALS    18
         ADD    T4           SPACE-M
         SUB    KD1          SPACE-M-1
         ARS    18
         STA    A51
         STA    A55B
         CLA    T9           SPACE
         ALS    18
         SUB    T14          SPACE-P
         SUB    KD1          SPACE-P-1
         ARS    18
         STA    A54
         STA    A55A
         STA    A57
         CLA    T3           COSTAB
         STA    A32
         STA    A58
         CLA    T8           SPECT
         STA    A33
         STA    A36
         STA    A39
         STA    A59
*WHEN ALL SET  UP BEGIN BY DIVIDING AUTOCOR OF ZERO BY 2
  A5     ZET    T7           T7=CONTENTS OF TYPE
         TRA    A8
*FIXED
  A6     CLA    **           (**=AACC)
         ARS    1
  A7     STO    **           (**=AACC)
```

****
* ASPECT *
****
(PAGE 5)

```
       TRA    A10
*FLOATING
  A8   CLA    **              (**=AACC)
       FDP    KL2
  A9   STQ    **              (**=AACC)
       TRA    A22             AVOID SCALING CHECK
*IF DATA IS FIXED POINT DECIDE IF IT NEEDS DOWN-SCALING
*TO PREVENT OVERFLOW IN THE COSINE TRANSFORM.  IT WILL BE
*DOWN SCALED IF TWICE THE SUM OF THE MAGNITUDES OF THE
*CORRELATION CFROM LAG 0 TO N) OVERFLOWS.
*MARK  SUMMATION
  A10  STZ    SUMHI
       TOV    *+1
       CLA    K0
       LXD    T18,4           T18=PZE 0,0,N+1
  A11  ADM    **,4            (**=AACC+1)
       TOV    A13
  A12  TIX    A11,4,1
       TRA    A14
*ADD L TO SUMHI FOR EACH OVERFLOW, AND GO BACK
  A13  XCA
       CLA    K1
       ADD    SUMHI
       STO    SUMHI
       XCA
       TRA    A12
*WHEN DONE CHECK IF SUMHI ZERO
  A14  ZET    SUMHI
       TRA    A16             THERE WAS OVERFLOW
*FOR SUMHI ZERO CHECK BIT1 OF SUM IN AC
       ALS    1
       TOV    A15             YES
       CLA    K0 NO SEALING NEEDED
       TRA    A17
*IF BIT 1 IS 1 WE NEED TO SCALE DATA DOWN 1 BIT
  A15  CLA    K1
       TRA    A17
*IF OVERFLOW IN SUMHI WE NEED TO SCALE DOWN BY C(SUMHI)+1
  A16  CLA    SUMHI
       ADD    K1
*SET SCALE CONSTANT AND THEN DO IT (UNLESS SCALE IS ZERO)
  A17  STA    A20
       ALS    18
  A18  STO    **              (**=SCALE)
       TZE    A22
*SCALE DOWN
       LXD    T18,4           T18=PZE 0,0,N+1
  A19  CLA    **,4            (**=AACC+1)
  A20  ARS    **              (**=SCALE CONSTANT)
  A21  STO    **,4            (**=AACC+1)
       TIX    A19,4,1
*CHECK IF COLLAPSING IS VALID (ONLY FOR M LESS THAN OR =N)
  A22  CLA    T4              T4=PZE 0,0,M
       CAS    T2              T2=PZE 0,0,N
       TRA    CSP2            NOT VALID
       NOP                    VALID
*IF VALID GO DO IT (NOTE COLAPS FILLS IN ZEROS IF N LESS THAN 2M)
 CLPS  TSX    $COLAPS,4
  A23  TSX    **              (**=AACC)
       TSX    T18             T18=PZE 0,0,N+1
       TSX    T7              T7=CONTENTS OF TYPE
  A24  TSX    **              (**=SPACE)
       TSX    T17             T17=PZE 0,0,2M
*THEN RESTORE THE AUTOCOR OF ZERO LAG TO ITS ORIGINAL VALUE
*UNLESS THE USER HAD US COLLAPSE IT ON TOP OF ITSELF (SPACE=AACC)
  A25  CLA    T1              T1=PZE AACC
       CAS    T9              T9=PZE SPACE
       TRA    *+2             OK TO RESTORE
       TRA    A40             AVOID RESTORING AC(0)
*RESTORE FIXED OR FLOATING
       ZET    T7              T7=CONTENTS OF TYPE
       TRA    A28             FLOATING
*FIXED
 A26   CLA    **              (**=AACC)
       ALS    1
```

```
  A27  STO     **            (**=AACC)
       TRA     A40
*FLOATING
  A28  LDQ     **            (**=AACC)
       FMP     KL2
  A29  STO     **            (**=AACC)
       TRA     A40
*IF COLLAPSING IS NOT VALID COMPUTE SPECTRUM DIRECTLY FROM
*AACC, THEN DOUBLE THE SPECTRUM, RESTORE AC(0) AND EXIT
*(DONT WORRY ABOUT AC(0) SINCE SPACE WAS NOT USED)
 CSP2  TSX     $COSP,4
  A30  TSX     **            (**=AACC)
  A31  TSX     **            (**=AACC)
       TSX     T2            T2=PZE 0,0,N
  A32  TSX     **            (**=COSTAB)
       TSX     T4            T4=PZE 0,0,M
       TSX     T5            T5=PZE 0,0,JMIN
       TSX     T6            T6=PZE 0,0,JMAX
       TSX     T7            T7=CONTENTS OF TYPE
  A33  TSX     **            (**=SPECT)
*FIXED OR FLOATING
       ZET     T7
       TRA     A37           FLOATING
*FIXED
  A34  CLA     **            (**=AACC)
       ALS     1
  A35  STO     **            (**=AACC)
  DBX1 TSX     DUBLX,4
  A36  TSX     **            (**=SPECT)
       TSX     T19           T19=JMAX-JMIN+1
       TRA     LV            GO EXIT
*FLOATING
  A37  LDQ     **            (**=AACC)
       FMP     KL2
  A38  STO     **            (**=AACC)
  DBL1 TSX     DUBLL,4
  A39  TSX     **            (**=SPECT)
       TSX     T19           T19=JMAX-JMIN+1
       TRA     LV            GO EXIT
*IF COLLAPSING WAS PERFORMED
*THEN END-POINT ADJUST THE COLLAPSED CORRELATION AND DOUBLE IT
  A40  STZ     **            (**=SPACE-2M)
       ZET     T7            T7=CONTENTS OF TYPE
       TRA     DBL2
*FIXED POINT
 DBX2  TSX     $DUBLX,4
  A44  TSX     **            (**=SPACE)
       TSX     T16           T16=PZE 0,0,2M+1
       TRA     SPLT1
*FLOATING POINT
  DBL2 TSX     $DUBLL,4
  A48  TSX     **            (**=SPACE)
       TSX     T16           T16=PZE 0,0,2M+1
*NOW SPLIT THE ADJUSTED COLLAPSED AUTOCORRELATION ON TOP OF ITSELF
 SPLT1 TSX     $SPLIT,4
  A49  TSX     **            (**=SPACE)
       TSX     T16           T16=PZE 0,0,2M+1
       TSX     T7            T7=CONTENTS OF TYPE
  A50  TSX     **            (**=SPACE)
  A51  TSX     **            (**=SPACE-M-1)
*NOW RESPLIT THE SYMMETRIC PART ON TOP OF ITSELF
 SPLT2 TSX     $SPLIT,4
  A52  TSX     **            (**=SPACE)
       TSX     T15           T15=PZE 0,0,M+1
       TSX     T7            T7=CONTENTS OF TYPE
  A53  TSX     **            (**=SPACE)
  A54  TSX     **            (**=SPACE-P-1)
*REVERSE THE RESPLIT PARTS AND SET AS(P)=0 FOR COSP
 REV   TSX     $RVPRTS,4
  A55  TSX     **            (**=SPACE)
  A55A TSX     **            (**=SPACE-P-1)
       TSX     T15           (T15=PZE 0,0,M+1)
  A55B STZ     **            (**=SPACE-M-1)
*NOW COMPUTE SPECTRUM FROM THE RESPLIT PARTS
 CSP1  TSX     $COSP,4
```

```
  A56  TSX     **            (**=SPACE)
  A57  TSX     **            (**=SPACE-P-1)
       TSX     T14           T14=PZE 0,0,P
  A58  TSX     **            (**=COSTAB)
       TSX     T4            T4=PZE 0,0,M
       TSX     T5            T5=PZE 0,0,JMIN
       TSX     T6            T6=PZE 0,0,JMAX
       TSX     T7            T7=CONTENTS OF TYPE
  A59  TSX     **            (**=SPECT)
*FINAL EXIT
  LV   LXD     ASPECT-2,4
       AXT     **,1          (**=XR1)
       AXT     **,2          (**=XR2)
       TRA     12,4
*TEMPORARIES, ETC
*INPUT ARGUMENTS
  T1   PZE     **            (**=AACC)
  T2   PZE     0,0,**        (**=N)
  T3   PZE     **            (**=COSTAB)
  T4   PZE     0,0,**        (**=M)
  T5   PZE     0,0,**        (**=JMIN)
  T6   PZE     0,0,**        (**=JMAX)
  T7   PZE     **            (**=CONTENTS OF TYPE=0.0(FXD)=1.0(FLTG))
  T8   PZE     **            (**=SPECT)
  T9   PZE     **            (**=SPACE)
  T10  PZE     **            (**=SCALE)
  T14  PZE     0,0,**        (**=P FOR COSP=M/2 OR (M-1)/2)
  T15  PZE     0,0,**        (**=M+1 FOR REV, SPLT2,)
  T16  PZE     0,0,**        (**=SM+1 FOR SPLT2, DUBLX, DUBLL,)
  T17  PZE     0,0,**        (**=2M FOR CLPS)
  T18  PZE     0,0,**        (**=N+1 FOR CLPS, SCALING,)
  T19  PZE     0,0,**        (**=JMAX-JMIN+1)
 SUMHI PZE     **            OVERFLOW REG FOR COR. MAGN. SUM
  K0   PZE     0
  K1   PZE     1
  KD1  PZE     0,0,1
  KL1  DEC     1.0
  KL2  DEC     2.0
*
*
*USE OF SPACE WHEN M IS EVEN  (P=M/2)
*
*      AFTER   AFTER END       AFTER     AFTER     AFTER
*      COLAPS  POINT ADJUST    FIRST     SECOND    RVPRTS
*              AND DUBL        SPLIT     SPLIT     AND END
*                                                  POINT SET
* -2M  BLANK      0.0          A1(M)
* -2M  BLANK      0.0          A1(M)     (SAME
*-2M+1 XC(2M-1)  2XC(2M-1)               BUT
*                                        THESE
* ETC                                    NOT
*                              A1(2)     USED)
*-M-1                          A1(1)     A1(1)        0.0=AS(P)
* -M    XC(M)     2XC(M)       S1(M)     A2(P)      A2(1)=AS(P-1)
* ETC
* -P-1                         S1(P+1)   A2(1)        A2(P)=AS(0)
*  -P   XC(P)     2XC(P)       S1(P)     S2(P)        S2(0)=SS(P)
*ETC
*  -1   XC(1)     2XC(1)       S1(1)     S2(1)     S2(P-1)=SS(1)
*SPACE  XC(0)     2XC(0)       S1(0)     S2(0)        S2(P)=SS(0)
*
*THUS WHEN M EVEN   RSS = SPACE,   RAS = SPACE-P-1
*
*
*USE OF SPACE WHEN M IS ODD  (Q=(M+1)/2,  P=(M-1)/2=Q-1)
*
*      AFTER   AFTER END       AFTER     AFTER     AFTER
*      COLAPS  POINT ADJUST    FIRST     SECOND    RVPRTS
*              AND DUBL        SPLIT     SPLIT     AND END
*                                                  POINT SET
* -2M   BLANK     0.0          A1(M)
*-2M+1 XC(2M-1)  2XC(2M-1)     A1(M-1
*                                        SAME      SAME
*ETC
*-M-1                          A1(1)     A1(1)          0.0=NOT USED
```

```
* -M     XC(M)    2XC(M)        S1(M)      A2(Q)        A2(1)=AS(P)
*-M+1    XC(M-1)
*ETC
*                                          A2(2)      A2(Q-1)=AS(1)
*-Q(=-P+1)                                 A2(1)        A2(Q)=AS(0)
*-Q+1(=-P)                                 S2(Q)        S2(1)=SS(P)
*ETC
*  -1    XC(1)    2XC(1)        S1(1)      S2(2)      S2(Q-1)=SS(1)
*SPACE   XC(0)    2XC(0)        S1(0)      S2(1)        S2(Q)=SS(0)
*THUS WHEN M ODD   RSS = SPACE,    RAS = SPACE-P-1
*
      END
```

```
*      ASPEC2 (SUBROUTINE)             3/15/65   LAST CARD IN DECK IS NO. 0205
*      FAP
*ASPEC2
       COUNT   250
       LBL     ASPEC2
       ENTRY   ASPEC2 (ACOR, MXLAG, FREQLO, FRQDEL, NFREQS, IERRLO,
*                      SPECT, IANS)
*
*                      ----ABSTRACT----
*
* TITLE - ASPEC2
*         AUTOSPECTRUM BY COSINE TRANSFORM OF AUTOCORRELATION
*
*              ASPEC2  COMPUTES THE COSINE TRANSFORM VALUES
*
*                                         MXLAG
*                 SPECT(J) = AC(0) + 2 *   SUM  AC(I)*COS(W(J)*I)
*                                          I=I
*
*                 FOR  J = 1,2,...,NFREQS
*
*                 WHERE
*                    W(J) = FREQLO + (J-1)*FRQDEL, AND
*                    AC(I), MXLAG, FREQLO, FRQDEL, AND NFREQS ARE INPUTS
*
*              THE COMPUTATIONS ARE SPEEDED UP BY THE USE OF SUBROUTINE
*              SEQSAC  AND FUNCTION  NEXCOS  TO GENERATE COSINE VALUES.
*              WHILE THREE OR FOUR TIMES SLOWER THAN SUBROUTINE  ASPECT,
*              THE COMPUTATIONS HERE REQUIRE NO TEMPORARIES.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 74 REGISTERS
* SPEED      - ON THE 7090  ASPEC2  TAKES ABOUT
*                 100 + 118*NFREQS + 140*NFREQS*(MXLAG+1) MACHINE CYCLES
*              WHERE  NFREQS  AND  MXLAG  ARE DEFINED ABOVE
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - SEQSAC, NEXCOS
*        AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     CALL ASPEC2(ACOR, MXLAG, FREQLO, FRQDEL, NFREQS, IERRLO, SPECT,
*    1                IANS)
*
*
* INPUTS
*
*    ACOR(I)   I=1...MXLAG+1   CONTAINS THE AUTOCORRELATIONS  AC(J),
*                 J=0...MXLAG.
*
*    MXLAG     MUST BE GRTHN OR EQUAL ZERO
*
*    FREQLO    IS THE LOWEST FREQUENCY, W(1) OF THE ABSTRACT, IN RADIANS
*
*    FRQDEL    IS THE FREQUENCY INCREMENT IN RADIANS
*
*    NFREQS    IS THE DESIRED NUMBER OF OUTPUT SPECTRAL VALUES
*              MUST EXCEED ZERO
*
*    IERRLO    IS THE DESIRED  IANS  OUTPUT FOR ILLEGAL  MXLAG
*
*
* OUTPUTS      NO COMPUTATIONS ARE MADE FOR ILLEGAL  MXLAG  OR  NFREQS
*                 VALUES.
*
*    SPECT(J)  J=1...NFREQS  CONTAINS THE VALUES DEFINED IN THE ABSTRACT
*
*    IANS      = 0,          IF ALL OK
*              = IERRLO,     IF MXLAG ILLEGAL
```

```
*               = IERRLO+1,  IF NFREQS ILLEGAL
*
*
* EXAMPLES
*
* 1. INPUTS  - ACOR(1...4) = 1.0,.5,.5,.5
*              FREQLO = 3.14159265/6.0 (30 DEGREES)
*              FREQDEL = 2.0*FREQLO     NFREQS=2     IERRLO=1
*   USAGE   -        DIMENSION  SPECT(2,4), IANS(4)
*                    DO 10  LACOR=1,4
*                    MXLAG = LACOR-1
*                 10 CALL ASPEC2(ACOR, MXLAG, FREQLO, FRQDEL, NFREQS,
*                   1           IERRLO, SPECT(1,LACOR), IANS(LACOR))
*   OUTPUTS - SPECT(1...2,1) = 1.000, 1.000
*              SPECT(1...2,2) = 1.866, 1.000
*              SPECT(1...2,3) = 2.366, 0.000
*              SPECT(1...2,4) = 2.366, 0.000
*              IANS(1...4)    = 0,0,0,0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  ACOR(1...4) = 2.,1.,1.,1.
*                 NFREQS=1
*   USAGE   - SAME AS EXAMPLE 1.
*   OUTPUTS - SPECT(1,1...4) = 2.000,3.732,4.732,4.732
*              IANS(1...4)    = 0,0,0,0
*
* 3. ILLEGAL CASES  (D IS A DUMMY VARIABLE BELOW)
*   USAGES  -        CALL ASPEC2(D,-1,D,D,1,3,D,IANS1)
*                    CALL ASPEC2(D,0,D,D,0,3,D,IANS2)
*   OUTPUTS - IANS1=3     IANS2=4
*
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS  SEQSAC,  NEXCOS
*
       HTR     0               XR1
       HTR     0               XR2
       HTR     0               XR4
       BCI     1,ASPEC2
*
* ONLY ENTRY.  ASPEC2(ACOR, MXLAG, FREQLO, FRQDEL, NFREQS, IERRLO,
*                      SPECT, IANS)
*
ASPEC2 SXD     ASPEC2-4,1
       SXD     ASPEC2-3,2
       SXD     ASPEC2-2,4
*
* DIVIDE ACOR(1) BY 2, SET ADDRESSES, DECREMENTS, CHECK MXLAG, NFREQS
*
       CLA*    1,4             ACOR(1)
       STO     ACOR1           (SAVE IT)
       FDP     KL2
       STQ*    1,4
       CLA     1,4             A(ACOR)
       ADD     K1              A(ACOR)+1
       STA     FMP
       CLA     7,4             A(SPECT)
       ADD     K1              A(SPECT)+1
       STA     STO
       CLA*    6,4             IERRLO
       PDX     0,1             TO XR1
       CLA*    2,4             MXLAG
       ADD     KD1             MXLAG+1
       STD     TXL1
       TMI     LEAVE
       TZE     LEAVE
       TXI     *+1,1,1
       CLA*    5,4             NFREQS
       STD     TXL2
       SUB     KD1
       TZE     CLAFL
       TMI     LEAVE
*
* ALL OK, INITIALIZE FOR FREQUENCY INCREMENTING
*
```

* ASPEC2 *

(PAGE 3)


```
      CLAFL CLA*    3,4            FREQLO
            STO     FREQ
            CLA*    4,4            FRQDEL
            STO     FRQDEL
     *
     * OUTER LOOP COUNTS SPECTRAL VALUES WITH XR2 = 1...NFREQS
     *
            AXT     1,2
      TSX1  TSX     $SEQSAC,4
            TSX     KZ,0
            TSX     FREQ,0
     *
     * INNER LOOP COUNTS ACOR VALUES WITH XR1 = 1...MXLAGS+1
     *
            AXT     1,1
            STZ     SUM
      TSX2  TSX     $NEXCOS,4
            XCA
      FMP   FMP     **,1           ** = A(ACOR)+1
            FAD     SUM
            STO     SUM
            TXI     *+1,1,1
      TXL1  TXL     TSX2,1,**      ** = MXLAG+1
     *
     * STORE RESULT AND INDEX FOR MORE
     *
            XCA
            FMP     KL2
      STO   STO     **,2           ** = A(SPECT)+1
            CLA     FREQ
            FAD     FRQDEL
            STO     FREQ
            TXI     *+1,2,1
      TXL2  TXL     TSX1,2,**      ** = NFREQS
            AXT     0,1            (IANS = 0)
     *
     * EXIT, SETTING IANS AND RESTORING ACOR(1)
     *
      LEAVE PXD     0,1
            LDQ     ACOR1
            LXD     ASPEC2-4,1
            LXD     ASPEC2-3,2
            LXD     ASPEC2-2,4
            STO*    8,4
            STQ*    1,4
            TRA     9,4
     *
     * CONSTANTS, TEMPORARIES
     *
      KZ    PZE     0
      K1    PZE     1
      KD1   PZE     0,0,1
      KL2   DEC     2.0
     FRQDEL PZE     **,**,**       INPUT
      FREQ  PZE     **,**,**       FREQLO, FREQLO+FRQDEL,...
      SUM   PZE     **,**,**
      ACOR1 PZE     **,**,**       ACOR(1)
            END
```

```
*       AVRAGE  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0078
*       FAP
*AVRAGE
        COUNT   150
        LBL     AVRAGE
        ENTRY   AVRAGE (X,LX,XAVG)
*
*                     ----ABSTRACT----
*
* TITLE - AVRAGE
*         FIND AVERAGE OF FLOATING VECTOR
*
*               AVRAGE COMPUTES THE MEAN OF A FLTG VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 24 REGISTERS
* SPEED      - 52.4 + 8.4*LX  MACHINE CYCLES ON 7090, LX = VECTOR LENGTH
*              57.4 + 8.4*LX   ON 709
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL AVRAGE(X,LX,XAVG)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLTG VECTOR
*
*    LX         SHOULD EXCEED ZERO
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LX LSTHN 1
*
*    XAVG       IS (1/LX)*SUM (FROM I=1 TO LX) OF X(I)  (FLTG)
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)=1.,2.,3.,4.  U=0.
*    USAGE   -          CALL AVRAGE(X,4,XAVG)
*                       CALL AVRAGE(X,1,Y)
*                       CALL AVRAGE(X,0,U)
*    OUTPUTS - XAVG=2.5  Y=1.  U=0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR     0                 XR4
        BCI     1,AVRAGE
* ONLY ENTRY.   AVRAGE(X,LX,XAVG)
AVRAGE  SXD     AVRAGE-2,4
 K1     CLA     1,4
        ADD     K1                A(X)+1
        STA     ADD1
* CHECK LX AND FLOAT IT
        CLA*    2,4               LX
        TMI     LEAVE
        PDX     0,4               LOOP SET
        TXL     LEAVE,4,0
        LRS     18
        ORA     OCTK
        FAD     OCTK              FLOATED LX
        STO     FLX
* SUM   X(1...LX), DIVIDE, STORE, EXIT
        PXD     0,0
 ADD1   FAD     **,4              **=A(X)+1
        TIX     ADD1,4,1
        FDP     FLX
        LXD     AVRAGE-2,4
        STQ*    3,4
 LEAVE  LXD     AVRAGE-2,4
        TRA     4,4
```

* AVRAGE *
****
(PAGE 2)

```
* CONSTANTS, VARIABLES
 OCTK  OCT     233000000000
 FLX   PZE     **                  = LX FLOATED
       END
```

```
*      BLKSUM (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0168
*      FAP
*BLKSUM
       COUNT   200
       LBL     BLKSUM
       ENTRY   BLKSUM (X, LX, LBLOK, DVSR, XBSMOD, LXBSOD)
*
*
*                    ----ABSTRACT----
*
* TITLE - BLKSUM
*        SUMMATION OF VECTOR OVER ABUTTING BLOCKS OF CONSTANT LENGTH
*
*              BLKSUM  COMPUTES
*
*                          1        I*L
*                 S(I)  =  ---      SUM     X(J)
*                          D    J=(I-1)*L+1
*
*                 FOR  I = 1,2,...,N=(LX/L)ROUNDED DOWN
*
*              WHERE  X(1...LX),  LX,  L,  AND  D  ARE INPUTS.
*
*              THE OUTPUT VECTOR MAY REPLACE THE INPUT VECTOR, AND THE
*              LENGTH  N  IS AN ADDITIONAL OUTPUT FROM  BLKSUM.
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 49 REGISTERS
* SPEED      - REQUIRES  80 + 29*N + 8.4*L*N  MACHINE CYCLES ON THE 7090
*                WHERE  L  AND  N  ARE DEFINED ABOVE.
* AUTHOR     - S.M. SIMPSON,   JULY 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*      AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*    CALL BLKSUM(X, LX, LBLOK, DVSR, XBSMOD, LXBSOD)
*
*
* INPUTS
*
*   X(I)       I=1...LX  IS A FLOATING POINT VECTOR.
*
*   LX         MUST BE GRTHN= 1 .
*
*   LBLOK      IS THE BLOCK LENGTH  L  OF THE ABSTRACT.
*              MUST EXCEED ZERO AND BE LSTHN=  LX.
*
*   DVSR       IS THE DIVISOR  D  OF THE ABSTRACT.
*              MUST BE NON-ZERO.
*
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF  LX,  LBLOK,  OR  DVSR
*                 ILLEGAL.
*
*   XBSMOD(I) I=1...LXBSOD  ARE THE SUMS  S(1...N)  OF THE ABSTRACT.
*              EQUIVALENCE (X,XBSMOD) IS PERMITTED.
*
*   LXBSOD     WILL  =  (LX/LBLOK)  ROUNDED DOWN.
*
*
* EXAMPLES
*
* 1. MISCELLANEOUS VALUES OF  LX,  LBLOK
*   INPUTS  - X(1...4) = 2.,4.,6.,8.  DVSR=2.0
*              S(1...4,1...4,1...4) = -9.,-9.,...
*              LS(1...4,1...4) = -9,-9,...
*   USAGE   -        DO 10  LX=1,4
*                    DO 10  L=1,LX
*               10   CALL BLKSUM(X,LX,L,DVSR,S(1,L,LX),LS(L,LX))
```

```
*    OUTPUTS - S(1...4,1...4,1) =  1.,-9.,-9.,-9.,,-9.,-9.,-9.,-9.,,
*                                 -9.,-9.,-9.,-9.,,-9.,-9.,-9.,-9.
*              S(1...4,1...4,2) =  1., 2.,-9.,-9.,, 3.,-9.,-9.,-9.,,
*                                 -9.,-9.,-9.,-9.,,-9.,-9.,-9.,-9.
*              S(1...4,1...4,3) =  1., 2., 3.,-9.,, 3.,-9.,-9.,-9.,,
*                                  6.,-9.,-9.,-9.,,-9.,-9.,-9.,-9.
*              S(1...4,1...4,4) =  1., 2., 3., 4.,, 3., 7.,-9.,-9.,,
*                                  6.,-9.,-9.,-9.,,10.,-9.,-9.,-9.
*              LS(1...4,1...4)  =  1,-9,-9,-9,, 2, 1,-9,-9,,
*                                  3, 1, 1,-9,, 4, 2, 1, 1
*
* 2. OUTPUT REPLACING INPUT
*    INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   -       CALL BLKSUM(X,4,2,DVSR,X,LS)
*    OUTPUTS - X(1...4) = 3.,7.,6.,8.  LS = 2
*
* 3. ILLEGAL CASES
*    INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   -       CALL BLKSUM(X,-1,2,1.0,S,LS)
*                    CALL BLKSUM(X, 3,0,1.0,S,LS)
*                    CALL BLKSUM(X, 3,4,1.0,S,LS)
*                    CALL BLKSUM(X, 3,2,0.0,S,LS)
*    OUTPUTS - S = -9.  LS = -9
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR     0               XR2
       HTR     0               XR4
       BCI     1,BLKSUM
*
* ONLY ENTRY.  BLKSUM (X, LX, LBLOK, DVSR, XBSMOD, LXBSOD)
*
BLKSUM SXD     BLKSUM-3,2
       SXD     BLKSUM-2,4
*
* CHECK AND SET  DVSR,  LBLOK,  LXBSOD  AND SET ADDRESSES
*
       CLA*    4,4             DVSR
       TZE     LEAVE
       STO     DVSR
       CLA*    3,4             LBLOK
       ARS     18
       STO     LBLOK
       TMI     LEAVE
       TZE     LEAVE
       CLA*    2,4             LX
       CAS*    3,4             AGAINST LBLOK
       NOP                     OK
       TRA     LRS             OK
       TRA     LEAVE           NG
 LRS   LRS     35
       DVP*    3,4             LX/LBLOK ROUNDED DOWN
       CLM
       LLS     18
       XCA
       STO*    6,4             EQUALS LXBSOD
       STD     TXL
       CLA     1,4             A(X)
       ADD     K1              A(X)+1
       STA     FAD
       CLA     5,4             A(XBSMOD)
       ADD     K1              A(XBSMOD)+1
       STA     STQ
       AXT     1,2             XR2 WILL CONTROL OUTPUT STORAGE
*
* DOUBLE LOOP STARTS.  XR2=1...LXBSOD,  XR4=LBLOK...1  REPEATED
*
 PXD   PXD     0,0             (SUMMATION IN AC)
       LXA     LBLOK,4
 FAD   FAD     **,4            ** = A(X)+1,-LBLOK,-2*LBLOK,...
       TIX     FAD,4,1
       FDP     DVSR
```

```
 STQ   STQ     **,2            ** = A(XBSMOD)+1
       CAL     FAD
       SUB     LBLOK
       STA     FAD
       TXI     *+1,2,1
 TXL   TXL     PXD,2,**        ** = LXBSOD
*
* EXIT
*
 LEAVE LXD     BLKSUM-3,2
       LXD     BLKSUM-2,4
       TRA     7,4
*
* CONSTANTS TEMPORARIES
*
 K1    PZE     1
 DVSR  PZE     **,**,**
 LBLOK PZE     **,0,0
       END
```

```
*       BOOST (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0146
*       FAP
*BOOST
        COUNT   150
        LBL     BOOST
        ENTRY   BOOST  ( X, LX, XRIZE, XBUSTD)
        ENTRY   XBOOST (IX,LIX,IXRIZE,IXBSTD)
        ENTRY   DPRESS ( X, LX, XSINK, XLWRD)
        ENTRY   XDPRSS (IX,LIX,IXSINK,IXLWRD)
*
*                     ----ABSTRACT----
*
* TITLE - BOOST WITH SECONDARY ENTRIES XBOOST, DPRESS, AND XDPRSS
*        ADD A CONSTANT TO ELEMENTS OF A FXD OR FLTG VECTOR
*
*               BOOST ADDS A FLTG CONSTANT TO A VECTOR.
*               XBOOST ADDS A FXD CONSTANT TO A VECTOR.
*               DPRESS SUBTRACTS A FLTG CONSTANT FROM A VECTOR.
*               XDPRSS SUBTRACTS A FXD CONSTANT FROM A VECTOR.
*
*               THE CONSTANT MAY BE ONE OF THE VECTOR ELEMENTS.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 34   REGISTERS
* SPEED       - BOOST  -  36 + 12.4*LX  MACHINE CYCLES, LX=VECTOR LENGTH
*               XBOOST -  38 + 8*LX
*               DPRESS -  38 + 12.4*LX
*               XDPRSS -  38 + 8*LX
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL BOOST ( X, LX, XRIZE,XBUSTD)
*     CALL XBOOST(IX,LIX,IXRIZE,IXBSTD)
*     CALL DPRESS( X, LX, XSINK, XLWRD)
*     CALL XDPRSS(IX,LIX,IXSINK,IXLWRD)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLTG PT VECTOR
*
*    LX         SHOULD EXCEED 0
*
*    XRIZE      IS ANY FLTG VARIABLE. EQUIVALENCE (XRIZE, SOME X(I)) OK.
*
*    XSINK      IS ANY FLTG VARIABLE. EQUIVALENCE (XSINK, SOME X(I)) OK.
*
*    IX(I)      I=1...LIX  IS A FXD PT VECTOR
*
*    LIX        SHOULD EXCEED 0
*
*    IXRIZE     IS ANY FXD VARIABLE.  EQUIVALENCE(IXRIZE, SOME IX(I)) OK.
*
*    IXSINK     IS ANY FXD VARIABLE. EQUIVALENCE (IXSINK, SOME IX(I)) OK.
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUTS IF LX OR LIX LSTHN 1.
*
*    XBUSTD(I) I=1...LX  HAS VALUES  XBUSTD(I)=X(I)+XRIZE
*               EQUIVALENCE (XBUSTD,X) IS PERMITTED.
*
*    XLWRD(I)  I=1...LX  HAS VALUES  XLWRD(I)=X(I)-XSINK
*               EQUIVALENCE (XLWRD,X) IS PERMITTED.
*
*    IXBSTD(I) I=1...LIX  HAS VALUES  IXBSTD(I)=IX(I)+IXRIZE
*               EQUIVALENCE (IXBSTD,IX) IS PERMITTED.
*
*    IXLWRD(I) I=1...LIX  HAS VALUES  IXLWRD(I)=IX(I)-IXSINK
*               EQUIVALENCE (IXLWRD,IX) IS PERMITTED.
*
*               IF ANY OF THE ABOVE EQUIVALENCES OBTAIN, THE INITIAL
```

****
* BOOST *
****
(PAGE 2)


```
*               VALUE OF THE QUANTITY IS ALWAYS USED AS THE ADDEND OR
*               SUBTRAHEND.
*
* EXAMPLES
*
* 1. INPUTS   - X(1...4)=1.,2.,3.,4.  Y(1...4)=1.,2.,3.,4.
*               IX(1...4)=1,2,3,4  IY(1)=1
*               W=WW=0.0  IW=IWW=0
*    USAGE    -        CALL BOOST(X,4,2.,Z)
*                      CALL DPRESS(Y,4,2.,U)
*                      CALL XBOOST(IX,4,2,IZ)
*                      CALL XDPRSS(IY,1,2,IU)
*                      CALL BOOST(X,0,2.,W)
*                      CALL DPRESS(Y,0,2.,WW)
*                      CALL XBOOST(IX,0,2,IW)
*                      CALL XDPRSS(IY,0,2,IWW)
*
*    OUTPUTS - Z(1...4)=3.,4.,5.,6.  U(1...4)=-1.,0.,1.,2.
*              IZ(1...4)=3,4,5,6  IU(1)=-1
*              W=WW=0.0  IW=IWW=0  (NO OUTPUTS FROM LAST 4 CALLS)
*
* 2. INPUTS   - SAME AS EXAMPLE 1.
*    USAGE    -        CALL BOOST(X,4,2.,X)
*                      CALL DPRESS(Y,4,Y(3),Y)
*    OUTPUTS - X(1...4)=3.,4.,5.,6.  Y(1...4)=-2.,-1.,0.,1.
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR     0                  XR4
       BCI     1,BOOST
* PRINCIPAL ENTRY.  BOOST(X,LX,XRIZE,XBUSTD)
 BOOST CLA     FAD
 SETUP STO     MODIFY
       SXD     BOOST-2,4
 K1    CLA     1,4
       ADD     K1                 A(X)+1
       STA     GET
       CLA     4,4
       ADD     K1                 A(XBUSTD)+1
       STA     STORE
       CLA*    3,4                XRIZE
       STO     TEMP
* CHECK LX
       CLA*    2,4                LX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* LOOP
 GET   CLA     **,4               **=A(X)+1
MODIFY NOP                   = FAD TEMP, ADD TEMP, FSB TEMP OR SUB TEMP
 STORE STO     **,4               **=A(XBUSTD)+1
       TIX     GET,4,1
* EXIT
 LEAVE LXD     BOOST-2,4
       TRA     5,4
* SECOND ENTRY.  XBOOST(IX,LIX,IXRIZE,IXBSTD)
XBOOST CLA     ADD
       TRA     SETUP
* THIRD ENTRY.  DPRESS(X,LX,XSINK,XLWRD)
DPRESS CLA     FSB
       TRA     SETUP
* FOURTH ENTRY.  XDPRSS(IX,LIX,IXSINK,IXLWRD)
XDPRSS CLA     SUB
       TRA     SETUP
* CONSTANTS, VARIABLES.
 ADD   ADD     TEMP
 SUB   SUB     TEMP
 FAD   FAD     TEMP
 FSB   FSB     TEMP
 TEMP  PZE     **,**,**           ADDEND OR SUBTRAHEND
       END
```

```
************************
*    CALL              *
************************
 REFER TO
     LOCATE
```


```
************************
*    CALL              *
************************
 REFER TO
     LOCATE
```


```
************************
*    CALL2             *
************************
 REFER TO
     LOCATE
```


```
************************
*    CALL2             *
************************
 REFER TO
     LOCATE
```

```
*     CARIGE  (SUBROUTINE)            9/29/64   LAST CARD IN DECK IS NO. 0097
*     LABEL
CCARIGE
      SUBROUTINE CARIGE(ITAPE,NSPACE)
C
C                    ----ABSTRACT----
C
C TITLE - CARIGE
C        SPACE CARRIAGE N LINES OR RESTORE PAGE
C
C              CARIGE WRITES OUT CARRIAGE CONTROL HOLLERITH ON A GIVEN
C              OUTPUT TAPE FOR OFF-LINE PRINTING UNDER PROGRAM CONTROL.
C              IT WILL EITHER SPACE THE PRINTED PAGE N LINES (WHERE N
C              MAY BE ZERO) OR GIVE A SINGLE PAGE RESTORE.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME + 1 TAPE UNIT)
C STORAGE    - 47 REGISTERS
C SPEED      -
C AUTHOR     - S.M. SIMPSON, SEPTEMBER 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
C       AND FORTRAN SYSTEM ROUTINES - (STH), (FIL)
C
C FORTRAN USAGE
C     CALL CARIGE(ITAPE,NSPACE)
C
C INPUTS
C
C    ITAPE     IS A FORTRAN-II INTEGER SPECIFYING THE LOGICAL TAPE TO
C                 BE USED.
C              IS NOT EXAMINED FOR LEGALITY.
C
C    NSPACE    IS A FORTRAN-II INTEGER SPECIFYING THE NUMBER OF SPACES
C                 (CARRIAGE RETURNS) DESIRED.
C              =+N  PRODUCES N CARRIAGE RETURNS
C              =-N  PRODUCES 1 PAGE RESTORE
C              = 0  STRAIGHT RETURN WITH NO EFFECT
C
C OUTPUTS      CARRIAGE CONTROL HOLLERITH IS WRITTEN OUT ON ITAPE.
C
C EXAMPLES
C
C 1. INPUTS  - NSPACE(1...5)=1,0,-5,4,9
C
C    USAGE   -        DO 5  I=1,5
C                     WRITE OUTPUT TAPE 2,10,I
C                     CALL CARIGE(2,NSPACE(I))
C                5    WRITE OUTPUT TAPE 2,10,I
C               10    FORMAT(17H THIS IS A MARKER,6X,2HI=,I1)
C
C    OUTPUTS - 2 PAGES OF PRINTED OUTPUT FROM LOGICAL UNIT 2, AS FOLLOWS
C
C              PAGE 1
C
C (LINE 1)      THIS IS A MARKER      I=1
C                 (BLANK LINE)
C               THIS IS A MARKER      I=1
C               THIS IS A MARKER      I=2
C               THIS IS A MARKER      I=2
C (LINE  6)     THIS IS A MARKER      I=3
C
C
C              PAGE 2
C
C (LINE  1)     THIS IS A MARKER      I=3
C               THIS IS A MARKER      I=4
C                 (BLANK LINE)
C                 (BLANK LINE)
C                 (BLANK LINE)
C                 (BLANK LINE)
C               THIS IS A MARKER      I=4
C               THIS IS A MARKER      I=5
```

****
* CARIGE *
****
(PAGE 2)

```
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C                  (BLANK LINE)
C (LINE 18)     THIS IS A MARKER       I=5
C
C
C PROGRAM FOLLOWS BELOW
C
      IF (NSPACE)  10,9999,30
 10   WRITE OUTPUT TAPE ITAPE,20
 20   FORMAT(1H1)
      GO TO 9999
 30   DO 40  I=1,NSPACE
 40   WRITE OUTPUT TAPE ITAPE,50
 50   FORMAT(1H )
 9999 RETURN
      END
```

```
*     CHISQR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO.
*     LABEL
CCHISQR
      SUBROUTINE CHISQR(NBLOCS,ICOUNT,N,CHISQ,IANS)
C
C                  ----ABSTRACT----
C
C TITLE - CHISQR
C       COMPUTES CHI-SQUARE FOR EQUALLY LIKELY PROBABILITY CASE.
C
C             CHISQR COMPUTES CHI SQUARE WHEN GIVEN THE DISTRIBUTION
C             COUNT AND THE NUMBER OF EQUALLY LIKELY BLOCKS INTO WHICH
C             THE DATA IS PUT. NUMBER OF BLOCKS = NBLOCKS, N = TOTAL
C             NUMBER OF OBSERVATIONS, ICOUNT = DISTRIBUTION COUNT.
C
C             CHISQ=SUM((ICOUNT(I)-N/NBLOCKS)**2/(N/NBLOCKS))
C
C             SUMMED OVER NBLOCKS, WHERE FLOATING OPERATIONS ARE ASSUMED
C             RATHER THAN THE INDICATED INTEGER OPERATIONS.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 105 REGISTERS
C SPEED     -
C AUTHOR    - J.N. GALBRAITH
C
C                  ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL CHISQR(NBLOCS,ICOUNT,N,CHISQ,IANS)
C
C INPUTS
C
C    NBLOCKS   IS NUMBER OF EQUALLY LIKELY BLOCKS.
C              MUST BE GRTHN 1.
C
C    ICOUNT(I) I=1...NBLOCKS IS THE DISTRIBUTION COUNT. I.E. THE NUMBER
C              OF VALUES IN I-TH EQUALLY LIKELY BLOCK.
C              MUST BE NON-NEGATIVE
C
C    N         IS TOTAL NUMBER OF OBSERVATIONS (=SUM(ICOUNT(I))).
C              MUST BE GRTHN=1.
C
C OUTPUTS
C
C    CHISQ     IS THE CHI-SQUARE VALUE
C
C    IANS      =0  NORMAL
C              =1  ILLEGAL NBLOCS
C              =2  ILLEGAL N
C
C EXAMPLES
C
C 1. INPUTS  - NBLOCS=3   ICOUNT(1...3)=1,3,5   N=9
C    OUTPUTS - CHISQ=2.666667   IANS=0
C
C 2. INPUTS  - NBLOCS=1   ICOUNT(1)=1   N=9
C    OUTPUTS - ERROR   IANS=1
C
C 3. INPUTS  - NBLOCS=3   ICOUNT(1...3)=1,3,5   N=0
C    OUTPUTS - ERROR   IANS=2
C
C 4. INPUTS  - NBLOCS=5   ICOUNT(1...5)=1,2,3,4,5   N=15
C    OUTPUTS - CHISQ=3.333333   IANS=0
C
      DIMENSION ICOUNT(100)
      IANS=0
      IF(NBLOCS-1)  990,990,5
 5    IF(N)  992,992,10
 10   P=1./FLOATF(NBLOCS)
      EXPNO=P*FLOATF(N)
      CHISQ=0
```

```
************************
*    CHISQR            *
************************
(PAGE  2)
```


```
      DO 25  I=1,NBLOCS
      DIF=FLOATF(ICOUNT(I))-EXPNO
   25 CHISQ=CHISQ+DIF*DIF
      CHISQ=CHISQ/EXPNO
   26 RETURN
  990 IANS=1
      GO TO 26
  992 IANS=2
      GO TO 26
      END
```

```
*       CHOOSE (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0083
*       FAP
*CHOOSE
        COUNT   100
        LBL     CHOOSE
        ENTRY   CHOOSE (ZIFRST, X,X1,X2, Y,Y1,Y2,..., Z,Z1,Z2)
*
*
*                    ----ABSTRACT----
*
* TITLE - CHOOSE
*          SET A LIST OF VARIABLES TO ONE OF TWO SETS OF VALUES
*
*
*               CHOOSE SETS A VARIABLE LENGTH LIST OF VARIABLES (X,Y,...,
*               Z) FROM THE FIRST LIST (X1,Y1,...,Z1) OF A PAIR OF LISTS
*               OF CONSTANTS IF ZIFRST=0. OR FROM THE SECOND LIST IF
*               ZIFRST NOT=0.  I.E., IF ZIFRST=0. CHOOSE SETS X=X1,Y=Y1,
*               ...,Z=Z1.  IF ZIFRST NOT=0. CHOOSE SETS X=X2,Y=Y2,...,
*               Z=Z2.
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 17 REGISTERS
* SPEED      - 6 + 16*N  MACHINE CYCLES, WHERE N = NO. OF SETTINGS
* AUTHOR     - S.M. SIMPSON, APRIL 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*        AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL CHOOSE(ZIFRST, X,X1,X2, Y,Y1,Y2, ..., Z,Z1,Z2)
*
* INPUTS
*
*    ZIFRST     = 0. IF VALUES X1,Y1,...,Z1 ARE TO BE CHOSEN.
*               NOT= 0. IF VALUES X2,Y2,...,Z2 ARE TO BE CHOSEN.
*
*    X1,X2,Y1,Y2,...,Z1,Z2,  ARE ANY MODE
*
*
* OUTPUTS       ILLEGAL RETURN OCCURS IF ARGUMENT COUNT IS NOT
*               1  +  MULTIPLE OF  3 .
*
*    X,Y,...,Z ARE FORMED AS DESCRIBED IN ABSTRACT.
*
*
* EXAMPLES
*
* 1. USAGES  -        CALL CHOOSE(-0.,X1,1.,2., IX1,1,2)
*                     CALL CHOOSE(1,X2,1.,2.)
*                     CALL CHOOSE(-.0001,X3,1.,2., IX3,1,2, X4,X3,X3)
*    OUTPUTS - X1=1.  IX1=1  X2=2.  X3=2.  IX3=2  X4=2.
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
        BCI     1,CHOOSE
*
* ONLY ENTRY.  CHOOSE(ZIFRST, X,X1,X2, Y,Y1,Y2, ..., Z,Z1,Z2)
*
CHOOSE  CLA     K3
        ZET*    1,4
        ADD     *-1
        STA     GET
 CAL    CAL     2,4
        ANA     AMASK
        LAS     TSXZ
        TRA     2,4
```

****
*   CHOOSE   *
****
(PAGE 2)

```
        TRA    GET
        TRA    2,4
  GET   CLA*   **,4   **=3 (ZIFRST=0), OR 4 (ZIFRST NOT=0)
        STO*   2,4
        TXI    CAL,4,-3
  K3    PZE    3
  AMASK OCT    777777700000
  TSXZ  TSX    0,0
        END
```

```
*      CHPRTS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0148
*      FAP
*CHPRTS
       COUNT   150
       LBL     CHPRTS
       ENTRY   CHPRTS (SYM,ANT,N)
       ENTRY   RVPRTS (SYM,ANT,N)
*
*                ----ABSTRACT----
*
* TITLE - CHPRTS, WITH SECONDARY ENTRY  RVPRTS
*       FAST REVERSAL OF SPECIAL VECTORS  (AS PRODUCED BY SPLIT)
*
*             CHPRTS REVERSES THE STORAGE ORDER OF TWO VECTORS (CALLED
*             ANT AND SYM) AND CHANGES THE SIGN OF ANT.
*
*             RVPRTS REVERSES THE STORAGE ORDER OF TWO VECTORS.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 76 REGISTERS
* SPEED     - ABOUT 6.5*(COMBINED LENGTH OF THE VECTORS) MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON JR
*
*                ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL CHPRTS(SYM,ANT,N)
*     CALL RVPRTS(SYM,ANT,N)
*
* INPUTS
*
*    SYM(I)     I=1...LS IS FIRST VECTOR TO BE REVERSED
*               (NAME NEED NOT BE FLOATING POINT)
*
*    ANT(I)     I=1...LA IS SECOND VECTOR TO BE REVERSED (AND SIGN
*                 CHANGED FOR CHPRTS ENTRY)
*               (NAME NEED NOT BE FLOATING POINT)
*
*    N          = LS+LA
*               IF N IS EVEN  LS = LA = N/2
*               IF N IS ODD   LS = (N+1)/2   LA = (N-1)/2
*               IS FORTRAN II INTEGER
*
* OUTPUTS
*
*    SYM(I)     I=1...LS IS THE REVERSED SYM SERIES.
*
*    ANT(I)     I=1...LA IS THE REVERSED ANT SERIES (WITH SIGN CHANGED
*                 IF THE CHPRTS ENTRY WAS USED).
*               (NOTE- PROGRAM EXITS WITHOUT MODIFYING SYM OR ANT
*                IF N IS LSTHN= 1)
*
* EXAMPLES
*
* 1. INPUTS  - SYM(1...3) = 3.,2.,1.  ANT(1...3) = 4.,1.,2.  N=6
*    OUTPUTS - CHPRTS  SYM(1...3) = 1.,2.,3.  ANT(1...3) = -2.,-1.,-4.
*              RVPRTS  SYM(1...3) = 1.,2.,3.  ANT(1...3) = 2.,1.,4.
*
* 2. INPUTS  - SYM(1...3) = 3.,2.,1.  ANT(1...2) = 4.,5.  N=5
*    OUTPUTS - CHPRTS  SYM(1...3) = 1.,2.,3.  ANT(1...2) = -5.,-4.
*              RVPRTS  SYM(1...3) = 1.,2.,3.  ANT(1...2) = 5.,4.
*
* 3. INPUTS  - SYM(1) = 1.  ANT(1)=2.  N=2
*    OUTPUTS - CHPRTS  SYM(1)=1.  ANT(1)=-2.
*              RVPRTS  SYM(1)=1.  ANT(1)=2.
*
       PZE
       BCI     1,CHPRTS
CHPRTS CLA     K3
       STO     C10
       CLA     K4
```

```
         STO    C12A
         TRA    C2
  RVPRTS CLA    K5
         STO    C10
         CLA    K6
         STO    C12A
   C2    SXD    CHPRTS-2,4
         SXA    LV+1,1
         SXA    LV+2,2
*  FIGURE THE LENGTHS OF SYM(LS) AND ANT(LA)
         CLA*   3,4            GET N
         ARS    18             IN ADDRESS
         CAS    K1
         TRA    *+3
         TRA    LV       EXIT UNLESS
         TRA    LV       N EXCEEDS 1
         LBT
         TRA    C3             EVEN
         ARS    1              ODD      LA=(N-1)/2=N/2  TRUNCATED
         STA    LA
         ADD    K1                      LS=(N+1)/2=LA+1
         TRA    C4
   C3    ARS    1              EVEN     LA=LS=N/2
         STA    LA
   C4    STA    LS
* SET DECREMENT AND ADDRESSES
         CLA    LA             DECR = LA/2 ROUNDED UP
         LRS    1
         RND
         ALS    18
         STD    C18
         CLA    1,4
         ADD    K1             SYM+1
         STA    C14
         STA    C17
         SUB    LS
         SUB    K1             SYM(LS)-1
         STA    C15
         STA    C16
         CLA    2,4
         ADD    K1             ANT+1
         STA    C10
         STA    C13
         SUB    LA             ANT(LA)-1
         SUB    K1
         STA    C11
         STA    C12
         AXT    1,1            IR1 COUNTS UP FROM 1 TO LA/2
         AXT    -1,2           IR2 COUNTS DOWN FROM -1 TO -LA/2
   C10   NOP                   (**=ANT+1)          REVERSE AND
   C11   LDQ    **,2           (**=ANT(LA)-1)      POSSIBLY CHANGE
   C12   STO    **,2           (**=ANT(LA)-1)        SIGN OF
         XCA                                         ANTISYMMETRIC
   C12A  NOP                                         PART
   C13   STO    **,1           (**=ANT+1)
   C14   CLA    **,1           (**=SYM+1)            REVERSES
   C15   LDQ    **,2           (**=SYM(LS)-1)        SYMMETRIC
   C16   STO    **,2           (**=SYM(LS)-1)        PART
   C17   STQ    **,1           (**=SYM+1)
         TXI    *+1,2,-1
         TXI    *+1,1,1
   C18   TXL    C10,1,**       (**=LA/2 ROUNDED UP)
   LV    LXD    CHPRTS-2,4
         AXT    **,1
         AXT    **,2
         TRA    4,4
   K1    PZE    1
   LS    PZE    **             (**=LS=LENGTH OF SYM)
   LA    PZE    **             (**=LS=LENGTH OF ANT)
   K3    CLS    **,1           FOR CHPRTS
   K4    CHS                   FOR CHPRTS
   K5    CLA    **,1           FOR PVPRTS
   K6    NOP                   FOR RVPRTS
         END
```

```
*       CHSIGN  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0077
*       FAP
*CHSIGN
        COUNT   100
        LBL     CHSIGN
        ENTRY   CHSIGN (X,LX,XNEG)
*
*                     ----ABSTRACT----
*
* TITLE - CHSIGN
*         CHANGE ALL SIGN BITS OF A VECTOR
*
*               CHSIGN CHANGES SIGN BITS IN A FLOATING OR FIXED VECTOR
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 18 REGISTERS
* SPEED      - 27 + 6*LX  MACHINE CYCLES,  LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL CHSIGN(X,LX,XNEG)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FIXED OR FLOATING VECTOR
*
*    LX         SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LX LSTHN 1 .
*
*    XNEG(I)    I=1...LX  IS  XNEG(I)= -X(I)
*
*               EQUIVALENCE (XNEG,X)  IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4) = 1.,-1.,2.,0.  IX(1...4) = -1,1,-2,-0  Y=0.
*    USAGE   -        CALL CHSIGN( X,4, XNEG)
*                     CALL CHSIGN(IX,4,IXNEG)
*                     CALL CHSIGN( X,4, X)
*                     CALL CHSIGN(IX,1,IX)
*                     CALL CHSIGN( X,0, Y)
*    OUTPUTS - XNEG(1...4) = -1.,1.,-2.,-0.  IXNEG(1...4) = 1,-1,2,0
*                 X(1...4) = -1.,1.,-2.,-0.  IX(1) = 1
*               Y = 0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,CHSIGN
* ONLY ENTRY.   CHSIGN(X,LX,XNEG)
CHSIGN  SXD     CHSIGN-2,4
 K1     CLA     1,4
        ADD     K1             A(X)+1
        STA     GET
        CLA     3,4
        ADD     K1             A(XNEG)+1
        STA     STORE
        CLA*    2,4            LX
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
* REVERSING LOOP
 GET    CLS     **,4           **=A(X)+1
 STORE  STO     **,4           **=A(XNEG)+1
        TIX     GET,4,1
* EXIT
```

```
     LEAVE  LXD     CHSIGN-2,4                                    0075
            TRA     4,4                                           0076
            END                                                   0077
```

```
************************
*    CHUSET            *
************************
 REFER TO
     INDEX
```


```
************************
*    CHUSET            *
************************
 REFER TO
     INDEX
```

```
*     CLKON    (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0041
*     LABEL
CCLKON
      SUBROUTINE CLKON
C
C                    ----ABSTRACT----
C
C TITLE - CLKON
C          CHECKS IF INTERVAL TIMER IS ON
C
C              CLKON OPERATES CLOCK1 TO DETERMINE IF THE INTERVAL TIMER
C              IS ON.  IF IT IS NOT ON, CLKON PRINTS AN ON-LINE MESSAGE
C
C                    OPERATOR, PLEASE TURN INTERVAL TIMER ON
C
C              REPEATEDLY UNTIL THE TIMER IS TURNED ON.  IF THE TIMER
C              IS ON, CLKON RETURNS TO THE CALLER.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME, INTERVAL TIMER, AND ON-LINE
C              PRINTER)
C STORAGE    - 46   REGISTERS
C AUTHOR     - R.A. WIGGINS   MAY, 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CLOCK1
C         AND FORTRAN SYSTEM ROUTINES - (FIL),(SPH)
C
C FORTRAN USAGE
C     CALL CLKON
C
C PROGRAM FOLLOWS BELOW
C
 10   JOB=0
      CALL CLOCK1 (JOB,TIME)
      IF (JOB) 20,40,40
 20   PRINT 30
 30   FORMAT(1H05X39HOPERATOR, PLEASE TURN INTERVAL TIMER ON)
      GO TO 10
 40   RETURN
      END
```

CLOCK1 (7090)


```
*      CLOCK1 (7090) (SUBROUTINE)       3/15/65   LAST CARD IN DECK IS NO. 0147
*      FAP
*CLOCK1 (7090)
       COUNT   130
       LBL     CLOCK1
       ENTRY   CLOCK1 (JOB,TIME)
*
*                    ----ABSTRACT----
*
* TITLE - CLOCK1
*       FOR REAL TIME TIMING IN SECONDS USING 7090 INTERVAL CLOCK
*
*             CLOCK1 ALLOWS FORTRAN ACCESS TO THE CORE STORAGE CLOCK
*             SO THAT IT MAY BE USED AS A TIMER.  IT WILL RETURN THE
*             ELAPSED TIME IN SECONDS AS A FLOATING POINT NUMBER OR IN
*             FIXED POINT MULTIPLES OF 1/60 SECOND.
*
*             CLOCK1 WILL ALSO TELL IF THE INTERVAL CLOCK IS RUNNING
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090, 7094 (MAIN FRAME, CORE STORAGE CLOCK
*                        AND INTERVAL TIMER)
* STORAGE   - 57 REGISTERS
* AUTHOR    - S.M. SIMPSON,   MAY, 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL CLOCK1(JOB,TIME)
*
* INPUTS
*
*    JOB       DEFINES WHAT CLOCK1 DOES.
*              =0  CHECKS TO SEE IF CLOCK IS RUNNING.
*              =1  REMEMBERS PRESENT CORE STORAGE CLOCK VALUE.
*              =2  TELLS ELAPSED TIME FROM LAST TIME JOB=1
*                    (IN SECONDS, FLOATING POINT).
*              =3  TELLS ELAPSED TIME FROM LAST TIME JOB=1
*                    (IN FORTRAN II INTEGER MULTIPLES OF 1/60 SECOND).
*              IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*    JOB       IS UNDISTURBED EXCEPT FOR THE CASE OF INPUT JOB=0 AND
*                THE CORE STORAGE CLOCK (REGISTER 5) IS NOT RUNNING.
*                IN THIS CASE JOB IS SET = -1.
*
*    TIME      IF INPUT JOB = 0 IS UNDISTURBED.
*                           = 1 IS UNDISTURBED.
*                           = 2 IS SET = NO. SECONDS (IN FLOATING POINT)
*                                  WHICH HAVE ELAPSED SINCE THE LAST USE
*                                  WITH JOB = 1.
*                           = 3 IS SET = NO. OF COUNTS (IN FORTRAN II
*                                  INTEGERS) (1 COUNT = 1/60 SEC)
*                                  MODULO 2**17 .
*              MAY DIFFER ON SUCCESSIVE RUNS BY .016667 SEC.
*
* EXAMPLES
*
* 1. INPUTS  - ASSUME THE FOLLOWING USE OF CLOCK1
*                   10   CALL CLOCK1(JOB1,TIME1)
*                   20   DO 30  I=1,32765
*                   30   CONTINUE
*                   40   CALL CLOCK1(JOB2,TIME2)
*              JOB1=0  JOB2=2  CLOCK IS NOT ON.
*
*    OUTPUTS - JOB1=-1  TIME1 IS UNDISTURBED  TIME2 CONTAINS A
*                MEANINGLESS NUMBER.
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT CLOCK IS ON.
*    OUTPUTS - JOB1=0  TIME1 IS UNDISTURBED  TIME2 CONTAINS A
```

```
*                  MEANINGLESS NUMBER
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT CLOCK IS ON JOB1=1  JOB2=2
*    OUTPUTS - JOB1=1  TIME1 UNDISTURBED  JOB2=2  TIME2= .28 (7090)
*                                                   OR .13 (7094)
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT CLOCK IS ON JOB1=1  JOB2=3
*    OUTPUTS - JOB1=1  TIME1 UNDISTURBED  JOB2=3  TIME = 17  (7090)
*                                                   OR  8  (7094)
*
       HTR     0
       BCI     1,CLOCK1
CLOCK1 SXD     CLOCK1-2,4
       CLA     5
       STO     TEMP
       SXA     LV,1
       CLA     1,4
       STA     SJ1
       STA     SJ2
       CLA     2,4
       STA     STORE
*FIND OUT WHICH JOB
  SJ1  CLA     **           **=JOB
       TMI     LV
       CAS     KD1
       TRA     J2OR3
       TRA     JOB1
       TRA     JOBZ
 J2OR3 SUB     KD1
       CAS     KD1
       TRA     JOB3
       TRA     JOB2
       TRA     LV
* WAIT A SECOND (IN THE 709)
  JOBZ LXA     K32K,1
 LOOP  TIX     LOOP,1,1
*DID CLOCK INCREMENT (YES IF NOW DIFFERENT FROM TEMP)
       CLA     5
       CAS     TEMP
       TRA     LV
       TRA     NOCLOK
       TRA     LV
*INDICATE CLOCK NOT RUNNING
NOCLOK CLS     KD1
 SJ2   STO     **             **=JOB
       TRA     LV
*FOR JOB 1 SAVE REG 5 AND EXIT
 JOB1  CLA     5
       STO     ORG
       TRA     LV
*FOR JOB 2 OR 3 SET DIFFERENCE
  JOB3 CLA     TEMP
       SUB     ORG
       ANA     KMSK
       ALS     18
       TRA     STORE
  JOB2 CLA     TEMP
       SUB     ORG
       ORA     KOCT
       FAD     KOCT
       FDP     KCONV
       STQ     TEMP
       CLA     TEMP
 STORE STO     **             **=TIME
  LV   AXT     **,1           **= XR1
       TRA     3,4
 KD1   PZE     0,0,1
 KOCT  OCT     233000000000
  KMSK OCT     000000377777
 K32K  PZE     32767
 KCONV DEC     60.0
 TEMP  PZE     **            ** = TEMPORARY
 ORG   PZE     **             **=CLOCK SAVE
       END
```

```
************************
*    CMPARL            *
************************
 REFER TO
     CMPARV
```


```
************************
*    CMPARL            *
************************
 REFER TO
     CMPARV
```

```
*      CMPARP (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0150
*      FAP
*CMPARP
       COUNT   150
       LBL     CMPARP
       ENTRY   CMPARP (IANS,X1,Y1,X2,Y2,...,XN,YN)
       ENTRY   CMPARS (IANS,X1,X2,...,XN)
*
*                    ----ABSTRACT----
*
* TITLE - CMPARP WITH SECONDARY ENTRY CMPARS
*        COMPARE PAIRS OF VARIABLES OR A SET OF VARIABLES FOR EQUALITY
*
*              CMPARP IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
*              WHICH TREATS ITS ARGUMENTS, BEYOND THE FIRST ONE, IN
*              PAIRS.  THE TWO ELEMENTS IN EACH PAIR ARE COMPARED FOR
*              IDENTITY.
*
*              CMPARS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
*              WHICH TREATS ITS ARGUMENTS, BEYOND THE FIRST ONE, AS A
*              SET OF QUANTITIES.  THE ELEMENTS IN THIS SET ARE
*              COMPARED TO SEE IF THEY ARE ALL IDENTICAL.
*
*              BOTH ENTRIES CONSIDER +0 TO BE THE SAME AS -0 AND LEAVE
*              THE RESULT OF THE TEST IN THEIR FIRST ARGUMENT.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 53 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON JR.,   OCTOBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE OF CMPARP
*     CALL CMPARP(IANS,X1,Y1,X2,Y2,...,XN,YN)
*
* INPUTS
*
*    X1,Y1     IS FIRST PAIR TO BE TESTED, ANY MODE
*    X2,Y2     IS SECOND PAIR TO BE TESTED, ANY MODE
*         ETC
*    XN,YN     IS N-TH PAIR TO BE TESTED, WHERE N MUST EXCEED 0 .
*
* OUTPUTS      ILLEGAL RETURN RESULTS IF ARGUMENT COUNT IS EVEN OR LESS
*                 THAN 3 .
*
*    IANS      =0 IF X1=Y1 AND X2=Y2 AND...AND XN=YN   (+0=-0)
*              =K IF XK NOT= YK   (K IS LOWEST SUCH INDEX IF MORE THAN
*                 ONE)
*
* FORTRAN USAGE OF CMPARS
*     CALL CMPARS(IANS,X1,X2,...,XN)
*
* INPUTS
*
*    X1,X2,...,XN ARE THE N QUANTITIES (ARBITRARY MODE) TO BE TESTED.
*                    N MUST BE GRTHN=2 .
*
* OUTPUTS      AN ILLEGAL RETURN RESULTS IF ARGUMENT COUNT IS LESS
*                 THAN 3 .
*
*    IANS      =0 IF X1=X2=...=XN   (+0 EQUALS -0 IN THE TEST)
*              =K IF XK NOT= XK+1   (K IS THE LOWEST SUCH INDEX)
*
*
* EXAMPLES
*
* 1. INPUTS  - A1,A2,A3 = 1.,2.,3.  B1,B2,B3 = 1.,2.,3.   IX1 = IY1 = 1
*              AZ = 0.   BZ = -0.   C3 = D3 = 3.
*
*    USAGE   -       CALL CMPARP(IANS1,AZ,BZ,A1,B1,A2,B2,A3,B3,IX1,IY1)
```

```
*                  CALL CMPARP(IANS2,A1,B1,IX1,IY1,A1,AZ,A3,B2)
*                  CALL CMPARS(IANS3,A3,B3,C3,D3)
*                  CALL CMPARS(IANS4,A3,B3,B2,D3)
*
*    OUTPUTS - IANS1 = 0  IANS2 = 3  IANS3 = 0  IANS4 = 2
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR1
       HTR     0              XR2
       HTR     0              XR4
* PRINCIPAL ENTRY.  CMPARP(IANS,X1,Y1,X2,Y2,...,XN,YN)
       BCI     1,CMPARP
CMPARP CLA     KDM2
       TRA     SETUP
* SECONDARY ENTRY.  CMPARS(IANS,X1,X2,...,XN)
       BCI     1,CMPARS
CMPARS CLA     KDM1
 SETUP STD     TXI2
       STA     EXIT
       STA     CALX
       SXD     CMPARP-2,4
       SXD     CMPARP-3,2
       SXD     CMPARP-4,1
       CLA     1,4            A(IANS)
       STA     STOANS
* POSITION 1,4 TO X1 AND CLEAR XR1
       TXI     *+1,4,-1
       LXA     AMASK,1
* LOOP TO MOVE ALONG ARGUMENTS 1 OR 2 AT A TIME
 TXI1  TXI     *+1,1,1        XR1 COUNTS COMPARISONS
       CAL     2,4            TSX Y1,0  TSX Y2,0 ...
       TSX     TSXZCK,2
       TRA     PXDZ           RUN OFF
       CAL     1,4            MAYBE OK
       TSX     TSXZCK,2
       TRA     PXDZ           RUN OFF
       TRA     ISARG          NEW PAIR
* SET IANS=0  IF RUN OFF ARGUMENTS, AND LEAVE
 PXDZ  PXD     0,0
* EXIT
STOANS STO     **             ** = A(IANS)
       LXD     CMPARP-3,2
       LXD     CMPARP-4,1
 EXIT  TRA     **,4           ** = 1(CMPARP) OR 2(CMPARS)
* FOR MORE ARGUMENTS MAKE COMPARISON
 ISARG CLA*    1,4            X1,X2,... (EITHER ENTRY)
       TNZ     CAS
       NZT*    2,4            SPECIAL TREATMENT
       TRA     TXI2           FOR FIRST MAGNITUDE ZERO
 CAS   CAS*    2,4            Y2,Y3,... OR X2,X3,...
       TRA     *+2
 TXI2  TXI     TXI1,4,**      ** = -2  OR  -1
* COMPARISON FAILS, RUN TO END OF ARGUMENTS, SET IANS, EXIT
 CALX  CAL     **,4           ** = 1(CMPARP) OR 2(CMPARS)
       TSX     TSXZCK,2
       TRA     *+2            END
       TXI     CALX,4,-1      MORE
       PXD     0,1            END
       TRA     STOANS
* INTERNAL SUBROUTINE TO CHECK IF AC = TSX X,0
*     LINKAGE XR2
*     RETURNS TO 1,2 IF NOT,  TO 2,2 IF SO
TSXZCK ANA     AMASK
       LAS     TSXZ
       TRA     *+2
       TRA     2,2            YES
       TRA     1,2            NO
* CONSTANTS, VARIABLES
 AMASK OCT     777777700000
 TSXZ  TSX     0,0
 KDM1  PZE     2,0,-1
 KDM2  PZE     1,0,-2
       END
```

```
************************
*    CMPARS            *
************************
 REFER TO
     CMPARP
```



```
************************
*    CMPARS            *
************************
 REFER TO
     CMPARP
```

```
*      CMPARV (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0155
*      FAP
*CMPARV
       COUNT   200
       LBL     CMPARV
       ENTRY   CMPARV (V1,V2,LV,IANS)
       ENTRY   CMPARL (V1,V2,LV,IANS)
*
*
*                    ----ABSTRACT----
*
* TITLE - CMPARV  WITH SECONDARY ENTRY CMPARL
*         FAST COMPARE TWO ARBITRARY MODE VECTORS FOR IDENTITY
*
*              CMPARL COMPARES TWO VECTORS, V1(I) AND V2(I)  I=1...LV,
*              ELEMENT BY ELEMENT (36 BIT COMPARISON)  CHECKING FOR
*              IDENTITY.  IT EITHER CONFIRMS THAT THE TWO VECTORS ARE
*              IDENTICAL OR, IF THEY ARE NOT, IT RETURNS THE FIRST INDEX
*              FOR WHICH THE ELEMENTS WERE FOUND TO DIFFER.
*
*              CMPARV IS IDENTICAL TO CMPARL EXCEPT THAT IT CONSIDERS
*              +0 TO BE THE SAME AS -0 .
*
* LANGUAGE   - FAP SUBROUTINES, FORTRAN II COMPATIBLE
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 50 REGISTERS
* SPEED      - (FOR CASES WHERE THE VECTORS MATCH.  LESS OTHERWISE.)
*              62 + 6*LV  MACHINE CYCLES IF  LV  EXCEEDS  1
*                  48  MACHINE CYCLES IF  LV  =  1
* AUTHOR     - S.M. SIMPSON, JULY 1963
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL CMPARV(V1,V2,LV,IANS)
*     CALL CMPARL(V1,V2,LV,IANS)
*
* INPUTS
*
*    V1(I)     I=1...LV  IS FIRST VECTOR (ANY KIND OF MODE)
*
*    V2(I)     I=1...LV  IS SECOND VECTOR (ANY KIND OF MODE)
*
*    LV        MUST EXCEED ZERO
*
* OUTPUTS
*
*    IANS      = +1  MEANS VECTORS IDENTICAL
*              =  0  MEANS LV IS ILLEGAL, NO COMPARISONS MADE
*              = -K  MEANS ELEMENTS NO. K WERE THE FIRST ELEMENTS
*                    FOUND TO DISAGREE.  THE ORDER IN WHICH THE ELEMENTS
*                    ARE COMPARED IS ACCORDING TO THE INDEX ORDER -
*                           1, LV, LV-1, ..., 2
*
*
* EXAMPLES
*
* 1. MISC. CASES OF FLOATING AND FIXED VECTORS
*    INPUTS  - V1(1...5) = 1.,2.,3.,4.,5.    IV1(1...5) = 1,2,3,4,5
*              V2(1...5) = 1.,2.,2.,5.,5.    IV2(1...5) = 2,2,3,5,5
*              X1= OCT000000000000    X2= OCT400000000000
*    USAGE   -       CALL CMPARV(V1,V2,2,IANS1)
*                    CALL CMPARL(V1,V2,2,JANS1)
*                    CALL CMPARV(IV1,IV1,5,IANS2)
*                    CALL CMPARL(IV1,IV1,5,JANS2)
*                    CALL CMPARV(V1,V2,5,IANS3)
*                    CALL CMPARL(V1,V2,5,JANS3)
*                    CALL CMPARV(IV1,IV2,5,IANS4)
*                    CALL CMPARL(IV1,IV2,5,JANS4)
*                    CALL CMPARV(X1,X2,1,IANS5)
*                    CALL CMPARL(X1,X2,1,JANS5)
```

CMPARV
(PAGE 2)

```
*                    CALL CMPARV(V1,V2,0,IANS6)
*    OUTPUTS - IANS1=IANS2=JANS1=JANS2=+1
*              IANS3=JANS3 = -4    (4TH ELEMENT IS FIRST MISMATCH FOUND)
*              IANS4=JANS4 = -1    (1ST ELEMENT IS FIRST MISMATCH FOUND)
*              IANS5=+1  (DOESN'T DISTINGUISH +0 AND -0
*              JANS5 = -1          (DISTINGUISHES +0 AND -0)
*              IANS6 = 0           (ILLEGAL LV)
*
*
* PROGRAM FOLLOWS BELOW
*
       HTR      0               XR1
       HTR      0               XR4
       BCI      1,CMPARV
* PRINCIPAL ENTRY.  CMPARV(V1,V2,LV,IANS)
CMPARV STZ      ZIFCV
       TRA      SETUP
* SECOND ENTRY.  CMPARL(V1,V2,LV,IANS)
CMPARL SXD      ZIFCV,4
 SETUP SXD      CMPARV-2,4
       SXD      CMPARV-3,1
* CHECK LV
       STZ      IANS
       CLA*     3,4             LV
       TMI      LEAVE
* SET LOOP INDEX REGISTER WITH LV
       PDX      0,1
       TXL      LEAVE,1,0
* FIRST COMPARE V1(1) WITH V2(1) SINCE COMPARE LOOP DOESNT DISTINGUISH
*     1. ALL ELEMENTS MATCHING
* FROM
*     2. ALL ELEMENTS BUT FIRST MATCHING
       CLA      KD1
       STO      IANS
       CLA*     1,4
       CAS*     2,4
       TRA      DIFF            NO
       TRA      NO1OK           YES
* DISREGARD ZERO MISMATCH FOR CMPARV
 DIFF  NZT      ZIFCV           NO
       TZE      NO1OK
       TRA      LEAVE
* THEN EXIT IF LV=1
 NO1OK CLS      KD1
       STO      IANS
       TXL      LEAVE,1,1
* OTHERWISE SET ADDRESSES AND ENTER LOOP
       CLA      1,4             A(V1)
       ADD      K1
       STA      CLA
       CLA      2,4             A(V2)
       ADD      K1
       STA      CAS
* COMPARE LOOP (6 CYCLES PER CHECK)
 CLA   CLA      **,1            **=A(V1)+1
 CAS   CAS      **,1            **=A(V2)+1
       TRA      *+2             MISMATCH
 TIX   TIX      CLA,1,1         MATCH
* WHEN IT GETS HERE IT EITHER FELL THRU LOOP (OK)
*   OR JUMPED OUT (MISMATCH).  WE HAVE TO CHECK WHICH.
       TXL      LEAVE,1,1     (XR1=1 I.F.F. FELL THRU)
* JUMPED OUT.  WE MUST DISREGARD MISMATCH FOR
* THE ENTRY CMPARV IF THE MAGNITUDES ARE ZERO.
       ZET      ZIFCV
       TRA      PXD
       NZT*     CAS
       TZE      TIX
* OTHERWISE SET IANS
 PXD   PXD      0,1
       STO      IANS
* EXIT, SETTING IANS
 LEAVE CLS      IANS
       STO*     4,4
       LXD      CMPARV-3,1
       TRA      5,4
```

```
     * VARIABLES, CONSTANTS                                                    0150
      ZIFCV PZE      0,0,**          **=0 IF CMPARV, NOT=0 IF CMPARL           0151
      IANS  PZE      0,0,**                                                    0152
      KD1   PZE      0,0,1                                                     0153
      K1    PZE      1                                                         0154
            END                                                                0155
```

```
*       CMPRA  (FUNCTION)                  9/4/64   LAST CARD IN DECK IS NO. 0103
*       FAP
*CMPRA
        COUNT   100
        LBL     CMPRA
        ENTRY   CMPRA   F(X1,X2)
        ENTRY   XCMPRA  F(X1,X2)
        ENTRY   CMPRFL  F(X1,X2)
*
*
*                    ----ABSTRACT----
*
* TITLE - CMPRA WITH SECONDARY ENTRIES XCMPRA AND CMPRFL.
*         COMPARE ARITHMETICALLY TWO WORDS, -0 LSTHN +0 .
*
*               CMPRA COMPARES TWO WORDS X1 AND X2 TO SEE IF X1 IS LSTHN,
*               EQUAL TO, OR GRTHN X2.  -0 IS CONSIDERED LSTHN +0 .
*
*               XCMPRA IS IDENTICAL TO CMPRA.
*
*               CMPRFL COMPARES THE CHARACTERISTIC AND 22 MOST
*               SIGNIFICANT BINARY DIGITS OF TWO FLOATING POINT NUMBERS.
*
*
* LANGUAGE   - FAP FUNCTIONS, FORTRAN-II COMPATIBLE
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 18 REGISTERS
* SPEED      - 16 OR 26 MACHINE CYCLES ON 7090.
* AUTHOR     - R.A. WIGGINS,  11/63
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     X  = CMPRA F(X1,X2)
*     JX = XCMPRAF(X1,X2)
*     FL = CMPRFLF(X1,X2)
*
*
* INPUTS
*
*    X1         IS A WORD IN ANY MODE.
*
*    X2         IS A WORD IN ANY MODE.
*
*
* OUTPUTS
*
*    X          = -1 (FIXED POINT) IF X1 LSTHN X2.  (-0 LSTHN +0)
*               =  0               IF X1 = X2.
*               = +1 (FIXED POINT) IF X1 GRTHN X2.
*
*    JX         = -1 IF X1 LSTHN X2. (-0 LSTHN +0)
*               =  0 IF X1 = X2.
*               = +1 IF X1 GRTHN X2.
*
*    FL         = -1 (FIXED POINT) IF TX1 LSTHN TX2. (-0 LSTHN +0).
*               =  0               IF TX1 = TX2.
*               = +1 (FIXED POINT) IF TX1 GRTHN TX2.
*                 WHERE TX1 AND TX2 REPRESENT THE CHARACTERISTIC AND
*                 MOST SIGNIFICANT 22 BINARY DIGITS OF X1 AND X2.
*
*
* EXAMPLES
*
* 1. INPUTS  - X1 = 6HABCDEF  X2 = 6HABCDEF
*    USAGE   -        X = CMPRAF(X1,X2)
*                     JX=XCMPRAF(X1,X2)
*    OUTPUTS - X = 0,  JX = 0
*
* 2. INPUTS  - X1 = 1.2345678   X2 = 1.2345679
*    USAGE   -        X = CMPRFLF(X1,X2)
```

```
      *    OUTPUTS - X = 0
      *
      * 3. INPUTS  - X1 = 0.  X2 = -0.
      *    USAGE   -        X = CMPRAF(X1,X2)
      *    OUTPUTS - X = +1
      *
      *
      * PROGRAM FOLLOWS BELOW
      *
      *
             BCI      1,CMPRA
      XCMPRA BSS      0
       CMPRA SXA      LV,4
             AXT      0,4
             STQ      TEMP
             CAS      TEMP
             TXI      *+1,4,1
             TXI      *+1,4,1
             PXD      ,4
             SUB      =1B17
       LV    AXT      **,4
             TRA      1,4
      CMPRFL ARS      6
             ALS      6
             LGR      6
             LGL      6
             TRA      CMPRA
       TEMP  PZE
             END
```

```
************************
*    CMPRFL            *
************************
REFER TO
    CMPRA
```


```
************************
*    CMPRFL            *
************************
REFER TO
    CMPRA
```

```
*     CNTRDB (SUBROUTINE)               9/9/64   LAST CARD IN DECK IS NO. 0250
*     LABEL
CCNTRDB
      SUBROUTINE CNTRDB(ITAPE,ISENSE,GZFAMP,VOFXY,LXV,
     1                  LYV,LXDIM,VZERO,SPACE,IANS)
C
C
C                    ----ABSTRACT----
C
C TITLE - CNTRDB
C        CONTOUR A MATRIX ON THE PRINTER IN DECIBELS
C
C              CNTRDB IS DESIGNED PRIMARILY TO PLOT AMPLITUDE OR
C              POWER SPECTRA IN TWO DIMENSIONS OVER ONE-HALF THE
C              TWO-DIMENSIONAL PLANE.
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094  (MAIN FRAME PLUS ONE TAPE UNIT AND
C                 POSSIBLY THE ON-LINE PRINTER)
C STORAGE    - 550 REGISTERS
C SPEED      - A 25 BY 50 MATRIX TAKES ABOUT 15 SECONDS ON THE 7094.
C AUTHOR     - S.M.SIMPSON, MARCH 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - SETVEC,CONTUR,SAME
C       AND FORTRAN SYSTEM ROUTINES - LOG,EXP,(FIL),(STH)
C
C FORTRAN USAGE
C     CALL CNTRDB(ITAPE,ISENSE,GZFAMP,VOFXY,LXV,LYV,LXDIM,VZERO,
C    1            SPACE,IANS)
C
C INPUTS
C
C    ITAPE     IS LOGICAL OUTPUT TAPE NO.
C              MUST EXCEED ZERO AND BE LESS THAN 20 .
C
C    ISENSE    IS IGNORED UNLESS IT LIES IN THE RANGE 1-6 .
C                 IF IT IS IN THE RANGE 1-6 THEN DEPRESSING
C                 SENSE SWITCH ISENSE WILL CAUSE ON-LINE
C                 MONITORING OF THE CONTOURING (ONLY WHILE DEPRESSED).
C
C    GZFAMP    GRTHN 0 INDICATES THE DATA IN VOFXY IS AMPLITUDE DATA.
C              EQUAL 0 INDICATES THE DATA IN VOFXY IS POWER     DATA.
C              LSTHN 0 INDICATES THE DATA IN VOFXY IS DECIBEL   DATA.
C
C    VOFXY(IX,IY) IX=1...LXV, IY=1...LYV  THE DATA MATRIX TO BE
C                 CONTOURED.
C              VALUES SHOULD EXCEED ZERO UNLESS GZFAMP LSTHN 0.
C
C    LXV       SHOULD EXCEED 1 .
C
C    LYV       SHOULD EXCEED 1 .
C
C    LXDIM     IS THE DIMENSION IN THE USER'S PROGRAM OF THE INDEX IX
C                 IN VOFXY(IX,IY).
C              MUST EQUAL OR EXCEED LXV.
C
C    VZERO     IS IGNORED IF GZFAMP IS LESS THAN ZERO.  OTHERWISE THE
C                 DECIBELS WILL BE COMPUTED BY
C                    DB = 20 LOG(VOFXY/VZERO)   IF GZFAMP EXCEEDS 0.0
C                    DB = 10 LOG(VOFXY/VZERO)   IF GZFAMP EQUALS  0.0
C
C    SPACE(I)  I=1...LSPACE  MUST BE AVAILABLE FOR SCRATCH WHERE
C              LSPACE = 204 + LXV + XMAXOF(4,484/LXV)
C
C
C OUTPUTS      NO OUTPUTS OR ONLY PARTIAL ONES IF IANS IS NON-ZERO.
C
C              THE PRINCIPAL OUTPUTS ARE OFF-LINE AND POSSIBLY ON-LINE.
C              THE CONTOUR AREA OCCUPIES 121 COLUMNS (12 INCHES) AND
C              145 ROWS (24 INCHES).
C              VOFXY(1,  1) IS UPPER LEFT, VOFXY(LXV,  1) IS UPPER RIGHT
```

```
C             VOFXY(1,LYV) IS LOWER LEFT, VOFXY(LXV,LYV) IS LOWER RIGHT
C
C             VOFXY IS CONVERTED INTO DECIBELS (IF GZFAMP GRTHN=0),
C             CONTOURED, AND THEN RECONVERTED TO ITS ORIGINAL UNITS.
C
C             THE CONTOUR LEVELS AND CORRESPONDING CHARACTERS ARE
C                         DB=0.0 IS THE CHARACTER Z
C
C                 DB   CHAR       DB   CHAR       DB   CHAR       DB   CHAR
C               -1.0    1       -17.0   1         1.0   A        17.0   K
C               -2.0    2       -20.0   2         2.0   B        20.0   L
C               -3.0    3       -25.0   3         3.0   C        25.0   M
C               -4.0    4       -30.0   4         4.0   D        30.0   N
C               -5.0    5       -35.0   5         5.0   E        35.0   O
C               -6.0    6       -40.0   6         6.0   F        40.0   P
C               -8.0    7       -50.0   7         8.0   G        50.0   Q
C              -10.0    8       -60.0   8        10.0   H        60.0   R
C              -12.0    9       -70.0   9        12.0   I        70.0   S
C              -14.0    0       -80.0   0        14.0   J        80.0   T
C
C
C             THE ABOVE TABLE IS PRINTED FOLLOWING THE CONTOURS.
C
C    VOFXF(IX,IY) MAY BE LEFT SLIGHTLY MODIFIED.
C
C
C    IANS     =     0    IF ALL OK
C             =    -1    IF LXV   ILLEGAL
C             =    -2    IF LYV   ILLEGAL
C             =    -3    IF LXDIM ILLEGAL
C             =    -4    IF VZERO ILLEGAL
C             = -100+K  IF CONTUR FOUND AN ERROR IT FLAGGED WITH IANS=K
C
C EXAMPLES
C
C 1. IN THIS EXAMPLE WE SET UP A  25*51  MATRIX WHICH APPROXIMATES THE
C    WEIGHTED SUPERPOSITION OF  3  POINT SOURCE FIELDS EMANATING FROM
C    INDICES  (0,10),  (25,52),  AND  (15,25).
C
C    USAGE   -       DIMENSION  VOFXY(30,51), SPACE(1000)
C                    DO 20  I=1,25
C                    DO 20  J=1,51
C                    RAD1  = SQRTF(FLOATF(      I**2 + (J-10)**2)) - .5
C                    RAD2  = SQRTF(FLOATF((I-25)**2 + (J-52)**2))
C                    RAD3  = SQRTF(FLOATF((I-15)**2 + (J-25)**2)) + .5
C                20  VOFXY(I,J) = 50.0/RAD1 + 100.0/RAD2 + 25.0/RAD3
C                    CALL CNTRDB(2,1,1.0,VOFXY,25,51,30,35.,SPACE,IANS)
C
C    OUTPUTS - IANS = 0,  A PAGE RESTORE OCCURS ON LOGICAL  2,  THREE
C             ROWS OF COLUMN LABELLING ARE PRINTED,  145  ROWS OF
C             CONTOURING OCCUR,  3  MORE ROWS OF COLUMN LABELLING ARE
C             PRINTED,  3  BLANK ROWS OCCUR, AND  4  ROWS GIVING
C             CONTOUR CODING ARE PRINTED.  THE FIRST  67  COLUMNS OF
C             THE FIRST  48  PRINTED ROWS APPEAR AS SHOWN BELOW.
C
C          0000000000000000000000000000000000000000000000000000000000000
C          0000000000111111111122222222223333333333444444444455555555556
C          0123456789012345678901234567890123456789012345678901234567890
C       -72                                       0
C       -71                                          0
C       -70         9                                  0
C       -69              9                               0
C       -68                  9                             0
C       -67                     9                            0
C       -66                        9                           0
C       -65                          9                          0
C       -64      8                      9                         0
C       -63          8                     9                          0
C       -62              8                   9                         0
C       -61                8                   9                         0
C       -60  7               8                  9
C       -59       7            8                  9
C       -58          7           8                 9
C       -576           7          8                9
C       -56     6        7          8              9
```

```
C       -55   5   6        7         8              9
C       -54  4   5  6       7         8              9
C       -53  3  4  5  6      7         8              9
C       -521 2  3 4 5  6      7        8               9
C       -51 Z 12 3 4  5 6      7        8               9
C       -50B AZ1 23 4 5  6     7         8              9
C       -49DCBAZ1 23 4 5 6      7         8              9
C       -48FE*BAZ12 34 5  6     7          8              9
C       -47GF*C*Z12 3 4 5 6     7          8               9
C       -46HG**B*1 23 4 5 6      7         8                9
C       -45GF*C*Z12 3 4 5 6      7         8                9
C       -44F*CBAZ12 34  5 6     7          8                 9
C       -43DCBAZ1 23 4 5  6     7          8                 9
C       -42BA Z1 23 4  5 6      7          8                  9
C       -41 Z1 2 3 4  5 6      7          8                  9
C       -401 2  3 4  5  6      7          8                  9
C       -39  3  4  5  6      7          8                   9
C       -38   4  5   6      7           8                   9
C       -37    5   6      7            8                    9
C       -36      6      7            8                      9
C       -35  6       7             8                         9
C       -34            7          8                           9
C       -33          7          8                              9
C       -32       7           8                                 9
C       -31 7              8                                     9
C       -30              8                                         9
C       -29            8
C       -28          8
C
C
C PROGRAM FOLLOWS BELOW.
C
C
      DIMENSION  VOFXY(2),SPACE(41,3)
C
C BRING IN AND CHECK SOME STUFF.
C
      LX=LXV
      LY=LYV
      LXDM=LXDIM
      VZER=VZERO
 10   IANSR=-1
      IF (LX-2)                9999,20,20
 20   IANSR=-2
      IF (LY-2)                9999,30,30
 30   IANSR=-3
      IF (LXDM-LX)             9999,40,40
C
C CONSTRUCT DB AND CHARACTER TABLES
C
 40   CALL SETVEC(SPACE(1,1),-80.,-70.,-60.,-50.,-40.,-35.,-30.,-25.,
     1 -20.,-17.,-14.,-12.,-10.,-8.,-6.,-5.,-4.,-3.,-2.,-1.,0.,1.,2.,3.,
     2 4.,5.,6.,8.,10.,12.,14.,17.,20.,25.,30.,35.,40.,50.,60.,70.,80.)
      CALL SETVEC(SPACE(1,2),1H0,1H9,1H8,1H7,1H6,1H5,1H4,1H3,1H2,1H1,
     1 1H0,1H9,1H8,1H7,1H6,1H5,1H4,1H3,1H2,1H1,1HZ,1HA,1HB,1HC,1HD,1HE,
     2 1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT)
C
C CONVERT TO DBS IF NECESSARY.
C
      CONST=8.6858896
      IF (GZFAMP)              110,60,65
 60   CONST=4.3429448
 65   IANSR=-4
      IF (VZER)                9999,9999,70
 70   DO 100   IX=1,LX
      DO 100   IY=1,LY
      IV=IX+LXDM*(IY-1)
      VOFXY(IV)=CONST*LOGF(VOFXY(IV)/VZER)
 100  CONTINUE
C
C FORM THE PLOT
C
 110  CALL CONTUR(ITAPE,ISENSE,VOFXY,LX,LY,LXDM,1.0,FLOATF(LX),
     1           121,0,1.0,FLOATF(LY),145,-72,1,0,
     2           SPACE(1,2),41,0.0,SPACE(1,1),SPACE(1,3),IANSR)
```

```
      IF (IANSR)             140,150,140
  140 IANSR=IANSR-100
      GO TO 9999
C
C RESTORE VOFXY IF NECESSARY.
C
  150 IF (GZFAMP)            220,160,160
  160 DO 200    IX=1,LX
      DO 200    IY=1,LY
      IV=IX+LXDM*(IY-1)
      VOFXY(IV)=VZER*EXPF(VOFXY(IV)/CONST)
  200 CONTINUE
C
C RECORD SCALES
C
  220 DO 230    I=1,41
  230 SPACE(I,1)=SAMEF(XFIXF(SPACE(I,1)))
      WRITE OUTPUT TAPE ITAPE,240,(SPACE(I,1),SPACE(I,2),I=1,41)
  240 FORMAT(///30H CONTOUR CODING USED ABOVE IS ,/,
     1            (10(I4,5HDB = ,A1,1H,)))
C
C EXIT, SETTING IANS.
C
 9999 IANS=IANSR
      RETURN
      END
```

```
*     CNTROW (SUBROUTINE)                    9/9/64   LAST CARD IN DECK IS NO. 0520
*     LABEL
CCNTROW
      SUBROUTINE CNTROW(VEC,LVEC,FXLO,FXHI,NCOLS,CHLVLS,NCHRS,DELEVL,
     1                  VLEVL,SPACE,PLOTVC,IANS)
C
C
C                    ----ABSTRACT----
C
C TITLE - CNTROW
C        FIND CONTOUR LEVELS FOR PLOTTING A ROW OF DATA
C
C              CNTROW CONSIDERS A GIVEN VECTOR RANGE AS CONTINUING DATA,
C              CONTINUITY TO BE PROVIDED BY CUBIC INTERPOLATION BETWEEN
C              THE POINTS.  ANOTHER GIVEN VECTOR PROVIDES DESIRED
C              CONTOUR LEVEL VALUES.  ALTERNATIVELY THE LEVELS ARE
C              DEFINED BY A GIVEN LEVEL AND AN INCREMENT.  IN ANY CASE
C              FOR EACH SUCH LEVEL CNTROW INTERPOLATES THE FIRST VECTOR
C              TO FIND ALL CORRESPONDING INDICES (FRACTIONAL IN
C              GENERAL).  THESE INDICES ARE ROUNDED TO UNITS
C              CORRESPONDING TO COLUMN NUMBERS ON A PRINTED PAGE AND FOR
C              EACH SUCH INDEX A HOLLERITH CHARACTER (A SEPARATE
C              CHARACTER FOR EACH LEVEL VALUE IS PROVIDED BY A THIRD
C              VECTOR) IS INSERTED INTO THE APPROPRIATE POSITION OF A
C              HOLLERITH VECTOR.  THIS HOLLERITH VECTOR (WHICH WILL BE
C              ALL SPACES IF NO CONTOUR LEVELS INTERSECT THE DATA) IS
C              THE ONLY OUTPUT OF CNTROW.  IN THE CASE THAT 2 LEVELS TRY
C              TO CROWD INTO ONE COLUMN POSITION, AN ASTERISK IS
C              INSERTED.  IF MORE THAN 2, A DOLLAR SIGN IS INSERTED.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094 (MAIN FRAME ONLY)
C STORAGE    - 802 REGISTERS
C SPEED      - TAKES ON THE ORDER OF  1/10  SECOND ON THE  7090  FOR
C                  A  120  COLUMN ROW WITH VECTOR LENGTH  50 .
C AUTHOR     - S.M.SIMPSON, MARCH 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CUFIT1,QUFIT1,FASCUB
C                                     RND,RNDDN,RNDUP
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C     CALL CNTROW(VEC,LVEC,FXLO,FXHI,NCOLS,CHLVLS,NCHRS,DELEVL,
C    1            VLEVL,SPACE,PLOTVC,IANS)
C
C
C INPUTS
C
C    VEC(I)    I=1...LVEC IS THE DATA TO BE CONTOURED.  THE RANGE OF THE
C                    INDEX I INVOLVED IS SPECIFIED BY FXLO AND FXHI.
C
C    LVEC      MUST BE GRTHN= 2
C
C    FXLO      IS A FLOATING POINT NUMBER (MAY BE FRACTIONAL) WHICH
C                    REPRESENTS THE INDEX I OF VEC(I) WHICH IS TO
C                    CORRESPOND TO THE FIRST COLUMN OF THE OUTPUT
C                    (IN PLOTVC(1)).
C              FXLO MUST BE GRTHN= 1.0 .
C
C    FXHI      REPRESENTS THE INDEX I OF VEC(I) WHICH IS TO CORRESPOND
C                    TO THE LAST COLUMN OF THE OUTPUT (IN PLOTVC(NCOLS))
C              FXHI MUST EXCEED FXLO, AND MUST BE LSTHN= LVEC.
C
C    NCOLS     IS THE NO. OF COLUMNS OF THE OUTPUT.
C              MUST EXCEED ONE.
C
C    CHLVLS(I) I=1,2,...,NCHRS  GIVES THE CHARACTERS USED FOR PLOTTING
C                    THE CONTOUR LEVELS, EACH REGISTER IN FORMAT (A1).
C                    THE CHARACTERS * AND $ SHOULD NOT APPEAR HERE SINCE
C                    CNTROW USES THEM TO INDICATE CONFLICT OF CONTOURS.
C                    THE RELATION OF CHLVLS TO THE ACTUAL LEVELS DEPENDS
C                    ON THE MANNER IN WHICH THE LEVELS ARE DEFINED, AS
```

```
C                      SPECIFIED BELOW.
C
C     NCHRS      IS THE NO. OF CHARACTERS IN THE CHLVLS VECTOR.  IT IS
C                      ALSO THE NO. OF CONTOUR LEVELS IN THE CASE
C                      DELEVL=0.0 AS DESCRIBED BELOW.
C                MUST EXCEED ZERO.
C
C     DELEVL     INDICATES THE MANNER OF CONTOUR LEVEL SPECIFICATION.  IF
C                      DELEVL IS NON-ZERO, SUCCESSIVE CONTOUR LEVELS ARE
C                      ASSUMED TO BE SEPARATED BY DELEVL UNITS WITH A
C                      STARTING VALUE OF VLEVL (DELEVL MAY NOT BE
C                      NEGATIVE).  ON THE OTHER HAND IF DELEVL IS ZERO,
C                      VLEVL IS INTERPRETED AS A VECTOR GIVING A FIXED SET
C                      OF ARBITRARY LEVELS.
C                MUST BE GRTHN= 0.0 .
C
C     VLEVL(I)   I=... HAS INTERPRETATION DEPENDING ON DELEVL.
C                      IF DELEVL IS NON-ZERO, VLEVL IS A SINGLE CONSTANT
C                      VLEVL(1), WHICH IS THE CONTOUR LEVEL OF VEC TO BE
C                      ASSOCIATED WITH THE CHARACTER CHLVLS(1).
C                      VLEVL+DELEVL IS TO BE ASSOCIATED WITH CHLVLS(2),
C                      ETC.  THIS ASSOCIATION IS TAKEN TO BE CYCLIC IF
C                      NECESSARY, I.E.,
C
C                      VLEVL+(NCHRS-1)*DELEVL HAS CHARACTER CHLVLS(NCHRS)
C                      VLEVL+NCHRS*DELEVL     HAS CHARACTER CHLVLS(1)
C                                ETC.
C                      ALSO
C                      VLEVL-DELEVL           HAS CHARACTER CHLVLS(NCHRS)
C                                ETC.  FOR LOWER LEVELS.
C
C                      IF DELEVL IS ZERO, VLEVL IS TAKEN AS A VECTOR,
C                      VLEVL(1...NCHRS), OF INDIVIDUAL CONTOUR LEVELS
C                      WHICH CORRESPOND 1 TO 1 WITH THE CHARACTERS OF
C                      CHLVLS(1...NCHRS).  THESE CONTOUR LEVELS MUST BE
C                      MONOTONELY INCREASING IN SIZE.
C
C     SPACE(I)   I=1,2,...,LSPACE  MUST BE AVAILABLE FOR SCRATCH,
C                WHERE LSPACE = 2+XMAXOF(4,4*NCOLS/NINDRS),
C                WHERE NINDRS = (FXHI) ROUNDED UP - (FXLO) ROUNDED DOWN.
C
C
C OUTPUTS        (OUTPUT OCCURS ONLY FOR IANS=0)
C
C     PLOTVC(I)  I=1,2,...,NCOLS  WILL BE FILLED WITH CHARACTERS FROM
C                      CHLVLS(I), BLANKS, AND POSSIBLY ASTERISKS AND
C                      DOLLAR SIGNS, ALL IN FORMAT(1A1)
C
C     IANS       =  0  NORMALLY
C                = -1  FOR ILLEGAL LVEC
C                = -2  FOR ILLEGAL FXLO
C                = -3  FOR ILLEGAL FXHI
C                = -4  FOR ILLEGAL NCOLS
C                = -5  FOR ILLEGAL NCHRS
C                = -6  FOR ILLEGAL DELEVL
C                = -7  FOR ILLEGAL VLEVL  (NOT MONOTONE IN CASE DELEVL=0.)
C
C
C EXAMPLES             WE SHALL ASSUME THE FOLLOWING VECTORS AS INPUTS.
C
C               VEC2(1...2)    = 0.0,10.0
C               VEC3(1...3)    = 0.0,5.0,10.0
C               VEC4(1...4)    = 0.0,3.33333333,6.66666667,10.0
C               VEC5(1...5)    = 0.0,2.5,5.0,7.5,10.0
C               VEC9(1...9)    = -47.0,-13.0,3.0,7.0,5.0,3.0,7.0,23.0,57.0
C               VEC2R(1...2    = 10.0,0.0
C               VEC3R(1...3)   = 10.0,5.0,0.0
C               VEC4R(1...4)   = 10.0,6.66666667,3.33333333,0.0
C               VEC5R(1...5)   = 10.0,7.5,5.0,2.5,0.0
C               CLVLS1(1...10) = 1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9
C               CLVLS2(1...11) = 1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,
C                                1H0
C               CLVLS9(1...14) = 1HD,1HC,1HB,1HA,1H0,1H1,1H2,1H3,1H4,1H5,
C                                1H6,1H7,1H8,1H9
```

```
C              VLEVL2(1...11) = 0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,
C                                10.0
C
C 1. USAGES  -        CALL CNTROW(VEC2,2,1.,2.,21,CLVLS1,10,1.,0.0,SPACE,
C                    1            PLV2A,INS2A)
C                     CALL CNTROW(VEC3,3,1.,3.,21,CLVLS1,10,1.,0.0,SPACE,
C                    1            PLV3A,INS3A)
C                     CALL CNTROW(VEC4,4,1.,4.,21,CLVLS1,10,1.,0.0,SPACE,
C                    1            PLV4A,INS4A)
C                     CALL CNTROW(VEC5,5,1.,5.,21,CLVLS1,10,1.,0.0,SPACE,
C                    1            PLV5A,INS5A)
C    OUTPUTS - PLV2A(1...21)=...=PLV5A(1...21)=21H0 1 2 3 4 5 6 7 8 9 0
C              INS2A=...=INS5A = 0
C
C 2. USAGES  -        CALL CNTROW(VEC2,2,1.,2.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV2B,INS2B)
C                     CALL CNTROW(VEC3,3,1.,3.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV3B,INS3B)
C                     CALL CNTROW(VEC4,4,1.,4.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV4B,INS4B)
C                     CALL CNTROW(VEC5,5,1.,5.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV5B,INS5B)
C    OUTPUTS - PLV2B(1...21)=...=PLV5B(1...21)=21H0 1 2 3 4 5 6 7 8 9 0
C              INS2B=...=INS5B = 0
C
C 3. USAGES  -        CALL CNTROW(VEC2R,2,1.,2.,21,CLVLS1,10,1.,0.0,
C                    1      SPACE,PLV2RA,INS2RA)
C                     CALL CNTROW(VEC3R,3,1.,3.,21,CLVLS1,10,1.,0.0,
C                    1      SPACE,PLV3RA,INS3RA)
C                     CALL CNTROW(VEC4R,4,1.,4.,21,CLVLS1,10,1.,0.0,
C                    1      SPACE,PLV4RA,INS4RA)
C                     CALL CNTROW(VEC5R,5,1.,5.,21,CLVLS1,10,1.,0.0,
C                    1      SPACE,PLV5RA,INS5RA)
C    OUTPUTS - PLV2RA(1...21)=..=PLV5RA(1...21)=21H0 9 8 7 6 5 4 3 2 1 0
C              INS2RA=...=INS5RA = 0
C
C 4. USAGES  -        CALL CNTROW(VEC2R,2,1.,2.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV2RB,INS2RB)
C                     CALL CNTROW(VEC3R,3,1.,3.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV3RB,INS3RB)
C                     CALL CNTROW(VEC4R,4,1.,4.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV4RB,INS4RB)
C                     CALL CNTROW(VEC5R,5,1.,5.,21,CLVLS2,11,0.,VLEVL2,
C                    1      SPACE,PLV5RB,INS5RB)
C    OUTPUTS - PLV2RB(1...21)=..=PLV5RB(1...21)=21H0 9 8 7 6 5 4 3 2 1 0
C              INS2RB=...=INS5RB = 0
C
C 5. USAGE   -        CALL CNTROW(VEC9,9,2.5,8.5,16,CLVLS9,14,5.0,-20.0,
C                    1            SPACE,PLTVC9,IANS9)
C    OUTPUTS - PLTVC9(1...16) = 16HA01   1    12**$     IANS9 = 0
C
C
C PROGRAM FOLLOWS BELOW
C
C
C DUMMY DIMENSIONS
C
      DIMENSION    VEC(2),CHLVLS(2),VLEVL(2),PLOTVC(2),SPACE(2)
C
C TRUE DIMENSIONS
C
      DIMENSION    COEFS(4)
      EQUIVALENCE  (CZ,COEFS(1)),(C1,COEFS(2)),(C2,COEFS(3)),
     1             (C3,COEFS(4))
C
C BRING IN SOME STUFF
C
      LVC=LVEC
      FXL=FXLO
      FXH=FXHI
      NCLS=NCOLS
      NCRS=NCHRS
      DLEVL=DELEVL
      VLVL=VLEVL
C
```

```
C CHECK INPUTS
C
      IANSR=-1
      IF (LVC-2)  9999,5,5
 5    IANSR=-2
      IF (FXL-1.0)  9999,10,10
 10   IANSR=-3
      IF (FXH-FXL)  9999,9999,15
 15   IF (FXH-FLOATF(LVC))  20,20,9999
 20   IANSR=-4
      IF (NCLS-2)  9999,30,30
 30   IANSR=-5
      IF (NCRS)  9999,9999,35
 35   IANSR=-6
      IF (DLEVL)  9999,40,40
 40   IF (DLEVL)  80,50,80
 50   IF (NCRS-1)  80,80,60
 60   IANSR=-7
      N=NCRS-1
      DO 70  I=1,N
      IF (VLEVL(I+1)-VLEVL(I))  9999,9999,70
 70   CONTINUE
 80   IANSR=0
C
C INPUTS OK.  INITIALIZE.
C
C               1. SET INDEX RANGE FOR VEC(I) AND NO. OF INDEX RANGES
C                  FOR LOOP (NINDRS)
C               2. FILL OUTPUT VECTOR WITH SPACES
C               3. INITIALIZE MAJOR LOOP INDEX IXVEC TO IXLO
C               4. SET CONSTANTS COLC1,COLCZ,STAR,DOLAR,SPACES,
C                  DELX,ABSDEL
C
      IXLO=XFIXF(RNDDNF(FXL))
      IXHI=XFIXF(RNDUPF(FXH))
      NINDRS=IXHI-IXLO
      DO 100  I=1,NCLS
B     PLOTVC(I)=606060606060
 100  CONTINUE
      IXVEC=IXLO
      COLC1=(FLOATF(NCLS-1))/(FXH-FXL)
      COLCZ=1.0-FXL*COLC1
      STAR=1H*
      DOLAR=1H$
B     SPACES=606060606060
      NF=XMAXOF(4,4*NCLS/NINDRS)
      FNF=FLOATF(NF)
      DELX1=FNF
      VBIGST=10.0E30
      MXDOLS = 100
      NDOLS = 0
C
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C MAJOR LOOP ON IXVEC (INDEXING AT 700 UP THRU IXHI-1)
C
C     THE FIRST ENTERPRISE IS TO OBTAIN A CUBIC (MAY BE DEGENERATE)
C     WHICH IS VALID FOR THE REGION IXVEC TO IXVEC+1 .  ORDINARILY THE
C     CUBIC IS FITTED TO VEC(IXVEC-1,IXVEC,IXVEC+1,IXVEC+2).  HOWEVER,
C     VEC(IXVEC-1) OR VEC(IXVEC+2) OR BOTH MAY BE UNAVAILABLE IN THE
C     CASE THAT IXVEC LIES AT ONE END OF THE RANGE.
C
C
C VEC(IXVEC-1) IS UNAVAILABLE I.F.F. IXVEC=1
C
 130  IF (IXVEC-1)  160,160,140
C
C IF VEC(IXVEC-1) IS AVAILABLE, VEC(IXVEC+2) WILL BE AVAILABLE ALSO
C     I.F.F. LVC EXCEEDS IXVEC+1 .  JUMP AHEAD TO CUBIC FIT IN THIS CASE
C
 140  IF (LVC-IXVEC-1)  150,150,200
C
C WHEN VEC(IXVEC-1) IS AVAILABLE AND VEC(IXVEC+2) IS NOT, WE SET TO FIT
C     A PARABOLA TO VEC(IXVEC-1),VEC(IXVEC),VEC(IXVEC+1) WITH XLO=-FNF
```

```
C      AND THEN JUMP AHEAD TO FIT IT.
C
 150  IXFS=IXVEC-1
      XLO=-FNF
      GO TO 190
C
C WHEN VEC(IXVEC-1) IS UNAVAILABLE, VEC(IXVEC+2) WILL BE UNAVAILABLE
C      ALSO I.F.F. LVC=2 .
C
 160  IF (LVC-2)  170,170,180
C
C WHEN BOTH VEC(IXVEC-1) AND VEC(IXVEC+2) ARE UNAVAILABLE WE SET THE
C      CUBIC AS A LINEAR SEGMENT SUCH THAT
C               F(0.0)=VEC(IXVEC)         F(FNF)=VEC(IXVEC+1)
C
 170  C2=0.0
      C3=0.0
      C1=(VEC(IXVEC+1)-VEC(IXVEC))/FNF
      CZ=VEC(IXVEC)+C1
      GO TO 210
C
C WHEN VEC(IXVEC-1) IS UNAVAILABLE BUT VEC(IXVEC+2) IS AVAILABLE, WE
C      SET TO FIT A PARABOLA TO VEC(IXVEC),VEC(IXVEC+1),VEC(IXVEC+2)
C      WITH XLO=0.0
C
 180  IXFS=IXVEC
      XLO=0.0
C
C USE QUFIT1 TO FIND THE PARABOLA.
C
 190  C3=0.0
      CALL QUFIT1(VEC(IXFS),XLO,DELX1,COEFS)
      GO TO 210
C
C USE CUFIT1 TO FIND THE CUBIC.
C
 200  IXFS=IXVEC-1
      XLO=-FNF
      CALL CUFIT1(VEC(IXFS),XLO,DELX1,COEFS)
C
C
C MERGE POINT AFTER FINDING CUBIC.
C
C NOW EVALUATE CUBIC F(X) FOR X=0.0,1.0,2.0,...,(FNF-1.0)
C               INTO SPACE(2,3,...,NF+1)
C    EXCEPT GET ONE MORE VALUE AT IXVEC=IXHI-1
C      (NOTE INSERTION RATHER THAN COMPUTATION OF END VALUES)
C
 210  NFEV=NF
      IF (IXVEC+1-IXHI)  220,215,215
 215  NFEV=NF+1
      SPACE(NF+2)=VEC(IXHI)
 220  CALL FASCUB(COEFS,0.,1.,NF,SPACE(2))
      SPACE(2)=VEC(IXVEC)
C
C IF THIS IS FIRST RANGE, WE HAVE SOME INITIALIZING.
C
      IF (IXVEC-IXLO)  230,230,300
C
C THE INITIALIZING CONSISTS OF
C      1. SETTING SPACE(1) = CUBIC F(X) AT X=-1.0
C      2. VA AND VB = CONTOUR LEVELS SUCH THAT
C            VB LSTHN SPACE(1) LSTHN= VA
C      3. IXLEVA = INDEX OF LEVEL VA (VLEVL CORR. TO IXLEVA=0)
C
 230  CALL FASCUB(COEFS,-1.0,1.0,1,SPACE(1))
      TEMP=SPACE(1)
      IF (DLEVL)  270,270,240
C
C CONSTANT INCREMENT CASE
C
 240  IF (TEMP-VLEVL)   260,250,250
 250  IXLEVA = XFIXF(RNDUPF((TEMP-VLEVL)/DLEVL))
 255  VA=VLEVL+FLOATF(IXLEVA)*DLEVL
      VB=VA-DLEVL
```

```
      GO TO 300
  260 IXLEVA = XFIXF(RNDDNF((TEMP-VLEVL)/DLEVL))
      GO TO 255
C
C LIST CASE (SPECIAL CASES IF TEMP LSTHN= VLEVL(1) OR GRTHN VLEVL(NCRS))
C
  270 IXLEVA=0
      VB=-VBIGST
  275 VA=VLEVL(IXLEVA+1)
      IF (TEMP-VA)  300,300,280
  280 IXLEVA=IXLEVA+1
      VB=VA
      IF (IXLEVA-NCRS)  275,285,285
  285 VA=VBIGST
      GO TO 300
C
C INITIALIZE FOR THE SCAN OF
C     SPACE(2),SPACE(3),...,SPACE(NFEV+1)
C
  300 IEQ=0
      IXSP=2
C
C SCAN
C
  320 TEMP=SPACE(IXSP)
      IF (TEMP-VA)  330,325,370
  325 IEQ=1
      GO TO 370
  330 IF (TEMP-VB)  410,335,340
  335 IEQ=1
      GO TO 410
C
C INDEX FOR MORE.  IF NONE, RESET SPACE(1) AND ON TO NEXT IXVEC.
C
  340 IXSP=IXSP+1
      IF (IXSP-NFEV-1)  320,320,350
  350 SPACE(1)=SPACE(NF+1)
      GO TO 700
C
C THIS SEQUENCE RESETS FOR THE CASE WHERE VA WAS EQUALLED OR EXCEEDED
C     (THEN ON TO FIND COLUMN, ETC.)
C
  370 V=VA
      IXLEV=IXLEVA
      VB=VA
      IXLEVA=IXLEVA+1
      IF (DLEVL)  380,380,375
  375 VA=VA+DLEVL
      GO TO 450
  380 IF (IXLEVA-NCRS)  385,390,390
  385 VA=VLEVL(IXLEVA+1)
      GO TO 450
  390 VA=VBIGST
      GO TO 450
C
C THIS SEQUENCE RESETS FOR THE CASE WHERE VB WAS EQUALLED OR SUBCEEDED
C     (THEN ON TO FIND COLUMN NO.)
C
  410 V=VB
      IXLEV=IXLEVA-1
      VA=VB
      IXLEVA=IXLEVA-1
      IF (DLEVL)  425,425,420
  420 VB=VB-DLEVL
      GO TO 450
  425 IF (IXLEVA)  430,430,435
  430 VB=-VBIGST
      GO TO 450
  435 VB=VLEVL(IXLEVA)
      GO TO 450
C
C DETERMINATION OF COLUMN NO.
C
C     DEFINITIONS -  FIXUNF IS INDEX OF CONTOUR WRT 0,1,...,NF-1  (FLTG)
C                    FIXVC  IS INDEX OF CONTOUR WRT IXVEC         (FLTG)
```

CNTROW (PAGE 7)


```
C                       ICOLNO IS COLUMN NO.                    (ROUNDED)
C                         (MUST BE IN RANGE  1 TO NCOLS)
C
C WE HAVE TO WATCH OUT FOR THE CASE IN WHICH SPACE(IXSP) = SPACE(IXSP-1)
C
 450  TEMP=SPACE(IXSP)-SPACE(IXSP-1)
      IF (TEMP)  460,455,460
 455  FIXUNF=FLOATF(IXSP-3)+.5
      GO TO 465
 460  FIXUNF=FLOATF(IXSP-3)+(V-SPACE(IXSP-1))/TEMP
 465  FIXVC=FLOATF(IXVEC)+FIXUNF/FNF
      ICOLNO = XFIXF(RNDF(COLCZ+FIXVC*COLC1))
      IF (ICOLNO)          660,660,470
 470  IF (ICOLNO-NCLS)     500,500,660
C
C CHECK WHETHER OR NOT THIS COLUMN IS ALREADY OCCUPIED.
C
 500  CHLAST=PLOTVC(ICOLNO)
      IOCC = 1
      IF (CHLAST-SPACES)  540,570,540
C
C IT IS.  FIND OUT WHETHER * OR $ OR SOMETHING ELSE.
C      AND ACT ACCORDINGLY.
C
 540  IF (CHLAST-STAR)  550,545,550
 545  CHAR=DOLAR
      GO TO 650
 550  IF (CHLAST-DOLAR)   575,560,575
 560  NDOLS = NDOLS+1
      IF (NDOLS-MXDOLS)   660,660,700
C
C IT IS NOT OCCUPIED YET.
C FIRST TAKE CARE OF THE EASY CASE, DLEVL=0 .
C
 570  IOCC = 0
 575  IF (DLEVL)    580,580,600
 580  CHAR=CHLVLS(IXLEV+1)
      GO TO 627
C
C FOR THE OTHER CASE, THE INDEX FOR CHLVLS IS A MODULO
C      TYPE FUNCTION OF IXLEV.
C
 600  IXCR=XMODF(IXLEV,NCRS)
      IF (IXCR)  620,625,625
 620  IXCR=IXCR+NCRS
 625  CHAR=CHLVLS(IXCR+1)
 627  IF (IOCC)   630,650,630
 630  IF (CHAR-CHLAST)   635,660,635
 635  CHAR = STAR
      GO TO 650
C
C OK.  MOVE THE CHARACTER INTO POSITION
C
 650  PLOTVC(ICOLNO)=CHAR
C
C CHECK IEQ FOR RETURN AND CLEAR IT.
C
 660  IF (IEQ)  665,320,665
 665  IEQ=0
      GO TO 340
C
C INDEX IXVEC AND GO BACK FOR MORE IF WE AREN'T DONE.
C
 700  IANSR=0
      IXVEC=IXVEC+1
      NDOLS = 0
      IF (IXVEC-IXHI)  130,9999,9999
C
C EXIT, SETTING IANS
C
 9999 IANS=IANSR
      RETURN
      END
```

```
*     COLABL (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0123
*     LABEL
CCOLABL
      SUBROUTINE COLABL(ITAPE,ICOLLO,NCOLLO,NCOLS,ISPACE)
C
C
C                    ----ABSTRACT----
C
C TITLE - COLABL
C        LABEL PRINTER COLUMNS WITH INCREASING 3-DIGIT INTEGERS
C
C              COLABL  LABELS A SPECIFIED RANGE OF PRINTER COLUMNS
C              (OFF-LINE)  WITH 3-DIGIT INTEGERS DISPLAYED VERTICALLY,
C              WHERE USER SPECIFIES LEFTMOST INTEGER AND WHERE
C              SUBSEQUENT INTEGERS ARE INDEXED BY UNITY.
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME + 1 TAPE UNIT)
C STORAGE    - 185 REGISTERS
C SPEED      - TAKES ABOUT  1/6  SECOND TO LABEL  130  COLUMNS ON  7094.
C AUTHOR     - S.M.SIMPSON, MARCH 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - GENHOL
C       AND FORTRAN SYSTEM ROUTINES - (SPH),(FIL),(STH)
C
C FORTRAN USAGE
C     CALL COLABL(ITAPE,ICOLLO,NCOLLO,NCOLS,ISPACE)
C
C
C INPUTS
C
C    ITAPE     IS LOG. TAPE UNIT FOR OUTPUT.
C              1 LSTHN= ITAPE LSTHN= 20 .
C
C    ICOLLO    IS COLUMN NO. WRT PRINTER WHERE LABELLING STARTS.
C              MUST BE  GRTHN=  1 .
C
C    NCOLLO    IS LABEL FOR PRINTER COL. NO. ICOLLO.
C              MUST BE  GRTHN=  0 .
C
C    NCOLS     IS NO. OF SUCCESSIVE COLUMNS TO BE LABELLED.
C              MUST BE  GRTHN=  1 .
C
C    ISPACE(I) I=1...NCOLS  IS SCRATCH AREA.
C
C
C OUTPUTS      STRAIGHT RETURN FOR ILLEGAL ITAPE. OTHER INPUTS NOT CHKD.
C
C              ONLY OUTPUT IS 3 LINES OF PRINTED OUTPUT AS ILLUSTRATED
C              IN THE EXAMPLES.
C
C
C EXAMPLES
C
C 1. USAGES  -        DIMENSION  ISPACE(130)
C                     CALL COLABL(2,2,2,130,ISPACE)
C    OUTPUTS - COLS. 2-131 LABELLED   0   000   011   1    OFF-LINE.
C                                     0...011...900...3
C                                     2   901   901   1
C
C 2. USAGES  -        CALL COLABL(2,51,1,15,ISPACE)
C    OUTPUTS - COLS. 51-65 LABELLED
C                                    000000000000000
C                                    000000000111111
C                                    123456789012345
C
C 3. USAGE   -        CALL COLABL(2,17,4,1,ISPACE)
C    OUTPUT  - COL. 17 IS LABELLED 4 .
C
C
```

```
      C PROGRAM FOLLOWS BELOW.
      C
      C
      C DUMMY DIMENSION
      C
            DIMENSION  ISPACE(2)
      C
      C TRUE DIMENSION
      C
            DIMENSION  FMT(3)
      C
      C CHECK INPUTS
      C
            IF (ITAPE)              9999,9999,5
       5    IF (ITAPE-20)           45,45,9999
      C
      C INPUTS OK.  SET UP FORMAT VECTOR.
      C
       45   NBLANK=ICOLLO-1
            IF (NBLANK)  50,50,60
       50   CALL GENHOL(FMT)
            PRINT 55
       55   FORMAT(7H(130I1))
            GO TO 70
       60   CALL GENHOL(FMT)
            PRINT 65,NBLANK
       65   FORMAT(1H(,I3,8HX,130I1))
            GO TO 70
      C
      C SET UP AND EXECUTE LOOPS.
      C
       70   NCOLHI=NCOLLO+NCOLS-1
            DO 100 IROW=1,3
            DO 95 NCOLNO=NCOLLO,NCOLHI
            IHUNS=NCOLNO/100
            ITENS=(NCOLNO-IHUNS*100)/10
            IONES=NCOLNO-IHUNS*100-ITENS*10
            IF (IROW-2)  75,80,85
       75   INO=IHUNS
            GO TO 90
       80   INO=ITENS
            GO TO 90
       85   INO=IONES
            GO TO 90
       90   IXSP=NCOLNO-NCOLLO+1
       95   ISPACE(IXSP)=INO
       99   WRITE OUTPUT TAPE ITAPE,FMT,(ISPACE(I),I=1,NCOLS)
       100  CONTINUE
       9999 RETURN
            END
```

```
*      COLAPS (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0127
*      FAP
*COLAPS
       COUNT   100
       LBL     COLAPS
       ENTRY   COLAPS (X,N,TYPE,XC,M)
*
*                    ----ABSTRACT----
*
* TITLE - COLAPS
*       COLLAPSE ONE-SIDED VECTOR INTO SMALLER RANGE
*
*             COLAPS COLLAPSES A VECTOR X OF LENGTH N TO A VECTOR XC OF
*             LENGTH M.  THE COLLAPSED SERIES IS DEFINED BY
*
*               XC(I) = X(I) + X(I+M) + X(I+2M) + ... + X(I+K(I)*M)
*
*             FOR   I = 1,2,...,M
*                   K(I) = (N/M)   FOR I LSTHN= N(MODULO M)
*                        = (N/M)-1 FOR I GRTHN  N(MODULO M)
*                   N/M IS ROUNDED DOWN
*
*             IF M IS GRTHN N ZEROS ARE FILLED INTO XC FOR ALL I GRTHN N
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 50 REGISTERS
* SPEED     - ABOUT 6N + 14M MACHINE CYCLES
* AUTHOR    - J. CLARK
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL COLAPS(X, N, TYPE, XC, M)
*
* INPUTS
*
*    X(I)       I=1...N IS FLOATING OR FIXED (FORTRAN II) POINT VECTOR
*                 OF NUMBERS.  (NAME NEED NOT BE FLOATING POINT.)
*
*    N          IS FORTRAN II INTEGER.
*               MUST BE GRTHN= 1 .
*
*    TYPE       = 0. IF X IS FIXED POINT.
*               NOT= 0. IF X IS FLOATING POINT.
*
*    M          IS THE LENGTH OF THE COLLAPSED SERIES.
*               IS FORTRAN II INTEGER.
*               MUST BE GRTHN= 1 .
*               MAY  BE GRTHN= N .
*
* OUTPUTS
*
*    XC(I)      I=1...M IS THE COLLAPSED X SERIES.  (NAME NEED NOT BE
*                 FLOATING POINT.)
*               PROGRAM EXITS WITHOUT COMPUTATION IF N OR M IS ILLEGAL.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...6)= 1.,3.,4.,2.,-1.,-2.  N=6  TYPE=1. M=3
*    OUTPUTS - XC(1...3) = 3.,2.,2.
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT M=5
*    OUTPUTS - XC(1...5) = -1.,3.,4.,2.,-1.
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT M=8
*    OUTPUTS - XC(1...8) = 1.,3.,4.,2.,-1.,-2.,0.,0.
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT M=1
*    OUTPUTS - XC(1) = 7.
*
* 5. INPUTS  - X(1...6)  = 1,3,4,2,-1,-2  N=6  TYPE=0.  M=2
```

```
*     OUTPUTS - XC(1...2) = 4,3
*
             PZE
             BCI     1,COLAPS
      COLAPS SXD     COLAPS-2,4
             SXA     G,1
             SXA     G+1,2
             CLA*    3,4
             TZE     K
             CLA     C2
             STO     E
             TRA     A1
      K      CLA     C1
             STO     E
      A1     CLA     1,4
             ADD     D
             STA     E
             CLA     4,4
             ADD     D
             STA     C
             CLA*    2,4
             TMI     G
             TZE     G
             STD     E+2
             STD     C+2
             CLA*    5,4
             TMI     G
             TZE     G
             STD     E+1
             STD     G-1
             STD     B+1
             AXT     1,2
* BASIC LOOP
      A      PXA     0,2
             PAX     0,1
             CLM
      E      NOP     **
             TXI     *+1,1,**
             TXL     *-2,1,**
      C      STO     **,2
             TXI     *+1,2,1
             TXH     B,2,**
             TXL     A,2,**
      G      AXT     **,1
             AXT     **,2
             TRA     6,4             EXIT
      B      CLA     =0
             TXL     C,2,**
             TRA     G
     D       PZE     1
      C1     ADD     0,1
      C2     FAD     0,1
             END
```

```
*     CONTUR (SUBROUTINE)                  9/9/64   LAST CARD IN DECK IS NO. 0641
*     LABEL
CCONTUR
      SUBROUTINE CONTUR(ITAPE,ISENSE,VOFXY,LVX,LVY,LXDIM,FXLO,FXHI,
     1                 NCOLS,NCOLLO,FYLO,FYHI,NROWS,ARGLO,ARGDEL,ZFAFXD,
     2                 CHLVLS,NCHRS,DELEVL,VLEVL,SPACE,IANS)
C
C
C                   ----ABSTRACT----
C
C TITLE - CONTUR
C        CONTOUR OF MATRIX SUBSET ON OFF-LINE PRINTER
C
C              CONTUR CONSIDERS A GIVEN MATRIX AS CONTINUOUS DATA,
C              CONTINUITY AT NON-INTEGRAL INDICES BEING PROVIDED BY
C              CUBIC INTERPOLATION (APPLIED SUCCESSIVELY IN THE TWO
C              DIRECTIONS), AND CONTUR IS ASKED TO CONTOUR AN
C              ARBITRARILY SPECIFIED RECTANGULAR SUBSET OF THE MATRIX
C              INSIDE AN ARBITRARY SIZED RECTANGULAR REGION ON AN
C              OFF-LINE PRINTED OUTPUT PAGE.  A GIVEN VECTOR PROVIDES
C              DESIRED CONTOUR LEVELS, OR ALTERNATIVELY THE LEVELS ARE
C              DEFINED BY A GIVEN LEVEL AND AN INCREMENT.  THE USER ALSO
C              SPECIFIES THE CHARACTERS USED TO CONTOUR EACH LEVEL.
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094  (MAIN FRAME PLUS ONE TAPE UNIT)
C STORAGE    - 587 REGISTERS
C SPEED      - A 120 COLUMN, 150 ROW CONTOUR OF A 25 BY 50 MATRIX TAKES
C                    ABOUT 20 SECONDS ON THE 7090.
C              AN 80 COLUMN, 48 ROW CONTOUR OF A 25 BY 25 MATRIX TAKES
C                    ABOUT 2 TO 3 SECONDS.
C AUTHOR     - S.M.SIMPSON, MARCH 1964
C
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - RNDDN,RNDUP,COLABL,ARBCOL,
C                                     CNTROW,XSAME,SWITCH
C      AND FORTRAN SYSTEM ROUTINES - (STH),(FIL),(SPH)
C
C FORTRAN USAGE
C     CALL CONTUR(ITAPE,ISENSE,VOFXY,LVX,LVY,LXDIM,FXLO,FXHI,
C    1            NCOLS,NCOLLO,FYLO,FYHI,NROWS,ARGLO,ARGDEL,
C    2            ZFAFXD,CHLVLS,NCHRS,DELEVL,VLEVL,SPACE,IANS)
C
C
C INPUTS
C
C    ITAPE     IS OUTPUT TAPE NO.
C              MUST EXCEED ZERO AND BE LESS THAN 21 .
C
C    ISENSE    PROVIDES AN ON-LINE MONITORING OPTION UNDER SENSE SWITCH
C                 CONTROL.  IF ISENSE LIES BETWEEN 1 AND 6 INCLUSIVELY
C                 THEN THE OUTPUT APPEARS ON-LINE AS WELL AS OFF-LINE
C                 WHILE THE CORRESPONDING SENSE SWITCH IS DOWN.  OTHER
C                 VALUES OF ISENSE ARE IGNORED.
C
C    VOFXY(IX,IY) IX=1,2,...,LVX  IY=1,2,...,LVY  IS THE DATA, PART OR
C                 ALL OF WHICH IS TO BE CONTOURED.
C
C    LVX       MUST BE GRTHN= 2 .
C
C    LVY       MUST BE GRTHN= 2 .
C
C    LXDIM     IS THE VALUE TO WHICH IX IN VOFXY(IX,IY) IS DIMENSIONED
C                 IN THE CALLING PROGRAM.
C              MUST BE GRTHN= LVX (EQUALS LVX IF MATRIX IS TIGHT PACKED)
C
C    FXLO      IS A FLOATING POINT NUMBER (MAY BE FRACTIONAL) WHICH
C                 REPRESENTS THE VALUE OF THE INDEX IX IN VOFXY(IX,IY)
C                 WHICH IS TO CORRESPOND TO THE LEFTMOST COLUMN IN THE
C                 OUTPUT CONTOUR AREA.  DATA WITH INDEX IX LESS THAN
C                 FXLO IS NOT CONTOURED.
```

```
      C             FXLO MUST BE GREATER THAN OR EQUAL TO 1.0 .
      C
      C     FXHI    SIMILARLY IS THE VALUE OF IX CORRESPONDING TO THE
      C                RIGHTMOST OUTPUT COLUMN.  DATA WITH INDEX IX
      C                GREATER THAN FXHI IS NOT CONTOURED.
      C             FXHI MUST BE LESS THAN OR EQUAL TO LVX, AND MUST EXCEED
      C                FXLO.
      C
      C     NCOLS   SPECIFIES HOW MANY COLUMNS THE OUTPUT IS TO OCCUPY.
      C                THE COLUMN NUMBERS OCCUPIED BY THE PLOT WILL BE
      C                   ICOLLO,ICOLLO+1,...,ICOLLO+NCOLS-1
      C                WHERE ICOLLO= 7 IF ZFAFXD=0.0, (SEE ZFAFXD BELOW)
      C                            =14 IF ZFAFXD NOT =0.0 .
      C             MUST BE GRTHN= 2, AND LSTHN= 125 .
      C
      C     NCOLLO  IS A LABELLING INDEX FOR COLUMN NO. ICOLLO AS DEFINED
      C                ABOVE.  EACH OUTPUT COLUMN USED WILL BE LABELLED BY
      C                A 3-DIGIT INTEGER (EXPRESSED VERTICALLY) STARTING FROM
      C                THE INTEGER NCOLLO, AND INDEXED BY UNITY THRU
      C                NCOLLO+NCOLS-1 .  COLUMN LABELLING IMMEDIATELY
      C                PRECEEDS THE FIRST OUTPUT ROW AND FOLLOWS THE LAST
      C                OUTPUT ROW.
      C             SHOULD BE NON-NEGATIVE AND BE LESS THAN (1000-NCOLS).
      C
      C     FYLO    IS A FLOATING POINT NUMBER (MAY BE FRACTIONAL) WHICH
      C                REPRESENTS THE VALUE OF IY IN VOFXY(IX,IY) WHICH IS
      C                TO CORRESPOND TO THE FIRST ROW OF THE PRINTED OUTPUT.
      C                DATA WITH INDEX IY LESS THAN FYLO IS NOT CONTOURED.
      C             FYLO MUST GRTHN= 1.0 .
      C
      C     FYHI    SIMILARLY IS THE VALUE OF IY CORRESPONDING TO THE LAST
      C                ROW OF PRINTED OUTPUT.  DATA WITH INDEX IY GREATER
      C                THAN FYHI IS NOT CONTOURED.
      C             MUST BE LESS THAN OR EQUAL TO LVY, AND MUST EXCEED FYLO.
      C
      C     NROWS   SPECIFIES THE NO. OF ROWS THE PRINTED OUTPUT IS TO TAKE.
      C             MUST BE GRTHN= 2 .
      C
      C     ARGLO   IS A FLOATING OR FIXED NUMBER (CORRESPONDING TO THE
      C                VARIABLE IY) TO BE PRINTED AT THE LEFTMOST END OF THE
      C                FIRST OUTPUT ROW AS A LABEL.  MODE IS DETERMINED
      C                BY ZFAFXD.
      C
      C     ARGDEL  IS FLOATING OR FIXED WITH ARGLO, AND IS THE INCREMENT
      C                BETWEEN SUCCESSIVE ROWS.
      C
      C     ZFAFXD  =0.0  IMPLIES ARGLO AND ARGDEL ARE FIXED.
      C             NOT =0.0 IMPLIES ARGLO AND ARGDEL ARE FLOATING.  FIXED
      C                LABELS ARE PRINTED IN FORMAT(I6), FLOATING LABELS IN
      C                FORMAT(E13.4).
      C
      C     CHLVLS(I) I=1,2,...,NCHRS  GIVES THE CHARACTERS USED FOR PLOTTING
      C                   THE CONTOUR LEVELS, EACH REGISTER IN FORMAT (A1).
      C                   THE CHARACTERS * AND $ SHOULD NOT APPEAR HERE SINCE
      C                   CNTROW USES THEM TO INDICATE CONFLICT OF CONTOURS.
      C                   THE RELATION OF CHLVLS TO THE ACTUAL LEVELS DEPENDS
      C                   ON THE MANNER IN WHICH THE LEVELS ARE DEFINED, AS
      C                   SPECIFIED BELOW.
      C
      C     NCHRS   IS THE NO. OF CHARACTERS IN THE CHLVLS VECTOR.  IT IS
      C                   ALSO THE NO. OF CONTOUR LEVELS IN THE CASE
      C                   DELEVL=0. AS DESCRIBED BELOW.
      C             MUST EXCEED ZERO.
      C
      C     DELEVL  INDICATES THE MANNER OF CONTOUR LEVEL SPECIFICATION.  IF
      C                   DELEVL IS NON-ZERO, SUCCESSIVE CONTOUR LEVELS ARE
      C                   ASSUMED TO BE SEPARATED BY DELEVL UNITS WITH A
      C                   STARTING VALUE OF VLEVL (DELEVL MAY NOT BE
      C                   NEGATIVE).  ON THE OTHER HAND IF DELEVL IS ZERO,
      C                   VLEVL IS INTERPRETED AS A VECTOR GIVING A FIXED SET
      C                   OF ARBITRARY LEVELS.
      C             MUST BE GRTHN= 0.0 .
      C
      C     VLEVL(I)  I=...  HAS INTERPRETATION DEPENDING ON DELEVL.
      C                   IF DELEVL IS NON-ZERO, VLEVL IS A SINGLE CONSTANT
```

```
C                   VLEVL(1), WHICH IS THE CONTOUR LEVEL OF VOFXY TO
C                   BE ASSOCIATED WITH THE CHARACTER CHLVLS(1).
C                   VLEVL+DELEVL IS TO BE ASSOCIATED WITH CHLVLS(2),
C                   ETC.  THIS ASSOCIATION IS TAKEN TO BE CYCLIC IF
C                   NECESSARY, I.E.,
C
C                   VLEVL+(NCHRS-1)*DELEVL HAS CHARACTER CHLVLS(NCHRS)
C                   VLEVL+NCHRS*DELEVL     HAS CHARACTER CHLVLS(1)
C                             ETC.
C                   ALSO
C                   VLEVL-DELEVL           HAS CHARACTER CHLVLS(NCHRS)
C                             ETC.  FOR LOWER LEVELS.
C
C                   IF DELEVL IS ZERO, VLEVL IS TAKEN AS A VECTOR,
C                   VLEVL(1...NCHRS), OF INDIVIDUAL CONTOUR LEVELS
C                   WHICH CORRESPOND 1 TO 1 WITH THE CHARACTERS OF
C                   CHLVLS(1...NCHRS).  THESE CONTOUR LEVELS MUST BE
C                   MONOTONELY INCREASING IN SIZE.
C
C    SPACE(I)  I=1...LSPACE  IS NEEDED FOR SCRATCH WHERE
C                   LSPACE = L + NCOLS + 3 + XMAXOF(4,4*NCOLS/L)
C                AND     L = (FXHI) ROUNDED UP - (FXLO) ROUNDED DOWN .
C
C
C OUTPUTS      NO OUTPUT, OR ONLY PARTIAL OUTPUT OCCURS IF IANS IS NEG.
C
C              THE PRINCIPAL OUTPUTS OCCUR OFF-LINE ON LOGICAL ITAPE
C              AND POSSIBLY ON-LINE ACCORDING TO ISENSE AND THE STATUS
C              OF THE SENSE SWITCHES.  SEE THE EXAMPLES BELOW FOR
C              ILLUSTRATIONS.  THE OUTPUT IS PRECEDED BY A PAGE RESTORE.
C
C    IANS      =    0 NORMALLY
C              = - 1 FOR ILLEGAL ITAPE
C              = - 2 FOR ILLEGAL LVX
C              = - 3 FOR ILLEGAL LVY
C              = - 4 FOR ILLEGAL LXDIM
C              = - 5 FOR ILLEGAL FXLO
C              = - 6 FOR ILLEGAL FXHI
C              = - 7 FOR ILLEGAL NCOLS
C              = - 8 FOR ILLEGAL FYLO
C              = - 9 FOR ILLEGAL FYHI
C              = -10 FOR ILLEGAL NROWS
C              =-105 FOR ILLEGAL NCHRS
C              =-106 FOR ILLEGAL DELEVL
C              =-107 FOR ILLEGAL VLEVL  (NOT MONOTONE IN CASE DELEVL=0.)
C                    (THE LAST THREE ILLEGALITIES BEING CAUGHT BY
C                     SUBROUTINE CNTROW)
C
C
C EXAMPLES
C              THE FIRST 6 EXAMPLES BELOW CONTOUR DATA REPRESENTING A
C              SIMPLE PLANE, WITH CONSTANT VALUES IN THE IY DIRECTION.
C              THEY WILL UTILIZE MATRICES DEFINED AS FOLLOWS.
C
C              DIMENSION  VOFXY(4,4),VXY22(4,2),VXY23(4,3),VXY24(4,4),
C             1           VXY32(4,2),VXY42(4,2),VXY33(4,3),SPACE(152)
C             VXY22(1...2,,IY) = 0.0,10.0                 FOR IY=1,2
C             VXY23(1...2,,IY) = 0.0,10.0                 FOR IY=1,2,3
C             VXY24(1...2,,IY) = 0.0,10.0                 FOR IY=1,2,3,4
C             VXY32(1...3,,IY) = 0.0,5.0,10.0             FOR IY=1,2
C             VXY42(1...4,,IY) = 0.0,3.333333,6.666667,10.0  FOR IY=1,2
C             VXY33(1...3,,IY) = 0.0,5.0,10.0             FOR IY=1,2,3
C             CHLVLS(1...20)   = 1H0,1H1,1H2,...,1H9,1HA,1HB,...,1HJ
C
C 1. INPUTS  - ITAPE=2  ISENSE=1  VOFXY=VXY22 DEFINED ABOVE
C             LVX=2  LVY=2  LXDIM=4  FXLO=1.0  FXHI=2.0
C             NCOLS=21  NCOLLO=0  FYLO=1.  FYHI=2.  NROWS=2
C             ARGLO=0.  ARGDEL=1.  ZFAFXD=1.  NCHRS=20
C             DELEVL=1.0  VLEVL=0.0
C    USAGE   -      CALL CONTUR(ITAPE,ISENSE,VOFXY,LVX,LVY,LXDIM,
C                  1            FXLO,FXHI,NCOLS,NCOLLO,FYLO,FYHI,
C                  2            NROWS,ARGLO,ARGDEL,ZFAFXD,CHLVLS,
C                  3            NCHRS,DELEVL,VLEVL,SPACE,IANS)
```

```
C     OUTPUTS - IANS=0 AND THE PRINTED OUTPUT WILL BE (ENDING IN COL. 34)
C
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C                 0.        0 1 2 3 4 5 6 7 8 9 A
C                 0.1000E 010 1 2 3 4 5 6 7 8 9 A
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C
C  2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT VOFXY=VXY23, LVX=2, FXHI=2.0,
C                   LVY=3, FYHI=3.0, NROWS=2
C     USAGE   - SAME AS EXAMPLE 1.
C     OUTPUTS - IDENTICAL TO EXAMPLE 1.
C
C  3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT VOFXY=VXY24, LVX=2, FXHI=2.0,
C                   LVY=4, FYHI=4.0, NROWS=3
C     USAGE   - SAME AS EXAMPLE 1.
C     OUTPUTS - IANS=0 AND THE PRINTED OUTPUT IS
C
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C                 0.        0 1 2 3 4 5 6 7 8 9 A
C                 0.1000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.2000E 010 1 2 3 4 5 6 7 8 9 A
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C
C  4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT VOFXY=VXY32, LVX=3, FXHI=3.0,
C                   LVY=2, FYHI=2.0, NROWS=4
C     USAGE   - SAME AS EXAMPLE 1.
C     OUTPUTS - IANS=0 AND THE PRINTED OUTPUT IS
C
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C                 0.        0 1 2 3 4 5 6 7 8 9 A
C                 0.1000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.2000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.3000E 010 1 2 3 4 5 6 7 8 9 A
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C
C  5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT VOFXY=VXY42, LVX=4, FXHI=4.0,
C                   LVY=2, FYHI=2.0, NROWS=5
C     USAGE   - SAME AS EXAMPLE 1.
C     OUTPUTS - IANS=0 AND THE PRINTED OUTPUT IS
C
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C                 0.        0 1 2 3 4 5 6 7 8 9 A
C                 0.1000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.2000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.3000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.4000E 010 1 2 3 4 5 6 7 8 9 A
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C
C  6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT VOFXY=VXY33, LVX=3, FXHI=3.0,
C                   LVY=3, FYHI=3.0, NROWS=6
C     USAGE   - SAME AS EXAMPLE 1.
C     OUTPUTS - IANS=0 AND THE PRINTED OUTPUT IS
C
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C                 0.        0 1 2 3 4 5 6 7 8 9 A
C                 0.1000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.2000E 010 1 2 3 4 5 6 7 8 9 A
```

```
C                 0.3000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.4000E 010 1 2 3 4 5 6 7 8 9 A
C                 0.5000E 010 1 2 3 4 5 6 7 8 9 A
C                           000000000000000000000
C                           000000000011111111112
C                           012345678901234567890
C
C 7. FOR EXAMPLES 7.,8., AND 9., WE SHALL CONTOUR A 1/R TYPE FUNCTION.
C
C    INPUTS  - SAME AS EX. 1. EXCEPT VOFXY SHOULD BE SET UP AS FOLLOWS.
C                    DIMENSION  VOFXY(25,25)
C                    DO 10  IX=1,25
C                    DO 10  IY=1,25
C                    R=SQRTF(FLOATF((IX-1)**2+(IY-1)**2))
C                    VOFXY(IX,IY)=20.0/(R+.5)
C              AND SET LVX=LVY=LXDIM=25, FXHI=FYHI=25.0, NCOLS=80,
C                    NROWS=48 (THIS RATIO OF NCOLS/NROWS MAPS CIRCLES
C                    INTO CIRCLES.)
C
C    USAGE   - SAME AS EXAMPLE 1.
C    OUTPUTS - IANS=0 AND THE FIRST 64 COLUMNS OF PRINTED OUTPUT ARE
C
C                    000000000000000000000000000000000000000000000000000
C                    000000000011111111112222222222333333333344444444445
C                    012345678901234567890123456789012345678901234567890
C          0.        $$$$$9 876 5   4    3         2
C          0.1000E 01$$$$*987 6 5   4    3         2
C          0.2000E 01DC BA987 6 5  4     3         2
C          0.3000E 019 9  87 6 5   4     3         2
C          0.4000E 01    7 6  5   4     3          2
C          0.5000E 01    6   5   4     3          2
C          0.6000E 01      5    4      3          2
C          0.7000E 01         4       3          2
C          0.8000E 01      4        3           2
C          0.9000E 01              3           2
C          0.1000E 02           3             2
C          0.1100E 02        3               2
C          0.1200E 02  3                    2
C          0.1300E 02                     2
C          0.1400E 02                    2
C          0.1500E 02                  2
C          0.1600E 02                2
C          0.1700E 02             2
C          0.1800E 02        2
C          0.1900E 02
C          0.2000E 02
C          0.2100E 02
C          0.2200E 02
C          0.2300E 02
C          0.2400E 02                                                 1
C          0.2500E 02                                                1
C          0.2600E 02                                              1
C          0.2700E 02                                            1
C          0.2800E 02                                           1
C          0.2900E 02                                         1
C          0.3000E 02                                       1
C          0.3100E 02                                    1
C          0.3200E 02                                  1
C          0.3300E 02                               1
C          0.3400E 02                            1
C          0.3500E 02                         1
C          0.3600E 02                    1
C          0.3700E 02                1
C          0.3800E 02      1
C          0.3900E 02
C          0.4000E 02
C          0.4100E 02
C          0.4200E 02
C          0.4300E 02
C          0.4400E 02
C          0.4500E 02
C          0.4600E 02
C          0.4700E 02
C                    000000000000000000000000000000000000000000000000000
C                    000000000011111111112222222222333333333344444444445
```

```
C                   012345678901234567890123456789012345678901234567890
C
C 8. THIS EXAMPLE IS THE SAME AS EXAMPLE 7. EXCEPT THAT WE USE ONLY
C              THE UPPER LEFT CORNER OF THE DATA.
C    INPUTS  - SAME AS EXAMPLE 7. EXCEPT FXHI=FYHI=10.
C    USAGE   - SAME AS EXAMPLE 1.
C    OUTPUTS - IANS=0 AND THE FIRST 64 COLUMNS OF PRINTED OUTPUT ARE
C
C                   000000000000000000000000000000000000000000000000000
C                   000000000011111111112222222222333333333344444444445
C                   012345678901234567890123456789012345678901234567890
C         0.        $$$$$$**$DCBA 9   8  7   6     5       4
C         0.1000E 01$$$*$***EDCBA 9  8  7    6     5       4
C         0.2000E 01**$1**G*DCB A 9  8  7   6     5        4
C         0.3000E 012*JI*FEDC B A 9 8   7   6     5        4
C         0.4000E 01IHGFE DC B A 9  8  7    6     5        4
C         0.5000E 01E  D C  B A  9  8  7   6      5        4
C         0.6000E 01    B    A  9  8  7    6     5        4
C         0.7000E 01   A      9   8  7    6     5         4
C         0.8000E 019     9     8   7    6      5        4
C         0.9000E 01         8     7    6      5         4
C         0.1000E 02   8        7     6       5         4
C         0.1100E 02         7       6       5         4
C         0.1200E 02   7           6       5          4              3
C         0.1300E 02            6         5          4               3
C         0.1400E 02        6            5          4               3
C         0.1500E 02                  5           4                3
C         0.1600E 02               5             4                3
C         0.1700E 02           5               4                 3
C         0.1800E 02    5                    4                  3
C         0.1900E 02                       4                   3
C         0.2000E 02                     4                   3
C         0.2100E 02                  4                     3
C         0.2200E 02              4                        3
C         0.2300E 02        4                            3
C         0.2400E 02                                   3
C         0.2500E 02                                 3
C         0.2600E 02                               3
C         0.2700E 02                             3
C         0.2800E 02                           3
C         0.2900E 02                        3
C         0.3000E 02                    3
C         0.3100E 02               3
C         0.3200E 02      3
C         0.3300E 02
C         0.3400E 02
C         0.3500E 02
C         0.3600E 02
C         0.3700E 02
C         0.3800E 02
C         0.3900E 02
C         0.4000E 02                                                2
C         0.4100E 02                                              2
C         0.4200E 02                                           2
C         0.4300E 02                                         2
C         0.4400E 02                                      2
C         0.4500E 02                                  2
C         0.4600E 02                              2
C         0.4700E 02                          2
C                   000000000000000000000000000000000000000000000000000
C                   000000000011111111112222222222333333333344444444445
C                   012345678901234567890123456789012345678901234567890
C
C 9. THIS EXAMPLE IS THE SAME AS EXAMPLE 7. EXCEPT THAT WE USE ONLY AN
C              INTERNAL SQUARE OF THE DATA.
C    INPUTS  - SAME AS EXAMPLE 7. EXCEPT FXLO=FYLO=5., FXHI=FYHI=10.0
C    USAGE   - SAME AS EXAMPLE 1.
C    OUTPUTS - IANS=0 AND THE FIRST 64 COLUMNS OF PRINTED OUTPUT ARE
C
C                   000000000000000000000000000000000000000000000000000
C                   000000000011111111112222222222333333333344444444445
C                   012345678901234567890123456789012345678901234567890
C         0.                                             2
C         0.1000E 01                                   2
C         0.2000E 01                                 2
```

* CONTUR *
****
(PAGE 7)

```
C          0.3000E 01                                2
C          0.4000E 01                              2
C          0.5000E 01                            2
C          0.6000E 01                          2
C          0.7000E 01                        2
C          0.8000E 01                     2
C          0.9000E 01                   2
C          0.1000E 02                2
C          0.1100E 02             2
C          0.1200E 02          2
C          0.1300E 02       2
C          0.1400E 02    2
C          0.1500E 02 2
C          0.1600E 02
C          0.1700E 02
C          0.1800E 02
C          0.1900E 02
C          0.2000E 02
C          0.2100E 02
C          0.2200E 02
C          0.2300E 02
C          0.2400E 02
C          0.2500E 02
C          0.2600E 02
C          0.2700E 02
C          0.2800E 02
C          0.2900E 02
C          0.3000E 02
C          0.3100E 02
C          0.3200E 02
C          0.3300E 02
C          0.3400E 02
C          0.3500E 02
C          0.3600E 02
C          0.3700E 02
C          0.3800E 02
C          0.3900E 02
C          0.4000E 02
C          0.4100E 02
C          0.4200E 02
C          0.4300E 02
C          0.4400E 02
C          0.4500E 02
C          0.4600E 02
C          0.4700E 02
C                    000000000000000000000000000000000000000000000000000
C                    000000000011111111112222222222333333333344444444445
C                    012345678901234567890123456789012345678901234567890
C
C
C PROGRAM FOLLOWS BELOW
C
C DUMMY DIMENSIONS
C
      DIMENSION    VOFXY(2),CHLVLS(2),VLEVL(2),SPACE(2)
C
C TRUE DIMENSIONS
C
      DIMENSION FMT(3)
      EQUIVALENCE  (ARG,IARG)
C
C BRING IN SOME OFTEN USED ARGUMENTS
C
      ITP=ITAPE
      LX=LVX
      LY=LVY
      LXDM=LXDIM
      FXL=FXLO
      FXH=FXHI
      NCLS=NCOLS
      NCLLO=XABSF(NCOLLO)
      FYL=FYLO
      FYH=FYHI
      NRWS=NROWS
C
```

```
C SET A FEW PRELIMINARY CONSTANTS
C
      IXLO=RNDDNF(FXL)
      IXHI=RNDUPF(FXH)
      IYLO=RNDDNF(FYL)
      IYHI=RNDUPF(FYH)
C
C CHECK ITAPE,LVX,LVY,LXDIM,FXLO,FXHI,NCOLS,FYLO,FYHI,NROWS.
C
      IANSR=-1
      IF (ITP)                9999,9999,5
 5    IF (ITP-20)             10,10,9999
 10   IANSR=-2
      IF (LX-1)               9999,9999,20
 20   IANSR=-3
      IF (LY-1)               9999,9999,30
 30   IANSR=-4
      IF (LXDM-LX)            9999,40,40
 40   IANSR=-5
      IF (IXLO)               9999,9999,50
 50   IANSR=-6
      IF (IXHI-IXLO)          9999,9999,55
 55   IF (IXHI-LX)            60,60,9999
 60   IANSR=-7
      IF (NCLS)               9999,9999,65
 65   IF (NCLS-125)           70,70,9999
 70   IANSR=-8
      IF (IYLO)               9999,9999,80
 80   IANSR=-9
      IF (IYHI-IYLO)          9999,9999,85
 85   IF (IYHI-LY)            90,90,9999
 90   IANSR=-10
      IF (NRWS )              9999,9999,100
C
C GIVE PAGE RESTORE AND THE COLUMN INDICATOR
C
 100  WRITE OUTPUT TAPE ITP,105
 105  FORMAT(1H1)
C
C SET UP THE REMAINING CONSTANTS
C
      IXLOR=XMAXOF(1,IXLO-1)
      IXHIR=XMINOF(IXHI+1,LX)
      LVEC=IXHIR-IXLOR+1
      IYLOR=XMAXOF(1,IYLO-1)
      IYHIR=XMINOF(IYHI+1,LY)
      DIFF=FLOATF(IXLOR-1)
      FXLO2=FXL-DIFF
      FXHI2=FXH-DIFF
      ISPVEC=1
      ISPPLT=ISPVEC+LVEC
      ISPPND=ISPPLT+NCLS-1
      ISPSPA=ISPPLT+NCLS
      C1FY=0.0
      IF (NRWS-1)       115,115,110
 110  C1FY=(FYH-FYL)/FLOATF(NRWS-1)
 115  CZFY=FYL-C1FY
      FMT(1)=6H(I6,13
      FMT(2)=4H0A1)
      ICOLLO=7
      IF (ZFAFXD)      120,130,120
 120  FMT(1)=6H(E13.4
      FMT(2)=6H,130A1
      FMT(3)=1H)
      ICOLLO=14
 130  ARG=ARGLO
      IXROW=1
      CALL COLABL(ITP,ICOLLO,NCLLO,NCLS,SPACE)
C
C FIND THE UNROUNDED INDEX, FY, CORRESPONDING TO THIS ROW IXROW
C      (SPECIAL TREATMENT FOR IXROW=1 AND =NRWS IS TO AVOID
C       ROUNDOFF UNCERTAINTIES)
C
 300  IF (IXROW-1)            305,305,310
 305  FY=FYL
```

****
* CONTUR *
****
(PAGE 9)

```
      GO TO 330
  310 IF (IXROW-NRWS)        320,315,315
  315 FY=FYH
      GO TO 330
  320 FY=CZFY+C1FY*FLOATF(IXROW)
C
C INTERPOLATE THE COLUMN WHOSE INDEX IS FY
C
  330 NY = IYHIR-IYLOR+1
      IXV = IXLOR+(IYLOR-1)*LXDM
      CALL ARBCOL(VOFXY(IXV),LVEC,NY,LXDM,FY,SPACE(ISPVEC))
C
C CONTOUR SPACE(ISPVEC,...)  INTO SPACE(ISPPLT) AND CHECK IANS.
C
  700 CALL CNTROW(SPACE(ISPVEC),LVEC,FXLO2,FXHI2,NCLS,CHLVLS,NCHRS,
     1            DELEVL,VLEVL,SPACE(ISPSPA),SPACE(ISPPLT),IANSR)
      IF (IANSR)                720,730,720
  720 IANSR=IANSR-100
      GO TO 9999
C
C PRINT THE OUTPUT OFF-LINE FIRST, THEN ON-LINE IF REQUESTED
C
  730 WRITE OUTPUT TAPE ITP,FMT,ARG,(SPACE(I),I=ISPPLT,ISPPND)
      IF (SWITCHF(ISENSE))   770,770,760
  760 PRINT FMT,ARG,(SPACE(I),I=ISPPLT,ISPPND)
C
C THEN INCREMENT ARG (=IARG), IXROW, AND CHECK FOR FINISH.
C
  770 IF (ZFAFXD)               780,790,780
  780 ARG=ARG+ARGDEL
      GO TO 795
  790 IARG=IARG+XSAMEF(ARGDEL)
  795 IXROW=IXROW+1
      IF (IXROW-NRWS)           300,300,800
C
C BEFORE RETURNING, REOUTPUT THE COLUMN LABELLING
C
  800 CALL COLABL(ITP,ICOLLO,NCLLO,NCLS,SPACE)
      IANSR=0
C
C EXIT
C
 9999 IANS=IANSR
      RETURN
      END
```

```
*     CONVLV (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0098
*     LABEL
CCONVLV
      SUBROUTINE CONVLV(LX,XX,LY,YY,CC)
C
C                    ----ABSTRACT----
C
C TITLE - CONVLV
C        COMPLETE CONVOLUTION OF TWO TRANSIENTS
C
C              CONVLV CONVOLVES TWO TRANSIENTS, X(I) I=0,1,...,LX-1
C              AND Y(I) I=0,1,...,LY-1 , TO PRODUCE THE COMPLETE
C              CONVOLUTION FUNCTION
C
C                             LX-1
C                     C(I)  =  SUM ( X(J)*Y(I-J) )
C                              J=0
C
C                     FOR I = 0,1,...,LX+LY-2
C                     WHERE
C                        LX AND LY ARE INPUT PARAMETERS
C                        Y(K) IS ASSUMED = 0.0 FOR K OUTSIDE OF
C                             THE RANGE 0 TO LY-1
C              NOTE THAT THE CONVOLUTION IS INDEPENDENT OF THE ORDER
C              OF THE INPUTS X AND Y.
C
C              TECHNIQUE USED IS AN ALGORITHM BASED ON ANALOGY TO
C              MULTIPLICATION OF POLYNOMIALS
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 96  REGISTERS
C SPEED     - ABOUT .49 * (LX*LY)  MILLISEC ON THE 709
C             ABOUT .082 * (LX*LY)  MILLISEC ON THE 7090
C AUTHOR    - J. CLAERBOUT
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUITNES - (NONE)
C        AND FORTRAN SYSTEM ROUTINES - (NONE)
C
C FORTRAN USAGE
C     CALL CONVLV(LX,XX,LY,YY,CC)
C
C INPUTS
C
C    LX        IS NO. OF TERMS IN X VECTOR
C              MUST EXCEED ZERO (PROGRAM EXITS IF ZERO OR LESS)
C
C    XX(I)     I=1,...,LX  CONTAINS  X(0),...,X(LX-1) RESPECTIVELY
C
C    LY        IS NO. OF TERMS IN Y VECTOR
C              MUST EXCEED ZERO  (PROGRAM EXITS IF ZERO OR LESS)
C
C    YY(I)     I=1...LY    CONTAINS  Y(0),...,Y(LY-1) RESPECTIVELY
C              EQUIVALENCE (XX,YY) IS PERMITTED
C
C OUTPUTS
C
C    CC(I)     I=1,...,LX+LY-1 CONTAINS C(0),...,C(LX+LY-2) RESPECTIVELY
C                        WHERE C(I) IS GIVEN IN ABSTRACT
C
C EXAMPLES
C
C 1. SHOWING REVERSIBILITY OF X AND Y
C    INPUTS  - LX = 3  XX(1...3) = 1.,2.,3.
C              LY = 2  YY(1...2) =  10.,1.
C
C USAGE   -          CALL CONVLV(LX,XX,LY,YY,CC1)
C                    CALL CONVLV(LY,YY,LX,XX,CC2)
C    OUTPUTS - CC1(1...4) = CC2(1...4) = 10.,21.,32.,3.
C
C 2. ILLEGAL INPUT CASES (NO OUTPUT)
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT START WITH OUTPUT VECTORS
C                CLEANED, I.E. CC1(1...4) = CC2(1...4) = 0.,0.,0.,0.
```

****
* CONVLV *
****
(PAGE 2)

```
C    USAGE   -        CALL CONVLV(-2,XX,LY,YY,CC1)
C                     CALL CONVLV(LX,XX,0,YY,CC2)
C    OUTPUTS - CC1(1...4) = 0.,0.,0.,0.    (ILLEGAL LX)
C              CC2(1...4) = 0.,0.,0.,0.    (ILLEGAL LY)
C
C PROGRAM FOLLOWS BELOW
C
C DUMMY DIMENSION STATEMENTS
      DIMENSION  XX(2),YY(2),CC(2)
C CHECK LEGALITIES
      IF (LX)  9999,9999,10
   10 IF (LY)  9999,9999,20
C CLEAR OUTPUT VECTOR
   20 LC=LX+LY-1
      DO 30  I=1,LC
   30 CC(I)=0.0
C CONVOLVE
      DO 40  I=1,LX
      DO 40  J=1,LY
      K=I+J
   40 CC(K-1)=CC(K-1)+XX(I)*YY(J)
C EXIT
 9999 RETURN
      END
```

```
*      CONVLV-II (SUBROUTINE)          10/2/64   LAST CARD IN DECK IS NO. 0148
*      FAP
*CONVLV-II
       COUNT   125
       LBL     CONVLV
       ENTRY   CONVLV (LX,XX,LY,YY,CC)
*
*                    ----ABSTRACT----
*
* TITLE - CONVLV-II
*       COMPLETE CONVOLUTION OF TWO TRANSIENTS
*
*             CONVLV-II CONVOLVES TWO TRANSIENTS X(I) I=0,1,...,LX-1
*             AND Y(I) I=0,1,...,LY-1 , TO PRODUCE THE COMPLETE
*             CONVOLUTION FUNCTION
*
*                          LX-1
*                    C(I)  =  SUM ( X(J)*Y(I-J) )
*                           J=0
*
*                    FOR I = 0,1,...,LX+LY-2
*                    WHERE
*                       LX AND LY ARE INPUT PARAMETERS
*                       Y(K) IS ASSUMED = 0.0 FOR K OUTSIDE OF
*                            THE RANGE 0 TO LY-1
*             NOTE THAT THE CONVOLUTION IS INDEPENDENT OF THE ORDER
*             OF THE INPUTS X AND Y.
*
*             CONVLV-II IS A FAP PROGRAM FUNCTIONALLY IDENTICAL TO THE
*             FORTRAN PROGRAM CONVLV BUT IS ABOUT 35 PERCENT FASTER.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 56 REGISTERS
* SPEED     - ABOUT .32 * (LX*LY) MILLISEC ON 709
*                   .051 * (LX*LY) MILLISEC ON 7090
* AUTHOR    - J. CLAERBOUT AND R. WIGGINS
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL CONVLV(LX,XX,LY,YY,CC)
*
* INPUTS
*
*    LX         IS NO. OF TERMS IN X VECTOR
*               FOR MAXIMUM SPEED THE X VECTOR SHOULD BE THE LONGEST
*               IF X AND Y HAVE DIFFERENT LENGTHS
*               MUST EXCEED ZERO  (PROGRAM EXITS IF ZERO OR LESS)
*
*    XX(I)      I=1,...,LX  CONTAINS  X(0),...,X(LX-1) RESPECTIVELY
*
*    LY         IS NO. OF TERMS IN Y VECTOR
*               MUST EXCEED ZERO  (PROGRAM EXITS IF ZERO OR LESS)
*
*    YY(I)      I=1...LY    CONTAINS  Y(0),...,Y(LY-1) RESPECTIVELY
*               EQUIVALENCE (XX,YY) IS PERMITTED
*
* OUTPUTS
*
*    CC(I)      I=1,...,LX+LY-1 CONTAINS C(0),...,C(LX+LY-2) RESPECTIVELY
*                          WHERE C(I) IS GIVEN IN ABSTRACT
*
* EXAMPLES
*
* 1. SHOWING REVERSIBILITY OF X AND Y
*    INPUTS  - LX = 3  XX(1...3) = 1.,2.,3.
*              LY = 2  YY(1...2) =  10.,1.
*
* USAGE   -        CALL CONVLV(LX,XX,LY,YY,CC1)
*                  CALL CONVLV(LY,YY,LX,XX,CC2)
```

CONVLV-II (PAGE 2)

```
*    OUTPUTS - CC1(1...4) = CC2(1...4) = 10.,21.,32.,3.
*
* 2. ILLEGAL INPUT CASES (NO OUTPUT)
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT START WITH OUTPUT VECTORS
*                 CLEANED, I.E. CC1(1...4) = CC2(1...4) = 0.,0.,0.,0.
*    USAGE   -        CALL CONVLV(-2,XX,LY,YY,CC1)
*                     CALL CONVLV(LX,XX,0,YY,CC2)
*    OUTPUTS - CC1(1...4) = 0.,0.,0.,0.   (ILLEGAL LX)
*              CC2(1...4) = 0.,0.,0.,0.   (ILLEGAL LY)
*
* PROGRAM FOLLOWS BELOW
*  IN PROGRAM NOTES BELOW
*      X=XX
*      Y=YY
*      C=CC
       HTR     0
       HTR     0
       HTR     0
       BCI     1,CONVLV
CONVLV SXD     CONVLV-4,1
       SXD     CONVLV-3,2
       SXD     CONVLV-2,4
       CLA*    1,4            GET AND SET UP
       ARS     18
       STA     LAD1           LENGTH OF X SERIES.
       TMI     LEAVE
       TZE     LEAVE
       CLA*    3,4            GET AND SET UP
       ARS     18             LENGTH OF
       STA     LAD2           Y SERIES.
       TMI     LEAVE
       TZE     LEAVE
       CLA     2,4            SET
       ADD     =1             UP
       STA     S2             ADDRESS.
       CLA     4,4            SET
       ADD     =1             UP
       STA     S1             ADDRESS.
       CLA     5,4            SET UP
       SUB     LAD2           ADDRESS
       ADD     =2             EQUAL TO
       STA     S3             LOC(C)-LY+2.
       STA     S4
       CLA     5,4
       ADD     =1
       STA     Z              =BES C
       CLA     LAD1
       ADD     LAD2
       SUB     =1
       PAX     ,1             LC=LX+LY-1 GOES TO XR1
*      FILL    ANSWER BLOCK WITH ZEROS
 Z     STZ     **,1
       TIX     *-1,1,1
       LXA     LAD2,2
*      OUTER   LOOP
 S     LXA     LAD1,1
*      CENTRAL LOOP
 S1    LDQ     **,2           **=BES Y
 S2    FMP     **,1           **=BES X
 S3    FAD     **,1           **=BES C -LY+1 (INITIALLY)
 S4    STO     **,1           **=BES C -LY+1 (INITIALLY)
       TIX     S1,1,1
*      END CENTRAL LOOP
       CLA     S3             RESET CENTRAL
 .     ADD     =1             LOOP FOR
       STA     S3             THE NEXT
       STA     S4             LAG.
       TIX     S,2,1
*      END OF OUTER LOOP
 LEAVE LXD     CONVLV-4,1
       LXD     CONVLV-3,2
       TRA     6,4
 LAD1  PZE                    LENGTH X SERIES IN ADDR.
 LAD2  PZE                    LENGTH Y SERIES IN ADDR.
       END
```

```
************************
*    COSISP            *
************************
 REFER TO
     COSP
```


```
************************
*    COSISP            *
************************
 REFER TO
     COSP
```

```
*     COSIS1  (SUBROUTINE)              9/10/64   LAST CARD IN DECK IS NO. 0263
*     LABEL
CCOSIS1
      SUBROUTINE COSIS1 (JOB,XX,LX,COSTAB,SINTAB,M,JMIN,JMAX,
     1                   CTR,STR,ADD,SPACE,IANS)
C
C
C                    ----ABSTRACT----
C
C TITLE - COSIS1
C        FAST COSINE AND/OR SINE TRANSFORMS OF ODD-LENGTH SERIES
C
C              COSIS1 PRODUCES A HIGH-SPEED COSINE AND/OR SINE TRANSFORM
C              (OR PORTION THEREOF) FROM AN ODD-LENGTH SERIES X(I),
C              I=-N,-N+1,...,N
C
C                                N
C                    CT(J)  =   SUM  ( X(I)*COS(I*J*PI/M) )
C                               I=-N
C                    AND/OR
C                                N
C                    ST(J)  =   SUM  ( X(I)*SIN(I*J*PI/M) )
C                               I=-N
C
C                          FOR J = JMIN,JMIN+1,...,JMAX
C                    WHERE
C                          PI = 3.14159265
C                          N,M,JMIN AND JMAX ARE INPUT PARAMETERS
C                          COS(J*PI/M) AND/OR SIN(J*PI/M) J=0,1,...,M
C                            ARE REQUIRED AS INPUT TABLES
C                          0 LSTHN= JMIN LSTHN JMAX LSTHN= M
C
C              SPEED IS ATTAINED BY
C                    1. SPLITTING THE X(I) SERIES INTO ODD AND EVEN
C                         PARTS AND, IF N=M, RESPLITTING THESE INTO
C                         THEIR ODD AND EVEN SUBPARTS.
C
C                    2. USING THE HIGH-SPEED LOOPING LOGIC OF SUBROUTINE
C                         COSISP TO PERFORM THE TRANSFORMATIONS OF THE
C                         SHORTENED SUBPARTS.
C
C              AN OPTION IS PROVIDED FOR ADDING CT(I) AND OR ST(I) TO
C              THE OUTPUT AREA RATHER THAN STORING THEM THERE.
C
C              2*N+4 TEMPORARY REGISTERS ARE REQUIRED BY COSIS1 UNLESS
C              THE USER IS WILLING TO SACRIFICE X(I) (IN WHICH 3 EXTRA
C              REGISTERS BEHIND X(I) ARE NEEDED).
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 406  REGISTERS
C SPEED      - FOR  M=N  COSIS1 TAKES ABOUT
C                 20*M*(JMAX-JMIN+1) MACHINE CYCLES (ON THE 7090)
C                   TO PRODUCE EITHER THE SINE OR COSINE TRANSFORM,
C              TWICE THAT TIME FOR BOTH.
C              FOR  M NOT= N  SUBSTITUTE  2*N  FOR  M  IN ABOVE FORMULA.
C AUTHOR     - R.A. WIGGINS, JUNE, 1963     GEOSCIENCE, INC.
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CHPRTS,COSISP,COSP,IXCARG,MOVREV,
C                                     SISP,SPLIT
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL COSIS1(JOB,XX,LX,COSTAB,SINTAB,M,JMIN,JMAX,CTR,STR,ADD,
C    1           SPACE,IANS)
C
C
C INPUTS
C
C    JOB       INDICATES WHETHER USER DESIRES THE COSINE TRANSFORM, THE
C                SINE TRANSFORM, OR BOTH.
C              =1 INDICATES COSINE TRANSFORM ONLY.
```

```
C               =2 INDICATES SINE TRANSFORM ONLY.
C               =3 INDICATES COSINE AND SINE TRANSFORMS.
C
C    XX(I)      I=1,...,LX CONTAINS THE SERIES X(J), J=-N,-N+1,...,N  AS
C                 DESCRIBED IN THE ABSTRACT.
C
C    LX         =2*N+1
C               MUST EXCEED ZERO.
C               MUST BE ODD.
C
C    COSTAB(I) I=1,...,M+1 CONTAINS COS(J*PI/M) J=0,1,...,M
C               IS DUMMY ARGUMENT IF JOB=2
C
C    SINTAB(I) I=1,...,M+1 CONTAINS SIN(J*PI/M) J=0,1,...,M
C               IS DUMMY ARGUMENT IF JOB=1
C
C    M          MUST EXCEED 0
C               IF = N COSIS1 BECOMES MUCH MORE EFFICIENT - SEE SPEED.
C
C    JMIN       MUST BE NON-NEGATIVE.
C
C    JMAX       MUST BE GRTHN JMIN, LSTHN= M
C
C    ADD        =0. IMPLIES THAT OUTPUTS ARE TO BE STORED IN THE OUTPUT
C                 AREA WITHOUT ADDITION.
C               NOT= 0. IMPLIES THAT THE OUTPUTS ARE TO ADDED INTO THE
C                 OUTPUT AREA.
C
C    SPACE(I)   I=1,...,LX+3 IS A BLOCK OF TEMPORARIES NEEDED BY COSIS1.
C               MAY BE EQUIVALENT TO XX(I) (XX(I) WILL BE DESTROYED).
C                 NOTE THAT 3 ADDITIONAL SPACES ARE NEEDED IN ADDITION TO
C                 THE LENGTH OF XX(I).
C
C
C OUTPUTS
C
C    CTR(I)     I=1,...,JMAX-JMIN+1 CONTAINS CT(J) J=JMIN,...,JMAX AS
C                 DEFINED IN THE ABSTRACT.
C               IS DUMMY ARGUMENT IF JOB=2
C
C    STR(I)     I=1,...,JMAX-JMIN+1 CONTAINS ST(J) J=JMIN,...,JMAX AS
C                 DEFINED IN THE ABSTRACT.
C               IS DUMMY ARGUMENT IF JOB=1
C
C    IANS       =0  NORMALLY
C               =1  IF JOB ILLEGAL (NOT = 1,2, OR 3)
C               =3  IF LX ILLEGAL (LSTHN= 0, OR ODD)
C               =6  IF M ILLEGAL (LSTHN= 0)
C               =7  IF JMIN ILLEGAL (LSTHN 0)
C               =8  IF JMAX ILLEGAL (LSTHN= JMIN, OR GRTHN M)
C
C
C EXAMPLES
C
C 1. COMPLETE SPECTRUM, NOT TRYING TO SAVE SPACE, 2*M+1 NOT = LX
C    INPUTS  - LX=7  XX(1...7) = -36.,-27.,-18.,2.,22.,33.,44.
C               M=2  COSTAB(1...3) = 1.,0.,-1.  SINTAB(1...3) = 0.,1.,0.
C               JOB=3  JMIN=0  JMAX=M  ADD=0.
C    USAGE   -        CALL COSIS1 (JOB,XX,LX,COSTAB,SINTAB,M,JMIN,JMAX,
C                  1               CTR,STR,ADD,SPACE,IANS)
C    OUTPUTS - IANS=0
C               CTR(1...3) = 20.,-4.,-4.  STR(1...3) = 0.,-40.,0.
C
C 2. COMPLETE SPECTRUM SAVING SPACE, 2*M+1 NOT= LX
C    INPUTS  - SAME AS EXAMPLE 1.
C    USAGE   -        CALL COSIS1 (JOB,XX,LX,COSTAB,SINTAB,M,JMIN,JMAX,
C                  1               CTR,STR,ADD,XX,IANS)
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT XX(1...10) ARE DESTROYED.
C
C 3. COMPLETE COSINE SPECTRUM, NOT TRYING TO SAVE SPACE, 2*M+1=LX
C    INPUTS  - LX=5  XX(1...5) = -17.,-5.,9.,7.,18.
C               M=2  COSTAB(1...3) = 1.,0.,-1.
C               JOB=1  JMIN=0  JMAX=M  ADD=0.
C    USAGE   -        CALL COSIS1 (JOB,XX,LX,COSTAB,DUMMY,M,JMIN,JMAX,
C                  1               CTR,DUMMY,ADD,SPACE,IANS)
```

```
C     OUTPUTS - IANS=0
C               CTR(1...3) = 12.,8.,8.
C
C 4. COMPLETE SINE SPECTRUM, NOT TRYING TO SAVE SPACE, 2*M+1=LX, AND
C       ADDING OUTPUT INTO OUTPUT AREA.
C     INPUTS  - LX=5  XX(1...5) = -17.,-5.,9.,7.,18.
C               M=2  SINTAB(1...3) = 0., 1., 0.  STR(1...3) = 1., 1., 1.
C               JOB=2  JMIN=0  JMAX=M  ADD=1.
C     USAGE   -        CALL COSIS1 (JOB,XX,LX,DUMMY,SINTAB,M,JMIN,JMAX,
C                     1             DUMMY,STR,ADD,SPACE,IANS)
C     OUTPUTS - IANS=0
C               STR(1...3) = 1., 13., 1.
C
C 5. ERROR EXITS WITH NO COMPUTATION
C     USAGE   -        CALL COSIS1 (0,XX,3,COSTAB,SINTAB,3,0,3,
C                     1             CTR,STR,ADD,SPACE,IANS1)
C                      CALL COSIS1 (3,XX,2,COSTAB,SINTAB,3,0,3,
C                     1             CTR,STR,ADD,SPACE,IANS2)
C                      CALL COSIS1 (3,XX,3,COSTAB,SINTAB,0,0,3,
C                     1             CTR,STR,ADD,SPACE,IANS3)
C                      CALL COSIS1 (3,XX,3,COSTAB,SINTAB,3,-1,3,
C                     1             CTR,STR,ADD,SPACE,IANS4)
C                      CALL COSIS1 (3,XX,3,COSTAB,SINTAB,3,5,4,
C                     1             CTR,STR,ADD,SPACE,IANS5)
C     OUTPUTS - IANS1=1 (ILLEGAL JOB)
C               IANS2=3 (ILLEGAL LX)
C               IANS3=6 (ILLEGAL M)
C               IANS4=7 (ILLEGAL JMIN)
C               IANS5=8 (ILLEGAL JMAX AND JMIN)
C
C
C     PROGRAM FOLLOWS
C
      DIMENSION XX(2),CM(2)
      COMMON CM
      J=JOB
      L=LX
      JMN=JMIN
      JMX=JMAX
C CHECK LEGALITIES OF INPUT PARAMETERS
      IAN=0
      IF (J*(4-J)) 10,10,20
 10   IAN=1
      GO TO 999
 20   IF (L *XMODF(L,2))  30,30,40
 30   IAN=3
      GO TO 999
 40   IF (M)  50,50,60
 50   IAN=6
      GO TO 999
 60   IF (JMN)  70,80,80
 70   IAN=7
      GO TO 999
 80   IF (JMX-JMN)  100,100,90
 90   IF (M-JMX)  100,110,110
 100  IAN=8
      GO TO 999
 110  CONTINUE
      M1=M
      IF (ADD) 120,130,120
 120  M1=-M1
 130  CONTINUE
      N=L/2
      LS=N+1
      CALL IXCARG (SPACE,ISS)
      ISA=ISS+LS+1
C SPLIT XX ONCE ONTO SPACE
      CALL SPLIT (XX,L,1.,CM(ISS),CM(ISA-1))
      CALL MOVREV(N,1,CM(ISA-1),1,CM(ISA+1),1)
      CM(ISA-1)=0.
      CM(ISA)=0.
      ISL=ISS+L+2
      CM(ISL)=0.
C CHECK IF FURTHER SPLITTING IS VALID
      IF (M-N)  300,200,300
```

COSIS1
(PAGE 4)

```
C YES IT IS VALID
  200  CONTINUE
       LSS=N/2
       GO TO (210,230,210),J
C SPLIT AND REVERSE SYMMETRICAL PART
  210  CONTINUE
       IAS=ISS+LSS+1
       CALL SPLIT (CM(ISS),LS,1.,CM(ISS),CM(IAS))
       CALL CHPRTS(CM(ISS),CM(IAS),LS)
       GO TO (220,10,230),J
C ONLY COSINE TRANSFORM WANTED - CALL COSP.
  220  CONTINUE
       CALL COSP (CM(ISS),CM(IAS),LSS,COSTAB,M1,JMN,JMX,1.,CTR)
       GO TO 999
C SPLIT AND REVERSE ANTISYMMETRICAL PART
  230  CONTINUE
       IAA=ISA+LSS+1
       CALL SPLIT (CM(ISA),LS,1.,CM(ISA),CM(IAA))
       CALL CHPRTS(CM(ISA),CM(IAA),LS)
       GO TO (10,240,250),J
C ONLY SINE TRANSFORM WANTED - USE SISP
  240  CONTINUE
       CALL SISP (CM(ISA),CM(IAA),LSS,SINTAB,M1,JMN,JMX,1.,STR)
       GO TO 999
C BOTH COSINE AND SINE TRANSFORMS WANTED - USE COSISP
  250  CONTINUE
       CALL COSISP (CM(ISS),CM(IAS),CM(ISA),CM(IAA),LSS,COSTAB,SINTAB,
      1             M1,JMN,JMX,1.,CTR,STR)
       GO TO 999
C FURTHER SPLITTING IS NOT VALID
  300  CONTINUE
       IAS=ISS
       IAA=ISA
       LSS=N
       GO TO (220,240,250),J
C THAT'S ALL.
  999  IANS=IAN
       RETURN
       END
```

```
*       COSP (SUBROUTINE)                  9/29/64   LAST CARD IN DECK IS NO. 0877
*       FAP
*COSP
        COUNT   1000
        LBL     COSP
        ENTRY   COSP (SSX,ASX,L,COSTAB,M,JMIN,JMAX,TYPE,COSTR)
        ENTRY   SISP (SAX,AAX,L,SINTAB,M,JMIN,JMAX,TYPE,SINTR)
        ENTRY   COSISP (SSX,ASX,SAX,AAX,L,COSTAB,SINTAB,M,JMIN,JMAX,TYPE,
*                       COSTR,SINTR)
*
*                    ----ABSTRACT----
*
* TITLE - COSP  WITH SECONDARY ENTRY POINTS SISP AND COSISP
*       FAST COSINE AND/OR SINE TRANSFORMS FROM 2 OR 4 EVEN-ODD PARTS
*
*             COSP COMPUTES COSINE SUMS, CT(J) J=JMIN,...,JMAX , ON
*             TWO INPUT SERIES, SS(I) AND AS(I) I=0,1,...,L , ACCORDING
*             TO              L
*                            SUM ( SS(I)*COS(I*J*(PI/M)) )     J EVEN
*                            I=0
*                CT(J) =
*                             L
*                            SUM ( AS(I)*COS(I*J*(PI/M)) )     J ODD
*                            I=0
*
*                  FOR J =JMIN,JMIN+1,...,JMAX
*               WHERE
*                     PI = 3.14159265
*                     M = INPUT PARAMETER
*                     COS(I*(PI/M))  I=0,1,...,M  IS AN INPUT TABLE
*                     SS(I),AS(I), MAY BE EITHER FIXED OR FLOATING POINT
*                         (THE COSINE TABLE MUST CORRESPOND IN TYPE)
*                     0 LSTHN= JMIN LSTHN JMAX LSTHN= M
*
*             SISP COMPUTES SINE SUMS, ST(J)
*                             L
*                            SUM ( AA(I)*SIN(I*J*(PI/M)) )      J EVEN
*                            I=0
*                ST(J) =
*                             L
*                            SUM ( SA(I)*SIN(I*J*(PI/M)) )      J ODD
*                            I=0
*
*                  FOR J = JMIN,JMIN+1,...,JMAX
*               WHERE
*                     SIN(I*(PI/M))  I=0,1,...,M  IS AN INPUT TABLE
*                     AA,SA, AND THE SINE TABLE ARE FIXED OR FLOATING
*
*             COSISP COMPUTES BOTH CT(J) AND ST(J) AS DEFINED ABOVE
*
*             AN OPTION IS PROVIDED FOR ADDING THE TRANSFORMS INTO THE
*             OUTPUT AREAS.
*
*             NOTE THAT THE FUNDAMENTAL FREQUENCY AS DEFINED BY THE
*             INPUT TABLES HAS PERIOD = EVEN NO. OF POINTS = 2M
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 504 REGISTERS
* SPEED     -         709-FIXED PT    709-FLOATING PT
*             COSP    34*K*(L+1)      37*K*(L+1)          MACHINE CYCLES
*             SISP    39*K*(L+1)      43*K*(L+1)          MACHINE CYCLES
*             COSISP  67*K*(L+1)      72*K*(L+1)          MACHINE CYCLES
*                WHERE  K = JMAX-JMIN+1
*                  (REDUCE ESTIMATES ABOUT 10 PERCENT FOR 7090)
* AUTHOR    - S.M. SIMPSON,  OCT 26, 61
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE OF COSP
*     CALL COSP  (SSX,ASX,L,COSTAB,M,JMIN,JMAX,TYPE,COSTR)
*
```

```
* INPUTS TO COSP
*
*    SSX(I)     I=1...L+1  CONTAINS SS(J) J=0,1,...,L  FIXED OR FLOATING
*
*    ASX(I)     I=1...L+1  CONTAINS AS(J) J=0,1,...,L  FIXED OR FLOATING
*               EQUIVALENCE (SSX,ASX) IS PERMITTED
*
*    L          MUST EXCEED 0
*
*    COSTAB(I) I=1...M+1  CONTAINS COS(J*PI/M)  J= 0,1,...,M
*               COSTAB IS FIXED OR FLOATING
*                  FOR FIXED POINT IT IS ASSUMED THAT THE BINARY POINT
*                  IS BETWEEN THE SIGN BIT AND BIT 1 SO THAT VALUES
*                  1.0 AND -1.0 SHOULD BE ENTERED AS OCTAL 377777777777
*                  AND 777777777777 RESPECTIVELY. THE BINARY POINT OF
*                  SSX AND ASX IS IMMATERIAL, BUT OVERFLOW MAY ARISE.
*
*    M          IS LENGTH OF COSTAB.
*               IF NEGATIVE, COSP ADDS TRANSFORM CT(I) TO THE OUTPUT
*                 BEFORE STORING IN THE OUTPUT AREA.
*               MUST NOT =0
*
*    JMIN       DEFINES  LOWEST MULTIPLE OF FUNDAMENTAL DESIRED
*               MUST BE GRTHN= 0 AND LSTHN JMAX
*
*    JMAX       DEFINES  HIGHEST MULTIPLE OF FUNDAMENTAL DESIRED
*               MUST BE GRTHN JMIN AND LSTHN= M
*
*    TYPE       = 0.0  SIGNIFIES  SS,AS, AND COSTAB ARE FIXED PT.
*               NOT= 0.0  MEANS   SS,AS, AND COSTAB ARE FLTG. PT.
*
* OUTPUTS FROM COSP
*
*    COSTR(I)   I=1...JMAX-JMIN+1 CONTAINS CT(J) J=JMIN...JMAX  AS
*                 DEFINED IN ABSTRACT.
*
*               (PROGRAM EXITS WITHOUT COMPUTATION IF L,M,JMIN,
*                 OR JMAX ILLEGAL)
*
* FORTRAN USAGE OF SISP
*     CALL SISP  (SAX,AAX,L,SINTAB,M,JMIN,JMAX,TYPE,SINTR)
*
* INPUTS TO SISP
*
*    SAX(I)     I=1...L+1  CONTAINS  SA(J) J=0,1,...,L
*
*    AAX(I)     I=1...L+1  CONTAINS  AA(J) J=0,1,...,L
*               EQUIVALENCE (SAX,AAX) IS PERMITTED.
*
*    L          SAME MEANING AS FOR COSP
*
*    SINTAB(I) I=1...M+1 CONTAINS SIN(J*PI/M)  J=0,1,...,M
*
*    M          SAME MEANING AS FOR COSP
*
*    JMIN       SAME MEANING AS FOR COSP
*
*    JMAX       SAME MEANING AS FOR COSP
*
*    TYPE       SAME MEANING AS FOR COSP
*
* OUTPUTS FROM SISP
*
*    SINTR(I)   I=1...JMAX-JMIN+1 CONTAINS ST(J) J=JMIN...JMAX AS
*                 DEFINED IN ABSTRACT
*
* FORTRAN USAGE OF COSISP
*     CALL COSISP(SSX,ASX,SAX,AAX,L,COSTAB,SINTAB,M,JMIN,JMAX,
*    1              TYPE,COSTR,SINTR)
*
*               WHERE ARGUMENTS ARE THE SAME AS FOR COSP AND SISP
*               EQUIVALENCE (SSX,ASX,SAX,AAX) IS PERMITTED.
*
* EXAMPLES
*
```

```
* 1. USE OF COSP, SISP, COSISP  WHEN ALL INPUTS EQUATED, FIXED AND
*              FLOATING, ALL FREQUENCIES
*    INPUTS  - X(1...4) = 1.,2.,3.,4.   IX(1...4) = 100,200,300,400  L=3
*              COSTAB(1...3)=1.0,0.0,-1.0  SINTAB(1...3)=0.0,1.0,0.0 M=2
*              ICOSTB(1...3)=OCT377777777777,000000000000,777777777777
*              ISINTB(1...3)=OCT000000000000,377777777777,000000000000
*              JMIN = 0   JMAX = 2
*    USAGE   -        CALL COSP (X,X,L,COSTAB,M,JMIN,JMAX,1.,C1)
*                     CALL COSP (IX,IX,L,ICOSTB,M,JMIN,JMAX,0.,IC1)
*                     CALL SISP (X,X,L,SINTAB,M,JMIN,JMAX,1.,S1)
*                     CALL SISP (IX,IX,L,ISINTB,M,JMIN,JMAX,0.,IS1)
*                     CALL COSISP (X,X,X,X,L,COSTAB,SINTAB,M,JMIN,JMAX,
*                                1.,C2,S2)
*                     CALL COSISP (IX,IX,IX,IX,L,ICOSTB,ISINTB,M,JMIN,
*                                JMAX,0.,IC2,IS2)
*    OUTPUTS - C1(1...3) = C2(1...3) = 10.,-2.,-2.
*              S1(1...3) = S2(1...3) = 0.,-2.,0.
*              IC1(1...3) = IC2(1...3) = 1000,-200,-200
*              IS1(1...3) = IS2(1...3) = 0,-200,0
*
* 2. PARTIAL FREQUENCY COVERAGE
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT JMIN = 1
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - C1(1...2) = C2(1...2) = -2.,-2.
*              S1(1...2) = S2(1...2) = -2.,0.
*              IC1(1...2) = IC2(1...2) = -200,-200
*              IS1(1...2) = IS2(1...2) = -200,0
*
* 3. USE OF COSISP TO FIND COEFFICIENTS OF TRIGONOMETRICAL SERIES FOR
*              AN EVEN-LENGTH VECTOR
*       (SEE CARSLAW, 1930, FOURIER SERIES AND INTEGRALS, P324,325)
*              GIVEN XX(I)  I=1...2*M  CONTAINING X(J) J=0,1,...,2*M-1
*              FIND A(0),A(1),...A(M) AND B(1),B(2),...,B(M-1) SUCH THAT
*
*              X(J)=A(0)+A(1)COS(J*D)+...+A(M-1)COS((J-1)*D)+A(M)COS(PI)
*                      +B(1)SIN(J*D)+...+B(M-1)SIN((J-1)*D)
*                   WHERE D=PI/M   J=0,1,...,2*M-1
*              SOLUTION
*    INPUTS  - COSTAB(1...M+1) = COS(J*PI/M)  J = 0,1,...,M
*              SINTAB(1...M+1) = SIN(J*PI/M)  J = 0,1,...,M
*              L = 2*M-1
*    USAGE   -        CALL COSISP(X,X,X,X,L,COSTAB,SINTAB,M,0,M,1.,AA,BB)
*                     AA(1) = AA(1)/FLOATF(2*M)
*                     AA(M+1) = AA(M+1)/FLOATF(2*M)
*                     DO 10  I=2,M
*                     AA(I)=AA(I)/FLOATF(M)
*                  10 BB(I)=BB(I)/FLOATF(M)
*    OUTPUTS - AA(1...M+1) WILL CONTAIN  A(0),A(1),...A(M)  AS REQUIRED
*              BB(2...M) WILL CONTAIN  B(1),...B(M-1)  AS REQUIRED
*              (BB(1)=BB(M+1)=0.)
*
* 4. USE OF COSISP TO INVERT COEFFICIENTS OF TRIG SERIES FOR AN EVEN-
*              LENGTH VECTOR
*              GIVEN A(0),...A(M)  B(1)...B(M-1)  AS DEFINED ABOVE
*              FIND X(J) = TRIG SERIES ABOVE   J = 0,1,...,2*M-1
*              SOLUTION
*    INPUTS  - AA(I) AND BB(I) ARE SAME AS OUTPUTS OF EXAMPLE 3.
*    USAGE   -        CALL COSISP(AA,AA,BB,BB,M,COSTAB,SINTAB,
*                   1           M,0,M,1.,XS,XA)
*                     I2M=2*M
*                     DO 20 I=2,M
*                     J=I2M+2-I
*                     XS(J)=XS(I)
*              20     XA(J)=-XA(I)
*                     DO 30 I=1,I2M
*              30     XBAC(I)=XA(I)+XS(I)
*    OUTPUTS - XBAC(1...2*M) WILL CONTAIN  X(0,1,...,2*M-1)  AS REQUIRED
*
* 5. ILLUSTRATION OF FINDING TRIG SERIES
*    INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   - SAME AS EXAMPLE 3.
*    OUTPUTS - AA(1...3) = 2.5,-1.,-.5
*              BB(1...3) = 0.,-1.,0.
*
* 6. ILLUSTRATION OF INVERTING TRIG SERIES
```

```
*    INPUTS  - SAME AS EXAMPLE 5. WITH AA,BB, SAME AS OUTPUTS FROM EX 5.
*    USAGE   - SAME AS EXAMPLE 4.
*    OUTPUTS - XBAC(1...4) = 1.,2.,3.,4.
*
* 7. USE OF SYMMETRIES TO REDUCE TIME IN COMPUTING TRANSFORMS ABOUT
*              MIDPOINT OF AN ODD-LENGTH SERIES
*              GENERAL FORM
*                            I=M
*                     C(J) = SUM ( X(I)*COS(I*J*PI/M) )
*                            I=-M
*                AND
*                            I=M
*                     S(J) = SUM ( X(I)*SIN(I*J*PI/M) )
*                            I=-M
*                        J = JMIN...JMAX
*              SUPPOSE  X(-6...6)=1.,3.,1.,2.,1.,1.,5.,4.,3.,3.,5.,4.,1.
*              FIRST SPLIT X ABOUT ITS MIDPOINT INTO ITS SYMMETRIC AND
*                ANTISYMMETRIC PARTS
*                    SX(1...7) = 5.,5.,4.,5.,6.,7.,2.
*                    AX(1...7) = 0.,3.,2.,1.,4.,1.,0.
*              THEN SPLIT EACH OF THESE ABOUT THEIR MIDPOINTS
*                SSX(1...4) = 5.,10.,12.,7.  ASX(1...4) = 0.,2.,2.,-3.
*                SAX(1...4) = 1.,6.,4.,0.  AAX(1...4) = 0.,2.,-2.,0.
*    INPUTS  - THEN REVERSE ALL THE VECTORS AND CHANGE SIGNS OF ASX
*              AAX TO GIVE
*                SSX(1...4) = 7.,12.,10.,5.  ASX(1...4) = 3.,-2.,-2.,0.
*                SAX(1...4) = 0.,4.,6.,1.  AAX(1...4) = 0.,2.,-2.,0.
*                L=3  M=6  COSTAB(1...7)=COS(J*PI/6)
*                          SINTAB(1...7)=SIN(J*PI/6)  J = 0...6
*    USAGE   -        CALL COSISP (SSX,ASX,SAX,AAX,3,COSTAB,SINTAB,M,0,M,
*                                  1.,COSTR,SINTR)
*    OUTPUTS - COSTR(1...7) = C(0...6) = 34.,.26795,3.,5.,1.,3.73205,0.
*              SINTR(1...7) = S(0...6) = 0.,8.19615,0.,3.,3.46410,
*                                        -2.19615,0.
*
* 8. ADDITION OF OUTPUTS TO VALUES ALREADY IN THE OUTPUT AREA
*
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT M=-2
*              C1(1...3) = C2(1...3) = 1.,2.,3.
*              S1(1...3) = S2(1...3) = 1.,-1.,-2.
*              IC1(1...3)= IC2(1...3)= 100,200,300
*              IS1(1...3)= IS2(1...3)= 100,-100,-200
*    USAGE   - SAME AS EXAMPLE 1.
*    CUTPUTS - C1(1...3) = C2(1...3) = 11.,0.,1.
*              S1(1...3) = S2(1...3) = 1.,-3.,-2.
*              IC1(1...3)= IC2(1...3)= 1100,0,100
*              IS1(1...3)= IS2(1...3)= 100,-300,-200
*
* PROGRAM FOLLOWS BELOW
*      NOTATION DIFFERENCES IN PROGRAM NOTES ARE
*      RSS=SSX       RAS=ASX       RAA=AAX       RSA=SAX
*      P=L
*
*
       HTR     0
       BCI     1,COSP
 COSP  SXD     *-2,4          SET UP EXIT
       SXA     LV+1,1
       SXA     LV+2,2
       CLA     K10
       STA     EXIT
*SET ARGUMENT TABLE
       CLA     1,4
       STA     T1
       CLA     2,4
       STA     T2
       CLA*    3,4
       STD     T5
       CLA     4,4
       STA     T6
       CLA*    5,4
       STO     T8
       CLA*    6,4
       STD     T9
       CLA*    7,4
```

****
* COSP *
****
(PAGE 5)

```
         STD     T10
         CLA*    8,4
         STO     T11
         CLA     9,4
         STA     T12
 *SET COSP SWITCHES
         CLA     KA18        KA6
         STA     Z30
         CLA     KA6         Z90
         STA     Z33
         CLA     KA15        Z107
         STA     Z106
         CLA     KA19        Z130
         STA     Z109B
         CLA     KT1         TRA  Z104
         STO     Z114
         STO     Z112
         CLA     KT2         TRA  Z102
         STO     Z121A
         STO     Z122A
         TRA     Z14
 *SET EXIT
  SISP   SXD     COSP-2,4
         SXA     LV+1,1
         SXA     LV+2,2
         CLA     K10
         STA     EXIT
 *SET ARGUMENT TABLE
         CLA     1,4
         STA     T3
         CLA     2,4
         STA     T4
         CLA*    3,4
         STD     T5
         CLA     4,4
         STA     T7
         CLA*    5,4
         STO     T8
         CLA*    6,4
         STD     T9
         CLA*    7,4
         STD     T10
         CLA*    8,4
         STO     T11
         CLA     9,4
         STA     T13
 *SET SISP SWITCHES
         CLA     KA14        KA9
         STA     Z30
         CLA     KA9         Z50
         STA     Z33
         CLA     KA7         Z100
         STA     Z56
         STA     Z66
         STA     Z76
         STA     Z86
         CLA     KA16        Z115
         STA     Z106
         CLA     KZ1         ZET  SWE
         STO     Z114
         STO     Z112
         CLA     KZ2         ZET  SWO
         STO     Z121A
         STO     Z122A
         TRA     Z14
 *SET EXIT
 COSISP SXD     COSP-2,4    SET UP EXIT
         SXA     LV+1,1
         SXA     LV+2,2
         CLA     K14
         STA     EXIT
 *SET UP ARGUMENT TABLE
         CLA     1,4
         STA     T1
         CLA     2,4
```

```
          STA     T2
          CLA     3,4
          STA     T3
          CLA     4,4
          STA     T4
          CLA*    5,4
          STD     T5
          CLA     6,4
          STA     T6
          CLA     7,4
          STA     T7
          CLA*    8,4
          STO     T8
          CLA*    9,4
          STD     T9
          CLA*    10,4
          STD     T10
          CLA*    11,4
          STO     T11
          CLA     12,4
          STA     T12
          CLA     13,4
          STA     T13
     *SET COSISP SWITCHES
          CLA     KA14          KA9
          STA     Z30
          CLA     KA9           Z50
          STA     Z33
          CLA     KA6           Z90
          STA     Z56
          STA     Z66
          STA     Z76
          STA     Z86
          CLA     KA15          Z107
          STA     Z106
          CLA     KZ1           ZET  SWE
          STO     Z114
          STO     Z112
          CLA     KZ2           ZET  SWO
          STO     Z121A
          STO     Z122A
          CLA     KA16          Z115
          STA     Z109B
          TRA     Z14
     *MAKE COMMON SETTINGS FOR COSP, SISP, COSISP AS IF IT WERE COSISP
     *FIRST FOR FIXED POINT OR FLOATING POINT
      Z14 ZET     T11
          TRA     Z15           FLOATING
          CLA     MPY           FIXED
          LDQ     ADD
          TRA     Z16
      Z15 CLA     FMP           FLOATING
          LDQ     FAD
      Z16 STO     Z51
          STO     Z61
          STO     Z71
          STO     Z81
          STO     Z91
          STQ     Z52
          STQ     Z62
          STQ     Z72
          STQ     Z82
          STQ     Z92
          STO     Z54
          STO     Z64
          STO     Z74
          STO     Z84
          STO     Z94
          STQ     Z55
          STQ     Z65
          STQ     Z75
          STQ     Z85
          STQ     Z95
          SLQ     Z108
          SLQ     Z109A
```

```
         SLQ     Z116
         SLQ     Z118
         CLA     KA2           SMSE
         STA     Z52
         STA     Z62
         STA     Z72
         STA     Z82
         CLA     KA3           SMSO
         STA     Z55
         STA     Z65
         STA     Z75
         STA     Z85
         CLA     KA4           SMCE
         STA     Z92
         CLA     KA5           SMCO
         STA     Z95
*THEN ADDRESSES
         CLA     T7            SINTAB (OR HASH)
         STA     Z50
         STA     Z53
         STA     Z60
         STA     Z63
         STA     Z70
         STA     Z73
         STA     Z80
         STA     Z83
         CLA     T4            RAA  (OR HASH)
         STA     Z51
         STA     Z61
         STA     Z71
         STA     Z81
         CLA     T3            RSA  (OR HASH)
         STA     Z54
         STA     Z64
         STA     Z74
         STA     Z84
         CLA     T6            COSTAB  (OR HASH)
         STA     Z90
         STA     Z93
         CLA     T1            RSS  (OR HASH)
         STA     Z91
         CLA     T2            RAS  (OR HASH)
         STA     Z94
         CLA     T8            M
         TZE     LV
         TMI     Z17
         CLA     Z131A
         STD     Z108
         STD     Z109A
         STD     Z116
         STD     Z118
         CLA     T8
 Z17     STD     Z101
         STD     Z103
         ADD     T8            2M
         STD     2M
         CLA     T5            P
         TMI     LV
         TZE     LV
         STD     Z105
         CLA     T12           COSTR  (OR HASH)
         STA     Z108
         STA     Z109A
         CLA     T13           SINTR  (OR HASH)
         STA     Z116
         STA     Z118
*FOR JMIN EVEN SET JE=JMIN+1,JO=JMIN+1,ESTOR=0,OSTOR=1
*     JMIN ODD  SET JO=JMIN,JE=JMIN+1),OSTOR=0,ESTOR=1
 Z20     CLA     T9            JMIN
         TMI     LV
         CAS     T10
         TRA     LV
         TRA     LV
         ARS     18
         LBT
```

```
        TRA     Z21             IS EVEN
        ALS     18              IS ODD
        STD     JO
        ADD     KD1
        STD     JE
        STZ     OSTOR
        CLA     K1
        STA     ESTOR
        TRA     Z23
 Z21    ALS     18              IS EVEN
        STD     JE
        ADD     KD1
        STD     JO
        STZ     ESTOR
        CLA     K1
        STA     OSTOR
*CLEAR DUMMY SWITCHES
 Z23    STZ     DUME
        STZ     DUMO
*NOW BEGIN LOOPING
*INITIALIZE Z105 SWITCH, CLEAR SUM REGISTERS, SET TRAVEL SWITCHES
*  FORWARD
 Z30    CLA     **              (**=KA6 COSP,  **=KA9 OTHERWISE)
        STA     Z105
        STZ     SMSE
        STZ     SMSO
        STZ     SMCE
        STZ     SMCO
        STZ     SWE
        STZ     SWO
        CLA     JE
        STD     Z100
        CLA     JO
        STD     Z102
*SET MINUS JE,JO
        LDC     JE,1
        SXD     MJE,1
        LDC     JO,1
        SXD     MJO,1
*XR4 WILL CONTROL MOTION FOR EVEN HARMONIC INDEX
*XR2 WILL CONTROL MOTION FOR ODD  HARMONIC INDEX
*XR1 WILL CONTROL MOTION FOR DATA INDEX
*DATA INDEX=SINE INDEX=COSINE INDEX=0
        AXT     0,7
 Z33    TRA     **              (**=Z90 FOR COSP,  =Z50  OTHERWISE)
*LOOP FOR FORWARD MOTION ON SINE WAVE FOR BOTH HARMONICS
*    THIS PART IS FOR EVEN HARMONICS (XR4) SUMMED IN SMSE
 Z50    LDQ     **,4            (**=SINTAB)
 Z51    NOP                     (MPY OR FMP $$,1  WITH ** = RAA)
 Z52    NOP                     (ADD OR FAD SMSE)
        STO     SMSE
*   THIS PART IS FOR ODD HARMONICS (XR2),  SUMMED IN SMSO
 Z53    LDQ     **,2            (**=SINTAB)
 Z54    NOP                     (MPY OR FMP  **,1  WITH **=RSA)
 Z55    NOP                     (ADD OR FAD SMSO
        STO     SMSO
*NOW GO TO COSINE SUMS IF COSISP, OR AVOID IF SISP
 Z56    TRA     **              (**=Z90 FOR COSISP,  **=Z100 FOR SISP)
*LOOP FOR FORWARD MOTION ON SINE WAVE OF EVEN HARMONIC AND
*    REVERSE MOTION ON SINE WAVE OF ODD HARMONIC
*   FOR EVEN
 Z60    LDQ     **,4            (**=SINTAB)
 Z61    NOP                     (MPY OR FMP **,1  WITH **=RAA)
 Z62    NOP                     (ADD OR FAD  SMSE)
        STO     SMSE
*   FOR ODD
 Z63    CLS     **,2            (**=SINTAB)
        XCA
 Z64    NOP                     (MPY OR FMP  **,1  WITH **=RSA)
 Z65    NOP                     (ADD OR FAD SMSO)
        STO     SMSO
 Z66    TRA     **              (**=Z90 IF COSISP, **=Z100 IF SISP)
*LOOP FOR REVERSE MOTION ON SINE WAVE OF EVEN HARMONIC AND
*     FORWARD MOTION ON SINE WAVE OF ODD HARMONIC
*   FOR EVEN
```

```
 Z70   CLS     **,4          (**=SINTAB)
       XCA
 Z71   NOP                   (MPY OR FMP  **,1  WITH **=RAA)
 Z72   NOP                   (ADD OR FAD SMSE)
       STO     SMSE
*  FOR ODD
 Z73   LDQ     **,2          (**=SINTAB)
 Z74   NOP                   (MPY OR FMP **,1  WITH **=RSA)
 Z75   NOP                   (ADD OR FAD SMSO)
       STO     SMSO
  Z76  TRA     **            (**=Z90 COSISP,  **=Z100 IF SISP)
*LOOP FOR REVERSE MOTION ON SINE WAVE FOR BOTH HARMONICS
*    THIS PART IS FOR EVEN HARMONICS
 Z80   CLS     **,4          (**=SINTAB)
       XCA
 Z81   NOP                   (MPY OR FMP  **,1  WITH **=RAA)
 Z82   NOP                   (ADD OR FAD SMSE)
       STO     SMSE
*    THIS PART IS FOR ODD HARMONICS
 Z83   CLS     **,2          (**=SINTAB)
       XCA
 Z84   NOP                   (MPY OR FMP  **,1    WITH **=RSA)
 Z85   NOP                   (ADD OR FAD SMSO)
       STO     SMSO
*NOW GO TO COSINE SUMS IF COSISP, OR AVOID IF SISP
 Z86   TRA     **            (**=Z90 FOR COSISP, **=Z100 FOR SISP)
*LOOP FOR FORWARD OR BACKWARD MOTION ON COSINE WAVE
*    THIS PART FOR EVEN HARMONICS   SUMMED IN SMCE
 Z90   LDQ     **,4          (**=COSTAB)
 Z91   NOP                   (MPY OR FMP  **,1  WITH **=RSS)
 Z92   NOP                   (ADD OR FAD SMCE)
       STO     SMCE
*    THIS PART IS FOR ODD HARMONICS   SUMMED IN SMCO
 Z93   LDQ     **,2          (**=COSTAB)
 Z94   NOP                   (MPY OR FMP  **,1  WITH **=RAS)
 Z95   NOP                   (ADD OR FAD SMCO)
       STO     SMCO
*INCREMENT INDEX FOR EVEN HARMONICS (BY +JE FOR FORWARD
*    TRAVEL, BY -JE FOR REVERSE TRAVEL)
 Z100  TXI     **+1,4,**     (**=JE  FORWARD)    (**=-JE  REVERSE)
*CHECK IF INDEX HAS RUN OFF END (GREATER THAN M FOR
*    FORWARD TRAVEL, LESS THAN ZERO FOR REVERSE)
*    (HOWEVER FOR REVERSE TRAVEL XR4 GOING NEGATIVE MEANS
*     XR4 GETS GREATER THAN M, SO SAME TEST APPLIES)
 Z101  TXH     Z120,4,**     **=M
*INCREMENT INDEX FOR ODD HARMONICS (BY+JO OR -(JO))
*     AND MAKE SAME KIND OF END TEST
 Z102  TXI     **+1,2,**     (**=JO FORWARD)     (**=-JO REVERSE)
 Z103  TXH     Z110,2,**     (**=M)
*INCREMENT DATA INDEX BY 1 AND CHECK FOR END OF DATA
*    LOOPING BACK TO PLACE DETERMINED BY WHETHER COSP OR
*    SISP OR COSISP AND FORWARD OR BACKWARD AND EVEN OR ODD
 Z104  TXI     **+1,1,1
 Z105  TXL     **,1,**       (TXL  **A,1,**B     **B=P)
*                            **A=Z90 FOR COSP
*                              FOR SISP OR COSISP  (INITIAL = Z50)
*                            **A=Z50  EVEN AND ODD HARMONICS FORWARD
*                            **A=Z60  EVEN FORWARD, ODD REVERSE
*                            **A=Z70  EVEN REVERSE, ODD FORWARD
*                            **A=Z80  EVEN AND ODD REVERSE
 Z106  TRA     **            (**=Z107 FOR COSP OR COSISP,
*                                             **=Z115 FOR SISP)
*READJUSTMENTS WHEN ODD HARMONIC INDEX RUNS OFF END
*FORWARD OR BACKWARD
 Z110  ZET     SWO
       TRA     Z113          BACKWARD
       CLA     K1
       STO     SWO
*IF FORWARD SET TO GO BACKWARD ON ODD
 Z111  SXD     TEMP,2
       CLA     2M
       SUB     TEMP
       PDX     0,2
       CLA     MJO
       STD     Z102
```

```
     *IF COSP GO BACK, IF NOT REMAKE FORK AT Z105
     *                             COSP                 SISP OR COSISP
      Z112  NOP                    (TRA  Z104   OR      ZET SWE)
            TRA     Z112A
            CLA     KA10           (KA10 = PZE Z60)
            STA     Z105
            TRA     Z104
      Z112A CLA     KA12           (KA12=PZE Z80)
            STA     Z105
            TRA     Z104
     *IF BACKWARDS SET TO GO FORWARDS ON ODD
      Z113  STZ     SWO
            PXA     0,2
            PAC     0,2
            CLA     JO
            STD     Z102
     *IF COSP GO BACK, IF NOT REMAKE FORK AT Z105
     *                             COSP                 SISP OR COSISP
      Z114  NOP                    (TRA  Z104      OR   ZET  SWE)
            TRA     Z114A
            CLA     KA9            (KA9=PZE  Z50)
            STA     Z105
            TRA     Z104
      Z114A CLA     KA11           (KA11=PZE Z70)
            STA     Z105
            TRA     Z104
     *READJUSTMENT WHEN EVEN HARMONIC INDEX RUNS OFF END
     *WHICH WAY WERE WE GOING
      Z120  ZET     SWE
            TRA     Z122           BACKWARDS
     *IF FORWARD, REVERSE SWE, READJUST IR4 AND DECREM OF TXI
      Z121  CLA     K1
            STO     SWE
            SXD     TEMP,4         RESET I*JE TO 2M-I*JE
            CLA     2M
            SUB     TEMP
            PDX     0,4
            CLA     MJE
            STD     Z100
     *IS COSP GO BACK, IF NOT REMAKE FORK AT Z105
      Z121A NOP                    (TRA Z102(COSP) ZET SWO (SISP,COSISP))
            TRA     Z121B
            CLA     KA11           (KA11=Z70)
            STA     Z105
            TRA     Z102
      Z121B CLA     KA12           (KA12=Z80)
            STA     Z105
            TRA     Z102
     *  IF BACKWARDS
      Z122  STZ     SWE
            PXA     0,4
            PAC     0,4
            CLA     JE
            STD     Z100
     *IF COSP GO BACK, IF NOT REMAKE FORK AT Z105
      Z122A NOP                    (TRA Z102 (COSP),ZET SWO (SISP,COSISP))
            TRA     Z122B
            CLA     KA9            (KA9=Z50)
            STA     Z105
            TRA     Z102
      Z122B CLA     KA10           (KA10=Z60)
            STA     Z105
            TRA     Z102
     *COSP OR COSISP RESULT STORAGE FOR COSINE TRANSFORMS
     *WAS LAST EVEN HARMONIC A DUMMY
      Z107  ZET     DUME
            TRA     Z109           YES
     *IF NOT STORE SMCE IN COSTR BLOCK
            LXA     ESTOR,4
            CLA     SMCE
      Z108  ADD     **,4             (**=COSTR)
            STO*    *-1
     *WAS LAST ODD HARMONIC A DUMMY
      Z109  ZET     DUMO
            TRA     Z109B          YES
```

```
*IF NOT STORE SMCO IN COSTR BLOCK
       LXA     OSTOR,4
       CLA     SMCO
 Z109A ADD     **,4           (**=COSTR)
       STO*    *-1
 Z109B TRA     **             (**=Z115 COSISP, **=Z130 COSP)
*COSISP OR SISP RESULT STORAGE FOR SINE TRANSFORMS
*WAS LAST EVEN HARMONIC A DUMMY
 Z115  ZET     DUME
       TRA     Z117          YES
*IF NOT STORE SMSE IN SINTR BLOCK
       LXA     ESTOR,4
       CLA     SMSE
 Z116  ADD     **,4           (**=SINTR)
       STO*    *-1
*WAS LAST ODD HARMONIC A DUMMY
 Z117  ZET     DUMO
       TRA     Z130          YES
*IF NOT STORE SMSO IN SINTR BLOCK
       LXA     OSTOR,4
       CLA     SMSO
 Z118  ADD     **,4           (**=SINTR)
       STO*    *-1
*RESET FOR NEXT LOOP STORAGE
  Z130 CLA     ESTOR
       ADD     K2
       STO     ESTOR
       CLA     OSTOR
       ADD     K2
       STO     OSTOR
*INDEX JE BY TWO AND CHECK IF TOO BIG
       CLA     JE
       ADD     KD2
       STD     JE
       CAS     T10           COMPARE WITH JMAX
       TRA     Z135          TOO BIG
       NOP                   OK
*IF NEW JE OK, INDEX JO BY TWO AND CHECK ITS SIZE
  Z131 CLA     JO
       ADD     KD2
       STD     JO
       CAS     T10
       TRA     Z133          TOO BIG
 Z131A NOP                    OK
*RETURN TO BEGINNING OF LOOP
  Z132 TRA     Z30
*IF JO TOO BIG SET SWITCH
 Z133  CLA     K1
       STO     DUMO
*IS JE ALSO TOO BIG
       ZET     DUME
       TRA     LV            YES - ALL FINISHED
       TRA     Z132          NO - ONE MORE TO GO
*IF JE TOO BIG SET SWITCH
 Z135  CLA     K1
       STO     DUME
       TRA     Z131          GO CHECK JO
*FINAL EXIT
  LV   LXD     COSP-2,4
       AXT     **,1          (**=IR1)
       AXT     **,2          (**=IR2)
  EXIT TRA     **,4          (**=10 FOR COSP OR SISP, **=14 FOR COSISP)
*CONSTANTS, TEMPORARIES, ETC
  SWE  PZE     **            (**=0 WHILE EVEN HARMONIC GOING FORWARDS)
*                            (**=1 WHILE EVEN HARMONIC GOING BACKWARD)
  SWO  PZE     **            (**=0 WHILE ODD HARMONIC FORWARDS)
*                            (**=1 WHILE ODD HARMONIC BACKWARDS)
  JE   PZE     0,0,**        **=JE
 MJE   PZE     0,0,**        **=25 COMP OF JE
  JO   PZE     0,0,**        **=JO
 MJO   PZE     0,0,**        **=25 COMP OF JO
  DUME PZE     **            (**=0 FOR REAL EVEN,**=1 FOR DUMMY EVEN)
  DUMO PZE     **            (**=0 FOR REAL ODD,**=1 FOR DUMMY ODD)
 ESTOR PZE     **      (**=ZERO INDEX OF INITIAL EVEN HARMONIC STORAGE)
 OSTOR PZE     **      (**=ZERO INDEX OF INITIAL ODD HARMONIC STORAGE)
```

```
       MPY  MPY    **,1
       FMP  FMP    **,1
       ADD  ADD    **
       FAD  FAD    **
       SMSE PZE    **               SUM FOR EVEN HARMONIC SINE TRANSFORM
       SMSO PZE    **               SUM FOR ODD HARMONIC SINE TRANSFORM
       SMCE PZE    **               SUM FOR EVEN HARMONIC COSINE TRANSFORM
       SMCO PZE    **               SUM FOR ODD HARMONIC COSINE TRANSFORM
       2M   PZE    0,0,**           (**=2M)
       TEMP PZE    **
       T1   PZE    **               (**=RSS)
       T2   PZE    **               (**=RAS)
       T3   PZE    **               (**=RSA)
       T4   PZE    **               (**=RAA)
       T5   PZE    0,0,**           (**=P)
       T6   PZE    **               (**=COSTAB)
       T7   PZE    **               (**=SINTAB)
       T8   PZE    0,0,**           (**=M)
       T9   PZE    0,0,**           (**=JMIN)
       T10  PZE    0,0,**           (**=JMAX)
       T11  PZE    **               (**=TYPE)
       T12  PZE    **               (**=COSTR)
       T13  PZE    **               (**=SINTR)
       K0   PZE    0
       K1   PZE    1
       K2   PZE    2
       K10  PZE    10
       K14  PZE    14
       KT1  TRA    Z104
      KT2   TRA    Z102
       KZ1  ZET    SWE
      KZ2   ZET    SWO
       KD1  PZE    0,0,1
       KD2  PZE    0,0,2
       KA2  PZE    SMSE
       KA3  PZE    SMSO
       KA4  PZE    SMCE
       KA5  PZE    SMCO
       KA6  PZE    Z90
       KA7  PZE    Z100
       KA8  PZE    Z30
       KA9  PZE    Z50
      KA10  PZE    Z60
       KA11 PZE    Z70
       KA12 PZE    Z80
       KA13 PZE    KA8
       KA14 PZE    KA9
       KA15 PZE    Z107
       KA16 PZE    Z115
       KA17 PZE    Z120
       KA18 PZE    KA6
       KA19 PZE    Z130
            END
```

```
*      COSTBL (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0199
*      FAP
*COSTBL
       COUNT   200
       LBL     COSTBL
       ENTRY   COSTBL (N,COSTAB)
       ENTRY   SINTBL (N,SINTAB)
       ENTRY   COSTBX (N,ICOSTB)
       ENTRY   SINTBX (N,ISINTB)
*
*                ----ABSTRACT----
*
* TITLE - COSTBL WITH SECONDARY ENTRY POINTS SINTBL, COSTBX, SINTBX
*        GENERATE COSINE OR SINE HALF-WAVE TABLES, FIXED OR FLOATING
*
*              COSTBL GENERATES A HALF-WAVE COSINE TABLE FLOATING POINT
*              SINTBL GENERATES A HALF-WAVE SINE TABLE FLOATING POINT
*              COSTBX GENERATES A HALF-WAVE COSINE TABLE FIXED POINT
*              SINTBX GENERATES A HALF-WAVE SINE TABLE FIXED POINT
*                 WHERE
*                    THE HALF-WAVE LENGTH IS AN INPUT PARAMETER.
*                    FOR FIXED POINT TABLES THE BINARY POINT IS BETWEEN
*                        THE SIGN BIT AND BIT 1.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 121 REGISTERS
* SPEED     - ABOUT 2N MILLISEC ON 709, WHERE N = HALF-WAVE LENGTH
* AUTHOR    - JON CLAERBOUT
*
*                ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*      AND FORTRAN SYSTEM  ROUTINES - COS,SIN
*
* FORTRAN USAGE OF COSTBL
*     CALL COSTBL(N,COSTAB)
*
* INPUTS TO COSTBL
*    N          DEFINES THE HALF-WAVE LENGTH TO BE N+1
*               MUST EXCEED ZERO (PROGRAM EXITS IF N IS NEGATIVE OR ZERO)
*
* OUTPUTS FROM COSTBL
*    COSTAB(I) I=1...N+1  CONTAINS TABLE(J) = COS(J*PI/N)  J=0,1,...,N
*                    I.E. COSTAB(I) CONTAINS TABLE(I-1)
*
* FORTRAN USAGE OF SINTBL
*     CALL SINTBL(N,SINTAB)
* INPUTS TO SINTBL
*    N          SAME MEANING AS FOR COSTBL
* OUTPUTS FROM SINTBL
*    SINTAB(I) I=1...N+1 CONTAINS  TABLE(J) = SIN(J*PI/N) FOR J=0,1...N
*
* FORTRAN USAGE OF COSTBX
*     CALL COSTBX(N,ICOSTB)
* INPUTS TO COSTBX
*    N          SAME MEANING AS FOR COSTBL
* OUTPUTS FROM COSTBX
*    ICOSTB(I) I=1...N+1 IS SAME AS FOR COSTBL BUT DATA IS FIXED POINT
*
* FORTRAN USAGE OF SINTBX
*     CALL SINTBX(N,ISINTB)
* INPUTS TO SINTBX
*    N          SAME MEANING AS FOR COSTBL
* OUTPUTS FROM SINTBX
*    ISINTB(I) I=1...N+1 IS SAME AS FOR SINTBL BUT DATA IS FIXED POINT
*
* EXAMPLES
* 1. GENERAL BEHAVIOR FOR N=4
*    INPUTS  - N=4
*    USAGE   -        CALL COSTBL(N,COSTAB)
*                     CALL SINTBL(N,SINTAB)
*                     CALL COSTBX(N,ICOSTB)
*                     CALL SINTBX(N,ISINTB)
*    OUTPUTS - NOTE - THESE NUMBERS ARE GOOD TO 8 OCTAL PLACES.
```

```
*              COSTAB(1...5) = 1.0,.70711,0.0,-.70711,-1.0
*              SINTAB(1...5) = 0.0,.70711,1.0,.70711,0.0
*              ICOSTB(1...5) = OCT 377777777777,265011714000,
*                                   000000000000,665011714000,777777777777
*              ISINTB(1...5) = OCT 000000000000,265011714000,
*                                   377777777777,265011714000,000000000000
*
        HTR    0
        BCI    1,COSTBL
COSTBL  CLA    *
        STO    FL
        TRA    *+3
COSTBX  STZ    FL
        STZ    CORS
        SXD    COSTBL-2,4
        SXA    SV,1
        CLA    KCOS           (TSX  $COS,4)
        STO    AL
        CLA    2,4            GET COSINS
        STA    B3
        ADD    =1             COSINS+1
        STA    A
        STA    B
        STA    B1
        STA    B2
        STA    B4
        TRA    D
SINTBL  CLA    *
        STO    FL
        TRA    *+4
SINTBX  STZ    FL
        CLA    *
        STO    CORS
        SXD    COSTBL-2,4
        SXA    SV,1
        CLA    KSIN           (TSX  $SIN,4)
        STO    AL
* SET UP FIXING LOOP
        CLA    2,4            GET SINS
        ADD    =1             SINS+1
        STA    A
        STA    B
        STA    B1
        STA    B2
        STA    L2
* SET UP COMPUTATION LOOP
 D      CLA*   1,4            GET N
        TZE    SV
        TMI    SV
        STD    N
        ADD    KD1            FORM N+1
        STD    AN
        STD    BN
        CLA    N              FLOAT N
        ARS    18
        ORA    ORF
        FAD    ORF
        STO    NFL
        CLA    =3.14159265         FORM PI/N
        FDP    NFL
        STQ    INCR
        STZ    ARG
* LOOP
        AXT    1,1            COS                  SIN
        CLA    ARG
 AL     NOP    **             TSX $COS,4           TSX $SIN,4
 A      STO    **,1           **=COSINS+1          **=SINS+1
        CLA    ARG
        FAD    INCR
        STO    ARG
        TXI    *+1,1,1
 AN     TXL    AL,1,**        **=N+1
        ZET    FL             FIX IF ZERO
        TRA    SV             EXIT - NOT ZERO
        AXT    1,1
```

****
* COSTBL *
****
(PAGE 3)

```
BC      CLM
B       LDQ     **,1            **=COSINS+1
        LLS     8
        SSP
        SUB     =0200
        STA     RTSH
B1      CLA     **,1            **=COSINS+1
        LRS
        ANA     =0000777777777
        ALS     8
        LLS
RTSH    ARS     **              ** FROM B+4
B2      STO     **,1            **=COSINS+1
        TXI     *+1,1,1
BN      TXL     BC,1,**         **=N+1
        CLA     CORS
        TNZ     L1
        CLA     =0377777777777                          SET FIRST AND
B3      STO     **              **=COSINS               LAST VALUES
        SSM                                             IN TABLE = 1
        LXD     BN,1
B4      STO     **,1            **=COSINS+1
        TRA     SV
L1      CLA     N
        ARS     18
        LBT                     IF = 0, N EVEN - EXIT
        TRA     *+2
        TRA     SV
        CLA     N               N ODD - SET MDPT = 1
        ARS     1               GET (N+1)/2
        ADD     KD1
        STD     MD
        CLA     =0377777777777
        LXD     MD,1
L2      STO     **,1            ** = SINS+1
SV      AXT     **,1
        LXD     COSTBL-2,4
        TRA     3,4
 N      PZE     **              **=N IN DECR
FL      PZE     **              **=0,FXD
INCR    PZE     **              **=PI/N.
ARG     PZE     **              **=I*PI/N,  I=0,1,...,N
ORF     OCT     233000000000
NFL     PZE     **              **=FLOATF(N)
KD1     PZE     0,0,1
KCOS    TSX     $COS,4
KSIN    TSX     $SIN,4
CORS    PZE     **              **=0 IF COS
MD      PZE     0,0,**          **=(N+1)/2
        END
```

```
************************
*    COSTBX            *
************************
 REFER TO
    COSTBL
```


```
************************
*    COSTBX            *
************************
 REFER TO
    COSTBL
```

```
*       CPYFL2  (SUBROUTINE)              9/9/64   LAST CARD IN DECK IS NO. 0303
*       FAP
*CPYFL2
        COUNT   300
        LBL     CPYFL2
        ENTRY   CPYFL2 (ITPIN,ITPOUT,LRECMX,ZFEOFW,SPACE,IANS)
*
*                     ----ABSTRACT----
*
* TITLE - CPYFL2
*       FAST COPY FILE FROM ONE TAPE TO ANOTHER - VERSION 2
*
*               CPYFL2 COPIES ONE FILE OF BCD AND/OR BINARY RECORDS FROM
*               ONE TAPE TO ANOTHER TAPE.  THE END-OF-FILE MARK IS ALSO
*               COPIED IF THE USER DESIRES IT.
*
*               IF A REDUNDANCY IS ENCOUNTERED ON EITHER TAPE, CPYFL2
*               WILL ATTEMPT TO RECOPY 20 TIMES BEFORE GIVING AN ERROR
*               EXIT.  AN ERROR EXIT WILL ALSO OCCUR IF AN END-TAPE
*               CONDITION IS SENSED.
*
*               CPYFL2 DERIVES ITS SPEED BY OPERATING BOTH TAPES
*               SIMULTANEOUSLY IF THEY ARE ON DIFFERENT DATA CHANNELS.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN (II) COMPATIBLE)
* EQUIPMENT   - 709, 7090, OR 7094 (MAIN FRAME AND TWO TAPE UNITS)
* STORAGE     - 178 REGISTERS
* SPEED       - FOR THE 7090 (556 BPI)
*                0.00927 SECONDS/14 WORD RECORD  IF THE TAPES ARE ON
*                                                   DIFFERENT CHANNELS.
*                0.01828 SECONDS/14 WORD RECORD  IF THE TAPES ARE ON
*                                                   THE SAME CHANNEL.
*               (NOTE THAT FORTRAN COPYING (READING ONE RECORD AND THEN
*               WRITING ONE RECORD) REQUIRES 0.02845 SECONDS/PER 14 WORD
*               RECORD.)
* AUTHOR        R.A. WIGGINS   JULY, 1964
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*      AND FORTRAN SYSTEM ROUTINES - (IOS), (TCO), (WRS), (RCH), (TRC),
*                                    (ETT), (WEF), (BSR), (RDS)
*
* FORTRAN USAGE
*     CALL CPYFL2(ITPIN,ITPOUT,LRECMX,ZFEOFW,SPACE,IANS)
*
* INPUTS
*
*    ITPIN      LOGICAL TAPE NUMBER THAT FILE IS COPIED FROM.
*
*    ITPOUT     LOGICAL TAPE NUMBER THAT FILE IS TO BE COPIED ONTO.
*
*    LRECMX     MAXIMUM NUMBER OF WORDS PER RECORD THAT WILL BE COPIED.
*                 IF RECORDS ARE ENCOUNTERED THAT ARE LONGER, THEY WILL
*                 BE CHOPPED AT THIS LENGTH AND THE REMAINING WORDS WILL
*                 BE LOST.
*               SOME STANDARD FORTRAN RECORD LENGTHS ARE
*                  BCD CARDS                  -  14 WORDS
*                  BCD OUTPUT RECORDS         -  22 WORDS
*                  BINARY CARDS               -  27 WORDS
*                  BINARY OUTPUT RECORDS      - 256 WORDS
*               MUST BE GRTHN 1 BUT IS NOT CHECKED.
*
*    ZFEOFW     IS ZERO IF AN END-OF-FILE IS TO BE WRITTEN ON ITPOUT.
*               IS NOT ZERO IF NO END-OF-FILE IS TO BE WRITTEN.
*
*    SPACE(I)   I=1,...,2*LREC  IS TEMPORARY STORAGE SPACE NEEDED BY
*                 CPYFL2.
*
* OUTPUTS
*
*    ONE FILE FROM TAPE ITPIN IS COPIED ONTO ITPOUT.  ITPIN IS LEFT
*    POSITIONED AFTER THE END-OF-FILE MARK AND  ITPOUT  IS LEFT
*    POSITIONED AFTER THE END-OF-FILE MARK IF  ZFEOFW = 0.,  OR AFTER
*    THE LAST RECORD COPIED IF  ZFEOFW NOT= 0.  IF A PERMANENT
```

```
*    REDUNDANCY IS ENCOUNTERED ON ITPIN, IT IS LEFT POSITIONED AFTER THE
*    REDUNDANT RECORD AND ITPOUT IS LEFT POSITIONED AFTER THE PREVIOUS
*    RECORD.  IF A PERMANENT REDUNDANCY IS ENCOUNTERED ON ITPOUT, ITPIN
*    IS LEFT POSITIONED AFTER THE RECORD IMMEDIATELY SUCCEEDING THE
*    REDUNDANT RECORD AND ITPOUT IS LEFT POSITIONED AFTER THE REDUNDANT
*    RECORD.  IF AN END-TAPE CONDITION IS SENSED ON ITPIN, ITPOUT IS
*    LEFT POSITIONED AFTER THE LAST SUCCESSFULLY READ RECORD.  IF AN
*    END-TAPE CONDITION IS SENSED ON ITPOUT, ITPIN IS LEFT POSITIONED
*    ONE RECORD BEYOND THE LAST RECORD SUCCESSFULLY COPIED.
*
*    SPACE(I)  CONTAINS THE NUMBER OF RECORDS COPIED IN THIS FILE
*                (FIXED POINT).
*
*    IANS      = 0 IF ALL OK.
*              = 1 IF REDUNDANCY ON ITPIN (AFTER 20 ATTEMPTS TO RECOPY).
*              = 2 IF REDUNDANCY ON ITPOUT (AFTER 20 ATTEMPTS TO RECOPY)
*              = 3 IF REDUNDANCIES ON ITPIN AND ITPOUT (AFTER 20
*                    ATTEMPTS TO RECOPY).
*              = 4 IF END TAPE ON ITPIN.
*              = 5 IF END TAPE AND REDUNDANCY ON ITPIN.
*              = 6 IF END TAPE ON ITPIN, REDUNDANCY ON ITPOUT.
*              = 7 IF END TAPE ON ITPIN, REDUNDANCIES ON ITPIN AND
*                    ITPOUT.
*              = 8 IF END TAPE ON ITPOUT.
*              = 9 IF END TAPE ON ITPOUT, REDUNDANCY ON ITPIN.
*              =10 IF END TAPE AND REDUNDANCY ON ITPOUT.
*              =11 IF END TAPE ON ITPOUT, REDUNDANCIES ON ITPIN AND
*                    ITPOUT.
*              =12 IF END TAPE ON ITPIN AND ITPOUT.
*              =13 IF END TAPE ON ITPIN AND ITPOUT, REDUNDANCY ON ITPIN.
*              =14 IF END TAPE ON ITPIN AND ITPOUT, REDUNDANCY OF ITPOUT
*              =15 IF END TAPE AND REDUNDANCIES ON ITPIN AND ITPOUT.
*
*
* EXAMPLE
*
* 1. A COMPREHENSIVE TEST  -  END-OF-FILE CONTROL AND COPYING
*                             ALTERNATING BCD AND BINARY RECORDS.
*    INPUTS  - ITPIN = 6      ITPOUT = 8     LRECMX = 10     ZFEOF1 = 1.
*              IA(1...5) = 1,2,3,4,5                          ZFEOF2 = 0.
*    USAGE   - C SET UP ITPIN WITH ALTERNATING BCD AND BINARY RECORDS.
*                    REWIND ITPIN
*                    WRITE OUTPUT TAPE ITPIN,10,IA(1)
*               10   FORMAT(5I6)
*                    WRITE TAPE ITPIN, (IA(I),I=1,3)
*                    WRITE OUTPUT TAPE ITPIN,10,(IA(I),I=1,5)
*                    END FILE ITPIN
*                    REWIND ITPIN
*                    REWIND ITPOUT
*              C COPY THE FILE TWICE
*                    CALL CPYFL2(ITPIN,ITPOUT,LRECMX,ZFEOF1,SPACE,IANS)
*                    REWIND ITPIN
*                    CALL CPYFL2(ITPIN,ITPOUT,LRECMX,ZFEOF2,SPACE,IANS)
*              C READ THE FILE FROM ITPOUT (LAST READ SHOULD CAUSE EXIT)
*                    REWIND ITPOUT
*                    READ INPUT TAPE ITPOUT,IB(1)
*                    READ TAPE ITPOUT,(IB(I),I=2,4)
*                    READ INPUT TAPE ITPOUT,(IB(I),I=5,I0)
*                    READ TAPE ITPOUT,(IB(I),I=11,13)
*                    READ INPUT TAPE ITPOUT,(IB(I),I=14,18)
*                    READ INPUT TAPE ITPOUT,IB(19)
*    OUTPUTS - IANS = 0     IB(1...18) = 1, 1,2,3, 1,2,3,4,5,
*                                        1, 1,2,3, 1,2,3,4,5
*
*
* PROGRAM FOLLOWS BELOW
*
 XR4   HTR     0
       BCI     1,CPYFL2
CPYFL2 SXD     XR4,4               SAVE
       SXA     IR1,1               INDEX
       SXA     IR2,2               REGISTERS.
       CLA     11                  SAVE
       STO     NIF1                TRAPPING
       CLA     13                  INSTRUCTIONS.
```

****
* CPYFL2 *
****
(PAGE 3)


```
         STO    NIF2
         AXT    20,2
         CLA*   4,4
         STO    WEFSW
         CLA*   3,4
         STD    IN
         PDX    ,1
         SXA    LREC,1
         CAL    5,4                 CHANNEL
         ADD    =1
         SUB    LREC                COMMANDS.
         STA    IN
         SUB    LREC
         STA    UT
         AXT    0,1
   TPSET LXD    XR4,4
         CLA*   2,4                 GET OUTPUT TAPE NO. (ITPOUT)
         ZET    MODE                TEST FOR
         ADD    =020                BINARY MODE
         TSX    $(IOS),4            SET UP INSTRUCTIONS IN (IOS)
         LXD    XR4,4               RESTORE IR 4.
         LDQ*   $(TCO)              SET UP
         SLQ    TCOA                INSTRUCTIONS
         LDQ*   $(WRS)              BY
         STQ    WRSA                A.
         LDQ*   $(RCH)
         SLQ    RCHA
         LDQ*   $(TRC)
         SLQ    TRCA
         LDQ*   $(ETT)
         STQ    ETTA
         LDQ*   $(WEF)
         ZET    WEFSW
         LDQ    NOP
         STQ    WEFA
         LDQ*   $(BSR)
         STQ    BSRA1
         CLA*   1,4                 GET INPUT TAPE NO. (ITPIN)
         ZET    MODE                TEST FOR
         ADD    =020                BINARY MODE.
         TSX    $(IOS),4            SET UP INSTRUCTIONS IN (IOS).
         LDQ*   $(TCO)              SET
         SLQ    TCOB                UP
         SLQ    TCOB1
         SLQ    TCOB2
         LDQ*   $(RDS)              INSTRUCTIONS
         STQ    RDSB                DESIGNATED
         LDQ*   $(BSR)
         STQ    BSRB1
         STQ    BSRB2
         LDQ*   $(RCH)              BY
         SLQ    RCHB                B.
         CLA    SCHB
         LLS    0
         XCA
         SLQ    SCHB
         LDQ*   $(ETT)
         STQ    ETTB
         XCL
         ANA    =03000
         ARS    9
         PAX    ,4
         SXD    ENBIN,4
         SXA    ENBIN,4
         ALS    1
         ADD    =8B35
         STA    ICB
         CLA    TRA2                STORE
         STO    11                  TRAPPING
         STO    13                  INSTRUCTIONS.
    TCOB TCOB   *                   DELAY IF CHANNEL IN OPERATION.
    RDSB RTDB   **                  READ SELECT.
         ENB    ENBIN               ENABLE INPUT DATA CHANNEL.
    RCHB RCHB   IN                  RESET AND LOAD CHANNEL.
    SCHB SCHB   DC                  MONITOR
```

```
       TRA    *-1          READING PROCESS.
 ICB   LXD    **,4        *REENTRY FROM TRAP.
       ENB    =0           DISENABLE CHANNELS FOR OUTPUT.
       TRA    TRA1+1,4     CHECK
       TRA    ENDFIL
       TRA    ENDFIL
       TRA    ENDFIL
       TRA    ENDFIL       REASON
       TRA    REDUNB       FOR
       TRA    REDUNB       TRAPPING.
 TRA1  CLA    =077777      END RECORD.
       ANS    DC           SWITCH
       LDQ    IN           INPUT
       CLA    UT           AND
       STA    IN           OUTPUT
       XCA
       STA    UT           ADDRESSES.
       ANA    =077777      FIND
       SUB    DC           NO.
       ALS    18           OF
       STD    UT           WORDS IN REC.
       AXT    20,2
       PXD    ,0           CHECK
 ETTB  ETTB                FOR
       ADD    =4B17        REDUNDANCY ON ITPOUT
 TCOA  TCOA   *
 TRCA  TRCA   REDUNA       OR
 ETTA  ETTA                END TAPE
       ADD    =8B17        CONDITIONS
       TMI    WEFA         GO WRITE END-OF-FILE ON ITPOUT.
       TNZ    END          LEAVE IF END TAPE OF END OF FILE)
 WRSA  WTDA   **           WRITE THIS
 RCHA  RCHA   UT           RECORD.
       TXI    TCOB,1,1     BUMP RECORD COUNTER AND GO TO NEXT REC
REDUNA ADD    =2B17        SIGNAL ITPOUT REDUNDANCY
       TNX    END,2,1      SLICE REDUNDANCY COUNTER
 TCOB2 TCOB   *
 BSRB2 BSRB   **
 BSRA1 BSRA   **           PREPARE TO RETRY WRITING
       TRA    TCOB1
ENDFIL CLS    =1B17
       TRA    ETTB
 WEFA  WEFA   **           NOP IF ZFEOFW NOT= 0.
       CLA    =1           SIGNAL TO LEAVE AFTER WRITING
       TRA    TCOA         CHECK FOR END TAPE
REDUNB CLA    =1B17        SIGNAL ITPIN REDUNDANCY
       TNX    ETTB,2,1     SLICE REDUNDANCY COUNTER
       CLA    MODE         CHANGE
       ADD    =1           MODE
       ANA    =1           AND
       STO    MODE         PREPARE TO
 TCOB1 TCOB   *            RETRY READING
 BSRB1 BSRB   **           IN ANOTHER MODE
       TRA    TPSET
 END   LXD    XR4,4        RESET IR 4.
       STD*   6,4          SET IANS, AND
       CLA    NIF1         RESTORE
       STO    13           TRAPPING
       CLA    NIF2         INSTRUCTIONS
       STO    14
       PXD    ,1
       STO*   5,4
 IR1   AXT    **,1
 IR2   AXT    **,2
       TRA    7,4
 TRA2  TRA    ICB
*      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 NOP   NOP
 MODE  PZE
 NIF1  PZE
 NIF2  PZE
 LREC  PZE
 IN    IORT   **,,**
 UT    IORT   **,,**
 ENBIN PZE    0
```

```
      DC     PZE
      WEFSW  PZE
     *      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
             END
```

```
*     CROSS (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0086
*     LABEL
CCROSS
      SUBROUTINE CROSS (LX,X,LY,Y,LC,C)
C
C                   ----ABSTRACT----
C
C TITLE - CROSS
C       CROSSCORRELATION OF TRANSIENTS BEGINNING WITH ZERO LAG
C
C              CROSS FINDS THE CROSSCORRELATION OF TRANSIENTS
C              BEGINNING WITH ZERO LAG.
C
C                              LX
C                    C(K+1) = SUM ( X(I) * Y(I-K) )
C                              I=1
C
C                    FOR  K = 0,...,LC
C
C              WHERE X AND Y ARE TRANSIENT SERIES OF LENGTH LX AND LY
C              RESPECTIVELY, AND LC IS AN INPUT PARAMETER.  THE COMPU-
C              TATION IS MADE AS THOUGH X AND Y HAD ZEROS EXTENDING
C              BEYOND THEIR ENDS.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 107  REGISTERS
C SPEED      -
C AUTHOR     - R.A. WIGGINS
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FDOT, STZ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL CROSS (LX,X,LY,Y,LC,C)
C
C INPUTS
C
C    LX        LENGTH OF X SERIES.
C
C    X(I)      I=1,...,LX IS THE X TRANSIENT SERIES.
C
C    LY        LENGTH OF Y SERIES.
C
C    Y(I)      I=1,...,LY IS THE Y TRANSIENT SERIES.
C
C    LC        IS THE DESIRED LENGTH OF THE CROSSCORRELATION.
C
C    NOTE  --  IF LC, LX, OR LY ARE LESS THAN 1, THE ROUTINE EXITS WITH
C              NO COMPUTATION.
C
C OUTPUTS
C
C    C(I)      I=1,...,LC IS THE CROSSCORRELATION SERIES.  THIS VECTOR
C                IS SET TO ZERO BEFORE COMPUTATIONS ARE MADE.
C
C EXAMPLES
C
C 1. INPUTS  - LX=3  X(1...3) = 1.,2.,3.  LY=2  Y(1...2) = 2.,1.
C              LC=5  C(1...5) = .1,.1,.1,.1,.1
C    OUTPUTS -       C(1...5) = 4.,7.,6.,0.,0.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LC=2
C    OUTPUTS - C(1...5) = 4.,7.,.1,.1,.1
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LC=0
C    OUTPUTS - C(1...5) = .1,.1,.1,.1,.1
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LX=0
C    OUTPUTS - C(1...5) = 0.,0.,0.,0.,0.
C
C 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LY=0
```

****
*   CROSS   *
****
(PAGE 2)

```
      C    OUTPUTS - C(1...5) = 0.,0.,0.,0.,0.
      C
      C PROGRAM FOLLOWS BELOW
      C
            DIMENSION X(2),Y(2),C(2)
            IF (LC) 30,30,10
         10 CALL STZ (LC,C)
            IF (XMINOF(LX,LY)) 30,30,15
         15 LC1=XMINOF(LX,LC)
            DO 20 I=1,LC1
         20 CALL FDOT (XMINOF(LY+I-1,LX)-I+1,X(I),Y,C(I))
         30 RETURN
            END
```

```
*     CROST (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0098
*     LABEL
CCROST
      SUBROUTINE CROST (LX,XX,LY,YY,K,LC,CC)
C
C                      ----ABSTRACT----
C
C TITLE - CROST
C
C       CROSSCORRELATION OF TRANSIENTS BEGINNING WITH ANY LAG
C
C              CROST FINDS LC TERMS OF THE CROSSCORRELATION C(J) OF TWO
C              TRANSIENTS X(I) AND Y(I) OF LENGTH LX AND LY RESPECTIVELY
C              BEGINNING WITH ANY LAG K
C
C                                LX
C                    C(J)  =  SUM ( X(I) * Y(I-J) )
C                               I=1
C
C                    FOR  J = K,...,K+LC-1
C
C              WHERE THE COMPUTATION IS MADE AS THOUGH X AND Y HAD ZEROS
C              EXTENDING BEYOND BOTH ENDS.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 134  REGISTERS
C SPEED      -
C AUTHOR     - R.A. WIGGINS
C
C                      ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CROSS, REVERS
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL CROST (LX,XX,LY,YY,K,LC,CC)
C
C INPUTS
C
C    LX        IS LENGTH OF XX.
C              IF LSTHN 1 NO COMPUTATION IS MADE.
C
C    XX(I)     I=1,...,LX CONTAINS X(1),...,X(LX) AS DESCRIBED IN THE
C                ABSTRACT.
C
C    LY        IS LENGTH OF YY.
C              IF LSTHN 1 NO COMPUTATION IS MADE.
C
C    YY(I)     I=1,...,LY CONTAINS Y(1),...,Y(LY) AS DESCRIBED IN THE
C                ABSTRACT.
C
C    K         IS THE INITIAL LAG.
C
C    LC        IS THE NUMBER OF LAGS WANTED.
C              IF LSTHN 1 NO COMPUTATION IS MADE.
C
C OUTPUTS
C
C    CC(I)     I=1,...,LC CONTAINS C(K),...,C(K+LC-1) AS DESCRIBED IN
C                THE ABSTRACT.
C
C EXAMPLES
C
C 1. INPUTS  - LX=3  XX(1...3) = 1.,2.,3.  LY=2  YY(1...2) = 2.,1.
C              LC=5  CC(1...5) = .1,.1,.1,.1,.1  K=0
C    OUTPUTS -       CC(1...5) = 4.,7.,6.,0.,0.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT K=2
C    OUTPUTS - CC(1...5) = 6.,0.,0.,0.,0.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT K=-2
C    OUTPUTS - CC(1...5) = 0.,1.,4.,7.,6.
C
C 4. INPUTS  - LX=2  XX(1...2) = 2.,1.  LY=3  YY(1...3) = 1.,2.,3.
```

****
*   CROST   *
****
(PAGE 2)

```
C               LC=5  CC(1...5) = .1,.1,.1,.1,.1  K=-2
C    OUTPUTS -        CC(1...5) =  6.,7.,4.,1.,0.
C
C 5. INPUTS  - SAME AS EXAMPLE 4. EXCEPT LX=0
C    OUTPUTS - CC(1...5) = 0.,0.,0.,0.,0.
C
C 6. INPUTS  - SAME AS EXAMPLE 4. EXCEPT LY=0
C    OUTPUTS - CC(1...5) = 0.,0.,0.,0.,0.
C
C 7. INPUTS  - SAME AS EXAMPLE 4. EXCEPT LC=0
C    OUTPUTS - CC(1...5) = .1,.1,.1,.1,.1
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION XX(2),YY(2),CC(2)
      I1=XMAXOF(1,-K-LC+1)
      LC1=XMINOF(LC,-K)
      CALL CROSS (LY-I1,YY(I1+1),LX,XX,LC1,CC)
      CALL REVERS(LC1,CC)
      I1=XMAXOF(0,K)+1
      I2=XMAXOF(1,LC1+1)
      CALL CROSS (LX-I1+1,XX(I1),LY,YY,LC-I2+1,CC(I2))
      RETURN
      END
```

```
*     CRSVM (SUBROUTINE)                 9/10/64   LAST CARD IN DECK IS NO. 0219
*     LABEL
CCRSVM
      SUBROUTINE CRSVM (NRAC,NCARB,NCBC,LAA,AA,LBB,BB,ZFNBTR,
     1 IFSTLG,LCC,CC)
C
C                       ----ABSTRACT----
C
C TITLE - CRSVM
C        CROSSCORRELATION OF TRANSIENT VECTORS OF MATRICES.
C
C               CRSVM FINDS  LCC  TERMS OF THE TRANSIENT CROSSCORRELATION
C               OF A VECTOR OF  NRAC X NCARB  MATRICES  A(I)  WITH A
C               VECTOR OF  NCARB X NCBC  MATRICES (AFTER TRANSPOSITION)
C               B(K)  BEGINNING WITH A FIRST LAG  IFSTLG
C
C                            INF
C                   C(J)  =  SUM   ( A(I)*B(I-J) )
C                          I=-INF
C               OR
C                            INF         ------
C                   C(J)  =  SUM   ( A(I)*B(I-J) )
C                          I=-INF
C
C               FOR   J = IFSTLG,IFSTLG+1,...,IFSTLG+LCC-1
C
C                                      ----
C               WHERE  INF = INFINITY,  B(I) = B(I) TRANSPOSE,  AND THE
C               ASSUMPTION IS MADE THAT THE VECTORS ARE ZERO BEYOND THE
C               RANGE OF DEFINITION.
C
C LANGUAGE   - FORTRAN II (SUBROUTINE)
C EQUIPMENT  - 709, 7090, OR 7094 (MAIN FRAME ONLY)
C STORAGE    - 327 REGISTERS
C SPEED      -    LET  MD3TIM(NRAC,NCARB,NCBC,LAA)  =  TIME FOR  MDOT3
C               TO FIND THE DOT PRODUCT OF  2  SERIES OF LENGTH  LAA.
C               THEN TIME FOR ONE LAG OF A CROSS CORRELATION IS
C                     (MD3TIM(NRAC,NCARB,NCBC,LAA)
C                    + .00085 - MD3TIM(1,1,1,1))   SECONDS
C               ON THE 7094 MOD 1 .
C                  FOR THE  3/63  VERSIONS OF  MDOT3  AND  MATML3  THIS
C               BECOMES
C                     (.000036*NRAC*NCARB*NCBC + .000170*NRAC*NCBC
C                    + .000040*NCBC + .000024) * LAA  +  .00010  SECONDS.
C               THUS THE TIME FOR HALF OF  MXLAGS  LAGS OF AN
C               AUTOCORRELATION OF A SERIES OF LENGTH  LAA
C               WILL BE ABOUT
C                     (.000036*NRAC*NCARB*NCBC + .000170*NRAC*NCBC
C                         + .000040*NCBC + .000024)
C                    * ((LAA*(LAA-MXLAGS))/2 + .00010) * MXLAGS  SECONDS.
C AUTHOR     - R.A. WIGGINS   AUGUST, 1964
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MDOT3, SETKS, STZ
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C     CALL CRSVM (NRAC,NCARB,NCBC,LAA,AA,LBB,BB,ZFNBTR,IFSTLG,LCC,CC)
C
C INPUTS
C
C    NRAC       NUMBER OF ROWS IN THE  AA  AND  CC  MATRICES.
C               MUST EXCEED ZERO.
C
C    NCARB      NUMBER OF COLUMNS IN THE  AA  MATRICES, NUMBER OF ROWS
C                 (AFTER TRANSPOSITION) IN THE  BB  MATRICES.
C               MUST EXCEED ZERO.
C
C    NCBC       NUMBER OF COLUMNS IN THE  BB  AND  CC  MATRICES.
C               MUST EXCEED ZERO.
C
C    LAA        NUMBER OF  NRAC X NCARB  MATRIX ELEMENTS IN THE VECTOR OF
C                 MATRICES  AA.
```

```
C              MUST EXCEED ZERO.
C
C    AA(I)     I=1,...,NRAC*NCARB*LAA  CONTAINS THE VECTOR OF MATRICES
C                A(1),...,A(LAA)  STORED CLOSELY SPACED BY COLUMNS.
C
C    LBB       NUMBER OF  NCARB X NCBC  MATRIX ELEMENTS IN THE VECTOR OF
C                MATRICES  BB.
C              MUST EXCEED ZERO.
C
C    BB(I)     I=1,...,NCARB*NCBC*LBB  CONTAINS THE VECTOR OF MATRICES
C                B(1),...,B(LBB)  STORED CLOSELY PACKED BY COLUMNS (IF
C                ZFNBTR=0.) OR BY ROWS (IF  ZFNBTR=1.).
C
C    ZFNBTR    =0. IMPLIES THAT THE MATRICES IN  BB(I)  ARE STORED BY
C                    COLUMNS.
C              =1. IMPLIES THAT THE MATRICES IN  BB(I)  ARE STORED BY
C                    ROWS.
C
C    IFSTLG    INDEX OF THE FIRST LAG OF THE CROSSCORRELATION.
C
C    LCC       NUMBER OF LAGS OF THE CROSSCORRELATION TO BE COMPUTED.
C              MUST EXCEED ZERO.
C
C OUTPUTS
C
C    STRAIGHT RETURN WITH NO COMPUTATIONS IF  NRAC, NCARB, NCBC,
C      LAA, LBB,  OR  LCC  LSTHN= 0.
C
C    CC(I)     I=1,...,NRAC*NCBC*LCC  CONTAINS THE CROSSCORRELATION
C                VECTOR OF MATRICES  C(IFSTLG),...,C(IFSTLG+LCC-1)  AS
C                DEFINED IN THE ABSTRACT STORED CLOSELY SPACED BY
C                COLUMNS.
C
C EXAMPLES
C
C 1. INPUTS  - NRAC=1  NCARB=2  NCBC=3
C              LAA=4  AA(1...8)=(1.,2.),(3.,-2.),(5.,-4.),(1.,-1.)
C              LBB=2  BB(1...12)=( 3., 4., 1.)  (-2.,-2., 4.)
C                                ( 2., 3.,-1.), (-3., 2.,-5.)
C              (NOTE THAT  BB  IS STORED AS                              )
C              ( BB(1...12)=3.,2.,4.,3.,1.,-1.,-2.,-3.,-2.,2.,4.,-5. )
C              ZFNBTR=0.  IFSTLG=-2  LCC=7
C    OUTPUTS - CC(1...21)= (0,0,0), (-8.,2.,-6.), (-7.,0.,21.),
C                (7.,-12.,45.), (8.,4.,18.), (1.,1.,2.), (0,0,0)
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  ZFNBTR=1.  -  THIS CAUSES
C              CRSVM TO SEE THE ARRAY  BB(I)  AS  3 X 2  MATRICES
C                     BB(1...12)=(3., 3.)  (-2., 2.)
C                                (2., 1.)  (-3., 4.)
C                                (4.,-1.), (-2.,-5.)
C    OUTPUTS - CC(1...21)= (0,0,0), (2.,5.,-12.), (-1.,-13.,6.),
C                (-15.,-27.,24.), (-1.,-1.,27.), (0.,1.,5.), (0,0,0)
C
C
C PROGRAM FOLLOWS BELOW
C
C
C
C DUMMY DIMENSION
C
      DIMENSION AA(2),BB(2),CC(2)
C
C BRING IN PARAMETERS AND SET SOME USEFUL COMBINATIONS
C
      CALL SETKS (NRAC,N, NCARB,M, NCBC,L, LAA,LA, LBB,LB,
     1 IFSTLG,K, LCC,LC, 1,ICC)
      NM=N*M
      NL=N*L
      ML=M*L
C
C LEAVE IF ANY VALUES ILLEGAL
C
      IF (XMINOF(N,M,L,LA,LB,LC))  100,100,10
 10   CONTINUE
C
```

CRSVM

(PAGE 3)

```
C CLEAR THE OUTPUT AREA
C
      CALL STZ (LC*NL,CC)
C
C IF NEGATIVE LAGS ARE SPECIFIED DO THESE FIRST
C
      IF (K)  20,50,50
 20   CONTINUE
C
C SET UP MDOT3 CONTROL PARAMETERS
C
C     IC    IS MATRIX INDEX -1 OF NEXT OUTPUT
C     ICC   IS VECTOR INDEX OF NEXT OUTPUT
C     LCM   IS MATRIX NO. OF PRODUCTS TO COMPUTE
C     IB    IS MATRIX INDEX -1 OF BB FOR NEXT PRODUCT
C     IBB   IS VECTOR INDEX OF BB FOR NEXT PRODUCT
C
      IC   = XMAXOF(0,-K-LB+1)
      ICC  = IC*NL+1
      LCM  = XMINOF(-K,LB-1,LC-IC)
      IB   = XMINOF(-K,LB-1)
      IBB  = IB*ML+1
C
C IF THERE ARE NO PRODUCTS, LEAVE
C
      IF (LCM)  100,100,30
 30   CONTINUE
C
C COMPUTE THE NEGATIVE LAGS
C
      DO 40 I=1,LCM
      CALL MDOT3 (N,M,L,XMINOF(LB-IB,LA),AA,BB(IBB),ZFNBTR,CC(ICC),1)
      IB=IB-1
      IBB=IBB-ML
 40   ICC=ICC+NL
C
C ADJUST  K  AND  LC  FOR POSITIVE LAG COMPUTATION
C
      LC=LC+K
      K=0
 50   CONTINUE
C
C SET UP MDOT3 CONTROL PARAMETERS
C
C     LCM   IS MATRIX NO. OF PRODUCTS TO COMPUTE
C     IA    IS MATRIX INDEX -1 OF AA FOR NEXT PRODUCT
C     IAA   IS VECTOR INDEX OF AA FOR NEXT PRODUCT
C     ICC   IS VECTOR INDEX OF NEXT OUTPUT (ALREADY SET)
C
      LCM  = XMINOF (LA-K,LC-K)
      IA   = K
      IAA  = IA*NM+1
C
C LEAVE IF THERE ARE NO PRODUCTS
C
      IF (LCM)  100,100,60
 60   CONTINUE
C
C COMPUTE THE POSITIVE LAGS
C
      DO 70 I=1,LCM
      CALL MDOT3 (N,M,L,XMINOF(LA-IA,LB),AA(IAA),BB,ZFNBTR,CC(ICC),1)
      IA=IA+1
      IAA=IAA+NM
 70   ICC=ICC+NL
C
C THAT'S ALL
C
 100  CONTINUE
      RETURN
      END
```

```
*      CSOUT (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0126
*      FAP
*CSOUT
       COUNT   150
       LBL     CSOUT
       ENTRY   CSOUT (ITAPE,NSPACE,C1,C1NAME,C2,C2NAME,... )
*
*
*                     ----ABSTRACT----
*
* TITLE - CSOUT
*       CONSTANTS OUTPUTTED IN FIXED FORMAT
*
*             CSOUT WRITES A LIST OF VARIABLES AND THEIR NAMES ON A
*             LOGICAL TAPE ACCORDING TO A FIXED FORMAT WITH INITIAL
*             SPACING (OR PAGE RESTORE).
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
* STORAGE     - 49 REGISTERS
* SPEED       -
* AUTHOR      - R.A. WIGGINS,  JULY, 1964
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - CARIGE, HRADJ
*      AND FORTRAN SYSTEM ROUTINES - (STH),(FIL)
*
* FORTRAN USAGE
*     CALL CSOUT (ITAPE,NSPACE,C1,C1NAME,C2,C2NAME, ... ,CN,CNNAME)
*
*
* INPUTS
*
*    ITAPE      IS LOGICAL TAPE NUMBER OF DESIRED OUTPUT TAPE.
*
*    NSPACE     IS DESIRED NUMBER (MAY BE ZERO) OF SPACES BEFORE ANY
*                 OUTPUT.  IF NEGATIVE AN INITIAL PAGE RESTORE OCCURS.
*
*    C1,C2,...,CN ARE THE FIXED OR FLOATING POINT VARIABLES TO BE
*                 PRINTED.
*
*    C1NAME,C2NAME,...,CNNAME ARE THE HOLLERITH NAMES OF C1,C2,...,CN
*                 RESPECTIVELY IN FORMAT(A6) OR (A5) OR ... (A1).
*
*
* OUTPUTS       1. NSPACE SPACES OR A PAGE RESTORE OCCURS
*               2. THE VARIABLES AND THEIR NAMES ARE WRITTEN AS THEY
*                    WOULD BE BY THE FORTRAN STATEMENTS
*
*                     WRITE OUTPUT TAPE ITAPE, 10, C1NAMR,C1,C2NAMR,C2,
*                    1 ..., CNNAMR,CN
*                10   FORMAT(5(2XA6,3H = G14.7))
*
*                    WHERE  CNNAMR = HRADJF(CNNAME).
*
*
* EXAMPLES
*
* 1. INPUTS   - C1=1.  C1NAME=3HONE  C2=2  C2NAME=3HTWO
*               ITAPE = 2  NSPACE = 2
*    USAGE    -       CALL CSOUT (ITAPE,NSPACE,C1,C1NAME,C2,C2NAME)
*    OUTPUTS - THE FOLLOWING 3 LINES
*
*
*                    ONE =  1.0000000          TWO =  2
*               WILL BE PRINTED OFF LINE FROM LOGICAL TAPE 2 (UNDER
*                 PROGRAM CONTROL).
*
* 2. EXAMPLE WITH LITERAL ARGUMENTS.  THE LAST ARGUMENT IS IGNORED SINCE
*      IT HAS NO NAME.
*    USAGE    -       CALL CSOUT(2,1,.01,1HX,5)
*    OUTPUTS - THE FOLLOWING 2 LINES
*
```

```
*                      X =  0.1000000E-01
*              WILL BE PRINTED OFF LINE FROM LOGICAL TAPE 2 (UNDER
*                PROGRAM CONTROL).
*
*
* PROGRAM FOLLOWS BELOW
*
XR4    HTR     0               STORAGE REGISTER FOR IR 4
       BCI     1,CSOUT
CSOUT  SXD     XR4,4           SAVE IR4
       CLA     1,4             AND
       STA     ITAPE           USE
       CLA     2,4             CARIGE
       STA     NSPACE          TO
       TSX     $CARIGE,4      *MAKE THE
ITAPE  TSX     **,0            INITIAL
NSPACE TSX     **,0            SPACES.
       LXD     XR4,4           RESET IR 4
       CLA*    1,4             AND
       TSX     $(STH),4       *GO INITIALIZE
       PZE     FORMAT,,1       (STH). (1 IN DECREMENT INDICATES THAT
       LXD     XR4,4           RESET IR 4       IS STORED IN THE REVERSE
       TRA     LKAHD           AND GO CHECK.    OF THE NORMAL ORDER)
LOOP   CLA*    2,4             GET NEXT NAME
       SXD     XR4,4           AND
       TSX     $HRADJ,4        RIGHT ADJUST IT.
       LXD     XR4,4
       XCA                     PUT IN MQ
       STR                    *AND FEED IT TO (IOH).
       LDQ*    1,4             GET NEXT VARIABLE
       STR                     AND FEED IT TO (IOH).
LKAHD  CAL     3,4             THEN CHECK IF
       ANA     =0777777700000  NEXT ARGUMENT (VARIABLE)
       LAS     TSX             IS TSX  ,0
       TRA     *+2             IT IS NOT
       TRA     A1              IT IS, GO CHECK NEXT ARGUMENT.
EXIT   SXD     XR4,4           IT IS NOT, PREPARE TO LEAVE
       TSX     $(FIL),4       *GO ROUND-OUT (STH).
       LXD     XR4,4           AND
       TRA     3,4            *RETURN
A1     CAL     4,4             CHECK NEXT ARGUMENT (NAME)
       ANA     =0777777700000  FOR
       LAS     TSX             TSX ,0 FORM.
       TRA     *+2             IT IS NOT.
       TIX     LOOP,4,2        IT IS, BUMP IR4 AND GO WRITE
       TIX     EXIT,4,1        IT IS NOT, BUMP IR4 AND PREPARE TO RETURN.
*
* CONSTANTS
*
TSX    TSX     0,0
FORMAT BCI     4,(5(2XA6,3H = G14.7))
       END
```

```
*     CUFIT1 (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0325
*     FAP
*CUFIT1
      COUNT   300
      LBL     CUFIT1
      ENTRY   CUFIT1 (FOFX,XLO,DELX,COEFS)
*
*
*                  ----ABSTRACT----
*
* TITLE - CUFIT1
*       FIND CUBIC WHICH EXACTLY FITS 4 EQUALLY SPACED POINTS
*
*             CUFIT1 FINDS C0,C1,C2, AND C3 SUCH THAT THE CUBIC
*             POLYNOMIAL
*                                           2       3
*                      F(X)= C0 + C1*X + C2*X  + C3*X
*
*             TAKES ON SPECIFIED VALUES AT 4 EQUALLY SPACED VALUES
*             OF X, NAMELY AT XLO, XLO+DELX, XLO+2*DELX AND XLO+3*DELX,
*             WHERE XLO AND DELX ARE PARAMETERS.
*
*             CUFIT1 HAS TWO AUTOMATIC HI SPEED BYPASSES, ONE EFFECTIVE
*             IN CASES WHERE XLO=-3 AND DELX=+2, THE OTHER APPLYING
*             TO REPEATED CALLS OF CUFIT1 WITH IDENTICAL VALUES OF XLO
*             AND DELX.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 158 REGISTERS
* SPEED       -                            ON 709    ,    ON 7090
*             1. GENERAL.                614MC=7.37MS,  545MC=1.19MS
*             2. REPEAT CALL IN GENERAL
*                WITH SAME  XLO,  DELX.  496MC=5.95MS,  445MC=.970MS
*             3. CASE IN WHICH  XLO = -3
*                AND  DELX = 2 .         290MC=3.48MS,  268MC=.584MS
*
*             WHERE  MC = MACHINE CYCLES,  MS = MILLISECONDS.
*
* AUTHOR     - S.M. SIMPSON, MARCH 1964
*
*
*                  ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*      AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL CUFIT1(FOFX,XLO,DELX,COEFS)
*
*
* INPUTS
*
*    FOFX(I)   I=1...4 CONTAINS THE VALUES THAT THE POLYNOMIAL
*                MUST ASSUME, AS DETAILED BELOW
*
*    XLO       IS DEFINED IN THE ABSTRACT
*
*    DELX      IS DEFINED IN THE ABSTRACT.   DELX SHOULD NOT BE ZERO
*                BUT MAY BE NEGATIVE.
*
*
* OUTPUTS    STRAIGHT RETURN WITH NO OUTPUTS IF DELX=0.
*
*    COEFS(I)  I=1...4  WILL CONTAIN C0,C1,C2,C3 DETERMINED SO THAT
*                THE POLYNOMIAL F(X) GIVEN IN THE ABSTRACT WILL SATISFY
*                    F(XLO)         = FOFX(1)
*                    F(XLO+DELX)    = FOFX(2)
*                    F(XLO+2*DELX)  = FOFX(3)
*                    F(XLO+3*DELX)  = FOFX(4)
*
*
* EXAMPLES
*
* 1. INPUTS  - FOFX(1...4) = 2.,4.,10.,26.  XLO=-1.  DELX=1.0
```

```
*    USAGE   -        CALL CUFIT1(FOFX,XLO,DELX,COEFS1)
*                     CALL CUFIT1(FOFX,XLO,DELX,COEFS2)
*    OUTPUTS - COEFS1(1...4) = COEFS2(1...4) = 4.0,3.0,2.0,1.0
*
* 2. INPUTS  - FOFX(1...4) = -14.,2.,10.,58.  XLO=-3.  DELX=2.0
*              COEFS4(1...4) = -99.,-99.,-99.,-99.
*    USAGE   -        CALL CUFIT1(FOFX,XLO,DELX,COEFS3)
*                     CALL CUFIT1(FOFX,XLO,0.0,COEFS4)
*    OUTPUTS - COEFS3(1...4) = 4.0,3.0,2.0,1.0
*              COEFS4(1...4) = -99.,-99.,-99.,-99.
*
* 3. INPUTS  - FOFX(1...4) = 2.0,3.0,4.0,5.0  XLO=2.0  DELX=1.0
*    USAGE   -        CALL CUFIT1(FOFX,XLO,DELX,COEFS5)
*    OUTPUTS - COEFS5(1...4) = 0.0,1.0,0.0,0.0
*
*
* PROGRAM FOLLOWS BELOW
*
       HTR     0              XR1
       HTR     0              XR4
       BCI     1,CUFIT1
*
* ONLY ENTRY.  CUFIT1(FOFX,XLO,DELX,COEFS)
*
CUFIT1 SXD     CUFIT1-2,4
       SXD     CUFIT1-3,1
*
* EXIT ON ZERO DELX
*
       NZT*    3,4            DELX
       TRA     LEAVE
*
* BRING IN FM3,FM1,F1,F3 AND SET ADDRESSES
*
       CLA     1,4            A(FOFX(1))
       ADD     K1
       STA     CLAF
       AXT     4,1
 CLAF  CLA     **,1           **=A(FOFX(1))+1
       STO     FM3+1,1
       TIX     CLAF,1,1
       CLA     4,4            A(COEFS(1))
       STA     STOCZ
       SUB     K1
       STA     STOC1
       SUB     K1
       STA     STOC2
       SUB     K1
       STA     STOC3
*
* (SO FAR IT HAS TAKEN ABOUT 29 HI SPEEDS)
*
*
* SET TRIAL VALUES OF C0, C1, C2, C3 (FOR XLO=-3.0, DELX=2.0)
*    (C3)       1    (-1  +3  -3  +1) (FM3)
*    (C2) =   ----   (+3  -3  -3  +3) (FM1)
*    (C1)      48    (+1 -27 +27  -1) (F1)
*    (C0)            (-3 +27 +27  -3) (F3)
*
* FIRST C3
*
       CLA     FM1
       FSB     F1
       XCA
       FMP     K3L
       FSB     FM3
       FAD     F3
       XCA
       FMP     R48
 STOC3 STO     **             **=A(COEFS)-3
       STO     C3
*
* THEN C2
*
       CLA     FM3
```

```
         FSB     FM1
         FSB     F1
         FAD     F3
         XCA
         FMP     R16
   STOC2 STO     **           **=A(COEFS)-2
         STO     C2
*
* THEN C1
*
         CLA     F1
         FSB     FM1
         XCA
         FMP     K27L
         FAD     FM3
         FSB     F3
         XCA
         FMP     R48
   STOC1 STO     **           **=A(COEFS)-1
         STO     C1
*
* THEN C0
*
         CLA     FM1
         FAD     F1
         XCA
         FMP     K9L
         FSB     FM3
         FSB     F3
         XCA
         FMP     R16
   STOCZ STO     **           **=A(COEFS)
         STO     CZ
*
* (SETTING THE ABOVE TAKES 19 HI SPEEDS, 12 FADS, 7 FMPS)
*
* NOW WE ARE ALL DONE IN THE CASE THAT
*        XLO = -3.0 AND DELX = 2.0
*
         CLS*    2,4          -XLO
         CAS     K3L
         TRA     CKJUMP       NO
         TRA     *+2          MAYBE
         TRA     CKJUMP       NO
         CLA*    3,4          DELX
         CAS     K2L
         TRA     CKJUMP       NO
         TRA     LEAVE        EXIT
*
* (IF XLO=-3 AND DELX=2, THE CHECK TAKES 7 HI SPEEDS, OTHERWISE
*     AVERAGE = 3)
*
* OTHERWISE JUMP AHEAD IN THE CASE THAT
*    XLO AND DELX ARE BOTH THE SAME AS LAST CALL.
*
CKJUMP   CLA*    3,4          DELX IN AC
         LDQ*    2,4          XLO IN MQ
         CAS     LASDEL           FIRST CHECK DELX
         TRA     NEW          NEW
         TRA     *+2          MAYBE OLD
         TRA     NEW          NEW
         XCA                  CHECK XLO IF MAYBE
         CAS     LASXLO
         TRA     *+2          NEW
         TRA     REVISE       JUMP AHEAD
         XCA                  NEW, RESTORE AC, MQ
*
* STORE THE NEW XLO AND DELX
*
 NEW     STO     LASDEL
         STQ     LASXLO
*
* (TAKES 8 HI SPEEDS IF JUMP, 7 IF NOT)
*
* IN THE GENERAL CASE WE HAVE TO SET THE CONSTANTS
```

```
*    K, KSQUAR, KCUBE, G, 2G, 3G, 3GSQR
*         WHERE  K = 2/DELX
*                G = -K*(XLO+3/K) = -K*XLO-3
*
       CLA     K2L
       FDP     LASDEL
       STQ     K
       FMP     K
       STO     KSQUAR
       XCA
       FMP     K
       STO     KCUBE
       CLS     K
       XCA
       FMP     LASXLO
       FSB     K3L
       STO     G
       FAD     G
       STO     TWG
       FAD     G
       STO     THG
       XCA
       FMP     G
       STO     THGSQR
*
* (THESE SETTINGS TAKE 12 HI SPEEDS, 1 FDP, 4 FMPS, 3 FADS)
*
* COMPUTE AND STORE THE REVISED COEFFICIENTS AS FOLLOWS
*         (C3)      (K**3          0          0  0) (C3)
*         (C2)      (3(K**2)G      K**2       0  0) (C2)
*         (C1)  =   (3K(G**2)      2KG        K  0) (C1)
*         (C0)      (G**3          G**2       G  1) (C0)
*
REVISE LDQ     C3
       FMP     KCUBE
       STO*    STOC3
       LDQ     C3            C2=(K**2)*((3G)*C3+C2)
       FMP     THG
       FAD     C2
       XCA
       FMP     KSQUAR
       STO*    STOC2
       LDQ     C3            C1=K*((3G**2)*C3+2G*C2+C1)
       FMP     THGSQR
       STO     TEMP
       LDQ     C2
       FMP     TWG
       FAD     TEMP
       FAD     C1
       XCA
       FMP     K
       STO*    STOC1
       LDQ     C3            CZ=CZ+G(C1+G(C2+G*C3))
       FMP     G
       FAD     C2
       XCA
       FMP     G
       FAD     C1
       XCA
       FMP     G
       FAD     CZ
       STO*    STOCZ
*
* (REVISEON TAKES 16 HI SPEEDS, 9 FMPS, 6FADS)
*
* EXIT
*
 LEAVE LXD     CUFIT1-3,1
       TRA     5,4
*
* CONSTANTS
*
 K1    PZE     1
 K2L   DEC     2.0
 K3L   DEC     3.0
```

```
 K9L    DEC     9.0
 K27L   DEC     27.0
 R16    DEC     .0625          =1/16
 R48    DEC     .020833333     =1/48
*
* VARIABLES
*
 F3     PZE     **,**,**       NOTE -
 F1     PZE     **,**,**       ORDER
 FM1    PZE     **,**,**       OF F SEQUENCE
 FM3    PZE     **,**,**       IS IMPORTANT
LASXLO  PZE     **,**,**
LASDEL  PZE     **,**,**
 KCUBE  PZE     **,**,**       K**3
KSQUAR  PZE     **,**,**       K**2
 K      PZE     **,**,**       2/DELX
THGSQR  PZE     **,**,**       3*(G**2)
 THG    PZE     **,**,**       3*G
 TWG    PZE     **,**,**       2*G
 G      PZE     **,**,**
 CZ     PZE     **,**,**
 C1     PZE     **,**,**
 C2     PZE     **,**,**
 C3     PZE     **,**,**
 TEMP   PZE     **,**,**
        END
```

```
*      CVSOUT (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0220
*      FAP
*CVSOUT
       COUNT   200
       LBL     CVSOUT
       ENTRY   CVSOUT (ITAPE,NSPACE,FMTHED,FMTLIN,ILO,IHI,ARGLO,ARGDEL,
*                      SPACE,X1,X2,...,XN)
*
*                      ----ABSTRACT----
*
* TITLE - CVSOUT
*        OUTPUT COLUMN VECTORS BY NORMAL OR LITERAL FORMATS
*
*              CVSOUT IS A VARIABLE-LENGTH-CALLING-SEQUENCE PROGRAM
*              WHICH OUTPUTS AN ARBITRARY NO. OF VECTORS IN COLUMN
*              FASHION ONTO A SPECIFIED TAPE UNIT.  IT PROVIDES A
*              LEFTMOST COLUMN WITH VALUES INCREMENTED BY A SPECIFIED
*              AMOUNT FROM A SPECIFIED BASE.  USER SUPPLIES HEADING
*              FORMAT AND LINE FORMAT AS EITHER NORMAL FORMAT VECTORS
*              OR LITERAL ONES.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
* STORAGE     - 84 REGISTERS
* SPEED       -
* AUTHOR      - S.M. SIMPSON JR,   SEPTEMBER 1963
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - CARIGE,FMTOUT,VECOUT
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL CVSOUT(ITAPE,NSPACE,FMTHED,FMTLIN,ILO,IHI,ARGLO,ARGDEL,
*    1            SPACE,X1,X2,...,XN)
*
* INPUTS       DEFINE A NORMLIT FORMAT VECTOR AS EITHER
*                 A) A NORMAL FORMAT VECTOR,
*              OR B) LITERAL HOLLERITH IN A CALLING SEQUENCE WHOSE
*                 CHARACTERS (READING CONTINUOUSLY FROM LEFT TO RIGHT)
*                 ARE THE DESIRED FORMAT STRIPPED OF THE ENCLOSING
*                 PARENTHESES.  THE FIRST AND SECOND CHARACTERS MUST
*                 NOT BE QUOTE ( UNQUOTE OR QUOTE ) UNQUOTE
*                 RESPECTIVELY.  (TWO BLANKS FOLLOWED BY ( WOULD BE OK.)
*
*    ITAPE     IS DESIRED LOGICAL TAPE NUMBER
*
*    NSPACE    IS DESIRED NO. OF INITIAL SPACES (MAY BE ZERO) BEFORE
*                 ANY PRINTING.
*              IF NEGATIVE A PAGE RESTORE OCCURS BEFORE PRINTING.
*
*    FMTHED(I) I=1,2,... OR I=1,0,-1,... IS A NORMLIT FORMAT VECTOR TO
*                 BE PRINTED AS A HEADING FOR THE COLUMNS.
*
*    FMTLIN(I) I=1,2,... OR I=1,0,-1,... IS A NORMLIT FORMAT VECTOR
*                 GIVING THE PRINTING FORMAT FOR A SINGLE LINE OF
*                 OUTPUT.  THE LIST OF QUANTITIES PRINTED ON A LINE IS
*                 ARG(L),X1(I),X2(I),...,XN(I) WHERE ARG(L) IS DEFINED
*                 BELOW.  FMTLIN MUST INCLUDE THE (FLTG) FORMAT FOR
*                 ARG(L) AS WELL AS FOR THE X VECTORS.
*
*    ILO       IS FIRST SUBSCRIPT OF VECTOR RANGE TO BE PRINTED.
*              MUST EXCEED 0 (NOT CHECKED).
*
*    IHI       IS LAST SUBSCRIPT OF VECTOR RANGE TO BE PRINTED.
*              MUST BE GRTHN= ILO (NOT CHECKED).
*
*    ARGLO     INITIAL VALUE OF QUANTITY ARG(L) TO APPEAR IN LEFTMOST
*                 COLUMN.
*              MUST BE FLOATING POINT.
*
*    ARGDEL    INCREMENT FOR ARG, FLOATING POINT.
*                 ARG(L)=ARGLO+(L-1)*ARGDEL WHERE L = LINE INDEX.
*
```

```
*    SPACE(I)  I=1...N+1 IS SCRATCH AREA WHERE N = NO. OF VECTORS.
*
*    X1(I)     I=ILO...IHI  IS FIRST VECTOR, ANY MODE
*    X2(I)     I=ILO...IHI  IS SECOND VECTOR, ANY MODE
*      ETC
*    XN(I)     I=ILO...IHI  IS N-TH VECTOR, ANY MODE.  N MUST EXCEED
*                 ZERO.
*
* OUTPUTS      THE VECTORS ARE PRINTED COLUMNWISE AS ILLUSTRATED BELOW.
*
*
* EXAMPLES
*
* 1. USING NORMAL FORMATS
*    INPUTS  - X(1...10) = 1.1,2.2,...,10.10   IX1(1...10) = 1,2,...,10
*              IX2(1...10) = 2,3,...,11   IX3(1...10) = 3,4,...,12
*              FMTH(1...6) = 34H(26H  ARGX    X    IX1 IX2 IX3,//)
*              FMTL(1...3) = 18H(F6.2,F6.1,2X,3I4)
*    USAGE   -       DIMENSION SPACE(5)
*                    CALL CVSOUT(2,3,FMTH,FMTL,4,10,-.03,.01,SPACE,X,
*                   1   IX1,IX2,IX3)
*    OUTPUTS - THE FOLLOWING 12 LINES
*
*
*
*               ARGX    X    IX1 IX2 IX3
*
*              -0.03   4.4     4   5   6
*              -0.02   5.5     5   6   7
*              -0.01   6.6     6   7   8
*               0.00   7.7     7   8   9
*               0.01   8.8     8   9  10
*               0.02   9.9     9  10  11
*               0.03  10.1    10  11  12
*              WILL BE PRINTED OFF-LINE FROM LOGICAL 2 (UNDER PROGRAM
*                                                           CONTROL)
*
* 2. USING LITERAL FORMATS
*    INPUTS  - X,IX1,IX2,IX3  SAME AS IN EXAMPLE 1.
*    USAGE   -       CALL CVSOUT(2,3,32H26H  ARGX    X    IX1 IX2 IX3,
*                   1 //,16HF6.2,F6.1,2X,3I4,4,10,-.03,.01,SPACE,X,
*                   2 IX1,IX2,IX3)
*    OUTPUTS - IDENTICAL TO THOSE OF EXAMPLE 1.
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS CARIGE, FMTOUT, VECOUT
       HTR     0             XR1
       HTR     0             XR2
       HTR     0             XR4
       BCI     1,CVSOUT
* ONLY ENTRY.  CVSOUT(ITAPE,NSPACE,FMTHED,FMTLIN,ILO,IHI,ARGLO,ARGDEL,
*              SPACE,X1,X2,...,XN)
CVSOUT SXD     CVSOUT-2,4
       SXD     CVSOUT-3,2
       SXD     CVSOUT-4,1
 K1    CLA     1,4           A(ITAPE)
       STA     C1
       STA     F1
       STA     V1
       CLA     2,4           A(NSPACE)
       STA     C2
       CLA     3,4           A(FMTHED)
       STA     F2
       CLA     4,4           A(FMTLIN)
       STA     V2
       CLA     9,4           A(SPACE)
       STA     V3
       STA     STO
* SET UP LOOP CONTROLS
       CLA*    6,4           IHI
       STD     TXL2          TO LOOP CONTROL.
       CLA*    7,4           ARGLO
       STO*    9,4           TO SPACE(1).
```

```
       CLA*    5,4           ILO
       PDX     0,2           TO XR2.
       CLA     8,4           A(ARGDEL)
       STA     FAD
* OPERATE THE CARRIAGE
       TSX     $CARIGE,4
 C1    TSX     **,0          **=A(ITAPE)
 C2    TSX     **,0          **=A(NSPACE)
* AND PRINT THE HEADING
       TSX     $FMTOUT,4
 F1    TSX     **,0          **=A(ITAPE)
 F2    TSX     **,0          **=A(FMTHED)
* THEN COUNT THE VECTORS
       LXD     CVSOUT-2,4
       AXT     0,1
       TXI     *+1,4,-9
       SXA     NXTLIN,4      (SAVE FOR INITIALIZING LOOP)
 CAL   CAL     1,4           TSX X1,0   TSX X2,0...
       ANA     AMASK
       LAS     TSXZ
       TRA     *+2           DONE
       TRA     *+2           MORE
       TRA     COVER         DONE
       TXI     *+1,1,1
       TXI     CAL,4,-1
* FINISHED
 COVER SXD     TXL1,1        STORE N,
       TXI     *+1,1,1
       SXD     NP1,1         AND N+1.
       SXA     LEAVE,4       (SAVE FOR EXITING TO 1,4)
* SET NEXT LINE OF OUTPUT IN SPACE(1...N+1)
*
*    XR4 ACQUIRES VECTOR ADDRESSES
*    XR2 ACQUIRES VECTOR ELEMENTS (ILO TO IHI)
*    XR1 STORES IN SPACE VECTOR (2...N+1)
NXTLIN AXT     **,4          (1,4 IS THEN TSX X1,0)
       AXT     1,1
* START LOOP
 CLA   CLA     1,4
       ADD     K1            TSX XK+1,0
       STA     *+1
       CLA     **,2          **=A(XK)+1
 STO   STO     **,1          **=A(SPACE)
       TXI     *+1,4,-1
       TXI     *+1,1,1
 TXL1  TXL     CLA,1,**      **=N
* GO OUTPUT ONE LINE
       TSX     $VECOUT,4
 V1    TSX     **,0          **=A(ITAPE)
 V2    TSX     **,0          **=A(FMTLIN)
 V3    TSX     **,0          **=A(SPACE)
       TSX     KD1,0         1
       TSX     NP1,0         TO N+1
* CHECK FOR MORE AFTER INCREMENTING SPACE(1)
       CLA*    V3
 FAD   FAD     **            **=A(ARGDEL)
       TNZ     *+2
       SSP
       STO*    V3
       TXI     *+1,2,1
 TXL2  TXL     NXTLIN,2,**   **=IHI
* EXIT
 LEAVE AXT     **,4
       LXD     CVSOUT-3,2
       LXD     CVSOUT-4,1
       TRA     1,4
* CONSTANTS, TEMPORARIES
 KD1   PZE     0,0,1
 AMASK OCT     777777700000
 TSXZ  TSX     0,0
 NP1   PZE     0,0,**        **=NO. OF VECTORS + 1
       END
```

```
*     DADECK (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0069
*     LABEL
CDADECK
      SUBROUTINE DADECK (ITPIN,ITPOUT)
C
C
C                     ----ABSTRACT----
C
C TITLE - DADECK
C        LIST DATA DECK AND REPOSITION TAPE TO FRONT OF DECK
C
C              DADECK LISTS THE DATA ON TAPE ITPIN FROM THE PLACE WHERE
C              THE TAPE IS POSITIONED WHEN DADECK IS CALLED TO THE END
C              OF FILE.  THE TAPE IS REPOSITIONED BEFORE RETURN IS MADE.
C              ALL 80 COLUMNS OF A CARD WILL BE LISTED.  THE OUTPUT
C              FORMAT SPACES ONE COLUMN TO THE RIGHT SO THAT CHARACTERS
C              IN COLUMN ONE WILL NOT AFFECT THE CARRIAGE CONTROL.  THE
C              LISTING IS MADE ON ITPOUT.
C
C              DADECK MAKES NO COMMENTS AND DOES NOT RESTORE THE
C              CARRIAGE.  IF THERE IS NO DATA ON ITPIN, THERE WILL BE NO
C              OUTPUT AT ALL FROM DADECK.
C
C LANGUAGE   - FORTRAN II
C EQUIPMENT  - 709/7090/7094 (MAIN FRAME AND TAPE DRIVES)
C STORAGE    - 100 REGISTERS
C SPEED      - PROPORTIONAL TO NO. OF DATA CARDS
C AUTHOR     - J. N. GALBRAITH, JR.  AND  R. A. WIGGINS
C
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - EOFSET,RSKIP
C       AND FORTRAN SYSTEM ROUTINES - (TSH),(RTN),(STH),(FIL)
C
C FORTRAN USAGE
C     CALL DADECK(ITPIN,ITPOUT)
C
C
C INPUTS
C
C    ITPIN     FORTRAN II INTEGER. LOGICAL TAPE NUMBER OF INPUT TAPE
C              (TAPE CONTAINING DATA DECK).
C
C    ITPOUT    FORTRAN II INTEGER. LOGICAL TAPE NUMBER OF OUTPUT TAPE
C              (TAPE ON WHICH DATA DECK WILL BE WRITTEN).
C
C
C OUTPUTS
C
C    PRINTED AS DESCRIBED ABOVE.
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION DATA(14)
      INUM=0
      CALL EOFSET(0,EOF,ITAPE)
      IF (EOF)  40,10,40
 10   CONTINUE
      READ INPUT TAPE ITPIN,20,(DATA(I),I=1,14)
 20   FORMAT(13A6,A2)
      WRITE OUTPUT TAPE ITPOUT,30,(DATA(I),I=1,14)
  30  FORMAT(1X13A6,A2)
      INUM=INUM+1
      GO TO 10
 40   CALL RSKIP(ITPIN,-INUM-1,EOF)
      CALL EOFSET (-1,EOF,ITAPE)
      RETURN
      END
```

```
*       DELTA (FUNCTIONS)                    9/4/64   LAST CARD IN DECK IS NO. 0140
*       FAP
*DELTA
        COUNT   75
        LBL     DELTA
        ENTRY   DELTA  F(ARG)
        ENTRY   XDELTA F(ARG)
        ENTRY   STEPR  F(ARG)
        ENTRY   XSTEPR F(ARG)
        ENTRY   STEPL  F(ARG)
        ENTRY   XSTEPL F(ARG)
        ENTRY   STEPC  F(ARG)
        ENTRY   XSTEPC F(ARG)
*
*
*                      ----ABSTRACT----
*
* TITLE - DELTA, WITH SECONDARY ENTRIES XDELTA, STEPR, XSTEPR, STEPL,
*                                       XSTEPL, STEPC, XSTEPC
*        DELTA FUNCTION AND STEP FUNCTIONS, FLOATING AND FIXED POINT
*
*              DELTA HAS VALUE EQUAL TO PLUS ZERO UNLESS THE MAGNITUDE
*              OF ITS ARGUMENT (WHICH MAY BE EITHER FIXED OR FLOATING
*              POINT) IS ZERO, IN WHICH CASE DELTA HAS VALUE EQUAL TO
*              1.0 (FLOATING).
*
*              XDELTA IS IDENTICAL TO DELTA EXCEPT THAT IT GIVES A
*              FIXED POINT OUTPUT.
*
*              STEPR HAS VALUE EQUAL TO PLUS ZERO UNLESS THE VALUE OF
*              ITS ARGUMENT (EITHER FIXED OR FLOATING POINT) EXCEEDS
*              ZERO, IN WHICH CASE STEPR HAS VALUE EQUAL 1.0 (FLOATING).
*
*              XSTEPR IS IDENTICAL TO STEPR EXCEPT THAT IT GIVES A
*              FIXED POINT OUTPUT.
*
*              STEPL HAS VALUE EQUAL TO PLUS 1.0 UNLESS THE VALUE OF
*              ITS ARGUMENT (EITHER FIXED OR FLOAITNG POINT) IS LESS
*              THAN ZERO, IN WHICH CASE STEPL HAS VALUE EQUAL 0.0 (FLTG)
*
*              XSTEPL IS IDENTICAL TO STEPL EXCEPT THAT IT GIVES A
*              FIXED POINT OUTPUT.
*
*              STEPC HAS VALUE EQUAL TO ZERO WHENEVER THE SIGN BIT OF
*              ITS ARGUMENT IS NEGATIVE.  OTHERWISE STEPC HAS VALUE =
*              1.0 (FLTG).
*
*              XSTEPC IS IDENTICAL TO STEPC EXCEPT THAT IT GIVES A
*              FIXED POINT OUTPUT.
*
* LANGUAGE   - FAP FUNCTIONS (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR  7090 (MAIN FRAME ONLY)
* STORAGE    - 17 REGISTERS
* SPEED      - 6, 8, OR 10 MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, APRIL 1964
*
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGES
*     Y = DELTAF(X)     OR      DELTAF(IX)
*     Y = STEPRF(X)     OR      STEPRF(IX)
*     Y = STEPLF(X)     OR      STEPLF(IX)
*     Y = STEPCF(X)     OR      STEPCF(IX)
*     IY=XDELTAF(X)     OR     XDELTAF(IX)
*     IY=XSTEPRF(X)     OR     XSTEPRF(IX)
*     IY=XSTEPLF(X)     OR     XSTEPLF(IX)
*     IY=XSTEPCF(X)     OR     XSTEPCF(IX)
*
*
* INPUTS
*
```

```
*    X          IS ANY FLOATING POINT NO.
*
*    IX         IS ANY FIXED POINT NO.
*
*
* OUTPUTS
*
*    Y          AS DESCRIBED IN ABSTRACT
*
*    IY         AS DESCRIBED IN ABSTRACT
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...6) = -2., -1., -0., 0., 1., 2.
*              IX(1...6) = -2, -1, -0, 0, 1, 2
*    USAGES  -       DO 10 I=1,6
*                    D1(I)   =  DELTAF( X(I))
*                    D2(I)   =  DELTAF(IX(I))
*                    ID1(I)  = XDELTAF( X(I))
*                    ID2(I)  = XDELTAF(IX(I))
*                    SR1(I)  =  STEPRF( X(I))
*                    SR2(I)  =  STEPRF(IX(I))
*                    ISR1(I) = XSTEPRF( X(I))
*                    ISR2(I) = XSTEPRF(IX(I))
*                    SL1(I)  =  STEPLF( X(I))
*                    SL2(I)  =  STEPLF(IX(I))
*                    ISL1(I) = XSTEPLF( X(I))
*                    ISL2(I) = XSTEPLF(IX(I))
*                    SC1(I)  =  STEPCF(X(I))
*                    SC2(I)  =  STEPCF(IX(I))
*                    ISC1(I) = XSTEPCF(X(I))
*                 10 ISC2(I) = XSTEPCF(IX(I))
*    OUTPUTS - D1(1...6) = D2(1...6) = 0., 0., 1., 1., 0., 0.
*              ID1(1...6) = ID2(1...6) = 0, 0, 1, 1, 0, 0
*              SR1(1...6) = SR2(1...6) = 0., 0., 0., 0., 1., 1.
*              ISR1(1...6) = SR2(1...6) = 0, 0, 0, 0, 1, 1
*              SL1(1...6) = SL2(1...6) = 0., 0., 1., 1., 1., 1.
*              ISL1(1...6) = ISL2(1...6) = 0, 0, 1, 1, 1, 1
*              SC1(1...6) = SC2(1...6) = 0., 0., 0., 1., 1., 1.
*              ISC1(1...6) = ISC2(1...6) = 0, 0, 0, 1, 1, 1
*
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
*
       BCI     1,DELTA
 DELTA TZE     GET1L          FIRST ENTRY
XDELTA TZE     GET1           SECOND ENTRY
 GETZ  PXD     0,0
       TRA     1,4
 STEPL TZE     GET1L          ANOTHER
 STEPR TZE     GETZ           ANOTHER
 STEPC TMI     GETZ           ANOTHER
 GET1L CLA     K1L
       TRA     1,4
XSTEPL TZE     GET1           ANOTHER
XSTEPR TZE     GETZ           ANOTHER
XSTEPC TMI     GETZ           ANOTHER
 GET1  CLA     KD1
       TRA     1,4
 KD1   PZE     0,0,1
 K1L   DEC     1.0
       END
```

```
*************************
*    DETRM              *
*************************
 REFER TO
     SIMEQ
```


```
*************************
*    DETRM              *
*************************
 REFER TO
     SIMEQ
```

```
*       DERIVA (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0159
*       FAP
*DERIVA
        COUNT   150
        LBL     DERIVA
        ENTRY   DERIVA (YOFX,LY,DELX,DYDX,YOFX1)
*
*                     ----ABSTRACT----
*
* TITLE - DERIVA
*       DERIVATIVE OF A VECTOR BY DIFFERENCING
*
*               DERIVA FORMS A VECTOR, DYDX(I) I=1...2Y , REPRESENTING
*               THE DERIVATIVE OF ANOTHER VECTOR, YOFX(I) I=1...LY ,
*               FROM THE DIFFERENCING FORMULAS
*
*                  DYDX(1)  =  (YOFX(2) - YOFX(1))/DELX
*
*                  DYDX(K)  =  (YOFX(K+1) - YOFX(K-1)/(2.0*DELX)
*                                  FOR K = 2,3,...,LY-1
*                  DYDX(LY) =  (YOFX(LY) - YOFX(LY-1))/DELX
*
*               WITH MINIMUM LENGTH OF LY = 2
*
*               THE OUTPUT DYDX(1...LY) MAY REPLACE THE INPUT YOFX.
*
*               DERIVA HAS ONE OTHER OUTPUT YOFX1 WHICH IT SETS= YOFX(1).
*               USING THIS QUANTITY IT IS POSSIBLE TO INVERT EXACTLY
*               THE DIFFERENTIATED VECTOR DYDX, AND REOBTAIN YOFX.
*               THIS INVERSION IS PERFORMED BY SUBROUTINE IDERIV, WHOSE
*               CALLING SEQUENCE IS THE REVERSE OF THAT OF DERIVA.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 61 REGISTERS
* SPEED       -  7090     709       7090        709
*               (68   OR  83) + (39.4  OR  42.6)*LY  MACHINE CYCLES
* AUTHOR      - S.M. SIMPSON,  AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL DERIVA(YOFX,LY,DELX,DYDX,YOFX1)
*
* INPUTS
*
*    YOFX(I)    I=1...LY  IS THE VECTOR TO BE DIFFERENTIATED
*
*    LY         SHOULD EXCEED 1
*
*    DELX       SHOULD BE NON-ZERO  (MAY BE NEGATIVE)
*
* OUTPUTS       STRAIGHT RETURN WITH NO ACTION IF LY LSTHN 2 OR DELX = 0.
*
*    DYDX(I)    I=1...LY   IS GIVEN IN ABSTRACT
*
*               EQUIVALENCE(DYDX,YOFX) IS PERMITTED
*
*    YOFX1      IS SET = YOFX(1)
*
* EXAMPLES
*
* 1. BEHAVIOUR WITH VARIOUS DELX, LY VALUES
*    INPUTS  - Y(1...5) =  2., 6., 14., 18., 18.
*              D4 = FY4 = D5 = FY5 = -999.
*    USAGE   -       CALL DERIVA( Y, 5,  1., D1, FY1)
*                    CALL DERIVA( Y, 5, -2., D2, FY2)
*                    CALL DERIVA( Y, 2,  1., D3, FY3)
*                    CAL  DERIVA( Y, 1,  1., D4, FY4)
*                    CALL DERIVA( Y, 5,  0., D5, FY5)
*    OUTPUTS - D1(1...5) =  4., 6., 6., 2., 0.          FY1 =  2.
*              D2(1...5) = -2.,-3.,-3.,-1.,0.           FY2 =  2.
```

```
*              D3(1..2)  =  4., 4.                      FY3 =  2.
*              D4 = FY4 = D5 = FY5 = -999. (NO OUTPUT CASES)
*
* 2. MULTIPLE DIFFERENTIATION WITH OUTPUT REPLACING INPUT
*    INPUTS  - Y(1...6) =  4., 8., 12., 24., 20., 24.
*    USAGE   -       DO  10  I=1,3
*                10  CALL DERIVA( Y, 6, 1., Y, FY(I))
*    OUTPUTS - Y(1...6) =  2., 0., -3., 0., 4., 4.
*              FY(1...3) = 4., 4.,  0.
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR     0             XR4
       BCI     1,DERIVA
* ONLY ENTRY. DERIVA(YOFX,LY,DELX,DYDX,YOFX1)
DERIVA SXD     DERIVA-2,4
* CHECK LY (GRTHN= 2) AND DELX (NON-ZERO)
       CLA*    2,4           LY
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,1
       TXI     *+1,4,-1      LY-1
       SXD     TXL,4
       LXD     DERIVA-2,4
       CLA*    3,4   DELX
       TZE     LEAVE
* OK, SETUP
       STO     REC2DX
       CLA     FLP5
       FDP     REC2DX        1/2*DELX
       STQ     REC2DX
       CLA     1,4           A(YOFX)
       STA     GET
       SUB     K1            A(YOFX)-1
       STA     GET1
       CLA     4,4           A(DYDX)
       STA     STO1
       ADD     K1            A(DYDX)+1
       STA     STORE
* FORM DYDX(1) AND YOFX1
       CLA*    1,4                              YOFX(1)
       STO     OLDSTY
       STO*    5,4           TO YOFX1
 GET1  CLA     **            ** = A(YOFX)-1     YOFX(2)
       STO     MIDDLY
       FSB     OLDSTY                           YOFX(2)-YOFX(1)
       FDP     FLP5
       FMP     REC2DX                           TIMES 1/DELX
 STO1  STO     **            ** = A(DYDX)       IS DYDX(1)
* BYPASS LOOP IF LY IS 2  (LY-1 IS 1)
       LXD     TXL,4
       TXL     UPK1,4,1
* OTHERWISE PROCEED TO LOOP
       AXT     2,4
* LOOP TO SET DYDX(2,3...K...LY-1)   K IN XR4
 GET   LDQ     **,4          ** = A(YOFX)
       CLS     OLDSTY
       STQ     OLDSTY
       FAD     OLDSTY                           Y(K+1)-Y(K-1)
       XCA
       FMP     REC2DX                           TIMES 1/2DELX
 STORE STO     **,4          ** = A(DYDX)+1     IS DYDX(K)
       CLA     MIDDLY
       LDQ     OLDSTY
       STO     OLDSTY
       STQ     MIDDLY
 UPK1  TXI     *+1,4,1
 TXL   TXL     GET,4,**      ** = LY-1
* NOW SET DYDX(LY).          XR4 NOW = LY
       CLA     MIDDLY
       FSB     OLDSTY                           Y(LY)-Y(LY-1)
       FDP     FLP5
       FMP     REC2DX                           TIMES 1/DELX
       STO*    STORE                            IS DYDX(LY)
```

```
     * EXIT
      LEAVE LXD     DERIVA-2,4
            TRA     6,4
     * CONSTANTS, TEMPORARIES
      FLP5  DEC     0.5
      K1    PZE     1
     REC2DX PZE     **,**,**     1/(2*DELX)
     MIDDLY PZE     **,**,**     HOLDS YOFX(K)       INITIAL=YOFX(2)
     OLDSTY PZE     **,**,**     HOLDS YOFX(K-1)     INITIAL=YOFX(1)
            END
```

```
*      DIFPRS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0117
*      FAP
*DIFPRS
       COUNT   100
       LBL     DIFPRS
       ENTRY   DIFPRS ( X, LX,XPRSDF)
       ENTRY   XDFPRS (IX,LIX,IXPRSD)
*
*                    ----ABSTRACT----
*
* TITLE - DIFPRS WITH SECONDARY ENTRY XDFPRS
*         DIFFERENCE FIXED OR FLOATING VECTOR ELEMENTS IN PAIRS
*
*              DIFPRS FORMS A FLOATING VECTOR WHOSE ELEMENTS ARE THE
*              DIFFERENCES OF SUCCESSIVE PAIRS OF THE ELEMENTS OF
*              ANOTHER FLOATING VECTOR, THE FIRST OUTPUT ELEMENT BEING
*              SET EQUAL TO THE FIRST INPUT ELEMENT.  OUTPUT MAY REPLACE
*              INPUT.
*
*              XDFPRS DOES THE SAME THING FOR FIXED VECTORS.
*
*              DIFPRS AND XDFPRS ARE THE EXACT INVERSE OPERATORS OF
*              SUBROUTINES INTSUM AND XNTSUM RESPECTIVELY.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 30   REGISTERS
* SPEED      - DIFPRS  30.6 + 12.4*LX  MACHINE CYCLES,
*              XDFPRS  37.0 +  8.0*LX   LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL DIFPRS( X, LX,XPRSDF)
*     CALL XDFPRS(IX,LIX,IXPRSD)
*
* INPUTS
*
*    X(I)      I=1...LX  IS A FLOATING VECTOR INPUT TO DIFPRS
*
*    LX        SHOULD EXCEED 0
*
*    IX(I)     I=1...LX  IS A FIXED VECTOR INPUT TO XDFPRS. THE POSITION
*                  OF THE BINARY POINT IS ARBITRARY.
*
*    LIX       SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1
*
*    XPRSDF(I) I=1...LX  IS  XPRSDF(1) = X(1)
*                        AND   XPRSDF(I) = X(I) -  X(I-1) , I=2...LX
*
*    IXPRSD(I) I=1...LX  IS  IXPRSD(1) = IX(1)
*                        AND   IXPRSD(I) = IX(I) - IX(I-1) , I=2...LX
*                WITH SAME BINARY POINT AS IX(I).
*
*              EQUIVALENCE (XPRSDF,X),(IXPRSD,IX)  IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4) = 1., 3., 6., 10. IX(1...4) = 1,3,6,10 XDF3=0.
*    USAGE   -       CALL DIFPRS( X,4, XDF1)
*                    CALL XDFPRS(IX,4,IXDF1)
*                    CALL DIFPRS( X,4, X)
*                    CALL DIFPRS( X,1, XDF2)
*                    CALL DIFPRS( X,0, XDF3)
*    OUTPUTS - XDF1(1...4) = 1., 2., 3., 4.      IXDF1(1...4) = 1,2,3,4
*                 X(1...4) = 1., 2., 3., 4.      XDF2 = 1.
*              XDF3 = 0. (NO OUTPUT CASE)
*
* 2. INPUTS  - IX(1...3) = OCT 000000000001, 000000000003, 000000000006
```

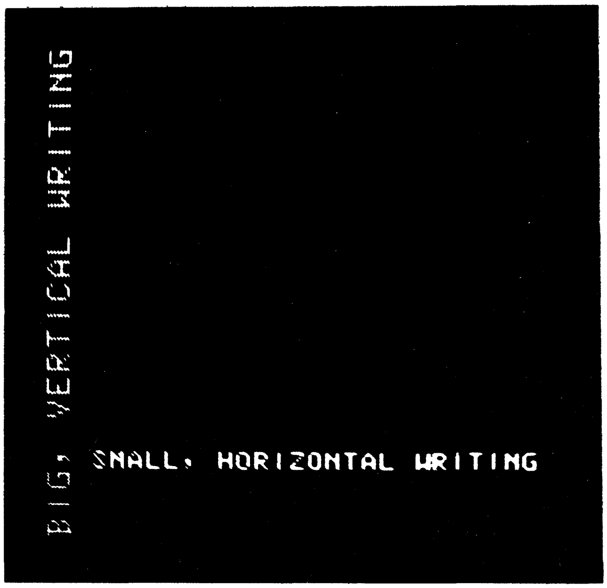
DIFPRS
(PAGE 2)

```
*     USAGE    -        CALL XDFPRS(IX,3,IX)
*     OUTPUTS - IX(1...3) = OCT 000000000001, 000000000002, 000000000003
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR     0             XR4
        BCI     1,DIFPRS
* PRINCIPAL ENTRY.  DIFPRS(X,LX,XPRSDF)
DIFPRS  CLA     FSB
 SETUP  STO     SUBTR
        SXD     DIFPRS-2,4
 K1     CLA     1,4           A(X)
        STA     GET
        ADD     K1            A(X)+1
        STA     SUBTR
        CLA     3,4           A(XPRSDF)
        STA     STORE
        CLA*    2,4           LX
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
        TXI     *+1,4,-1      LX-1
        TXL     LAST,4,0
* LOOP FOR ALL BUT XPRSDF(1)
 GET    CLA     **,4          ** = A(X)
 SUBTR  NOP                   FSB  **,4 OR  SUB  **,4   ** = A(X)+1
 STORE  STO     **,4          ** = A(XPRSDF)
        TIX     GET,4,1
* SET XPRSDF(1)
 LAST   LXD     DIFPRS-2,4
        CLA*    1,4           X(1)
        STO*    3,4           XPRSDF(1)
* EXIT
 LEAVE  LXD     DIFPRS-2,4
        TRA     4,4
* SECOND ENTRY. XDFPRS(IX,LIX,IXPRSD)
XDFPRS  CLA     SUB
        TRA     SETUP
* CONSTANTS
 FSB    FSB     **,4
 SUB    SUB     **,4
        END
```

```
*      DISPLA (709) (SUBROUTINE)        9/29/64   LAST CARD IN DECK IS NO. 0473
*      FAP
*DISPLA (709)
       COUNT   400
       LBL     DISPLA
       ENTRY   DISPLA
*
*                    ----ABSTRACT----
*
* TITLE - DISPLA (709)
*       WRITE HOLLERITH TEXT ON SCOPE
*
*             DISPLA PRODUCES TITLES, LABELS, AND LEGENDS FOR SCOPE
*             DISPLAYS.  IT CAN PLOT 64 CHARACTERS IN EITHER LARGE (36
*             CHARACTERS ACROSS THE SCOPE) OR SMALL (48 LETTERS ACROSS
*             THE SCOPE) MODES IN EITHER A HORIZONTAL OR VERTICAL
*             DIRECTION.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709  (MAIN FRAME AND SCOPE)
* STORAGE   - 220 REGISTERS
* SPEED     -
* AUTHOR    - DISPLA IS A CONVERSION BY THE M.I.T. COMPUTATION CENTER OF
*             THE SUBPROGRAM WRITE AS DESCRIBED IN M.I.T. LINCOLN LAB.
*             MEMO. NO. 54-0003.  THE VERSION HERE IS SLIGHTLY MODIFIED
*             BY J. GALBRAITH (TO MAKE IT INVARIANT TO USE OR NON-USE OF
*             STANDARD ERROR PROCEDURE).
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - (IOH)
*
* FORTRAN USAGE
*     CALL DISPLA
*     PRINT 10,(LIST)
* 10  FORMAT (DISCON,FMT)
*
* INPUTS
*
*      PRIMARILY WHAT APPEARS ON THE SCOPE IS WHAT WOULD HAVE BEEN
*        WRITTEN BY THE PRINT STATEMENT WHICH FOLLOWS THE CALL DISPLA
*        STATEMENT.  HOWEVER, THE BEGINNING CHARACTERS (CALLED DISCON
*        IN THE ABOVE FORMAT) OF THE FORMAT ARE USED TO CONTROL THE
*        MODE OF THE DISPLAY.
*
*    DISCON    IS A VARIABLE LENGTH HOLLERITH FIELD
*                  1. THE FIRST CHARACTER IS A CONTROL CHARACTER AND
*                       MUST BE ONE OF THE FOLLOWING
*
*                     CHARACTER   ACTION CAUSED
*
*                       +         SAME MODE AND ORIGIN.
*                       0 (ZERO)  SAME MODE, DOUBLE SPACE.
*                      (BLANK)    SAME MODE, SINGLE SPACE.
*                       1         CHANGE FILM FRAME, NEW MODE,
*                                   NEW ORIGIN.
*                       2         NEW MODE, NEW ORIGIN
*
*                    WHERE MODE REFERS TO THE SIZE OF THE CHARACTERS
*                      AND TO THE DIRECTION OF PLOTTING, AND ORIGIN
*                      REFERS TO THE LOCATION OF THE FIRST CHARACTER
*                      OF THE LINE.
*
*                   IF THIS CHARACTER IS A +,0, OR BLANK, NO OTHER
*                   CHARACTERS ARE USED.
*
*                  2. THE SECOND CHARACTER CONTROLS THE SIZE OF THE
*                       PLOTTED CHARACTERS.
*
*                       B   BIG CHARACTERS (20 BY 28 SCOPE UNITS)
*                       S   SMALL CHARACTERS (15 BY 21 SCOPE UNITS)
*
```

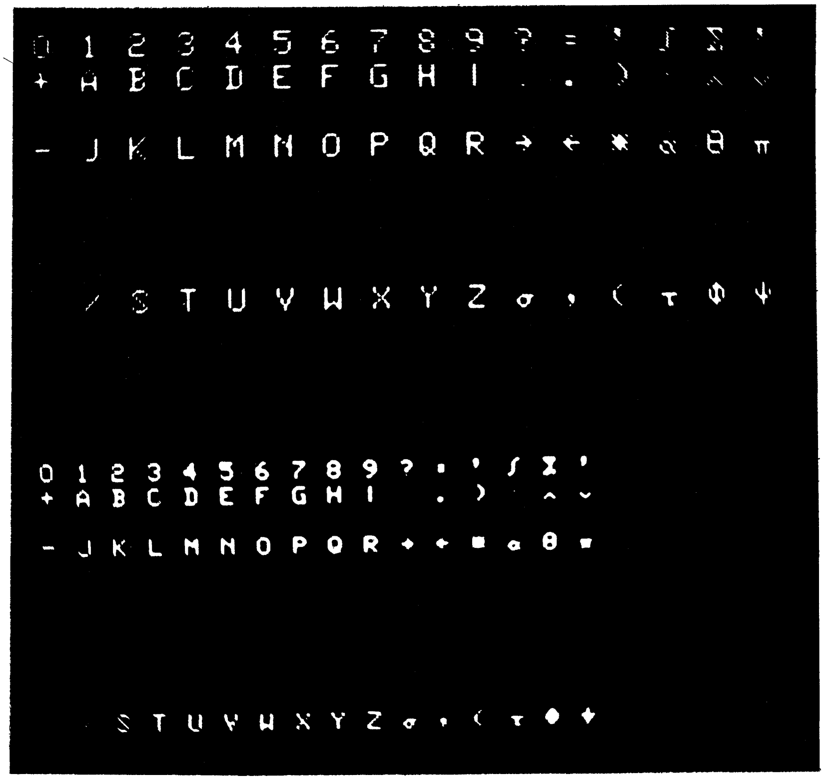
```
*                   3. THE THIRD CHARACTER CONTROLS THE DIRECTION OF
*                        PLOTTING.
*
*                         H   HORIZONTAL
*                         V   VERTICAL
*
*                   4. THE LAST SET OF INFORMATION CONSISTS OF TWO 0 TO
*                        4 DIGIT INTEGERS (GRTHN=0, LSTHN 1024) FOLLOWED
*                        BY COMMAS.  THE FIRST INTEGER INDICATES THE
*                        X-COORDINATE AND THE SECOND INTEGER THE Y-
*                        COORDINATE (IN SCOPE UNITS) OF THE LOWER LEFT
*                        CORNER AT WHICH PLOTTING BEGINS.
*
*               THERE MUST BE NO BLANKS BETWEEN ANY OF THESE CHARACTERS
*
*    FMT        IMMEDIATLY FOLLOWS DISCON
*               IS THE STANDARD FORMAT FOR THE INFORMATION WHICH IS TO
*                 BE WRITTEN ON THE SCOPE.
*               SHOULD NOT CALL FOR A LINE LONGER THAN 48 (FOR SMALL) OR
*                 36 (FOR BIG) CHARACTERS.  IF A LINE GOES BEYOND THE
*                 EDGE OF THE SCOPE, THE END IS WRITTEN BEGINNING AT THE
*                 OPPOSITE EDGE.
*
*    LIST       IS THE APPROPRIATE LIST WHICH CORRESPONDS TO FMT.
*
*      THE FOLLOWING IS A LIST OF THE SPECIAL CHARACTERS AND THEIR OCTAL
*        EQUIVALENTS WHICH DISPLA WILL RECOGNIZE IN ADDITION TO THE
*        STANDARD CHARACTERS.
*
*           APOSTROPHE      14         ARROW LEFT    53
*           INTEGRAL SIGN   15         ALPHA         55
*           SUMMATION SIGN  16         THETA         56
*           APOSTROPHE      17         PI            57
*           LOW POINT       32         SMALL SIGMA   72
*           MIDDLE POINT    35         TAU           75
*           CAP             36         PHI           76
*           CUP             37         PSI           77
*           ARROW RIGHT     52
*
* EXAMPLES
*
* 1. EXAMPLE OF BIG, VERTICAL WRITING AND CHANGING THE FILM FRAME.
*    USAGE   -        CALL DISPLA
*                     PRINT 10
*                  10 FORMAT(9H1BV10,10,21HBIG, VERTICAL WRITING)
*
* 2. EXAMPLE OF SMALL, HORIZONTAL WRITING ON SAME FILM FRAME.
*    USAGE   -        CALL DISPLA
*                     PRINT 20
*                  20 FORMAT(10H2SH120,90,25HSMALL, HORIZONTAL WRITING)
```

```
*    OUTPUTS -
```

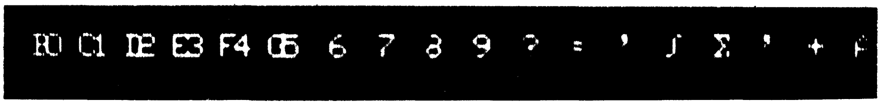



```
*
*
* 3. EXAMPLES OF ALL THE CHARACTERS, SINGLE SPACING AND DOUBLE SPACING
*      IN BOTH BIG AND SMALL.
*    INPUTS  - A(1...63) = OCT 016060606060,026060606060,...776060606060
*    USAGE   -         CALL DISPLA
*                      PRINT 30, (A(I),I=1,16)
*                   30 FORMAT(10H1BH56,900,16A2)
*                      CALL DISPLA
*                      PRINT 40, (A(I),I=17,32)
*                   40 FORMAT(1H 16A2)
*                      CALL DISPLA
*                      PRINT 50, (A(I),I=33,48)
*                   50 FORMAT(1H016A2)
*                      CALL DISPLA
*                      PRINT 60, (A(I),I=49,63)
*                   60 FORMAT(10H2BH56,600,16A2)
*                AND A SIMILAR SEQUENCE TO PLACE SMALL CHARACTERS IN
*                  THE BOTTOM OF THE FRAME.
```

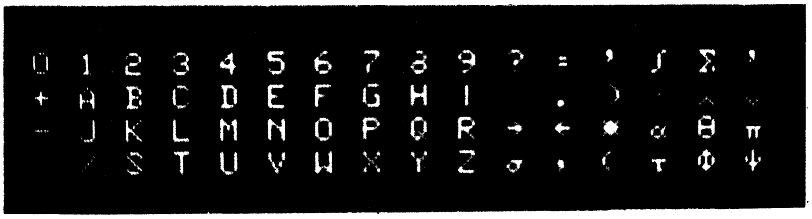
```
*     OUTPUTS -
*
```



```
*
*
*
* 4. EXAMPLE OF A LINE EXTENDING BEYOND THE EDGE OF THE SCOPE.
*     USAGE    -          CALL DISPLA
*                         PRINT 70, (A(I),I=1,24)
*                      70 FORMAT(10H1BH56,500,24A2)
*     OUTPUTS -
*
```



```
*
*
* 5. EXAMPLE OF DISPLAY SPACING UNDER FORMAT CONTROL
*     USAGE    -          CALL DISPLA
*                         PRINT 80, (A(I),I=1,63)
*                      80 FORMAT(10H1BH56,500,16A2/1H 16A2/1H 16A2/1H 16A2)
*     OUTPUTS -
*
```



```
        PZE
        BCI     1,DISPLA
DISPLA  CAL     1,4               CHECK FOR STANDARD ERROR PROCEDURE
        ANA     MASK2                  MASK2=770377000000
        TZE     NOERR             ZERO, NO STANDARD ERROR PROCEDURE
```

DISPLA (709)
(PAGE 5)

```
        CAL     ((S))         NOT ZERO, STANDARD ERROR PRESENT
        STA     3,4
        TRA     1,4
  NOERR CAL     ((S))
        STA     1,4
        WTV
        TRA     1,4
        REM
        REM
        REM WOS - WRITE ON SCOPE
 (STVH) LDQ     *+4
        CLA     *+2
        TRA*    $(IOH)
        MZE     ,,3
        TRA     WOS
  WOS   SXD     DISPLA-2,4
        SXA     OUT+1,1
        SXA     OUT+2,2
        CLA 1,4         FETCH PZE Z,,N. FORMAT WORDS ARE IN Z BSS N
        ARS 18
        ADD 1,4
        STA A
  (SVH) PDX WOS,1
        LXD MODE,2
        TSX A,4               FETCH FIRST CHARACTER
        LXA A5,1              IDENTIFY CONTROL CHARACTER
        ADD C,1
        TZE D,1
        TIX *-2,1,1
        TRA     OUT            ILLEGAL CONTROL CHARACTER
        REM
        TSX     CFF,4          TRANSFER VECTOR
        TRA     NUORG              ..
        ACL     INCR,2             ..
        ACL     INCR,2             ..
        ACL     ORGIN              ..
  D     ANA MASK
        STO ORGIN
  TRACE LXD MODE,2
        CAL ORGIN             WRITE RECORD
        ACL 6U3L,2            MOVE POINT OF ORIGIN
  E     ACL 2R,2              NEXT CHARACTER, MOVE POINT
        ANA MASK                                        AND STORE
        SLW POINT
        TSX FETCH,4           FETCH CHARACTER
  A5    PAX 5,4               IS IT BLANK
        SUB BLANK
        TNZ *+4
        CAL POINT
        ACL 7R,2
        TRA E+1
        LDQ PAT,4             NO. FETCH PATTERN
        LXA A5,1              DO 5 COLUMNS
        CAL POINT
  LP1   ACL     7D1R,2         NEXT COLUMN, MOVE POINT
        LXA A7,4              DO 7 ROWS
  LP2   ADD     1U,2           NEXT ROW, MOVE POINT
        RQL 1                 DO POINT
        TQP     ELP2
        SLW POINT             PLOT POINT
        STQ     T
        CPY     POINT
        LDQ     T
  ELP2  TIX     LP2,4,1
        TIX     LP1,1,1        COLUMN DONE
        TRA     E              CHARACTER DONE
        REM
  A     LDQ **,1              NEW WORD      ** = Z+N
        STQ WORD
        SXD WCNT,1
        LXA A7,1
        SXD CCNT,1
  FETCH LXD CCNT,1
        LDQ WORD
  ((S)) PXD     (STVH)        STORAGE TO TV HOLLERITH
```

```
A6     LGL 6
       STQ WORD
       TIX B,1,1
       LXD WCNT,1
       TIX A,1,1
OUT    LXD      DISPLA-2,4
       AXT      **,1
       AXT      **,2
       TRA 2,4
B      SXD CCNT,1
       TRA 1,4
       REM
NUORG  TSX FETCH,4              COMPUTE MODE
       STO T                    B OR S
       TSX FETCH,4              H OR V
       ADD T
       LRS 1
       ARS 4
       RND
A7     PAX 7,2                  SV=4,BV=3,SH=2,BH=1
       SXD MODE,2
       LXA A2,2                 COMPUTE ORIGIN
R      STZ T,2
       TSX FETCH,4
       CAS TEN
       NOP
MODE   TXI F,,**
       STO T
       CLA T,2
A2     ALS 2
       ADD T,2
       ALS 1
       ADD T
       STO T,2
WCNT   TXI      R+1,,**
F      TIX R,2,1
       CLA T-2
       ALS 18
       ADD T-1
       STO ORGIN
CCNT   TXI      TRACE,,**
*      SUBROUTINE TO CHANGE FILM FRAME
CFF    CFF
       TRA      1,4
       REM
       DEC -1,-1,2,-48,32       CC IS 1,2,0, ,+
C      SYN *
       PZE 6                    SV
       PZE 8                    BV
       PZE ,,6                  SH
       PZE ,,8                  BH
2R     SYN *
       PZE ,,30                 SV
       PZE ,,40                 BV
       PZE 994                  SH
       PZE 984                  BH
INCR   SYN *
       MZE ,,3                  SV
       MZE ,,4                  BV
       PZE 3                    SH
       PZE 4                    BH
1U     SYN *
       PZE 3,,21                SV
       PZE 4,,28                BV
       PZE 1003,,3              SH
       PZE 996,,4               BH
7D1R   SYN *
           21                   SV
           28                   BV
           ,,21                 SH
           ,,28                 BH
7R     SYN *
       PZE 1015,,1006           SV
       PZE 1012,,1000           BV
       PZE 18,,1015             SH
```

* DISPLA (709) *
(PAGE 7)

```
       PZE 24,,1012          BH
6U3L   SYN *
       REM
       OCT 30207744014          77 PSI
       OCT 070427750434         76 PHI
       OCT 020107422010         75 TAU
       OCT 070424040000    74 (                        0199
       OCT 001303400000    73 ,                        0200
       OCT 141104416010         72 SMALL SIGMA
       OCT 303214461303    71 Z                        0202
       OCT 006047401003    70 Y                        0203
       OCT 306240405143    67 X                        0204
       OCT 376401010177    66 W                        0205
       OCT 016306006007    65 V                        0206
       OCT 177004020077    64 U                        0207
       OCT 002017740201    63 T                        0208
       OCT 105054464242    62 S                        0209
       OCT 100200401002    61 /                        0210
ORGIN
       OCT 021700436010         57 PI
       OCT 175114462276         56 THETA
       OCT 141104414110         55 ALPHA
       OCT 124343707052    54 *                        0215
       OCT 020342502010         53 ARROW LEFT
       OCT 020102507010         52 ARROW RIGHT
       OCT 376111452306    51 R                        0218
       OCT 175015050336    50 Q                        0219
       OCT 376110442206    47 P                        0220
       OCT 175014060276    46 O                        0221
       OCT 376020202177    45 N                        0222
       OCT 376020600577    44 M                        0223
       OCT 377004020100    43 L                        0224
       OCT 376101210501    42 K                        0225
       OCT 101004020077    41 J                        0226
       OCT 020100402010    40 -                        0227
       OCT 040404010020         37 CUP
       OCT 200401010100         36 CAP
       OCT 000000400000         35 MIDDLE POINT
       OCT 000004050434    34 )                        0231
       OCT 001406000000    33 .                        0232
       OCT 4000000              32 LOW POINT
       OCT 000007740000    31 I                        0234
       OCT 376100402177    30 H                        0235
       OCT 175014062371    27 G                        0236
       OCT 376110440201    26 F                        0237
       OCT 377114460301    25 E                        0238
       OCT 203774060276    24 D                        0239
       OCT 175014060242    23 C                        0240
       OCT 203774462266    22 B                        0241
       OCT 370221044574    21 A                        0242
       OCT 020103702010    20 +                        0243
       OCT 000130340000         17 APOSTROPHE
       OCT 203435242343         16 SUMMATION SIGN
       OCT 201003700201         15 INTEGRAL SIGN
       OCT     000130340000      14 APOSTROPHE
       OCT 000241205000    13 =                        0248
       OCT 004015442206         12 QUESTION MARK
       OCT 015114452236    11 9                        0250
       OCT 155114462266    10 8                        0251
       OCT 003610441203    07 7                        0252
       OCT 171124462260    06 6                        0253
       OCT 117054261271    05 5                        0254
       OCT 060241137620    04 4                        0255
       OCT 105014462266    03 3                        0256
       OCT 345114462306    02 2                        0257
       OCT 001027760000    01 1                        0258
PAT    OCT 175014060276    00 0                        0259
MASK   OCT 1777001777
MASK2  OCT     770377000000
BLANK  BCD 100000
TEN    DEC 10
WORD   BSS     1
       BSS     1
       BSS     1
T      BSS     1
```

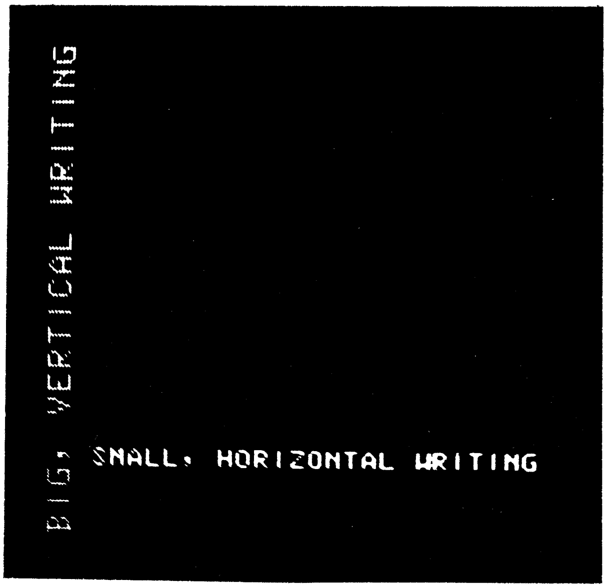
****
*   DISPLA (709)   *
****
(PAGE 8)

```
      POINT BSS     1                                                        0472
            END                                                              0473
```

```
*       DISPLA (7090) (SUBROUTINE)        9/4/64   LAST CARD IN DECK IS NO. 0480
*       FAP
*DISPLA (7090)
        COUNT   450
        LBL     DISPLA
        ENTRY   DISPLA
*
*                    ----ABSTRACT----
*
* TITLE - DISPLA (7090)
*         WRITE HOLLERITH TEXT ON SCOPE
*
*              DISPLA PRODUCES TITLES, LABELS, AND LEGENDS FOR SCOPE
*              DISPLAYS.  IT CAN PLOT 64 CHARACTERS IN EITHER LARGE (36
*              CHARACTERS ACROSS THE SCOPE) OR SMALL (48 LETTERS ACROSS
*              THE SCOPE) MODES IN EITHER A HORIZONTAL OR VERTICAL
*              DIRECTION.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090 (MAIN FRAME, DATA CHANNEL D, AND SCOPE)
* STORAGE   - 219 REGISTERS
* SPEED     -
* AUTHOR    - DISPLA IS A CONVERSION BY THE MIT COMPUTATION CENTER OF
*              THE SUBPROGRAM WRITE AS DESCRIBED IN M.I.T. LINCOLN LAB
*              MEMO. NO. 54-0003.
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - FRAME
*         AND FORTRAN SYSTEM ROUTINES - (IOH)
*
* FORTRAN USAGE
*     CALL DISPLA
*     PRINT 10,(LIST)
* 10  FORMAT (DISCON,FMT)
*
* INPUTS
*
*       PRIMARILY WHAT APPEARS ON THE SCOPE IS WHAT WOULD HAVE BEEN
*         WRITTEN BY THE PRINT STATEMENT WHICH FOLLOWS THE CALL DISPLA
*         STATEMENT.  HOWEVER, THE BEGINNING CHARACTERS (CALLED DISCON
*         IN THE ABOVE FORMAT) OF THE FORMAT ARE USED TO CONTROL THE
*         MODE OF THE DISPLAY.
*
*    DISCON    IS A VARIABLE LENGTH HOLLERITH FIELD
*                   1. THE FIRST CHARACTER IS A CONTROL CHARACTER AND
*                        MUST BE ONE OF THE FOLLOWING
*
*                      CHARACTER   ACTION CAUSED
*
*                         +         SAME MODE AND ORIGIN.
*                         0         SAME MODE, DOUBLE SPACE.
*                      (BLANK)      SAME MODE, SINGLE SPACE.
*                         1         CHANGE FILM FRAME, NEW MODE,
*                                     NEW ORIGIN.
*                         2         NEW MODE, NEW ORIGIN
*
*                      WHERE MODE REFERS TO THE SIZE OF THE CHARACTERS
*                        AND TO THE DIRECTION OF PLOTTING, AND ORIGIN
*                        REFERS TO THE LOCATION OF THE FIRST CHARACTER
*                        OF THE LINE.
*
*                      IF THIS CHARACTER IS A +,0, OR BLANK, NO OTHER
*                        CONTROL CHARACTERS ARE USED.
*
*                   2. THE SECOND CHARACTER CONTROLS THE SIZE OF THE
*                        PLOTTED CHARACTERS.
*
*                         B   BIG CHARACTERS (20 BY 28 SCOPE UNITS)
*                         S   SMALL CHARACTERS (15 BY 21 SCOPE UNITS)
*
```

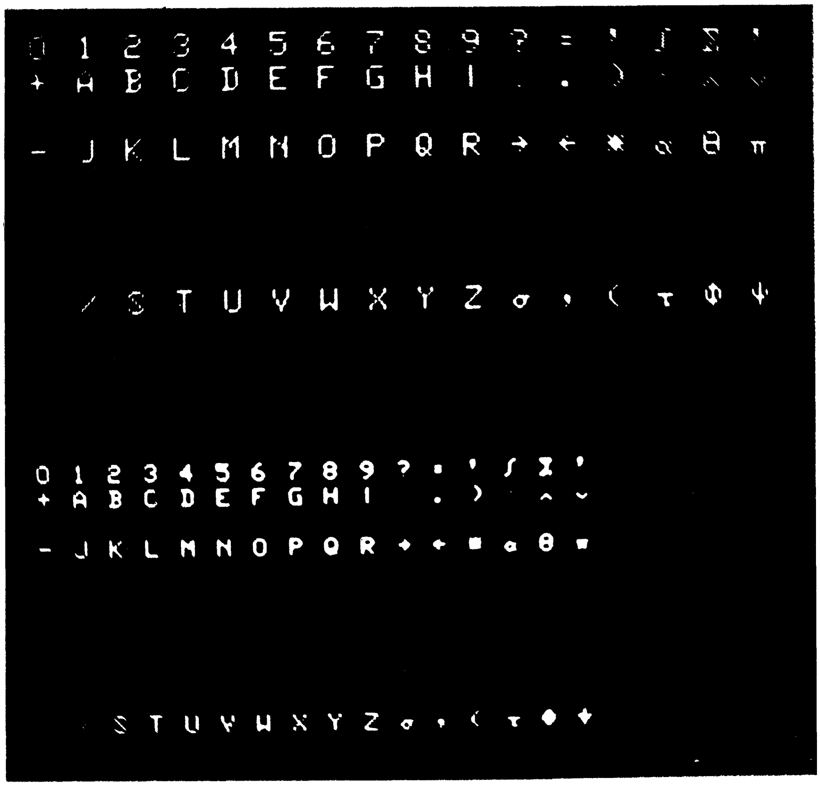
```
*                   3. THE THIRD CHARACTER CONTROLS THE DIRECTION OF
*                        PLOTTING.
*
*                         H   HORIZONTAL
*                         V   VERTICAL
*                              (NOTE - VERTICAL MODE CHARACTERS READ
*                               CORRECTLY WHEN PICTURE IS ROTATED
*                               90 DEGREES CLOCKWISE)
*
*                   4. THE LAST SET OF INFORMATION CONSISTS OF TWO 0 TO
*                        4 DIGIT INTEGERS (GRTHN=0, LSTHN 1024) FOLLOWED
*                        BY COMMAS.  THE FIRST INTEGER INDICATES THE
*                        X-COORDINATE AND THE SECOND INTEGER THE Y-
*                        COORDINATE (IN SCOPE UNITS) OF THE LOWER LEFT
*                        CORNER AT WHICH PLOTTING BEGINS.
*
*               THERE MUST BE NO BLANKS BETWEEN ANY OF THESE CHARACTERS
*
*     FMT       IMMEDIATLY FOLLOWS DISCON
*               IS THE STANDARD FORMAT FOR THE INFORMATION WHICH IS TO
*                 BE WRITTEN ON THE SCOPE.
*               SHOULD NOT CALL FOR A LINE LONGER THAN 48 (FOR SMALL) OR
*                 36 (FOR BIG) CHARACTERS.  IF A LINE GOES BEYOND THE
*                 EDGE OF THE SCOPE, THE END IS WRITTEN BEGINNING AT THE
*                 OPPOSITE EDGE.
*
*     LIST      IS THE APPROPRIATE LIST WHICH CORRESPONDS TO FMT.
*
*       THE FOLLOWING IS A LIST OF THE SPECIAL CHARACTERS AND THEIR OCTAL
*         EQUIVALENTS WHICH DISPLA WILL RECOGNIZE IN ADDITION TO THE
*         STANDARD CHARACTERS.
*
*            APOSTROPHE     14         ARROW LEFT    53
*            INTEGRAL SIGN  15         ALPHA         55
*            SUMMATION SIGN 16         THETA         56
*            APOSTROPHE     17         PI            57
*            LOW POINT      32         SMALL SIGMA   72
*            MIDDLE POINT   35         TAU           75
*            CAP            36         PHI           76
*            CUP            37         PSI           77
*            ARROW RIGHT    52
*
* EXAMPLES
*
* 1. EXAMPLE OF BIG, VERTICAL WRITING AND CHANGING THE FILM FRAME.
*
*     USAGE   -        CALL DISPLA
*                      PRINT 10
*                   10 FORMAT(9H1BV90,10,21HBIG, VERTICAL WRITING)
*
* 2. EXAMPLE OF SMALL, HORIZONTAL WRITING ON SAME FILM FRAME.
*
*     USAGE   -        CALL DISPLA
*                      PRINT 20
*                   20 FORMAT(10H2SH120,90,25HSMALL, HORIZONTAL WRITING)
```

```
*    OUTPUTS -
*
```

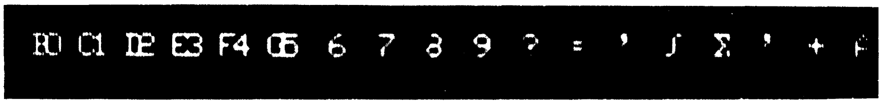



```
*
*
* 3. EXAMPLES OF ALL THE CHARACTERS, SINGLE SPACING AND DOUBLE SPACING
*      IN BOTH BIG AND SMALL.
*
*    INPUTS  - A(1...63) = OCT 016060606060,026060606060,...776060606060
*
*    USAGE   -          CALL DISPLA
*                       PRINT 30, (A(I),I=1,16)
*                    30 FORMAT(10H1BH56,900,16A2)
*                       CALL DISPLA
*                       PRINT 40, (A(I),I=17,32)
*                    40 FORMAT(1H 16A2)
*                       CALL DISPLA
*                       PRINT 50, (A(I),I=33,48)
*                    50 FORMAT(1H016A2)
*                       CALL DISPLA
*                       PRINT 60, (A(I),I=49,63)
*                    60 FORMAT(10H2BH56,600,16A2)
*                   AND A SIMILAR SEQUENCE TO PLACE SMALL CHARACTERS IN
*                     THE BOTTOM OF THE FRAME.
```

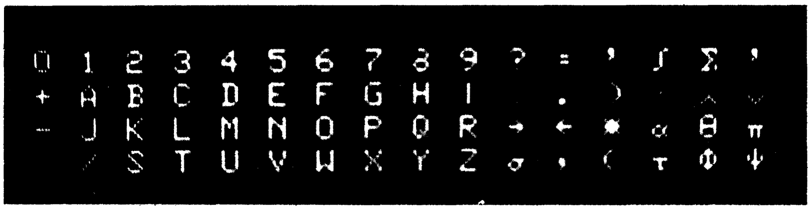
```
*     OUTPUTS -
```




```
* 4. EXAMPLE OF A LINE EXTENDING BEYOND THE EDGE OF THE SCOPE.
*    USAGE   -        CALL DISPLA
*                     PRINT 70, (A(I),I=1,24)
*                  70 FORMAT(10H1BH56,500,24A2)
*    OUTPUTS -
```




```
* 5. EXAMPLE OF DISPLAY SPACING UNDER FORMAT CONTROL
*    USAGE   -        CALL DISPLA
*                     PRINT 80, (A(I),I=1,63)
*                  80 FORMAT(10H1BH56,500,16A2/1H 16A2/1H 16A2/1H 16A2)
*    OUTPUTS -
```




```
*
       BSS     0
* FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
* TO CHANGE, ALTER THE LETTER DESIGNATION ONLY AND REASSEMBLE.
X      TAPENO  D1
SCPAD  EQU     X-105
       PZE
```

```
          BCI     1,DISPLA
   DISPLA CAL     1,4          CHECK FOR STANDARD ERROR PROCEDURE
          ANA     MASK2             MASK2=770377000000
          TZE     NOERR        ZERO, NO STANDARD ERROR PROCEDURE
          CAL     ((S))        NOT ZERO, STANDARD ERROR PRESENT
          STA     3,4
          TRA     1,4
    NOERR CAL     ((S))
          STA     1,4
          TRA     1,4
          REM
          REM
          REM WOS - WRITE ON SCOPE
   (STVH) LDQ     *+4
          CLA     *+2
          TRA*    $(IOH)
          MZE     ,,3
          TRA     WOS
    WOS   SXD     DISPLA-2,4
          SXA     OUT+1,1
          SXA     OUT+2,2
          CLA 1,4          FETCH PZE Z,,N. FORMAT WORDS ARE IN Z BSS N
          ARS 18
          ADD 1,4
          STA A
    (SVH) PDX WOS,1
          LXD MODE,2
          TSX A,4                FETCH FIRST CHARACTER
          LXA A5,1               IDENTIFY CONTROL CHARACTER
          ADD C,1
          TZE D,1
          TIX *-2,1,1
          TRA     OUT             ILLEGAL CONTROL CHARACTER
          REM
          TSX     $FRAME,4
          TRA     NUORG              ..
          ACL     INCR,2             ..
          ACL     INCR,2             ..
          ACL     ORGIN              ..
    D     ANA MASK
          STO ORGIN
    TRACE LXD MODE,2
          CAL ORGIN              WRITE RECORD
          ACL 6U3L,2             MOVE POINT OF ORIGIN
    E     ACL 2R,2               NEXT CHARACTER, MOVE POINT
          ANA MASK                                          AND STORE
          SLW POINT
          TSX FETCH,4            FETCH CHARACTER
    A5    PAX 5,4                IS IT BLANK
          SUB BLANK
          TNZ *+4
          CAL POINT
          ACL 7R,2
          TRA E+1
          LDQ PAT,4              NO. FETCH PATTERN
          LXA A5,1               DO 5 COLUMNS
          CAL POINT
    LP1   ACL     7D1R,2          NEXT COLUMN, MOVE POINT
          LXA A7,4               DO 7 ROWS
    LP2   ADD     1U,2            NEXT ROW, MOVE POINT
          RQL 1                  DO POINT
          TQP     ELP2
          SLW POINT              PLOT POINT
          WRS     SCPAD
          RCHX    IOC
          TCOX    *
    ELP2  TIX     LP2,4,1
          TIX     LP1,1,1         COLUMN DONE
          TRA     E               CHARACTER DONE
          REM
    A     LDQ **,1               NEW WORD      ** = Z+N
          STQ WORD
          SXD WCNT,1
          LXA A7,1
          SXD CCNT,1
```

```
FETCH LXD CCNT,1
      LDQ WORD
((S)) PXD      (STVH),0      STORAGE TO TV HOLLERITH
A6    LGL 6
      STQ WORD
      TIX B,1,1
      LXD WCNT,1
      TIX A,1,1
OUT   LXD      DISPLA-2,4
      AXT      **,1
      AXT      **,2
      TRA 2,4
B     SXD CCNT,1
      TRA 1,4
      REM
NUORG TSX FETCH,4            COMPUTE MODE
      STO T                  B OR S
      TSX FETCH,4            H OR V
      ADD T
      LRS 1
      ARS 4
      RND
A7    PAX 7,2                SV=4,BV=3,SH=2,BH=1
      SXD MODE,2
      LXA A2,2               COMPUTE ORIGIN
R     STZ T,2
      TSX FETCH,4
      CAS TEN
      NOP
MODE  TXI F,,**
      STO T
      CLA T,2
A2    ALS 2
      ADD T,2
      ALS 1
      ADD T
      STO T,2
WCNT  TXI      R+1,,**
F     TIX R,2,1
      CLA T-2
      ALS 18
      ADD T-1
      STO ORGIN
CCNT  TXI      TRACE,,**
      REM
      DEC -1,-1,2,-48,32       CC IS 1,2,0, ,+
C     SYN *
      PZE 6                  SV
      PZE 8                  BV
      PZE ,,6                SH
      PZE ,,8                BH
2R    SYN *
      PZE ,,30               SV
      PZE ,,40               BV
      PZE 994                SH
      PZE 984                BH
INCR  SYN *
      MZE ,,3                SV
      MZE ,,4                BV
      PZE 3                  SH
      PZE 4                  BH
1U    SYN *
      PZE 3,,21              SV
      PZE 4,,28              BV
      PZE 1003,,3            SH
      PZE 996,,4             BH
7D1R  SYN *
          21                 SV
          28                 BV
          ,,21               SH
          ,,28               BH
7R    SYN *
      PZE 1015,,1006         SV
      PZE 1012,,1000         BV
      PZE 18,,1015           SH
```

```
        PZE 24,,1012          BH
 6U3L   SYN *
        REM
        OCT 30207744014         77 PSI
        OCT 070427750434        76 PHI
        OCT 020107422010        75 TAU
        OCT 070424040000     74 (                               0199
        OCT 001303400000     73 ,                               0200
        OCT 141104416010        72 SMALL SIGMA
        OCT 303214461303     71 Z                               0202
        OCT 006047401003     70 Y                               0203
        OCT 306240405143     67 X                               0204
        OCT 376401010177     66 W                               0205
        OCT 016306006007     65 V                               0206
        OCT 177004020077     64 U                               0207
        OCT 002017740201     63 T                               0208
        OCT 105054464242     62 S                               0209
        OCT 100200401002     61 /                               0210
 ORGIN
        OCT 021700436010        57 PI
        OCT 175114462276        56 THETA
        OCT 141104414110        55 ALPHA
        OCT 124343707052     54 *                               0215
        OCT 020342502010        53 ARROW LEFT
        OCT 020102507010        52 ARROW RIGHT
        OCT 376111452306     51 R                               0218
        OCT 175015050336     50 Q                               0219
        OCT 376110442206     47 P                               0220
        OCT 175014060276     46 O                               0221
        OCT 376020202177     45 N                               0222
        OCT 376020600577     44 M                               0223
        OCT 377004020100     43 L                               0224
        OCT 376101210501     42 K                               0225
        OCT 101004020077     41 J                               0226
        OCT 020100402010     40 -                               0227
        OCT 040404010020        37 CUP
        OCT 200401010100        36 CAP
        OCT 000000400000        35 MIDDLE POINT
        OCT 000004050434     34 )                               0231
        OCT 001406000000     33 .                               0232
        OCT 4000000             32 LOW POINT
        OCT 000007740000     31 I                               0234
        OCT 376100402177     30 H                               0235
        OCT 175014062371     27 G                               0236
        OCT 376110440201     26 F                               0237
        OCT 377114460301     25 F                               0238
        OCT 203774060276     24 D                               0239
        OCT 175014060242     23 C                               0240
        OCT 203774462266     22 B                               0241
        OCT 370221044574     21 A                               0242
        OCT 020103702010     20 +                               0243
        OCT 000130340000        17 APOSTROPHE
        OCT 203435242343        16 SUMMATION SIGN
        OCT 201003700201        15 INTEGRAL SIGN
        OCT     000130340000     14 APOSTROPHE
        OCT 000241205000     13 =                               0248
        OCT 004015442206        12 QUESTION MARK
        OCT 015114452236     11 9                               0250
        OCT 155114462266     10 8                               0251
        OCT 003610441203     07 7                               0252
        OCT 171124462260     06 6                               0253
        OCT 117054261271     05 5                               0254
        OCT 060241137620     04 4                               0255
        OCT 105014462266     03 3                               0256
        OCT 345114462306     02 2                               0257
        OCT 001027760000     01 1                               0258
 PAT    OCT 175014060276     00 0                               0259
 MASK   OCT 1777001777
 MASK2  OCT     770377000000
 BLANK  BCD 100000
 TEN    DEC 10
 WORD   BSS     1
        BSS     1
        BSS     1
 T      BSS     1
```

****
*   DISPLA (7090)   *
****
(PAGE 8)

```
      POINT BSS       1
      IOC   IOCD      POINT,,1
            END
```

```
*       DIVIDE (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0087
*       FAP
*DIVIDE
        COUNT   150
        LBL     DIVIDE
        ENTRY   DIVIDE (X,LX,XDVSR,XDVDED)
*
*                    ----ABSTRACT----
*
* TITLE - DIVIDE
*         DIVIDE A FLOATING VECTOR BY A CONSTANT
*
*               DIVIDE FORMS A VECTOR EQUAL TO A GIVEN VECTOR DIVIDED
*               BY A FLTG CONSTANT.  OUTPUT MAY REPLACE INPUT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 23 REGISTERS
* SPEED      -      7090    709
*               34 + (19   OR  24)*LX  MACHINE CYCLES, LX = VECTOR LENGTH
*
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL DIVIDE(X,LX,XDVSR,XDVDED)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLTG VECTOR
*
*    LX         SHOULD EXCEED ZERO
*
*    XDVSR      IS A NON-ZERO FLTG QUANTITY. EQUIVALENCE(XDVSR,SOME X(I))
*                  IS PERMITTED.
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LX LSTHN 1 OR XDVSR=0.
*
*    XDVDED(I) I=1...LX  HAS VALUES = X(I)/XDVSR.
*               EQUIVALENCE (XDVDED,X) IS PERMITTED.
*
*               THE DIVISOR USED IS ALWAYS THE INITIAL VALUE OF XDVSR.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)=1.,2.,3.,4.  U=0.0  V=0.0
*    USAGE   -        CALL DIVIDE(X,4,2.,Y)
*                     CALL DIVIDE(X,1,2.,Z)
*                     CALL DIVIDE(X,0,2.,U)
*                     CALL DIVIDE(X,1,0.,V)
*                     CALL DIVIDE(X,4,X(2),X)
*    OUTPUTS - Y(1...4)=.5,1.0,1.5,2.0  Z=0.5
*               U=V=0.0  (NO OUTPUT CASES)  X(1...4)=.5,1.0,1.5,2.0
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR     0                   XR4
        BCI     1,DIVIDE
* ONLY ENTRY.   DIVIDE (X,LX,XDVSR,XDVDED)
DIVIDE SXD      DIVIDE-2,4
 K1     CLA     1,4
        ADD     K1                  A(X)+1
        STA     GET
        CLA     4,4
        ADD     K1                  A(XDVDED)+1
        STA     STORE
        CLA*    3,4                 XDVSR
        TZE     LEAVE
        STO     TEMP
        CLA*    2,4                 LX
```

```
************************
*    DIVIDE            *
************************
                (PAGE  2)
```


```
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
 * DIVISION LOOP
 GET    CLA     **,4                **=A(X)+1
        FDP     TEMP
 STORE  STQ     **,4                **=A(XDVDED)+1
        TIX     GET,4,1
 * EXIT
 LEAVE  LXD     DIVIDE-2,4
        TRA     5,4
 TEMP   PZE     **,**,**            =DIVISOR
        END
```

```
************************
*    DIVK              *
************************
 REFER TO
     ADDK


************************
*    DIVKS             *
************************
 REFER TO
     ADDK


************************
*    DO (PSEUDO ENTRY) *
************************
 REFER TO
     SEVRAL
```


```
************************
*    DIVK              *
************************
 REFER TO
     ADDK


************************
*    DIVKS             *
************************
 REFER TO
     ADDK


************************
*    DO (PSEUDO ENTRY) *
************************
 REFER TO
     SEVRAL
```

```
*      DOTJ  (SUBROUTINE)                10/2/64   LAST CARD IN DECK IS NO. 0142
*      FAP
*DOTJ
       COUNT   100
       LBL     DOTJ
       ENTRY   DOTJ   (LXY,IDX,X,IDY,Y,DOT,ADD,ORDER)
*
*                    ----ABSTRACT----
*
* TITLE - DOTJ
*       VECTOR DOT PRODUCT WITH ARBITRARY INCREMENTS
*
*             DOTJ EVALUATES THE FORMULAE
*
*               DOT  =  X(1)*Y(1) + X(1+IDX)*Y(1+IDY)
*                               + X(1+2*IDX)*Y(1+2*IDY) + ....          (1)
*
*             OR
*
*               DOT  =  X(1)*Y(1+(LXY-1)*IDY) + ...
*                               + X(1+(LXY-1)*IDX)*Y(1)                 (2)
*
*             FOR LXY TERMS OF X AND Y.  THE INCREMENTS IDX AND IDY
*             ARE INPUT PARAMETERS.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 59 REGISTERS
* SPEED     - ABOUT 18*LXY + 72 MACHINE CYCLES ON THE 7090.
* AUTHOR    - R.A. WIGGINS     3/63
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL DOTJ  (LXY,IDX,X,IDY,Y,DOT,ADD,ORDER)
*
* INPUTS
*
*    LXY       IS THE NUMBER OF TERMS IN X AND Y THAT ARE TO BE
*                MULTIPLIED.
*              MUST BE GRTHN=  1
*
*    IDX       IS THE INCREMENT FOR X AS ILLUSTRATED IN THE ABSTRACT.
*              MUST BE GRTHN=  0
*
*    X(I)      I=1,...,(LXY-1)*IDX+1  IS THE X VECTOR.
*
*    IDY       IS THE INCREMENT FOR Y AS ILLUSTRATED IN THE ABSTRACT.
*              MUST BE GRTHN=  1
*
*    Y(I)      I=1,...,(LXY-1)*IDY+1  IS THE Y VECTOR.
*
*    ADD       IS GRTHN ZERO THE INPUT VALUE OF DOT IS ADDED TO THE
*                DOT PRODUCT
*              IF LSTHN=ZERO, DOT IS CLEARED BEFORE THE PRODUCT IS FOUND
*
*    ORDER     IF GRTHN ZERO FORMULA (1) OF THE ABSTRACT IS EVALUATED.
*              IF LSTHN= ZERO FORMULA (2) OF THE ABSTRACT IS EVALUATED.
*
* OUTPUTS
*
*    DOT       IS THE DOT PRODUCT OF X AND Y AS DEFINED IN THE
*                ABSTRACT.
*
* EXAMPLES
*
* 1. INPUTS  - LXY=2 IDX=1 ADD=0  X(1...2)=1.,2.  IDY=2
*              Y(1...3) = 1.,2.,3.  ORDER = 1.
*    OUTPUTS - DOT = 7.
*
* 2. INPUTS  - LXY=2  IDX=3 X(1...4) = 1.,2.,3.,4.  ADD=1.  DOT=2.
*                     IDY=1 Y(1...2) = 1.,2.5  ORDER=1.
```

```
      *    OUTPUTS - DOT = 13.
      *
      * 3. INPUTS  - SAME AS EXAMPLE 2. EXCEPT ORDER=-1.
      *    OUTPUTS - DOT=8.5
      *
      * 4. INPUTS  - LXY=1  IDX=4 X(1)=2. IDY=7  Y(1)=3.  ADD=1.  DOT=2.
      *              ORDER=1.
      *    OUTPUTS - DOT = 8.
      *
      * PROGRAM FOLLOWS BELOW
      *
       XR1   PZE
       XR2   PZE
       XR4   PZE
             BCI    1,DOTJ
       DOTJ  SXD    XR4,4
             SXD    XR1,1
             SXD    XR2,2
             LDQ    =0
             CLA*   7,4                 =ADD
             TLQ    A2
             STZ*   6,4
       A2    CLA*   1,4                 =LXY
             TLQ    A3
             TRA    LV
       A3    SUB    =1B17
             STO    LXY
             CLA*   4,4
             TLQ    A4
             TRA    LV
       A4    STD    T2
             LDQ*   8,4                 =ORDER
             CAL*   2,4                 =IDX
             TQP    A5
             STO    IDX
             SUB    =0100000000000
             STD    T1
             LDQ    IDX
             MPY    LXY
             ARS    1
             PAX    ,1
             TRA    A6
       A5    STD    T1
             AXT    0,1
       A6    CAL    3,4                 =ADR(X)
             STA    X
             CAL    5,4                 =ADR(Y)
             STA    Y
             CAL    6,4                 =ADR(DOT)
             STA    DOT
             STA    DOT+1
             AXT    0,2
             LXD    LXY,4
             TXI    *+1,4,1
       X     LDQ    **,1                **=ADR(X)
       Y     FMP    **,2                **=ADR(Y)
       DOT   FAD    **                  **=ADR(DOT)
             STO    **                  **=ADR(DOT)
       T1    TXI    *+1,1,**
       T2    TXI    *+1,2,**
             TIX    X,4,1
       LV    LXD    XR1,1
             LXD    XR2,2
             LXD    XR4,4
             TRA    9,4
       IDX   PZE                        DECREMENT CONTAINS IDX
       LXY   PZE                        DECREMENT CONTAINS LXY-1
             END
```

```
*     DOTP  (SUBROUTINE)                  9/29/64   LAST CARD IN DECK IS NO. 0146
*     LABEL
CDOTP
      SUBROUTINE DOTP (NRA,NCA,AA,NRB,NCB,BB,IRB,ICB,DOT,ORDER)
C
C                    ----ABSTRACT----
C
C TITLE - DOTP
C        DISPLACED DOT PRODUCT OF 2-DIMENSIONAL ARRAYS
C
C              DOTP FINDS THE DISPLACED DOT PRODUCT OF TWO RECTANGULAR
C              ARRAYS A(I,J)  I=1,...,NRA  J=1,...,NCA  AND  B(I,J)
C              I=1,...,NRB  J=1,...,NCB ACCORDING TO THE FORMULAE
C
C                             M      M
C                    DOT  =  SUM ( SUM ( A(I1,J1)*B(I+IRB,J+ICB) ) )
C                            I=-M  J=-M
C
C              WHERE IF
C                    ORDER= 1. I1=I  J1=J
C                    ORDER= 2. I1=NRA-I+1  J1=J
C                    ORDER=-1. I1=I  J1=NCA-J+1
C                    ORDER=-2. I1=NRA-I+1  J1=NCA-J+1
C              AND
C                    M IS GRTHN MAX(NRA,NCA,NRB,NCB)  (A AND B ARE
C                      CONSIDERED TO BE ZERO WHEN THE SUMMATION IS
C                      OUTSIDE THE RANGE OF DEFINITION)
C                    NRA,NCA,NRB,NCB,IRB,ICB,  AND ORDER ARE INPUT
C                      PARAMETERS.
C
C              DOTP EXITS WITH NO COMPUTATION IF ILLEGAL PARAMETERS
C              ARE FOUND.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 264  REGISTERS
C SPEED      - ABOUT  .000029*NRA*NCA + .000190*NCA + .00078 SECONDS
C                ON THE 7094 MOD 1.
C AUTHOR     - R.A. WIGGINS   MAY,1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - DOTJ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL DOTP  (NRA,NCA,AA,NRB,NCB,BB,IRB,ICB,DOT,ORDER)
C
C INPUTS
C
C   NRA        NUMBER ROWS IN A.
C              MUST EXCEED 0
C
C   NCA        NUMBER COLUMNS IN A.
C              MUST EXCEED 0
C
C   AA(L)      L=1,...,NRA*NCA CONTAINS A(I,J) I=1,...,NRA  J=1,...,NCA
C                STORED CLOSELY PACKED.
C
C   NRB        NUMBER ROWS IN B
C              MUST EXCEED 0
C
C   NCB        NUMBER COLUMNS IN B
C              MUST EXCEED ZERO
C
C   BB(L)      L=1,...,NRB*NCB CONTAINS B(I,J) I=1,...,NRB  J=1,...,NCB
C                STORED CLOSELY PACKED.
C
C   IRB        DEFINES THE DISPLACEMENT ALONG THE COLUMNS OF A WITH
C                RESPECT TO B BEFORE THE PRODUCT IS TAKEN.
C              MAY BE ANY VALUE.
C
C   ICB        DEFINES THE DISPLACEMENT ALONG THE ROWS OF A WITH RESPECT
C                TO B BEFORE THE PRODUCT IS TAKEN.
```

```
C               MAY BE ANY VALUE.
C
C    ORDER      DEFINES THE TYPE OF REVERSAL OF A THAT IS MADE BEFORE
C                 THE PRODUCT IS FOUND  (SEE ABSTRACT)
C               = 1. IMPLIES NO REVERSAL
C               = 2. IMPLIES COLUMN REVERSAL
C               =-1. IMPLIES ROW REVERSAL
C               =-2. IMPLIES ROW AND COLUMN REVERSAL
C
C OUTPUTS
C
C    DOT        CONTAINS THE DOT PRODUCT EVALUATED AS DEFINED IN THE
C                 ABSTRACT.
C
C EXAMPLES
C
C 1. INPUTS  - NRA=4  NCA=4  NRB=4  NCB=4  IRB=0  ICB=0  ORDER=1.
C               AA(1...16) = 1.0,1.1,1.2,1.1  BB(1...16)=1.0,0.9,0.9,0.8,
C                            1.2,1.3,1.4,1.3              0.9,0.8,0.7,0.6,
C                            1.2,1.5,1.5,1.3              0.7,0.7,0.5,0.4,
C                            1.1,1.3,1.2,1.0              0.6,0.5,0.2,0.3
C               WHERE AA AND BB ARE STORED BY COLUMNS.
C    OUTPUTS - DOT = 12.84
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ORDER=2.
C    OUTPUTS - DOT = 12.95
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ORDER=-1.
C    OUTPUTS - DOT = 12.87
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ORDER=-2.
C    OUTPUTS - DOT = 13.07
C
C 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IRB=-2  ICB=-2  ORDER=-2.
C    OUTPUTS - DOT = 4.17
C
C 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IRB=2  ICB=2
C    OUTPUTS - DOT = 1.57
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(2),BB(2)
      DOT=0.
      J=1
C     I1 IS VECTOR INDEX OF THE FIRST POINT IN BB.
      I1=XMAXOF(1,1+IRB)+XMAXOF(0,ICB*NRB)
C     I2 IS VECTOR INDEX OF THE LAST POINT IN FIRST COLUMN OF BB.
      I2=XMINOF(NRA,NRA+IRB,NRB,NRB-IRB)+I1-1
C     LC IS THE LENGTH OF ROWS TO BE USED.
      LC=XMINOF(NCA,NCA+ICB,NCB,NCB-ICB)
      IF (XMINOF(I2-I1+1,LC))  100,100,10
 10   CONTINUE
C     JA1+JA2+1 IS VECTOR INDEX OF FIRST POINT IN A FOR ORDER = 1.
      JA1 = XMAXOF(0,-IRB)
      JA2 = XMAXOF(0,-ICB*NRA)
      K1 = ORDER+3.
      IF (K1-5) 20,20,100
 20   GO TO (30,40,50,50,60),K1
 30   J1 = (NCA-LC+1)*NRA-JA1-JA2
      J=-1
      GO TO 70
 40   J1 = (NCA-LC)*NRA+JA1-JA2+1
      GO TO 70
 50   J1 = JA1+JA2+1
      GO TO 70
 60   J1 = NRA+JA2-JA1
      J=-1
 70   CONTINUE
      DO 90  I=I1,I2
      CALL DOTJ (LC,NRA,AA(J1),NRB,BB(I),DOT,1.,ORDER)
 90   J1=J1+J
 100  RETURN
      END
```

```
************************
*    DPRESS            *
************************
 REFER TO
     BOOST
```



```
************************
*    DPRESS            *
************************
 REFER TO
     BOOST
```

```
*       DSPFMT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0312
*       FAP
*DSPFMT
        COUNT   310
        LBL     DSPFMT
        ENTRY   DSPFMT (CNTHOL,IORGX,IORGY,FMTEND,FMT)
*
*                    ----ABSTRACT----
*
* TITLE - DSPFMT
*        VARIABLE ORIGIN FORMAT GENERATOR FOR SCOPE SUBROUTINE DISPLA
*
*              DSPFMT SETS UP A FORMAT FOR THE SUBROUTINE DISPLA WHICH
*              ALLOWS THE USE OF A VARIABLE ORIGIN FOR THE ALPHANUMERIC
*              CHARACTERS WHICH APPEAR ON THE SCOPE.
*
* LANGUAGE  - FAP  SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 194 REGISTERS
* SPEED     -
* AUTHOR    - S.M. SIMPSON,  NOVEMBER, 1961
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL DSPFMT(CNTHOL,IORGX,IORGY,FMTEND,FMT)
*
* INPUTS
*
*    CNTHOL    IS AN ALPHANUMERIC WORD CONTAINING PRECISELY 3 HOLLERITH
*                LEFT ADJUSTED CHARACTERS.  THESE ARE THE CHARACTERS
*                USED TO CONTROL THE PLOTTING MODE OF DISPLA.
*
*    IORGX     IS THE X ORIGIN IN SCOPE UNITS.
*              IS FORTRAN II INTEGER.
*              MUST BE GRTHN=0, LSTHN 1024
*
*    IORGY     IS THE Y ORIGIN IN SCOPE UNITS.
*              IS FORTRAN II INTEGER.
*              MUST BE GRTHN=0, LSTHN 1024
*
*    FMTEND(I) I=1,0,-1,...  IS AN ARBITRARILY LONG VECTOR OF
*                HOLLERITH CHARACTERS (6 PER WORD) THAT COMPLETES
*                THE FORMAT CONTROLLING DISPLA.
*              DOES NOT INCLUDE THE RIGHT PARENTHESIS.
*              IS TERMINATED BY A FENCE (OCT 777777777777)
*              MAY BE MOST EASILY SET UP BY USING A HOLLERITH
*                ARGUMENT IN THE CALLING SEQUENCE.  THEN FORTRAN
*                TAKES CARE OF THE ORDERING AND THE FENCE.
*
* OUTPUTS
*
*    FMT(I)    I=1,2,...  IS THE HOLLERITH VECTOR OF THE COMPLETED
*                FORMAT.
*              IS OF LENGTH OF FMTEND PLUS THREE WORDS.
*
* EXAMPLES
*
* 1. INPUTS  - CNTHOL = 3H2SH  IORGX=128  IORGY=1000
*    USAGE   -       CALL DSPFMT (CNTHOL,IORGX,IORGY,7H2I6,3A6,FMT)
*    OUTPUTS - FMT(1...5) = 6H(12H2SH128,1000,2I6,3A6)
*
* 2. USAGE   -       CALL DSPFMT (3H2SH,10,2,3H4A6,FMT)
*    OUTPUTS - FMT(1...3) =  6H(8H2SH10,2,4A6)
*
        PZE
        BCI     1,DSPFMT
DSPFMT  SXD     *-2,4
        SXA     SV2,2
        SXA     SV1,1
*SET FMTEND
        CAL     4,4
```

```
       SUB     =1B35
       STA     ST50
*INITIAL STUFF ROUTINE FOR FIRST HOLERITH (LEFT PAREN)
       CLA     K1
       STA     KAY
       STA     EL
       CLA     5,4           FMT
       STA     ST10
*PUT IN LEFT PAREN
       CLA     LPRN
       TSX     STUFF,2
*GO FIND OUT HOW MANY DIGITS IN IRGX AND IN IRGY
       CLA*    2,4
       TSX     BCI,2
       STO     NDX
       CLA*    3,4
       TSX     BCI,2
       STO     NDY
*THE NUMBER OF DIGITS IN HOLERITH FIELD OF FMT IS
*      = 3(FOR CONTROL) + NDX + NDY + 2(FOR COMMAS)
       ADD     NDX
       ADD     K5
*SPREAD IT OUT IN HOLERITH AND STUFF INTO FORMAT
       ALS     18
       TSX     BCI,2
       TSX     STORN,1
*THEN PUT IN H
       CLA     AITCH
       TSX     STUFF,2
*NOW SET UP AND INSERT 3 CONTROL HOLERITH,STUFF SAVES 7 MQ)
       LDQ*    1,4
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
*NEXT SET UP AND PUT IN IRGX
       CLA*    2,4
       TSX     BCI,2
       TSX     STORN,1
*THEN A COMMA
       CLA     COMMA
       TSX     STUFF,2
*THEN IRGY
       CLA*    3,4
       TSX     BCI,2
       TSX     STORN,1
*THEN ANOTHER COMMA
       CLA     COMMA
       TSX     STUFF,2
*NOW KEEP PUTTING IN THE FORMAT END TILL HIT FENCE
       AXT     -1,1
  ST50 CLA     **,1          **=FMTEND
       CAS     FENCE
       TRA     *+2
       TRA     ST60          FENCE HIT, GO WIND UP
*PUT IN ALL SIX
       XCA
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       LGL     6
       TSX     STUFF,2
       TIX     ST50,1,1
*WHEN FENCE HIT FILL IN RIGHT PAREN
 ST60  CLA     RPRN
       TSX     STUFF,2
```

```
*NOW FILL IN REMAINDER OF REGISTER WITH BLANKS(UNTIL KAY=1)
 ST70  CLA     KAY
       CAS     K1
       TRA     *+2
       TRA     LV
*(USING BLANK IN COMMA SINCE UNSURE OF BCI 1,(6 BLANKS))
       CLA     COMMA
       LRS     6
       TSX     STUFF,2
       TRA     ST70
*EXIT
  LV   LXD     DSPFMT-2,4
 SV2   AXT     **,2
  SV1  AXT     **,1
       TRA     6,4
*CONSTANTS
 NDX   PZE                     NO. DIGITS
 NDY   PZE
 LPRN  BCI     1,      (
 RPRN  BCI     1,      )
 COMMA BCI     1,      ,
 FENCE OCT     777777777777
 AITCH BCI     1,      H
  K5   PZE     5
*
*INTERNAL SUBROUTINES
*
*STORN STUFFS NO. FROM BCI INTO FMT BLOCK (IGNORE LEADING ZEROES)
* A    TSX STORN,1
* A+1  RETURN
 STORN CLA     B4
       TNZ     ST400
       CLA     B3
       TNZ     ST300
       CLA     B2
       TNZ     ST200
       TRA     ST100
 ST400 CLA     B4
       TSX     STUFF,2
 ST300 CLA     B3
       TSX     STUFF,2
 ST200 CLA     B2
       TSX     STUFF,2
 ST100 CLA     B1
       TSX     STUFF,2
       TRA     1,1
* CALLING SEQUENCE FOR STUFF
*         (INTERNAL SUBROUTINE TO DSPFMT)
* A    TSX     STUFF,2
* A+1  RETURN
*
*      STUFF STORES BITS 30-35 OF THE AC AS FOLLOWS.
*     L K=1 K=2 K=3 K=4 K=5 K=6
*
*     ETC
*
*     3
*     2
* FMT 1 FMT(1,1)
*
*FL AND KAY MUST BE SET TO 1 BY DSPFMT, AND
* FMT STORED IN ADDRESS OF ST10, BEFORE STUFF
* IS FIRST USED.
*
*      STUFF SHIFTS THE AC APPROPRIATELY, ADDS FMT(L),
* AND STORES  RESULT AT FMT(L). IT THEN INCREMENTS
* K, AND IF K GRTR THAN 6, INCREMENTS L AND RESETS K=1.
* FOR K=1 IT FIRST CLEARS FMT(L).
 STUFF ANA     KANA            GET RID OF ANY OTHER BITS
       STO     TEMP            AND STORE
       SXA     STLV,2
       STQ     SVMQ
       LXA     EL,2            L-1 TO XR2
       TXI     *+1,2,-1
*IF K=1 CLEAR FMT(L)
```

****
*   DSPFMT   *
****
(PAGE 4)

```
        CLA     KAY
        CAS     K1
        TRA     *+2
        TRA     ST3
        TRA     ST4
  ST3   STZ*    ST10
 ST4    CLA     K6
        SUB     KAY
        XCA
        MPY     K6
        XCA
        STA     ST9
        LDQ     TEMP            GET BITS BWS ZEROES
        LGL     36              IN MQ
 ST9    LGL     **              **=6(6-K)
 ST10   ACL     **,2            **=FMT  XR2=L-1
        SLW*    ST10
        CLA     KAY
        ADD     K1
        STO     KAY
        CAS     K6
        TRA     ST12
        NOP
        TRA     STLV
 ST12   CLA     K1              RESET K TO 1
        STO     KAY             AND L TO L+1
        ADD     EL
        STO     EL
 STLV   AXT     **,2
        LDQ     SVMQ
        TRA     1,2
 TEMP   PZE     **              **=6 BITS TO BE STUFFED
 KAY    PZE     **              **=K
 EL     PZE     **              **=L
 K6     PZE     6
 K1     PZE     1
 KANA   OCT     77
 SVMQ   PZE     **
*       INTERNAL SUBROUTINE BCI
*       TSX     BCI,2           WITH FORTRAN INTEGER IN ACRETURN WITH
*                               BCI IN B1,B2,B3,B4, RIGHT ADJUSTED
*                               AC = NO DIGITS ON EXIT (ZERO)=1)
  BCI   SXA     BCIR,2
        STZ     B1
        STZ     B2
        STZ     B3
        STZ     B4
        ARS     18
        AXT     0,2
        SUB     =1000
        TMI     *+2
        TXI     *-2,2,1
        ADD     =1000
        SXA     B4,2
        AXT     0,2
        SUB     =100
        TMI     *+2
        TXI     *-2,2,1
        ADD     =100
        SXA     B3,2
        AXT     0,2
        SUB     =10
        TMI     *+2
        TXI     *-2,2,1
        ADD     =10
        SXA     B2,2
        STA     B1
*FIGURE OUT HOW MANY DIGITS
        CLA     K1
        NZT     B4
        TRA     *+3
        ADD     K3
        TRA     BCIR
        NZT     B3
        TRA     *+3
```

```
************************
*    DSPFMT          *
************************
(PAGE  5)
```


```
           ADD     K2
           TRA     BCIR
           ZET     B2
           ADD     K1
     BCIR  AXT     **,2
           TRA     1,2
       B4  PZE     **                MOST SIG DIG
       B3  PZE     **
       B2  PZE     **
       B1  PZE     **                LEAST SIG DIG
       K2  PZE     2
       K3  PZE     3
           END
```

```
***********************
*    DUBLL            *
***********************
 REFER TO
     DUBLX
```


```
***********************
*    DUBLL            *
***********************
 REFER TO
     DUBLX
```

```
*      DUBLX (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0128
*      FAP
*DUBLX
       COUNT   100
       LBL     DUBLX
       ENTRY   DUBLX (IX,LX)
       ENTRY   DUBLL (X,LX)
       ENTRY   HALVX (IX,LX)
       ENTRY   HALVL (X,LX)
*
*                    ----ABSTRACT----
*
* TITLE - DUBLX , WITH SECONDARY ENTRY POINTS DUBLL, HALVX, HALVL.
*       FAST DOUBLING OR HALVING OF A VECTOR (FIXED OR FLOATING)
*
*             DUBLX DOUBLES THE MAGNITUDES OF THE NUMBERS IN A FIXED
*             POINT VECTOR. OVERFLOW IS NOT CHECKED.
*
*             DUBLL DOUBLES THE MAGNITUDES OF THE NUMBERS IN A FLOATING
*             POINT VECTOR.
*
*             HALVX HALVES THE MAGNITUDES (WITHOUT ROUNDING) OF THE
*             NUMBERS IN A FIXED POINT VECTOR.
*
*             HALVL HALVES THE MAGNITUDES OF THE NUMBERS IN A FLOATING
*             POINT VECTOR.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 45 REGISTERS
* SPEED     - 10N MACHINE CYCLES (N= LENGTH OF VECTOR)
* AUTHOR    - S.M. SIMPSON
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL DUBLX (IX,LX)
*     CALL DUBLL ( X,LX)
*     CALL HALVX (IX,LX)
*     CALL HALVL ( X,LX)
*
* INPUTS
*
*    X(I)      I=1...LX IS VECTOR OF FLOATING POINT NUMBERS.
*
*    IX(I)     I=1...LX IS VECTOR OF FIXED POINT NUMBERS.
*
*    LX        IS FORTRAN II INTEGER
*              MUST EXCEED ZERO
*                (LX=0 IS TREATED AS LX=1, LX NEG AS LX POS)
*
* OUTPUTS
*
*    X(I)      I=1...LX IS INPUT VECTOR HALVED OR DOUBLED.
*
*    IX(I)     I=1...LX IS INPUT VECTOR HALVED OR DOUBLED.
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...3) =  1,-4,9    LX=3
*    OUTPUTS - DUBLX     IX(1...3) = 2,-8,18
*              HALVX     IX(1...3) = 0,-2,4
*
* 2. INPUTS  - IX(1...3) = OCT 000001000000,-000004000000,000011000000
*              LX=3
*    OUTPUTS - DUBLX     IX(1...3) = OCT 000002000000, -000010000000,
*                                        000022000000
*              HALVX     IX(1...3) = OCT 000000400000, -000002000000,
*                                        000004400000
*
* 3. INPUTS  - X(1...3) = 1.,-4.,9.  LX=3
*    OUTPUTS - DUBLL   X(1...3) = 2.,-8.,18.
```

```
*              HALVL   X(1...3) = .5,-2.,4.5
*
* 4. INPUTS  - X(1) = 3.17   LX=1
*    OUTPUTS - DUBLL   X(1) = 6.34
*              HALVL   X(1) = 1.585
*
        PZE
        BCI    1,DUBLX
  DUBLX CLA    K3           SET ALS 1
        STO    D6
        TRA    D1
  HALVX CLA    K5           SET ARS 1
        STO    D6
   D1   CLA    K2           SET CLA**,4
        STO    LOOP             AND
        CLA    K4           STO**,4
        STO    D7
        TRA    D5
  DUBLL CLA    K8           SET ACL K10
        STO    D6
        TRA    D2
  HALVL CLA    K9           SET SUB K10
        STO    D6
   D2   CLA    K6           SET CAL**,4
        STO    LOOP            AND
        CLA    K7           SLW **,4
        STO    D7
   D5   SXD    DUBLX-2,4            SAVE XR4
        CLA    1,4                  SET Y+1
        ADD    K1
        STA    LOOP
        STA    D7
        CLA*   2,4            SET XR4 FOR N DATA
        PDX    0,4
*                                 ** BELOW=Y+1
*                             DUBLX  HALVX  DUBLL  HALVL
  LOOP  NOP                 CLA**,4 CLA**,4 CAL**,4 CAL**,4
        TZE    D7+1
   D6   NOP                 ALS 1   ARS 1  ACL K10 SUB K10
   D7   NOP                 STO**,4 STO**,4 SLW**,4 SLW**,4
        TIX    LOOP,4,1
        LXD    DUBLX-2,4
        TRA    3,4                  EXIT
   K1   PZE    1
   K2   CLA    **,4
   K3   ALS    1
   K4   STO    **,4
   K5   ARS    1
   K6   CAL    **,4
   K7   SLW    **,4
   K8   ACL    K10
   K9   SUB    K10
   K10  OCT    001000000000
        END
```

```
************************
*    ENDFIL            *
************************
 REFER TO
     REREAD
```

```
************************
*    EOFSET            *
************************
 REFER TO
     REREAD
```


```
************************
*    ENDFIL            *
************************
 REFER TO
     REREAD
```

```
************************
*    EOFSET            *
************************
 REFER TO
     REREAD
```

```
*       EXCHVS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0083
*       FAP
*EXCHVS
        COUNT   100
        LBL     EXCHVS
        ENTRY   EXCHVS (LXY,X,Y)
*
*                     ----ABSTRACT----
*
* TITLE - EXCHVS
*         EXCHANGE ANY TWO VECTORS
*
*               EXCHVS EXCHANGES ANY TWO VECTORS
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 22   REGISTERS
* SPEED       - 31 + 10*LXY MACHINE CYCLES,  LXY = LENGTH OF VECTORS
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL EXCHVS(LXY,X,Y)
*
* INPUTS
*
*    LXY        IS COMMON LENGTH OF THE TWO VECTORS. SHOULD EXCEED 0
*    X(I)       I=1...LXY  IS A VECTOR IN ANY MODE
*    Y(I)       I=1...LXY  IS A VECTOR IN ANY MODE
*
* OUTPUTS       STRAIGHT RETURN WITH NO ACTION IF LXY  LSTHN 1
*
*    X(I)       I=1...LXY  IS  X(I) = INPUT VALUE OF Y(I)
*
*    Y(I)       I=1...LXY  IS  Y(I) = INPUT VALUE OF X(I)
*
*               EQUIVALENCE (X,Y)  IS PERMITTED (SAME THING AS NO ACTION)
*
* EXAMPLES
*
* 1. INPUTS   - X(1...3) = 1., 2., 3.    IX1(1...3)= 2, 4, 6
*               IX2 = 4   IX3 = 5   IX4 = 0   IX5 = 1
*    USAGE    -       CALL EXCHVS(3, X, IX1)
*                     CALL EXCHVS(1,IX2,IX3)
*                     CALL EXCHVS(0,IX4,IX5)
*                     CALL EXCHVS(3, X, X)
*    OUTPUTS - X(1...3) = 2, 4, 6   IX1(1...3) = 1., 2., 3.
*               IX2 = 5   IX3 = 4
*               IX4 = 0   IX5 = 1   (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,EXCHVS
* ONLY ENTRY.   EXCHVS(LXY,X,Y)
EXCHVS SXD      EXCHVS-2,4
        CLA     2,4
        ADD     K1             A(X)+1
        STA     GET1
        STA     STORE1
        CLA     3,4
        ADD     K1             A(Y)+1
        STA     GET2
        STA     STORE2
 K1     CLA*    1,4            LXY
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
* EXCHANGE LOOP
```

****
*   EXCHVS   *
****
(PAGE 2)

```
      GET1   CLA    **,4          ** = A(X)+1
      GET2   LDQ    **,4          ** = A(Y)+1
     STORE2  STO    **,4          ** = A(Y)+1
     STORE1  STQ    **,4          ** = A(X)+1
             TIX    GET1,4,1
     * EXIT
      LEAVE  LXD    EXCHVS-2,4
             TRA    4,4
             END
```

```
*      EXPAND (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0379
*      FAP
*EXPAND
       COUNT    500
       LBL      EXPAND
       ENTRY    EXPAND (X, LX, MLPLYR, XPNDED, LXPNDD)
*
*
*                    ----ABSTRACT----
*
* TITLE - EXPAND
*        HI-SPEED EXPANSION OF A VECTOR UNDER CUBIC INTERPOLATION
*
*               EXPAND  TAKES A VECTOR OF LENGTH  LX  AND CREATES AN
*               OUTPUT VECTOR OF LENGTH  M*(LX-1)+1  IN WHICH THE FIRST,
*               THE  (M+1)-TH,  ...,  THE  (M*(LX-1)+1)-TH  VALUES ARE
*               THE SAME AS THOSE OF THE INPUT VECTOR, AND IN WHICH THE
*               REMAINING VALUES ARE INTERPOLATED WITH UNIFORM SPACING
*               BY EXACT FITTING CUBICS, EXCEPT THAT QUADRATICS ARE USED
*               AT THE TWO ENDS.  THE EXPANSION FACTOR  M  IS AN INPUT
*               PARAMETER GREATER THAN ZERO.  LINEAR INTERPOLATION IS
*               USED IF  LX = 2 .  A PURE COPY OCCURS IF  LX = 1  OR IF
*               M = 1 .
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE     - 189 REGISTERS
* SPEED       - ON THE 7090, EXPAND TAKES ABOUT
*                  360+10*LX+800*(M-1)+94*(M-1)*LX  MACHINE CYCLES,
*               PROVIDED  LX  EXCEEDS  3 .
* AUTHOR      - S.M. SIMPSON,  JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - INTOPR
*       AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     CALL EXPAND (X, LX, MLPLYR, XPNDED, LXPNDD)
*
*
* INPUTS
*
*    X(I)       I=1...LX  IS THE VECTOR TO BE EXPANDED.
*
*    LX         SHOULD EXCEED ZERO.
*
*    MLPLYR     IS THE EXPANSION FACTOR,  M,  DESCRIBED IN ABSTRACT.
*               SHOULD EXCEED ZERO.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LX  LSTHN=  0,  OR IF
*                  LX  EXCEEDS  1  BUT  MLPLYR  IS  LSTHN=  0 .  FOR
*                  LX = 1  XPNDED(1)  IS SET  =  X(1)  REGARDLESS OF
*                  MLPLYR.
*
*    XPNDED(I) I=1...LXPNDD  IS THE EXPANDED VECTOR DESCRIBED IN
*                  ABSTRACT.  IF  MLPLYR = 1,  XPNDED(I) = X(I).
*
*    LXPNDD     = MLPLYR*(LX-1)+1  IF  LX  EXCEEDS  1 .
*               = 1                IF  LX  =  1 .
*
*
* EXAMPLES
*
* 1. THIS EXAMPLE EXERCISES ALL PATHS IN  EXPAND,  EXCLUDING ILLEGAL  LX
*               AND  MLPLYR  VALUES, ON A SIMPLE LINEAR  X  SERIES.
*
*    INPUTS  - X(1...5) = 0.,6.,12.,18.,24.  AND LET  XPNDED  BE A THREE
*                  DIMENSIONAL ARRAY OF DIMENSION  XPNDED(16,5,3), ALL
*                  ELEMENTS OF WHICH HAVE BEEN SET TO THE CONSTANT
*                  VALUE = -9.0 .
*    USAGE   -        DIMENSION LXPNDD(5,3)
*                     DO 10  MLPLYR = 1,3
```

```
*                 DO 10  LX = 1,5
*             10  CALL EXPAND (X, LX, MLPLYR, XPNDED (1, LX, MLPLYR),
*                1             LXPNDD(LX, MLPLYR))
*    OUTPUTS - XPNDED(1...16,1,1) = 0.,-9.,-9.,...,-9.
*              XPNDED(1...16,2,1) = 0.,6.,-9.,-9.,...,-9.
*              XPNDED(1...16,3,1) = 0.,6.,12.,-9.,-9.,...,-9.
*              XPNDED(1...16,4,1) = 0.,6.,12.,18.,-9.,-9.,...,-9.
*              XPNDED(1...16,5,1) = 0.,6.,12.,18.,24.,-9.,-9.,...,-9.
*              XPNDED(1...16,1,2) = 0.,-9.,...,-9.
*              XPNDED(1...16,2,2) = 0.,3.,6.,-9.,...,-9.
*              XPNDED(1...16,3,2) = 0.,3.,6.,9.,12.,-9.,...,-9.
*              XPNDED(1...16,4,2) = 0.,3.,...,15.,18.,-9.,...,-9.
*              XPNDED(1...16,5,2) = 0.,3.,...,21.,24.,-9.,...,-9.
*              XPNDED(1...16,1,3) = 0.,-9.,...,-9.
*              XPNDED(1...16,2,3) = 0.,2.,4.,6.,-9.,...,-9.
*              XPNDED(1...16,3,3) = 0.,2.,...,10.,12.,-9.,...,-9.
*              XPNDED(1...16,4,3) = 0.,2.,...,16.,18.,-9.,...,-9.
*              XPNDED(1...16,5,3) = 0.,2.,...,22.,24.,-9.,-9.,-9.
*              LXPNDD(1...5,1...3) = 1,2,3,4,5,,1,3,5,7,9,,1,4,7,10,13
*
* 2. ILLEGAL CALL STATEMENTS
*
*    INPUTS  - SAME AS EXAMPLE 1., EXCEPT LXPNDD = -9
*                 CALL EXPAND (X,0,3,XPNDED(1,1,1),LXPNDD)
*                 CALL EXPAND (X,2,0,XPNDED(1,1,1),LXPNDD)
*                 CALL EXPAND (X,-3,-1,XPNDED(1,1,1),LXPNDD)
*
*    OUTPUTS - XPNDED(1...16,1,1) = -9.,-9.,...,-9.  LXPNDD = -9
*
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS INTOPR(NDATA, YLO, DELY, Y, OPER)
*
       HTR     0                XR1
       HTR     0                XR2
       HTR     0                XR4
       BCI     1,EXPAND
*
* ONLY ENTRY.  EXPAND(X, LX, MLPLYR, XPNDED, LXPNDD)
*
EXPAND SXD     EXPAND-4,1
       SXD     EXPAND-3,2
       SXD     EXPAND-2,4
*
* ADDRESS SETTINGS
*
       CLA     1,4              A(X)
       ADD     K1               A(X)+1
       STA     CLA1
       STA     LDQ1
       SUB     K1               A(X)
       STA     LDQ2
       SUB     K1               A(X)-1
       STA     LDQ3
       SUB     K1               A(X)-2
       STA     LDQ4
       CLA     4,4              A(XPNDED)
       ADD     K1               A(XPNDED)+1
       STA     STO1
       STA     STO2
*
* CHECK OUT LX AND MLPLYR
*
       CLA*    2,4              LX
       STO     LX
       STD     TXL1
       CAS     KD1
       TRA     LXGR1
       TRA     EVEN             (LX = 1)
       TRA     LEAVE
 LXGR1 CLA*    3,4              MLPLYR, CALLED M FOR SHORT
       STO     M
       STD     TXI1
       STD     TXI2
```

EXPAND
(PAGE 3)

```
        CAS     KD1
        TRA     MGR1
        TRA     EVEN            (M = 1)
        TRA     LEAVE
  MGR1  LRS     18
        ORA     OCTK
        FAD     OCTK            M FLOATED
        STO     DELY
*
* INTERPOLATE BETWEEN X(1) AND X(2), LX GRTHN= 2 .
*
  LEFT  CLA     KD3
        STO     NDATA           NDATA = 3
        CLA     KD1
        STO     IXLO            IXLO = IFITLO = NSETS = 1
        STO     IFITLO
        STO     NSETS
        CLA     LX
        SUB     KD2
        TNZ     TSXLFT
        CLA     KD2
        STO     NDATA           RESET NDATA TO 2 FOR LX = 2
TSXLFT  TSX     INTRP,4
*
* INTERPOLATE BETWEEN X(2) AND X(LX-1), PROVIDED LX EXCEEDS 3
*
CENTER  CLA     KD4
        STO     NDATA           NDATA = 4
        CLA     KD2
        STO     IXLO            IXLO = 2   (IFITLO STILL = 1)
        CLA     LX
        SUB     KD3
        STO     NSETS           NSETS = LX-3
        TMI     RIGHT
        TZE     RIGHT
        TSX     INTRP,4
*
* INTERPOLATE BETWEEN X(LX-1) AND X(LX), PROVIDED LX EXCEEDS 2
*
 RIGHT  CLA     KD3
        STO     NDATA           NDATA = 3
        CLA     KD1
        STO     NSETS           NSETS = 1
        CLA     LX
        SUB     KD1
        STO     IXLO            IXLO = LX-1
        SUB     KD1
        STO     IFITLO          IFITLO = LX-2
        TMI     EVEN
        TZE     EVEN
        TSX     INTRP,4
*
* FINALLY INSERT X(1,2,...,LX) INTO XPNDED(1,M+1,...,M*(LX-1)+1)
*    AND COMPUTE AND SET LXPNDD
*
  EVEN  AXT     1,1             I OF X(I)
        AXT     1,2             J OF XPNDED(J)
  CLA1  CLA     **,1            ** = A(X)+1
  STO1  STO     **,2            ** = A(XPNDED)+1
  TXI1  TXI     *+1,2,**        ** = M
        TXI     *+1,1,1
  TXL1  TXL     CLA1,1,**       ** = LX
        LXD     EXPAND-2,4
        CLA     LX
        SUB     KD1
        TNZ     XCA1A
        CLA     KD1
        TRA     STO1A
 XCA1A  XCA
        MPY     M
        ALS     17
        ADD     KD1
 STO1A  STO*    5,4             LXPNDD = 1  (LX=1),  OR = M*(LX-1)+1
*
```

```
* EXIT
*
 LEAVE LXD     EXPAND-4,1
       LXD     EXPAND-3,2
       LXD     EXPAND-2,4
       TRA     6,4
*
* INTERNAL SUBROUTINE INTRP
*
*      LINKAGE XR4, RETURNS TO 1,4
*
*      ASSUMES
*              NDATA  IS SET (= 2, 3, OR 4)
*              IXLO   IS SET (= 1, 2, OR LX-1)
*              IFITLO IS SET (IXLO OR IXLO-1)
*              NSETS  IS SET (= 1 OR LX-3)
*
*      FORMS AND STORES INTERPOLATIONS (EXCLUDING ENDS) BETWEEN
*              X(IXLO)           AND  X(IXLO+1)
*              X(IXLO+1)         AND  X(IXLO+2)
*                               ETC
*              X(IXLO+NSETS-1)   AND  X(IXLO+NSETS)
*      WHERE THE OPERATORS FOR THE FIRST SET ARE FITTED TO
*              X(IFITLO), X(IFITLO+1), ..., X(IFITLO+NDATA-1)
*
 INTRP SXA     INTSV4,4
*
* FOR PURPOSES OF  INTOPR,  THE FIRST DATA POINT IS AT ARGUMENT
*    YLO = 0.0,  THE SECOND AT ARGUMENT  =  DELY  =  M,  THE THIRD AT
*    2M,  ETC.  HENCE THE  Y  FOR WHICH WE WANT AN OPERATOR IS  Y = 1.0
*    (IN THE CASE THAT  IFITLO = IXLO)  OR  Y = M+1  (FOR
*    IFITLO = IXLO-1).  THE TOTAL NUMBER OF DIFFERENT OPERATORS WE WANT
*    IS  M-1 .
*
* INITIALIZE Y, YCOUNT, DECREMENT AT TXL2, AND FRSXR2
*
       LDQ     YLO             (PREPARE FOR IFITLO = IXLO)
       CLA     IXLO
       SUB     IFITLO
       TZE     XCA
       LDQ     DELY
 XCA   XCA
       FAD     K1L
       STO     Y               Y
       CLA     KD1
       STO     YCOUNT          YCOUNT
       CLA     IFITLO
       ADD     NSETS
       SUB     KD1
       STD     TXL2            DECREMENT AT TXL2
       CLA     IXLO
       SUB     KD1
       XCA
       MPY     M
       ALS     17
       ADD     KD2
       STO     FRSXR2          FRSXR2
*
* LOOP FOR SUCCESSIVE Y VALUES BEGINS HERE
*
* ACQUIRE OPERATOR, THEN INITIALIZE XR1,XR2
*
GETOPR STZ     OPER3           (CLEAR FOR CASES
       STZ     OPER4            NDATA = 2 OR 3)
       SXA     GETSV4,4
       TSX     $INTOPR,4
       TSX     NDATA,0
       TSX     YLO,0
       TSX     DELY,0
       TSX     Y,0
       TSX     OPER1,0
GETSV4 AXT     **,4            ** = XR4 PRIOR TO INTOPR
       LXD     IFITLO,1
       LXD     FRSXR2,2
*
```

```
* INNER LOOP OVER THE SETS TO FORM
*
*    X(I)*OPER1 + X(I+1)*OPER2 + X(I+2)*OPER3 + X(I+3)*OPER4
*
*    RESULT IS STORED IN  XPNDED(J)
*    ASSUMES  I  IS IN  XR1,  J  IS IN  XR2
*
*    I  JUMPS BY  1,  J  JUMPS BY  M
*
 EVAL  STZ     SUM
 LDQ1  LDQ     **,1            ** = A(X)+1
       FMP     OPER1
       FAD     SUM
       STO     SUM
 LDQ2  LDQ     **,1            ** = A(X)
       FMP     OPER2
       FAD     SUM
       STO     SUM
 LDQ3  LDQ     **,1            ** = A(X)-1
       FMP     OPER3
       FAD     SUM
       STO     SUM
 LDQ4  LDQ     **,1            ** = A(X)-2
       FMP     OPER4
       FAD     SUM
 STO2  STO     **,2            ** = A(XPNDED)+1
 TXI2  TXI     *+1,2,**        ** = M
       TXI     *+1,1,1
 TXL2  TXL     EVAL,1,**       ** = IFITLO+NSETS-1
*
* RESET FOR ANOTHER Y VALUE AND CHECK COMPLETION
*
       CLA     FRSXR2          INCREASE INITIAL OUTPUT STORAGE BY 1
       ADD     KD1
       STO     FRSXR2
       CLA     Y               INCREMENT Y BY 1.0
       FAD     K1L
       STO     Y
       CLA     YCOUNT          INCREMENT AND CHECK Y COUNTER
       ADD     KD1             (1...M-1)
       STO     YCOUNT
       CAS     M
       HPR     *               (IMPOSSIBLE)
       TRA     INTSV4          DONE
       TRA     GETOPR          MORE Y VALUES
*
* EXIT FROM INTERNAL SUBROUTINE
*
INTSV4 AXT     **,4            ** = XR4 AT START OF INTRP
       TRA     1,4
*
* CONSTANTS
*
 K1    PZE     1
 KD1   PZE     0,0,1
 KD2   PZE     0,0,2
 KD3   PZE     0,0,3
 KD4   PZE     0,0,4
 K1L   DEC     1.0
 YLO   DEC     0.0
 OCTK  OCT     233000000000
*
* TEMPORARIES
*
 IXLO  PZE     0,0,**          ** = 1, 2, OR LX-1
IFITLO PZE     0,0,**          ** = IXLO OR IXLO-1
 NSETS PZE     0,0,**          ** = 1 OR LX-3
 M     PZE     0,0,**          ** = M = MLPLYR
 LX    PZE     0,0,**          ** = LX
 NDATA PZE     0,0,**          ** = 2, 3, OR 4
 DELY  PZE     **,**,**        ** = FLOATF(M)
 Y     PZE     **,**,**        ** = DELY*(IXLO-IFITLO)+1,
*                                       (+2,+3,...,+M-1)
YCOUNT PZE     0,0,**          ** = 1,2,...,M-1
FRSXR2 PZE     0,0,**          ** = M*(IXLO-1)+2  (+3,+4,...,+M)
```

```
      OPER4 PZE      **,**,**                                        0374
      OPER3 PZE      **,**,**                                        0375
      OPER2 PZE      **,**,**                                        0376
      OPER1 PZE      **,**,**                                        0377
      SUM   PZE      **,**,**                                        0378
            END                                                      0379
```

```
*      FACTOR (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0488
*      FAP
*FACTOR
       COUNT   450
       LBL     FACTOR
       ENTRY   FACTOR (SPECT,N,L,WAVE,SPACE)
*
*
*                   ----ABSTRACT----
*
* TITLE - FACTOR
*        FACTOR POWER SPECTRUM TO FIND MINIMUM PHASE WAVELET
*
*              FACTOR USES THE METHOD OF KOLMOGOROV (REF.- 1. ROBINSON,E.
*              A., M.I.T. PH.D. THESIS,GEOPHYSICAL ANALYSIS GROUP REPORT
*              7,1954. 2. SIMPSON ET AL., SCIENTIFIC REPORT NO. 2 OF
*              CONTRACT AF 19(604)7378.) TO FACTOR THE POWER SPECTRUM
*              AND THUS PRODUCE THE MINIMUM PHASE WAVELET.
*              THE RESTRICTIONS ON APPLICABILITY OF THE METHOD REQUIRE
*              THAT THE INPUT SPECTRUM BE NON-NEGATIVE AND NON-ZERO.
*              HENCE SPECT(I), THE INPUT SPECTRUM, IS CHECKED AND ANY
*              VALUES WHICH ARE LESS THAN 10**(-6) OF THE MAXIMUM VALUE
*              OF SPECT(I) ARE SET EQUAL TO 10**(-6) OF THE MAXIMUM.(THIS
*              FEATURE MAY EASILY BE REMOVED FROM THE SYMBOLIC DECK).
*
*              ONE HALF OF THE NATURAL LOG OF THE SPECTRUM IS COMPUTED
*              AND EXPANDED IN A COSINE SERIES. THE COEFFICIENTS OF THE
*              EXPANSION ARE COMPUTED BY TRAPEZOIDAL RULE INTEGRATION
*              (SAME AS TRIGONOMETRIC INTERPOLATION. HENCE THE FIRST AND
*              LAST TERMS IN THE SPECTRUM ARE WEIGHTED BY 1/2 AND THE
*              SUMMATION AND COSINE WEIGHTING ARE DONE SIMULTANEOUSLY
*              BY SUBROUTINE COSP. THE COEFFICIENTS OF THE COSINE
*              EXPANSION ARE TRAN(I), I=1,L.  THE EXPONENTIAL
*
*                                      L
*                     EXP**(TRAN(1)+ SUM(TRAN(I)*(Z**(I-1))))
*                                     I=2
*
*              MUST BE EXPANDED IN A CONTINUED PRODUCT OF POLYNOMIALS IN
*              Z. THE POLYNOMIALS ARE THEN MULTIPLIED OUT AND GROUPED IN
*              THE FORM
*
*                                        L
*                                   P = SUM (W(I)*(Z**(I-1)))
*                                       I=1
*
*              WHERE L IS THE LENGTH OF THE WAVELET, AND W(I) IS THE
*              DESIRED WAVELET.
*
*    PROGRAM NOTES -
*              THE EXPANSION OF THE EXPONENTIAL AND MULTIPLICATION OF
*              THE RESULTING POLYNOMIALS MAY BE SIMPLIFIED BY THE
*              FOLLOWING CONSIDERATIONS - THE EXPONENTIAL MAY BE
*              REPRESENTED AS A CONTINUED PRODUCT OF POLYNOMIALS
*              WHERE THE ITH POLYNOMIAL IS OF THE FORM
*
*                             L-1
*                        P(I)=(SUM( C(I,J)*(Z**I))+ 1)*EXP**(TRAN(1))
*                             I=1
*              WHERE
*                   C(I,J)= (TRAN(1)/1)*(TRAN(2)/2)*.....*(TRAN(I)/(J/I))
*                                                         FOR J=K*I
*                   C(I,J)= 0 FOR  J NOT =K*I
*               THE C(I,0) TERMS ARE 1 FOR ALL I.
*
*              WE ARE ONLY INTERESTED IN THE FIRST L TERMS OF THE WAVELET
*              SO WE NEED ONLY CONSIDER TERMS IN THE POLYNOMIALS WITH
*              EXPONENTS LESS THAN OR =M,M=L-1. WE CAN THEN COMPUTE THE
*              WAVELET COEFFICIENTS BY PARTIAL CONVOLUTION OF THE
*              POLYNOMIAL COEFFICIENTS. THAT IS,
*
*                 WAVE(I)= C(1,J)*C(2,J)*...C(M,J)
*
*              WHERE WAVE(I) IS THE WAVELET, M=L-1, AND THE * SYMBOL
*              DENOTES CONVOLUTION.
```

```
*              IT WILL BE NOTED THAT IF THE CONVOLUTION IS REPRESENTED
*             IN STEPS BY
*                      B(M-1)= C(M-1,J)*C(M,J),   B(K)=C(K,J)*B(K+1)
*             BY CAREFUL INSPECTION OF THE FORM OF THE C(I,J) ONE CAN
*             WRITE DOWN THE B(N) BY INSPECTION FOR N=L/2 (ROUNDED DOWN)
*             +1. THIS CUTS DOWN THE TOTAL LABOR BY NEARLY 1/2.
*                   B(N)= 1,0,0,.....,0,C(N,N),C(N+1,N+1),....,C(M,M)
*               FACTOR SETS UP B(N) AND THEN USES AN INTERNAL SUBROUTINE
*             TO SET UP C(N-1,J) FOR J=0,M. THE INTERNAL SUBROUTINE
*             PARCON COMPUTES THE PARTIAL CONVOLUTION WHICH IS B(N-1).
*               THE NEXT C(I,J) IS SET UP BY CCOM AND THE NEXT B(I-1)
*             COMPUTED BY PARCON. THIS IS REPEATED UNTIL ALL THE PARTIAL
*             CONVOLUTIONS HAVE BEEN DONE. THE RESULTING WAVELET IS THEN
*             SCALED BY EXP**(TRAN(1)).
*               THE OUTPUT OF PARCON FOR ONE STAGE IS THE INPUT FOR THE
*             NEXT STAGE SO THAT THE ADDRESSES B1 AND B2 IN THE PARCON
*             ROUTINE ARE REVERSED BETWEEN STAGES.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709,7090 (MAIN FRAME ONLY)
* STORAGE   - 308 DECIMAL REGISTERS
* SPEED     - 2200+94L+16L**2+3L**3+270N+37L*N MACHINE CYCLES
* AUTHOR    - J.N. GALBRAITH    NOV. 1, 1961
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - MAXAB, COSTBL, COSP
*       AND FORTRAN SYSTEM ROUTINES - LOG, EXP
*
* FORTRAN USAGE
*     CALL FACTOR(SPECT,N,L,WAVE,SPACE)
*
* INPUTS
*
*    SPECT(I)  I=1,N  SPECTRUM FROM ZERO TO PI
*
*    N         NUMBER OF POINTS IN SPECTRUM
*              MUST BE GRTHN 0.
*
*    L         LENGTH OF DESIRED WAVELET.
*              MUST BE GRTHN 0, LSTHN= N.
*
*   SPACE(I)   I=1,NSPACE.  NSPACE=3*L+N+1. WORK SPACE FOR COMPUTATIONS.
*              THE QUANTITIES B2(I),C(I),TRAN(I),WORK(I), AND COST(I)
*              WHICH ARE MENTIONED IN THE ABOVE ABSTRACT ARE IN SPACE(I)
*              IN THE FOLLOWING MANNER- (SEE OUTPUTS FOR LOCATION OF B1)
*              B2(I),I=1,L IS SPACE(1) TO SPACE(L). SPACE FOR PARTIAL
*              CONVOLUTION.
*              COST(I),I=1,L+1 IS SPACE(1) TO SPACE(L+1). SPACE FOR
*                     COSINE TABLE FOR COSINE SERIES EXPANSION.
*              C(I),I=1,L IS SPACE(L+2) TO SPACE(2L+1). SPACE FOR COLUMN
*                     OF C(I,J) MATRIX.
*              WORK(I),I=1,N IS SPACE(2L+2) TO SPACE(2L+N+1). WORK SPACE
*                     FOR SPECTRUM.
*              TRAN(I),I=1,L IS SPACE(2L+N+2) TO SPACE(3L+N+1). SPACE
*                     FOR COSINE TRANSFORM.
*
*    NOTE-
*     NO CHECKS ARE MADE ON THE VALUES OF N AND L. BOTH MUST BE GREATER
*     THAN 0, AND L MUST BE LESS THAN OR =N. ILLEGAL VALUES MAY RESULT
*     IN INCORRECT WAVELETS OR PROGRAM LOOPS.
*
* OUTPUTS
*
*   WAVE(I)    I=1,L  OUTPUT MINIMUM PHASE WAVELET. SAME SPACE IS USED
*              FOR B1(I),I=1,L. IF THE INPUT SPECTRUM CAN BE DESTROYED,
*              SPECT AND WAVE CAN BE THE SAME. WE NOTE THAT N IS GRTHN
*              OR EQUAL TO L SO THAT THERE IS NO SPACE DIFFICULTY
*              INVOLVED IN THIS EQUIVALENCE.
*
* EXAMPLES
*
* 1. INPUTS  -
*              FOR A CONTINUOUS SPECTRUM
```

```
    *                 SPECT= 1.25+COS(W), W=0,PI
    *             THE WAVELET IS
    *                 WAVE= 1.,.5,0.,0.,......,0.
    *             FOR THE DISCRETE CASE THE NUMBERS WILL NOT COME OUT
    *             EXACTLY THE SAME DUE TO ROUND OFF AND APPROXIMATION.
    *             FOR A TEST CASE THE INPUT SPECTRUM CAN BE SET UP WITH A
    *             FORTRAN LOOP.   SPECT(I)=1.25 +COSF(FLOATF(I-1)*W) ,I=1,N
    *                                     W =PI/FLOATF(N-1)
    *             WHERE N IS THE LENGTH OF THE SPECTRUM.
    *             RESULTS ARE GIVEN BELOW FOR N=500
    *
    *    OUTPUTS - WAVE(1...5)= 1.000E00,0.5000E00,-0.4899E-06,-0.1327E-07
    *
    *             THE HIGHER TERMS ARE EVEN SMALLER WITH WAVE(20) LESS THAN
    *             10**(-8)
    *
    * PROGRAM FOLLOWS BELOW
    *
           PZE
           BCI     1,FACTOR
    FACTOR SXA     RETURN,1                SAVE IR1
           SXA     RETURN+1,2              SAVE IR2
           SXA     RETURN+2,4              SAVE IR4
           SXD     FACTOR-2,4
           CLA     1,4           SPECTRUM ADDRESS
           STA     MAX+2
           ADD     ONE
           STA     LOOP1
           CLA*    2,4           GET N IN DECREMENT
           STD     END1
           STO     N
    *     CALL FACTOR,SPECT,N,L,WAVE,SPACE)
           SUB     DONE          N-1
           STO     NN
           ADD     DONE
           LRS     18            N IN ADDRESS
           STO     NA
           SUB     ONE
           ORA     CONST
           FAD     CONST
           STO     NF            NF=FLOATING (N-1)
           CLA*    3,4           GET L IN DECREMENT
           STO     L
           STD     END3
           ARS     18            L IN ADDRESS
           STO     LA
           CLA     4,4           GET B1 AND WAVELET ADDRESS
           STA     WAVAD
           STA     PAR+1
           STA     BFST
           STA     LOOP2
           STA     LOOP3+1
           STA     LOOP4+2
           CLA     5,4           GET B2 AND COST ADDRESS
           STA     B2AD
           STA     PAR+2
           STA     CST+2
           STA     CSP+4
           SUB     LA
           SUB     ONE           ADDRESS OF C
           STA     PAR+3
           STA     COM+1
           SUB     LA            ADDRESS OF WORK
           STA     WGT+3
           STA     WGT+5
           STA     CSP+1
           STA     CSP+2
           ADD     ONE           ADDRESS OF WORK+1
           STA     END1-2
           STA     WGT
           STA     WGT+2
           SUB     NA
           SUB     ONE           ADDRESS OF TRAN
           STA     CSP+9
           STA     LOOP3
```

```
         STA    COM+2
         STA    SCALE
   MAX   TSX    $MAXAB,4                  FIND MAXIMUM OF SPECTUM
         PZE    N
         PZE    **                        LOCATION OF SPECTUM
         PZE    BIGSP
         PZE    INDEX
         LDQ    BIGSP                     MAX. OF SPECTUM
         FMP    DEC                       10**(-6) OF MAX
         STO    BIGSP
         AXT    1,1
   LOOP1 CLA    **,1          **=SPECT+1
         CAS    BIGSP
         TRA    *+3                       SPECT LARGER
         TRA    *+2                       SPECT EQUAL
         CLA    BIGSP                     SPECT LESS
         TSX    $LOG,4                    LOG(SPECT)
         FDP    NF            LOG(SPECT)/(N-1)
         STQ    **,1          **=WORK+1
         TXI    *+1,1,1
   END1  TXL    LOOP1,1,**    **=N
         TXI    *+1,1,-1
   WGT   CLA    **,1          **=WORK+1. WEIGHT LAST
         FDP    TWOD          TERM IN SPECTRUM BY 1/2
         STQ    **,1          **=WORK+1
         CLA    **            **=WORK. WEIGHT FIRST
         FDP    TWOD          TERM IN SPECTRUM BY 1/2
         STQ    **            **=WORK
         CLA    L
         SUB    DONE
         STO    LL
   CST   TSX    $COSTBL,4     GO TO COSINE TABLE
         PZE    NN
         PZE    **            COST
  * COSP GIVES HALF OF COSINE TRANSFORM OF LOG(SPECT) EXCEPT FOR
  *      TRAN(1) WHICH IS 2 TIMES NEEDED VALUE.
   CSP   TSX    $COSP,4       GO TO COSINE TRANSFORM
         PZE    **            WORK SPACE FOR SPECTRUM
         PZE    **            WORK SPACE FOR SPECTRUM
         PZE    NN            N-1
         PZE    **            COST
         PZE    NN            N-1
         PZE    ZERO          JMIN=0
         PZE    LL            JMAX=L-1
         PZE    ONED          1.0
         PZE    **            TRAN(COSTR)
  * TRAN CONTAINS COSINE TRANSFORM OF 1/2 LOG(SPECT).  FIRST TERM
  * MUST BE WEIGHTED BY 1/2.  (SEE SYMBOLIC ADDRESS **SCALE**.)
         CLA    L
         ARS    1                         L/2
         ANA    MASK
         ADD    DONE                      L/2+1
         STO    M                         M=L/2+1
         CLA    ONED                      1.0
   BFST  STO    **            **=B1.      B1(0)=1.0
         AXT    1,1
         CLA    M                         M
         SUB    DONE                      M-1
         STD    END2
   LOOP2 STZ    **,1                      CLEAR B1
         TXI    *+1,1,1
   END2  TXL    *-2,1,**      **=M-1
         LXD    M,1                       IR1=M
   LOOP3 CLA    **,1                      TRAN
         STO    **,1                      B1
         TXI    *+1,1,1
   END3  TXL    LOOP3,1,**    L IN DECREMENT
         AXT    1,2
         CLA    M
         STO    P
         SUB    DONE
         STD    END23
         AXT    1,1
   CONV  CLA    P
         SUB    DONE
```

```
        STO     P
        SXD     K,2
 COM    TSX     CCOM,4
        PZE     **                    C
        PZE     **            TRAN
 PAR    TSX     PARCON,4
        PZE     **                    LOCATION OF B1
        PZE     **                    LOCATION OF B2
        PZE     **                    LOCATION OF C
        CLA     PAR+1                 EXCHANGE
        LDQ     PAR+2                 LOCATIONS
        STO     PAR+2                 OF B1
        STQ     PAR+1                 AND B2
        TXI     *+1,2,1
        TXI     *+1,1,1
 END23  TXL     CONV,1,**     **=M-1
        CLA     M                     GET M
        ARS     18                    M IN ADDRESS
        LBT                           LOW BIT TEST
        TRA     *+4                   M EVEN, B2 CONTAINS WAVELET
        CLA     WAVAD         M ODD, B1=WAVELET
        STA     LOOP4
        TRA     *+3
        CLA     B2AD          B2 ADDRESS. B2 = WAVELET.
        STA     LOOP4
 SCALE  CLA     **            **=TRAN(1)
        FDP     TWOD
        XCA
        TSX     $EXP,4
        STO     NORM          SCALE FOR WAVELET
        CLA     LL
        STD     END4
        AXT     0,1
 LOOP4  LDQ     **,1                  B2 OR B1
        FMP     NORM          SCALE FOR WAVELET
        STO     **,1                  WAVELET
        TXI     *+1,1,1
 END4   TXL     LOOP4,1,**    **=L-1
RETURN  AXT     **,1                  RESTORE IR1
        AXT     **,2                  RESTORE IR2
        AXT     **,4                  RESTORE IR4
        TRA     6,4
 L      PZE     0
 LL     PZE     0             L-1
 K      PZE     0
 N      PZE     0
 NN     PZE     0             N-1
 M      PZE     0
 P      PZE     0
 NF     PZE     0
 NA                           N IN ADDRESS
 LA                           L IN ADDRESS
 WAVAD                        WAVELET AND B1 ADDRESS
 B2AD                         B2 ADDRESS
 NORM   PZE     0
 BIGSP  PZE     0
 INDEX  PZE     0
 CONST  OCT     +233000000000
 MASK   OCT     777777000000
 ZERO   PZE     0
 ONE    PZE     1,0,0
 DONE   PZE     0,0,1
 ONED   DEC     1.0
 TWOD   DEC     2.0
 DEC    DEC     .000001
*CCOM -COMPUTES C(P,J) FOR J=0 TO L-1
*CALLING SEQUENCE
*       TSX     CCOM,4
*       PZE     LOCATION OF C(P,0)
*       PZE     LOCATION OF TRAN
*       RETURN
 CCOM   SXA     BACK,1        SAVE IR1
        SXA     BACK+1,2      SAVE IR2
        SXA     BACK+2,4      SAVE IR4
        CLA     L             GET L
```

```
       STD    ADDR2+2
       CLA    P               GET P
       ARS    18              L IN ADDRESS
       CHS
       ADD    1,4             ADDRESS OF C(P,P)
       STA    ADDR3
       STA    ADDR4
       CLA    1,4             LOCATION OF C(0)
       STA    ADDR1
       ADD    ONE
       STA    ADDR2
       CLS    P
       ARS    18
       ADD    2,4             TRAN
       STA    STO1
       CLA    ONED            1.0
 ADDR1 STO    **              C(0)
       AXT    2,1             CLEAR
 ADDR2 STZ    **,1            C(1) TO
       TXI    *+1,1,1         C(L)
       TXL    ADDR2,1,**      **=L
 STO1  CLA    **              TRAN(P)
 ADDR3 STO    **              C(P,P)
       STO    TEMP1
       STO    TEMP2
       CLA    LL
       LRS    35              INTO MQ
       DVP    P               (L-1)/P
       LLS    53              INTO AC
       SUB    DONE            (L-1)/P-1
       TZE    BACK            IF ZERO,NO MORE TO DO
       STD    END             NOT ZERO, SET TO DO (L-1)/P-1 TIMES
       CLA    P
       PDX    ,2              P IN IR2
       SXD    END-2,2
       AXT    1,1
       CLA    TWOD            GET 2.0
       STO    R               INITIALIZE R
 LOOP  LDQ    TEMP1
       FMP    TEMP2           TRAN(1)
       FDP    R
 ADDR4 STQ    **,2            **=C. C(R+1) COMPUTED.
       STQ    TEMP1           SAVE FOR NEXT C
       CLA    R               GET R
       FAD    ONED            INCREMENT BY 1.0
       STO    R               RE-SET R
       TXI    *+1,2,**        **=P. INCREMENT C STORAGE INDEX
       TXI    *+1,1,1         INCREMENT LOOP COUNTER
 END   TXL    LOOP,1,**       **=L-1/P-1. END LOOP CHECK.
 BACK  AXT    **,1            RESTORE IR1
       AXT    **,2            RESTORE IR2
       AXT    **,4            RESTORE IR4
       TRA    3,4             RETURN
 TEMP1 PZE    0,0,0           WILL CONTAIN PARTIAL SUM FOR C(P)
 TEMP2 PZE    0,0,0           WILL CONTAIN TRAN(P)
 R     PZE
*PARCON COMPUTES A PARTIAL CONVOLUTION OF C AND B1
*CALLING SEQUENCE
*      TSX    PARCON,4
*      PZE    LOCATION OF B1
*      PZE    LOCATION OF B2
*      PZE    LOCATION OF C(X,0)
PARCON SXA    EXT,1           SAVE IR1
       SXA    EXT+1,2         SAVE IR2
       SXA    EXT+2,4         SAVE IR4
       CLA    2,4             GET LOCATION OF B2
       STA    REG1
       STA    REG3
       STA    REG3+1
       ADD    ONE
       STA    REG2
       CLA    3,4             LOCATION OF C
       STA    REG5
       CLA    ONED            1.0
 REG1  STO    **              B2(0)=1.0
```

****
*   FACTOR   *
****
(PAGE 7)

```
        AXT    2,1
        CLA    L             GET L
        STD    REG2+2
        SUB    DONE
        STD    REG8
REG2    STZ    **,1          CLEAR B2(1) TO B2(L)
        TXI    *+1,1,1
        TXL    REG2,1,**     DECREMENT=L
        CLA    M
        SUB    K             K GOES FROM 1 TO M-1. SET BY CALLING LOOP.
        PDX    ,1            IR1=M-K
        SXD    REG3+2,1
        PDC    ,2
        SXD    REG3+3,2
        SXD    S,1           S=IR1=M-K
REG7    AXT    0,2           ZERO IR2
        LXA    EXT+2,4       RESET IR4
        CLA    S             GET S
        STD    REG6
        CLS    S
        ARS    18
        ADD    1,4           LOCATION OF B1(S)
        STA    REG4
        AXT    0,4
REG5    LDQ    **,4          C(0)
REG4    FMP    **,2          B1(S)
REG3    FAD    **,1          B2
        STO    **,1          B2
        TXI    *+1,4,**      (M-K) IN DECREMENT
        TXI    *+1,2,**      -(M-K) IN DECREMENT
REG6    TXL    REG5,4,**     **=S
        TXI    *+1,1,1
REG8    TXL    REG7-1,1,**   **=L-1
EXT     AXT    **,1          RESTORE IR1
        AXT    **,2          RESTORE IR2
        AXT    **,4          RESTORE IR4
        TRA    4,4           RETURN
S       PZE    0
        END
```

```
*      FAPSUM (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0065
*      FAP
*FAPSUM
       COUNT   50
       LBL     FAPSUM
       ENTRY   FAPSUM (LD,DATA,SUMCK)
*
*                      ----ABSTRACT----
*
* TITLE - FAPSUM
*        COMPUTES A LOGICAL SUMCHECK
*
*              FAPSUM COMPUTES A SUMCHECK BY SUMMING THE CONTENTS OF A
*              VECTOR WITH THE -ADD AND CARRY LOGICAL WORD- INSTRUCTION.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 14 REGISTERS
* SPEED     - LENGTH OF VECTOR TIMES 4 MACHINE CYCLES
* AUTHOR    - J.F. CLAERBOUT,    JUNE, 1962
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL FAPSUM(LD,DATA,SUMCK)
*
* INPUTS
*
*    DATA(I)    I=1...LD IS A DATA VECTOR.
*               (NEED NOT HAVE FLOATING NAME).
*
*    LD         IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*    SUMCK      IS LOGICAL SUMCHECK FOR THE DATA.
*               (NEED NOT HAVE FLOATING POINT NAME)
*
* EXAMPLES
*
* 1. INPUTS  - DATA(1...3) = 1.,-2.,-3.  LD=3
*    OUTPUTS - SUMCK = OCT 60660000001
*
* 2. INPUTS  - DATA(1...4) = 1,-2,-3,4    LD=4
*    OUTPUTS - SUMCK = OCT 000012000001
*
* 3. INPUTS  - DATA(1...2) = 6HAB    , 6H  45    LD=2
*    OUTPUTS - SUMCK = OCT 020264664141
*
       BCI     1,FAPSUM
FAPSUM SXA     SV4,4
       CLA     2,4
       ADD     =1
       STA     A
       CLA*    1,4
       PDX     ,4
       CLM
 A     ACL     **,4
       TIX     A,4,1
 SV4   AXT     **,4
       SLW*    3,4
       TRA     4,4
       END
```

```
*      FASCN1 (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0198
*      FAP
*FASCN1
       COUNT    200
       LBL      FASCN1
       ENTRY    FASCN1 (VECT,ILO,IHI,VALUE,IFIND,IANS)
*
*                   ----ABSTRACT----
*
* TITLE - FASCN1
*       FAST SCAN VECTOR FOR ELEMENT EQUAL OR GREATER THAN GIVEN VALUE
*
*             FASCN1 SCANS A VECTOR RANGE AT HIGH SPEED TO FIND THE
*             FIRST ELEMENT (IF ANY) EQUAL TO OR GREATER THAN A GIVEN
*             VALUE.  VECTOR MAY BE FIXED POINT OR FLOATING POINT.
*             PROGRAM IS MOST EFFICIENT FOR LONG VECTORS.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 107 REGISTERS
* SPEED     - 100 + 5.2 N  MACHINE CYCLES WHERE N = NO. ELEMENTS SCANNED
* AUTHOR    - S.M. SIMPSON JR, JUNE 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL FASCN1(VECT,ILO,IHI,VALUE,IFIND,IANS)
*
* INPUTS
*
*    VECT(I)   I=ILO,...,IHI  IS THE FORTRAN-TYPE VECTOR TO BE SCANNED
*              VECT(I) MAY BE FIXED POINT OR FLOATING POINT
*
*    ILO       MUST BE GRTHN= 1
*
*    IHI       MUST BE GRTHN= ILO
*
*    VALUE     IS TO BE COMPARED (BY A CAS INSTRUCTION) AGAINST
*              VECT(ILO,...,IHI).  VALUE SHOULD BE FIXED POINT OR
*              FLOATING POINT MODE ACCORDING TO MODE OF VECT(I).
*
* OUTPUTS
*
*    IFIND     IS NOT DISTURBED IF VECT(ILO,...,IHI) ALL LESS THAN VALUE
*              IS FIRST INDEX GRTHN=ILO SUCH THAT VECT(IFIND)
*                GRTHN= VALUE IF ONE IS FOUND.
*
*    IANS      =  0 MEANS VECT(ILO,...,IHI) ALL LESS THAN VALUE
*              =  1 MEANS VECT(IFIND) WAS FOUND TO BE GRTHN= VALUE
*              = -2 MEANS ILLEGAL ILO
*              = -3 MEANS ILLEGAL IHI
*
* EXAMPLES
*
* 1. SHOWING USE ON BOTH FIXED AND FLOATING DATA
*    INPUTS  - X(1...7) = 9.,8.,7.,6.,7.,8.,9.        VAL =8.
*              IX(1...7) = 9,8,7,6,7,8,9             IVAL =8
*    USAGE   -        CALL FASCN1(X,3,7,VAL,IFIND1,IANS1)
*                     CALL FASCN1(IX,3,7,IVAL,IFIND2,IANS2)
*    OUTPUTS - IANS1 = IANS2 = 1       IFIND1 = IFIND2 = 6
*
* 2. SHOWING CASE WHEN VALUE NEVER FOUND
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  VAL = 10.
*    USAGE   -        CALL FASCN1(X,3,7,VAL,IFIND,IANS)
*    OUTPUTS - IANS = 0
*
* 3. ILLEGAL REQUESTS
*    USAGE   -        CALL FASCN1(X,0,3,VAL,IFIND,IANS1)
*                     CALL FASCN1(X,5,4,VAL,IFIND,IANS2)
*    OUTPUTS - IANS1 = -2 (ILLEGAL ILO)      IANS2 = -3 (ILLEGAL IHI)
*
       HTR      0
```

```
       HTR      0
       HTR      0
       BCI      1,FASCN1
FASCN1 SXD      FASCN1-4,1
       SXD      FASCN1-3,2
       SXD      FASCN1-2,4
*SET ADDRESSES
       CLA      1,4             A(A(VECT))
       ADD      K1              +1
       STA      C1
       STA      C2
       STA      C3
       STA      C4
       STA      C5
       STA      C6
       STA      C7
       STA      C8
       STA      C9
       STA      C10
       CLA      2,4             A(A(ILO))
       STA      GET2
       CLA      3,4             A(A(IHI))
       STA      GET3
       CLA      5,4
       STA      PUT5
       CLA      6,4
       STA      PUT6
*CHECK ILO, IHI AND MAKE SETTINGS
       CLS      K2
       STO      IANS
 GET2  CLA      **              A(ILO)
       STO      ILO
       TMI      LEAVE
       TZE      LEAVE
*(SET TO COUNT ON XR1 FROM ILO TO IHI)
       PDX      0,1
       CLS      K3
       STO      IANS
 GET3  CLA      **              A(IHI)
       CAS      ILO
       NOP
       TRA      IHIOK
       TRA      LEAVE
 IHIOK STD      GOBAK
       STD      MAYBE
*MAKE TRIAL SETTING OF IANS=0
       STZ      IANS
*PUT VALUE IN AC
       CLA      4,4             A(A(VALUE))
       STA      GET4
 GET4  CLA      **              A(VALUE)
*COMPARE IN BLOCKS OF LENGTH 10
 C1    CAS      **,1            A(VECT)+1
       TXI      C2,1,1
       NOP
       TRA      MAYBE
 C2    CAS      **,1            DITTO
       TXI      C3,1,1
       NOP
       TRA      MAYBE
 C3    CAS      **,1            DITTO
       TXI      C4,1,1
       NOP
       TRA      MAYBE
 C4    CAS      **,1            DITTO
       TXI      C5,1,1
       NOP
       TRA      MAYBE
 C5    CAS      **,1            DITTO
       TXI      C6,1,1
       NOP
       TRA      MAYBE
 C6    CAS      **,1            DITTO
       TXI      C7,1,1
       NOP
```

```
       TRA    MAYBE
 C7    CAS    **,1         DITTO
       TXI    C8,1,1
       NOP
       TRA    MAYBE
 C8    CAS    **,1         DITTO
       TXI    C9,1,1
       NOP
       TRA    MAYBE
 C9    CAS    **,1         DITTO
       TXI    C10,1,1
       NOP
       TRA    MAYBE
 C10   CAS    **,1         DITTO
       TXI    GOBAK,1,1
       NOP
       TRA    MAYBE
*GO BACK AND COMPARE NEXT 10 ELEMENTS
*IF WE HAVENT RUN OFF END
 GOBAK TXL    C1,1,**      **=IHI
*NONE FOUND IF WE INDEXED OFF END
       TRA    LEAVE
*IN CASE OF JUMP FROM LOOP TO MAYBE, ELEMENT IN IV
*HAS BEEN FOUND=OR GREATER THAN VALUE, PROVIDED
*THAT WE HAVE NOT INDEXED BEYOND VECT(IHI)
 MAYBE TXL    FIND,1,**    **=IHI
       TRA    LEAVE
*ELEMENT DEFINITELY FOUND=OR GREATER THAN LEVEL
*SET IFIND,IANS,AND EXIT
 FIND  PXD    0,1
 PUT5  STO    **           A(IFIND)
       CLA    K1
       STO    IANS
*LEAVE,STORING IANS
 LEAVE CLA    IANS
       ALS    18
  PUT6 STO    **           A(IANS)
       LXD    FASCN1-4,1
       LXD    FASCN1-3,2
       LXD    FASCN1-2,4
       TRA    7,4
*CONSTANTS
 K1    PZE    1
 K2    PZE    2
 K3    PZE    3
*VARIABLES
 IANS  PZE    **           -2,-3,0,1
 ILO   PZE    0,0,**
       END
```

```
************************
*    FASCOR            *
************************
 REFER TO
     PROCOR


************************
*    FASCR1            *
************************
 REFER TO
     PROCOR
```


```
************************
*    FASCOR            *
************************
 REFER TO
     PROCOR


************************
*    FASCR1            *
************************
 REFER TO
     PROCOR
```

```
*      FASCUB (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0259
*      FAP
*FASCUB
       COUNT   250
       LBL     FASCUB
       ENTRY   FASCUB (COEFS,XLO,DELX,NF,FOFX)
*
*
*                    ----ABSTRACT----
*
* TITLE - FASCUB
*        FAST EVALUATE CUBIC FOR EVENLY SPACED ARGUMENTS
*
*              FASCUB PRODUCES N EVENLY SPACED VALUES OF THE THIRD
*              ORDER POLYNOMIAL
*                                         2      3
*                    F(X) = A  + A X + A X  + A X  ,
*                            0    1     2      3
*
*              X = XLO, XLO+DELX, ..., XLO+(N-1)DELX,  BY HIGH-SPEED
*              ITERATIVE TECHNIQUES, WHERE XLO AND DELX ARE PARAMETERS.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 141 REGISTERS
* SPEED      - K1 + K2 + 27.2*N  MACHINE CYCLES
*              WHERE
*                    K1 = 120 M.C. IF DELX=1.0     250 M.C. OTHERWISE
*                    K2 =  10 M.C. IF XLO =0.0     140 M.C. OTHERWISE
* AUTHOR     - S.M.SIMPSON, MARCH 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
*
* FORTRAN USAGE
*     CALL FASCUB(COEFS,XLO,DELX,NF,FOFX)
*
*
* INPUTS
*
*    COEFS(I)  I=1,2,3,4  CONTAIN A0,A1,A2,A3, RESPECTIVELY, THE
*              COEFFICIENTS OF F(X) IN THE ABSTRACT.
*
*    XLO       IS FIRST VALUE OF ARGUMENT OF F(X) IN THE ABSTRACT.
*
*    DELX      IS ARGUMENT INCREMENT.  SHOULD BE NON-ZERO.
*
*    NF        IS NUMBER OF SUCCESSIVE EVALUATIONS OF F(X) DESIRED.
*              MUST EXCEED ZERO.
*
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF NF ILLEGAL.
*
*    FOFX(I)   I=1,...,NF  CONTAIN THE N VALUES OF F(X) AS DESCRIBED IN
*                    THE ABSTRACT.
*
*
* EXAMPLES
*
* 1. INPUTS  - COEFS(1...4)=1.0,2.0,-1.0,3.0
*    USAGE   -       CALL FASCUB(COEFS,0.0,2.0,4,FOFX1)
*    OUTPUTS - FOFX1(1...4)=1.0,25.0,185.0,625.0
*
* 2. INPUTS  - COEFS(1...4)=0.0,3.0,1.0,-4.0
*    USAGE   -       CALL FASCUB(COEFS,-2.0, 1.0,3,FOFX2)
*    OUTPUTS - FOFX2(1...3) = 30.0,2.0,0.0
*
* 3. INPUTS  - SAME AS EXAMPLE 2., EXCEPT FOFX5(1...4) = -99.
*    USAGE   -       CALL FASCUB(COEFS,1.0,-1.0,2,FOFX3)
*                    CALL FASCUB(COEFS,-2.0,-1.0,1,FOFX4)
*                    CALL FASCUB(COEFS,-2.0,-1.0,0,FOFX5)
```

```
*    OUTPUTS - FOFX3(1...2) = 0.0,0.0
*              FOFX4(1)     = 30.0
*              FOFX5(1...4) = -99.
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR      0              XR1
       HTR      0              XR4
       BCI      1,FASCUB
*
* ONLY ENTRY.  FASCUB(COEFS,XLO,DELX,NF,FOFX)
*
FASCUB SXD      FASCUB-2,4
       SXD      FASCUB-3,1     (XR1 COUNTS OUTPUTS)
       CLA      1,4            A(COEFS)
       STA      CLACO
       AXT      3,1
 CLACO CLA      **,1           **=A(COEFS)          LOOP TO GET
       STO      A1+1,1                              A1,A2,A3
       TIX      CLACO,1,1      (NOTE XR1 IS LEFT = 1)
       CLA*     4,4            NF
       TMI      LEAVE
       TZE      LEAVE
       STD      CKF2
       STD      CKF3
       STD      TXL
       CLA      5,4            A(FOFX)
       ADD      K1
       STA      STOF
       ADD      K1
       STA      FADF
       CLA*     2,4            XLO
       STO      XLO
       CLA*     3,4            DELX
       STO      DELX
       CAS      K1L            IS IT = 1.0
       TRA      NOT1
       TRA      DEL1
       TRA      NOT1
*
* HI-SPEED SETTING OF CONSTANTS IF DELX=1.0
*
 DEL1  CLA      A1             IT IS ONE, HI-SPEED
       FAD      A2
       FAD      A3
       STO      BZ             BZ=A1+A2+A3
       LDQ      A3
       FMP      K3L
       STO      B2             B2=3*A3
       FAD      A2
       FAD      A2
       STO      B1             B1=2*A2+3*A3
       FAD      B2
       STO      CZ             CZ=B1+B2
       CLA      B2
       FAD      B2
       STO      C1             C1=2*B2
       STO      DZ             DZ=C1
       TRA      CKXLO
*
* SLOWER CONSTANT SETTINGS IN GENERAL
*
 NOT1  LDQ      A3
       FMP      DELX
       FAD      A2
       XCA
       FMP      DELX
       FAD      A1
       XCA
       FMP      DELX
       STO      BZ             BZ=J*(A1+J(A2+J*A3))
       CLA      DELX
```

```
         FAD    DELX
         STO    TWODEL
         FAD    DELX
         XCA
         FMP    A3
         STO    B2           B2=3*A3*J
         FAD    A2
         FAD    A2
         XCA
         FMP    DELX
         STO    B1           B1=2*A2*J+3*A3*J*J
         LDQ    B2
         FMP    DELX
         FAD    B1
         XCA
         FMP    DELX
         STO    CZ           CZ
         LDQ    B2
         FMP    TWODEL
         STO    C1           C1
         LDQ    C1
         FMP    DELX
         STO    DZ           DZ
*
* CHECK XLO AND SET F1 IN THE CASE XLO=0.
* (IN THIS CASE H1 AND G1 ARE ALREADY OK BY SYNONYMS)
*
 CKXLO ZET      XLO
       TRA      GENXL
       CLA*     1,4          COEFS(1)=AZ
       STO*     STOF
       TRA      TXI1
*
* SET F1,G1,H1 IN GENERAL
*
 GENXL LDQ      A3
       FMP      XLO
       FAD      A2
       XCA
       FMP      XLO
       FAD      A1
       XCA
       FMP      XLO
       FAD*     1,4          COEFS(1)=AZ
       STO*     STOF         F1=AZ+XLO(A1+XLO(A2+XLO*A3))
       LDQ      B2
       FMP      XLO
       FAD      B1
       XCA
       FMP      XLO
       FAD      BZ
       STO      G            G1 STORED
       LDQ      C1
       FMP      XLO
       FAD      CZ
       STO      H            H1 STORED
*
* SET FOFX(2) IF NF GRTHN 1
*
 TXI1  TXI      *+1,1,1
 CKF2  TXH      LEAVE,1,**   **=NF
       CLA      G
       FAD*     FADF         (XR1 = 2 NOW)
       STO*     STOF
       TXI      *+1,1,1
 CKF3  TXH      LEAVE,1,**   **=NF
*
* ENTER LOOP SO AS TO COMPUTE G2 FROM H1,
* THEN F3 FROM G2, THEN CYCLE (H2,G3,F4), (H3,G4,F5), ...
*
       CLA      H
       TRA      FADG
*
* LOOP TO PRODUCE FOFX(3,4,...,NF)
*
```

FASCUB
(PAGE 4)

```
 CLADZ CLA      DZ
       FAD      H
       STO      H
 FADG  FAD      G
       STO      G
 FADF  FAD      **,1            **=A(FOFX)+2
 STOF  STO      **,1            **=A(FOFX)+1
       TXI      *+1,1,1
 TXL   TXL      CLADZ,1,**      **=NF
*
* EXIT
*
 LEAVE LXD      FASCUB-3,1
       TRA      6,4
*
* CONSTANTS, TEMPORARIES
*
 K1    PZE      1
 K1L   DEC      1.0
 K3L   DEC      3.0
 XLO   PZE      **,**,**
TWODEL PZE      **,**,**        2*DELX
 DELX  PZE      **,**,**        CALLED J IN EQUATIONS BELOW
 DZ    PZE      **,**,**        J*C1
 C1    PZE      **,**,**        2*J*B2
 CZ    PZE      **,**,**        J*(B1+J*B2)
 H     SYN      CZ
 B2    PZE      **,**,**        3*J*A3
 B1    PZE      **,**,**        J*(2*A2+3*J*A3)
 BZ    PZE      **,**,**        J*(A1+J*(A2+J*A3))
 G     SYN      BZ
 A3    PZE      **,**,**
 A2    PZE      **,**,**
 A1    PZE      **,**,**
       END
```

```
***********************
*    FASEPC           *
***********************
 REFER TO
     PROCOR
```

```
***********************
*    FASEP1           *
***********************
 REFER TO
     PROCOR
```


```
***********************
*    FASEPC           *
***********************
 REFER TO
     PROCOR
```

```
***********************
*    FASEP1           *
***********************
 REFER TO
     PROCOR
```

```
*     FASTRK (SUBROUTINE)              9/8/64   LAST CARD IN DECK IS NO. 0118
*     FAP
*FASTRK
      COUNT   100
      LBL     FASTRK
      ENTRY   FASTRK (IXVEC,IXSTRT,IXLOOK,MXTRAK,IANS)
*
*
*                   ----ABSTRACT----
*
* TITLE - FASTRK
*        FAST TRACK THROUGH A VECTOR OF INDICES
*
*              FASTRK TRACKS THROUGH A FORTRAN FIXED POINT VECTOR EACH
*              OF WHOSE ELEMENTS REPRESENTS THE INDEX OF ANOTHER ELEMENT
*              OR ELSE HAS VALUE ZERO, SEARCHING FOR A GIVEN INDEX
*              AND STARTING FROM A GIVEN ELEMENT, SUBJECT TO A LIMIT
*              ON THE NO. OF SEARCHES, AND WHERE SEARCHING STOPS WHEN
*              A ZERO ELEMENT IS FOUND.
*
*              THUS FASTRK LOOKS DOWNSTREAM IN A BRANCHING PROCESS LIKE
*              A RIVER COMPLEX TO SEE IF RIVER A FLOWS INTO RIVER B.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 26   REGISTERS
* SPEED      - 38 + 12*(N-1)  MACHINE CYCLES
*                 WHERE N = NO. OF SEARCHES MADE
* AUTHOR     - S.M. SIMPSON,  JULY 1963
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL FASTRK(IXVEC,IXSTRT,IXLOOK,MXTRAK,IANS)
*
*
* INPUTS
*
*    IXVEC(I)  I=1,2,...  IS INDEX VECTOR OF ARBITRARY LENGTH.  EACH
*                 ELEMENT MUST BE GREATER THAN OR = ZERO
*
*    IXSTRT    SPECIFIES THAT IXVEC(IXSTRT) IS THE FIRST ELEMENT
*                 TO BE COMPARED AGAINST IXLOOK
*              MUST EXCEED ZERO
*
*    IXLOOK    IS THE INDEX SEARCHED FOR
*              MUST EXCEED ZERO
*
*    MXTRAK    IS THE MAXIMUM NO. OF TRIES TO BE MADE
*              MUST EXCEED ZERO
*
*
* OUTPUTS      (NO  LEGALITY CHECKS ARE MADE ON THE INPUTS)
*
*    IANS      =  0  MEANS TRAIL STOPPED AT A ZERO ELEMENT
*              = -1  MEANS TRAIL STOPPED BECAUSE MAX NO. OF TRIES
*                       WAS ABOUT TO BE EXCEEDED
*              = +K, WHERE K POSITIVE, MEANS SEARCH WAS SUCCESSFUL,
*                       AND THAT IXVEC(K) = IXLOOK.
*
*
* EXAMPLES
*
* 1. INPUTS  - IXVEC(1...12) =  7, 9, 0, 5, 4, 3, 6, 8, 3, 1,12, 4
*    USAGE   -       CALL FASTRK(IXVEC,10, 3,10,IANS1)
*                    CALL FASTRK(IXVEC,10, 3, 4,IANS2)
*                    CALL FASTRK(IXVEC,10, 3, 3,IANS3)
*                    CALL FASTRK(IXVEC, 8, 8,10,IANS4)
*                    CALL FASTRK(IXVEC, 4, 4,10,IANS5)
*                    CALL FASTRK(IXVEC,12, 3,10,IANS6)
*                    CALL FASTRK(IXVEC, 7,12,10,IANS7)
```

```
*                     CALL FASTRK(IXVEC, 3,12,10,IANS8)
*                     CALL FASTRK(IXVEC, 2, 9,10,IANS9)
*    OUTPUTS - IANS1 = 6  (IE IXVEC(6) = 3)
*              IANS2 = 6
*              IANS3 =-1  (TOO MANY TRACKS. TAKES 4 TO GO FROM 10 TO 6)
*              IANS4 = 8
*              IANS5 = 5
*              IANS6 =-1  (NOTE LOOP AT 4-5-4-5 ETC)
*              IANS7 = 0  (STOPS AT IXVEC(3))
*              IANS8 = 0       DITTO
*              IANS9 = 2
*
*
* PROGRAM FOLLOWS BELOW
*
        HTR     0               XR4
        BCI     1,FASTRK
FASTRK  SXD     FASTRK-2,4
        SXA     LEAVE,2
        SXA     LEAVE+1,1
        CLA     1,4
        ADD     *-1             A(IXVEC)+1
        STA     CLA
        CLA*    4,4             SET FOR MAX NO.
        PDX     0,2             OF LOOKS
        CLA*    2,4             SET TO LOOK AT IXVEC(IXSTRT) FIRST
* BEGIN TRACKING LOOP
 NEXT   PDX     0,1
 CLA    CLA     **,1            **=A(IXVEC)+1
        TZE     SETANS
        CAS*    3,4
        TRA     *+2
        TRA     ARRIVE
        TIX     NEXT,2,1
* SET IANS AND LEAVE
ENOUGH  CLS     KD1
        TRA     SETANS
ARRIVE  PXD     0,1
SETANS  STO*    5,4
 LEAVE  AXT     **,2
        AXT     **,1
        TRA     6,4
 KD1    PZE     0,0,1
        END
```

```
*       FDOT (SUBROUTINE)                    9/4/64   LAST CARD IN DECK IS NO. 0100
*       FAP
*FDOT
        COUNT   80
        LBL     FDOT
        ENTRY   FDOT (LXY,X,Y,ANS)
        ENTRY   FDOTR (LXY,X,Y,ANS)
*
*                    ----ABSTRACT----
*
* TITLE - FDOT  , WITH SECONDARY ENTRY POINT FDOTR
*        FAST DOT PRODUCT OF TWO VECTORS
*
*              FDOT COMPUTES THE DOT PRODUCT OF TWO VECTORS.
*
*              FDOTR COMPUTES THE DOT PRODUCT OF A VECTOR WITH THE
*              REVERSE OF ANOTHER VECTOR.
*
*              THUS FDOT CORRESPONDS TO ONE LAG OF A CROSSCORRELATION,
*              FDOTR TO ONE LAG OF A CONVOLUTION.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 40 REGISTERS
* SPEED     - LENGTH OF VECTOR TIMES 25.4 MACHINE CYCLES - 7090
*                                    28.6 MACHINE CYCLES - 709
* AUTHOR    - R.A. WIGGINS,   4/10/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL FDOT  (LXY,X,Y,ANS)
*     CALL FDOTR (LXY,X,Y,ANS)
*
* INPUTS
*
*    X(I)       I=1...LXY IS FLOATING POINT VECTOR
*
*    Y(I)       I=1...LXY IS FLOATING POINT VECTOR
*
*    LXY        IS FORTRAN II INTEGER
*               MUST BE GRTHN=1
*
* OUTPUTS
*
*    ANS        IS FLOATING POINT DOT PRODUCT OF X AND Y.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3)=1.,2.,3.  Y(1...3)=1.,2.,3.  LXY=3
*    OUTPUTS - FDOT  ANS=14.    FDOTR  ANS=10.
*
* 2. INPUTS  - X(1)=1.  Y(1)=2.  LXY=1
*    OUTPUTS - FDOT  ANS=2.     FDOTR  ANS=2.
*
*
* PROGRAM FOLLOWS BELOW
*
        PZE
        BCI     1,FDOT
 FDOT   SXD     *-2,4          SAVE
        SXA     RET,1          INDECES.
        CLA*    1,4            A(LXY)
        PDX     ,1             SET IR 1 FOR FDOT.
        CLA     TIX            SET FDOT
        STP     SW             SWITCH.
        TRA     A
 FDOTR  SXD     FDOT-2,4       SAVE
        SXA     RET,1          INDECES.
        AXT     1,1            SET IR 1 FOR FDOTR.
        CLA     TXI            SET FDOTR
        STP     SW             SWITCH.
```

****
*   FDOT   *
****
(PAGE 2)

```
A       CAL     2,4             A(A(X))
        ADD     =1B35
        STA     X
        CAL     3,4             A(A(Y))
        ADD     =1B35
        STA     Y
        STZ     ANS
        CLA*    1,4             A(LXY)
        TZE     RET-2
        TMI     RET-2
        PDX     ,4              SET IR 4.
X       LDQ     **,1            A(X)
Y       FMP     **,4            A(Y)
        FAD     ANS
        STO     ANS
SW      PZE     *+1,1,1         EITHER TXI *+1,1,1 OR TIX *+1,1,1
        TIX     X,4,1
        LXD     FDOT-2,4        RESET IR4.
        CLA     ANS
        STO*    4,4             A(ANS)
RET     AXT     **,1            RESET IR 1.
        TRA     5,4             RETURN.
TIX     TIX     0,,0
TXI     TXI     0,,0
ANS     PZE
        END
```

```
************************
*     FDOTR            *
************************
 REFER TO
     FDOT
```



```
************************
*     FDOTR            *
************************
 REFER TO
     FDOT
```

```
*     FIRE2  (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0151
*     LABEL
CFIRE2
      SUBROUTINE FIRE2 (NRA,NCAT,NCAN,AA,NRR,NCR,RR,NRG,GG,FF,C)
C
C                    ----ABSTRACT----
C
C TITLE - FIRE2
C       TWO-DIMENSIONAL FILTER BY RECURSION
C
C             FIRE2 INCREASES THE LENGTH OF ONE DIMENSION OF A 2-
C             DIMENSIONAL LEAST-SQUARE FILTER BY ONE.  THUS, GIVEN
C             THE FILTER F(I,J) I=1,...,NRA  J=1,...,NCAN-1 THAT IS THE
C             SOLUTION TO THE EQUATION
C
C                NRA  NCAN-1
C                SUM ( SUM  ( F(I,J)*R(I-K,J+L-2) ) ) = G(K,L)
C                I=1    J=1
C
C             FOR    K = 1,...,NRA
C                    L = 1,...,NCAN-1
C
C             THEN FIRE2 INCREASES THE J DIMENSION BY ONE SO THAT
C             THE EQUATIONS ARE SATISFIED FOR L = 1,...,NCAN.
C
C             TO PERFORM THE RECURSION, FIRE2 MAKES USE OF THE
C             PREDICTION ERROR OPERATORS AA AND THE ERROR MATRIX STORED
C             IN COMPUTATION SPACE C AS GIVEN BY RLSPR2.
C
C LANGUAGE    - FORTRAN II SUBROUTINE
C EQUIPMENT   - 709, 7090, 7094 (MAIN FRAME ONLY)
C STORAGE     - 271 REGISTERS
C SPEED       - ABOUT .00075*M*N**2 SECONDS ON THE 7094 MOD 1
C               FOR N GRTHN 7 AND M GRTHN 25 .
C AUTHOR      - R.A. WIGGINS  8/63   GEOSCIENCE, INC.
C
C                    ----USAGE---
C
C TRANSFER VECTOR CONTAINS ROUTINES - DOTJ,DOTP,IXCARG,MATML3,STZ
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL FIRE2 (NRA,NCAT,NCAN,AA,NRR,NCR,RR,NRG,GG,FF,C)
C
C INPUTS
C
C    NRA      NUMBER ROWS IN AA AND F.
C             MUST BE GRTHN= 1
C
C    NCAT     NUMBER OF COLUMNS OF AA AND F TOTAL.  I.E. THIS IS THE
C               UPPER LIMIT ON THE NUMBER OF COLUMNS TO WHICH F CAN
C               BE EXTENDED.
C             MUST BE GRTHN= 1
C
C    NCAN     NUMBER OF COLUMNS OF AA AND F NOW.  I.E. THIS IS THE
C               PRESENT LENGTH OF THE PREDICTORS, THE FUTURE LENGTH OF
C               THE FILTER.
C             MUST BE GRTHN= 0 LSTHN= NCAT
C
C    AA(L)    L=1,...,NRA*NCAT*NRA  CONTAINS THE PREDICTION ERROR
C               OPERATORS A(I,J,K) OF LENGTH NCAN AS GIVEN BY RLSPR2.
C
C    NRR      NUMBER ROWS OF R.
C             MUST BE GRTHN= 1 AND ODD.
C
C    NCR      NUMBER COLUMNS OF R.
C             MUST BE GRTHN= 1
C
C    RR(I)    I=1,...,NRR*NCR CONTAINS R(J,K)  J=-NRR/2,...,-1,0,1,...
C               NRR/2  K=0,...,NCR-1, AN AUTOCORRELATION ARRAY.
C
C    NRG      NUMBER ROWS OF G.
C             MUST BE GRTHN= 1
C
C    GG(I)    I=1,...,NRG CONTAINS G(K,L)  K=-NRG/2,...,NRG/2  L=NCAN.
```

```
C               IS ASSUMED TO BE ZERO OUTSIDE THE LIMITS OF DEFINITION.
C
C    FF(L)      L=1,...,NRA*(NCAN-1) CONTAINS F(I,J) I=1,...,NRA  J=1,...
C                 ,NCAN-1 AS DESCRIBED IN THE ABSTRACT.
C
C    C(I)       I=1,...,4*NRA*NRA CONTAINS DATA FROM RLSPR2 THAT IS
C                 NEEDED BY FIRE2.
C
C OUTPUTS
C
C    FF(L)      L=1,...,NRA*NCAN CONTAINS F(I,J) I=1....,NRA  J=1,...,
C                 NCAN AS DESCRIBED IN THE ABSTRACT.
C
C
C EXAMPLES
C
C 1. EXTENSION OF A ONE-DIMENSIONAL PREDICTOR
C    INPUTS  - NRA = 1  NCAT = 5  NCAN = 5
C              AA(1...5) = 1.000,-0.499,0.246,-0.117,0.047  C(1) = 1.001
C              NRR = 1  NCR = 2  RR(1...2) = 1.25,.50
C              NRG = 1  G(1) = 0.
C              FF(1...4) = 0.997,-0.493,0.235,-0.094
C    USAGE   -       CALL FIRE2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,NRG,GG,FF,C)
C    OUTPUTS - FF(1...5) = 0.999,-0.498,0.246,-0.117,0.047
C
C 2. CONSTRUCTION OF A GENERAL FILTER USING RLSPR2.
C    INPUTS  - NRA = 3  NCAT = 5  NCAN = 0
C              NRR = 3  NCR = 3  RR(1...9) = 0.302, 0.105, 0.010,
C                                            1.340, 0.621, 0.020,
C                                            0.302, 0.105, 0.010
C              NRG = 1  NCG = 5  GG(1...5) = 0.,0.,1.,0.,0.
C    USAGE   -       DO 10 I=1,NCAT
C                    CALL RLSPR2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,C,IANS)
C                    IG = 1+(I-1)*NRG
C                    CALL FIRE2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,NRG,GG,FF,C)
C               10   CONTINUE
C    OUTPUTS - IANS = 0  FF(1...15) = 0.002, 0.181,-0.393, 0.181, 0.002,
C                                     0.045,-0.609, 1.417,-0.609, 0.045,
C                                     0.002, 0.181,-0.393, 0.181, 0.002
C
C
C PROGRAM FOLLOWS BELOW
C
C
      DIMENSION AA(2),RR(2),GG(2),FF(2),C(2),CM(2)
      COMMON CM
      L=NRA
      M=NCAN-1
      MT=NCAT
      LL=L*L
      LMT=L*MT
      CALL IXCARG (C,IC1)
      IC2=IC1+LL
      IC3=IC2+LL
      IC4=IC3+L
      IF (M)  10,10,20
   10 CALL STZ (LMT,FF)
   20 CONTINUE
      M1=(NRR+1)/2
      JC1=IC3
      IG1=NRG-(NRG-L)/2
      DO 50  I=1,L
      CALL DOTP (L,M,FF,NRR,NCR,RR,M1-I,1,CM(JC1),-2.)
      IF (XMINOF(IG1,NRG-IG1+1))  40,40,30
   30 CONTINUE
      CM(JC1)=CM(JC1)-GG(IG1)
   40 IG1=IG1-1
   50 JC1=JC1+1
      CALL MATML3 (L,L,1,CM(IC2),CM(IC3),0,CM(IC4),0)
      M=M+1
      LM=L*M
      IA=LM
      DO 60  I1=1,LM
      CALL DOTJ (L,1,CM(IC4),LMT,AA(IA),FF(I1),1,1)
   60 IA=IA-1
```

****
*   FIRE2   *
****
(PAGE 3)

```
      RETURN
      END
```

```
*       FIXV  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0104
*       FAP
*FIXV
        COUNT   100
        LBL     FIXV
        ENTRY   FIXV  (X,LX,IXFIXD)
        ENTRY   FIXVR (X,LX,IXFIXD)
*
*                     ----ABSTRACT----
*
* TITLE - FIXV  WITH SECONDARY ENTRY FIXVR
*        FIX A FLOATING VECTOR WITH OR WITHOUT ROUNDING
*
*               FIXV FIXES A FLOATING VECTOR TO A FORTRAN-II FIXED POINT
*               INTEGER VECTOR WITH TRUNCATION OF THE FRACTIONAL PART.
*               FIXVR ROUNDS THE FRACTIONAL PART.
*
* LANGAUGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 35 REGISTERS
* SPEED      -              7090  709
*               FIXV   31 + (27  OR 28)*LX  MACHINE CYCLES,
*               FIXVR  33 + (33  OR 34)*LX       LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL FIXV  (X,LX,IXFIXD)
*     CALL FIXVR (X,LX,IXFIXD)
*
* INPUTS
*
*    X(I)       I=1...LX  IS THE FLOATING VECTOR
*
*    LX         SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUTS IF LX LSTHN 1
*
*    IXFIXD(I) I=1...LX  IS THE FIXED FORM OF X(1...LX)
*                   WITH TRUNCATION IF FIXV  IS USED
*                   WITH ROUNDING   IF FIXVR IS USED
*               EQUIVALENCE (X, IXFIXD) IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5) = 1.2,1.5,1.9,2.0,-3.5
*               EQUIVALENCE (X, IX3)  IX4=0
*    USAGE   -        CALL FIXV (X,5,IX1)
*                     CALL FIXVR(X,5,IX2)
*                     CALL FIXV (X,1, X)
*                     CALL FIXV (X,0,IX4)
*    OUTPUTS - IX1(1...5) = 1,1,1,2,-3
*               IX2(1...5) = 1,2,2,2,-4
*               IX3(1) = X(1) = 1  IX4 = 0 (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,FIXV
* PRINCIPAL ENTRY.  FIXV(X,LX,IXFIXD)
 FIXV   CLA     NORND
 SETUP  STA     TRA
        SXD     FIXV-2,4
 K1     CLA     1,4
        ADD     K1             A(X)+1
        STA     GET
        CLA     3,4
        ADD     K1
        STA     STORE
```

```
      CLA*    2,4
      TMI     LEAVE
      PDX     0,4
      TXL     LEAVE,4,0
* FIXING LOOP
 GET  CLA     **,4          **=A(X)+1
      UFA     OCTK1
      LRS     0
      ANA     OCTK2
      LLS     0
 TRA  TRA     **            **=ALS OR ROUND
 ALS  ALS     18
 STORE STO    **,4          **=A(IXFXD)+1
      TIX     GET,4,1
* EXIT
 LEAVE LXD    FIXV-2,4
      TRA     4,4
* ROUNDING INSERTION
ROUND RQL     8
      RND
      TRA     ALS
* SECOND ENTRY.  FIXVR(X,LX,IXFIXD)
 FIXVR CLA    RND
      TRA     SETUP
* CONSTANTS
 NORND TRA    ALS
 RND  TRA     ROUND
 OCTK1 OCT    233000000000
 OCTK2 OCT    000000377777
      END
```

```
***********************
*    FIXVR            *
***********************
 REFER TO
     FIXV
```


```
***********************
*    FIXVR            *
***********************
 REFER TO
     FIXV
```


```
***********************
*    FLDATA           *
***********************
 REFER TO
     FXDATA
```


```
***********************
*    FLDATA           *
***********************
 REFER TO
     FXDATA
```

```
*       FLOATM (FUNCTION)                9/29/64   LAST CARD IN DECK IS NO. 0090
*       FAP
*FLOATM
        COUNT    80
        LBL      FLOATM
        ENTRY    FLOATM (INTEGR)
*
*                    ----ABSTRACT----
*
* TITLE - FLOATM
*         FLOAT ANY MACHINE LANGUAGE INTEGER
*
*               FLOATM ASSUMES ITS ARGUMENT IS A 35 BIT PLUS SIGN
*               INTEGER (BINARY POINT TO RIGHT OF BIT 35) AND CONVERTS
*               IT TO EQUIVALENT FLOATING POINT FORM.  THERE ARE NO
*               RESTRICTIONS ON THE ARGUMENT.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II FUNCTION)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 25 REGISTERS
* SPEED     - ABOUT 17 MACHINE CYCLES IF INTEGER LSTHN  2**27
*             ABOUT 46 MACHINE CYCLES IF INTEGER GRTHN= 2**27
* AUTHOR    - S.M. SIMPSON JR,  NOV/1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*         AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     FLTG = FLOATMF(INTEGR)
*
* INPUTS
*
*    INTEGR    IS ANY 35 BIT PLUS SIGN INTEGER
*
* OUTPUTS
*
*    FLTG      IS THE EQUIVALENT FLOATING POINT FORM OF INTEGR
*
* EXAMPLES
*
* 1. INPUTS  - INTEGR = OCT 000000000004
*    OUTPUTS - FLTG   = 4.
*
* 2. INPUTS  - INTEGR = OCT 400000000004
*    OUTPUTS - FLTG   = -4.
*
* 3. INPUTS  - INTEGR = OCT 377777777777
*    OUTPUTS - FLTG   =  34359738367.0   (GOOD TO 8 PLACES)
*
* 4. INPUTS  - INTEGR = OCT 777777777777
*    OUTPUTS - FLTG   = -34359738367.0
*
* 5. INPUTS  - INTEGR = OCT 000000000000
*    OUTPUTS - FLTG   = 0.0
*
* 6. INPUTS  - INTEGR = OCT 400000000000
*    OUTPUTS - FLTG   = -0.0
*
* 7. INPUTS  - INTEGR = OCT 001000000000
*    OUTPUTS - FLTG   =  134217728.0
*
        HTR      0
        BCI      1,FLOATM
FLOATM SXD      FLOATM-2,4
* CHECK FOR SPECIAL CASE OF MAGNITUDES EXCEEDING 2**27-1
        LAS      K001
        TRA      BIG
        TRA      BIG
        ORA      K233
        FAD      K233
 LEAVE TRA      1,4
* HANDLE BIG NUMBERS
 BIG   LRS      27
```

****
* FLOATM *
****
(PAGE 2)

```
        STQ    TEMP
        ORA    K266
        FAD    K233
        STO    TEMP2
        CLA    TEMP
        ARS    8
        ORA    K233
        FAD    K233
        FAD    TEMP2
        TRA    LEAVE
K001    OCT    001000000000
K233    OCT    233000000000
K266    OCT    266000000000
TEMP    PZE    **
TEMP2   PZE    **
        END
```

```
*      FLOATV (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0080
*      FAP
*FLOATV
       COUNT   100
       LBL     FLOATV
       ENTRY   FLOATV (IX,LIX,XFLOTD)
*
*                    ----ABSTRACT----
*
* TITLE - FLOATV
*         FLOAT A VECTOR
*
*              FLOATV CONVERTS A FORTRAN-II FIXED VECTOR TO FLOATING PT.
*
* LANGUAGE   - FAP SUBROUTINE, FORTRAN-II COMPATIBLE
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 22 REGISTERS
* SPEED      - 27 + 17.4*LX  MACHINE CYCLES WHERE LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL FLOATV(IX,LIX,XFLOTD)
*
* INPUTS
*
*    IX(I)     I=1...LIX  IS THE FIXED POINT VECTOR
*
*    LIX       MUST EXCEED ZERO
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF  LIX LSTHN 1
*
*    XFLOTD(I) I=1...LIX  IS THE FLOATED FORM OF IX(1...LIX)
*
*              EQUIVALENCE (IX, XFLOTD)  IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...3) = 1,-3,7  EQUIVALENCE (IX,X2)  X3=0.0
*
*    USAGE   -       CALL FLOATV(IX,3,X1)
*                    CALL FLOATV(IX,1,IX)
*                    CALL FLOATV(IX,-1,X3)
*    OUTPUTS - X1(1...3) = 1.0,-3.0,7.0  X2(1) = IX(1) = 1.0
*              X3 = 0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0             XR4
       BCI     1,FLOATV
* ONLY ENTRY.  FLOATV(IX,LIX,XFLOTD)
FLOATV SXD     FLOATV-2,4
 K1    CLA     1,4
       ADD     K1            A(IX)+1
       STA     GET
       CLA     3,4
       ADD     K1            A(XFLOTD)+1
       STA     STORE
       CLA*    2,4           LIX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* FLOATING LOOP
 GET   CLA     **,4          **=A(IX)+1
       LRS     18
       ORA     OCTK
       FAD     OCTK
 STORE STO     **,4          **=A(XFLTD)+1
       TIX     GET,4,1
```

```
************************
*    FLOATV            *
************************
(PAGE  2)
```


```
* EXIT
 LEAVE LXD     FLOATV-2,4
       TRA     4,4
* CONSTANTS
 OCTK  OCT     233000000000
       END
```

```
*     FMTOUT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0070
*     LABEL
CFMTOUT
      SUBROUTINE FMTOUT(ITAPE,FMT)
C
C                    ----ABSTRACT----
C
C TITLE - FMTOUT
C        WRITE OUTPUT TAPE WITH NORMAL OR LITERAL FORMAT VECTOR
C
C              FMTOUT HAS TWO ARGUMENTS, ITAPE AND FMT, WHERE FMT IS
C              A NORMAL OR LITERAL FORMAT VECTOR.  THE FUNCTION OF
C              FMTOUT IS THAT OF THE FORTRAN STATEMENT
C
C                    WRITE OUTPUT TAPE ITAPE,FMT
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
C STORAGE    - 51 REGISTERS
C SPEED      -
C AUTHOR     - S.M. SIMPSON JR.,   SEPTEMBER 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FNDFMT,RPLFMT
C       AND FORTRAN SYSTEM ROUTINES - (STH),(FIL)
C
C FORTRAN USAGE
C
C     CALL FMTOUT(ITAPE,FMT)
C
C INPUTS       DEFINE A NORMLIT FORMAT VECTOR AS EITHER
C                 A) A NORMAL FORMAT VECTOR,
C              OR B) LITERAL HOLLERITH IN A CALLING SEQUENCE WHOSE
C                 CHARACTERS (READING CONTINUOUSLY FROM LEFT TO RIGHT)
C                 ARE THE DESIRED FORMAT STRIPPED OF THE ENCLOSING
C                 PARENTHESES.  THE FIRST AND SECOND CHARACTERS MUST
C                 NOT BE QUOTE ( UNQUOTE OR QUOTE ) UNQUOTE
C                 RESPECTIVELY.  (TWO BLANKS FOLLOWED BY ( WOULD BE OK.)
C
C    ITAPE     IS LOGICAL NUMBER OF DESIRED OUTPUT TAPE.
C
C    FMT(I)    I=1,2,... OR I=1,0,-1,... IS THE NORMLIT FORMAT VECTOR
C                 TO BE WRITTEN OUT ON TAPE ITAPE.
C
C OUTPUTS      THE FORMAT IS WRITTEN OUT AS ILLUSTRATED BELOW.
C
C EXAMPLES
C
C 1. INPUTS  - FMT(1...4) = 21H(16H ORDINARY FORMAT)
C    USAGE   -       CALL FMTOUT(2,FMT)
C                    CALL FMTOUT(2,18H15H LITERAL FORMAT )
C    OUTPUTS -       THE FOLLOWING TWO LINES
C              ORDINARY FORMAT
C              LITERAL FORMAT
C                    WILL BE PRINTED OFF-LINE FROM LOGICAL TAPE 2.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION COM(2)
      COMMON COM
      CALL FNDFMT(FMT,IXCFMT)
      CALL RPLFMT(COM,COM(IXCFMT))
      GO TO 20
 10   CALL RPLFMT(COM(IXCFMT),COM)
      GO TO 9999
 20   WRITE OUTPUT TAPE ITAPE,COM
      GO TO 10
 9999 RETURN
      END
```

```
*       FNDFMT (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0202
*       FAP
*FNDFMT
        COUNT   150
        LBL     FNDFMT
        ENTRY   FNDFMT (FMT,IXCFMT)
*
*                    ----ABSTRACT----
*
* TITLE - FNDFMT
*        ACCESS TO LITERAL OR ORDINARY FORMAT
*
*              FNDFMT SUPPLIES THE INDEX WITH RESPECT TO THE COMMON
*              BLOCK OF A FORMAT STATEMENT.  THE FORMAT IS SUPPLIED
*              AS AN ARGUMENT WHICH IS EITHER OF THE ORDINARY FORM
*              (A HOLLERITH VECTOR WHOSE FIRST CHARACTER IS A LEFT
*              PARENTHESIS) OR  IS A LITERAL HOLLERITH VECTOR ARGUMENT
*              REPRESENTING THE FORMAT MINUS ITS ENCLOSING PARENTHESES
*              AND TERMINATED BY AN ALL-ONES FENCE.  IN THE LATTER
*              CASE FNDFMT REVERSES THE LITERAL HOLLERITH AND ADDS
*              THE NECESSARY PARENTHESES.  SUBSEQUENT CALLS OF FNDFMT
*              WITH THE REVERSED HOLLERITH WILL NOT LEAD TO RE-REVERSAL.
*
*              AN ORDINARY TYPE FORMAT MUST NOT CONTAIN A ) AS THE
*              FIRST CHARACTER AFTER ITS (, (ILLEGAL ANYWAY).
*
*              A LITERAL TYPE FORMAT MUST NOT CONTAIN A ) AS ITS
*              SECOND CHARACTER, OR A ( AS ITS FIRST CHARACTER.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 88 REGISTERS
* SPEED      -
* AUTHOE     - S.M. SIMPSON, SEPTEMBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - REVER
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL FNDFMT(FMT,IXCFMT)
*
* INPUTS
*
*    FMT(I)    IS A REVERSED OR UNREVERSED LITERAL HOLLERITH VECTOR, OR
*                AN ORDINARY FORMAT VECTOR, AS DESCRIBED IN ABSTRACT.
*
* OUTPUTS
*
*    IXCFMT    IS THE INDEX WITH RESPECT TO COMMON OF THE FORMAT
*              = 77461 (OCTAL) - XLOCF(FORMAT) + 1
*                WHERE XLOCF(FORMAT) = XLOCF(FMT) IF FMT(I) ORDINARY
*                                    = XLOCF(FENCE)  OTHERWISE
*                                       (THE FENCE IS WIPED OUT)
*
*    FMT(I)    IS UNDISTURBED IF, ON INPUT, IT WAS EITHER A NORMAL
*                 FORMAT VECTOR OR A PREVIOUSLY REVERSED LITERAL FORMAT
*                 VECTOR.  IF, ON INPUT, FMT(I) WAS A LITERAL FORMAT
*                 THE FOLLOWING REVERSAL TRANSFORMATION OCCURS.
*                              (INPUT)         (OUTPUT)
*                    FMT(1) =     6HABCDEF        6HZ)000M
*                    FMT(0) =     6HGHIJKL        6HTUVWXY
*                            ETC
*                    FMT(-N+1) =  6HUVWXYZ        6HFGHIJK
*                    FMT(-N) =    OCT777777777777  6H(ABCDE
*                 WHERE M = N+1
*
* EXAMPLES
*
* 1. WITH ORDINARY FORMATS
*    INPUTS   - FMT1(1...2) = 12H( I5,3X, F9.5)
*    USAGE    -        CALL FNDFMT(FMT1,IXCF1)
```

```
*    OUTPUTS - IXCF1 = 77461(OCTAL)-XLOCF(FMT1)+1, AND
*                      FMT1(1...2) IS UNCHANGED.
*
* 2. WITH LITERAL FORMATS AND REPEATED USAGE
*    INPUTS  - FMT2(3) = 6HI5,12X   FMT2(2) = 6H,5F9.5
*              FMT2(1) = OCT777777777777   FMT3(2) = 3H5I5
*              FMT3(1) = OCT777777777777
*    USAGE   -       DO 10  I=1,2
*                    CALL FNDFMT(FMT2(3),IXCF2(I))
*                10  CALL FNDFMT(FMT3(2),IXCF3(I))
*    OUTPUTS - FMT2(1) = 6H(I5,12   FMT2(2) = 6HX,5F9.
*              FMT2(3) = 6H5)0002
*              IXCF2(1) = IXCF2(2) = 77461-XLOCF(FMT2(1))+1
*              FMT3(1) = 4H(5I5   FMT3(2) = 6H )0001
*              IXCF3(1) = IXCF3(2) = 77461-XLOCF(FMT3(1))+1
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS REVER
       HTR     0              XR1
       HTR     0              XR4
       BCI     1,FNDFMT
* ONLY ENTRY.  FNDFMT(FMT,IXCFMT)
FNDFMT SXD     FNDFMT-2,4
       SXD     FNDFMT-3,1
* GET FIRST TWO CHARACTERS OF FMT(1)
       CLA*    1,4            FMT(1)
       STA     LFMT           (PUT ASIDE POSSIBLE LENGTH)
       XCA
       PXA     0,0
       LGL     6              BITS S,1...5
       STO     C1
       PXA     0,0
       LGL     6              BITS 6...11
       STO     C2
* CHECK FOR C2 = ).  IF SO, MUST BE A REVERSED FORMAT.
       CLA     C2
       CAS     RPAREN
       TRA     *+2            NO
       TRA     CASE2          YES
* IF NOT, CHECK FOR C1 = (.  IF SO, MUST BE ORDINARY FORMAT.
       CLA     C1
       CAS     LPAREN
       TRA     *+2            NO
       TRA     CASE1          YES
* IF NOT WE HAVE CASE OF UNREVERSED LITERAL HOLLERITH
*
* FIRST FIND ITS LENGTH, LFMT (DOESNT INCLUDE THE FENCE)
       AXT     0,1            XR1 IS COUNTER
       CLA     1,4            TSX A(FMT),0
       PAC     0,4            -A(FMT) TO XR4
 CAL1  CAL     0,4
       LAS     FENCE
       NOP                    (IMPOSSIBLE)
       TRA     COVER
       TXI     *+1,1,1
       TXI     CAL1,4,-1
* MAKE SETTINGS DEPENDING ON LFMT.
 COVER LXD     FNDFMT-2,4
       SXD     TXL2,1
       SXD     CARRY,1        (TEMP)
       SXA     LFMT,1
       PXA     0,1
       ADD     1,4            A(FMT)+LFMT
       STA     LDQ2
       ADD     K1             A(FMT)+LFMT+1
       STA     SLW2
       SUB     K1
       SUB     K1             A(FMT)+LFMT-1
       STA     TSX1
       STA     TSX3
* REVERSE FMT(1)...FMT(1-LFMT+1)
       TSX     $REVER,4
 TSX1  TSX     **,0           **=A(FMT)+LFMT-1
```

FNDFMT
(PAGE 3)

```
       TSX    CARRY,0
 TSX3  TSX    **,0          **=A(FMT)+LFMT-1
       LXD    FNDFMT-2,4
* INITIALIZE CARRY REGISTER TO LPAREN
       CLA    LPAREN
       STO    CARRY
* FORM FMT(-LFMT+1,-LFMT+2,...,0)
*
       AXT    1,1
 LDQ2  LDQ    **,1          **=A(FMT)+LFMT
       CAL    CARRY
       LGL    30
 SLW2  SLW    **,1          **=A(FMT)+LFMT+1
       PXA    0,0
       LGL    6
       SLW    CARRY
       TXI    *+1,1,1
 TXL2  TXL    LDQ2,1,**     **=LFMT
* THEN FORM AND SET FMT(1)
       CAL    CARRY
       LGL    30
       ACL    RPADJ
       ACL    LFMT
       SLW*   1,4
* FINALLY FORM ADDRESS OF FORMAT AS IN CASE 2
*
* CASE 2.   FORMAT HAS BEEN PREVIOUSLY REVERSED.
*     LENGTH IS GIVEN BY C4,C5,C6 (IN LFMT)
 CASE2 CLA    LFMT
       ADD    1,4           TSX A(FMT)+L,0
       TRA    LEAVE
* CASE 1.   FORMAT IS ALREADY CORRECT IN FMT
 CASE1 CLA    1,4           TSX A(FMT),0
* EXIT ROUTINE.  SETS IXCFMT GIVEN MACHINE LOCATION OF FMT IN
*     ADDRESS OF AC
* IXCFMT = 77461+1-ADDRESS
 LEAVE LXD    FNDFMT-3,1
       STA    LFMT          (SET ADDRESS ASIDE)
       CLA    KCOMON
       SUB    LFMT
       ALS    18
       STO*   2,4
       TRA    3,4
* CONSTANTS, TEMPORARIES
 K1    PZE    1
 RPADJ BCI    1,0)0000
LPAREN BCI    1,00000(
RPAREN BCI    1,00000)
KCOMON OCT    000000077462
 FENCE OCT    777777777777
 LFMT  PZE    **            **=FMT LENGTH (ALSO TEMP FOR ADDRESS)
 C1    PZE    **
 C2    PZE    **
 CARRY PZE    **            (ALSO USED AS TEMP FOR LFMT IN DECR)
       END
```

```
*      FRAME (709) (SUBROUTINE)         9/29/64   LAST CARD IN DECK IS NO. 0033
*      FAP
*FRAME  (709)
       COUNT    30
       LBL      FRAME
       ENTRY    FRAME
*
*                  ----ABSTRACT----
*
* TITLE - FRAME
*        ADVANCE FILM FRAME ON SCOPE
*              FRAME ADVANCES THE FILM IN THE SCOPE CAMERA ONE FRAME.
*
* LANGUAGE   - FAP, SUBROUTINE  (FORTRAN COMPATIBLE)
* EQUIPMENT - 709  (MAIN FRAME AND SCOPE)
* STORAGE    - 4 REGISTERS
* SPEED      - 500 MS FOR FRAME TO ADVANCE.
* AUTHOR     - R.A. WIGGINS  DEC, 1962
*
*                  ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL FRAME
*
*        THE FILM IS ADVANCED ONE FRAME
*
       BCI      1,FRAME
 FRAME WRS      24
       CFF
       TRA      1,4
       END
```

```
*      FRAME (7090) (SUBROUTINE)         9/4/64   LAST CARD IN DECK IS NO. 0046
*      FAP
*FRAME (7090)
       COUNT   40
       LBL     FRAME
       ENTRY   FRAME
*
*                    ----ABSTRACT----
*
* TITLE - FRAME (7090)
*       ADVANCE FILM FRAME ON SCOPE
*
*             FRAME ADVANCES THE FILM IN THE SCOPE CAMERA ONE FRAME.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090 (MAIN FRAME, DATA CHANNEL D, AND SCOPE)
* STORAGE   - 9 REGISTERS
* SPEED     - 500 MS FOR FRAME TO ADVANCE.
* AUTHOR    - MIT COMPUTATION CENTER STAFF
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL FRAME
*
*       THE FILM IS ADVANCED ONE FRAME.
*
*
* PROGRAM FOLLOWS BELOW
*
* FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
* TO CHANGE, ALTER THE LETTER DESIGNATION ONLY AND REASSEMBLE.
 X     TAPENO  D1
 SCPAD EQU     X-105
       BCI     1,FRAME
 FRAME PSE     SCPAD
       WRS     SCPAD
       RCHX    IOCD
       WRS     SCPAD
       RCHX    IOCD
       PSE     SCPAD            OCTAL FOR CFFX
       TRA     1,4
IOCD   IOCD    0,0,0
       END
```

```
*     FRQCT1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0094
*     LABEL
CFRQCT1
      SUBROUTINE FRQCT1(IX,NX,IXLO,IXHI,ICT,IANS)
C
C                        ----ABSTRACT----
C
C TITLE - FRQCT1
C        FREQUENCY DISTRIBUTION OF A FIXED POINT VECTOR
C
C              FRQCT1 MAKES A FREQUENCY COUNT OF AN INTEGER SEQUENCE WITH
C              VALUES IN A SPECIFIED RANGE.  FOR EACH INTEGER VALUE IN
C              THE INCLUSIVE RANGE IXLO TO IXHI, THE NUMBER OF
C              OCCURRENCES OF THIS VALUE IN THE INTEGER SEQUENCE IS
C              COUNTED.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 117 REGISTERS
C SPEED     -
C AUTHOR    - S. M. SIMPSON
C
C                        ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL FRQCT1(IX,NX,IXLO,IXHI,ICT,IANS)
C
C INPUTS
C
C    IX(I)      I=1...NX IS THE GIVEN INTEGER SEQUENCE
C               IXLO LSTHN OR = IX(I) LSTHN OR = IXHI.
C
C    NX         IS THE NUMBER OF IX VALUES IN THE SEQUENCE.
C               MUST BE GRTHN 0.
C
C    IXLO       IS AN INTEGER
C               LSTHN OR = ALL IX(I)
C               IXLO MAY BE NEG.
C
C    IXHI       IS AN INTEGER
C               GRTHN OR = ALL IX(I)
C               IXHI MAY BE NEG.
C
C OUTPUTS
C
C    ICT(I)     I=1...NCT IS THE FREQUENCY COUNT WHERE
C                    ICT(1) = NUMBER OF MEMBERS OF THE INPUT SEQ = IXLO
C                    ICT(2) = NUMBER OF MEMBERS OF THE INPUT SEQ = IXLO+1
C                                   ETC.
C                    ICT(NCT) = NUMBER OF MEMBERS OF THE INPUT SEQ = IXHI
C                    WHERE NCT = IXHI-IXLO+1
C
C    IANS       = 0  NORMAL
C               = 1  ILLEGAL NX
C               = 2  ILLEGAL IXLO
C
C EXAMPLES OF FRQCT1
C
C 1. INPUTS  - IXLO=3     IXHI=10     NX=3     IX(1...3)=4,4,4
C    OUTPUTS - ICT(1...8) = 0,3,0,0,0,0,0,0   IANS=0
C
C 2. INPUTS  - IXLO=5     IXHI=12     NX=7     IX(1...7)=5,6,7,8,9,10,11
C    OUTPUTS - ICT(1...8) = 1,1,1,1,1,1,1,0   IANS=0
C
C 3. INPUTS  - IXLO=5     IXHI=12     NX=0
C    OUTPUTS - ERROR     IANS=1
C
C 4. INPUTS  - IXLO=13     IXHI=12     NX=7
C    OUTPUTS - ERROR     IANS=2
C
      DIMENSION  IX(2),ICT(2)
C SET UP AND CLEAR ICT(I).
```

* FRQCT1 *
****
(PAGE 2)

```
      IANS=0
      NCT=IXHI-IXLO+1
      NSHIFT=IXLO-1
      IF (NX)  9991,9991,10
   10 IF (NCT)  9992,9992,15
   15 DO 20  I=1,NCT
   20 ICT(I)=0
C SCAN IX(I) TO MAKE COUNTS (PUT EACH IX IN RANGE 1 TO NCT FIRST).
      DO 35  I=1,NX
      IXI=IX(I)-NSHIFT
      IF (IXI)  9992,9992,30
   30 IF (IXI-NCT)  35,35,9992
   35 ICT(IXI)=ICT(IXI)+1
      GO TO 9999
 9999 RETURN
 9991 IANS=1
      GO TO 9999
 9992 IANS=2
      GO TO 9999
      END
```

```
*      FRQCT2 (SUBROUTINE)            9/29/64   LAST CARD IN DECK IS NO. 0211
*      FAP
*FRQCT2
       COUNT   200
       LBL     FRQCT2
       ENTRY   FRQCT2 (X,LX,B,LB,ICOUNT,IANS)
*
*                    ----ABSTRACT----
*
* TITLE - FRQCT2
*        FREQUENCY COUNT OF NUMBER OF VALUES OF A SERIES IN GIVEN RANGES.
*
*               FRQCT2 MAKES A FREQUENCY COUNT OF A FLOATING POINT,
*               FORTRAN INTEGER, OR MACHINE LANGUAGE INTERGER SERIES FOR
*               THE NUMBER OF VALUES LYING IN SPECIFIED RANGES. IT IS
*               USEFUL IN COMPUTING EMPIRICAL PROBABILITY DENSITIES.
*
*               THERE ARE LB RANGE LIMITS, B(I), I=1, LB, AND HENCE LB+1
*               RANGES. A NUMBER, X(J), IS SAID TO BE IN THE I-TH RANGE
*               IF B(I-1) LSTHN OR EQUAL X(J) LSTHN B(I). A NUMBER IS IN
*               THE FIRST RANGE IF IT IS LSTHN B(1), AND IN THE LB+1
*               RANGE IF GRTHN OR EQUAL B(LB). THE INPUT SERIES X(I) MUST
*               BE THE SAME MODE (FLOATING, INTEGER, ETC.) AS THE RANGE
*               LIMITS BECAUSE THE METHOD USES CAS INSTRUCTIONS.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 117 REGISTERS
* SPEED     -
* AUTHOR    - J. N. GALBRAITH
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL FRQCT2(X,LX,B,LB,ICOUNT,IANS)
*
* INPUTS
*
*    X(I)       I=1...LX IS THE GIVEN SERIES.
*               MAY BE FLOATING, FORTRAN INTEGER, OR MACHINE INTEGER.
*
*    LX         IS THE LENGTH OF THE X SERIES.
*               MUST BE GRTHN 0.
*
*    B(I)       I=1...LB IS VECTOR OF RANGE LIMITS. B(I) LSTHN B(I+1).
*               RANGES INTO WHICH THE SERIES IS DIVIDED ARE (-INFINITY,
*               LSTHN B(1)),(GRTHN OR =B(1),LSTHN B(2)) ETC.
*               MAY BE FLOATING, FORTRAN INTEGER, OR MACHINE INTEGER,
*                 BUT MUST BE THE SAME AS X(I)
*
*    LB         NUMBER OF RANGE LIMITS.
*               MUST BE GRTHN 0.
*               NOTE - NUMBER OF RANGES =1+ NUMBER OF RANGE LIMITS.
*
* OUTPUTS
*
*    ICOUNT(I) I=1...LB+1=NUMBER OF X VALUES IN EACH RANGE OF B.
*               ICOUNT(1)=NO. X LSTHN B(1). ICOUNT(2)=NO. X LSTHN B(2),
*                 GRTHN OR =B(1).
*               ICOUNT(LB)=NO. X LSTHN B(LB),GRTHN OR=B(LB-1).
*               ICOUNT(LB+1)=NO. X GRTHN OR =B(LB).
*
*    IANS       IANS=0, NORMAL
*               IANS=1, ILLEGAL LX
*               IANS=2, ILLEGAL LB
*               IANS=3, WEIRD ERROR
*
* EXAMPLES
*
* 1. INPUTS  - X(1...15) = -21.,-20.,-15.,-14.,-12.,-11.,-8.,-7.,0.,1.,
*               2.1,3.,4.,5.,6.   LX=15  B(1...5)= -20.,-16.,-7.5,0.,.9
*               LB=5
```

```
*    OUTPUTS - ICOUNT(1...6) = 1,1,5,1,1,6,    IANS=0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT B(1...5)=-21.,-11.5,0.,4.5,6.
*    OUTPUTS - ICOUNT(1...6) =0,5,3,5,1,1   IANS=0
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT B(1...5)=-21.,-11.5,0,4.5,6.1
*    OUTPUTS - ICOUNT(1...6) =0,5,3,5,2,0   IANS=0
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT B(1)=0. B(2)=.5  LB=2
*    OUTPUTS - ICOUNT(1...3) =8,1,6   IANS=0
*
* 5. INPUTS  - SAME AS EXAMPLE 4. EXCEPT LB=0
*    OUTPUTS - ERROR IANS =2
*
* 6. INPUTS  - SAME AS EXAMPLE 4. EXCEPT  LX=0  LB=2
*    OUTPUTS - ERROR IANS = 1
*
* SAVE IRS AND CHECK FOR ILLEGAL PARAMETERS
       PZE    0
       BCI    1,FRQCT2
FRQCT2 SXA    RETURN,1
       SXA    RETURN+1,2
       SXA    RETURN+2,4
       SXD    FRQCT2-2,4
       STZ*   6,4           IANS=0
       CLA*   2,4           GET LX
       TZE    ERR1
       TMI    ERR1
       STD    END
       CLA*   4,4           GET LB
       TZE    ERR2
       TMI    ERR2
       ARS    18            LB IN ADDRESS
       STO    LB
       ARS    1             LB/2 (IN ADDRESS)
       STO    LBHALF
       CLA    1,4           ADDRESS OF X
       ADD    K1MLI         A(X+1)
       STA    XADD
       STA    TESTLO
       CLA    3,4           ADDRESS OF B
       ADD    K1MLI         A(B+1)
       STA    BTEST1
       STA    BADD
       SUB    LB
       STA    TESTHI
       CLA    5,4           ADDRESS OF ICOUNT
       ADD    K1MLI         A(ICOUNT+1)
       STA    STZCNT
       STA    EQUAL
       STA    STOCNT
       LXA    LB,1
       TXI    *+1,1,1
       SXD    END1,1
       AXT    1,4
       AXT    1,1
STZCNT STZ    **,1          ZERO ICOUNT(I),I=1,LB+1
       TXI    *+1,1,1
 END1  TXL    STZCNT,1,**   **=LB+1
       AXT    1,1
 LOOP  CLA    K1MLI
       STO    LBLO          INITIAL LBLO=1
       CLA    LB
       STO    LBHI          INITIAL LBHI=LB
       CLA    LBHALF
       STO    LBCOM         INITIAL LBCOM=LB/2
       AXT    1,2
TESTLO CLA    **,1          GET X.  (**=A(X+1))
BTEST1 CAS    **,4          B(1)  SEE IF IN LOWEST RANGE
       TRA    TESTHI
       TRA    NEXIND
       TRA    EQUAL
TESTHI CAS    **            **=A(B(LB)). SEE IF IN HIGHEST RANGE
       TRA    HIEST
       TRA    HIEST
```

```
      SEARCH LXA    LBCOM,2
       XADD  CLA    **,1         GET X(IR1)
       BADD  CAS    **,2         COMPARE WITH B(LBCOM)
             TRA    GRATER       X GREATER, NEW LBLO (=LBCOM)
             TRA    NEXIND       GOT IT, INDEX ICOUNT(IR2+1)
       LESS  PXA    0,2          X LESS, NEW LBHI (=LBCOM)
             SUB    LBLO         LBCOM-LBLO=DIF
             CAS    K1MLI
             TRA    *+3          DIF GREATER THAN ONE
             TRA    EQUAL        DIF=1, GOT IT, INDEX ICOUNT(IR2)
             TRA    ERROR           IMPOSSIBLE
             ARS    1            DIF/2
             ADD    LBLO         NEW LBCOM
             LDQ    LBCOM
             STQ    LBHI
             STO    LBCOM
             TRA    SEARCH
      GRATER PXA    0,2
             SUB    LBHI         LBCOM-LBHI=-DIF
             SSP                 DIF
             CAS    K1MLI
             TRA    *+3
             TRA    NEXIND       GOT IT, INDEX ICOUNT(IR2+1)
             TRA    ERROR        IMPOSSIBLE
             ARS    1
             ADD    LBCOM
             LDQ    LBCOM
             STO    LBCOM
             STQ    LBLO
             TRA    SEARCH
      NEXIND TXI    *+1,2,1
       EQUAL CLA    **,2         **=A(ICOUNT+1)
             ADD    K1FX
      STOCNT STO    **,2         **=A(ICOUNT+1)
             TXI    *+1,1,1
       END   TXL    LOOP,1,**    **=LX
      RETURN AXT    **,1
             AXT    **,2
             AXT    **,4
             TRA    7,4
       HIEST LXA    LB,2
             TRA    NEXIND
       ERR1  CLA    K1FX
             STO*   6,4
             TRA    7,4
       ERR2  CLA    K2FX
             STO*   6,4
             TRA    7,4
       ERROR CLA    K3FX
             STO*   6,4
             TRA    7,4
      * CONSTANTS AND TEMPORARIES
       K1FX  PZE    0,0,1
       K2FX  PZE    0,0,2
       K3FX  PZE    0,0,3
       K1MLI PZE    1,0,0
       LB    PZE    0
      LBHALF PZE    0
      LBLO   PZE    0
      LBCOM  PZE    0
      LBHI   PZE    0
             END
```

```
*       FSKIP (SUBROUTINE)                  9/4/64   LAST CARD IN DECK IS NO. 0103
*       FAP
*FSKIP
        COUNT   75
        LBL     FSKIP
        ENTRY   FSKIP (ITAPE,NFILES)
*
*
*                    ----ABSTRACT----
*
* TITLE - FSKIP
*       SKIP FORWARD OR BACKWARD OVER FILES ON TAPE
*
*              FSKIP SKIPS AN ARBITRARY NUMBER OF FILES FORWARD OR
*              BACKWARD ON A TAPE.  THE END POSITION OF THE TAPE IS
*              AT THAT EDGE OF THE FILE MARK WHICH IS FURTHEST FROM
*              THE BEGINNING OF THE TAPE.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME, DATA CHANNEL AND TAPE UNIT)
* STORAGE    - 50 REGISTERS
* SPEED     -
* AUTHOR    - J.F. CLAERBOUT, AUGUST, 1962
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   (IOS), (RDS), (BSR), (TCO),
*                                       (TEF), (TRC)
* FORTRAN USAGE
*     CALL FSKIP (ITAPE,NFILES)
*
*
* INPUTS
*
*    ITAPE     IS LOGICAL TAPE NUMBER
*              IS FORTRAN II INTEGER
*
*    NFILES    IS NUMBER OF FILES TO BE SKIPPED.
*              IF GRTHN 0 SKIPS AHEAD.
*              IF LSTHN 0 BACKS UP NFILES (OR TO LOAD POINT, WHICHEVER
*                COMES FIRST). IF THE TAPE IS PRESENTLY PART WAY THROUGH
*                A FILE, IT COUNTS AS ONE FILE.
*                A FILE IS DEFINED TO BE ARBITRARY NUMBER OF BCD OR
*                BINARY RECORDS FOLLOWED BY AN END OF FILE.
*              IF =0 DOES NOT MOVE THE TAPE
*              IS FORTRAN II INTEGER
*
*
* OUTPUTS    THE TAPE IS MOVED
*
*
* PROGRAM FOLLOWS BELOW
*
        BCI     1,FSKIP
 FSKIP SXA      SKIP9,4
        CLA*    1,4
        STD     SKIP3
        CAL     SKIP3
        TSX     $(IOS),4
        LXA     SKIP9,4
        CAL*    $(RDS)          SET UP
        ANA     =O7000
        STA     SKPBT1          FOR BTT
        CLA*    $(BSR)
        SSM
        STO     SKPBSF          SET UP BACKSPACE FILE INSTRUCTION.
        LDQ*    $(TCO)          SET UP
        SLQ     SKIP4           CHANNEL DELAY.
        SLQ     SKIP5
        LDQ*    $(TEF)
        SLQ     SKPTEF
        SLQ     A
        LDQ*    $(TRC)
```

****
* FSKIP *
****
(PAGE 2)

```
         SLQ     TRC1
         CLA*    2,4
         TZE     3,4          NO SKIPPING WANTED.
         PDX     ,4           SET FOR N FILE JUMPS
         TPL     SKIP1        SKIP AHEAD ON TAPE.
*                             SKIP BACK.
         XEC*    $(BSR)       GET OVER EOF MARK WHICH IS JUST BEFORE
*                             PRESENT POSITION.
SKPBSF   BSFA    **           GO BACK
         TIX     *-1,4,1      OVER N FILES.
         XEC*    $(RDS)       PASS OVER EOF
 SKIP4   TCOA    *
 A       TEFA    SKIP9
         XEC*    $(BSR)       MUST BE AT BEGINNING OF TAPE
SKPBT1   BTT     **           TURN OFF BEGINNING OF TAPE LIGHT.
         TRA     SKIP9        AT BEGINNING OF TAPE
* DONE BACK SKIP
*
* DONE FORWARD SKIP
 SKIP1   XEC*    $(RDS)       PASS A RECORD
 SKIP5   TCOA    *            DELAY.
SKPTEF   TEFA    SKIP2        GO TO EOF COUNTER.
         TRA     SKIP1        NO EOF, KEEP PASSING RECORDS.
 SKIP2   TIX     SKIP1,4,1    COUNT EOF*S.
 SKIP9   AXT     **,4
 TRC1    TRCA    *+1
         TRA     3,4
 SKIP3   PZE     16
         END
```

```
*      FT24  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0847
*      FAP
*FT24
       COUNT   750
       LBL     FT24
       ENTRY   FT24 (D,A,B)
*
*                    ----ABSTRACT----
*
* TITLE - FT24
*       HIGH SPEED 24 POINT SPECTRUM
*
*             FT24 COMPUTES THE SINE AND COSINE TRANSFORMS OF 24 DATA
*             POINTS.  THE TRANSFORMS ARE EVALUATED AT FREQUENCIES
*
*                 FREQ  = (I-1)*PI/12     I=1...13
*
*             WHERE PI = 3.14159265
*               AND FREQ = PI IS EQUIVALENT TO THE FOLDING FREQUENCY
*                               FOR THE DATA SERIES
*
*             FT24 GAINS ITS SPEED FROM
*
*               1. STRAIGHT LINE PROGRAMMING RATHER THAN IN LOOPS
*               2. GROUPING TERMS TO MINIMIZE THE NUMBER OF MULTIPLIES
*                    NECESSARY
*               3. SUBGROUPING ADDITIONS TO TAKE ADVANTAGE OF VARIOUS
*                    SYMMETRIES
*               4. SELECTION OF THE NUMBER OF FREQUENCIES SO AS TO
*                    MAXIMIZE THE NUMBER OF SYMMETRIES GENERATED
*               5. USING FIXED POINT ARITHMETIC
*
*             THE EQUATIONS USED WERE DEVELOPED IN SCIENTIFIC REPORT
*             NO. 1 OF AIR FORCE CONTRACT AF 19(604)7378, APPENDIX J.
*
* LANGUAGE  - FAP SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - IBM 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 777 REGISTERS
* SPEED     - ABOUT 4750 MACHINE CYCLES.
* AUTHOR    - CHEH PAN
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - FXDATA, FLDATA
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL FT24  (D,A,B)
*
* INPUTS
*    D(I)      I=1...24 IS THE DATA VECTOR THE TRANSFORM IS TO BE
*                MADE OF.
* OUTPUTS
*    A(I)      I=1...13 IS THE COSINE TRANSFORM
*                A(1) = (1/24) * (SUM (FROM I=1 TO 24) OF D(I))
*                A(13)= (1/24) * (SUM (FROM I=1 TO 24) OF
*                                     D(I)*COS((I-1)*PI))
*                A(J) = (1/12) * (SUM (FROM I=1 TO 24) OF
*                                     D(I)*COS((J-1)*(I-1)*PI))
*                     FOR J = 2,3,...,12
*
*    B(I)      I=1...13 IS THE SINE TRANSFORM
*                B(1) = B(13) = 0.0
*                B(J) = (1/12) * (SUM (FROM I=1 TO 24) OF
*                                     D(I)*SIN((J-1)*(I-1)*PI))
*                     FOR J = 2,3,...,12
*
* EXAMPLES
*
* 1. INPUTS  - D(1...24) = 12.,12.,0.,0.,...,0.
*
*    OUTPUTS - A(1...13) = 1.,1.966,1.866,1.707,1.500,1.259,1.000,
*                               0.741,0.500,0.293,0.134,0.034,0.000
```

```
 *              B(1...13) = 0.,0.259,0.500,0.707,0.866,0.966,1.000,
 *                                0.966,0.866,0.707,0.500,0.259,0.000
 *
        HTR     0
        BCI     1,FT24
  FT24  SXD     *-2,4
        CLA     1,4
        ADD     =1
        STA     MOVED
        CLA     2,4
        ADD     =1
        STA     MOVEA
        CLA     3,4
        ADD     =1
        STA     MOVEB
 *      MOVE DATA INTO PROGRAM
        AXT     24,4
  MOVED CLA     **,4
        STO     X0+1,4
        TIX     *-2,4,1
        CALL    FXDATA,KD24,X0,MXDATA,SCALE
 *      INSERT  INDIVIDUAL FORMULAE
  CA0   CLM
        AXT     24,4
        ADD     X0+1,4
        TIX     *-1,4,1
        STO     A0
  CA12  CLM
        AXT     1,4
        ADD     X0+1,4
        SUB     X0,4
        TXI     *+1,4,2
        TXL     *-3,4,24
        STO     A12
  CA1   CLA     X0
        SUB     X12
        STO     A1
        CLA     X1
        SUB     X11
        SUB     X13
        ADD     X23
        XCA
        MPY     C1
        ADD     A1
        STO     A1
        CLA     X2
        SUB     X10
        SUB     X14
        ADD     X22
        XCA
        MPY     C2
        ADD     A1
        STO     A1
        CLA     X3
        SUB     X9
        SUB     X15
        ADD     X21
        XCA
        MPY     C3
        ADD     A1
        STO     A1
        CLA     X4
        SUB     X8
        SUB     X16
        ADD     X20
        ARS     1
        ADD     A1
        STO     A1
        CLA     X5
        SUB     X7
        SUB     X17
        ADD     X19
        XCA
        MPY     C5
        ADD     A1
```

```
        STO     A1
CB1     CLA     X6
        SUB     X18
        STO     B1
        CLA     X1
        ADD     X11
        SUB     X13
        SUB     X23
        XCA
        MPY     S1
        ADD     B1
        STO     B1
        CLA     X2
        ADD     X10
        SUB     X14
        SUB     X22
        ARS     1
        ADD     B1
        STO     B1
        CLA     X3
        ADD     X9
        SUB     X15
        SUB     X21
        XCA
        MPY     S3
        ADD     B1
        STO     B1
        CLA     X4
        ADD     X8
        SUB     X16
        SUB     X20
        XCA
        MPY     S4
        ADD     B1
        STO     B1
        CLA     X5
        ADD     X7
        SUB     X17
        SUB     X19
        XCA
        MPY     S5
        ADD     B1
        STO     B1
CA2     CLA     X0
        SUB     X6
        ADD     X12
        SUB     X18
        STO     A2
        CLA     X1
        SUB     X5
        SUB     X7
        ADD     X11
        ADD     X13
        SUB     X17
        SUB     X19
        ADD     X23
        XCA
        MPY     C2
        ADD     A2
        STO     A2
        CLA     X2
        SUB     X4
        SUB     X8
        ADD     X10
        ADD     X14
        SUB     X16
        SUB     X20
        ADD     X22
        ARS     1
        ADD     A2
        STO     A2
CB2     CLA     X3
        SUB     X9
        ADD     X15
        SUB     X21
```

FT24
(PAGE 4)

```
         STO    B2
         CLA    X1
         ADD    X5
         SUB    X7
         SUB    X11
         ADD    X13
         ADD    X17
         SUB    X19
         SUB    X23
         ARS    1
         ADD    B2
         STO    B2
         CLA    X2
         ADD    X4
         SUB    X8
         SUB    X10
         ADD    X14
         ADD    X16
         SUB    X20
         SUB    X22
         XCA
         MPY    S4
         ADD    B2
         STO    B2
CA3      CLA    X0
         SUB    X4
         ADD    X8
         SUB    X12
         ADD    X16
         SUB    X20
         STO    A3
         CLA    X1
         SUB    X3
         SUB    X5
         ADD    X7
         ADD    X9
         SUB    X11
         SUB    X13
         ADD    X15
         ADD    X17
         SUB    X19
         SUB    X21
         ADD    X23
         XCA
         MPY    C3
         ADD    A3
         STO    A3
CB3      CLA    X2
         SUB    X6
         ADD    X10
         SUB    X14
         ADD    X18
         SUB    X22
         STO    B3
         CLA    X1
         ADD    X3
         SUB    X5
         SUB    X7
         ADD    X9
         ADD    X11
         SUB    X13
         SUB    X15
         ADD    X17
         ADD    X19
         SUB    X21
         SUB    X23
         XCA
         MPY    S3
         ADD    B3
         STO    B3
CA4      CLA    X0
         SUB    X3
         ADD    X6
         SUB    X9
         ADD    X12
```

****
*   FT24   *
****
(PAGE 5)

```
             SUB     X15
             ADD     X18
             SUB     X21
             STO     A4
             CLA     X1
             SUB     X2
             SUB     X4
             ADD     X5
             ADD     X7
             SUB     X8
             SUB     X10
             ADD     X11
             ADD     X13
             SUB     X14
             SUB     X16
             ADD     X17
             ADD     X19
             SUB     X20
             SUB     X22
             ADD     X23
             ARS     1
             ADD     A4
             STO     A4
       CB4   CLA     X1
             ADD     X2
             SUB     X4
             SUB     X5
             ADD     X7
             ADD     X8
             SUB     X10
             SUB     X11
             ADD     X13
             ADD     X14
             SUB     X16
             SUB     X17
             ADD     X19
             ADD     X20
             SUB     X22
             SUB     X23
             XCA
             MPY     S4
             STO     B4
       CA5   CLA     X0
             SUB     X12
             STO     A5
             CLA     X5
             SUB     X7
             SUB     X17
             ADD     X19
             XCA
             MPY     C1
             ADD     A5
             STO     A5
             CLS     X2
             ADD     X10
             ADD     X14
             SUB     X22
             XCA
             MPY     C2
             ADD     A5
             STO     A5
             CLS     X3
             ADD     X9
             ADD     X15
             SUB     X21
             XCA
             MPY     C3
             ADD     A5
             STO     A5
             CLA     X4
             SUB     X8
             SUB     X16
             ADD     X20
             ARS     1
             ADD     A5
```

```
          STO     A5
          CLA     X1
          SUB     X11
          SUB     X13
          ADD     X23
          XCA
          MPY     C5
          ADD     A5
          STO     A5
    CB5   CLA     X6
          SUB     X18
          STO     B5
          CLA     X5
          ADD     X7
          SUB     X17
          SUB     X19
          XCA
          MPY     S1
          ADD     B5
          STO     B5
          CLA     X2
          ADD     X10
          SUB     X14
          SUB     X22
          ARS     1
          ADD     B5
          STO     B5
          CLS     X3
          SUB     X9
          ADD     X15
          ADD     X21
          XCA
          MPY     S3
          ADD     B5
          STO     B5
          CLS     X4
          SUB     X8
          ADD     X16
          ADD     X20
          XCA
          MPY     S4
          ADD     B5
          STO     B5
          CLA     X1
          ADD     X11
          SUB     X13
          SUB     X23
          XCA
          MPY     S5
          ADD     B5
          STO     B5
    CA6   CLA     X0
          SUB     X2
          ADD     X4
          SUB     X6
          ADD     X8
          SUB     X10
          ADD     X12
          SUB     X14
          ADD     X16
          SUB     X18
          ADD     X20
          SUB     X22
          STO     A6
    CB6   CLA     X1
          SUB     X3
          ADD     X5
          SUB     X7
          ADD     X9
          SUB     X11
          ADD     X13
          SUB     X15
          ADD     X17
          SUB     X19
          ADD     X21
```

FT24

(PAGE 7)

```
         SUB   X23
         STO   B6
  CA7    CLA   X0
         SUB   X12
         STO   A7
         CLS   X5
         ADD   X7
         ADD   X17
         SUB   X19
         XCA
         MPY   C1
         ADD   A7
         STO   A7
         CLS   X2
         ADD   X10
         ADD   X14
         SUB   X22
         XCA
         MPY   C2
         ADD   A7
         STO   A7
         CLA   X3
         SUB   X9
         SUB   X15
         ADD   X21
         XCA
         MPY   C3
         ADD   A7
         STO   A7
         CLA   X4
         SUB   X8
         SUB   X16
         ADD   X20
         ARS   1
         ADD   A7
         STO   A7
         CLS   X1
         ADD   X11
         ADD   X13
         SUB   X23
         XCA
         MPY   C5
         ADD   A7
         STO   A7
  CB7    CLS   X6
         ADD   X18
         STO   B7
         CLA   X5
         ADD   X7
         SUB   X17
         SUB   X19
         XCA
         MPY   S1
         ADD   B7
         STO   B7
         CLS   X2
         SUB   X10
         ADD   X14
         ADD   X22
         ARS   1
         ADD   B7
         STO   B7
         CLS   X3
         SUB   X9
         ADD   X15
         ADD   X21
         XCA
         MPY   S3
         ADD   B7
         STO   B7
         CLA   X4
         ADD   X8
         SUB   X16
         SUB   X20
         XCA
```

```
        MPY     S4
        ADD     B7
        STO     B7
        CLA     X1
        ADD     X11
        SUB     X13
        SUB     X23
        XCA
        MPY     S5
        ADD     B7
        STO     B7
CA8     CLA     X0
        ADD     X3
        ADD     X6
        ADD     X9
        ADD     X12
        ADD     X15
        ADD     X18
        ADD     X21
        STO     A8
        CLS     X1
        SUB     X2
        SUB     X4
        SUB     X5
        SUB     X7
        SUB     X8
        SUB     X10
        SUB     X11
        SUB     X13
        SUB     X14
        SUB     X16
        SUB     X17
        SUB     X19
        SUB     X20
        SUB     X22
        SUB     X23
        ARS     1
        ADD     A8
        STO     A8
CB8     CLA     X1
        SUB     X2
        ADD     X4
        SUB     X5
        ADD     X7
        SUB     X8
        ADD     X10
        SUB     X11
        ADD     X13
        SUB     X14
        ADD     X16
        SUB     X17
        ADD     X19
        SUB     X20
        ADD     X22
        SUB     X23
        XCA
        MPY     S4
        STO     B8
CA9     CLA     X0
        SUB     X4
        ADD     X8
        SUB     X12
        ADD     X16
        SUB     X20
        STO     A9
        CLS     X1
        ADD     X3
        ADD     X5
        SUB     X7
        SUB     X9
        ADD     X11
        ADD     X13
        SUB     X15
        SUB     X17
        ADD     X19
```

* FT24 *
(PAGE 9)

```
         ADD    X21
         SUB    X23
         XCA
         MPY    C3
         ADD    A9
         STO    A9
   CB9   CLS    X2
         ADD    X6
         SUB    X10
         ADD    X14
         SUB    X18
         ADD    X22
         STO    B9
         CLA    X1
         ADD    X3
         SUB    X5
         SUB    X7
         ADD    X9
         ADD    X11
         SUB    X13
         SUB    X15
         ADD    X17
         ADD    X19
         SUB    X21
         SUB    X23
         XCA
         MPY    S3
         ADD    B9
         STO    B9
   CA10  CLA    X0
         SUB    X6
         ADD    X12
         SUB    X18
         STO    A10
         CLS    X1
         ADD    X5
         ADD    X7
         SUB    X11
         SUB    X13
         ADD    X17
         ADD    X19
         SUB    X23
         XCA
         MPY    C2
         ADD    A10
         STO    A10
         CLA    X2
         SUB    X4
         SUB    X8
         ADD    X10
         ADD    X14
         SUB    X16
         SUB    X20
         ADD    X22
         ARS    1
         ADD    A10
         STO    A10
   CB10  CLA    X3
         SUB    X9
         ADD    X15
         SUB    X21
         STO    B10
         CLA    X1
         ADD    X5
         SUB    X7
         SUB    X11
         ADD    X13
         ADD    X17
         SUB    X19
         SUB    X23
         ARS    1
         ADD    B10
         STO    B10
         CLS    X2
         SUB    X4
```

****
*   FT24   *
****
(PAGE 10)

```
        ADD     X8
        ADD     X10
        SUB     X14
        SUB     X16
        ADD     X20
        ADD     X22
        XCA
        MPY     S4
        ADD     B10
        STO     B10
  CA11  CLA     X0
        SUB     X12
        STO     A11
        CLS     X1
        ADD     X11
        ADD     X13
        SUB     X23
        XCA
        MPY     C1
        ADD     A11
        STO     A11
        CLA     X2
        SUB     X10
        SUB     X14
        ADD     X22
        XCA
        MPY     C2
        ADD     A11
        STO     A11
        CLS     X3
        ADD     X9
        ADD     X15
        SUB     X21
        XCA
        MPY     C3
        ADD     A11
        STO     A11
        CLA     X4
        SUB     X8
        SUB     X16
        ADD     X20
        ARS     1
        ADD     A11
        STO     A11
        CLS     X5
        ADD     X7
        ADD     X17
        SUB     X19
        XCA
        MPY     C5
        ADD     A11
        STO     A11
  CB11  CLS     X6
        ADD     X18
        STO     B11
        CLA     X1
        ADD     X11
        SUB     X13
        SUB     X23
        XCA
        MPY     S1
        ADD     B11
        STO     B11
        CLS     X2
        SUB     X10
        ADD     X14
        ADD     X22
        ARS     1
        ADD     B11
        STO     B11
        CLA     X3
        ADD     X9
        SUB     X15
        SUB     X21
        XCA
```

```
          MPY    S3
          ADD    B11
          STO    B11
          CLS    X4
          SUB    X8
          ADD    X16
          ADD    X20
          XCA
          MPY    S4
          ADD    B11
          STO    B11
          CLA    X5
          ADD    X7
          SUB    X17
          SUB    X19
          XCA
          MPY    S5
          ADD    B11
          STO    B11
          LDQ    SCALE
          FMP    =12.
          STO    SCALE
          CALL   FLDATA,KD26,B0,SCALE
          LDQ    A0
          FMP    =.5
          STO    A0
          LDQ    A12
          FMP    =.5
          STO    A12
          AXT    13,4         MOVE COEFS
          CLA    A0+1,4       BACK TO MAIN
   MOVEA  STO    **,4
          CLA    B0+1,4
   MOVEB  STO    **,4
          TIX    MOVEA-1,4,1
    SV4   LXD    FT24-2,4
          TRA    4,4
   MXDATA PZE    0,0,100000
    SCALE PZE
    KD26  PZE    0,0,26
    KD24  PZE    0,0,24
    S1    OCT    102203734074        SIN(PI/12)
    S3    OCT    265011714631        SIN(PI/4)
    S4    OCT    335547535014       SIN(PI/3)
    S5    OCT    367215650717           SIN(5*PI/12)
    C1    EQU    S5                   COS(PI/2)
    C2    EQU    S4           COS(PI/6)
    C3    EQU    S3            COS(PI/4)
    C5    EQU    S1           COS(5*PI/12)
    A12   PZE
    A11   PZE
    A10   PZE
    A9    PZE
    A8    PZE
    A7    PZE
    A6    PZE
    A5    PZE
    A4    PZE
    A3    PZE
    A2    PZE
    A1    PZE
    A0    PZE
    B12   PZE                 ALWAYS ZERO
    B11   PZE
    B10   PZE
    B9    PZE
    B8    PZE
    B7    PZE
    B6    PZE
    B5    PZE
    B4    PZE
    B3    PZE
    B2    PZE
    B1    PZE
    B0    PZE                 ALWAYS ZERO
```

****
*   FT24   *
****
(PAGE 12)

```
X23     PZE
X22     PZE
X21     PZE
X20     PZE
X19     PZE
X18     PZE
X17     PZE
X16     PZE
X15     PZE
X14     PZE
X13     PZE
X12     PZE
X11     PZE
X10     PZE
X9      PZE
X8      PZE
X7      PZE
X6      PZE
X5      PZE
X4      PZE
X3      PZE
X2      PZE
X1      PZE
X0      PZE
        END
```

```
*      FT24  -II (SUBROUTINE)           9/29/64   LAST CARD IN DECK IS NO. 0146
*      LABEL
CFT24  -II
       SUBROUTINE FT24 (DD,AA,BB)
C
C                    ----ABSTRACT----
C
C TITLE - FT24  -II
C       HIGH SPEED 24 POINT SPECTRUM
C
C             FT24 COMPUTES THE SINE AND COSINE TRANSFORMS OF 24 DATA
C             POINTS.  THE TRANSFORMS ARE EVALUATED AT FREQUENCIES
C
C                 FREQ  = (I-1)*PI/12     I=1...13
C
C             WHERE PI = 3.14159265
C               AND FREQ = PI IS EQUIVALENT TO THE FOLDING FREQUENCY
C                               FOR THE DATA SERIES
C
C             FT24 GAINS ITS SPEED FROM
C
C               1. STRAIGHT LINE PROGRAMMING RATHER THAN IN LOOPS
C               2. GROUPING TERMS TO MINIMIZE THE NUMBER OF MULTIPLIES
C                     NECESSARY
C               3. SUBGROUPING ADDITIONS TO TAKE ADVANTAGE OF VARIOUS
C                    SYMMETRIES.
C               4. SELECTION OF THE NUMBER OF FREQUENCIES SO AS TO
C                    MAXIMIZE THE NUMBER OF SYMMETRIES GENERATED
C
C             THE EQUATIONS USED WERE DEVELOPED IN SCIENTIFIC REPORT
C             NO. 1 OF AIR FORCE CONTRACT AF 19(604)7378, APPENDIX J.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - IBM 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 818 REGISTERS
C SPEED     - ABOUT 4100 MACHINE CYCLES.
C AUTHOR    - R.A. WIGGINS  JUNE, 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL FT24  (DD,AA,BB)
C
C INPUTS
C    DD(I)      I=1...24 IS THE DATA VECTOR THE TRANSFORM IS TO BE
C                 MADE OF.
C OUTPUTS
C    AA(I)      I=1...13 IS THE COSINE TRANSFORM
C                 AA(1) = (1/24) * (SUM (FROM I=1 TO 24) OF D(I))
C                 AA(13)= (1/24) * (SUM (FROM I=1 TO 24) OF
C                                            D(I)*COS((I-1)*PI))
C                 AA(J) = (1/12) * (SUM (FROM I=1 TO 24) OF
C                                            D(I)*COS((J-1)*(I-1)*PI))
C                      FOR J = 2,3,...,12
C
C    BB(I)      I=1...13 IS THE SINE TRANSFORM
C                 BB(1) = BB(13) = 0.0
C                 BB(J) = (1/12) * (SUM (FROM I=1 TO 24) OF
C                                            D(I)*SIN((J-1)*(I-1)*PI))
C                      FOR J = 2,3,...,12
C
C EXAMPLES
C
C 1. INPUTS  - DD(1...24) = 12.,12.,0.,0.,...,0.
C
C    OUTPUTS - AA(1...13) = 1.,1.966,1.866,1.707,1.500,1.259,1.000,
C                           0.741,0.500,0.293,0.134,0.034,0.000
C              BB(1...13) = 0.,0.259,0.500,0.707,0.866,0.966,1.000
C                           0.966,0.866,0.707,0.500,0.259,0.000
C
C PROGRAM FOLLOWS BELOW
C
```

****
*   FT24 -II   *
****
(PAGE 2)

```
      DIMENSION DD(24),AA(13),BB(13),X(24),A(13),B(13)
      EQUIVALENCE (X(1), X0),(X(2),X1),(X(3),X2),(X(4),X3),(X(5),X4)
      EQUIVALENCE (X(6),X5),(X(7),X6),(X(8),X7),(X(9),X8),(X(10),X9)
      EQUIVALENCE (X(11),X10),(X(12),X11),(X(13),X12),(X(14),X13)
      EQUIVALENCE (X(15),X14),(X(16),X15),(X(17),X16),(X(18),X17)
      EQUIVALENCE (X(19),X18),(X(20),X19),(X(21),X20),(X(22),X21)
      EQUIVALENCE (X(23),X22),(X(24),X23)
      EQUIVALENCE (A(1),A0),(A(2),A1),(A(3),A2),(A(4),A3),(A(5),A4)
      EQUIVALENCE (A(6),A5),(A(7),A6),(A(8),A7),(A(9),A8),(A(10),A9)
      EQUIVALENCE (A(11),A10),(A(12),A11),(A(13),A12)
      EQUIVALENCE (B(1),B0),(B(2),B1),(B(3),B2),(B(4),B3),(B(5),B4)
      EQUIVALENCE (B(6),B5),(B(7),B6),(B(8),B7),(B(9),B8),(B(10),B9)
      EQUIVALENCE (B(11),B10),(B(12),B11),(B(13),B12)
      EQUIVALENCE (C1,S5),(C2,S4),(C3,S3),(C5,S1)
B     S1=177411017560
B     S3=200552023632
B     S4=200673317272
B     S5=200756433522
      A0=0.
      DO 10 I=1,24
      X(I)=DD(I)/12.
   10 A0=A0+X(I)
      A0=A0/2.
      A12=0.
      DO 20 I=1,24,2
   20 A12=A12+X(I)-X(I+1)
      A12=A12/2.
      A1=(X0-X12)+(X1-X11-X13+X23)*C1+(X2-X10-X14+X22)*C2+
     1      (X3-X9-X15+X21)*C3+(X4-X8-X16+X20)*.5+(X5-X7-X17+X19)*C5
      B1=(X6-X18)+(X1+X11-X13-X23)*S1+(X2+X10-X14-X22)*.5+
     1      (X3+X9-X15-X21)*S3+(X4+X8-X16-X20)*S4+(X5+X7-X17-X19)*S5
      A2=(X0-X6+X12-X18)+(X1-X5-X7+X11+X13-X17-X19+X23)*C2+
     1      (X2-X4-X8+X10+X14-X16-X20+X22)*.5
      B2=(X3-X9+X15-X21)+(X1+X5-X7-X11+X13+X17-X19-X23)*.5+
     1      (X2+X4-X8-X10+X14+X16-X20-X22)*S4
      A3=(X0-X4+X8-X12+X16-X20)+(X1-X3-X5+X7+X9-X11-X13+X15+X17-X19-X21+
     1     X23)*C3
      B3=(X2-X6+X10-X14+X18-X22)+(X1+X3-X5-X7+X9+X11-X13-X15+X17+X19-X21
     1     -X23)*S3
      A4=(X0-X3+X6-X9+X12-X15+X18-X21)+(X1-X2-X4+X5+X7-X8-X10+X11+X13-
     1     X14-X16+X17+X19-X20-X22+X23)*.5
      B4=(X1+X2-X4-X5+X7+X8-X10-X11+X13+X14-X16-X17+X19+X20-X22-X23)*S4
      A5=(X0-X12)+(X5-X7-X17+X19)*C1+(-X2+X10+X14-X22)*C2+
     1     (-X3+X9+X15-X21)*C3+(X4-X8-X16+X20)*.5+(X1-X11-X13+X23)*C5
      B5=(X6-X18)+(X5+X7-X17-X19)*S1+(X2+X10-X14-X22)*.5+
     1     (-X3-X9+X15+X21)*S3+(-X4-X8+X16+X20)*S4+(X1+X11-X13-X23)*S5
      A6=(X0-X2+X4-X6+X8-X10+X12-X14+X16-X18+X20-X22)
      B6=(X1-X3+X5-X7+X9-X11+X13-X15+X17-X19+X21-X23)
      A7=(X0-X12)+(-X5+X7+X17-X19)*C1+(-X2+X10+X14-X22)*C2+
     1      (X3-X9-X15+X21)*C3+(X4-X8-X16+X20)*.5+(-X1+X11+X13-X23)*C5
      B7=(-X6+X18)+(X5+X7-X17-X19)*S1+(-X2-X10+X14+X22)*.5+
     1     (-X3-X9+X15+X21)*S3+(X4+X8-X16-X20)*S4+(X1+X11-X13-X23)*S5
      A8=(X0+X3+X6+X9+X12+X15+X18+X21)-(X1+X2+X4+X5+X7+X8+X10+X11+X13+
     1     X14+X16+X17+X19+X20+X22+X23)*.5
      B8=(X1-X2+X4-X5+X7-X8+X10-X11+X13-X14+X16-X17+X19-X20+X22-X23)*S4
      A9=(X0-X4+X8-X12+X16-X20)+(-X1+X3+X5-X7-X9+X11+X13-X15-X17+X19+
     1      X21-X23)*C3
      B9=(-X2+X6-X10+X14-X18+X22)+(X1+X3-X5-X7+X9+X11-X13-X15+X17+X19-
     1      X21-X23)*S3
      A10=(X0-X6+X12-X18)+(-X1+X5+X7-X11-X13+X17+X19-X23)*C2+
     1      (X2-X4-X8+X10+X14-X16-X20+X22)*.5
      B10 =(X3-X9+X15-X21)+(X1+X5-X7-X11+X13+X17-X19-X23)*.5+
     1      (-X2-X4+X8+X10-X14-X16+X20+X22)*S4
      A11=(X0-X12)+(-X1+X11+X13-X23)*C1+(X2-X10-X14+X22)*C2+
     1      (-X3+X9+X15-X21)*C3+(X4-X8-X16+X20)*.5+(-X5+X7+X17-X19)*C5
      B11=(-X6+X18)+(X1+X11-X13-X23)*S1+(-X2-X10+X14+X22)*.5+
     1      (X3+X9-X15-X21)*S3+(-X4-X8+X16+X20)*S4+(X5+X7-X17-X19)*S5
      DO 30 I=1,13
      AA(I)=A(I)
   30 BB(I)=B(I)
      RETURN
      END
```

```
*      FXDATA (SUBROUTINE)              10/1/64   LAST CARD IN DECK IS NO. 0247
*      FAP
*FXDATA
       COUNT   230
       LBL     FXDATA
       ENTRY   FXDATA (LX,X,MXDATA,SCALE)
       ENTRY   FLDATA (LX,X,SCALE)
*
*                    ----ABSTRACT----
*
* TITLE - FXDATA WITH SECONDARY ENTRY FLDATA
*       SCALE, CONVERT FLTG. VECTOR TO MACHINE INTEGERS OR CONVERSELY
*
*             FXDATA CONVERTS A FLOATING POINT VECTOR X(I) I=1...LX
*             TO A MACHINE LANGUAGE INTEGER VECTOR (WITH BINARY POINT
*             TO RIGHT OF BIT 35) IX(I) I=1...LX , SUCH THAT THE
*             GREATEST MAGNITUDE OF IX = MXDATA (AN INPUT PARAMETER).
*             ROUNDING RATHER THAN TRUNCATION OCCURS IN THE CONVERSION.
*             THE OUTPUT INTEGERS ARE NECESSARILY LESS THAN 2EXP17
*             IN MAGNITUDE SINCE MXDATA IS A FORTRAN INTEGER.
*
*             FLDATA PERFORMS THE INVERSE OF FXDATA. IT WILL HANDLE
*             INTEGERS UP TO 2EXP35 - 1, HOWEVER.
*
* LANGUAGE   - FAP SUBROUTINE (WITH FORTRAN II TYPE CALLING SEQUENCE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 102 REGISTERS
* SPEED      - FXDATA TAKES ABOUT 38*LX MACHINE CYCLES
*              FLDATA TAKES ABOUT 38*LX MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON
*
*                    ----USAGE OF FXDATA----
*
* TRANSFER VECTOR CONTAINS ROUTINES -    NONE
*       AND FORTRAN SYSTEM ROUTINES -    NONE
*
* FORTRAN USAGE OF FXDATA
*     CALL FXDATA(LX,X,MXDATA,SCALE)
*
* INPUTS TO FXDATA
*
*    LX        IS LENGTH OF X SERIES
*              IS A FORTRAN INTEGER WHICH MUST EXCEED ZERO
*
*    X(I)      I=1,2,...,LX IS A FLOATING POINT VECTOR
*
*    MXDATA    IS DESIRED MAXIMUM MAGNITUDE OF FIXED SERIES.
*              IS A FORTRAN INTEGER WHICH MUST EXCEED ZERO
*
* OUTPUTS FROM FXDATA
*
*     X(I)     I=1,2,...,LX CONTAINS THE MACHINE LANGUAGE INTEGER
*                 VERSION OF THE INPUT SERIES, DEFINED BY
*                        X(I) = XFIXF(X(I)*SCALE)
*                  WHERE
*                        SCALE = FLOATF(MXDATA)/XMAX
*                         XMAX = GREATEST MAGNITUDE OF ORIGINAL X(I)
*                   AND THE FUNCTION XFIXF(Y) IS EQUIVALENT TO
*                        1. ROUND Y TO THE NEAREST FORTRAN INTEGER
*                        2. SHIFT Y RIGHT ARITHMETICALLY 18 PLACES
*              X(I) IS LEFT=0.0 IF XMAX IS FOUND = 0.0
*
*    SCALE     = FLOATF(MXDATA)/XMAX NORMALLY
*              = -1.0 IF LX OR MXDATA IS ILLEGAL (X(I) LEFT AS IS)
*              = -2.0 IF XMAX IS FOUND = 0.0
*
* FORTRAN USAGE OF FLDATA
*     CALL FLDATA(LX,X,SCALE)
*
* INPUTS TO FLDATA
*
*    LX        IS LENGTH OF X SERIES
*              IS A FORTRAN INTEGER WHICH MUST EXCEED ZERO
*
*    X(I)      I=1...LX IS A SERIES CONSIDERED TO BE 35-BIT INTEGERS
```

```
*                    PLUS SIGN (BINARY POINT TO RIGHT OF BIT 35)
*
*    SCALE      IS A FLOATING POINT SCALE FACTOR USED IN FLOATING X(I)
*               MUST EXCEED 0.0
*
* OUTPUTS FROM FLDATA
*
*    X(I)       I=1...LX IS THE FLOATED, SCALED FORM OF THE INPUT X(I),
*                          X(I) = FLOATF(X(I))/SCALE
*                  WHERE
*                     FLOATF( ) IS AN OPERATION WHICH CONVERTS ANY
*                     36-BIT CONFIGURATION (CONSIDERED AS A 35-BIT
*                     PLUS SIGN INTEGER) TO A FLOATING POINT NUMBER
*               HOWEVER X(I) IS LEFT UNDISTURBED IF EITHER
*                        1. LX IS ZERO OR NEGATIVE
*                   OR 2. SCALE IS ZERO OR NEGATIVE
*
* EXAMPLES OF FXDATA
*
* 1. INPUTS  - LX=5  X(1...5)= 230.,-400.,57.,-170.,99.8   MXDATA=10
*    OUTPUTS - X(1...5) = OCT 000000000006,400000000012,000000000001,
*                  400000000004,000000000002     SCALE = 0.0250
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX=3  MXDATA=100
*    OUTPUTS - X(1...3) = OCT 000000000072,400000000144,000000000016
*              SCALE = 0.250
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX = 1
*    OUTPUTS - X(1) = OCT 000000000012   SCALE = 0.04347826
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT X(1...5)= 0.,0.,...
*    OUTPUTS - X(1...5)= 0.,0.,...   SCALE = -2.0
*
* 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT MXDATA = -2
*    OUTPUTS - X(1...5) = SAME AS INPUT   SCALE = -1.0
*
* 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX = 0
*    OUTPUTS - SAME AS EXAMPLE 5.
*
* EXAMPLES OF FLDATA  (THE FIRST 4 BELOW ARE THE INVERSES OF THE FIRST
*                      FOUR EXAMPLES OF FXDATA)
*
* 1. INPUTS  - LX=5  X(1...5) = OCT 000000000006,400000000012,
*                     000000000001,400000000004,000000000002  SCALE=.025
*    OUTPUTS - X(1...5) = 240.,-400.,40.,-160.,80.
*
* 2. INPUTS  - LX=3  X(1...3) = OCT 000000000072,400000000144,
*                        000000000016   SCALE = 0.250
*    OUTPUTS - X(1...3) = 232.,-400.,56.
*
* 3. INPUTS  - LX=1  X(1) = OCT 000000000012   SCALE = 0.04347826
*    OUTPUTS - X(1) = 230.
*
* 4. INPUTS  - LX=5  X(1...5) = OCT 000000000000,...   SCALE = -2.0
*    OUTPUTS - X(1...5) = 0.0,...
*
* 5. INPUTS  - LX= -3  X(1...3) = 1.,2.,3.      SCALE = 3.4
*    OUTPUTS - X(1...3) = 1.,2.,3.
*
* 6. INPUTS  - LX=4  X(1...4)  = OCT 377777777777,001000000000,
*                     112402762000,007346545000       SCALE = 1.0
*                  (IE X = 2EXP35-1,2EXP27,10EXP10,10EXP9)
*    OUTPUTS - X(1...4) = 34359738367.0,134217728.0,10000000000.0,
*                        1000000000.0
*
       HTR     0
       HTR     0
       BCI     1,FXDATA
FXDATA SXD     FXDATA-3,1
       SXD     FXDATA-2,4
       CLA     2,4
       ADD     K1
       STA     F1
       STA     F4
       STA     F7
```

FXDATA
(PAGE 3)

```
* GET N, CHECK IT, AND CHECK MXDATA.
       CLS     KF1
       STO     SCALE
       CLA*    1,4           SET N
       STO     N             IN IR1
       TZE     F7A
       TMI     F7A
       CLA*    3,4
       TMI     F7A
       TZE     F7A
       LXD     N,1
       STZ     TMAX          STORE TRIAL MAX
 F     CLA     TMAX          FIND
 F1    SBM     **,1          MAXIMUM
       TPL     F3            OF
 F2    CAL*    F1            ALL
       STO     TMAX          ABSOLUTE
 F3    TIX     F,1,1         VALUES
* CHECK FOR CASE ALL X(I)=0
       CLS     KF2
       STO     SCALE
       CLA     TMAX
       TZE     F7A
       CLA*    3,4           FLOAT MXDATA
       ARS     18
       ORA     ORF
       FAD     ORF
       FDP     TMAX
       XCA
*                           FIXED DATA=SCALE*FLOATING DATA
       STO     SCALE         SCALE=MAX OF FIXED DATA/MAX OF FLOATING DATA
       LXD     N,1           SET TO SCALE N QUANTITIES
  F4   LDQ     **,1
       FMP     SCALE         SCALE
       UFA     ORF
       LRS
       ANA     AN
       LLS
       RQL     8
       RND
  F7   STO     **,1
       TIX     F4,1,1
 F7A   CLA     SCALE
       STO*    4,4
       LXD     FXDATA-3,1
       TRA     5,4
 TMAX  PZE
 ORF   OCT     233000000000
 SCALE PZE
 AN    OCT     000000377777
 N     PZE
 K1    PZE     1
 K001  OCT     001000000000
 K266  OCT     266000000000
 TEMP  PZE     **
 TEMP2 PZE     **
 KF1   DEC     1.0
 KF2   DEC     2.0
*      CALL    FLDATA,N,DATA,SCALE
FLDATA SXD     FXDATA-3,1
       SXD     FXDATA-2,4
       CLA*    1,4
       TMI     F14
       TZE     F14
       STO     N
       CLA     2,4
       ADD     K1
       STA     F10
       STA     F11
       CLA*    3,4           GET SCALE
       TMI     F14
       TZE     F14
       STO     SCALE
       LXD     N,1
 F10   CLA     **,1          FLOAT
```

****
*   FXDATA   *
****
(PAGE 4)

```
        LAS     K001
        TRA     Q
        TRA     Q
        ORA     ORF             NUMBERS
        FAD     ORF
        FDP     SCALE
F11     STQ     **,1
        TIX     F10,1,1
F14     LXD     FXDATA-3,1
        TRA     4,4
* HANDLE BIG NUMBERS
Q       LRS     27
        STQ     TEMP
        ORA     K266
        FAD     ORF
        STO     TEMP2
        CLA     TEMP
        ARS     8
        ORA     ORF
        FAD     ORF
        FAD     TEMP2
        TRA     F11-1
        END
```

```
*     GENHOL (SUBROUTINE)              3/15/65   LAST CARD IN DECK IS NO. 0144
*     FAP
*GENHOL
      COUNT   140
      LBL     GENHOL
      ENTRY   GENHOL (HOL)
*
*                   ----ABSTRACT----
*
* TITLE - GENHOL
*       GENERATE HOLLERITH FIELD
*
*             GENHOL GENERATES THE HOLLERITH FIELD THAT WOULD HAVE BEEN
*             PRINTED BY AN IMMEDIATELY SUCCEEDING PRINT STATEMENT.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709, 7090  (MAIN FRAME ONLY)
* STORAGE   - 48 REGISTERS
* SPEED     -
* AUTHOR    - R.A. WIGGINS,  NOV., 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - (IOH)
*
* FORTRAN USAGE
*     CALL GENHOL(HOL)
*     PRINT FMT, LIST
*
* INPUTS
*
*    LIST      IS A LIST OF VARIABLES FOR TRANSMISSION AS DEFINED IN
*                THE FORTRAN REFERENCE MANUAL.
*
*    FMT       IS A STANDARD FORMAT ENTRY TELLING HOW THE LIST IS TO
*                BE TRANSMITTED INTO HOLLERITH.  THE FORMAT MAY IMPLY
*                AN ARBITRARY NUMBER OF LINES OF PRINTED OUTPUT, BUT
*                NONE OF THESE LINES MAY EXCEED 132 CHARACTERS.
*
* OUTPUTS
*
*    HOL(I)    I=1...N IS THE HOLLERITH EQUIVALENT TO THE LINE(S)
*                WHICH WOULD NORMALLY BE PRINTED BY THE PRINT STATEMENT.
*                ACTUAL PRINTING DOES NOT OCCUR.
*
*                LET NLINES = NO. OF LINES IMPLIED BY THE FORMAT
*                    NC(J)  = NO. OF CHARACTERS (INCLUDING SPACES)
*                             IMPLIED BY THE FORMAT FOR THE J-TH LINE
*                    NR(J)  = NO. OF REGISTERS OF HOL(I) WHICH WILL
*                             BE OCCUPIED BY THE CHARACTERS FOR THE
*                             J-TH LINE
*                THEN
*                    NR(J)  = MAXIMUM(3, (NC(J)+5)/6 )
*                    N      = SUM(J=1...NLINES) OF NR(J)
*                    HOL(1...NR(1))  HAS CHARACTERS FOR LINE 1
*                                    (6 PER REGISTER, LEFT ADJUSTED)
*                    HOL(NR(1)+1,...,NR(1)+NR(2))  FOR LINE 2
*                              ETC.
*                ALL SPARE CHARACTER POSITIONS IN HOL(I), IF ANY,
*                WILL BE FILLED WITH BLANKS (OCTAL 60)
*
* EXAMPLES
*
* 1. EXAMPLE OF GENERATION OF HOLLERITH CHARACTERS WITH NO LIST.
*    INPUTS  - FMT(1...7) = 6H(34H HOLLERITH CHARACTERS WITH NO LIST)
*    USAGE   -        CALL GENHOL (HOL)
*                     PRINT FMT
*    OUTPUTS - HOL(1...6) = 6H HOLLERITH CHARACTERS WITH NO LIST
*
* 2. EXAMPLE OF INSERTION OF A NUMBER FROM A LIST.
*    INPUTS  - FMT(1...6) = 6H(25H BOMB SEISMIC RECORD NO. I4)
*              LIST(1) = 42
*    USAGE   -        CALL GENHOL (HOL)
*                     PRINT FMT, LIST(1)
```

GENHOL
(PAGE 2)

```
*    OUTPUTS - HOL(1...5) = 6H BOMB SEISMIC RECORD NO.  42
*
* 3. EXAMPLE OF GENERATION OF A VARIABLE FORMAT STATEMENT.
*    INPUTS  - FMT(1...5) = 6H(1H(I3,3HI7,I3,10HX,2H /2I4))
*                LIST(1) = 5  LIST(2) = 35
*    USAGE   -        CALL GENHOL (HOL)
*                     PRINT FMT, LIST(1), LIST(2)
*    OUTPUTS - HOL(1...4) = 6H(  5I7, 35X,2H /2I4)
*
* 4. EXAMPLE OF REPEATED LINES.  NOTE THAT THE MINIMUM LINE LENGTH IS
*      18 CHARACTERS.
*    INPUTS  - FMT(1...15)= 6H(24H1....6....12....18....24/18H1....6....
*                             12....18/12H1....6....12/6H1....6/1H1//1H1
*                             )
*    USAGE   -        CALL GENHOL (HOL)
*                     PRINT FMT
*    OUTPUTS - HOL(1...22)= 6H1....6....12....18....241....6....12
*                             ....181....6....12      1....6
*                                   1
*                                   1
*
       HTR     0
       BCI     1,GENHOL
GENHOL SXD     *-2,4          GET
       CAL     1,4            POSITION
       ADD     =1B35
       STA     HOL            OF HOL.
       SXA     N,0            RESET N COUNTER.
       CAL     2,4            CHECK FOR STANDARD ERROR PROCEDURE.
       ANA     =0770377000000
       TZE     NOERR
       CAL     SSH            NOT ZERO, STANDARD ERROR PRESENT.
       STA     4,4
       TRA     2,4
 NOERR CAL     SSH            ZERO, NO STANDARD ERROR PROCEDURE.
       STA     2,4
       TRA     2,4
 SSH   PZE     (SSH)          STORAGE TO STORAGE HOLLERITH
*
*               GHO - GENERATE HOLLERITH.
 (SSH) LDQ     *+4
       CLA     *+2
       TRA*    $(IOH)
       MZE     ,,3
       TRA     GHO
*           REENTRY FROM (IOH).
 GHO   SXA     OUT,4          SAVE
       SXA     OUT+1,2        INDEX
       SXA     OUT+2,1        REGISTERS.
       CAL     1,4            GET ADDRESS
       ARS     18
       ADD     1,4
       STA     HOL1           OF RECORD BEGINNING,
       PDX     ,2             AND SAVE
       SXD     A,2
       SXD     N+1,2          LENGTH OF RECORD.
 N     AXT     **,1           SET
       TXI     *+1,1,**       INDEX COUNT
       SXA     N,1            FOR HOL.
       AXT     1,2
 HOL1  CLA     **,2
 HOL   STO     **,1
       TIX     *+1,1,1
       TXI     *+1,2,1
 A     TXL     HOL1,2,**
 OUT   AXT     **,4
       AXT     **,2
       AXT     **,1
       TRA     2,4
       END
```

```
*     GETHOL (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0175
*     LABEL
CGETHOL
      SUBROUTINE GETHOL(JOB,HARG,HOL,NCRS,IXCOM,ICOUNT)
C
C                    ----ABSTRACT----
C
C TITLE - GETHOL
C        GET HOLLERITH DATA FROM CALLING SEQUENCE
C
C              GETHOL ASSUMES ONE OF ITS ARGUMENTS IS HOLLERITH DATA
C              GENERATED IN THE CALLING SEQUENCE (STORED FAP-WISE AND
C              TERMINATED BY AN ALL-ONES FENCE).  THEN, AT THE OPTION
C              OF THE USER, IT EITHER
C                 1.  MOVES THE HOLLERITH TO AN OUTPUT ARGUMENT
C                     REVERSING THE STORAGE ORDER
C               OR
C                 2.  REVERSES THE STORAGE ORDER OF THE HOLLERITH
C                     AT ITS PRESENT LOCATION (THE FENCE IS ALSO MODIFIED
C                     AS A FLAG SO THAT GETHOL WILL NOT RE-REVERSE THE
C                     DATA ON SUBSEQUENT CALLS FOR EITHER OPTION)
C              IN EITHER CASE THE FENCE IS NOT PART OF THE NEW HOLLERITH
C              VECTOR AND GETHOL RETURNS AS OUTPUTS THE NO. OF CHARACTERS
C              IN THE NEW VECTOR  (SIX TIMES VECTOR LENGTH)  AND THE
C              INDEX OF THIS VECTOR WITH RESPECT TO THE FORTRAN
C              COMMON BLOCK.
C
C              FOR OPTION 2. IT ALSO ADDS ONE TO AN OUTPUT COUNTER.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 169 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON,  MARCH 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - REVERS
C        AND FORTRAN SYSTEM ROUTINES - XLOC
C
C FORTRAN USAGE
C       CALL GETHOL(JOB,HARG,HOL,NCRS,IXCOM,ICOUNT)
C
C INPUTS
C
C    JOB        = 0    SPECIFIES THAT USER WANTS OPTION 1. (SEE ABSTRACT)
C               NOT= 0 SPECIFIES THAT USER WANTS OPTION 2. (SEE ABSTRACT)
C
C    HARG(I)    I=1,0,-1,...,-LHOL+2  CONTAINS THE LHOL WORDS OF
C                  HOLLERITH DATA TO BE ACQUIRED
C               I=-LHOL+1  IS THE FENCE  =  OCT 777777777777 (FIRST CALL)
C                  NOTES-
C                      IF GETHOL HAS BEEN CALLED BY THE SAME CALL
C                      STATEMENT PREVIOUSLY WITH JOB NOT= 0, THE
C                      FENCE WILL HAVE BEEN CHANGED TO = OCT 777777777776
C
C                      GETHOL CONSIDERS IT AN ERROR IF ONE OF THESE TWO
C                      TYPES OF FENCES DOES NOT OCCUR WITHIN 106 CELLS
C                      OF HARG(1) (635 IS THE MAX NO. CHARACTERS
C                      EXPRESSIBLE ON 9 CONTINUATION CARDS IN A
C                      CALL GETHOL STATEMENT).
C
C OUTPUTS
C
C    HARG(I)    I=1,0,...-LHOL+1 IS UNCHANGED IF JOB=0, OR IF
C                             FENCE = OCT 777777777776
C                  FOR JOB NOT= 0  AND FENCE = OCT 777777777777
C                    HARG(1,0,...,-LHOL+2) IS REVERSED
C                    HARG(-LHOL+1) = FENCE IS SET = OCT 777777777776
C
C    HOL(I)     I=1...LHOL  IS UNDISTURBED FOR JOB NOT= 0
C                  FOR JOB = 0
C                  HOL(I) = HARG(2-I)    IF FENCE = OCT 777777777777
C                  HOL(I) = HARG(-LHOL+I+1) IF FENCE=OCT 777777777776
C
```

```
C    NCRS      IS THE NO. CHARACTERS OF HOLLERITH DATA = 6*LHOL
C              IS SET = -1 IF LHOL EXCEEDS 106
C                              (NO OTHER OUTPUT IN THIS CASE)
C
C    IXCOM     IS THE INDEX WITH RESPECT TO COMMON OF THE NEW
C                HOLLERITH VECTOR, I.E.  THE INDEX OF HOL(1) OR
C                OF HARG(-LHOL+2) WHICHEVER IS APPROPRIATE.
C
C    ICOUNT    IS NOT USED FOR OPTION 1.
C              IS INCREASED IN VALUE BY 1 FOR OPTION 2.
C
C EXAMPLES
C
C 1. USAGE WITH JOB=0, IGNORING THE IXCOM OUTPUT
C    USAGE   -       DIMENSION HOL(10)
C                    CALL GETHOL(0,18HFIRST,SECOND,THIRD,HOL,NCRS,IXCOM,
C                  1             ICOUNT)
C    OUTPUTS - HOL(1)= 6HFIRST,
C              HOL(2)= 6HSECOND
C              HOL(3)= 6H,THIRD     NCRS=18   ICOUNT IS UNDISTURBED
C
C 2. SIMILAR TO 1. BUT USING THE IXCOM OUTPUT FEATURE
C    USAGE   - DIMENSION  CM(2), HOL(10)
C                    COMMON  CM
C                    CALL GETHOL(0,12HFIRST,SECOND,HOL,NCRS,IXCOM,
C                  1             ICOUNT)
C    OUTPUTS - HOL(1) = CM(IXCOM)    = 6HFIRST,
C              HOL(2) = CM(IXCOM+1) = 6HSECOND   NCRS=12
C
C 3. USAGE WITH JOB NOT= 0
C    INPUTS  - SET ICOUNT=0
C    USAGE   -       CALL GETHOL(1,8H(5X,3I5),DUMMY,NCRS,IXCOM,ICOUNT)
C    CUTPUTS - NCRS=12  CM(IXCOM) = 6H(5X,3I  CM(IXCOM+1) = 2H5)
C              CM(IXCOM-1) = OCT777777777776  (THE NEW FENCE)
C              ICOUNT=1
C
C 4. REPEATED USE OF SAME CALL STATEMENT WITH JOB NOT= 0
C    USAGE   -       DIMENSION CM(2),SPACE(2,4),HOL(2,4),JOB(4),NCRS(4)
C                    COMMON  CM
C                    JOB(1) = 0
C                    JOB(2) = 1
C                    JOB(3) = 0
C                    JOB(4) = 1
C                    ICOUNT=0
C                    DO 10  I=1,4
C                    CALL GETHOL(JOB(I),7H1234567,HOL(1,I),NCRS(I),
C                  1             IXCOM,ICOUNT)
C                    SPACE(1,I) = CM(IXCOM)
C                 10 SPACE(2,I) = CM(IXCOM+1)
C    OUTPUTS - HOL(1,1)=HOL(1,3)=SPACE(1,I) = 6H123456  FOR I=1,2,3,4
C              HOL(2,1)=HOL(2,3)=SPACE(2,I) = 1H7       FOR I=1,2,3,4
C              NCRS(I) =12  FOR I=1,2,3,4    ICOUNT=2
C
C 5. ILLEGAL HOLLERITH DATA
C    INPUTS  - SPACE(1...150) = 6H2 LONG
C    USAGE   -       CALL GETHOL(JOB,SPACE(150),HOL,NCRS,IXCOM,ICOUNT)
C    CUTPUTS - NCRS = -1
C
C PROGRAM FOLLOWS BELOW
C FALSE DIMENSIONS
      DIMENSION ICM(2),CM(2),HOL(2)
      COMMON ICM
      EQUIVALENCE (CM,ICM),(FNCE1,IFNCE1),(FNCE2,IFNCE2)
B     FNCE1=777777777777
B     FNCE2=777777777776
      LOCCOM=XLOCF(CM)
C SEARCH FOR FENCE, SETTING SWITCH FOR TYPE FOUND IF ANY
      IXARG=LOCCOM-XLOCF(HARG)+1
      DO 50  I=1,106
      IXNXT=IXARG-I
      NEXT=ICM(IXNXT)
      IFSWCH=0
      IF(NEXT-IFNCE1)  40,70,40
   40 IFSWCH=1
      IF(NEXT-IFNCE2)  50,70,50
```

GETHOL

(PAGE 3)

```
 50   CONTINUE
C ILLEGAL IF FALLS THRU 50
      NCRS=-1
      GO TO 9999
C OK IF JUMPS HERE. FORK ON JOB
 70   LHL=I
      IF(JOB)  100,80,100
C FOR JOB=0  MOVE DATA, SET IXCOM, THEN GO CHECK REVERSAL
 80   DO 85  I=1,LHL
      IXNXT = IXARG-LHL+I
 85   HOL(I)=CM(IXNXT)
      IXCM=LOCCOM-XLOCF(HOL)+1
      GO TO 110
C FOR JOB NOT=0  SET IXCOM, NEW FENCE, INDEX ICOUNT, THEN CHECK REVERSAL
 100  IXCM=IXARG-LHL+1
      CM(IXCM-1)=FNCE2
      ICOUNT=ICOUNT+1
C CHECK REVERSAL. IF NOT GO EXIT
 110  IF(IFSWCH)  9990,120,9990
C REVERSE
 120  CALL REVERS(LHL,CM(IXCM))
C EXIT SEQUENCE
 9990 NCRS=6*LHL
      IXCOM=IXCM
 9999 RETURN
      END
```

```
*     GETRD1 (SUBROUTINE)                 10/1/64   LAST CARD IN DECK IS NO. 0172
*     LABEL
CGETRD1
      SUBROUTINE GETRD1(ITAPE,NX,IX,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - GETRD1
C       ACCESS ROUTINE FOR RAND CORP. MILLION RANDOM DIGITS FROM TAPE
C
C             GETRD1 FURNISHES THE NEXT NX SEQUENTIAL RANDOM DIGITS
C             AS FIXED POINT INTEGERS FROM A SPECIFIED TAPE UNIT.
C
C             THE TAPE UNIT CONTAINS THE MILLION DIGITS IN BCD FORM
C             AS LOADED OFF-LINE FROM THE 20000 CARDS CONTAINING THEM,
C             EACH CARD WITH FORMAT(50I1).  GETRD1 KEEPS A BUFFER OF
C             LENGTH 50 TO PREVENT MISSING ANY DIGITS, BUT DOES NOT
C             CHECK FOR THE POSSIBILITY THAT THE SUPPLY IS EXHAUSTED.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME PLUS 1 TAPE UNIT)
C STORAGE   - 229 REGISTERS
C SPEED     - SLOW, SINCE TAPE IS BCD
C AUTHOR    - S.M.SIMPSON JR.
C
C                     ----USAGE----
C
C TRANSVER VECTOR CONTAINS ROUTINES - (NONE)
C       AND FORTRAN SYSTEM ROUTINES - (TSH), (RTN)
C
C FORTRAN USAGE
C     CALL GETRD1(ITAPE,NX,IX,IANS)
C
C INPUTS
C
C    ITAPE     IS THE LOGICAL TAPE NO. OF THE RANDOM DIGITS TAPE
C              MUST LIE BETWEEN 1 AND 20 INCLUSIVE
C
C    NX        IS THE DESIRED NO. OF DIGITS
C              MUST EXCEED ZERO
C
C OUTPUTS
C
C    IX(I)     I=1...NX  WILL CONTAIN THE NEXT NX DIGITS AS FORTRAN
C                 FIXED POINT INTEGERS
C
C    IANS      =  0  NORMAL
C              = -1  FOR ILLEGAL ITAPE
C              = -2  FOR ILLEGAL NX
C
C EXAMPLES
C
C 1. ILLUSTRATING EFFECTS OF SUCCESSIVE CALLS
C    INPUTS  - THE FIRST THREE RAND DIGITS CARDS ARE AS FOLLOWS
C
C             C   COLUMN NUMBERS
C             A
C             R   00000000011111111112222222222333333333344444444445
C             D   12345678901234567890123456789012345678901234567890
C
C             1   10097325337652013586346735487680959091173929274945
C             2   37542048056489474296248052403720636104020082291665
C             3   08422689531964509303232090256015953347643508033606
C              ASSUME THE CARDS ARE LOADED ON LOGICAL TAPE 9
C
C    USAGE   -        REWIND 9
C                     CALL GETRD1(9,10,IX1,IANS1)
C                     CALL GETRD1(9,10,IX2,IANS2)
C                     CALL GETRD1(9, 1,IX3,IANS3)
C                     CALL GETRD1(9,29,IX4,IANS4)
C                     CALL GETRD1(9, 1,IX5,IANS5)
C                     CALL GETRD1(9,55,IX6,IANS6)
C                     REWIND 9
C                     CALL GETRD1(9, 3,IX7,IANS7)
C
```

```
C    OUTPUTS - IANS1=IANS2 = ETC = IANS7 = 0    (NO ILLEGALITIES)
C              IX1(1...10) = 1,0,0,9,7,3,2,5,3,3
C              IX2(1...10) = 7,6,5,2,0,1,3,5,8,6
C              IX3(1...1)  = 3
C              IX4(1...29) = 4,6,7,3,5,4,8,7,6,8,0,9,5,9,0,9,1,1,7,3,
C                            9,2,9,2,7,4,9,4,5
C              IX5(1...1)  = 3
C              IX6(1...55) = 7,5,4,2,0,4,8,0,5,6,4,8,9,4,7,4,2,9,6,2,
C                            4,8,0,5,2,4,0,3,7,2,0,6,3,6,1,0,4,0,2,0,
C                            0,8,2,2,9,1,6,6,5,0,8,4,2,2,6
C              IX7(1...3)  = 8,9,5  (NOT = 1,0,0  SINCE GETRD1 STILL
C                                    HAS 44 DIGITS IN ITS BUFFER TO
C                                    USE UP BEFORE READING FROM TAPE
C                                    AGAIN)
C
C 2. ILLUSTRATING ILLEGAL USAGE
C
C    USAGE   -       CALL GETRD1(0,1,IX,IANS1)
C                    CALL GETRD1(21,1,IX,IANS2)
C                    CALL GETRD1(9,-3,IX,IANS3)
C
C    OUTPUTS - IANS1 = IANS2 = -1   (ILLEGAL ITAPE)
C              IANS3 = -2           (ILLEGAL NX)
C
C PROGRAM FOLLOWS BELOW
C
C DUMMY DIMENSION STATEMENT
      DIMENSION IX(2)
C TRUE DIMENSION STATEMENT
      DIMENSION INP(50)
C CHECK LEGALITIES OF ITAPE,NX
      IANS=-1
      IF (ITAPE)  9999,9999,2
    2 IF (ITAPE-20)  4,4,9999
    4 IANS=-2
      IF (NX)  9999,9999,10
   10 IOUT=0
      IANS=0
      MORE=NX
C
C ANY DIGITS LEFT IN BUFFER FROM PREVIOUS CALL (IF NO, GO READ
C 50 DIGITS).
C
      IF (NBUF)  20,40,20
C
C IF YES, CHECK IF REQUEST CAN BE FILLED FROM BUFFER.
C
   20 IF (NX-NBUF)  30,30,24
C
C IT CANT.  EMPTY BUFFER AND THEN GO READ MORE DIGITS.
C
   24 DO 26  I=1,NBUF
   26 IX(I)=INP(I)
      IOUT=NBUF
      MORE=MORE-NBUF
      GO TO 40
C
C IT CAN BE FILLED FROM BUFFER.  SET UP TO DO SO AND EXIT.
C
   30 NBLOK=NBUF
      GO TO 66
C
C READ 50 DIGITS
C
   40 READ INPUT TAPE ITAPE,42,(INP(I),I=1,50)
   42 FORMAT(50I1)
C
C CHECK IF THIS IS LAST BLOCK OF 50 NEEDED.
C
      IF (MORE-50)  60,60,50
C
C NO. MOVE BLOCK OF 50 AND GO BACK FOR ANOTHER.
C
   50 DO 54  I=1,50
      II=I+IOUT
```

****
*   GETRD1   *
****
(PAGE 3)

```
   54 IX(II)=INP(I)
      IOUT=IOUT+50
      MORE=MORE-50
      GO TO 40
C
C YES.  SET FOR FINAL MOVE.
C
   60 NBLOK=50
C
C MOVE FINAL BLOCK AND SET UP BUFFER FOR NEXT CALL
C
   66 DO 68  I=1,MORE
      II=I+IOUT
   68 IX(II)=INP(I)
      NBUF=NBLOK-MORE
      IF (NBUF)  70,9999,70
   70 MRP1=MORE+1
      DO 74  I=MRP1,NBLOK
      II=I-MORE
   74 INP(II)=INP(I)
      GO TO 9999
 9999 RETURN
      END
```

```
*      GETX (FORTRAN FUNCTION)          9/4/64   LAST CARD IN DECK IS NO. 0127
*      FAP
*GETX
       COUNT    100
       LBL      GETX
       ENTRY    GETX    (X,I1,I2,...,IN)
       ENTRY    IGETX  (IX,I1,I2,...,IN)
*
*
*                    ----ABSTRACT----
*
* TITLE - GETX  WITH SECONDARY ENTRY POINT IGETX
*        ALLOWS VARIABLE DEPTH INDEXING OF VECTORS
*
*              GETX IS A VARIABLE LENGTH CALLING SEQUENCE PROGRAM THAT
*              GETS A NUMBER BY MEANS OF A CHAIN OF INDEX VALUES.
*              THUS, THE STATEMENT
*
*                    X1 = GETX (X,I1,I2)
*
*              IS EQUIVALENT TO THE FORTRAN STATEMENTS
*
*                    J1 = I1(I2)
*                    X1 = X(J1)
*
*              IGETX PERFORMS THE SAME FUNCTION AS GETX.
*
* LANGUAGE   - FAP (FORTRAN II FUNCTION)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME AND SENSE INDICATORS ONLY)
* STORAGE    - 31 REGISTERS
* SPEED      - (1+N)*19 MACHINE CYCLES WHERE N=NUMBER OF ARGUMENTS
* AUTHOR     - R.A. WIGGINS,  OCTOBER, 1963
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
*
*
* FORTRAN USAGE OF GETX
*     X1 = GETX (X,I1,I2,...,IN)
*
* INPUTS
*
*    X(I)      I=1,...,LX IS A VECTOR OF VALUES.  LX SHOULD BE GREATER
*                THAN ANY OF THE INDICES IN THE INDEX VECTOR I1.
*
*    I1(I)     I=1,...,LI1 IS A VECTOR OF INDICES.  LI1 SHOULD BE
*                GREATER THAN ANY OF THE INDICES IN THE VECTOR I2.
*
*    I2(I)     IS SIMILAR TO I1.
*
*    .
*    .
*    .
*
*    IN        IS AN INDEX VALUE.
*
* OUTPUTS
*
*    X1        IS NUMBER AS DESCRIBED IN THE ABSTRACT.
*
*
*
* FORTRAN USAGE OF IGETX
*     IX1 = IGETX (IX,I1,I2,...,IN)
*
* INPUTS AND OUTPUTS ARE ANALOGOUS TO GETX BUT FIXED POINT.
*
*
*
* EXAMPLES
*
```

```
* 1. INPUTS  - X(1...5) = 1.,2.,3.,4.,5.   IX(1...5) = 1,2,3,4,5
*              I1=4
*    USAGE   -          X1 =  GETX  (X,I1)
*                       IX1= IGETX (IX,I1)
*    OUTPUTS - X1=4.  IX1=4
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              I1(1...7) = 4,1,1,3,5,2,1
*              I2(1...3) = 1,7,5    I3 = 3
*    USAGE   -          X1 =  GETX  (X,I1,I2,I3)
*                       IX1= IGETX (IX,I1,I2,I3)
*    OUTPUTS - X1=5.  IX1=5
*
*
*
* PROGRAM FOLLOWS BELOW
*
 XR4    HTR      0
 XR1    HTR      0
        BCI      1,GETX
 GETX   BSS      0
 IGETX  SXD      XR4,4
        SXD      XR1,1
        STI      IND
        SXD      GETE,4
* FIND LAST ARGUMENT
 FIAT   LDI      1,4
        RIS      MASK1
        OFT      CTSXZ
 SWCH   TXI      OUT,4,1
        TIX      FIAT,4,1
* GET NUMBER
 OUT    AXT      0,1
        SXD      XR41,4
 GET    CLA      1,4
        STA      *+1
        CLA      **,1
        PDX      ,1
        TIX      *+2,1,1
        AXT      0,1
        TXI      *+1,4,1
 GETE   TXL      GET,4,**       ** CONTAINS INITIAL XR4
* LEAVE
        LDI      IND
        LXD      XR1,1
        LXD      XR41,4
        TRA      2,4
 IND    PZE
 MASK1  OCT      000000077777
 CTSXZ  OCT      770377700000
 MASK2  OCT      000000177777
 XR41   PZE
        END
```

```
*     GNFLT1 (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0163
*     LABEL
CGNFLT1
      SUBROUTINE GNFLT1(AMSPEC,LSPEC,FLTR,IANS)
C
C                   ----ABSTRACT----
C
C TITLE - GNFLT1
C       GENERATE SYMMETRICAL FILTER WITH GIVEN AMPLITUDE RESPONSE
C
C             GNFLT1 GENERATES A SYMMETRICAL (TWO-SIDED) SET OF FILTER
C             COEFFICIENTS WHOSE AMPLITUDE SPECTRUM APPROXIMATES A
C             GIVEN AMPLITUDE SPECTRUM AT EQUALLY SPACED POINTS BETWEEN
C             ZERO AND PI (RADIANS).  IF THE DESIRED AMPLITUDE SPECTRUM
C             IS AMP(I) I=0,1,....,M , THEN THE FILTER COEFFICIENTS,
C             FILTER(I) I=-M,-M+1,....,M , ARE GENERATED BY A WEIGHTED
C             ADDITION OF A SMOOTHED ORTHONORMAL SET OF OPERATORS
C             ACCORDING TO
C                                    M
C                   FILTER(S) =  SUM ( AMP(P)*ORTNRM(S,P,M) )
C                                   P=0
C             FOR S = -M,...,M
C             WHERE
C                ORTNRM(S,P,M) = NRM(P,M)*ORT(S,P,M)
C                     NRM(P,M) = 1/M    FOR P = 1,2,...,M-1
C                     NRM(P,M) = 1/2M   FOR P = 0  AND P = M
C             AND
C              ORT(S,P,M) = C(S)*((.54+.46*COS(S*PI/M))*COS(S*P*PI/M))
C                    C(S) = 0.5  FOR S = M  AND S = -M
C                    C(S) = 1.0  OTHERWISE
C                    PI   = 3.14159265
C
C             THE ORT(S,P,M) SET IS A SCALED VERSION OF THE ORTHONORMAL
C             SET GIVEN BY TUKEY AND HAMMING (1949, MEASURING NOISE
C             COLOR, BELL TEL. LAB. MEMO - MM-49-110-119.)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 232 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON JR.
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
C       AND FORTRAN SYSTEM ROUTINES - COS
C
C FORTRAN USAGE
C     CALL GNFLT1(AMSPEC,LSPEC,FLTR,IANS)
C
C INPUTS
C
C    AMSPEC(I) I=1,...,LSPEC CONTAINS THE DESIRED AMPLITUDE RESPONSE
C                  AMP(J) J=0,1,...,M   (M=LSPEC-1) , I.E.
C                  AMSPEC(I) = AMP(I-1) = RESPONSE AT (I-1)*PI/M RADIANS
C              AMSPEC(I) MUST NOT VANISH FOR ALL I VALUES
C
C    LSPEC     MUST EXCEED 2 AND BE LESS THAN OR = 1001
C
C    OUTPUTS
C
C    FLTR(I)   I=1,2,...,(2*M+1) ARE THE DESIRED FILTER COEFFICIENTS,
C                  FILTER(J) J= -M,-M+1,...,M , AS DEFINED IN ABSTRACT,
C                  I.E.  FLTR(I) = FILTER(I-M-1)
C
C    IANS       =  0 NORMALLY
C               = -1 FOR ILLEGAL AMSPEC (ALL ZERO)
C               = -2 FOR ILLEGAL LSPEC
C
C EXAMPLES
C
C 1. A NARROW LOW-PASS AND NARROW BAND-PASS FILTER
C    INPUTS  - A1(1...21) = 1.,0.,0.,...,0.,      L1 = 21
C              A2(1...21) = 0.,0.,1.,0.,...0.     L2 = 21
C    USAGE   -        CALL GNFLT1(A1,L1,FLTR1,IANS1)
```

```
C                       CALL GNFLT1(A2,L2,FLTR2,IANS2)
C     OUTPUTS - IANS1 = IANS2 =0
C               FLTR1(1...41) =
C                 .00100  .00214  .00257  .00326  .00420
C                 .00537  .00674  .00828  .00995  .01170
C                 .01350  .01530  .01706  .01872  .02026
C                 .02163  .02281  .02375  .02444  .02486
C                 .02500  .02486  .02444  .02375  .02281
C                 .02163  .02026  .01872  .01706  .01530
C                 .01350  .01170  .00995  .00828  .00674
C                 .00537  .00420  .00326  .00257  .00214  .00100
C               FLTR2(1...41) =
C                 .00200  .00408  .00415  .00383  .00260
C                 .00000 -.00417 -.00974 -.01610 -.02226
C                -.02700 -.02910 -.02760 -.02201 -.01252
C                 .00000  .01410  .02792  .03954  .04729
C                 .05000  .04729  .03954  .02792  .01410
C                 .00000 -.01252 -.02201 -.02760 -.02910
C                -.02700 -.02226 -.01610 -.00974 -.00417
C                 .00000  .00260  .00383  .00415  .00408  .00200
C
C 2. TEST CASE FOR WHITE LIGHT FILTER (FILTER SHOULD BE AN IMPULSE)
C     INPUTS  - A(1...11) = 1.,1.,...,1.    L = 11
C     USAGE   -         CALL GNFLT1(A,L,FLTR,IANS)
C     OUTPUTS - FLTR(1...21) = 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,...,0.
C
C 3. ILLEGAL CONDITIONS
C     INPUTS  - A(1...5) , 0.,0.,0.,0.,0.   B(1...5) = 1.,1.,1.,1.,1.
C     USAGE   -         CALL GNFLT1(A,5,FLTR,IANS1)
C                       CALL GNFLT1(B,2,FLTR,IANS2)
C                       CALL GNFLT1(B,1005,FLTR,IANS3)
C     OUTPUTS - IANS = -1  (ILLEGAL AMSPEC, ALL ZERO)
C               IANS2 = IANS3 = -2  (ILLEGAL LSPEC)
C
      DIMENSION AMSPEC(100),FLTR(2001)
C CHECK LSPEC,AMSPEC
      IANS=-2
      IF(LSPEC-3)  9999,10,10
 10   IF(LSPEC-1001)  20,20,9999
 20   IANS=-1
      DO 30  I=1,LSPEC
      IF(AMSPEC(I))  50,30,50
 30   CONTINUE
C ILLEGAL AMSPEC IF FALLS THRU 30
      GO TO 9999
C INPUTS OK, INITIALIZE LOOP WHICH FORMS FILTER (0,1,...M)
C IN FLTR(M+1,...,LSPEC)
 50   IANS=0
      M=LSPEC-1
      FM=FLOATF(M)
      PIOVM=3.14159265/FM
      IXS=0
C ENTER LOOP ON S=0,1,...M*******************************************
 100  FIXS=FLOATF(IXS)
      C=1.0
      IF (IXS-M)  115,110,110
 110  C=0.5
 115  ARG1=FIXS*PIOVM
      COS1=COSF(ARG1)
      SUM=0.0
      IXP=0
C ENTER LOOP ON IXP=0,1,...M*****************************************
 130  FIXP=FLOATF(IXP)
      FNRM=1.0/FM
      IF(IXP)  140,150,140
 140  IF(IXP-M)  160,150,160
 150  FNRM=0.5/FM
 160  ARG2=ARG1*FIXP
      ORT=C*((.54+.46*COS1)*COSF(ARG2))
C FIND AMSPEC AND BUMP SUM
      I=IXP+1
      AMP=AMSPEC(I)
      SUM = SUM + AMP*FNRM*ORT
C INDEX ON IXP AND CHECK FOR MORE
      IXP=IXP+1
```

```
      IF(IXP-M)  130,130,200
C STORE FILTER(S) AND FILTER(-5)
 200  I=IXS+M+1
      J=-IXS+M+1
      FLTR(I)=SUM
      FLTR(J)=SUM
C INDEX ON IXS AND CHECK FOR MORE
      IXS=IXS+1
      IF(IXS-M)  100,100,300
C ALL DONE
 300  GO TO 9999
C EXIT
 9999 RETURN
      END
```

```
*      GNHOL2  (SUBROUTINE)            9/29/64   LAST CARD IN DECK IS NO. 0157
*      FAP
*GNHOL2
       COUNT   100
       LBL     GNHOL2
       ENTRY   GNHOL2  (DATA,NDATA,FMT,HOL,NCRS,IXCOM,INDEX)
*
*                    ----ABSTRACT----
*
* TITLE - GNHOL2
*       GENERATE HOLLERITH CHARACTERS
*
*             GNHOL2 GENERATES HOLLERITH CHARACTERS FROM DATA AND
*             FORMAT INFORMATION IN THE CALLING SEQUENCE.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090
* STORAGE   - 74 REGISTERS
* SPEED     -
* AUTHOR    - R.A. WIGGINS     3/63
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - (IOH),(FIL)
*
* FORTRAN USAGE
*     CALL GNHOL2(DATA,NDATA,FMT,HOL,NCRS,IXCOM,INDEX)
*
* INPUTS
*
*    DATA(I)   I=1,...,NDATA CONTAINS FLOATING POINT NUMBERS, FIXED
*                POINT OR MACHINE LANGUAGE INTEGERS, OR ALPHANUMERIC
*                CHARACTERS WHICH ARE TO BE INSERTED IN THE HOLLERITH
*                OUTPUT ACCORDING TO THE FORMAT FMT.
*
*    NDATA     MUST BE GRTHN=0
*
*    FMT(I)    I=M,...,1 (M ARBITRARY) CONTAINS A FORMAT STATEMENT
*                WHICH IS TO BE INTERPRETED TO GENERATE THE HOLLERITH.
*              FMT IS STORED IN REVERSE ORDER. IE FMT(M) CONTAINS THE
*                FIRST WORD (AND IS THE ARGUMENT GIVEN GNHOL2), FMT(M-1)
*                CONTAINS THE SECOND WORD, ETC.
*              IS MOST EASILY GENERATED BY A HOLLERITH FIELD INSIDE
*                THE CALL STATEMENT (SEE EXAMPLES).
*
*    INDEX     IS ANY FIXED POINT INTEGER.
*
* OUTPUTS
*
*    HOL(I)    I=1,...,NCRS/6 CONTAINS THE HOLLERITH CHARACTERS (6 PER
*                WORD, IN FORTRAN ORDER) THAT IS GENERATED FROM THE
*                FORMAT AND DATA VECTORS.
*
*    NCRS      IS 6 TIMES THE NUMBER OF WORDS IN HOL.
*
*    IXCOM     IS THE INDEX, WITH RESPECT TO COMMON OF THE FIRST WORD
*                OF HOL.
*
*    INDEX     IS INCREASES BY ONE FROM THE INPUT VALUE.
*
* EXAMPLES
*
* 1. GENERATION OF HOLLERITH CHARACTERS WITH NO DATA.
*    INPUTS  - NDATA=0  INDEX=4
*    USAGE   -        COMMON HOL
*                     CALL GNHOL2 (DATA,NDATA,21H(16HSAMPLE HOLLERITH),
*                    1    HOL,NCRS,IXCOM,INDEX)
*    OUTPUTS - HOL(1...3) = 6HSAMPLE HOLLERITH  NCRS=18  IXCOM=1 INDEX=5
*
* 2. GENERATE HOLLERITH WITH DATA
*    INPUTS  - DATA(1)=5.  NDATA=1  INDEX=5
*    USAGE   -        COMMON HOL
*                     CALL GNHOL2 (DATA,NDATA,23H(14H ERROR FLAG = F4.1),
*                    1    HOL,NCRS,IXCOM,INDEX)
```

```
     *     OUTPUTS - HOL(1...3) = 6H ERROR FLAG = 5.0
     *               NCRS = 18  IXCOM = 1  INDEX = 6
     *
     * 3. GENERATE HOLLERITH FROM A FORMAT DEFINED OUTSIDE THE CALL STATEMENT
     *     INPUTS  - NDATA = 0  INDEX = 6  FMT(4) = 6H(16HSA
     *                                     FMT(3) = 6HMPLE H
     *                                     FMT(2) = 6HOLLERI
     *                                     FMT(1) = 3HTH)
     *     USAGE   -        COMMON HOL
     *                      CALL GNHOL2 (DATA,NDATA,FMT(4),HOL,NCRS,IXCOM,
     *                     1      INDEX)
     *     OUTPUTS - HOL(1...3) = 6HSAMPLE HOLLERITH  NCRS=18 IXCOM=1 INDEX=7
     *
             HTR     0
             BCI     1,GENHOL
     GNHOL2  SXD     *-2,4              SAVE
             SXA     EX,1               INDEX
             SXA     EX+1,2             REGISTERS.
             CAL     1,4                GET
             ADD     =1B35              ADDRESS OF
             STA     DATA               DATA.
             CAL*    2,4                GET NUMBER
             STD     NDATA              OF DATA WORDS.
             CAL     3,4                GET POSITION
             STA     FMT                OF FORMAT.
             CAL     4,4                GET POSITION
             ADD     =1B35              OF
             STA     HOL                HOL.
             ALS     18                 SET
             SUB     =32563B17          OUTPUT
             STD*    6,4                OF IXCOM.
             SXA     N,0                RESET N COUNTER.
             AXC     FMT-1,4            SET IR 4 FOR DUMMY PRINT.
      (SSH)  CLA     =4B17              DUMMY UNIT DESIGNATION
             LDQ     *+2
             TRA*    $(IOH)             *INITIALIZE (IOH)
             TRA     SSH                OUTPUT / STORAGE TO STORAGE HOLLERITH
     *                               REENTRY FROM (IOH)
      SSH    SXA     OUT,4              SAVE
             SXA     OUT+1,2            INDEX
             SXA     OUT+2,1            REGISTERS.
             CAL     1,4                GET
             ARS     18                 ADDRESS
             ADD     1,4                OF BEGINNING
             STA     HOL1               OF RECORD.
             PDX     ,2                 SAVE
             SXD     A,2                LENGTH
             SXD     N+1,2              OF RECORD.
      N      AXT     **,1               INCREMENT
             TXI     *+1,1,**           THE LENGTH
             SXA     N,1                OF HOL.
             AXT     1,2                MOVE
      HOL1   CLA     **,2               HOLLERITH
      HOL    STO     **,1               FROM (IOH)
             TIX     *+1,1,1            BUFFER TO
             TXI     *+1,2,1            HOL.
      A      TXL     HOL1,2,**
      OUT    AXT     **,4               RESTORE
             AXT     **,2               INDICES
             AXT     **,1               AND
             TRA     2,4                * RETURN TO (IOH).
     *                               DUMMY PRINT
      FMT    PZE     **,,1              FORMAT DESIGNATION.
             AXT     1,1                INDEXING.
      NDATA  TXL     *+2,1,**
             TRA     C
      DATA   LDQ     **,1               OUTPUT
             STR                        LIST.
             TXI     NDATA,1,1          INDEXING
      C      TSX     $(FIL),4           * RETURN TO (IOH).
             LXD     GNHOL2-2,4         FINAL ENTRY FROM (IOH).
             LXA     N,2                GET
             PXA     ,2                 NCRS
             XCA
             MPY     =6B17
```

```
           STQ*    5,4             FOR OUTPUT.
           CLA*    7,4             INCREMENT
           ADD     =1B17           INDEX
           STO*    7,4             BY ONE.
     EX    AXT     **,1            EXIT
           AXT     **,2
           TRA     8,4
           END
```

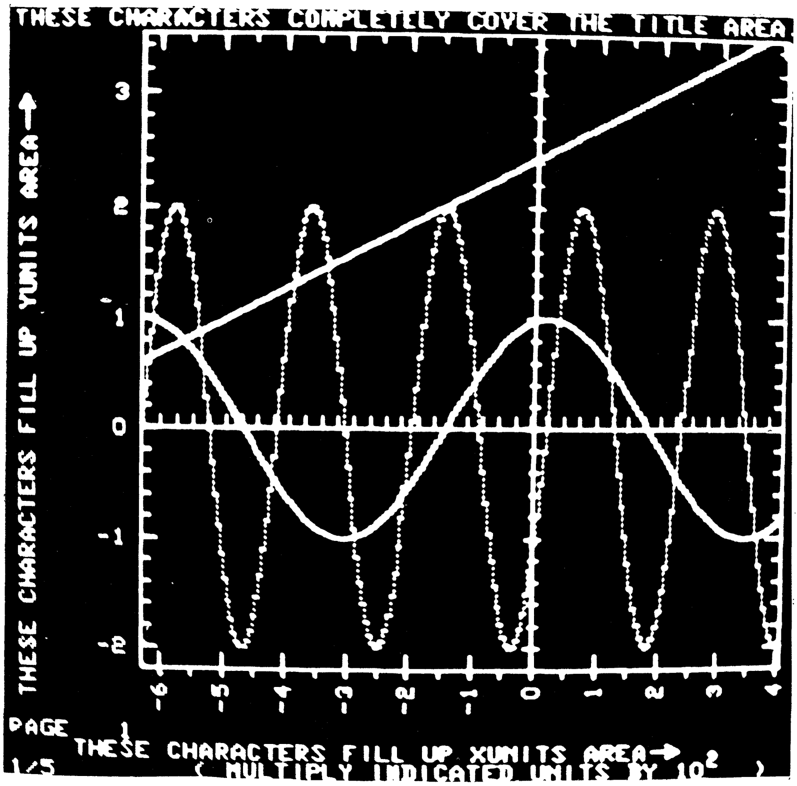
```
*     GRAPH (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 1102
*     LABEL
CGRAPH
      SUBROUTINE GRAPH(ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,YBOT,
     1                XMAX,XMIN,NOPPP,IPAGE,SPACE)
C
C                    ----ABSTRACT----
C
C TITLE - GRAPH
C       MULTIPLE FRAME SCOPE PLOTS OF VECTOR SETS
C
C             GRAPH MAKES A SIMULTANEOUS PLOT OF AN ARBITRARY NUMBER
C             OF SERIES ACROSS AS MANY SCOPE FRAMES AS NEEDED. RESULTING
C             PHOTOS CAN BE ABUTTED TO GIVE CONTINUOUS GRAPH.
C
C             USER SUPPLIES HOLLERITH LABELS, SCALING AND PLOTTING
C             PARAMETERS, FROM WHICH GRAPH DETERMINES SUITABLE CHECK
C             MARKS AND LABELS FOR AXES.
C
C             SUCCESSIVE FRAMES ARE SERIALIZED FROM AN INPUT VALUE
C
C             OPTIONS INCLUDE SOLID OR DOTTED MODE OF PLOTTING AND
C             HISTOGRAM-STYLE OR CUBIC-CURVE INTERPOLATION BETWEEN
C             SUCCESSIVE POINTS
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 PLUS 740 CRT RECORDER. (AND 780 CRT DISPLAY)
C STORAGE   - 1499 REGISTERS
C SPEED     - ON THE ORDER OF 2 SECONDS OR MORE PER FRAME (7090).
C AUTHOR    - S.M.SIMPSON JR, NOV 1961
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   DISPLA, LINE, XFIXM, FLOATM,
C                                       DSPFMT, FRAME, MVBLOK, SCPSCL
C                                       HSTPLT
C       AND FORTRAN SYSTEM ROUTINES -   (SPH), (FIL), LOG, EXP(2, XLOC
C
C                         NOTE-HSTPLT PLOTS THE DATA. THERE ARE SEVERAL
C                         VERSIONS OF THIS ROUTINE WHICH DIFFER IN THE
C                         PLOTTING STYLE USED (HISTOGRAM, CUBIC INTER-
C                         POLATION, VERTICAL LINES). USER SHOULD SELECT
C                         ONE (ALL HAVE CALLING SEQUENCES COMPATIBLE
C                         TO GRAPH)
C FORTRAN USAGE
C     CALL GRAPH (ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,YBOT,
C    1              XMAX,XMIN,NOPPP,IPAGE,SPACE)
C
C              PRELIMINARY DEFINITIONS
C              GRAPH PLOTS AN ARBITRARY NUMBER,NS,OF FLOATING POINT
C              SERIES IN THE SOLID MODE, PLUS AN ARBITRARY NUMBER,ND,
C              OF FLOATING POINT SERIES IN THE DOTTED MODE. NS OR
C              ND MAY BE ZERO. ALL SERIES HAVE THE SAME NO. OF TERMS, N.
C
C              LET THE SERIES TO BE PLOTTED SOLID BE DEFINED BY
C                YS1(1...N), YS2(1...N),..., YSNS(1...N)
C                    A TYPICAL MEMBER WILL BE REFERRED TO BY YS(1...N)
C
C              AND THE SERIES TO BE PLOTTED DOTTED BE
C                YD1(1...N), YD2(1...N),..., YDND(1...N)
C                    A TYPICAL MEMBER WILL BE REFERRED TO BY YD(1...N)
C
C              A TYPICAL SERIES DISREGARDING PLOTTING MODE IS Y(1...N)
C              Y(I) IS CONCEIVED AS CONTAINING THE FUNCTION YY(X),
C              WITH EQUAL INCREMENTS OF THE INDEPENDENT ARGUMENT X
C              OCCURRING BETWEEN SUCCESSIVE INDICES,IE
C                Y(1,2...N) = YY(XMIN,XMIN+DEL,XMIN+2*DEL,...,XMAX)
C                    WHERE DEL = (XMAX-XMIN)/N
C
C INPUTS
C
C    ISOL(I)   I=1...NS+1 IS A VECTOR WHICH GIVES THE LOCATIONS OF ALL
C                  SERIES WHICH ARE TO BE PLOTTED IN THE SOLID MODE
C                    ISOL(1)     =  XLOCF(YS1)
```

```
      C                     ISOL(2)     =  XLOCF(YS2)
      C                                ETC.
      C                     ISOL(NS)    =  XLOCF(YSNS)
      C                     ISOL(NS+1) =  0
      C                   (THE TERMINAL ZERO STOPS GRAPH FROM LOOKING FOR MORE
      C                     SERIES)
      C
      C     IDOT(I)    I=1...ND+1 IS A VECTOR WHICH GIVES THE LOCATIONS OF ALL
      C                   SERIES TO BE PLOTTED IN THE DOTTED MODE
      C                      IDOT(1) = XLOCF(YD1)
      C                      IDOT(2) = XLOCF(YD2)
      C                             ETC.
      C                      IDOT(ND) = XLOCF(YDND)
      C                      IDOT(ND+1) = 0
      C
      C     N          IS THE COMMON LENGTH OF ALL SERIES TO BE PLOTTED
      C                MUST EXCEED 1
      C
      C     TITLE(I)   I=1...8  CONTAINS 48 HOLERITH (8A6 FORMAT) TO BE USED
      C                  AS A HEADING TITLE ON ALL FRAMES.
      C                OPTIONALLY THESE 48 HOLERITH MAY BE GIVEN TO GRAPH BY
      C                THE LITERAL APPEARANCE IN THE CALLING SEQUENCE OF
      C                54H$$$$$$HOLERITHHOLERITHHOLERITHHOLERITHHOLERITHHOLERITH
      C                   THE FIRST MODE IS USFUL FOR HOLERITH WHICH THE USER
      C                   AQUIRES BY FORTRAN READ STATEMENTS.
      C                   THE SECOND MODE IS USEFUL WHEN THE TITLE TO BE USED
      C                   IS A CONSTANT OF THE USERS PROGRAM.
      C                THE TWO MODES HAVE A REVERSED SENSE OF STORAGE DIRECTION
      C                GRAPH DISTINGUISHES BETWEEN THE MODES BY THE PRESENCE OR
      C                ABSENCE OF 6 DOLLAR SIGNS IN TITLE(1).
      C                CONSEQUENTLY, FOR MODE 1 THE FIRST 6 OF THE 48 HOLERITH
      C                MUST NOT ALL BE DOLLAR SIGNS.
      C
      C     YUNITS(I) I=1...6 CONTAINS 36 HOLERITH (6A6 FORMAT) TO BE USED AS
      C                   A DESCRIPTIVE TITLE, ON THE VERTICAL AXIS, OF THE
      C                   UNITS OF Y(I).
      C                OPTIONALLY THESE 36 HOLERITH MAY BE GIVEN TO GRAPH BY
      C                THE LITERAL APPEARANCE IN THE CALLING SEQUENCE OF
      C                42H$$$$$$HOLERITHHOLERITHHOLERITHHOLERITHHOLE
      C                  (IF FIRST MODE IS USED YUNITS(1) MUST NOT = $$$$$$)
      C
      C     XUNITS(I) I=1...6 CONTAINS 36 HOLERITH (6A6 FORMAT) TO BE USED AS
      C                   A DESCRIPTIVE TITLE, ON THE HORIZONTAL AXIS, OF THE
      C                   UNITS OF X
      C                OPTIONALLY THESE 36 HOLERITH MAY BE GIVEN TO GRAPH BY
      C                THE LITERAL APPEARANCE IN THE CALLING SEQUENCE OF
      C                42H$$$$$$HOLERITHHOLERITHHOLERITHHOLERITHHOLE
      C                  (IF FIRST MODE IS USED XUNITS(1) MUST NOT = $$$$$$)
      C
      C     YTOP       DEFINES THE TOP OF THE PLOTTING AREA, SUCH THAT IF SOME
      C                   Y(I) = YTOP THEN Y(I) IS PLOTTED ON THE UPPERMOST
      C                   EDGE OF THE PLOTTING AREA
      C                IF ANY Y(I) EXCEEDS YTOP IT WILL BE TREATED AS
      C                  THOUGH IT WERE = YTOP
      C                YTOP MUST EXCEED YBOT
      C
      C     YBOT       DEFINES THE BOTTOM OF THE PLOTTING AREA, SUCH THAT IF
      C                   SOME Y(I) = YBOT THEN Y(I) IS PLOTTED ON THE
      C                   LOWERMOST EDGE OF THE PLOTTING AREA.
      C                IF ANY Y(I) IS LSTHN YBOT IT WILL BE TREATED AS
      C                  THOUGH IT WERE = YBOT
      C
      C                VALUES OF Y(I) BETWEEN YTOP AND YBOT ARE PLOTTED
      C                  PROPORTIONALLY BETWEEN THE UPPER AND LOWER EDGES.
      C
      C     XMAX       IS THE ARGUMENT VALUE CORRESPONDING TO Y(N) = YY(XMAX)
      C                XMAX MUST EXCEED XMIN
      C
      C     XMIN       IS THE ARGUMENT VALUE CORRESPONDING TO Y(1) = YY(XMIN)
      C
      C     NOPPP      IS THE DESIRED NO OF POINTS PER PAGE TO BE PLOTTED
      C                     Y(1...NOPPP)   APPEARS ON FIRST FRAME
      C                     Y(NOPPP...2*NOPPP-1)   ON SECOND FRAME, ETC.
      C                MUST EXCEED 2
```

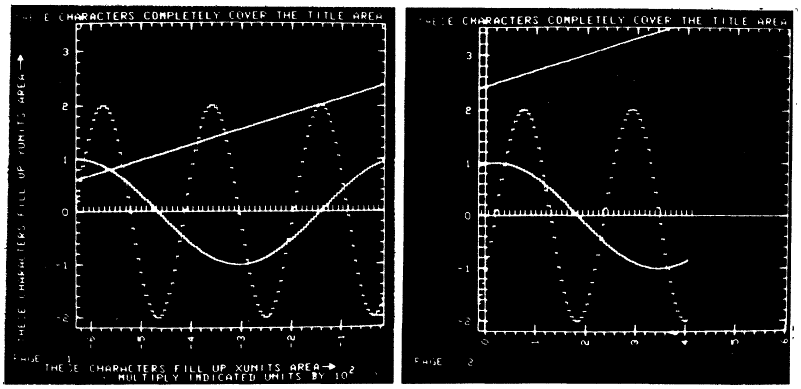
```
C               MUST BE LSTHN=401
C
C     IPAGE     IS AN INITIAL PAGE NO. TO BE PRINTED ON FIRST FRAME
C                 IPAGE+1,IPAGE+2,...  APPEARS ON SUCCESSIVE FRAMES
C               IS TREATED MODULO 1000
C
C     SPACE(I)  I=1...NOPPP MUST BE AVAILABLE TO GRAPH FOR SCRATCH WORK
C
C OUTPUTS
C
C
C     IPAGE     IS LEFT = ILAST+1  WHERE ILAST IS THE PAGE NO. APPEARING
C                 ON THE LAST FRAME USED BY GRAPH, THUS UPDATING IPAGE
C                 FOR A SUBSEQUENT CALL OF GRAPH.
C
C     SPACE(1)  IS USED AS AN ERROR INDICATOR
C               =0.0 IF NO TROUBLE
C               =1.0  IF N,YTOP,XMAX OR NOPPP IS ILLEGAL
C                       OR IF BOTH ISOL(1) AND IDOT(1) = 0
C
C SCOPE OUTPUTS
C               THE SCOPE OUTPUTS WILL BE DEFINED IN TERMS OF THE SCOPE
C               AREAS AFFECTED. IT SHOULD BE NOTED FIRST THAT GRAPH
C               DOES NOT CHANGE FRAMES BEFORE PLOTTING THE FIRST FRAME
C               NOR AFTER PLOTTING THE LAST ONE, THUS PERMITTING THE
C               USER TO PLOT ADDITIONAL INFORMATION ON THESE TWO FRAMES.
C               BY THE SAME TOKEN HOWEVER USER MUST CHANGE FRAMES
C               BETWEEN SUCCESSIVE CALLS TO AVOID SUPERPOSITION.
C
C               THE SCOPE FACE IS A SQUARE GRID OF POINTS (X,Y) WHERE
C               X AND Y CAN RANGE FROM 0 TO 1023. LET (0,0) BE THE
C               LOWER LEFT CORNER AND (1023,1023) THE UPPER RIGHT
C               CORNER WITH Y THE VERTICAL DIMENSION. THEN LET
C                       (X1,Y1)-(X2,Y2)
C               STAND FOR THE RECTANGULAR AREA WHOSE DIAGONAL RUNS
C               FROM (X1,Y1) TO (X2,Y2)
C               DEFINE THE FOLLOWING AREAS
C                 APLOT  = (175,150)-(1015,990)
C                 ATITLE =  (5,1000)-(1013,1021)
C                 AYUNIT =  (31,108)-(10,864)
C                 AYAROW =  (31,864)-(10,910)
C                 AYCKNO =  (76,140)-(160,1000)
C                 AYSCAL =  (71,120)-(50,981)
C                 AXUNIT =   (87,30)-(843,51)
C                 AXAROW =  (843,30)-(880,51)
C                 AXCKNO =  (167,55)-(1023,140)
C                 AXSCAL =   (144,5)-(1005,26)
C                 APAGE  =   (10,70)-(157,91)
C                 ACHEX  =      (0,0)-(63,21)
C                 AERROR = (100,500)-(688,521)
C               THEN
C
C     APLOT     IS THE PLOTTING AREA (SQUARE). IT IS BOXED IN ON ALL 4
C                 SIDES BY STRAIGHT LINES WITH CHECK MARKS ALL AROUND.
C                 THE VERTICAL CHECK MARK SEQUENCE (BETWEEN 20 AND 50)
C                 IS DETERMINED BY GRAPH SO AS TO DEFINE INTEGRAL POWERS
C                 OF 10 IN THE UNITS OF Y, AND A SIMILAR SEQUENCE IS
C                 DEVELOPED IN THE X DIRECTION.
C
C                 IF THE VALUE Y=0. FALLS BETWEEN YTOP AND YBOT, A
C                 CORRESPONDING HORIZONTAL AXIS IS DRAWN IN ON ALL FRAMES
C
C                 IF THE VALUE X=0. OCCURS ON SOME FRAME A VERTICAL
C                 AXIS IS DRAWN IN AND SUPPLIED WITH Y UNIT CHECK MARKS
C
C                 THE DATA Y(I) ARE PLOTTED EQUALLY SPACED IN THE X
C                 DIRECTION ACROSS THE PLOTTING AREA SUCH THAT Y(1)
C                 IS AT THE LEFT EDGE OF FRAME 1, Y(NOPPP)   AT THE
C                 RIGHT EDGE OF FRAME 1,Y(NOPPP)   AGAIN APPEARS AT
C                 THE LEFT EDGE OF FRAME 2, ETC. TILL THE DATA ARE GONE
C
C                 USE OF THE HISTOGRAM VERSION OF SUBROUTINE HSTPLT
C                 GIVES THE DATA PLOTTED AS HORIZONTAL BARS (WIDTH =
C                 PLOTTING WIDTH/(NOPPP-1)) CONNECTED BY VERTICAL LINES,
C                 THE LINES AND BARS BEING SOLID OR DOTTED ACCORDING
```

```
C                   TO ISOL, IDOT. THIS VERSION ALSO SUPPLIES AN
C                   ADDITIONAL SET OF CHECK MARKS TO THE BOTTOM OF THE
C                   PLOTTING AREA (OR TO THE Y=0. AXIS IF PRESENT)
C                   WHICH LOCATE THE CENTERS OF THE HISTOGRAM BARS. FOR
C                   NOPPP LESS THAN 81 THERE IS ONE SUCH CHECK FOR EACH
C                   BAR, FOR 81 LSTHN= NOPPP LISTHN= 160  ONE CHECK FOR
C                   EVERY OTHER BAR, FOR NOPPP GRTHN= 161 ONE CHECK FOR
C                   EVERY FIFTH BAR.
C
C                   THE CUBIC INTERPOLATOR VERSION OF HSTPLT FITS AND
C                   PLOTS A CUBIC CURVE BETWEEN SUCCESSIVE POINTS.  THE
C                   POINTS THEMSELVES ARE DARKENED.  THE HORIZONTAL AXIS
C                   WITH CHECKS IS PLOTTED AS IN THE ABOVE CASE.  THE
C                   VERTICAL BAR VERSION OF HSTPLT DRAWS A LINE FROM EACH
C                   POINT TO A HORIZONTAL AXIS.  THE VERTICAL POSITION OF
C                   THIS AXIS IS DEFINED BY THE FIRST VALUE IN THE SERIES.
C                   NO HORIZONTAL AXIS IS DRAWN FOR THE VALUE Y=0.
C
C                 OF THE REMAINING AREAS ONLY ATITLE, AXCKNO AND APAGE
C                 ARE CONTINUED BEYOND THE FIRST FRAME.
C
C     ATITLE      WILL SHOW THE 48 HOLERITH IN TITLE(1...8)
C
C     AYUNIT      WILL SHOW THE 36 HOLERITH IN YUNIT(1...6)
C
C     AYAROW      IS A VERTICAL ARROW
C
C     AYCKNO      SHOWS A SEQUENCE OF INTEGERS DEFINING THE VALUES OF
C                   Y CORRESPONDING TO THE CHECK MARKS ON THE VERTICAL AXIS
C
C     AYSCAL      MAY BE BLANK, OTHERWISE IT CONTAINS A DESCRIPTION OF
C                   HOW TO MODIFY THE INDICATED UNITS IN AYCKNO SO AS
C                   TO YIELD TRUE SCALE.
C
C     AXUNIT      WILL SHOW THE 36 HOLERITH IN XUNIT(1...6)
C
C     AXCKNO      IS LIKE AYCKNO BUT FOR THE HORIZONTAL AXIS
C
C     AXSCAL      IS LIKE AYSCAL BUT APPLIES TO AXCKNO
C
C     APAGE       IS THE PAGE NO SERIALIZING AREA STARTING WITH VALUE IPAGE
C
C     ACHEX       GIVES THE (NO. HISTOGRAM CHKMARKS)/(DATA POINT) RATIO
C                 =  BLANK     IF  RATIO = 1
C                 =  1/2       IF  1 CHK /(2 DATA PTS)
C                 =  1/5       IF  1 CHK /(5 DATA PTS)
C
C     AERROR      NOT USED NORMALLY
C                 SAYS ILLEGAL ARGUMENT FOR GRAPH  IF ANY ARGUMENT ILLEGAL
C
C EXAMPLES
C     EXAMPLES 1. THRU 4. ARE INTENDED TO BE RUN USING THE HISTOGRAM
C     STYLE VERSION OF SUBROUTINE HSTPLT. EXAMPLES 6. AND 7. SHOW EFFECTS
C     OF OTHER VERSIONS.
C
C 1. SINGLE FRAME EXAMPLE
C     INPUTS  - YS1(1...201)= COSF(0.),(.05),(.10),...
C                YS2(1...201)= .600,.615,.630,...
C                YD1(1...201)= 2.*SINF(0.),(.15),(.30),...
C                ISOL(1...2)= XLOCF(YS1),(YS2)      ISOL(3)=0
C                IDOT(1)= XLOCF(YD1)                IDOT(2)=0
C                YTOP= 3.5   YBOT= -2.2
C                N=201
C                XMAX= 405.   XMIN= -630.
C                TITLE(1...8)=
C                      48HTHESE CHARACTERS COMPLETELY COVER THE TITLE AREA
C                IPAGE=1
C                NOPPP=201
C     USAGE   -        CALL FRAME
C                      CALL GRAPH(ISOL,IDOT,N,TITLE,
C                    1 42H$$$$$$THESE CHARACTERS FILL UP YUNITS AREA,
C                    2 42H$$$$$$THESE CHARACTERS FILL UP XUNITS AREA,
C                    3 YTOP,YBOT,XMAX,XMIN,NOPPP,IPAGE,SPACE)
```

```
C     OUTPUTS - IPAGE=2    SPACE(1)=0.
C
C
```

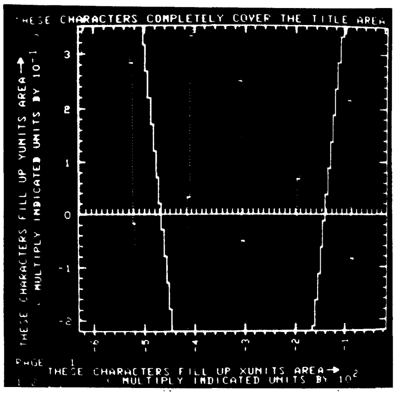


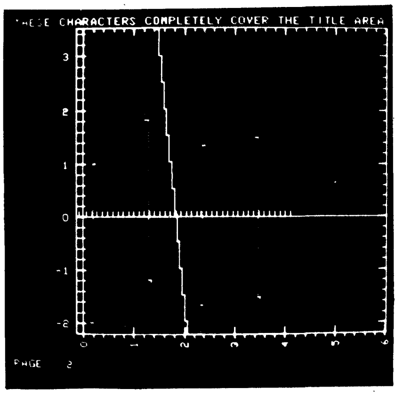

```
C
C
C 2. DOUBLE FRAME EXAMPLE
C     INPUTS  - SAME AS EXAMPLE 1 EXCEPT NOPPP= 120
C     USAGE   - SAME AS EXAMPLE 1
C     OUTPUTS - IPAGE= 3    SPACE(1)=0.
C
C
C
```



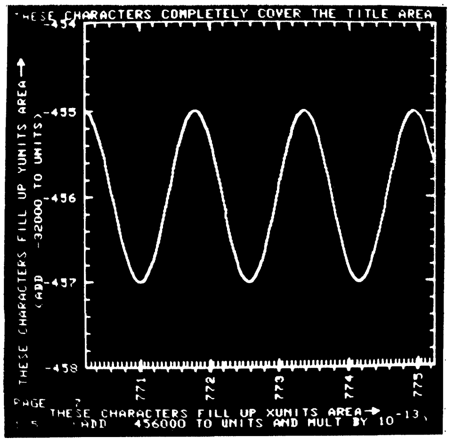

```
C
C
C
C
C 3. EXPLODED VIEW OF EXAMPLE 2
C     INPUTS  - SAME AS EXAMPLE 2. EXCEPT YTOP= .35    YBOT=-.22
C     USAGE   - SAME AS EXAMPLE 1
```

```
C     OUTPUTS - IPAGE= 3   SPACE(1)= .0
C
C
C
C
```



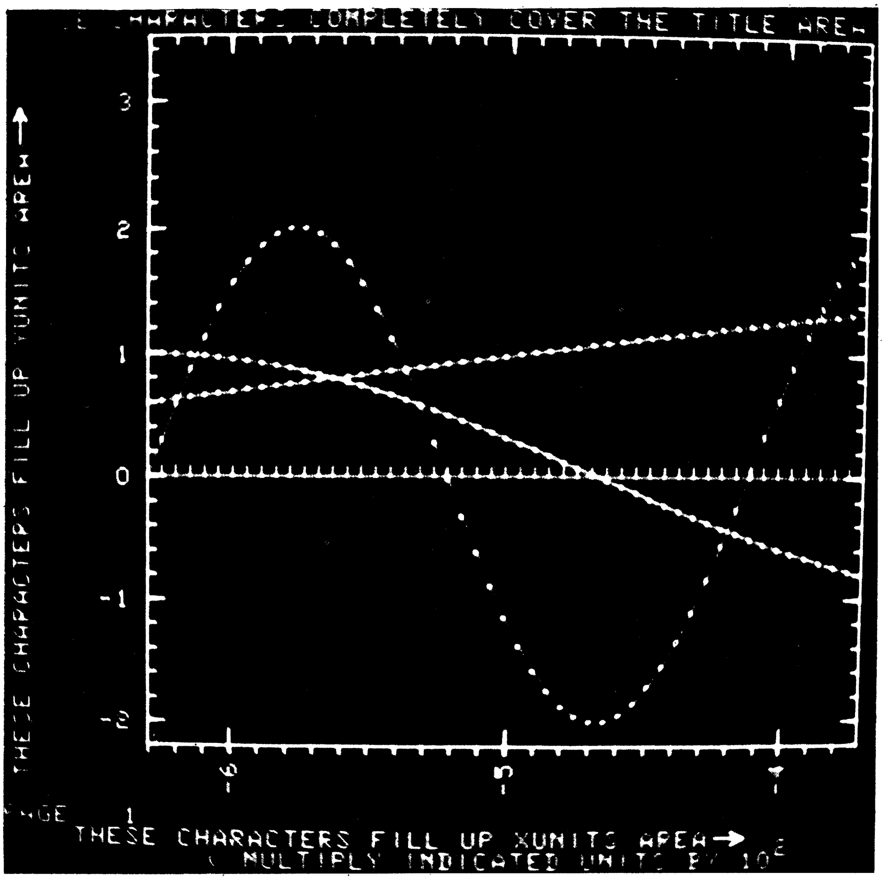



```
C
C
C
C 4. SOME EXTREME SCALING
C     INPUTS  - SAME AS EXAMPLE 1 EXCEPT
C               YS1(1...800)= -32456.+COSF(0.),(.05),(.10),...    N=800
C               ISOL(1)= XLOCF(YS1)   ISOL(2)=0   IDOT(1)=0
C               YTOP= -32454.   YBOT= -32458.
C               XMAX= .45678*10**(-7)   XMIN=.45677*10**(-7)
C               NOPPP=401   IPAGE=7
C     USAGE   - SAME AS EXAMPLE 1
C     OUTPUTS - IPAGE= 9   SPACE(1)= 0.
C
C
```



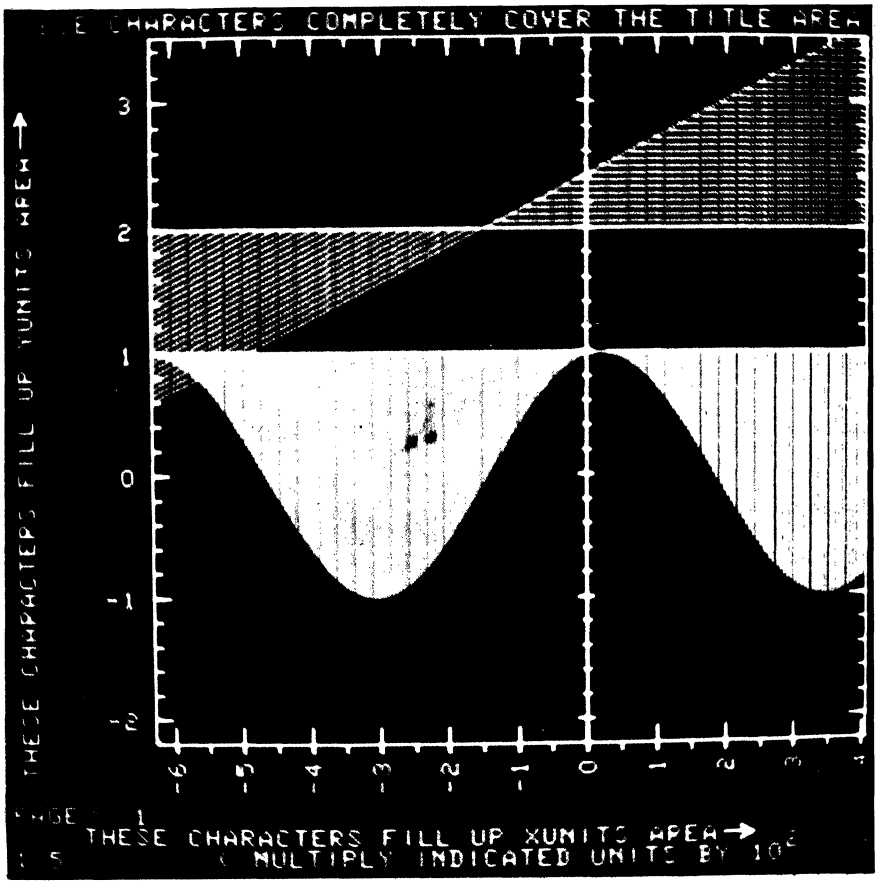

```
C
C
C
C
C 5. ERROR CONDITIONS
C     USAGE   -         CALL GRAPH(ISOL,IDOT,0,TITLE,YUNITS,XUNITS,10.,5.,
C                                 7.,2.,100,2,SPACE(1))
C                       CALL GRAPH(ISOL,IDOT,5,TITLE,YUNITS,XUNITS,5.,5.,
C                                 7.,2.,100,2,SPACE(2))
C                       CALL GRAPH(ISOL,IDOT,5,TITLE,YUNITS,XUNITS,10.,5.,
C                                 1.,2.,100,2,SPACE(3))
C                       CALL GRAPH(ISOL,IDOT,5,TITLE,YUNITS,XUNITS,10.,5.,
C                                 7.,2.,2,2,SPACE(4))
C                       CALL GRAPH(ISOL,IDOT,5,TITLE,YUNITS,XUNITS,10.,5.,
C                                 7.,2.,500,2,SPACE(5))
C     OUTPUTS - SPACE(1)=1. (ILLEGAL N)      SPACE(2)=1. (ILLEGAL YTOP)
```

```
C               SPACE(3)=1. (ILLEGAL XMAX)  SPACE(4)=1. (ILLEGAL NOPPP)
C               SPACE(5)=1. (ILLEGAL NOPPP)
C
C 6. USE OF CUBIC INTERPOLATION VERSION OF HSTPLT
C         (NOTE- THERE IS A 709 AND A 7090 VERSION OF THIS HSTPLT)
C    INPUTS  - SAME AS EXAMPLE 1 EXCEPT NOPPP=51
C    USAGE   - DITTO
C    OUTPUTS - DITTO
C                        (ONLY FIRST FRAME SHOWN BELOW)
C
```




```
C
C
C
C 7. USE OF VERTICAL BAR VERSION OF HSTPLT
C    NOTE THAT THE VERTICAL POSITION OF THE HORIZONTAL BAR, FROM
C      WHICH THE VERTICAL BARS ARE PLOTTED IS DEFINED BY THE INITIAL
C      VALUE OF EACH SERIES.
C    INPUTS  - SAME AS EXAMPLE 1 EXCEPT ISOL(2)=0    YD1(1...201)= 2.,
C              .615,.630,.645,...
C    USAGE   - SAME AS EXAMPLE 1
```

```
C     OUTPUTS - DITTO
C
```




```
C PROGRAM FOLLOWS BELOW
C
C DIMENSION STATEMENT FOR ARGUMENTS
      DIMENSION ISOL(100),IDOT(100),TITLE(8),YUNITS(6),XUNITS(6),
     1           SPACE(401)
C TRUE DIMENSION STATEMENTS
      DIMENSION HSTORG(4),FMT(10)
      DIMENSION HOLER(8)
      EQUIVALENCE (ONE,IONE),(ONEK,IONEK)
C CHECK LEGALITIES
      IF(N-1) 5,5,1
  1   IF(YTOP-YBOT)  5,5,2
  2   IF(XMAX-XMIN)  5,5,3
  3   IF(NOPPP-2)  5,5,4
  4   IF(NOPPP-401) 7,7,5
  7   IF (ISOL) 10,8,10
  8   IF (IDOT) 10,5,10
  5   CALL DISPLA
      PRINT 6
  6   FORMAT(39H2SH100,500,ILLEGAL ARGUMENT FOR GRAPH  )
      SPACE(1)=1.0
      GO TO 310
C INITIAL SETTINGS, ETC
   10 XRNGE=XMAX-XMIN
      YRNGE=YTOP-YBOT
      NN=N
      DELX=XRNGE/(FLOATF(NN-1))
```

```
      PGDELX=DELX*FLOATF(NOPPP-1)
      NDELX=(840.0*128.0)/FLOATF(NOPPP-1)
      IPAGE=XMODF(IPAGE,1000)
B     DOLARS=535353535353
      VLPOS=175.0
      VRPOS=1015.0
      HBPOS=150.0
      HTPOS=990.0
      NPLOTD=0
      IYSCLX=76
      IXSCLY=51
C SET UP FOR BAR PLOTTING BY HSTPLT
      IFRSTB=1
      ISKIPB=1
      IF (NOPPP-80)   415,415,1001
 1001 IF (NOPPP-160)   1002,1002,1003
 1002 ISKIPB=2
      GO TO 1004
 1003 ISKIPB=5
 1004 CALL DISPLA
      PRINT 1005,ISKIPB
 1005 FORMAT (9H2SH1,1,1/,1I1)
C SET CONVERSION CONSTANTS
 415  CONVL=(HTPOS-HBPOS)/(YTOP-YBOT)
      CONVK=HTPOS-YTOP*CONVL
      CNVL=(VRPOS-VLPOS)/(PGDELX)
      HSTORG(1)=175.0
      HSTORG(3)=1015.0
C CHECK IF Y=0 LINE APPEARS ON GRAPH     (420 IS NO)
      IF (YTOP)  420,420,416
  416 IF(YBOT)   418,420,420
C IF SO SET
  418 HSTORG(2)=CONVK
      HSTORG(4)=CONVK
      GO TO 422
C IF NOT SET
  420 HSTORG(2)=HBPOS
      HSTORG(4)=HBPOS
C INITIALIZE NO. PTS TO PLOT ON FIRST PAGE
  422 IF (NOPPP-NN)   424,424,426
  424 NOPTP=NOPPP
      GO TO 428
  426 NOPTP=NN
C DOES AN X=0 LINE OCCUR ON ANY FRAME   (440 IS NO)
 428  IF (XMAX)    440,432,430
  430 IF (XMIN)    432,440,440
C IF YES SET THE PAGE NO AND THE X COORD ON THAT PAGE
  432 ITEMP1=(-XMIN)/PGDELX
      IPGXZ=ITEMP1+IPAGE
      IPGXZ=XMODF(IPGXZ,1000)
      XZER=VLPOS + 840.0*(-XMIN-PGDELX*FLOATF(ITEMP1))/(PGDELX)
      GO TO 450
  440 IPGXZ=0
      XZER=0.0
C PLOT X AND Y AXIS LABELS WITH ARROWS
  450 IF (YUNITS-DOLARS)  4406,4402,4406
 4402 DO 4404 I=1,6
      J=7-I
 4404 HOLER(J)=YUNITS(I+32762)
      GO TO 4408
 4406 DO 4407 I=1,6
 4407 HOLER(I)=YUNITS(I)
 4408 CALL DISPLA
      PRINT 452,(HOLER(I),I=1,6)
  452 FORMAT(10H2SV31,108, 6A6)
      CALL LINE(20.,868.,20.,910.)
      CALL LINE(28.,895.,20.,910.)
      CALL LINE(12.,895.,20.,910.)
      IF (XUNITS-DOLARS)  4414,4410,4414
 4410 DO 4412  I=1,6
      J=7-I
 4412 HOLER(J)=XUNITS(I+32762)
      GO TO 4418
 4414 DO 4416  I=1,6
 4416 HOLER(I)=XUNITS(I)
```

GRAPH (PAGE 10)

```
 4418 CALL DISPLA
      PRINT 454,(HOLER(I),I=1,6)
  454 FORMAT(9H2SH87,30,6A6)
      CALL LINE(845.,40.,880.,40.)
      CALL LINE(865.,32.,880.,40.)
      CALL LINE(865.,48.,880.,40.)
C GO FIND CHECK MARK SETTINGS FOR VERTICAL SCALE
  456 ASSIGN 460 TO ISTOUT
      TOP = YTOP
      BOTTOM=YBOT
C GO TO INTERNAL SUBROUTINE TO SET UP CHECK MARKS,SCALES
      GO TO 21
C AFTER COMING BACK GO OUTPUT VERTICAL
  460 ASSIGN 462 TO ISCOUT
      VORH=1.0
      GO TO 520
C AFTER COMING BACK SCALE STUFFTO SCOPE UNITS FOR FRAME LOOP
  462 VUORG=HBPOS+CONVL*(BRLOU-YBOT)
      VUDEL=CONVL*DBRLOU
      VLORG=HBPOS+CONVL*(BRLOL-YBOT)
      VLDEL=CONVL*DBRLOL
      INTVL=NBRLOL
C GO SET HORIZONTAL SCALES
  464 ASSIGN 470 TO ISTOUT
      TOP =XMIN+PGDELX
      BOTTOM=XMIN
      GO TO 21
C GO OUTPUT HORIZONTAL SCALES
  470 ASSIGN 472 TO ISCOUT
      VORH=0.0
      GO TO 520
C SCALE STUFF TO SCOPE UNITS FOR FRAME LOOP AND GO THERE
  472 HUORG=VLPOS+CNVL*(BRLOU-XMIN)
      HUDEL=CNVL*DBRLOU
      HLORG=VLPOS+CNVL*(BRLOL-XMIN)
      HLDEL=CNVL*DBRLOL
      INTHL=NBRLOL
      GO TO 200
C THIS IS AN INTERNAL SUBROUTINE WHICH, GIVEN
C     TOP = VALUE AT TOP OF SCOPE  (IN Y UNITS)
C     BOTTOM = VALUE AT BOTTOM OF SCOPE  (IN Y UNITS)
C FINDS
C     BRLOU = Y VALUE FOR LOWEST UNLABELED BAR
C     DBRLOU = Y INCREMENT BETWEEN UNLABELED BARS
C     BRLOL = Y VALUE FOR LOWEST LABELED BAR
C     DBRLOL = Y INCREMENT BETWEEN LABELED BARS
C     NBRLOL = INTEGER TO PLOT NEXT TO LOWEST LABELED BAR
C              (MAX 3 DIGITS PLUS SIGN)
C     NEXP = POSITIVE OR NEGATIVE INTEGER TO PLOT AS EXPONENT
C                         (MAXIMUM 2 DIGITS PLUS SIGN)
C     NCONST = POSITIVE OR NEGATIVE INTEGER TO ADD TO LABELS
C              = NO. THOUSANDS TO BE ADDED TO LABELS
C                         (MAX 5 DIGITS PLUS SIGN)
C THE INCREMENTS FOUND WILL BE SUCH THAT THE TOTAL NO. OF CHECK MARKS
C WILL BE BETWEEN 20 AND 50
C (LOOP ALSO USED IN X DIRECTION)
C     FIRST OF ALL FIND THE CONSTANTS WHICH WOULD RESULT IF
C DATA RANGED FROM 0 TO (TOP-BOTTOM)
C INITIALIZE
 21   DATMAX=TOP-BOTTOM
      NCONST=0
      NBRLOL=0
      BRLOU=0.
      BRLOL=0.
C BEGIN BY FINDING NEXP AND DBRLOL=10**NEXP SUCH THAT
C       10.**NEXP LSTHN DATMAX LSTHN=10.**(NEXP+1)
C SET TRIAL NEXP
C (THE CONST IS LOG, TO BASE 10, OF E=2.718281828)
      NEXP=0.43429448*LOGF(DATMAX)
      IF(DATMAX-1.0)  33,34,34
 33   NEXP=NEXP-1
 34   DBRLOL=10.**NEXP
C SET DBRLOU ACCORDING TO HOW MANY TIMES DBRLOL GOES INTO DATMAX
      NTMS=DATMAX/DBRLOL
      IF(NTMS-1)  60,60,62
```

```
C NTMS IS 1, OR 0 (MEANS DBRLOL= DATMAX PLUS EPSILON)
 60   NEXP=NEXP-1
      DBRLOL=DBRLOL/10.
      DBRLOU=DBRLOL/2.
      GO TO 70
C IS NTMS 2,3, OR GREATER
 62   IF(NTMS-3)  64,64,65
C 2 OR 3
 64   DBRLOU=DBRLOL/10.
      GO TO 70
C 4,5...9, OR 10(DBRLOL=DATMAX/10. MINUS EPSILON)
 65   DBRLOU=DBRLOL/5.
 70   CONTINUE
B     ONE=1
B     ONEK=1750
      NTIMS=XFIXMF(0,BOTTOM/DBRLOL)
      IF(BOTTOM)  78,99,74
 74   IF(DBRLOL*FLOATMF(NTIMS)-BOTTOM)  76,78,76
 76   NTIMS=NTIMS+IONE
 78   NBRLOL=NTIMS
      BRLOL=DBRLOL*FLOATMF(NTIMS)
      BRLOU=BRLOL
 80   IF (BRLOU-BOTTOM)  84,86,82
 82   BRLOU=BRLOU-DBRLOU
      GO TO 80
 84   BRLOU=BRLOU+DBRLOU
 86   IF(NBRLOL)  88,98,94
C NEG
 88   IF (NBRLOL+IONEK)  90,90,99
 90   NCONST=NCONST-IONE
      NBRLOL=NBRLOL+IONEK
      GO TO 88
C POS
 94   IF (NBRLOL-IONEK)  99,96,96
 96   NCONST=NCONST+IONE
      NBRLOL=NBRLOL-IONEK
      GO TO 94
 98   NBRLOL=0
 99   NCONST=XFIXF(FLOATMF(NCONST))
      NBRLOL=XFIXF(FLOATMF(NBRLOL))
      GO TO ISTOUT,(460,470)
C THIS IS AN INTERNAL SUBROUTINE WHICH, GIVEN
C     NEXP = POSITIVE OR NEGATIVE INTEGER TO PLOT AS EXPONENT
C                            (MAX 2 DIGITS PLUS SIGN)
C     NCONST = POSITIVE OR NEGATIVE INTEGER TO BE ADDED TO LABELS
C                            (MAX 5 DIGITS PLUS SIGN)
C     VORH = 1.0 FOR VERTICAL, = 0.0 FOR HORIZONTAL
C PLOTS THE SCALE CONVERSION FIELD (VERTICAL OR HORIZONTAL)  AS FOLLOWS
C    IF NCONST = 0  IN MAGNITUDE THEN
C       A.  NO CONVERSION FIELD IS PLOTTED IF NEXP IS ALSO = 0
C       B.  IF NEXP = 1 IT PLOTS
C                  (MULTIPLY INDICATED UNITS BY 10)
C       C.  OTHERWISE IT PLOTS                 NEXP
C                  (MULTIPLY INDICATED UNITS BY 10    )
C    IF NCONST IS NON-ZERO IT PLOTS
C                  (ADD NCONST000 TO UNITS
C   AND A.  IF NEXP = 0  IT ADDS A RIGHT )
C       B.  IF NEXP = 1  IT ADDS
C                                 AND MULT BY 10)
C       C.  OTHERWISE IT ADDS
C                                            NEXP
C                                 AND MULT BY 10    )
C FIRST CHECK STRAIGHT EXIT FOR NCONST=NEXP=0
  520 IF (NCONST) 531,522,531
  522 IF (NEXP)  524,599,524
C NCONST IS ZERO, NEXP IS NOT, CHECK HOR OR VERT
  524 IF (VORH)  528,526,528
C HORIZONTAL
  526 CALL DISPLA
      PRINT 527
  527 FORMAT(46H2SH144,5,     ( MULTIPLY INDICATED UNITS BY 10)
      GO TO 542
C VERTICAL
  528 CALL DISPLA
      PRINT 529
```

```
  529 FORMAT(47H2SV71,120,      ( MULTIPLY INDICATED UNITS BY 10)
      GO TO 542
C NCONST IS NOT ZERO, CHECK HOR OR VER
  531 IF (VORH)  537,532,537
C HORIZONTAL.  CHECK NEXP = 0
  532 IF (NEXP)  535,533,535
  533 CALL DISPLA
      PRINT 534,NCONST
  534 FORMAT(13H2SH250,5,(ADD,1I6,13H000 TO UNITS))
      GO TO 599
  535 CALL DISPLA
      PRINT 536,NCONST
  536 FORMAT(13H2SH144,5,(ADD,1I6,27H000 TO UNITS AND MULT BY 10)
      GO TO 542
C VERTICAL WITH NCONST NOT = 0.  CHECK NEXP = 0
  537 IF (NEXP)  540,538,540
  538 CALL DISPLA
      PRINT 539,NCONST
  539 FORMAT(14H2SV71,300,(ADD,1I6,13H000 TO UNITS))
      GO TO 599
  540 CALL DISPLA
      PRINT 541,NCONST
  541 FORMAT(14H2SV71,120,(ADD,1I6,27H000 TO UNITS AND MULT BY 10)
      GO TO 542
C THIS PORTION OF THE INTERNAL ROUTINE FILLS IN THE RIGHT END
C                                    NEXP
C OF SCALE FIELD = ) IF NEXP = 1, =      )  OTHERWISE
C (NOTE PROGRAM DOES NOT GET HERE IF NEXP = 0)
C FIRST CHECK HORIZONTAL OR VERTICAL
  542 IF (VORH)  583,548,583
C HORIZONTAL (UP TO STATEMENT 583)
C CHECK NEXP = 1
  548 IF (NEXP-1)  554,572,554
C FILL IN NEXP ACCORDING TO HOW MANY DIGITS INVOLVED (INCLUDING SIGN)
  554 IF (NEXP)  556,558,558
C TWO OR THREE
  556 IF (NEXP+10)  568,568,564
C ONE OR TWO
  558 IF (NEXP-10)  560,564,564
C ONE DIGIT EXPONENT
  560  CALL DISPLA
      PRINT 562,NEXP
  562 FORMAT (10H2SH921,20,,1I1)
      GO TO 572
C TWO DIGIT EXPONENT
  564 CALL DISPLA
      PRINT 566,NEXP
  566 FORMAT(10H2SH921,20,,1I2)
      GO TO 572
C THREE DIGIT EXPONENT
  568 CALL DISPLA
      PRINT 570,NEXP
  570 FORMAT (10H2SH921,20,,1I3)
C FILL IN RIGHT PAREN AND LEAVE
  572 CALL DISPLA
      PRINT 574
  574 FORMAT (10H2SH984,5,))
      GO TO 599
C FILL IN EXPONENT BUSINESS FOR VERTICAL FIELD
C CHECK FOR NEXP = 1
  583 IF (NEXP-1)  585,594,585
C FILL IN NEXP ACCORDING TO NUMBER OF DIGITS
  585 IF (NEXP)  586,587,587
  586  IF (NEXP+10)  593,593,592
  587 IF (NEXP-10)  590,592,592
C ONE DIGIT
  590 CALL DISPLA
      PRINT 591,NEXP
  591 FORMAT(10H2SV61,897,,1I1)
      GO TO 594
C TWO DIGITS
  592 CALL DISPLA
      PRINT 5920,NEXP
 5920 FORMAT(10H2SV61,897,,1I2)
      GO TO 594
```

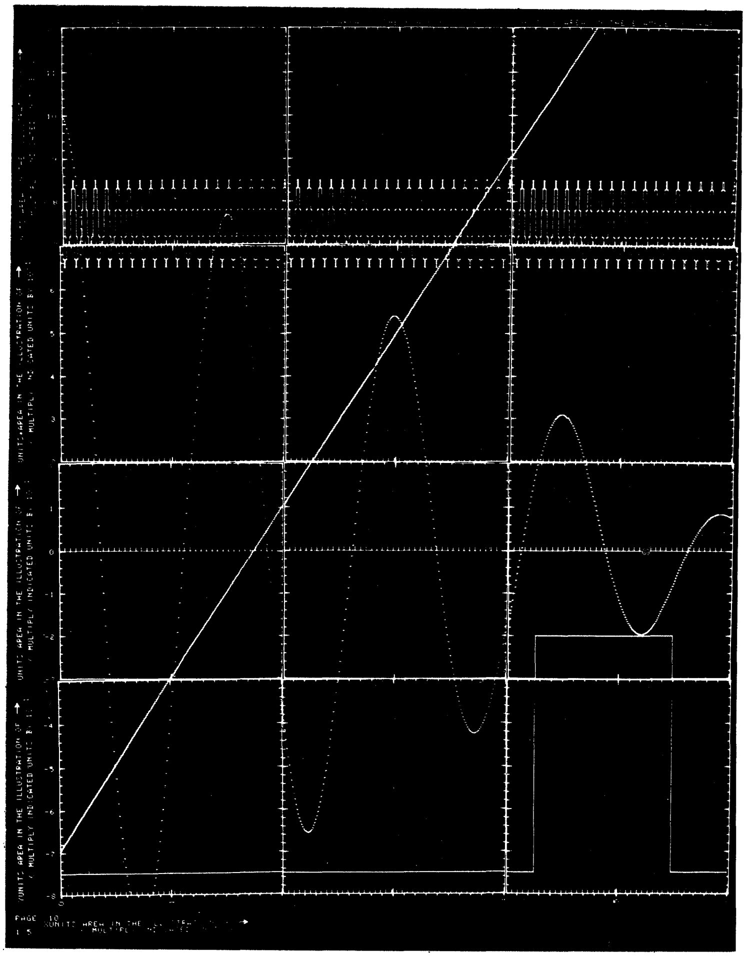
```
C THREE DIGITS
  593 CALL DISPLA
      PRINT 5930,NEXP
 5930 FORMAT(10H2SV61,897,,1I3)
C FILL IN RIGHT PAREN
  594 CALL DISPLA
      PRINT 595
  595 FORMAT (11H2SV71,960,))
C EXIT FROM INTERNAL SUBROUTINE
  599 GO TO ISCOUT, (462,472)
C THIS IS THE FRAME PLOTTING LOOP
C
C SEQUENCE OF EVENTS IS
C   PLOT TITLE
C   PLOT PAGE NO.
C   PLOT X AXIS IF IT OCCURS ON THIS FRAME
C   PLOT BOX
C   PLOT VERTICAL CHECK MARKS AND LABELS
C   PLOT HORIZONTAL CHECK MARKS AND LABELS
C   IF THERE IS DATA FOR SOLID CURVE
C      A.  GET AND SCALE DATA SUBSETS FOR THIS FRAME
C      B.  USE HSTPLT TO PLOT THEM
C   IF THERE IS DATA FOR DOTTED CURVE
C      A.  GET AND SCALE DATA SUBSETS FOR THIS FRAME
C      B.  USE HSTPLT TO PLOT THEM
C   IF MORE DATA, RESET, CALL FRAME, RETURN TO PLOT TITLE ABOVE
C   IF NOT, EXIT
C IMPORTANT PARAMETERS OF THE FRAME LOOP (MOSTLY SCOPE UNITS)
C
C VLPOS IS X COORD OF LEFT VERTICAL   FLTG PT
C VRPOS IS X COORD OF RIGHT VERTICAL
C HBPOS IS Y COORD OF BOTTOM LINE
C HTPOS IS Y COORD OF TOP LINE
C VUORG IS Y COORD OF FIRST UNLABELED CHECK MARK ON VERTICAL AXIS (0.0
C     MEANS NONE)
C VUDEL IS Y INCREMENT OF UNLABELED CHECK MARK ON VERTICAL AXIS (0.0
C     MEANS NONE)
C VLORG IS Y COORD OF FIRST LABELED CHECK MARK ON VERTICAL AXIS
C VLDEL IS Y INCREMENT OF LABELED CHECK MARK ON VERTICAL AXIS
C INTVL IS FIRST INTEGER TO LABEL CHECK MARK ON VERTICAL AXIS
C HUORG IS X COORD OF FIRST UNLABELED CHECK MARK ON HOR. AXIS(0.0 MEANS
C HUDEL IS X INCREMENT OF UNLABELED CHECK MARK ON HOR. AXIS(0.0 MEANS NO
C HLORG IS X COORD OF FIRST LABELED CHECK MARK ON HOR AXIS
C HLDEL IS X INCREMENT OF LABELED CHECK MARK ON HOR AXIS
C INTHL IS FIRST INTEGER TO LABEL CHECK MARK ON HOR AXIS
C IPAGE IS PAGE NO TO PLOT (INIT=1)
C IPGXZ IS THE PAGE NO. ON WHICH X=0 OCCURS (NEG IF NONE)
C XZER IS THE X COORD OF THE X=0 LINE (ON PAGE IT OCCURS ON)
C CONVK IS A CONVERSION CONSTANT
C CONVL IS A CONVERSION CONSTANT
C     WHERE DATA (SCOPE UNITS) = CONVK + (DATA(INPUT UNITS))*CONVL
C NOPPP IS THE NUMBER OF PTS PER PAGE (FRAME) TO PLOT  (ARGUMENT)
C NN IS THE TOTAL NUMBER OF DATA PTS                   (ARGUMENT)
C NPLOTD IS THE NUMBER OF DATA POINTS PLOTTED SO FAR(INITIALIZED TO ZERO
C ICHANL(I) = BUFFER BLOCK WHICH FEEDS HSTPLT DATA
C IYSCLX = X COORD OF LEFT EDGE OF Y SCALE FIELD IN SCOPE UNITS
C IXSCLY = Y COORD OF LOWER EDGE OF X SCALE FIELD IN SCOPE UNITS
C HSTORG(I)  I=1,4  = X1,Y1,X2,Y1  AXIS SPEC FOR HSTPLOT
C AXIS = INDICATOR FOR HSTPL, =0.0 MEANS PLOT HOR AXIS,
C NDELX = DELTA X FOR PLOT * 2**7
C NOPTP = NO OF POINTS TO PLOT ON EACH FRAME (INIT=MIN(NOPPP,NN)
C PLOT TITLE
  200 IF (TITLE-DOLARS)  2006,2002,2006
 2002 DO 2004   I=1,8
      J=9-I
 2004 HOLER(J)=TITLE(I+32760)
      GO TO 2010
 2006 DO 2008   I=1,8
 2008 HOLER(I)=TITLE(I)
 2010 CALL DISPLA
      PRINT 202, (HOLER(I),I=1,8)
 202  FORMAT (10H2SH5,1000,8A6)
CPLOT PAGE NO.
      CALL DISPLA
      PRINT 203, IPAGE
```

```
 203  FORMAT  (12H2SH1,55,PAGE,1I4)
C IF X AXIS OCCURS ON THIS PAGE, PLOT IT ALONG WITH CHECK MARKS (UNLABEL
      IF (IPAGE-IPGXZ)   209,204,209
C AXIS
 204  CALL LINE (XZER,HBPOS,XZER,HTPOS)
      IF (VUORG)     205,207,205
C UNLABELED
 205  TEMP1=VUORG
      TEMP2=XZER-5.0
      TEMP3=XZER+5.0
 206  CALL LINE (TEMP2,TEMP1,TEMP3,TEMP1)
      TEMP1=TEMP1+VUDEL
      IF(TEMP1-HTPOS)     206,206,207
C LABELED CHECKS PLOTTED WITHOUT LABELS
 207  TEMP1=VLORG
      TEMP2=XZER-10.0
      TEMP3=XZER+8.0
 208  CALL LINE (TEMP2,TEMP1,TEMP3,TEMP1)
      TEMP1=TEMP1+VLDEL
      IF (TEMP1-HTPOS)     208,208,209
C PLOT BOX
 209  CALL LINE(VLPOS,HBPOS,VLPOS,HTPOS)
      CALL LINE(VLPOS,HTPOS,VRPOS,HTPOS)
      CALL LINE(VRPOS,HTPOS,VRPOS,HBPOS)
      CALL LINE(VRPOS,HBPOS,VLPOS,HBPOS)
C PLOT UNLABELED CHECK MARKS ON VERTICAL AXIS IF THERE ARE ANY
      IF (VUORG)     210,220,210
 210  TEMP1=VUORG
      TEMP2=VLPOS+10.0
      TEMP3=VRPOS-10.0
 212  CALL LINE(VLPOS,TEMP1,TEMP2,TEMP1)
      CALL LINE(TEMP3,TEMP1,VRPOS,TEMP1)
      TEMP1=TEMP1+VUDEL
      IF (TEMP1-HTPOS)     212,212,220
C PLOT LABELED CHECK MARKS ON VERTICAL AXIS AND LABELS
 220  TEMP1=VLORG
      TEMP2=VLPOS+20.0
      TEMP3=VRPOS-20.0
      ITEMP1=INTVL
 222  CALL LINE(VLPOS,TEMP1,TEMP2,TEMP1)
      CALL LINE(TEMP3,TEMP1,VRPOS,TEMP1)
      ITEMP2=TEMP1-10.0
      CALL DSPFMT (3H2SH,IYSCLX,ITEMP2,4H 1I4,FMT)
      CALL DISPLA
      PRINT FMT,ITEMP1
      TEMP1=TEMP1+VLDEL
      ITEMP1=1+ITEMP1
      IF(TEMP1-HTPOS)     222,222,230
C PLOT UNLABELED CHECK MARKS ON HORIZONTAL AXIS IF THERE ARE ANY
 230  IF (HUORG)     232,240,232
 232  TEMP1=HUORG
      TEMP2=HBPOS-10.0
      TEMP3=HTPOS-10.0
 234  CALL LINE(TEMP1,TEMP2,TEMP1,HBPOS)
      CALL LINE (TEMP1,TEMP3,TEMP1,HTPOS)
      TEMP1=TEMP1+HUDEL
      IF (TEMP1-VRPOS)     234,234,236
C RESET CHECK MARK ORIGIN FOR NEXT FRAME
 236  HUORG=TEMP1-VRPOS+VLPOS
C PLOT LABELED CHECK MARKS ON HORIZONTAL AXIS AND LABELS
 240  TEMP1=HLORG
      ITEMP1=INTHL
      TEMP2=HBPOS-10.0
      TEMP3=HBPOS+20.0
      TEMP4=HTPOS-20.0
 242  CALL LINE (TEMP1,TEMP2,TEMP1,TEMP3)
      CALL LINE(TEMP1,TEMP4,TEMP1,HTPOS)
      ITEMP2=TEMP1+8.0
      CALL DSPFMT(3H2SV,ITEMP2,IXSCLY,4H 1I4,FMT)
      CALL DISPLA
      PRINT FMT,ITEMP1
      ITEMP1=1+ITEMP1
      TEMP1=TEMP1+HLDEL
      IF (TEMP1-VRPOS)     242,242,250
C RESET CHECK MARK ORIGIN AND INTEGER LABEL FOR NEXT FRAME
```

```
  250  HLORG=TEMP1-VRPOS+VLPOS
       INTHL=ITEMP1
C IF THERE IS DATA FOR SOLID PLOTS, PLOT THEM ONE AT A TIME
C PUTTING AXIS ONLY ON FIRST ONE (AXIS PLOT FOR AXIS=0.0)
  261  NSOL=0
       TYPE=0.0
C INDEX FOR NEXT SERIES
  262  NSOL=NSOL+1
C EXIT WHEN HIT ZERO ADDRESS
       IF(ISOL(NSOL)) 4,270,2630
C SET SERIES ADDRESS
 2630  ISRCE=ISOL(NSOL)-NPLOTD
C SET AXIS OR NOT (FIRST ONLY)
       IF (NSOL-1)    264,263,264
  263  AXIS=0.0
       GO TO 265
  264  AXIS=1.0
  265  ASSIGN 266 TO IPLTEX
       GO TO 290
  266  GO TO 262
C IF THERE IS DATA FOR DOTTED PLOTS, PLOT THEM ONE AT A TIME
C PUTTING AXIS ONLY ON FIRST ONE AND ONLY IF NO SOLID PLOTS WERE MADE
  270  NDOT=0
       TYPE=1.0
  272  NDOT=NDOT+1
       IF (IDOT(NDOT))    4,280,273
  273  ISRCE=IDOT(NDOT)-NPLOTD
       IF(ISOL(1))   277,275,277
  275  IF (NDOT-1)    277,276,277
  276  AXIS=0.0
       GO TO 278
  277  AXIS=1.0
  278  ASSIGN 279 TO IPLTEX
       GO TO 290
  279  GO TO 272
C SEE IF THERE IS MORE DATA YET TO PLOT   (300 IS NO)
  280  NPLOTD=NPLOTD+NOPPP
C INDEX THE PAGE NUMBER BY 1
       IPAGE=IPAGE+1
       IPAGE=XMODF(IPAGE,1000)
       IF (NPLOTD-NN)     282,300,300
C IF MORE TO PLOT SET SO FIRST POINT ON NEXT FRAME WILL BE
C SAME AS LAST POINT ON PRESENT FRAME
  282  NPLOTD=NPLOTD-1
C RESET NO OF POINTS TO BE PLOTTED ON NEXT FRAME (=NOPPP UNLESS NEXT IS
       NOPTP=NOPPP
       IF (NPLOTD+NOPPP-NN)     286,286,284
  284  NOPTP=NN-NPLOTD
C READJUST INDEX FOR FIRST BAR ON NEXT FRAME
       ITEMP1=IFRSTB
 2840  ITEMP1=ITEMP1+ISKIPB
       IF (ITEMP1-NOPPP)  2840,2841,2841
 2841  IFRSTB=ITEMP1-NOPPP+1
C INDEX THE FILM AND RETURN TO PLOT TITLE ON NEXT FRAME
  286  CALL FRAME
       GO TO 200
C THIS IS AN INTERNAL SUBROUTINE WHICH MOVES THE NEXT BLOCK OF
C DATA FROM A SPECIFIED SERIES INTO THE  SPACE BLOCK AND
C SCALES IT FOR PLOTTING. THEN IT PLOTS IT. ISRCE DEFINES DATA.
C TYPE=0.0 FOR SOLID, 1.0 DOTTED. AXIS=0.0 FOR AXIS,=1.0 NO AXIS)
  290  ISP=XLOCF(SPACE)
       CALL MVBLOK(NOPTP,ISRCE,ISP)
CGO SCALE DATA AND THEN PLOT IT
       CALL SCPSCL(SPACE,NOPTP,YTOP,YBOT,CONVK,CONVL)
       CALL HSTPLT(NOPTP,SPACE,HSTORG,NDELX,TYPE,AXIS,IFRSTB,ISKIPB)
       GO TO IPLTEX,  (266,279)
C FINAL EXIT
  300  SPACE(1)=0.0
  310  RETURN
       END
```

```
*     GRAPHX (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0153
*     LABEL
CGRAPHX
      SUBROUTINE GRAPHX(ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,YBOT,
     1                  XMAX,XMIN,NOPPP,IPAGE,SPACE,NFRMZV)
C
C                    ----ABSTRACT----
C
C TITLE - GRAPHX
C        SUBROUTINE GRAPH EXPANDED OVER ARBITRARY NO. OF VERTICAL FRAMES
C
C              GRAPHX IS FUNCTIONALLY IDENTICAL TO SUBROUTINE GRAPH,
C              EXCEPT THAT THE PLOTS ARE EXPANDED OVER AN ARBITRARILY
C              SPECIFIED NO. OF FRAMES, NFRMZV, IN THE VERTICAL
C              DIRECTION.  (THE HORIZONTAL SCALE IS UNMODIFIED.)
C
C              GRAPHX HAS 14 ARGUMENTS.  THE FIRST 13 OF THESE ARE
C              EQUIVALENT TO THE 13 ARGUMENTS OF GRAPH, AND THE 14TH
C              ARGUMENT IS NFRMZV.  THE ARGUMENT YTOP NOW REFERS TO
C              THE UPPER EDGE OF THE TOP ROW OF FRAMES, AND YBOT NOW
C              REFERS TO THE LOWER EDGE OF THE BOTTOM ROW OF FRAMES.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME AND SCOPE)
C STORAGE    - 123 REGISTERS
C SPEED      - TAKES MAXIMUM OF NFRMZV TIMES AS LONG AS GRAPH
C AUTHOR     - S.M. SIMPSON, APRIL 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - GRAPH,FRAME
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL GRAPHX(ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTOP,YBOT,
C    1             XMAX,XMIN,NOPPP,IPAGE,SPACE,NFRMZV)
C
C INPUTS
C
C    ISOL,IDOT,...,SPACE   HAVE SAME MEANING AS FOR GRAPH EXCEPT YTOP
C              AND YBOT ARE AS DESCRIBED IN ABSTRACT.
C
C    NFRMZV    IS THE DESIRED NO. OF VERTICAL FRAMES
C              MUST EXCEED ZERO AND BE LESS THAN 101
C
C OUTPUTS
C              OUTPUTS ARE SIMILAR TO THOSE OF GRAPH WITH ONE
C              ADDITIONAL ERROR FLAG.
C    SPACE(1)  = 2. IF NFRMZV IS ILLEGAL.
C
C EXAMPLES
C
C 1. SHOWING FOUR VECTORS PLOTTED ACROSS 4 FRAMES VERTICALLY.
C
C    USAGE   -        DIMENSION  Y1(600),Y2(600),Y3(600),Y4(600),ISOL(4)
C                     DIMENSION  IDOT(2),SPACE(300)
C                     PI2 = 2.0*3.14159265
C                     DO 10   I=1,600
C                     FLI = FLOATF(I)
C                     Y1(I) = .75 + .10*COSF(FLI*PI2/10.)
C                     Y2(I) = -.7 + FLI/250.
C                     Y3(I) = (1.0-FLI/650.)*COSF(FLI*PI2/150.)
C                10   Y4(I) = -.75
C                     DO 20  I=426,549
C                20   Y4(I) = -.2
C                     ISOL(1) = XLOCF(Y1)
C                     ISOL(2) = XLOCF(Y2)
C                     ISOL(3) = XLOCF(Y4)
C                     ISOL(4) = 0
C                     IDOT(1) = XLOCF(Y3)
C                     IDOT(2) = 0
C                     IPAGE = 1
C                     CALL GRAPHX(ISOL,IDOT,600,
C                    154H$$$$$$THIS IS THE TITLE AREA IN THE EXAMPLE OF G
C                    2RAPHX ,
```

```
C                      348H$$$$$$ YUNITS AREA IN THE ILLUSTRATION OF GRAPHX
C                      4,
C                      548H$$$$$$ XUNITS AREA IN THE ILLUSTRATION OF GRAPHX
C                      6,
C                      7  1.2,-.8,600.,0.,201,IPAGE,SPACE,4)
C
C    OUTPUTS - TWELVE FRAMES ARE PRODUCED WHICH WHEN CUT AND PASTED
C              TOGETHER APPEAR AS SHOWN BELOW.
C
```



```
C
C PROGRAM FOLLOWS BELOW.
C
C CHECK NFRMZV
      IF (NFRMZV)  20,20,10
 10   IF (NFRMZV-100)  30,30,20
 20   SPACE = 2.0
      GO TO 9999
C SET UP YTP,YBT, DELY FOR LOOP
 30   DELY = (YTOP-YBOT)/FLOATF(NFRMZV)
      YTP = YTOP
      YBT = YTP-DELY
C PRODUCE NFRMZV ROWS OF OUTPUT
      DO 70  I=1,NFRMZV
      CALL GRAPH(ISOL,IDOT,N,TITLE,YUNITS,XUNITS,YTP,YBT,
     1          XMAX,XMIN,NOPPP,IPAGE,SPACE)
      IF (I-NFRMZV)   60,70,70
 60   CALL FRAME
      YTP = YTP-DELY
      YBT = YBT-DELY
```

```
************************
*    GRAPHX            *
************************
               (PAGE  3)
```


```
   70   CONTINUE
  C EXIT
   9999 RETURN
        END
```

```
*     GRUP2 (SUBROUTINE)                 10/1/64   LAST CARD IN DECK IS NO. 0140
*     LABEL
CGRUP2
      SUBROUTINE GRUP2 (P,NDELX,DELX,XLO,YLIM,NWANT,IANS)
C
C                  ----ABSTRACT----
C
C TITLE - GRUP2
C       DIVIDES THE X AXIS INTO EQUALLY PROBABLE RANGES
C
C              GRUP2 PERFORMS A PROCESS KNOWN AS THE PROBABILITY
C              TRANSFORMATION WHEREBY A GIVEN PROBABILITY DENSITY IS
C              TRANSFORMED INTO A RECTANGULAR DENSITY.
C
C              THE PRINCIPAL INPUT IS A HISTOGRAM-TYPE PROBABILITY
C              DISTRIBUTION FUNCTION P(I),I=1...NDELX, WHERE P(I) =
C              PROBABILITY DENSITY FOR THE RANDOM VARIABLE X FALLING IN
C              THE I-TH RANGE OF X VALUES, WHERE ALL RANGES ARE OF EQUAL
C              LENGTH DELX, AND THE LOWEST RANGE IS FROM XLO TO XLO+DELX.
C
C              GRUP2 DIVIDES THE X AXIS INTO NWANT RANGES FROM XLO TO
C              NDELX*DELX+XLO, EACH RANGE HAVING EQUAL PROBABILITY DELP.
C              DELP=1./FLOATF(NWANT). GRUP2 RETURNS THE X VALUES
C              CORRESPONDING TO THE RANGES. THE DIVISION IS MADE BY
C              INTEGRATING THE PROBABILITY DISTRIBUTION ALONG THE X AXIS.
C              LINEAR INTERPOLATION IS MADE WHEN AN INTEGER MULTIPLE OF
C              1/NWANT LIES BETWEEN SUM UP TO J AND J+1 OF (P(I)*DELX).
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 201 REGISTERS
C SPEED     -
C AUTHOR    - J.N. GALBRAITH
C
C                  ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL GRUP2 (P,NDELX,DELX,XLO,YLIM,NWANT,IANS)
C
C INPUTS
C
C    P(I)       I=1...NDELX IS THE PROBABILITY DISTRIBUTION DEFINED
C               FROM XLO TO NDELX*DELX+XLO AND NORMALIZED SUCH THAT
C               THE SUM FROM I=1 TO NDELX OF P(I)*DELX =1.  IF P(I)
C               IS NORMALIZED SUCH THAT SUM (P(I)) LESS THAN 1. BY MORE
C               THAN .00001, AN ERROR MAY OCCUR WITH IANS=-4 .  IF P(I)
C               IS NORMALIZED SUCH THAT SUM (P(I)) GRTHN 1., THE YLIM
C               WILL BE COMPUTED IN THE USUAL MANNER WITH NORMALIZATION
C               ASSUMED = 1.0 .
C
C    XLO        IS LOWEST VALUE OF X FOR WHICH P(I) IS DEFINED.
C
C    DELX       IS THE INCREMENT IN X.
C               MUST BE GRTHN 0.
C
C    NDELX      IS THE NUMBER OF INCREMENTS.
C               MUST BE GRTHN 1.
C
C    NWANT      IS THE NUMBER OF EQUALLY LIKELY DIVISIONS WANTED.
C               MUST BE GRTHN 1.
C
C OUTPUTS
C
C    YLIM(I)    I=1...NWANT+1 IS THE VECTOR OF X VALUES WHICH
C               CORRESPOND TO EQUALLY LIKELY PROBABILITY DIVISIONS.
C               (YLIM(1)=XLO), (YLIM(NWANT+1)=XLO+FLOATF(NDELX)*DELX).
C
C    IANS       =  0  NORMAL
C               = -1  ILLEGAL NDELX
C               = -2  ILLEGAL DELX
C               = -3  ILLEGAL NWANT
```

GRUP2
(PAGE 2)

```
C              = -4  WEIRD ERROR (P PROBABLY NOT PROPERLY NORMALIZED)
C
C EXAMPLES
C
C 1. INPUTS  - ALL P=0.     NDELX=1     DELX=0.     XLO=0.     NWANT=0
C    OUTPUTS - ERROR  IANS = -1
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NDELX=20
C    OUTPUTS - ERROR  IANS= -2
C
C 3. INPUTS  - SAME AS EXAMPLE 2 EXCEPT DELX=.05  NWANT=1
C    OUTPUTS - ERROR  IANS= -3
C
C 4. INPUTS  - P(1...20) = 1.,.7,.5,1.3,2.,1.9,.6,.5,.4,.3,.2,.1,1.5,
C              1.5,1.5,1.5,1.5,.5,.5,2.     NDELX=20     DELX=.05
C              XLO=0.     NWANT=5
C    OUTPUTS - YLIM(1,...,6) = 0.,.2125,.35,.68333,.81666,1.     IANS=0
C
C 5. INPUTS  - SAME AS EXAMPLE 4. EXCEPT XLO=20.
C    OUTPUTS - YLIM(1,...,6) = 20.,20.2125,20.35,20.68333,20.81666,21.
C              IANS=0
C
C 6. INPUTS  - SAME AS EXAMPLE 5. EXCEPT DELX=.0005
C    OUTPUT  - ERROR  IANS=-4
C
C 7. INPUTS  - SAME AS EXAMPLE 5. EXCEPT DELX=100.
C    OUTPUTS - YLIM(1,...,6) = 20.,20.2,20.4,20.6,20.8,20.20   IANS=0
C
      DIMENSION P(200),YLIM(201)
C     CHECK NDELX
      IANS=-1
      IF(NDELX-1)  9999,9999,5
C     CHECK DELX
 5    IANS=-2
      IF(DELX)  9999,9999,10
 10   NUM1=NWANT-1
      IANS=-3
      IF(NUM1)  9999,9999,20
 20   YLIM(1)=XLO
      YLIM(NWANT+1)=XLO+FLOATF(NDELX)*DELX
      DELP=1./FLOATF(NWANT)
      PTEST=DELP
      ISTART=1
      SUM=0
      IANS=0
      DO 100  J=1,NUM1
      DO 50  I=ISTART,NDELX
      DELTA=P(I)*DELX
      SUM=SUM+DELTA
      IF (SUM-.99999*PTEST)  50,60,70
 50   CONTINUE
C     ERROR- USED ALL P WITHOUT FINDING ALL YLIM.
      GO TO 9777
 60   YLIM(J+1)=FLOATF(I)*DELX+XLO
      ISTART=I+1
      GO TO 90
C     INTERPOLATE
 70   SUM=SUM-DELTA
      FRACTX=(PTEST-SUM)/DELTA
      YLIM(J+1)=(FLOATF(I-1)+FRACTX)*DELX+XLO
      ISTART=I
 90   PTEST=PTEST+DELP
 100  CONTINUE
 9999 RETURN
 9777 IANS=-4
      GO TO 9999
      END
```

```
************************
*    HALVL             *
************************
 REFER TO
     DUBLX


************************
*    HALVX             *
************************
 REFER TO
     DUBLX
```


```
************************
*    HALVL             *
************************
 REFER TO
     DUBLX


************************
*    HALVX             *
************************
 REFER TO
     DUBLX
```

```
*       HLADJ  (FUNCTION)               9/29/64   LAST CARD IN DECK IS NO. 0110
*       FAP
*HLADJ
        COUNT   100
        LBL     HLADJ
        ENTRY   HLADJ F(HOL)
        ENTRY   HRADJ F(HOL)
*
*                     ----ABSTRACT----
*
* TITLE - HLADJ WITH SECONDARY ENTRY HRADJ
*       HOLLERITH LEFT ADJUST OR RIGHT ADJUST FUNCTION
*
*               HLADJ SHIFTS ITS HOLLERITH ARGUMENT LEFTWARDS UNTIL THE
*               LEADING CHARACTER IS NON-BLANK.  SPACES ARE INSERTED IN
*               POSITIONS VACATED.  NO ACTION IF ARGUMENT IS ALL SPACES.
*
*               HRADJ IS THE RIGHT SHIFTING ANALOG OF HLADJ.
*
* LANGUAGE   - FAP FUNCTIONS (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 46 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON JR.,   SEPTEMBER 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*   HOLADJ = HLADJF(HOL)
*   HOLADJ = HRADJF(HOL)
*
* INPUTS
*
*    HOL        = 6 HOLLERITH IN FORMAT(1A6)
*
* OUTPUTS
*
*    HOLADJ     = LEFT OR RIGHT ADJUSTED FORM OF HOL
*
* EXAMPLES
*
* 1. INPUTS   - HOL1 = HOL4 = 6HABCDEF,  HOL3 = HOL6 = 6H
*               HOL2 = 6H BC DE,   HOL5 = 6HAB DE ,
*    USAGE    -        H1 = HLADJF(HOL1)
*                      H2 = HLADJF(HOL2)
*                      H3 = HLADJF(HOL3)
*                      H4 = HRADJF(HOL4)
*                      H5 = HRADJF(HOL5)
*                      H6 = HRADJF(HOL6)
* OUTPUTS     - H1 = 6HABCDEF   H2 = 5HBC EF   H3 = 6H
*               H4 = 6HABCDEF   H5 = 6H AB DE   H6 = 6H
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,HLADJ
* PRINCIPAL ENTRY.  HLADJ F(HOL)
 HLADJ STZ     ZIFHL
        TRA     SETUP
* SECOND ENTRY.  HRADJ F(HOL)
 HRADJ SXA     ZIFHL,4
* FIRST SPREAD OUT THE 6 CHARACTERS, THEN BRANCH ON ENTRY
 SETUP SXD     HLADJ-2,4
        STO     HOL
        XCA
        AXT     6,4
 PXA    PXA     0,0
        LGL     6
        SLW     C+1,4
        TIX     PXA,4,1
        LDQ     HOL            RESTORE HOL
        ZET     ZIFHL
```

****
*   HLADJ   *
****
(PAGE 2)

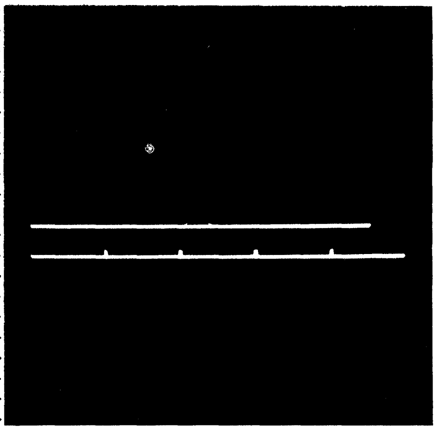
```
       TRA    RADJ
* LEFT ADJUST SEQUENCE
       AXT    6,4
 CALHL CAL    C+1,4          (GETS C1 FIRST)
       LAS    SPACE
       TRA    *+2
       TRA    RQL6
       TRA    LEAVE
 RQL6  RQL    6
       TIX    CALHL,4,1
       TRA    LEAVE
* RIGHT ADJUST SEQUENCE
 RADJ  AXT    1,4
 CALHR CAL    C+1,4          (GETS C6 FIRST)
       LAS    SPACE
       TRA    *+2
       TRA    RQL30
       TRA    LEAVE
 RQL30 RQL    30
       TXI    *+1,4,1
       TXL    CALHR,4,6
* EXIT
 LEAVE XCA                   RESULT TO AC
       LXD    HLADJ-2,4
       TRA    1,4
* CONSTANTS, TEMPORARIES
 SPACE OCT    000000000060
 ZIFHL PZE    **             **=0 IF HLADJ, NON-ZERO IF HRADJ
       PZE    **             C1  (LEFTMUST CHARACTER)
       PZE    **             C2
       PZE    **             C3
       PZE    **             C4
       PZE    **             C5
C      PZE    **             C6
 HOL   PZE    **,**,**
       END
```

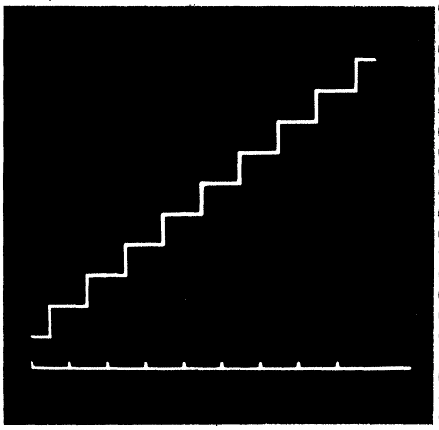
```
************************
*    HRADJ             *
************************
 REFER TO
     HLADJ
```


```
************************
*    HRADJ             *
************************
 REFER TO
     HLADJ
```

```
*       HSTPLT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0345
*       FAP
*HSTPLT
        COUNT   350
        LBL     HSTPLT
        ENTRY   HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
*                    ----ABSTRACT----
*
* TITLE - HSTPLT
*       HISTOGRAM PLOTTING FOR SUBROUTINE GRAPH
*
*              HSTPLT PLOTS THE INPUT DATA AS A HISTOGRAM OF SOLID OR
*              DOTTED LINES. A POINT IS THUS REPRESENTED AS A HORIZONTAL
*              LINE OF LENGTH NDELX/128 SCOPE UNITS. THE FIRST AND LAST
*              POINTS ARE 1/2 THIS LENGTH.  THE ENDS OF THE HORIZONTAL
*              BARS ARE CONNECTED WITH VERTICAL LINES TO MAKE THE
*              HISTOGRAM.
*
*              IF DESIRED, AN X AXIS WITH SHORT VERTICAL BARS AT
*              REGULAR INTERVALS IS PLOTTED.  THE INDEX OF THE FIRST BAR
*              AND THE SPACING OF THE BARS ARE CONTROLLED BY INPUT
*              ARGUMENTS.
*
*
* LANGUAGE   - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709, 7090 WITH SCOPE
* STORAGE    - 145 DECIMAL REGISTERS
* SPEED      - FAST (OPTIMUM)
* AUTHOR     - J.N. GALBRAITH  5/16/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - LINEH, LINEV
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL HSTPLT(LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
* INPUTS
*
*    NY(I)     I=1...LNY ARE FORTRAN II INTEGER DATA POINTS SCALED FOR
*                SCOPE PRESENTATION.
*              MUST BE GRTHN=0, LSTHN 1024
*
*    LNY       IS FORTRAN II INTEGER
*              SHOULD BE LSTHN 200 FOR GOOD RESOLUTION
*
*    ORG(I)    I=1...3 ARE FLOATING POINT NUMBERS GIVING THE X,Y
*                COORDINATES OF THE AXIS AND THE X COORDINATES OF THE
*                PLOTTED NY SERIES, ALL IN SCOPE UNITS
*              ORG(1)=LEFT X COORDINATE OF AXIS
*                       THE FIRST HORIZONTAL (HALF) BAR, CORRESPONDING
*                       TO NY(1), IS PLOTTED SO ITS LEFT EDGE HAS X
*                       COORDINATE = ORG(1)
*              ORG(2)=Y COORDINATE FOR AXIS
*              ORG(3)=RIGHT X COORDINATE OF AXIS
*
*    NDELX     THE SPACING, IN SCOPE UNITS, BETWEEN SUCCESSIVE DATA
*                POINTS MULTIPLIED BY (2**7)
*              IS FORTRAN II INTEGER
*
*    DOT       =0. SOLID LINES PLOTTED
*              NOT=0. DOTTED LINES PLOTTED
*
*    AXIS      =0. AXIS AND CROSSBARS ARE PLOTTED
*              NOT=0. NO AXIS IS PLOTTED
*
*    IFRSTB    IS THE INDEX OF THE FIRST DATA POINT FOR WHICH A
*                CROSSBAR IS PLOTTED ON THE AXIS
*              IS FORTRAN II INTEGER
*
*    ISKIPB    IS THE NUMBER OF INDICES WHICH ARE SKIPPED BETWEEN THE
*                PLOTTED CROSSBARS
```

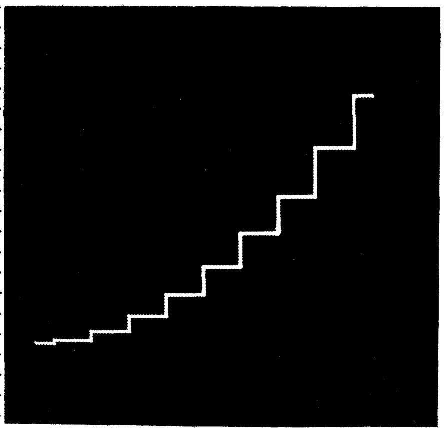
```
*              IS FORTRAN II INTEGER
*
*    OUTPUTS   DATA PLOTTED ON SCOPE
*
* EXAMPLES
*
* 1. INPUTS  - NY(1...10)=100,100,100,100,100,100,100,100,100,100
*              LNY=10  ORG(1...3)=10.,20.,1010.  NDELX=101*2**7
*              DOT=0.  AXIS=0.  IFRSTB=3  ISKIPB=2
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,180,260,340,420,500,580,660,740,820
*              IFRSTB=1  ISKIPB=1
*
*    OUTPUTS - FOR EXAMPLES 1 AND 2.
*
*
```

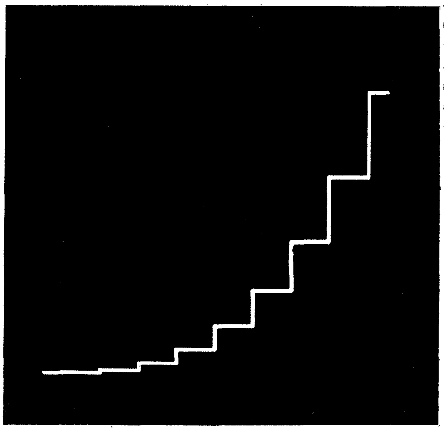




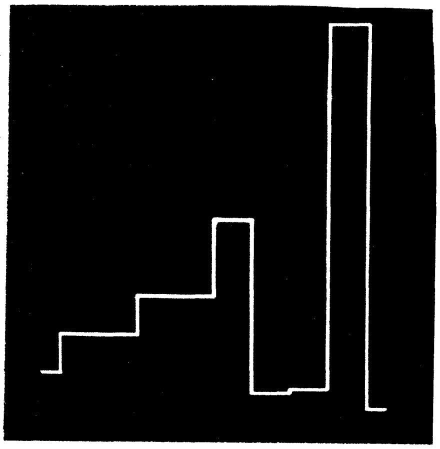
```
*
*
*
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,108,132,172,228,300,388,484,612,748
*              DOT=1.  AXIS=1.
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,101,108,127,164,225,316,443,612,829
*              AXIS=1.
*
```

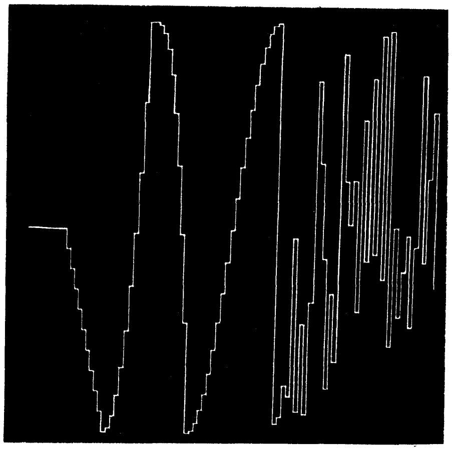
```
*    OUTPUTS - FOR EXAMPLES 3 AND 4.
*
*
```





```
*
*
*
*
* 5. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*               NY(1...10)=100,200,200,300,300,500,50,60,1000,10  AXIS=1.
*
* 6. INPUTS   - NY(1...100)=500,500,500,500,500,500,500,500,500,500,
*                           500,450,400,350,300,250,200,150,100, 50,
*                             0, 10, 40, 90,160,250,360,490,640,810,
*                          1000,999,992,973,936,875,784,657,488,271,
*                             0,  6, 25, 57, 95,146,206,273,345,421,
*                           500,579,655,727,794,854,905,943,975,994,
*                          1000,024,042,119,092,482,054,272,048,325,
*                           865,667,432,112,346,178,931,625,517,627,
*                           301,755,427,874,444,977,383,988,218,512,
*                           290,402,491,269,463,885,423,639,798,363
*               LNY=100  ORG(1..3)=10.,20.,1010.  NDELX=10*2**7
*               DOT=0.  AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 5 AND 6.
*
*
```





```
*
*
*
*
       PZE
       BCI     1,HSTPLT
HSTPLT SXA     BACK,1
```

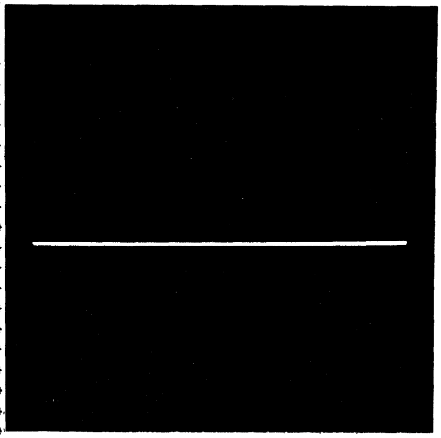
```
       SXA   BACK+1,4
       SXD   HSTPLT-2,4
       ZET*  5,4        TEST DOT
       TRA   DOT        DOTTED
       CLA   THREE      SOLID
       STO   MODE
RET    CLA   3,4        LOC ORG
       ADD   ADONE
       STA   OR1
       CLA*  1,4
       SUB   ONE
       STD   END
       ADD   ONE
       SUB*  7,4
       STD   INDEX+1
       CLA   NOP
       STO   INS
       CLA*  4,4
       ARS   7
       STO   DEL
       AXT   3,1
OR1    CLA   **,1       FIX ORG VECTOR
       UFA   CONST      COMPONENTS
       ANA   CONST+1    STORE
       ALS   18         IN
       STO   NORG+1,1   NORG TO NORG-2
       TIX   OR1,1,1
       CLA*  6,4
       TNZ   NOAX
       TSX   $LINEH,4
       PZE   NORG
       PZE   NORG-1
       PZE   NORG-2
       PZE   THREE
       CLA   NORG-1
       ADD   FIFT       FIFTEEN
       STO   BARLIM
       LXA   BACK+1,4
       CLA*  7,4        IFRSTB
       SUB   ONE
       XCA
       MPY   DEL             * DEL
       ALS   17              ((IFRSTB-1)*NDELX)/128
       ADD   NORG
       STO   NPLTX
       CLA*  8,4
       STD   INDEX
       XCA
       MPY   DEL
       ALS   17
       STO   SKIP       ISKIPB*DEL
       AXT   1,1
       TRA   START
BAR    CLA   SKIP
       ADD   NPLTX
       STO   NPLTX
START  TSX   $LINEV,4
       PZE   NPLTX
       PZE   NORG-1
       PZE   BARLIM
       PZE   THREE
INDEX  TXI   *+1,1,**
       TXL   BAR,1,**
NOAX   LXA   BACK+1,4
       CLA   DEL
       ARS   1
       STO   DEL2
       ADD   NORG
       STO   NLSTX
       CLA   NORG
       STO   NFSTX
       CLA   2,4
       STA   SUB
       STA   LDQ
       ADD   ADONE
```

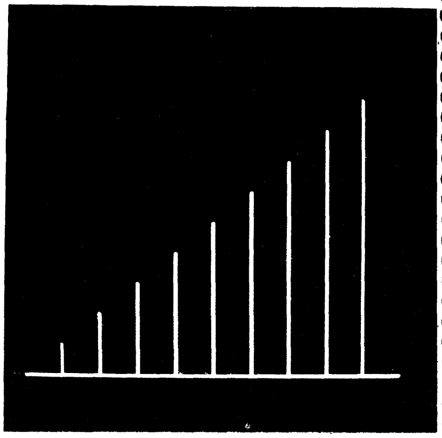
```
        STA     BEGIN
        AXT     1,1
        TRA     BEGIN
LOOP    CLA     NLSTX
        STO     NFSTX
        ADD     DEL
        STO     NLSTX
BEGIN   CLA     **,1
        STO     NYBOT
        STO     NYFST
SUB     SUB     **,1
        TZE     HOR
LDQ     LDQ     **,1
        TMI     NEXT
        CLA     NYBOT
        STO     NYTOP
        STQ     NYBOT
        TRA     VERT
NEXT    STQ     NYTOP
VERT    TSX     $LINEV,4
        PZE     NLSTX
        PZE     NYBOT
        PZE     NYTOP
        PZE     MODE
HOR     TSX     $LINEH,4
        PZE     NFSTX
        PZE     NYFST
        PZE     NLSTX
        PZE     MODE
        TXI     *+1,1,1
END     TXL     LOOP,1,**
INS     NOP
        CLA     TRA
        STO     INS
        XEC     BEGIN
        STO     NYFST
        CLA     NLSTX
        STO     NFSTX
        ADD     DEL2
        STO     NLSTX
        TRA     HOR
BACK    AXT     **,1
        AXT     **,4
        TRA     9,4
DOT     CLA     EIGHT
        STO     MODE
        TRA     RET
*       CONSTANTS AND TEMPORARY STORAGE
MODE
DEL
DEL2
ONE     PZE     0,0,1
ADONE   PZE     1,0,0
THREE   PZE     0,0,3
EIGHT   PZE     0,0,8
FIFT    PZE     0,0,15
CONST   OCT     233000000000
        OCT     000000377777
        PZE
        PZE
NORG    PZE
NYTOP   EQU     NORG
NYBOT   EQU     NORG-1
NFSTX   EQU     NORG-2
NPLTX
NLSTX   EQU     NPLTX
BARLIM
NYFST   EQU     BARLIM
SKIP    PZE     0
NOP     NOP
TRA     TRA     BACK
        END
```

```
*       HSTPLT-II (SUBROUTINE)           9/29/64   LAST CARD IN DECK IS NO. 0335
*       FAP
*HSTPLT-II
        COUNT   350
        LBL     HSTPLT
        ENTRY   HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
*                    ----ABSTRACT----
*
* TITLE - HSTPLT-II
*         BAR GRAPH PLOTTING FOR SUBROUTINE GRAPH
*
*               HSTPLT PLOTS A GRAPH OF THE INPUT DATA AS VERTICAL LINES
*               FROM A HORIZONTAL LINE.  THE Y-COORDINATE OF THE
*               HORIZONTAL LINE IS TAKEN AS THE FIRST DATA POINT.  THEN
*               ALL OTHER DATA IS PLOTTED RELATIVE TO THIS LINE.  SINCE
*               THE LAST POINT WOULD NORMALLY FALL ON THE EDGE OF THE
*               DISPLAY BOX, IT IS PLOTTED 4 SCOPE-UNITS TO THE LEFT OF
*               THE BOX BOUNDARY.
*
*               HSTPLT, AS USED BY GRAPH, LIMITS THE COMBINED LENGTH
*               OF ISOL AND IDOT TO 20 ENTRIES.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME AND SCOPE)
* STORAGE   - 188 REGISTERS
* SPEED     - FAST (OPTIMUM)
* AUTHOR    - R.A. WIGGINS,  9/5/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   LINEH, LINEV
*         AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*      CALL HSTPLT(LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
* INPUTS
*
*    NY(I)       I=1...LNY ARE FORTRAN II INTEGER DATA POINTS SCALED FOR
*                  SCOPE PRESENTATION.
*                MUST BE GRTHN=0, LSTHN 1024
*
*    LNY         IS FORTRAN II INTEGER
*                SHOULD BE LSTHN 200 FOR GOOD RESOLUTION
*
*    ORG(I)      I=1...3 ARE FLOATING POINT NUMBERS GIVING THE X,Y
*                  COORDINATES OF THE AXIS AND THE X COORDINATES OF THE
*                  PLOTTED NY SERIES, ALL IN SCOPE UNITS
*                ORG(1)=LEFT X COORDINATE OF AXIS AND FIRST NY POINT
*                ORG(2)=Y COORDINATE FOR AXIS
*                ORG(3)=RIGHT X COORDINATE OF AXIS
*
*    NDELX       THE SPACING, IN SCOPE UNITS, BETWEEN SUCCESSIVE DATA
*                  POINTS MULTIPLIED BY (2**7)
*                IS FORTRAN II INTEGER
*
*    DOT         =0. SOLID LINES PLOTTED
*                NOT=0. DOTTED LINES PLOTTED
*
*    AXIS        =0. THIS IS FIRST CURVE TO BE PLOTTED FOR THIS FRAME
*                NOT =0. THIS IS NOT THE FIRST CURVE FOR THIS FRAME
*                SET = 1. IF HSTPLT IS NOT BEING USED BY GRAPH.
*
*    IFRSTB      IS A DUMMY ARGUMENT FOR THIS HSTPLT
*
*    ISKIPB      IS A DUMMY ARGUMENT FOR THIS HSTPLT
*
* OUTPUTS        DATA PLOTTED ON SCOPE
*
* EXAMPLES
*
* 1. INPUTS   - NY(1...10)=100,100,100,100,100,100,100,100,100,100
*                LNY=10  ORG(1...3)=10.,20.,1010.  NDELX=14222
```

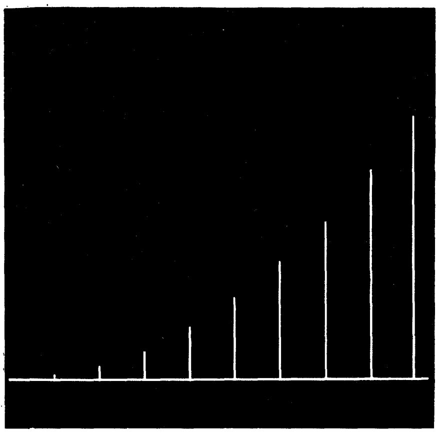
HSTPLT-II
(PAGE 2)

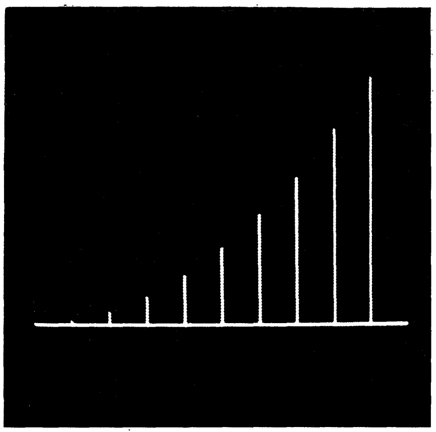
```
*              DOT=0.   AXIS=0.   IFRSTB=3  ISKIPB=2
*
* 2. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*               NY(1...10)=100,180,260,340,420,500,580,660,740,820
*               IFRSTB=1  ISKIPB=1
*
*    OUTPUTS - FOR EXAMPLES 1 AND 2.
*
```





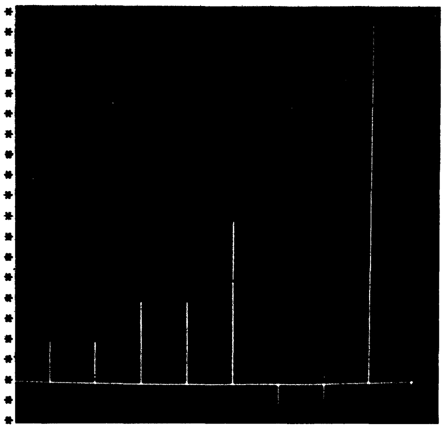
```
*
* 3. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*               NY(1...10)=100,108,132,172,228,300,388,484,612,748
*               DOT=1.  AXIS=1.
*
* 4. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*               NY(1...10)=100,101,108,127,164,225,316,443,612,829
*               AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 3 AND 4.
*
```

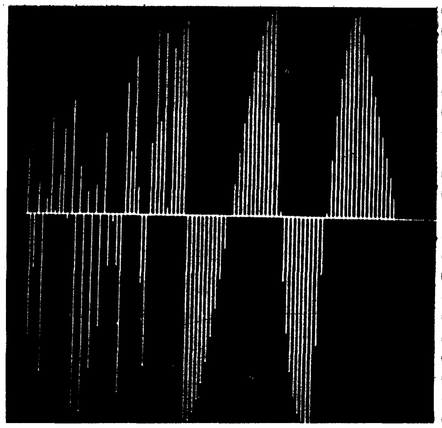




```
*
* 5. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*               NY(1...10)=100,200,200,300,300,500,50,60,1000,10  AXIS=1.
*
```

```
* 6. INPUTS   - NY(1...100)=500,500,500,500,500,500,500,500,500,500,
*                      500,450,400,350,300,250,200,150,100, 50,
*                        0, 10, 40, 90,160,250,360,490,640,810,
*                     1000,999,992,973,936,875,784,657,488,271,
*                        0,  6, 25, 57, 95,146,206,273,345,421,
*                      500,579,655,727,794,854,905,943,975,994,
*                     1000,024,042,119,092,482,054,272,048,325,
*                      865,667,432,112,346,178,931,625,517,627,
*                      301,755,427,874,444,977,383,988,218,512,
*                      290,402,491,269,463,885,423,639,798,363
*              LNY=100  ORG(1..3)=10.,20.,1010.  NDELX=1293
*              DOT=0.  AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 5 AND 6.
*
*
```





```
*
*
*
*
        PZE
        BCI     1,HSTPLT
HSTPLT  SXD     *-2,4
        SXA     ADR,1
        SXA     ADR+1,2
*       SET UP ADDRESSES
        CAL     3,4
        ADD     =1B35
        STA     ORG+1
        CAL     2,4
        ADD     =1B35
        STA     NY
        STA     NYO
*       FIX ORG(1-3)
 ORG    AXT     3,1
        CLA     **,1
        UFA     =0233000000000
        ANA     =0377777
        ALS     18
        STO     IORG+1,1
        TIX     ORG+1,1,1
*       DETERMINE WHAT IAXIS TO USE
        CLA     K
        TNZ     A1
 B1     CLA     =1B17
        STO     K
        PDX     ,1
        STZ     IFIRST
        CLA*    2,4
        STO     IAXIS
        CLA     IORG-1
        STO     YORG
        TRA     A4
 A1     CLA*    6,4
```

```
        TNZ   A2
        CLA   =1B17
        STO   IFIRST
        CLA   NYL
        SUB*  2,4
        TNZ   B1
        CLA   YORG
        SUB   IORG-1
        TNZ   B1
        CLA   =1B17
        STO   K
        TRA   A4
A2      CLA   IFIRST
        TNZ   A3
        CLA   K
        ADD   =1B17
        STO   K
        PDX   ,1
        CLA*  2,4
        STO   IAXIS+1,1
        TRA   A4
A3      CLA   K
        ADD   =1B17
        STO   K
        PDX   ,1
        CLA   NYL+1,1
        SUB*  2,4
        TZE   A4
        CLA*  2,4
        STO   IAXIS+1,1
        STZ   IFIRST
A4      CLA*  1,4
        PDX   ,2
        STD   A6+1
        SUB   =1B17
        STD   A6
*       PLOT HORIZONTAL AXIS
NY      CLA   **,2
        STO   NYL+1,1
        CLA   IAXIS+1
        PXA   ,1
        SSM
        ADD   *-3
        STA   *+3
        TSX   $LINEH,4
        TSX   IORG
        TSX   **
        TSX   IORG-2
        TSX   =4B17
*       PLOT VERTICAL LINES
        LXD   HSTPLT-2,4
        CLA*  5,4
        TZE   *+4
        CLA   =8B17
        STO   DOT
        TRA   *+3
        CLA   =4B17
        STO   DOT
        CLA   IAXIS+1,1
        STO   IA1
        CLA*  4,4
        ARS   7
        STO   NDELX
        CLA   IORG
        STO   IX
        AXT   2,2
B2      CLA   IX
        ADD   NDELX
        STO   IX
        STD   IX1
NYO     CLA   **,2
        STO   NY1
        SUB   IA1
        TZE   A6-1
        TMI   A5
```

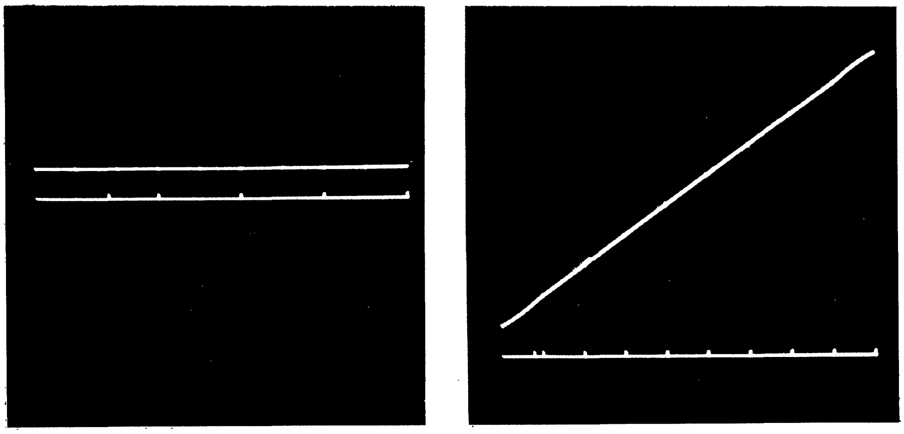
****
*   HSTPLT-II   *
****
(PAGE 5)

```
        TSX   $LINEV,4
        TSX   IX1
        TSX   IA1
        TSX   NY1
        TSX   DOT
        TRA   A6-1
 A5     TSX   $LINEV,4
        TSX   IX1
        TSX   NY1
        TSX   IA1
        TSX   DOT
        TXI   *+1,2,1
 A6     TXL   B2,2,**
        TXL   *+2,2,**
        TRA   A7
        CLA   IX
        SUB   =4B17
        STO   IX
        TRA   B2
 A7     LXD   HSTPLT-2,4
 ADR    AXT   **,1
        AXT   **,2
        TRA   9,4
        BES   3
 IORG   PZE
 K      PZE
IFIRST  PZE
 YORG   PZE
 DOT    PZE
 IA1    PZE
 NDELX  PZE
 IX     PZE
 IX1    PZE
 NY1    PZE
        BES   19
 IAXIS  PZE
        BES   19
 NYL    PZE
        END
```

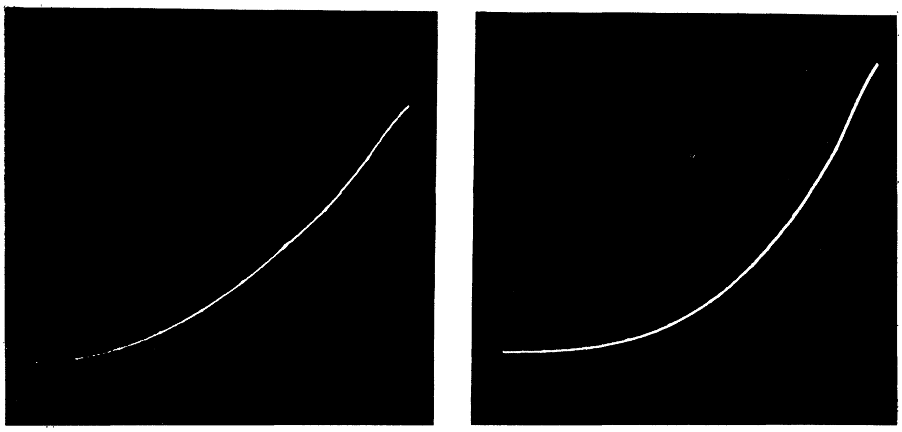
```
*      HSTPLT-III (709)(SUBROUTINE)     9/29/64   LAST CARD IN DECK IS NO. 0437
*      FAP
*HSTPLT -III (709)
       COUNT   400
       LBL     HSTPLT
       ENTRY   HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
*                    ----ABSTRACT----
*
* TITLE - HSTPLT-III (709)
*       CUBIC CURVE SCOPE PLOTTING FOR SUBROUTINE GRAPH
*
*             HSTPLT PLOTS THE INPUT DATA AS DARKENED POINTS WITH CUBIC
*             CURVES FITTED BETWEEN THE POINTS.  EXCEPT AT THE ENDS OF
*             THE DATA, THE NEAREST FOUR DATA POINTS ARE USED FOR
*             DETERMINING THE CUBIC.  AT THE END OF THE DATA SEQUENCE
*             THE NEXT POINT IS ASSUMED TO BE THE SAME AS THE FINAL
*             POINT.  IF DESIRED, AN X-AXIS WITH SHORT BARS SPACED AT
*             REGULAR INTERVALS IS PLOTTED.  THE BEGINNING POINT AND THE
*             SPACING OF THE BARS ARE CONTROLLED BY INPUT ARGUMENTS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 (MAIN FRAME AND SCOPE)
* STORAGE   - 256 REGISTERS
* SPEED     - FAST (OPTIMUM)
* AUTHOR    - R.A. WIGGINS,  9/5/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - LINEH
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL HSTPLT(LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
* INPUTS
*
*    NY(I)       I=1...LNY ARE FORTRAN II INTEGER DATA POINTS SCALED FOR
*                  SCOPE PRESENTATION.
*                MUST BE GRTHN=0, LSTHN 1024
*
*    LNY         IS FORTRAN II INTEGER
*                SHOULD BE LSTHN 200 FOR GOOD RESOLUTION
*
*    ORG(I)      I=1...3 ARE FLOATING POINT NUMBERS GIVING THE X,Y
*                  COORDINATES OF THE AXIS AND THE X COORDINATES OF THE
*                  PLOTTED NY SERIES, ALL IN SCOPE UNITS
*                ORG(1)=LEFT X COORDINATE
*                ORG(2)=Y COORDINATE FOR AXIS
*                ORG(3)=RIGHT X COORDINATE
*
*    NDELX       THE SPACING, IN SCOPE UNITS, BETWEEN SUCCESSIVE DATA
*                  POINTS MULTIPLIED BY (2**7)
*                IS FORTRAN II INTEGER
*
*    DOT         =0. SOLID LINES PLOTTED
*                NOT=0. DOTTED LINES PLOTTED
*
*    AXIS        =0. AXIS AND CROSSBARS ARE PLOTTED
*                NOT=0. NO AXIS IS PLOTTED
*
*    IFRSTB      IS THE INDEX OF THE FIRST DATA POINT FOR WHICH A
*                  CROSSBAR IS PLOTTED ON THE AXIS
*                IS FORTRAN II INTEGER
*
*    ISKIPB      IS THE NUMBER OF INDICES WHICH ARE SKIPPED BETWEEN THE
*                  PLOTTED CROSSBARS
*                IS FORTRAN II INTEGER
*
* OUTPUTS        DATA PLOTTED ON SCOPE
*
* EXAMPLES
* 1. INPUTS   - NY(1...10)=100,100,100,100,100,100,100,100,100,100
*                LNY=10  ORG(1...3)=10.,20.,1010.  NDELX=14222
```

HSTPLT -III (709)
(PAGE 2)

```
*              DOT=0.  AXIS=0.  IFRSTB=3  ISKIPB=2
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,180,260,340,420,500,580,660,740,820
*              IFRSTB=1  ISKIPB=1
*
*    OUTPUTS - FOR EXAMPLES 1 AND 2.
*
*
```

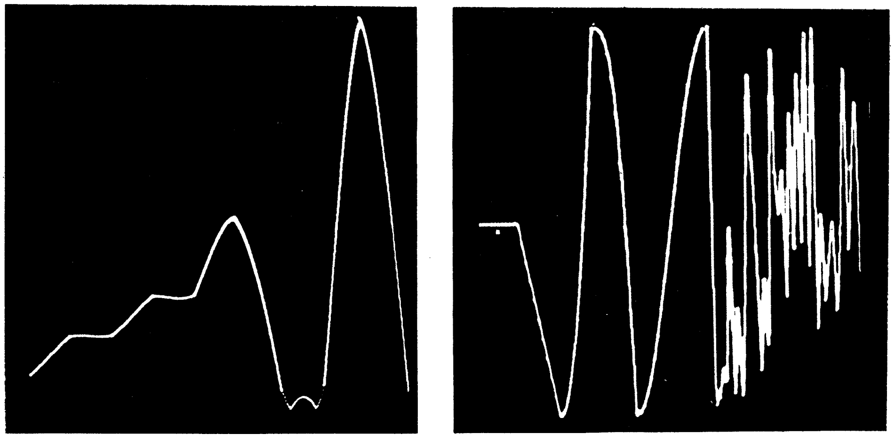


```
*
*
*
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,108,132,172,228,300,388,484,612,748
*              DOT=1.  AXIS=1.
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,101,108,127,164,225,316,443,612,829
*              AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 3 AND 4.
*
```



```
*
*
*
*
*
* 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,200,200,300,300,500,50,60,1000,10  AXIS=1.
*
```

```
* 6. INPUTS  - NY(1...100)=500,500,500,500,500,500,500,500,500,500,
*                         500,450,400,350,300,250,200,150,100, 50,
*                           0, 10, 40, 90,160,250,360,490,640,810,
*                        1000,999,992,973,936,875,784,657,488,271,
*                           0,  6, 25, 57, 95,146,206,273,345,421,
*                         500,579,655,727,794,854,905,943,975,994,
*                        1000,024,042,119,092,482,054,272,048,325,
*                         865,667,432,112,346,178,931,625,517,627,
*                         301,755,427,874,444,977,383,988,218,512,
*                         290,402,491,269,463,885,423,639,798,363
*             LNY=100  ORG(1..3)=10.,20.,1010.  NDELX=1293
*             DOT=0.  AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 5 AND 6.
*
```



```
*
*
*
*
*
       PZE
       BCI     1,HSTPLT
HSTPLT SXD     *-2,4
       SXA     ADR,1
       SXA     ADR+1,2
       CAL     3,4
       ADD     =1B35
       STA     ORG
*      FIX ORG(1-4)
       AXT     3,1
 ORG   CLA     **,1
       UFA     =0233000000000
       ANA     =0377777
       ALS     18
       STO     IORG+1,1
       TIX     ORG,1,1
       CLA*    4,4
       ARS     7
       STO     NDELX
*      DO HORIZONTAL AXIS IF AXIS=0
       CLA*    6,4
       TNZ     A2
       TSX     $LINEH,4
       TSX     IORG
       TSX     IORG-1
       TSX     IORG-2
       TSX     =4B17
       LXD     HSTPLT-2,4
       LDQ     NDELX
       MPY*    7,4
       LLS     17
       ADD     IORG
       STO     NX
       STD     NX1
```

*   HSTPLT -III (709) *
(PAGE 4)

```
        LDQ     NDELX
        MPY*    8,4
        LLS     17
        STO     NDELX1
        CLA*    1,4
        PDX     ,2
        CLA     IORG-1
        ADD     =12B17
        STD     LOOP1+4
        CLA*    8,4
        STD     A1
        LXD     IORG-1,1
        SXA     NX1,1
 LOOP1  WTV
        CPY     NX1
        TXI     *+1,1,4
        SXA     NX1,1
        TXL     LOOP1,1,**
        CLA     NX
        ADD     NDELX1
        STO     NX
        STD     NX1
 A1     TIX     LOOP1-2,2,**
*       SET UP CONSTANTS FOR MAIN LOOP
 A2     CLA*    5,4
        TNZ     *+4
        CLA     =4B17
        STO     DOT
        TRA     *+3
        CLA     =8B17
        STO     DOT
        CAL     2,4
        STA     NYADD
        ADD     =1B35
        STA     A3
        CLA*    1,4
        STD     A10+1
        ADD     =1B17
        STD     A11
        CLA     IORG
        SUB     NDELX
        STO     NX
        STO     NX1
        AXT     3,1
 A3     CLA     **,1
        STO     NY,1
        TIX     A3,1,1
        CLA     NY1
        STO     NY
*       MAIN LOOP
        AXT     3,1
 LOOP2  CLA     NY2
        SUB     NY1
        SSP
        ADD     NDELX
        LRS     35
        DVP     DOT
        XCA
        ALS     18
        DCT
        CLA     =1B17
        STO     NX2           NO. PLOTS BETWEEN EACH POINT
        CLM
        LDQ     NDELX
        LLS     18
        DVP     NX2
        STQ     DELX          SEPARATION (SCOPE UNITS) BETWEEN PLOTS
        LDQ     =0
        CLA     =1B34
        DVP     NX2
        STQ     DU
        STQ     DU1
        CLA     NY1
        STO     E3
        SSM
```

****
* HSTPLT -III (709) *
****
(PAGE 5)

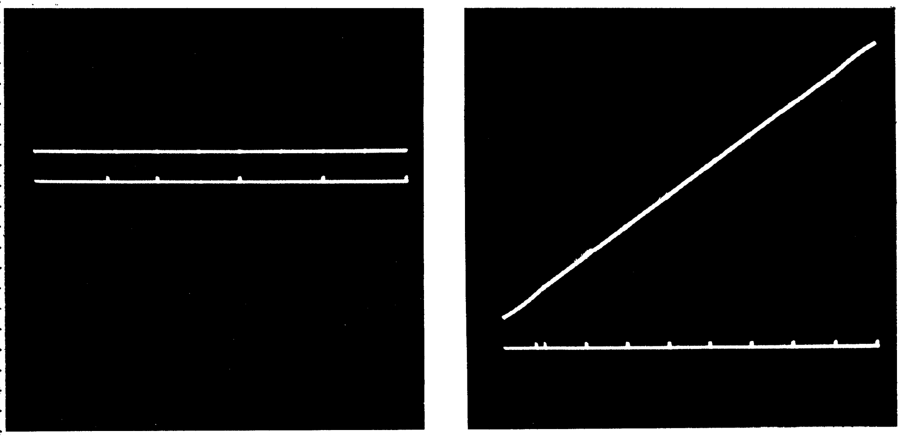
```
         ADD    NY2
         STO    E2
         SSM
         SUB    NY2
         ADD    NY3
         ARS    1
         STO    E1
         ALS    1
         SUB    NY2
         ADD    NY1
         ADD    NY1
         SUB    NY
         LDQ    =0
         LRS    17
         DVP    =6B17
         STQ    E
*        CENTRAL LOOP
         CLA    NY1
         STO    F3
         ARS    18
         STA    POINT
         CLA    NX
         ADD    NDELX
         STO    NX
         STO    NX1
         STD    POINT
         AXT    6,4
         WTV
         CPY    POINT
         TIX    *-2,4,1
         AXT    3,2
 A5      CLA    NX2
         SUB    =1B17
         TZE    A9
         STO    NX2
         CLA    DU
         SUB    =2B17
         XCA
         MPY    E
         LLS    17
         ADD    E1
         STO    A
         CLA    DU
         SUB    =1B17
         XCA
         MPY    A
         LLS    17
         ADD    E2
         XCA
         MPY    DU
         LLS    17
         ADD    E3
         STO    F+1,2
         ARS    18
         STA    POINT
         CLA    NX1
         ADD    DELX
         STO    NX1
         STD    POINT
         WTV
         CPY    POINT
         CLA    DU
         ADD    DU1
         STO    DU
         TIX    A5,2,1
         CLA    F
         STO    E3
         SUB    F1
         STO    E2
         SUB    F1
         ADD    F2
         STO    E1
         SUB    F1
         ADD    F2
         ADD    F2
```

****
*   HSTPLT -III (709) *
****
(PAGE 6)

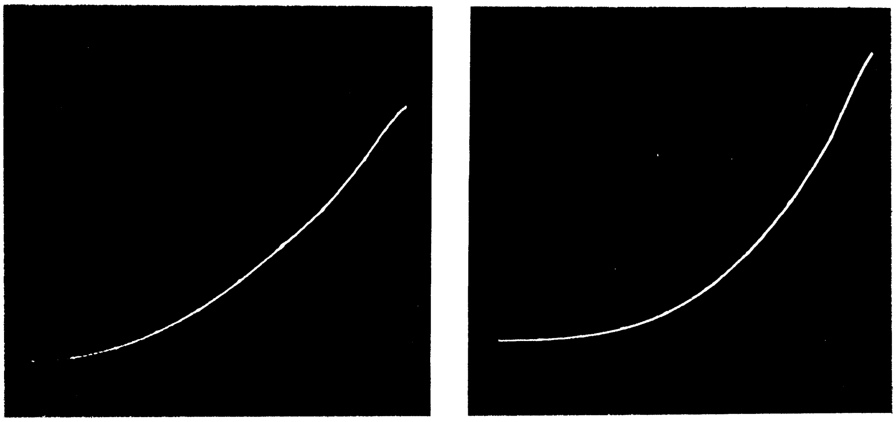
```
         SUB    F3
         STO    E
         CLA    NX2
         PDX    ,2
         SUB    =1B17
         TZE    A9
A6       CLA    E
         ADD    E1
         STO    E1
         ADD    E2
         STO    E2
         ADD    E3
         STO    E3
         PDX    ,4
         SXA    POINT,4
         CLA    NX1
         ADD    DELX
         STO    NX1
         STD    POINT
         WTV
         CPY    POINT
         TIX    A6,2,1
A9       CLA    NY1
         STO    NY
         CLA    NY2
         STO    NY1
         CLA    NY3
         STO    NY2
NYADD    CLA    **,1
         STO    NY3
A10      TXI    *+1,1,1
         TXL    LOOP2,1,**           M
         CLA    NY2
         STO    NY3
A11      TXL    LOOP2,1,**           M+1
         LXD    HSTPLT-2,4
ADR      AXT    **,1
         AXT    **,2
         TRA    9,4
NDELX    PZE
NDELX1   PZE
DELX     PZE
NX       PZE
NX1      PZE
NX2      PZE
DOT      PZE
A        PZE
POINT    PZE
F3       PZE
F2       PZE
F1       PZE
F        PZE
E3       PZE
E2       PZE
E1       PZE
E        PZE
DU       PZE
DU1      PZE
NY3      PZE
NY2      PZE
NY1      PZE
NY       PZE
         PZE
         PZE
IORG     PZE
         END
```

```
*       HSTPLT-III (7090) (SUBROUTINE)    9/8/64   LAST CARD IN DECK IS NO. 0445
*       FAP
*HSTPLT-III(7090)
        COUNT   450
        LBL     HSTPLT III
        ENTRY   HSTPLT (LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
*                    ----ABSTRACT----
*
* TITLE - HSTPLT-III (7090)
*        CUBIC CURVE SCOPE PLOTTING FOR SUBROUTINE GRAPH
*
*              HSTPLT PLOTS THE INPUT DATA AS DARKENED POINTS WITH CUBIC
*              CURVES FITTED BETWEEN THE POINTS.  EXCEPT AT THE ENDS OF
*              THE DATA, THE NEAREST FOUR DATA POINTS ARE USED FOR
*              DETERMINING THE CUBIC.  AT THE END OF THE DATA SEQUENCE
*              THE NEXT POINT IS ASSUMED TO BE THE SAME AS THE FINAL
*              POINT.  IF DESIRED, AN X-AXIS WITH SHORT BARS SPACED AT
*              REGULAR INTERVALS IS PLOTTED.  THE BEGINNING POINT AND THE
*              SPACING OF THE BARS ARE CONTROLLED BY INPUT ARGUMENTS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090 (MAIN FRAME AND SCOPE)
* STORAGE   - 258 REGISTERS
* SPEED     - FAST (OPTIMUM)
* AUTHOR    - R.A. WIGGINS,  9/5/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   LINEH
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*      CALL HSTPLT(LNY,NY,ORG,NDELX,DOT,AXIS,IFRSTB,ISKIPB)
*
* INPUTS
*
*    NY(I)      I=1...LNY ARE FORTRAN II INTEGER DATA POINTS SCALED FOR
*                 SCOPE PRESENTATION.
*               MUST BE GRTHN=0, LSTHN 1024
*
*    LNY        IS FORTRAN II INTEGER
*               SHOULD BE LSTHN 200 FOR GOOD RESOLUTION
*
*    ORG(I)     I=1...3 ARE FLOATING POINT NUMBERS GIVING THE X,Y
*                 COORDINATES OF THE AXIS AND THE X COORDINATES OF THE
*                 PLOTTED NY SERIES, ALL IN SCOPE UNITS
*               ORG(1)=LEFT X COORDINATE
*               ORG(2)=Y COORDINATE FOR AXIS
*               ORG(3)=RIGHT X COORDINATE
*               0.0 LSTHN= ORG(I) LSTHN= 1023.0
*
*    NDELX      THE SPACING, IN SCOPE UNITS, BETWEEN SUCCESSIVE DATA
*                 POINTS MULTIPLIED BY (2**7)
*               IS FORTRAN II INTEGER
*
*    DOT        =0. SOLID LINES PLOTTED
*               NOT=0. DOTTED LINES PLOTTED
*
*    AXIS       =0. AXIS AND CROSSBARS ARE PLOTTED
*               NOT=0. NO AXIS IS PLOTTED
*
*    IFRSTB     IS THE INDEX OF THE FIRST DATA POINT FOR WHICH A
*                 CROSSBAR IS PLOTTED ON THE AXIS
*               IS FORTRAN II INTEGER
*
*    ISKIPB     IS THE NUMBER OF INDICES WHICH ARE SKIPPED BETWEEN THE
*                 PLOTTED CROSSBARS
*               IS FORTRAN II INTEGER
*
*    OUTPUTS    DATA PLOTTED ON SCOPE
*
* EXAMPLES
* 1. INPUTS  - NY(1...10)=100,100,100,100,100,100,100,100,100,100
*               LNY=10  ORG(1...3)=10.,20.,1010.  NDELX=14222
```

```
*              DOT=0.   AXIS=0.   IFRSTB=3   ISKIPB=2
*
* 2. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,180,260,340,420,500,580,660,740,820
*              IFRSTB=1  ISKIPB=1
*
*    OUTPUTS - FOR EXAMPLES 1 AND 2.
*
*
*
```

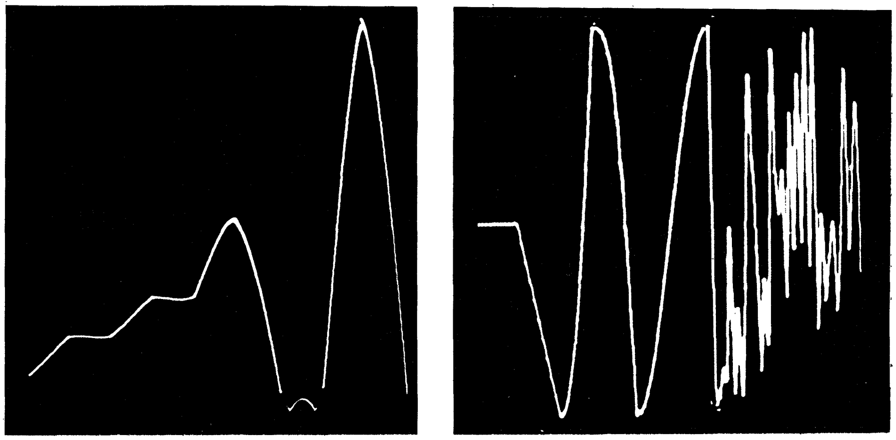


```
*
*
*
* 3. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,108,132,172,228,300,388,484,612,748
*              DOT=1.   AXIS=1.
*
* 4. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,101,108,127,164,225,316,443,612,829
*              AXIS=1.
*
*    OUTPUTS - FOR EXAMPLES 3 AND 4.
*
*
*
```



```
*
*
*
*
* 5. INPUTS   - SAME AS EXAMPLE 1. EXCEPT
*              NY(1...10)=100,200,200,300,300,500,50,60,1000,10   AXIS=1.
*
```

```
* 6. INPUTS   - NY(1...100)=500,500,500,500,500,500,500,500,500,500,
*                            500,450,400,350,300,250,200,150,100, 50,
*                              0, 10, 40, 90,160,250,360,490,640,810,
*                           1000,999,992,973,936,875,784,657,488,271,
*                              0,  6, 25, 57, 95,146,206,273,345,421,
*                            500,579,655,727,794,854,905,943,975,994,
*                           1000,024,042,119,092,482,054,272,048,325,
*                            865,667,432,112,346,178,931,625,517,627,
*                            301,755,427,874,444,977,383,988,218,512,
*                            290,402,491,269,463,885,423,639,798,363
*               LNY=100  ORG(1..3)=10.,20.,1010.  NDELX=1293
*               DOT=0.  AXIS=1.
*
*     OUTPUTS - FOR EXAMPLES 5 AND 6.
*
```



```
*
*
*
*
*
        BSS     0
* FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
* TO CHANGE, ALTER THE LETTER DESIGNATION ONLY AND REASSEMBLE.
 X      TAPENO  D1
 SCPAD  EQU     X-105
        PZE
        BCI     1,HSTIII
HSTPLT  SXD     *-2,4
        SXA     ADR,1
        SXA     ADR+1,2
        CAL     3,4
        ADD     =1B35
        STA     ORG
*       FIX ORG(1-4)
        AXT     3,1
 ORG    CLA     **,1
        UFA     =0233000000000
        ANA     =0377777
        ALS     18
        STO     IORG+1,1
        TIX     ORG,1,1
        CLA*    4,4
        ARS     7
        STO     NDELX
*       DO HORIZONTAL AXIS IF AXIS=0
        CLA*    6,4
        TNZ     A2
        TSX     $LINEH,4
        TSX     IORG,0
        TSX     IORG-1,0
        TSX     IORG-2,0
        TSX     =4B17,0
        LXD     HSTPLT-2,4
        LDQ     NDELX
```

```
        MPY*    7,4
        LLS     17
        ADD     IORG
        STO     NX
        STD     NX1
        LDQ     NDELX
        MPY*    8,4
        LLS     17
        STO     NDELX1
        CLA*    1,4
        PDX     ,2
        CLA     IORG-1
        ADD     =12B17
        STD     LOOP1+4
        CLA*    8,4
        STD     A1
        LXD     IORG-1,1
        SXA     NX1,1
 LOOP1  WRS     SCPAD
        RCHX    IO1
        TXI     *+1,1,4
        SXA     NX1,1
        TXL     LOOP1,1,**
        CLA     NX
        ADD     NDELX1
        STO     NX
        STD     NX1
 A1     TIX     LOOP1-2,2,**
*       SET UP CONSTANTS FOR MAIN LOOP
 A2     CLA*    5,4
        TNZ     *+4
        CLA     =4B17
        STO     DOT
        TRA     *+3
        CLA     =8B17
        STO     DOT
        CAL     2,4
        STA     NYADD
        ADD     =1B35
        STA     A3
        CLA*    1,4
        STD     A10+1
        ADD     =1B17
        STD     A11
        CLA     IORG
        SUB     NDELX
        STO     NX
        STO     NX1
        AXT     3,1
 A3     CLA     **,1
        STO     NY,1
        TIX     A3,1,1
        CLA     NY1
        STO     NY
*       MAIN LOOP
        AXT     3,1
 LOOP2  CLA     NY2
        SUB     NY1
        SSP
        ADD     NDELX
        LRS     35
        DVP     DOT
        XCA
        ALS     18
        DCT
        CLA     =1B17
        STO     NX2            NO. PLOTS BETWEEN EACH POINT
        CLM
        LDQ     NDELX
        LLS     18
        DVP     NX2
        STQ     DELX           SEPARATION (SCOPE UNITS) BETWEEN PLOTS
        LDQ     =0
        CLA     =1B34
        DVP     NX2
```

```
************************
*    HSTPLT-III(7090)   *
************************
                (PAGE  5)
```


```
          STQ     DU
          STQ     DU1
          CLA     NY1
          STO     E3
          SSM
          ADD     NY2
          STO     E2
          SSM
          SUB     NY2
          ADD     NY3
          ARS     1
          STO     E1
          ALS     1
          SUB     NY2
          ADD     NY1
          ADD     NY1
          SUB     NY
          LDQ     =0
          LRS     17
          DVP     =6B17
          STQ     E
*         CENTRAL LOOP
          CLA     NY1
          STO     F3
          ARS     18
          STA     POINT
          CLA     NX
          ADD     NDELX
          STO     NX
          STO     NX1
          STD     POINT
          AXT     6,4
          WRS     SCPAD
          RCHX    IO2
          TIX     *-2,4,1
          AXT     3,2
 A5       CLA     NX2
          SUB     =1B17
          TZE     A9
          STO     NX2
          CLA     DU
          SUB     =2B17
          XCA
          MPY     E
          LLS     17
          ADD     E1
          STO     A
          CLA     DU
          SUB     =1B17
          XCA
          MPY     A
          LLS     17
          ADD     E2
          XCA
          MPY     DU
          LLS     17
          ADD     E3
          STO     F+1,2
          ARS     18
          STA     POINT
          CLA     NX1
          ADD     DELX
          STO     NX1
          STD     POINT
          WRS     SCPAD
          RCHX    IO2
          CLA     DU
          ADD     DU1
          STO     DU
          TIX     A5,2,1
          CLA     F
          STO     E3
          SUB     F1
          STO     E2
          SUB     F1
```

```
          ADD    F2
          STO    E1
          SUB    F1
          ADD    F2
          ADD    F2
          SUB    F3
          STO    E
          CLA    NX2
          PDX    ,2
          SUB    =1B17
          TZE    A9
   A6     CLA    E
          ADD    E1
          STO    E1
          ADD    E2
          STO    E2
          ADD    E3
          STO    E3
          PDX    ,4
          SXA    POINT,4
          CLA    NX1
          ADD    DELX
          STO    NX1
          STD    POINT
          WRS    SCPAD
          RCHX   IO2
          TIX    A6,2,1
   A9     CLA    NY1
          STO    NY
          CLA    NY2
          STO    NY1
          CLA    NY3
          STO    NY2
   NYADD  CLA    **,1
          STO    NY3
   A10    TXI    *+1,1,1
          TXL    LOOP2,1,**              M
          CLA    NY2
          STO    NY3
   A11    TXL    LOOP2,1,**              M+1
          LXD    HSTPLT-2,4
   ADR    AXT    **,1
          AXT    **,2
          TRA    9,4
   IO1    IOCD   NX1,,1
   IO2    IOCD   POINT,,1
   NDELX  PZE
  NDELX1  PZE
   DELX   PZE
   NX     PZE
   NX1    PZE
   NX2    PZE
   DOT    PZE
   A      PZE
   POINT  PZE
   F3     PZE
   F2     PZE
   F1     PZE
   F      PZE
   E3     PZE
   E2     PZE
   E1     PZE
   E      PZE
   DU     PZE
   DU1    PZE
   NY3    PZE
   NY2    PZE
   NY1    PZE
   NY     PZE
          PZE
          PZE
   IORG   PZE
          END
```

```
*      HVTOIV (SUBROUTINE)            9/29/64   LAST CARD IN DECK IS NO. 0109
*      FAP
*HVTOIV
       COUNT   100
       LBL     HVTOIV
       ENTRY   HVTOIV (HV,LHV,IV)
*
*                ----ABSTRACT----
*
* TITLE - HVTOIV
*       SPREAD OUT HOLLERITH VECTOR AS FORTRAN INTEGERS
*
*             HVTOIV SPREADS OUT A VECTOR HV(I), I=1...LHV, AS A
*             FORTRAN INTEGER VECTOR IV(I), I=1...6*LHV. EACH REGISTER
*             OF HV(I) IS ASSUMED TO BE IN FORMAT(A6) AND IS SPREAD
*             OUT AS 6 INTEGERS.  THE INTEGER VALUES WILL LIE IN THE
*             RANGE +0 TO +63.
*
*             HVTOIV IS THE INVERSE OF SUBROUTINE IVTOHV
*
* LANGUAGE  - FAP  SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLU)
* STORAGE   - 39 REGISTERS
* SPEED     - 84*LHV MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON, MARCH 1963
*
*                ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL HVTOIV(HV,LHV,IV)
*
* INPUTS
*
*    HV(I)      I=1...LHV  IS HOLLERITH VECTOR IN A6 FORMAT
*
*    LHV        MUST EXCEED 0  (STRAIGHT EXIT FOR ILLEGAL LHV)
*
* OUTPUTS
*
*    IV(I)      I=1...6*LHV  IS THE INTEGER VECTOR EQUIVALENT TO HV(I)
*
* EXAMPLES
*
* 1. INPUTS  - HV(1) = 6HCHARAC  (= OCT233021512123)
*              HV(2) = 6HTERS T  (= OCT632551526063)
*              HV(3) = 6HO SPRE  (= OCT466062475125)
*              HV(4) = 6HAD OUT  (= OCT212460466463)
*              HV(5) = 5HIN IV   (= OCT314560315460)
*
*    USAGE   -        DIMENSION  HV(5),IV1(30),IV2(6),IV3(6)
*                     CALL HVTOIV(HV,5,IV1)
*                     CALL HVTOIV(HV(5),1,IV2)
*                     CALL HVTOIV(HV,0,IV3)
*    OUTPUTS - IV1(1...30) = 19,24,17,41,17,19,51,21,41,50,
*                            48,51,38,48,50,39,41,21,17,20,
*                            48,38,52,51,25,37,48,25,53,48
*              IV2(1...6)  = 25,37,48,25,53,48
*              IV3(I) IS NOT CHANGED (ILLEGAL LHV)
*
* PROGRAM FOLLOWS BELOW
       HTR     0
       HTR     0
       HTR     0
       BCI     1,HVTOIV
HVTOIV SXD     HVTOIV-2,4
       SXD     HVTOIV-3,2
       SXD     HVTOIV-4,1
* SETUP SEQUENCE
       CLA     1,4                A(HV)
       ADD     K1
       STA     GET
       CLA*    2,4                LHV
```

****
* HVTOIV *
****
(PAGE 2)

```
        TMI    LEAVE
        TZE    LEAVE
        STD    TESTLH
        CLA    3,4             A(IV)
        ADD    K1
        STA    STO
        CLA    KD6
        STD    TEST6
* (XR1 CONTROLS ACQUISITION, XR2 CONTROLS STORAGE)
        AXT    1,1
        AXT    1,2
* GET NEXT HOLLERITH
 GET    LDQ    **,1            **=A(HV)+1
* SHIFT AND STORE LOOP
 SHIFT  CLA    KD0
        LGL    6
        ALS    18
 STO    STO    **,2            **=A(IV)+1
        TXI    *+1,2,1
 TEST6  TXL    SHIFT,2,**      **=6,12,...
* BUMP DECREMENT OF TESTL AND INDEX 1. TEST FINISH
        CAL    TEST6
        ACL    KD6
        SLW    TEST6
        TXI    *+1,1,1
TESTLH  TXL    GET,1,**        **=LHV
* EXIT
 LEAVE  LXD    HVTOIV-3,2
        LXD    HVTOIV-4,1
        TRA    4,4
* CONSTANTS
 KD0    PZE    0
 KD6    PZE    0,0,6
 K1     PZE    1
        END
```

```
*      IDERIV (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0148
*      FAP
*IDERIV
       COUNT   150
       LBL     IDERIV
       ENTRY   IDERIV (YOFX1,DYDX,DELX,LY,YOFX)
*
*                     ----ABSTRACT----
*
* TITLE - IDERIV
*        INVERSION OF DIFFERENTIATION BY DIFFERENCING
*
*              IDERIV PERFORMS THE INVERSE OPERATION TO THAT OF
*              SUBROUTINE DERIVA, I.E. IT FINDS A VECTOR, YOFX,
*              WHOSE DERIVATIVE BY DIFFERENCING IS A GIVEN VECTOR,
*              DYDX.  THE INITIAL VALUE OF YOFX IS REQUIRED AS INPUT.
*
*              THE OUTPUT VECTOR YOFX MAY REPLACE THE INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 54   REGISTERS
* SPEED      -  7090        709      7090        709
*              (66.0  OR  70.6) + (25.4  OR  28.6)*LY  MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL IDERIV(YOFX1,DYDX,DELX,LY,YOFX)
*
*                 NOTE THAT THE ARGUMENTS ARE IDENTICAL TO THOSE OF
*                 SUBROUTINE DERIVA EXCEPT THAT THE ORDER IS REVERSED.
*                 SEE SUBROUTINE DERIVA FOR DETAILED DISCUSSION.
*
* INPUTS
*
*    YOFX1     IS STARTING VALUE FOR YOFX(1)
*
*    DYDX(I)   I=1...LY  IS THE DERIVATIVE AS PRODUCED BY DERIVA
*
*    DELX      WAS THE (NON-ZERO) DELTA X USED IN OBTAINING DYDX
*
*    LY        MUST EXCEED 1
*
* OUTPUTS      STRAIGHT RETURN WITH NO ACTION IF LY LSTHN 2 OR DELX *0.
*
*    YOFX(I)   I=1...LY  IS THE INTEGRATED FORM OF DYDX, WHERE
*                  YOFX(1) =  YOFX1
*                  YOFX(2) = DELX*DYDX(1)  + YOFX(1)
*                  YOFX(K) =  2*DELX*DYDX(K-1) +  YOFX(K-2)
*                        FOR K = 3,4,...,LY
*
*              EQUIVALENCE(YOFX,DYDX) IS PERMITTED.
*
* EXAMPLES     THESE EXAMPLES ARE THE INVERSES OF THE EXAMPLES USED
*              FOR DERIVA.
*
* 1. INPUTS  - D1(1...5) =  4., 6., 6., 2., 0.
*              D2(1...5) = -2.,-3.,-3.,-1., 0.
*              D3(1...2) =  4., 4.
*              Y4 = Y5 = -999.
*    USAGE   -       CALL IDERIV( 2., D1,  1., 5, Y1)
*                    CALL IDERIV( 2., D2, -2., 5, Y2)
*                    CALL IDERIV( 2., D3,  1., 2, Y3)
*                    CALL IDERIV( 2., D1,  1., 1, Y4)
*                    CALL IDERIV( 2., D1,  0., 5, Y5)
*    OUTPUTS - Y1(1...5) = Y2(1...5) =  2., 6., 14., 18., 18.
*              Y3(1...2) =  2., 6.    Y4=Y5 = -999. (NO OUTPUT CASES)
*
* 2. MULTIPLE INTEGRATION WITH OUTPUTS REPLACING INPUTS
*    INPUTS  - DY(1...6) =  2., 0., -3., 0., 4., 4.
```

IDERIV (PAGE 2)

```
*               FY(1...3) =  0., 4.,  4.   (NOTE REVERSAL OF ORDER FROM
*                                           THAT IN EXAMPLE OF DERIVA)
*    USAGE    -        DO  10   I=1,3
*                10   CALL IDERIV(FY(I), DY, 1., 6, DY)
*    OUTPUTS - DY(1...6) =  4., 8., 12., 24., 20., 24.
*
* PROGRAMS FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0             XR4
       BCI     1,IDERIV
* ONLY ENTRY. IDERIV(YOFX1, DYDX, DELX, LY, YOFX)
IDERIV SXD     IDERIV-2,4
* CHECK LY (GRTHN=2) AND DELX (NON-ZERO)
       CLA*    4,4           LY
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,1
       SXD     TXL,4
       LXD     IDERIV-2,4
       CLA*    3,4           DELX
       TZE     LEAVE
* OK, SETUP
       XCA
       FMP     FL2
       STO     TWODX         2*DELX
       CLA     2,4           A(DYDX)
       SUB     K1            A(DYDX)-1
       STA     GET2
       ADD     K2            A(DYDX)+1
       STA     GET
       CLA     5,4
       SUB     K1            A(YOFX)-1
       STA     STO2
       ADD     K2            A(YOFX)+1
       STA     STORE
       ADD     K2            A(YOFX)+3
       STA     FAD
* FORM AND SET YOFX(1...2)
       LDQ*    2,4           DYDX(1)
       FMP     TWODX
       FDP     FL2           DELX*DYDX(1)
       STQ     OLDDY         SET ASIDE.
       CLA*    1,4                               YOFX1
       STO*    5,4                               IS YOFX(1).
       FAD     OLDDY                             PLUS DELX*DYDX(1)
 GET2  LDQ     **            **=A(DYDX)-1
 STO2  STO     **            **=A(YOFX)-1        IT BECOMES YOFX(2).
       STQ     OLDDY         SAVE DYDX(2) FOR LOOP
* EXIT IF LY=2
       LXD     TXL,4
       TXL     LEAVE,4,2
* OTHERWISE PROCEED TO LOOP
       AXT     3,4
* LOOP TO SET YOFX(3,4,...K,...LY)    K IN XR4
 GET   CLA     **,4          **=A(DYOFX)+1       DYOFX(K)
       LDQ     OLDDY
       STO     OLDDY                             SET ASIDE.
       FMP     TWODX                             2*DELX*DYOFX(K-1)
 FAD   FAD     **,4          **=A(YOFX)+3        PLUS YOFX(K-2)
 STORE STO     **,4          **=(YOFX)+1         IS YOFX(K).
       TXI     *+1,4,1
 TXL   TXL     GET,4,**      **=LY
* EXIT
 LEAVE LXD     IDERIV-2,4
       TRA     6,4
* CONSTANTS, TEMPORARIES
 FL2   DEC     2.0
 K1    PZE     1
 K2    PZE     2
 TWODX PZE     **,**,**      2*DELX
 OLDDY PZE     **,**,**      DYDX(K-1)           STARTS AT DYDX(2)
       END
```

```
************************
*    IF (PSEUDO ENTRY) *
************************
 REFER TO
     SEVRAL
```


```
************************
*    IF (PSEUDO ENTRY) *
************************
 REFER TO
     SEVRAL
```

```
*     IFNCTN (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0443
*     FAP
*IFNCTN
      COUNT   400
      LBL     IFNCTN
      ENTRY   IFNCTN (YOFX, LYOFX, XFIRST, XLAST, LXOFY, YLO, YHI,
*                     IERRLO, XOFY, IANS)
*
*
*                    ----ABSTRACT----
*
* TITLE - IFNCTN
*        INVERSION OF A MONOTONE FUNCTION BY LINEAR INTERPOLATION
*
*              IFNCTN  TAKES A MONOTONELY INCREASING  (NON-DECREASING)
*              OR MONOTONELY DECREASING  (NON-INCREASING)  SET OF
*              FUNCTION VALUES
*
*                 Y(X)  FOR  X = X1, X1+DX, X1+2DX, ..., X2=X1+(LY-1)DX
*
*              AND PRODUCES, BY LINEAR INTERPOLATION, A SET OF FUNCTION
*              VALUES
*
*                 X(Y)  FOR  Y = YLO, YLO+DY, ..., YHI=YLO+(LX-1)DY
*
*              WHERE THE PROGRAM INPUTS ARE  Y(X),LY,X1,X2,LX,YLO, AND
*              YHI.  IF  Y(X)  HAS FLAT AREAS WHOSE HEIGHTS ARE IN THE
*              LIST OF ARGUMENTS OF  X(Y),  THEN THE VALUES CHOSEN FOR
*              X  ARE THE MIDPOINTS OF SUCH AREAS.
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094  (MAIN FRAME ONLY)
* STORAGE    - 208 REGISTERS
* SPEED      - IF  Y(X)  IS MONOTONE INCREASING,  IFNCTN  TAKES ABOUT
*                   280 + 25 LY + 70 LX  MACHINE CYCLES ON THE 7090,
*              AND IF  Y(X)  IS MONOTONE DECREASING,
*                   410 + 37 LY + 70 LX  MACHINE CYCLES
*              WITH  LY  AND  LX  AS DEFINED IN ABSTRACT.
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - MONOCK, REVER
*       AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     CALL IFNCTN(YOFX, LYOFX, XFIRST, XLAST, LXOFY, YLO, YHI, IERRLO,
*    1            XOFY, IANS)
*
*
* INPUTS
*
*    YOFX(I)   I=1...LYOFX  IS THE SET OF VALUES  Y(X)  OF THE ABSTRACT.
*              MUST BE EITHER NON-DECREASING OR NON-INCREASING.
*
*    LYOFX     IS THE QUANTITY  LY  OF THE ABSTRACT.
*              MUST BE  GRTHN=  2 .
*
*    XFIRST    IS THE ARGUMENT  X1  OF THE ABSTRACT, I.E.,  Y(XFIRST) =
*                 YOFX(1).
*
*    XLAST     IS THE ARGUMENT  X2  OF THE ABSTRACT, I.E.,  Y(XLAST) =
*                 YOFX(LYOFX).
*              XLAST MUST NOT  =  XFIRST  (BUT MAY BE LESS THAN  XLAST).
*
*    LXOFY     IS THE ARGUMENT  LX  OF THE ABSTRACT.
*              MUST BE  GRTHN=  1 .
*
*    YLO       IS GIVEN IN THE ABSTRACT.  LET  YMIN = MINIMUM(YOFX(I))
*                 AND  YMAX = MAXIMUM(YOFX(I)).  THEN  YLO  MUST SATISFY
*                 YMIN  LSTHN=  YLO  LSTHN  YMAX  FOR  LXOFY  GRTHN  1,
*                 OR  YMIN  LSTHN=  YLO  LSTHN=  YMAX  FOR  LXOFY = 1 .
*
```

```
*    YHI       IS GIVEN IN ABSTRACT, BUT IT IS NOT REFERRED TO IF
*                 LXOFY  =  1 .  IF  LXOFY  GRTHN=  2,  THEN  YHI  MUST
*                 SATISFY  YLO  LSTHN  YHI  LSTHN=  YMAX.
*
*    IERRLO    IS THE DESIRED ERROR VALUE FOR  IANS  IF  YOFX  IS
*                 ILLEGAL (NON-MONOTONE).
*              SHOULD EXCEED ZERO.
*
*
* OUTPUTS
*
*    YOFX(I)   OUTPUT IS SAME AS INPUT EXCEPT THAT THE SIGN BITS OF ALL
*                 ZERO MAGNITUDE ELEMENTS, IF ANY, WILL BE MADE POSITIVE
*                 IF NOT ALREADY (RESULT OF ACTION BY SUBROUTINE
*                 MONOCK).
*
*    XOFY(I)   I=1...LXOFY  IS THE INVERSE FUNCTION,  COMPUTED ONLY IF
*                 IANS  =  0 .
*
*    IANS      =  0             IF ALL OK
*              =  IERRLO        IF YOFX  ILLEGAL
*              =  IERRLO+1      IF LYOFX ILLEGAL
*              =  IERRLO+3      IF XLAST ILLEGAL
*              =  IERRLO+4      IF LXOFY ILLEGAL
*              =  IERRLO+5      IF YLO   ILLEGAL
*              =  IERRLO+6      IF YHI   ILLEGAL
*
*
* EXAMPLES
*
* 1. MONOTONE INCREASING CASE
*    INPUTS  - YOFX(1...9) = 1.,2.,3.,4.,4.,4.,4.,5.,6.     LYOFX=9
*              XFIRST=1.0  XLAST=9.0  LXOFY=5  YLO=1.5  YHI=5.5
*              IERRLO=1
*    USAGE   -       CALL IFNCTN(YOFX, LYOFX, XFIRST, XLAST, LXOFY,
*                  1             YLO, YHI, IERRLO, XOFY, IANS)
*    OUTPUTS - XOFY(1...5) = 1.5,2.5,3.5,7.5,8.5  IANS=0
*
* 2. MONOTONE DECREASING CASE
*    INPUTS  - SAME AS EXAMPLE 1., EXCEPT
*              YOFX(1...9) = 6.,5.,4.,4.,4.,4.,3.,2.,1.
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - XOFY(1...5) = 8.5,7.5,6.5,2.5,1.5  IANS=0
*
* 3. NEGATIVE DELTA X CASE
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  XFIRST = 9.0  XLAST = 1.0
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - XOFY(1...5) = 8.5,7.5,6.5,2.5,1.5  IANS = 0
*
* 4. MONOTONE DECREASING AND NEGATIVE DELTA X CASE
*    INPUTS  - SAME AS EXAMPLE 1. BUT WITH THE EXCEPTIONS OF BOTH
*              EXAMPLES 2. AND 3.
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - SAME AS EXAMPLE 1.
*
* 5. SPECIAL CASES WITH  LXOFY = 1,2,3  INVOLVING A FLAT AREA
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  YLO=2.0  YHI=6.0  AND THE
*              OUTPUT  XOFY  IS AN ARRAY WITH  DIMENSION  XOFY(3,3)  ALL
*              OF WHOSE ELEMENTS ARE INITIALLY SET  =  -99.
*    USAGE   -       DO 10  LXY=1,3
*               10   CALL IFNCTN(YOFX, LYOFX, XFIRST, XLAST, LXY, YLO,
*                  1             YHI, IERRLO, XOFY(1,LXY), IANS(LXY))
*    OUTPUTS - XOFY(1...3,1) = 2.0,-99.,-99.
*              XOFY(1...3,2) = 2.0,9.0,-99.
*              XOFY(1...3,3) = 2.0,5.5,9.
*              IANS(1...3)   = 0,0,0
*
* 6. SPECIAL CASE WITH  LYOFX = 2
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LYOFX=2  YLO=1.0  YHI=2.0
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - XOFY(1...5) = 1.0,3.0,5.0,7.0,9.0  IANS=0
*
* 7. ILLEGAL CASES
*    INPUTS  - YOFX2(1...3) = 0.,20.,10.     YOFX SAME AS EXAMPLE 1.
*    USAGES  -       CALL IFNCTN(YOFX2,3,1.,2.,2,1.,2.,1,XOFY,IANS1)
```

* IFNCTN *
****
(PAGE 3)

```
     *                    CALL IFNCTN(YOFX,1,1.,2.,2,1.,2.,1,XOFY,IANS2)
     *                    CALL IFNCTN(YOFX,2,1.,1.,2,1.,2.,1,XOFY,IANS4)
     *                    CALL IFNCTN(YOFX,2,1.,2.,0,1.,2.,1,XOFY,IANS5)
     *                    CALL IFNCTN(YOFX,2,1.,2.,2,0.,2.,1,XOFY,IANS6)
     *                    CALL IFNCTN(YOFX,2,1.,2.,2,1.,20.,1,XOFY,IANS7A)
     *                    CALL IFNCTN(YOFX,2,1.,2.,2,1.,1.,1,XOFY,IANS7B)
     *    OUTPUTS - IANS1,2,4,5,6,7A,7B, = 1,2,4,5,6,7,7
     *
     *
     * PROGRAM FOLLOWS BELOW
     *
     * TRANSFER VECTOR CONTAINS MONOCK(X, LX, ZFINCR, IANSNG, IANS)
     *                     AND  REVER(X, LX, XREVD)
     *
            HTR     0               XR1
            HTR     0               XR2
            HTR     0               XR4
            BCI     1,IFNCTN
     *
     * ONLY ENTRY.  IFNCTN(YOFX, LYOFX, XFIRST, XLAST, LXOFY, YLO, YHI,
     *                    IERRLO, XOFY, IANS)
     *
     IFNCTN SXD     IFNCTN-4,1
            SXD     IFNCTN-3,2
            SXD     IFNCTN-2,4
     *
     * SET ADDRESSES
     *
            CLA     1,4             A(YOFX)
            STA     TSXY1
            STA     TSXR1
            STA     TSXR3
            ADD     K1              A(YOFX)+1
            STA     CLAY1
            STA     CAS1
            STA     FSB1
            STA     CLA2
            STA     CLA3
            ADD     K1              A(YOFX)+2
            STA     FSB2
            STA     CAS2
            CLA     9,4             A(XOFY)
            ADD     K1              A(XOFY)+1
            STA     STO1
     *
     * AND DECREMENTS
     *
            CLA*    2,4             LYOFX
            STD     TXH
            SUB     KD1             LYOFX-1
            STD     TXL1
            CLA*    5,4             LXOFY
            SUB     KD1             LXOFY-1
            STD     TXL3
            SUB     KD1             LXOFY-2
            STD     TXL4
     *
     * CHECK  LYOFX GRTHN= 2,  XFIRST NOT= XLAST,  SET  DELX,  XNEXT=XFIRST
     *
            CLA*    8,4             IERRLO
            ADD     KD1
            PDX     0,1             IERRLO+1 TO XR1 FOR ERROR FLAGGING
            CLA*    2,4             LYOFX
            STO     LYOFX
            CAS     KD2
            NOP                     OK
            TRA     *+2             OK
            TRA     LEAVE           NG
            SUB     KD1
            LRS     18
            ORA     OCTK
            FAD     OCTK
            STO     TEMP1           FLOATED LYOFX-1
            TXI     *+1,1,2         IERRLO+3
            CLA*    4,4             XLAST
```

```
       STO     XLAST
       FSB*    3,4             XLAST-XFIRST
       TZE     LEAVE           EQUALITY ERROR
       FDP     TEMP1
       STQ     DELX            = (XLAST-XFIRST)/(LYOFX-1)
       CLA*    3,4
       STO     XNEXT           INITIALIZE XNEXT = XFIRST
       CLA*    6,4
       STO     YNEXT           AND YNEXT = YLO
*
* CHECK  LXOFY GRTHN= 1,  YLO LSTHN YHI  IF  LXOFY GRTHN 1,  FORM  DELY
*
       TXI     *+1,1,1         IERRLO+4
       CLA*    5,4             LXOFY
       STO     LXOFY
       CAS     KD1
       TRA     SUB1            OK
       TXI     YOXCK,1,2       OK, BUT BYPASS CHECKS, DELY
       TRA     LEAVE           NG
 SUB1  SUB     KD1
       LRS     18
       ORA     OCTK
       FAD     OCTK
       STO     TEMP1           LXOFY-1 FLOATED
       TXI     *+1,1,2         IERRLO+6
       CLA*    6,4             YLO
       CAS*    7,4             AGAINST YHI
       NOP                     NG
       TRA     LEAVE           NG
       CLA*    7,4             YHI
       STO     YHI
       FSB     YNEXT
       FDP     TEMP1
       STQ     DELY            DELY GRTHN ZERO
*
* NOW FIND OUT WHETHER  YOFX  IS INCREASING OR DECREASING AND GO CHECK
* IT.  ALSO SET  YMIN,  YMAX.
*
 YOXCK TXI     *+1,1,-6        IERRLO  + ZERO NOW
       STZ     ZFINCR          SET ZFINCR FOR INCREASING
       LXD     LYOFX,2
 CLAY1 CLA     **,2            ** = A(YOFX)+1     YOFX(LYOFX)
       LDQ*    1,4             YOFX(1) TO MQ
       STO     YMAX            TRIAL
       STQ     YMIN            SETTINGS
       CAS*    1,4
       TRA     MONCK           IS INCREASING, OK
       TRA     MONCK           IS CONSTANT, OK
       SXD     ZFINCR,2        REVERSE SENSE OF ZFINCR
       STQ     YMAX            AND YMAX
       STO     YMIN            AND YMIN
 MONCK TSX     $MONOCK,4
 TSXY1 TSX     **,0            ** = A(YOFX)
       TSX     LYOFX,0
       TSX     ZFINCR,0
       TSX     KD1,0           (IANSNG,0)
       TSX     TEMP1,0         (IANS,C)
       LXD     IFNCTN-2,4
       ZET     TEMP1
       TRA     LEAVE
*
* CHECK  YLO  GRTHN=  YMIN,  YHI  LSTHN=  YMAX
*
       TXI     *+1,1,5         IERRLO+5
       CLA     YMIN
       CAS     YNEXT           (YNEXT = YLO)
       TRA     LEAVE           NG
       NOP                     OK
       TXI     *+1,1,1         OK
       CLA     YHI
       CAS     YMAX
       TRA     LEAVE           NG
       NOP                     OK
*
* FOR MONOTONE DECREASING, REVERSE YOFX AND THE X VARIABLES THEN ENTER
```

```
*     LOOP
*
           NZT     ZFINCR
           TRA     START
           TSX     REV,2
           CLA     XNEXT
           LDQ     XLAST
           STO     XLAST
           STQ     XNEXT
           CLA     DELX
           CHS
           STO     DELX
*
* ENTER LOOP WITH XR1 = IYXNXT = 1,2,....,LYOFX
*                 XR2 = IXYNXT = 1,2,....,LXOFY
*
 START     AXT     1,1
           AXT     1,2
           TRA     CLA1
*
* RESET FOR NEXT  XNEXT,  FORCING EXACT EQUALITY WITH  XLAST  FOR
*     IYXNXT = LYOFX
*
YGRYOX     CLA     XNEXT
           FAD     DELX
           STO     XNEXT
 TXL1      TXL     CLA1,1,**         ** = LYOFX-1
           CLA     XLAST
           STO     XNEXT
*
* COMPARE YNEXT AGAINST  YOFX(IYXNXT)
*
 CLA1      CLA     YNEXT             (FIRST VALUE = YLO)
 CAS1      CAS     **,1              ** = A(YOFX)+1
           TXI     YGRYOX,1,1        (BUMP IYXNXT AND GO RESET XNEXT)
           TRA     EQUAL
*
* YNEXT IS NOW BRACKETED BY  YOFX(IYXNXT-1)  LSTHN  YNEXT  LSTHN
*     YOFX(IYXNXT)
 FSB1      FSB     **,1              ** = A(YOFX)+1
           STO     TEMP1             -(YOFX(IYXNXT)-YNEXT)
 CLA2      CLA     **,1              ** = A(YOFX)+1
 FSB2      FSB     **,1              ** = A(YOFX)+2
           STO     TEMP2
           CLA     TEMP1
           FDP     TEMP2
           FMP     DELX
           FAD     XNEXT             XNEXT-DELX*(YOFX(IYXNXT)-YNEXT)/
*                                        (YOFX(IYXNXT)-YOFX(IYXNXT-1))
 STO1      STO     **,2              ** = A(XOFY)+1
*
* RESET FOR NEXT  YNEXT,  FORCING  YNEXT  EXACTLY  =  YHI  FOR  IXYNXT
*     =  LXOFY
*
           CLA     YNEXT
           FAD     DELY
 TXL3      TXL     TXL4,2,**         ** = LXOFY-1   CHECK COMPLETION
           TRA     WINDUP
 TXL4      TXL     STO2,2,**         ** = LXOFY-2   CHECK FOR LAST Y
           CLA     YHI
 STO2      STO     YNEXT
           TXI     CAS1,2,1
*
* IF  YNEXT  =  YOFX(IYXNXT)  IT MAY ALSO  =  YOFX(IYXNXT+1)  ETC.
* THIS ROUTINE COUNTS  NSAMEY  =  NO. OF SUCH EQUALITIES.
*       IT HANDLES THE SPECIAL CASES IN WHICH  NSAMEY  =  1,  AND
*       IN WHICH THE EQUALITIES RUN OFF THE END OF  YOFX.
*
 EQUAL     PXD     0,1
           PDX     0,4
           LDQ     KD1               COUNT NSAMEY IN MQ
 TXI1      TXI     *+1,4,1           START IYXTEMP = IYXNXT+1
 TXH       TXH     COVER,4,**        ** = LYOFX
 CLA3      CLA     **,4              ** = A(YOFX)+1     YOFX(IYXTMP+1)
 CAS2      CAS     **,4              ** = A(YOFX)+2     YOFX(IYXTMP)
```

```
        TRA     COVER
        XCA
        ADD     KD1             (CAS CAN'T JUMP HERE, GUARANTEED BY
*                                MONOCK)
        XCA
        TRA     TXI1
*
* THEN  XOFY(IXYNXT)  =  XNEXT + DELX*(NSAMEY-1)/2,  FOR ANY  NSAMEY
*
 COVER XCA
        SUB     KD1
        LRS     18
        ORA     OCTK
        FAD     OCTK
        FDP     K2L
        FMP     DELX
        FAD     XNEXT
        TRA     STO1
*
* FOR MONOTONE DECREASING, RE-REVERSE  YOFX
*
WINDUP LXD      IFNCTN-2,4
        AXT     0,1             IANS=0 SETTING
        ZET     ZFINCR
        TSX     REV,2
*
* EXIT.  (ASSUMES XR4 RESTORED)
*
 LEAVE PXD      0,1
        STO*    10,4            IANS
        LXD     IFNCTN-4,1
        LXD     IFNCTN-3,2
        TRA     11,4
*
* INTERNAL SUBROUTINE TO REVERSE  YOFX
*
*       LINKAGE XR2, RETURN TO 1,2 (REFILLS XR4 FROM  IFNCTN-2)
*
 REV   TSX      $REVER,4
 TSXR1 TSX      **,0            ** = A(YOFX)
        TSX     LYOFX,0
 TSXR3 TSX      **,0            ** = A(YOFX)
        LXD     IFNCTN-2,4
        TRA     1,2
*
* CONSTANTS
*
 K1    PZE      1
 KD1   PZE      0,0,1
 KD2   PZE      0,0,2
 K2L   DEC      2.0
 OCTK  OCT      233000000000
*
* VARIABLES
*
 LYOFX PZE      0,0,**          INPUT
 LXOFY PZE      0,0,**          INPUT
 YHI   PZE      **,**,**        INPUT
 DELX  PZE      **,**,**
 DELY  PZE      **,**,**
 XNEXT PZE      **,**,**        XFIRST    (XLAST)    (+DELX...)
 YNEXT PZE      **,**,**        YLO                  (+DELY...)
 XLAST PZE      **,**,**        XLAST     (XFIRST)
 YMAX  PZE      **,**,**
 YMIN  PZE      **,**,**
 TEMP1 PZE      **,**,**
 TEMP2 PZE      **,**,**
ZFINCR PZE      0,0,**          **=0 IF MONO INCR., =LYOFX IF MONO DECR.
        END
```

```
************************
*    IGETX             *
************************
 REFER TO
     GETX
```



```
************************
*    IGETX             *
************************
 REFER TO
     GETX
```

```
*      IINTGR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0156
*      FAP
*IINTGR
       COUNT   150
       LBL     IINTGR
       ENTRY   IINTGR (YOFX1,YIGRTD,DELX,LY,YOFX,CIGRTN)
*
*                    ----ABSTRACT----
*
* TITLE - IINTGR
*        INVERSION OF TRAPEZOIDAL INTEGRAL
*
*              IINTGR PERFORMS THE INVERSE OPERATION TO THAT OF
*              SUBROUTINE INTGRA, I.E. IT FINDS A VECTOR, YOFX, WHOSE
*              TRAPEZOIDAL INTEGRAL IS A GIVEN VECTOR, YIGRTD.
*              THE INITIAL VALUE OF YOFX IS REQUIRED AS INPUT. THE
*              CONSTANT OF INTEGRATION IS AN OUTPUT.
*
*              THE OUTPUT VECTOR YOFX MAY REPLACE THE INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 49   REGISTERS
* SPEED      -  7090       709     7090       709
*              (45.2  OR  47.0) + (37.8  OR  41.0)*LY MACHINE CYCLES,
*                                              LY = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL IINTGR(YOFX1,YIGRTD,DELX,LY,YOFX,CIGRTN)
*
*                 NOTE THAT THE ARGUMENTS ARE IDENTICAL TO THOSE OF
*                 SUBROUTINE INTGRA EXCEPT THAT THE ORDER IS REVERSED.
*                 SEE SUBROUTINE INTGRA FOR DETAILED DISCUSSION.
*
* INPUTS
*
*    YOFX1     STARTING VALUE FOR YOFX(1)
*
*    YIGRTD(I) I=1...LY  IS THE TRAPEZOIDALLY INTEGRATED VECTOR
*
*    DELX      WAS THE (NON-ZERO) DELTA X USED IN OBTAINING YIGRTD
*
*    LY        MUST EXCEED ZERO
*
* OUTPUTS      STRAIGHT RETURN WITH NO ACTION IF LY LSTHN 1 OR DELX = 0.
*
*    YOFX(I)   I=1...LY  IS THE VECTOR WHICH INTEGRATES TO YIGRTD
*                 YOFX(1) = YOFX1
*                 YOFX(K) = (2/DELX)*(YIGRTD(K)-YIGRTD(K-1)) - YOFX(K-1)
*                     FOR K = 2,3,...,LY
*
*              EQUIVALENCE(YOFX,YIGRTD) IS PERMITTED
*
*    CIGRTN    IS SET = YIGRTD(1)
*
* EXAMPLES     THE EXAMPLES USED HERE ARE THE INVERSES OF THE EXAMPLES
*              USED FOR INTGRA
*
* 1. INPUTS  - YI1(1...7) =  0., 1., 2.,...,  6.      YF= 1.
*              YI2(1...7) =  0.,10.,20.,..., 60.
*              YI3(1...7) =  0.,-2.,-4.,...,-12.
*              YI4(1...7) =  1., 2., 3.,...,  7.
*              YI5(1...2) = -1., 0.
*              YI6(1)     = -1.,
*              Y7 = C7 = Y8 = C8 = -999.
*
*    USAGE   -       CALL IINTGR( YF, YI1,  1., 7, Y1, C1)
*                    CALL IINTGR( YF, YI2, 10., 7, Y2, C2)
*                    CALL IINTGR( YF, YI3, -2., 7, Y3, C3)
```

```
*                    CALL IINTGR( YF, YI4,  1., 7, Y4, C4)
*                    CALL IINTGR( YF, YI5,  1., 2, Y5, C5)
*                    CALL IINTGR( YF, YI6,  1., 1, Y6, C6)
*                    CALL IINTGR( YF, YI1,  1., 0, Y7, C7)
*                    CALL IINTGR( YF, YI1,  0., 7, Y8, C8)
*
*    OUTPUTS - Y1(I)=Y2(I)=Y3(I)=Y4(I) =  1.  FOR I=1...7
*              Y5(1...2) = 1.,1.  Y6(1) = 1.
*              C1=0.  C2=0.  C3=0.  C4=1.  C5=-1. C6=-1.
*              Y7=C7=Y8=C8 = -999.  (NO OUTPUT CASES)
*
* 2. MULTIPLE DIFFERENTIATION WITH OUTPUTS REPLACING INPUTS
*    INPUTS  - YI(1...7) =   0., 1., 6., 19., 44., 85., 146.
*              YOFX1(1...3) =  0., 0., 4.   (NOTE THIS IS REVERSED
*                                             FROM EXAMPLE FOR INTGRA)
*    USAGE   -        DO  10   I=1,3
*                  10  CALL IINTGR(YOFX1(I),YI,1.,7,YI,C(I))
*    OUTPUTS - YI(1...7) =  4., 4.,...,4.   C(1...3) = 0.,0.,0.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
       BCI     1,IINTGR
* ONLY ENTRY. IINTGR(YOFX1, YIGRTD, DELX, LY, YOFX, CIGRTN)
IINTGR SXD     IINTGR-2,4
* CHECK LY (AT LEAST = 1) AND DELX (NON-ZERO, UNLESS LY=1)
       CLA*    4,4            LY
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
       SXD     TXL,4          STORE  LY IF OK.
       TXL     LXD4,4,1       AVOID DELX BUSINESS IF LY = 1
       LXD     IINTGR-2,4
       CLA*    3,4            DELX
       TZE     LEAVE
       STO     TWOVDX         (TEMP FOR DELX)
       CLA     FL2
       FDP     TWOVDX         2.0/DELX
       STQ     TWOVDX
* SET OUTPUTS FOR LY AT LEAST = 1
 LXD4  LXD     IINTGR-2,4
       CLA*    2,4            YIGRTD(1)
       STO     LASTYI         SAVE FOR LOOP
       STO*    6,4            AND STORE IN CIGRTN
       CLA*    1,4            YOFX1
       STO*    5,4            YOFX(1)
* THEN SET LOOP FOR LARGER LY
       CLA     2,4
       ADD     K1             A(YIGRTD)+1
       STA     GET
       CLA     5,4
       ADD     K1             A(YOFX)+1
       STA     STORE
       ADD     K1             A(YOFX)+2
       STA     SUB
* BUT BYPASS THE LOOP IF LY=1
       LXD     TXL,4
       TXL     LEAVE,4,1
* OTHERWISE GO AHEAD WITH LOOP
       AXT     2,4
* LOOP TO SET YOFX(2...LY)
 GET   LDQ     **,4           ** = A(YIGRTD)+1
       CLS     LASTYI
       STQ     LASTYI
       FAD     LASTYI                          YIGRTD(K)-YIGRTD(K-1)
       XCA
       FMP     TWOVDX                          TIMES 2/DELX
 SUB   FSB     **,4           **=A(YOFX)+2     MINUS YOFX(K-1)
 STORE STO     **,4           **=A(YOFX)+1     BECOMES YOFX(K)
       TXI     *+1,4,1
 TXL   TXL     GET,4,**       **= LY
* EXIT
 LEAVE LXD     IINTGR-2,4
```

```
         TRA     7,4
* CONSTANTS, TEMPORARIES
 FL2     DEC     2.0
 K1      PZE     1
LASTYI   PZE     **,**,**      = PREVIOUS YIGRTD VALUE, START YIGRTD(1)
TWOVDX   PZE     **,**,**      = 2.0/DELX
         END
```

```
*     INDATA (SUBROUTINE)                10/1/64   LAST CARD IN DECK IS NO. 0488
*     LABEL
CINDATA
      SUBROUTINE INDATA(ITAPE,IRECNO,NOPTS,DATA,ERR)
C
C                    ----ABSTRACT----
C
C TITLE - INDATA
C        FAST AND CONVENIENT RETRIEVAL OF DATA FROM A SPECIAL TAPE
C
C              INDATA SEARCHES A TAPE CONSISTING OF MANY DATA SERIES AND
C              OTHER INFORMATION ABOUT EACH SERIES. THE REQUESTED
C              INFORMATION IS RETURNED TO THE CALLING PROGRAM. THE
C              DETAILS OF THE TAPE LAYOUT ARE DESCRIBED ALONG WITH THE
C              SUBROUTINE OUDATA.
C
C                 INDATA ACQUIRES ITS SPEED PRIMARILY THROUGH 1) HIGH
C              SPEED TAPE SCANNING ALONG WITH INTERNAL TABLES OF DATA
C              POSITION AND 2) BY ITS ABILITY TO READ DATA OF LIMITED
C              ACCURACY FROM A TIGHTLY PACKED FORMAT, FOR EXAMPLE, DATA
C              WITH AN ACCURACY OF ONE PART IN 4096 CAN BE STORED ON
C              ONE THIRD THE TAPE THAT WOULD BE REQUIRED USING FORTRAN
C              BINARY TAPE PROCEDURES.
C
C                 INDATA ACQUIRES ITS CONVENIENCE THROUGH THE FACTS THAT
C              1) THE PROGRAMMER HAS BOTH THE DATA, AND INFORMATION ABOUT
C              THE DATA AVAILABLE BY MEANS OF A SINGLE CALL STATEMENT
C              2) THE PROGRAMMER NEED NOT KNOW ANY DETAILS ABOUT DATA
C              ARRANGEMENT, PACKING, TAPE POSITION, ETC.
C
C              THE INDATA-OUDATA SYSTEM INCLUDES PROGRAMMED SUMCHECKS
C              ON DATA STORAGE AND RETRIEVAL WHICH ARE INDEPENDENT OF
C              BUILT-IN HARDWARE CHECKS.
C
C LANGUAGE  - FORTRAN II, SUBROUTINE, (WITH SUBROUTINES IN FAP)
C EQUIPMENT - 709 OR 7090 (DATA CHANNEL AND ONE TAPE UNIT)
C STORAGE   - 896 REGISTERS
C SPEED     -
C AUTHOR    - J.F. CLAERBOUT
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FAPSUM,FSKIP,LOC,MVBLOK,UNPAKN,
C                                     VARARG,XSAME,
C       AND FORTRAN SYSTEM ROUTINES - (FIL),(RLR),(SPH),(STH),(TSB).
C
C FORTRAN USAGE
C     CALL INDATA(ITAPE,IRECNO,NOPTS,DATA,ERR,...............)
C              THE NUMBER OF ARGUMENTS IN THE CALL STATEMENT IS VARIABLE
C              DEPENDING ON THE DESIRED INFORMATION, SEE EXAMPLES.
C
C INPUTS
C
C    ITAPE      IS THE LOGICAL TAPE NUMBER TO BE SEARCHED AND READ. IF
C               MORE THAN 2 DIFFERENT TAPE UNITS ARE TO BE USED DURING
C               ONE JOB, IT WILL BE NECESSARY TO CHANGE THE FIRST
C               DIMENSION CARD IN THIS PROGRAM. THIS IS DESCRIBED JUST
C               PRECEDING THAT CARD.
C
C    IRECNO     THE REQUESTED DATA RECORD NUMBER. THE RECORD NUMBERS OF
C               THE DATA RECORDS ON THE TAPE ARE SUPPLIED BY THE PERSONS
C               WHO ORIGINATED THE TAPE. THE NUMBERS MAY BE FIXED POINT,
C               FLOATING POINT, OCTAL, OR ALPHANUMERIC.
C               IF THE USER IS INTERESTED IN MERELY READING THE DATA
C               RECORDS IN THE SEQUENCE THAT THEY OCCUR ON THE TAPE, HE
C               MAY SET IRECNO=0 IN WHICH CASE THE NEXT RECORD IS READ
C               AND ITS ACTUAL RECORD NUMBER WILL BE RETURNED AS IRECNO.
C
C    NOPTS      NORMALLY THIS IS NOT AN INPUT. IF HOWEVER THE PROGRAM-
C               MER IS NOT REQUESTING DATA, BUT ONLY INFORMATION ABOUT
C               THE DATA, SUBSTANTIAL TIME CAN BE SAVED BY AVOIDING THE
C               ACTUAL DATA READ AND INTERPRETATION. THE DATA WILL NOT
C               BE READ IF NOPTS IS SET =(ANY NEGATIVE NUMBER) BEFORE
C               CALLING INDATA.
C
```

```
C      *********SEE ALSO  ((SPECIAL REQUESTS))  BELOW FOR MORE INPUTS.
C
C OUTPUTS
C
C    DATA(I)    I=1,NOPTS   IS THE RETURNED DATA SERIES. NORMALLY THIS IS
C               ASSUMED FLOATING POINT UNLESS OTHERWISE SPECIFIED BY THE
C               ORIGINATOR OF THE TAPE. DATA MUST BE DIMENSIONED TO THE
C               MAXIMUM OF EITHER 1) NOPTS+1 OR 2) SOME NUMBER DEPENDING
C               ON THE AMOUNT OF INFORMATION STORED ABOUT THE DATA...AS-
C               SUME 200 UNLESS SPECIFIED OTHERWISE BY ORIGINATOR OF TAPE
C
C    ERR        SPECIFIES AN ERROR CONDITION
C               =0.    IMPLIES NO ERROR CONDITION
C               =1.    REQUESTS NOT ON TAPE, ALL ELSE RETURNED PROPERLY
C               =2.    SUMCHECK ERROR ON TAPE, EVERYTHING RETURNED AS WELL
C                      AS POSSIBLE.
C               =3.    PROGRAM TRIED TO USE MORE TAPES THAN THERE ARE
C                      TABLES DIMENSIONED. SEE INPUTS- ITAPE.
C               =4.    END DATA INFORMATION ON TAPE (I.E. THE RECORD NO.
C                      CALLED WAS NOT FOUND).
C               =5.    ILLEGAL LENGTH OF CALL STATEMENT. I.E. CALL
C                      INDATA(ARG1,ARG2,.......ARGN)   N MUST BE ODD.
C               =6.    MORE DATA RECORDS ON TAPE THAN SIZE OF INTERNAL
C               BUFFER. CHANGE DIMENSION STATEMENT BELOW FOR IRECTB AND
C               DEFINITION OF MAXREC.
C
C               WHEN AN ERROR CONDITION OCCURS, ERR, ITAPE, IRECNO,
C               AND THE REQUEST (IF ERR = 1.) ARE PRINTED ON-LINE.
C
C    IRECNO     IF IRECNO WAS SET =0 ON INPUT THEN IT WILL BE RESET TO
C               THE RECORD NUMBER FOUND NEXT ON THE TAPE.
C
C    NOPTS      IF NOPTS WAS SET LSTHN 0 THEN THE CORRECT NOPTS FOR
C               THE IRECNO IS RETURNED.
C
C    **********SEE ALSO ((SPECIAL REQUESTS))  BELOW FOR MORE OUTPUTS.
C
C SPECIAL REQUESTS
C
C        INDATA IS A DEPARTURE FROM NORMAL FORTRAN PROGRAMMING IN THAT
C    THE CALLING PROCEDURE DEPENDS ON THE NEEDS OF THE PROGRAMMER.
C    THIS IS BEST EXPLAINED THRU THE EXAMPLES. THE BASIC IDEA IS
C    THAT ONE GENERALLY NEEDS MORE THAN JUST RAW DATA. IN SEISMIC TIME
C    SERIES FOR EXAMPLE ONE OFTEN ALSO NEEDS VARIOUS INFORMATION ABOUT
C    THE SEISMIC EVENT, ITS RECORDING, AND ITS DIGITIZATION. GENERALLY
C    THE ORIGINATOR OF THE TAPE WILL SUPPLY ALL OF THIS INFORMATION ON
C    THE TAPE. IF THIS INFORMATION IS ON THE TAPE AND IF THE PROGRAMMER
C    REQUESTS ANY PORTION OF IT, INDATA WILL RETURN HIS REQUEST.  A
C    REQUEST IS MADE BY MEANS OF A NAME (6 CHARACTERS OR LESS) SUPPLIED
C    BY THE TAPE ORIGINATOR. FOR EXAMPLE THE NAME ((DELTAT)) MIGHT
C    REFER TO THE DIGITIZATION SAMPLING TIME OF THE DATA.
C
C    A MAXIMUM OF 25 SPECIAL REQUESTS IS ALLOWED.
C
C EXAMPLES
C               IN THE FOLLOWING EXAMPLES THE VARIABLE NOPTS IS ASSUMED
C               TO BE NON-NEGATIVE ON ENTRY TO INDATA, EXCEPT AS NOTED.
C
C 1. USAGE    -        CALL INDATA(9,63,NOPTS,DATA,ERR)
C    OUTPUTS - TAPE NUMBER 9 IS SCANNED IN SEARCH OF DATA RECORD NUMBER
C               63 (WHICH IS GENERALLY NOT THE 63RD RECORD ON THE TAPE).
C               WHEN IT IS FOUND, NOPTS IS SET TO THE NUMBER OF POINTS IN
C               THE DATA RECORD, (DATA(I),I=1,NOPTS) IS RETURNED AND ERR
C               IS SET =0
C
C 2. INPUTS   - IRECNO = 0
C    USAGE    -        CALL INDATA(9,IRECNO,NOPTS,DATA,ERR)
C    OUTPUTS - INSTEAD OF SCANNING THE TAPE THE NEXT RECORD IN POSITION
C               IS READ. ITS RECORD NUMBER IS RETURNED AS IRECNO. OTHER-
C               WISE THE READ OCCURS AS IN EXAMPLE 1.
C
C 3. USAGE    -        CALL INDATA(9,62,NOPTS,DATA,ERR,6HDELTAT,DT)
C    OUTPUTS - THIS IS LIKE EXAMPLE 1. EXCEPT FOR THE INCLUSION OF TWO
C               MORE ARGUMENTS FOLLOWING ERR. THE FIRST OF THESE,
C                 (DELTAT) IS THE NAME WHICH WE CONVENTIONALLY USE FOR
```

```
C              THE TIME SAMPLING OF THE DATA SERIES. THUS IF ONE OF
C              OUR SEISMIC TAPES WERE ON TAPE DRIVE 9, SEISMIC RECORD
C              62 WOULD BE FOUND AND READ INTO DATA(I),I=1,NOPTS,
C              NOPTS WOULD BE SET =3301 , DT WOULD BE SET =.05 INDICAT-
C              ING 3301 POINTS WERE DIGITIZED AT 20 POINTS PER SECOND.
C                 GENERALLY THEN, AFTER ERR IN THE CALLING SEQUENCE
C              APPEAR ARGUMENTS IN PAIRS, THE FIRST BEING THE NAME OF
C              THE REQUEST, THE SECOND MEMBER OF THE PAIR BEING THE
C              FORTRAN VARIABLE NAME OF THE REQUESTED INFORMATION. SEE
C              THE FOLLOWING EXAMPLE.
C
C 4. USAGE   -        DIMENSION T(10)
C                     CALL INDATA(9,62,NOPTS,DATA,ERR,5HTITLE,T,6HDELTAT,
C                     DT,3HJOB,J)
C    OUTPUTS - THIS IS JUST LIKE EXAMPLE 3. EXCEPT THAT MORE SPECIAL
C              REQUESTS ARE MADE. NOTICE THAT THE SPECIAL REQUEST
C              (TITLE) IS A VECTOR AND THEREFORE IS DIMENSIONED. REQUEST
C              (JOB) IS FIXED POINT HENCE (J) IS A FIXED POINT VARIABLE
C              NAME.
C
C 5. USAGE   -        NOPTS =-1
C                     CALL INDATA(9,62,NOPTS,DATA,ERR,6HDELTAT,DT)
C    OUTPUTS - THIS IS JUST LIKE EXAMPLE 3. EXCEPT THAT DATA(I) IS NOT
C              READ FROM THE TAPE RESULTING IN A SUBSTANTIAL TIME SAVING
C              NOPTS AND DT ARE RETURNED AS IN EXAMPLE 3. DATA MUST
C              STILL BE DIMENSIONED AS IN EXAMPLE 3.
C
C
C    THE FOLLOWING EXAMPLES ARE TESTS THAT ARE TO VERIFY THAT ALL OF THE
C    FEATURES OF INDATA ACTUALLY WORK.  A SPECIAL INDATA-OUDATA TYPE
C    TAPE IS GENERATED FOR EACH TEST.
C
C 6. TEST OF SEARCHING ABILITY ON  1  TAPE UNIT
C    INPUTS  - ITAPE = 9  IRECNO(1...10) = 1,5,9,4,4,4,4,3,2,1
C    USAGE   - C      FIRST CONSTRUCT A TAPE WITH  10 FILES. EACH FILE
C              C      HAVING 1 UNPACKED INTEGER DATA POINT IDENTICAL WITH
C              C      THE RECORD NUMBER, AND WITH NO AUXILIARY INFO.
C                     DIMENSION I(3)
C                     REWIND ITAPE
C                     DO 10  I=1,10
C                     CALL OUDATA (ITAPE,I,1,I,1)
C               10    CONTINUE
C                     CALL OUDATA (ITAPE,0,1,DATA,1)
C                     REWIND ITAPE
C              C
C              C      NOW TEST INDATA
C                     DO 10  I=1,10
C                     CALL INDATA (ITAPE,IRECNO(I),NOPTS,DATA(I),ERR(I))
C               10    CONTINUE
C    OUTPUTS - DATA(1...10) = 1,5,9,4,4,4,4,3,2,1    NOPTS=1
C              ERR (1...10) = 0,0,0,0,0,0,0,0,0,0
C
C 7. TEST OF SEARCHING ABILITY ON 2 TAPE UNITS (AND ILLUSTRATE USE OF
C    NON-FIXED POINT RECORD NUMBERS.
C    INPUTS  - ITAPE(1...10)  = 9,12, 9, 9,12,12, 9, 9,12, 9
C              IRECNO(1...10) = 1,1., 5, 4,9.,5., 4, 4,3., 7
C    USAGE   - C      ASSUME TWO TAPES CONSTRUCTED AS IN EXAMPLE 6.
C              C      LOGICAL TAPE 9 CONTAINS FIXED POINT INTEGERS FOR
C              C         RECORD NUMBERS AND UNPACKED DATA,
C              C      LOGICAL TAPE 12 CONTAINS FLOATING POINT INTEGERS
C              C
C                     DO 10  I=1,10
C                     CALL INDATA (ITAPE(I),IRECNO(I),NOPTS,DATA(I),
C                   1           ERR(I))
C               10  CONTINUE
C    OUTPUTS - DATA(1...10) = 1,1., 5, 4,9.,5., 4, 4,3., 7
C              ERR (1...10) = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
C
C 8. TEST OF SEQUENTIAL READING OF RECORDS
C    INPUTS  - ITAPE = 9  IRECNO(1...5) = 0,0,0,0,0
C    USAGE   - C      ASSUME A DATA TAPE CONSTRUCTED AS IN EXAMPLE 6.
C              C
C                     REWIND ITAPE
C                     DO 10  I=1,5
C                     CALL INDATA (ITAPE,IRECNO(I),NOPTS,DATA(I),ERR(I))
```

```
      C              10   CONTINUE
      C     OUTPUTS - IRECNO(1...5) = 1,2,3,4,5
      C               DATA  (1...5) = 1,2,3,4,5
      C               ERR   (1...5) = 0,0,0,0,0
      C
      C 9. TEST OF NON-RETURN OF DATA AND GETTING SPECIAL REQUESTS.
      C     INPUTS  - (FOR CALLING INDATA)
      C               ITAPE =12  IRECNO = 0  NOPTS = -1
      C     USAGE   - C      CONSTRUCT A DATA TAPE WITH AUXILIARY INFORMATION
      C               C      FOR THIS PURPOSE LET
      C               C         DT = .05
      C               C         TITLE(1...8) = 6H REPRESENTATIVE TITLE FOR REC.
      C               C         DATA (1...5) = 1.,2.,3.,4.,5.
      C               C
      C                      REWIND 12
      C                      CALL OUDATA(12,6HSAMPLE,5,DATA,3,
      C                     1                 6HDELTAT,1,DT
      C                     2                 5HTITLE ,8,TITLE)
      C                      CALL OUDATA (12,0,1,DATA,1)
      C                      REWIND 12
      C               C
      C               C      DIMENSION DATA SO THAT INDATA WILL HAVE COMPUTATION
      C               C      SPACE - EVEN THOUGH DATA IS NOT WANTED.
      C                      DIMENSION DATA(200),TITLE(8)
      C               C
      C                      CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,
      C                     1                 6HDELTAT,DT,
      C                                       5HTITLE ,TITLE)
      C     OUTPUTS - IRECNO = 6HSAMPLE   NOPTS =5  ERR = 0.
      C               DATA CONTAINS MEANINGLESS NUMBERS
      C               DT = .05   TITLE(1...8) = 6H REPRESENTATIVE TITLE FOR REC
      C
      C10. TEST OF ERROR CONDITION  -  REQUESTS NOT ON TAPE, ALL ELSE RETURNED
      C     INPUTS  - ITAPE =12  IRECNO = 0  NOPTS = 0
      C     USAGE   - C      ASSUME A DATA TAPE CONSTRUCTED AS IN EXAMPLE 9.
      C               C
      C               C      DIMENSION DATA(200),TITLE(8)
      C                      REWIND ITAPE
      C                      CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,
      C                     1                 6HDELTAT,DT
      C                     2                 6HNO REQ,NON)
      C     OUTPUTS - IRECNO = 6HSAMPLE   DATA(1...5) = 1.,2.,3.,4.,5.
      C               ERR = 1.0      DT = .05      NOPTS = 5
      C               AND ON-LINE PRINTED MESSAGE
      C               NO REQ
      C               ERROR IN SUBROUTINE INDATA, ERROR CODE = 1., TAPE=12
      C                    RECORD NUMBER IN OCTAL=    622144474325
      C
      C11. TEST ERROR CONDITION  -  SUMCHECK ERROR ON TAPE, EVERYTHING
      C                                  RETURNED AS WELL AS POSSIBLE.
      C     INPUTS  - ITAPE =12  IRECNO = 0
      C     USAGE   - C      CONSTRUCT A DATA FILE WITH AUXILIARY INFORMATION
      C               C      BUT CHANGE THE SUMCHECK ON THE DATA
      C               C        LET DATA(1...5) = 1.,2.,3.,4.,5.  ZERO = 0.
      C                      REWIND 12
      C                      CALL OUDATA(12,6H ERROR,5,DATA,1,6HDELTAT,1,DT)
      C                      BACKSPACE 12
      C                      BACKSPACE 12
      C                      WRITE TAPE 12, (DATA(I),I=1,5),ZERO
      C                      END FILE 12
      C                      CALL OUDATA (12,0,1,DATA,1)
      C                      REWIND 12
      C               C
      C               C      TEST INDATA
      C                      CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,
      C                     1                 6HDELTAT,DT)
      C     OUTPUTS - IRECNO = 6H ERROR  NOPTS = 5  DATA(1...5) = 1.,2.,3.,4.,5
      C               ERR = 2.0    DT = .05   AND ON-LINE MESSAGE (SEE EX. 10)
      C
      C12. TEST ERROR CONDITION  -  PROGRAM TRIED TO USE MORE TAPES THAN
      C                                  THERE ARE TABLES DIMENSIONED
      C     INPUTS  - ITAPE(1...3) = 9,12,11   NOPTS(1...3) = -1,-1,-1
      C               IRECNO(1...3)= 0, 0, 0
      C     USAGE   - C      ASSUME TWO TAPES ARE SET UP AS IN EXAMPLE 7.
      C               C      THERE IS NO NEED TO SET UP A TAPE ON LOGICAL
```

```
C              C     UNIT 11 BECAUSE INDATA WILL NEVER GET THERE.
C              C
C                    DO 10  I=1,3
C                    CALL INDATA (ITAPE(I),IRECNO(I),NOPTS(I),DATA,
C                   1               ERR(I))
C                 10   CONTINUE
C     OUTPUTS - ERR(1...3) = 0.,0.,3.   AND ON-LINE MESSAGE (SEE EX. 10)
C
C13. TEST ERROR CONDITION  -  END DATA INFORMATION ON TAPE  (RECORD NO.
C                             CALLED FOR WAS NOT FOUND)
C     INPUTS  - ITAPE = 9  IRECNO = 41
C     USAGE   - C     ASSUME A DATA TAPE CONSTRUCTED AS IN EXAMPLE 6.
C              C
C                    CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR)
C     OUTPUTS - ERR = 4.0   AND ON-LINE MESSAGE (SEE EX. 10)
C
C14. TEST ERROR CONDITION  -  ILLEGAL LENGTH OF CALL STATEMENT
C     INPUTS  - ITAPE = 12  IRECNO = 0
C     USAGE   -          CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR,
C                   1               6HDELTAT)
C     OUTPUTS - ERR = 5.  AND ON-LINE MESSAGE (SEE EX. 10)
C
C15. TEST ERROR CONDITION  -  MORE RECORDS ON TAPE THAN SIZE OF
C                             INTERNAL BUFFER OF INDATA.
C     INPUTS  - ITAPE = 9   IRECNO = 105
C     USAGE   - C     CONSTRUCT A DATA TAPE AS IN EXAMPLE 6 EXCEPT
C              C        PUT 110 FILES ON IT RATHER THAN 10.
C              C
C                    CALL INDATA (ITAPE,IRECNO,NOPTS,DATA,ERR)
C     OUTPUTS - ERR = 6.   AND ON-LINE MESSAGE (SEE EX. 10)
C
C PROGRAM FOLLOWS BELOW
C
C      LET NTAPES BE THE NUMBER OF DATA TAPES BEING READ AND
C      LET MAXREC BE THE MAXIMUM NUMBER OF DATA RECORDS,
C      THEN A DIMENSION STATEMENT OF THE FOLLOWING TYPE IS NEEDED...
C      DIMENSION IRECTB(MAXREC,NTAPES),IPOSIT(NTAPES),LOGICL(NTAPES),
C     1LENTBL(NTAPES)
C      FOR EXAMPLE THREE CARDS OF THE FOLLOWING FORM MUST BE PRESENT
       NTAPES=2
       MAXREC=100
       DIMENSION IRECTB(100,2),IPOSIT(2),LOGICL(2),LENTBL(2)
C
C
C      DO NOT CHANGE THE FOLLOWING CARDS
       DIMENSION LOCS(50),REQS(2,25),DATA(5000)
       EQUIVALENCE(LOCS(6),REQS(1))
C      SET UP VARIABLE LENGTH CALL AND RETURN
       CALL VARARG(LOCS)
       GO TO 20
 10    RETURN
 20    CONTINUE
C       IS THE CALLING SEQUENCE LEGAL...  (ODD NO. OF ARGUMENTS)
       LCALL=0
 22    LCALL=LCALL+1
       IF(LOCS(LCALL))22,24,22
 24    IF(XMODF(LCALL,2))26,28,26
 26    ERR=5.
       GO TO 310
 28    CONTINUE
       ERR=0.
       NOPTSS=NOPTS
C      IS ITAPE ON TABLE OF LOGICAL TAPE NUMBERS
       DO 30  I=1,NTAPES
       IUNIT=I
       IF(ITAPE-LOGICL(I))  30,60,30
 30    CONTINUE
C      NO, IS THERE ROOM ON LIST OF TAPE NUMBERS FOR ITAPE
       DO 40  I=1,NTAPES
       IUNIT=I
       IF(LOGICL(I))  40,50,40
 40    CONTINUE
C      IF PROGRAM GETS HERE, THERE ARE TOO MANY TAPES BEING REFERRED TO
       ERR=3.
       GO TO 290
```

```
C     PUT ITAPE ON LIST OF TAPE NUMBERS
 50   LOGICL(IUNIT)=ITAPE
 60   CONTINUE
C     NOW WE KNOW WHICH TAPES ARE READ, AND WHICH TABLES TO REFER TO
C     IF THE REQUESTED RECNO IS ZERO, WE AVOID IRECTB SCAN
      IF(IRECNO)  70,100,70
 70   CONTINUE
C     IS REQUESTED RECORD ON IRECTB (FIRST TIME THRU,J=0,DOESNT MATTER)
      J=LENTBL(IUNIT)
      DO 80  I=1,J
      II=I
      IF(IRECNO-IRECTB(I,IUNIT))  80,90,80
 80   CONTINUE
C     NOT ON IRECTB
C     SKIP TAPE TO END OF KNOWN PORTION
      CALL FSKIP(ITAPE,LENTBL(IUNIT)-IPOSIT(IUNIT))
      IPOSIT(IUNIT)=LENTBL(IUNIT)
      GO TO 100
 90   CONTINUE
C     SKIP TO CORRECT POSITION ON TAPE, FSKIP IS FILE SKIPPING ROUTINE
C     (IPOSIT IS TAPE FILES FROM BEGINNING OF TAPE)
      CALL FSKIP(ITAPE,II-1-IPOSIT(IUNIT))
      IPOSIT(IUNIT)=II-1
 100  CONTINUE
C     READ NEXT SEISMOGRAM FROM DATA TAPE
C     READ ABOUT TAPE LAYOUT
      READ TAPE ITAPE,IREC,NALPHA,NOPTS,MODCOD,SCALE
C     AT END OF TAPE YET
      IF(IREC)  120,110,120
 110  ERR=4.
 115  CALL FSKIP(ITAPE,-1)
      GO TO 290
 120  CONTINUE
C     IS IREC ALREADY ON TABLE...
C                                      YES, NO, IMPOSSIBLE
      IF(IPOSIT(IUNIT)-LENTBL(IUNIT))  127,124,124
C     NO, AUGMENT TABLE LENGTH COUNTER, ADD IREC TO TABLE.
 124  CONTINUE
      LENTBL(IUNIT)=LENTBL(IUNIT)+1
      IF(LENTBL(IUNIT)-MAXREC)  126,126,125
 125  ERR=6.
      GO TO 115
 126  J=LENTBL(IUNIT)
      IRECTB(J,IUNIT)=IREC
 127  CONTINUE
C     AUGMENT TAPE POSITION COUNTER
      IPOSIT(IUNIT)=IPOSIT(IUNIT)+1
C     IS THIS THE DESIRED RECORD
C     IF IRECNO IS ZERO WE TAKE ANY RECORD
      IF (IRECNO) 130,138,130
 130  IF(IRECNO-IREC)135,140,135
 135  CALL FSKIP(ITAPE,1)
      GO TO 100
C     YES, THIS IS THE DESIRED RECORD
 138  IRECNO = IREC
 140  CONTINUE
C     PREPARE TO READ AUX INFO BLOCK
      READ TAPE ITAPE,(DATA(I),I=1,NALPHA)
      CALL FAPSUM(NALPHA-1,DATA,SUMCK)
      IF(DATA(NALPHA)-SUMCK)  150,160,150
 150  ERR=2.
 160  CONTINUE
C     PICK UP A REQUEST
 170  J=0
      CALL LOC(REQUES,LR)
 180  J=J+1
C     END OF REQUESTS YET...
      IF(REQS(1,J))185,260,185
 185  CALL MVBLOK(1,REQS(1,J),LR)
C     SCAN AUX. BLOCK TO SEE IF REQUEST WAS ON TAPE
 190  I=1
 200  ALPHA=DATA(I)
      LBLOK=XSAMEF(DATA(I+1))
      IF(REQUES-ALPHA)  210,250,210
C     HAS ALL OF AUX BLOK BEEN SCANNED WITHOUT SUCCESS
```

****
* INDATA *
****
(PAGE 7)

```
  210 IF(ALPHA)  240,220,240
  220 ERR=1.
      PRINT 230,REQUES
      WRITE OUTPUT TAPE 2,230,REQUES
  230 FORMAT(1H ,A6)
      GO TO 180
  240 I=I+2+LBLOK
      GO TO 200
C     MOVE REQUEST TO CALLING PROGRAM
  250 CALL LOC(DATA(I+2),L)
      CALL MVBLOK(LBLOK,L,REQS(2,J))
      GO TO 180
C     ALL SET TO GET DATA.
C     IF NOPTS WAS NEGATIVE, SKIP OVER DATA
  260 CONTINUE
      IF(NOPTSS)280,270,270
C     COMPUTE LENGTH OF DATA BLOCK (DON'T FORGET SUMCHECK)
  270 N=(NOPTS+MODCOD-1)/MODCOD+1
      READ TAPE ITAPE,(DATA(I),I=1,N)
      CALL FAPSUM(N-1,DATA,SUMCK)
      IF(DATA(N)-SUMCK)  272,275,272
  272 ERR=2.
  275 CALL UNPAKN(MODCOD,NOPTS,DATA,SCALE)
C     PASS OVER END OF FILE MARK
  280 CALL FSKIP(ITAPE,1)
      IF(ERR)  290,10,290
C     ERROR PRINT
  290 CONTINUE
      IF(ERR-4.)  310,300,310
  300 IF (IRECNO)  310,305,310
  305 ERR=0.
      GO TO 10
  310 CONTINUE
      PRINT 320,ERR,ITAPE,IRECNO
      WRITE OUTPUT TAPE 2,320,ERR,ITAPE,IRECNO
  320 FORMAT(     41H ERROR IN SUBROUTINE INDATA, ERROR CODE =,F3.0,
     17H, TAPE=,I2,25H  RECORD NUMBER IN OCTAL=,O15)
      GO TO 10
      END
```

```
*       INDEX  (FUNCTIONS)                9/4/64   LAST CARD IN DECK IS NO. 0269
*       FAP
*INDEX
        COUNT    300
        LBL      INDEX
        ENTRY    INDEX  F(I,ICRTCL)
        ENTRY    VINDEX F(I,ICRTCL,IJUMP)
        ENTRY    SETEST F(X,XNEW,XCRTCL)
        ENTRY    SETAPT F(X,XNEW,FVALUE)
        ENTRY    CHUSET F(X,X1,X2,ZIFX1)
*
*
*                     ----ABSTRACT----
*
* TITLE - INDEX, WITH SECONDARY ENTRIES VINDEX,SETEST,SETAPT, AND CHUSET
*        HYBRID SUBPROGRAMS FOR INCREMENTING, TESTING, AND SETTING
*
*              THESE PROGRAMS ARE OPERATED AS FORTRAN-II FUNCTIONS BUT
*              ALSO MODIFY THE CONTENTS OF THE STORAGE LOCATION
*              CORRESPONDING TO THEIR FIRST ARGUMENT.  THE FUNCTION
*              VALUES ARE DESIGNED FOR CONTROL APPLICATIONS IN  IF
*              STATEMENTS.
*
*              INDEX  ADDS A FIXED POINT  1  TO ITS FIRST ARGUMENT AND
*              HAS VALUE  -1.0, 0.0, 1.0  ACCORDING AS THE SUM IS
*              LSTHN, EQUAL TO, OR GRTHN  ITS SECOND ARGUMENT.
*
*              VINDEX  ADDS ITS THIRD ARGUMENT (FIXED POINT) TO ITS
*              FIRST AND IS OTHERWISE THE SAME AS  INDEX.
*
*              SETEST  SETS ITS FIRST ARGUMENT EQUAL ITS SECOND AND HAS
*              VALUE  -1.0, 0.0, 1.0  ACCORDING AS ITS SECOND ARGUMENT
*              IS  LSTHN, EQUAL TO, OR GRTHN  ITS THIRD ARGUMENT, THE
*              ARGUMENT MODES BEING ARBITRARY.
*
*              SETAPT  SETS ITS FIRST ARGUMENT EQUAL ITS SECOND, AND HAS
*              VALUE EQUAL ITS THIRD ARGUMENT.
*
*              CHUSET  SETS ITS FIRST ARGUMENT EQUAL ITS SECOND  (IF ITS
*              FOURTH HAS ZERO MAGNITUDE)  OR ITS THIRD  (IF ITS FOURTH
*              HAS NON-ZERO MAGNITUDE), AND HAS VALUE EQUAL ITS FOURTH
*              ARGUMENT.
*
*              THE COMPARISONS OF  INDEX,  VINDEX,  AND  SETEST  TREAT
*              PLUS AND MINUS ZERO AS EQUAL.
*
* LANGUAGE   - FAP FUNCTIONS  (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094  (MAIN FRAME ONLY)
* STORAGE    - 50    REGISTERS
* SPEED      - ABOUT  70  MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                     ----USAGE----
*
* NO TRANSFER VECTOR
*
*
*
* FORTRAN USAGE OF INDEX FUNCTION
*     ANS = INDEXF(I,ICRTCL)
*
* INPUTS TO INDEX
*
*    I  AND  ICRTCL  ARE FIXED POINT.
*
* OUTPUTS FROM INDEX
*
*    I         = I+1
*
*    ANS       = -1.0,0.0,+1.0  AS  I+1  LSTHN ICRTCL,  = ICRTCL
*                   GRTHN ICRTCL.
*
*
*
```

```
* FORTRAN USAGE OF VINDEX FUNCTION
*     ANS = VINDEXF(I,ICRTCL,IJUMP)
*
* INPUTS TO VINDEX
*
*    I,  ICRTCL,  AND  IJUMP  ARE FIXED POINT.
*
* OUTPUTS FROM VINDEX
*
*    I          = I+IJUMP
*
*    ANS        = -1.0,0.0,+1.0  AS  I+IJUMP  LSTHN ICRTCL,  = ICRTCL,
*                   GRTHN ICRTCL.
*
*
*
* FORTRAN USAGE OF SETEST FUNCTION
*     ANS = SETESTF(X,XNEW,XCRTCL)
*
* INPUTS TO SETEST
*
*    X,  XNEW,  AND  XCRTCL  ARE ANY MODE.
*
* OUTPUTS FROM SETEST
*
*    X          = XNEW.
*
*    ANS        = -1.0,0.0,+1.0  AS  XNEW  LSTHN XCRTCL,  = XCRTCL,
*                   GRTHN XCRTCL.
*
*
*
* FORTRAN USAGE OF SETAPT FUNCTION
*     ANS = SETAPTF(X,XNEW,FVALUE)
*
* INPUTS TO SETAPT
*
*    X,  XNEW,  AND  FVALUE  ARE ANY MODE.
*
* OUTPUTS FROM SETAPT
*
*    X          = XNEW
*
*    ANS        = FVALUE  (POSSIBLY A MISNAMED FIXED POINT VALUE WHICH
*                          WILL NOT HURT IN  IF  STATEMENT APPLICATIONS.
*                          USE  IANS = XSAMEF(SETAPTF(X,XNEW,IFVALU))
*                          IF THIS IS TROUBLESOME.)
*
*
*
* FORTRAN USAGE OF CHUSET FUNCTION
*       ANS = CHUSETF(X,X1,X2,ZIFX1)
*
* INPUTS TO CHUSET
*
*      X,  X1,  X2,  AND ZIFX1  ARE ANY MODE.
*
* OUTPUTS FROM CHUSET
*
*    X          = X1  IF MAGNITUDE OF  ZIFX1  IS ZERO,  = X2  OTHERWISE.
*
*    ANS        = ZIFX1  (COMMENT ABOUT  ANS  UNDER  SETAPT  APPLIES).
*
*
*
* EXAMPLES      CARE MUST BE TAKEN IN THE USE OF  INDEX  AND  VINDEX  FOR
*               SUBSCRIPTING PURPOSES AS ILLUSTRATED IN THE LOOP PORTIONS
*               OF EXAMPLES 1. AND 2.
*
* 1. INDEX
*    USAGE   -        I = -2
*                  ANS1 = INDEXF(I,0)
*                  ANS2 = INDEXF(I,0)
*                  ANS3 = INDEXF(I,0)
*                  J = 0
```

```
*                10 J = J
*                   X(J+1) = J
*                   IF (INDEXF(J,10))  10,10,20
*                20 CONTINUE
*    OUTPUTS - ANS1,2,3 = -1.0,0.0,1.0    I = 1
*              X(1...11) = 0.,1.,...,10.    J = 11
*
* 2. VINDEX AND SETEST
*    USAGE    -         K = 21
*                10 K = K
*                   Y(K) = K
*                   IF (VINDEXF(K,1,-1))  20,10,10
*                20 ANS1 = SETESTF(X1,1.0,2.0)
*                   ANS2 = SETESTF(X2,-0.0,-0.0)
*                   ANS3 = SETESTF(X3,-0.0,0.0)
*                   ANS4 = SETESTF(X4,0.0,-0.0)
*                   ANS5 = SETESTF(X5,0.0,0.0)
*                   ANS6 = SETESTF(X6,2.0,1.0)
*                   ANS7 = SETESTF(IX,2,1)
*    OUTPUTS - Y(1...21) = 1.,2.,...,21.  K = 0
*              ANS1,2,3,4,5,6,7 = -1.0,0.0,0.0,0.0,0.0,1.0,1.0
*              X1,2,3,4,5,6 = 1.0,-0.0,-0.0,0.0,0.0,2.0  IX = 2
*
* 3. SETAPT AND CHUSET
*    USAGE    -         ANS1 = SETAPTF(X,1.,2.)
*                       ANS2 = SETAPTF(IX,7,3.)
*                       ANS3 = CHUSETF(IY1,1,3,0.0)
*                       ANS4 = CHUSETF(IY2,1,3,-2.0)
*                       ANS5 = CHUSETF(IY3,1,3,1.5)
*    OUTPUTS - ANS1,2,3,4,5 = 2.,3.,0.,-2.,1.5    X = 1.    IX = 7
*              IY1,2,3 = 1,3,3
*
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       BCI     1,INDEX
*
* PRINCIPAL ENTRY.  INDEXF(I,ICRTCL)
*
 INDEX ADD     KD1               I+1
 SXD4  SXD     ZFSACS,4          SWITCH SETTING FOR  INDEX,  VINDEX,
*                                AND  SETEST
       STQ     CRTICL            ICRTCL OR XCRTCL TO CRTICL
       XCA                       I+1, I+IJUMP, OR XNEW TO MQ
       TRA     SXA4
*
* SECOND ENTRY.  VINDEXF(I,ICRTCL,IJUMP)
*
VINDEX ADD     32765             I+IJUMP
       TRA     SXD4
*
* THIRD ENTRY.  SETESTF(X,XNEW,XCRTCL)
*
SETEST XCA                       XNEW TO AC
       LDQ     32765             XCRTCL TO MQ
       TRA     SXD4
*
* FOURTH ENTRY.  SETAPTF(X,XNEW,FVALUE)
*
SETAPT CLA     LDQX2             XNEW  IN MQ IS OK, SET TO
       TRA     STACAC            PICK UP  FVALUE  FROM  32765
*
* FIFTH ENTRY.  CHUSETF(X,X1,X2,ZIFX1)
*
CHUSET ZET     32764             X1  IN MQ OK IF  ZIFX1 = 0
 LDQX2 LDQ     32765             OTHERWISE  X2  TO MQ
       CLA     CHUSET            SET TO PICK UP  ZIFX1  FROM  32764
STACAC STA     CLAAC             SET TERMINAL PICKUP FOR  SETAPT,  CHUSET
       STZ     ZFSACS            SET SWITCH FOR  SETAPT,  CHUSET
*
* AT THIS POINT WE HAVE MQ = NEW VALUE, CRTICL = CRITICAL VALUE (FIRST
*    3 ENTRIES).  BACK UP XR4 TILL -1,4 HAS STORAGE ADDRESS FOR NEW
```

```
*    VALUE (CLA I  OR  CLA X)
*
 SXA4  SXA     SV4,4
 CAL   CAL     -1,4
       ANA     ATMASK          KNOCK OUT ADDRESS AND TAG
       LAS     CLAZ
       TXI     CAL,4,1
       TRA     GOTCLA
       TXI     CAL,4,1
*
* THEN ANTICIPATE ENTRIES INDEX,VINDEX,SETEST, SET NEW VALUE, TEST,
*    TREAT LAST  2  ENTRIES
*
GOTCLA CLS     K1L             (ANTICIPATE FOR NEW LSTHN CRTICL)
       XCA
       STO*    -1,4            STORE NEW
 SV4   AXT     **,4            ** = XR4
       ZET     ZFSACS
       TRA     NTSPCS
 CLAAC CLA     **              ** = 32765 FOR SETAPT, = 32764 FOR CHUSET
       TRA     1,4
*
* COMPARE, FOR INDEX,VINDEX,SETEST, (NEW IN AC,-1.0 IN MQ)
*
NTSPCS TNZ     CAS             TEST
       NZT     CRTICL          FOR
       TRA     LDQZ            ZERO = ZERO
 CAS   CAS     CRTICL
       TRA     GETK1           NEW GRTHN  CRTICL
 LDQZ  LDQ     KZ              NEW EQUALS CRTICL
       XCA                     NEW LSTHN  CRTICL
       TRA     1,4
 GETK1 CLA     K1L
       TRA     1,4
*
* CONSTANTS, TEMPORARY
*
ATMASK OCT     777777000000
 CLAZ  CLA     0,0
 K1L   DEC     1.0
 KD1   PZE     0,0,1
 KZ    PZE     0
CRTICL PZE     **,**,**        ICRTCL OR XCRTCL
ZFSACS PZE     0,0,**          ** =0 IF SETAPT OR CHUSET, =XR4 OTHERWISE
       END
```

```
*      INTGRA (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0174
*      FAP
*INTGRA
       COUNT   150
       LBL     INTGRA
       ENTRY   INTGRA (CIGRTN,YOFX,LY,DELX,YIGRTD,YOFX1)
*
*                    ----ABSTRACT----
*
* TITLE - INTGRA
*        INDEFINITE INTEGRAL BY TRAPEZOIDAL RULE
*
*              INTGRA FORMS A VECTOR, YIGRTD(I) I=1...LY, REPRESENTING
*              THE INTEGRAL OF ANOTHER VECTOR, YOFX(I) I=1...LY, PLUS
*              A GIVEN INTEGRATION CONSTANT, CIGRTN.  THE INPUT VECTOR
*              YOFX IS CONSIDERED TO REPRESENT EQUALLY SPACED ORDINATE
*              VALUES OF A FUNCTION Y(X) AS FOLLOWS
*                    YOFX(1)  =  Y(X1)
*                    YOFX(2)  =  Y(X2)    WHERE X2 = X1+DELX
*                    YOFX(3)  =  Y(X3)    WHERE X3 = X2+DELX
*                           ETC
*                    YOFX(LY) =  Y(XN)    WHERE XN = XN-1 + DELX
*                                           NOTE- DELX MAY BE NEGATIVE
*
*              LET THE INTEGRAL TO BE COMPUTED BE REPRESENTED BY F(X)
*              WITH
*                                        X
*                     F(X)  =  C  +   INTEGRAL ( Y(U) DU )
*                                      U = X1
*
*              THEN THE OUTPUT VECTOR IS
*                        YIGRTD(1) = F(X1)   (THIS IS ALWAYS = C)
*                        YIGRTD(2) = F(X2)
*                               ETC
*                        YIGRTD(LY) =F(XN)
*
*              WHERE C = CIGRTN  AND THE TRAPEZOIDAL APPROXIMATION IS
*              USED FOR INTEGRATING.
*
*              THE OUTPUT VECTOR MAY REPLACE THE INPUT VECTOR.
*
*              INTGRA HAS ONE OTHER OUTPUT, YOFX1, WHICH IS SET EQUAL
*              TO THE VALUE OF YOFX(1).  USING THIS QUANTITY
*              IT IS POSSIBLE TO INVERT EXACTLY THE INTEGRATED VECTOR,
*              YIGRTD, AND REOBTAIN YOFX.  THIS INVERSION IS PERFORMED
*              BY SUBROUTINE IINTGR, WHOSE CALLING SEQUENCE IS THE
*              REVERSE OF THAT OF INTGRA.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 47   REGISTERS
* SPEED      -  7090       709       7090       709
*              (41.2  OR 43.0) + (37.8  OR 41.0)*LY  MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL INTGRA(CIGRTN,YOFX,LY,DELX,YIGRTD,YOFX1)
*
* INPUTS
*
*    CIGRTN    IS THE CONSTANT OF INTEGRATION
*
*    YOFX(I)   I=1...LY  IS THE VECTOR TO BE INTEGRATED
*
*    LY        SHOULD EXCEED 0
*
*    DELX      SHOULD BE NON-ZERO, MAY BE NEGATIVE
*
* OUTPUTS      STRAIGHT RETURN WITH NO ACTION IF LY LSTHN 1 OR DELX = 0.
*
```

```
*    YIGRTD(I) I=1...LY  IS THE INTEGRATED VECTOR
*                   YIGRTD(1) = CIGRTN
*                   YIGRTD(K) = YIGRTD(K-1) +
*                                  DELX*(YOFX(K) + YOFX(K-1))/2.0
*                             FOR K=2,3,...,LY
*
*              EQUIVALENCE(YIGRTD,YOFX) IS PERMITTED
*
*    YOFX1     IS SET EQUAL TO YOFX(1)
*
* EXAMPLES
*
* 1. WITH VARIOUS VALUES OF DELX, CIGRTN AND LY
*    INPUTS  - Y(1...7) = 1., 1.,...,1.
*              YI1=YI2=...=YI8 = -999.        YI7 = YI8 = -999.
*
*    USAGE   -        CALL INTGRA(  0., Y, 7,  1., YI1, YI1)
*                     CALL INTGRA(  0., Y, 7, 10., YI2, YI2)
*                     CALL INTGRA(  0., Y, 7, -2., YI3, YI3)
*                     CALL INTGRA(  1., Y, 7,  1., YI4, YI4)
*                     CALL INTGRA( -1., Y, 2,  1., YI5, YI5)
*                     CALL INTGRA( -1., Y, 1,  1., YI6, YI6)
*                     CALL INTGRA(  0., Y, 0,  1., YI7, YI7)
*                     CALL INTGRA(  0., Y, 7,  0., YI8, YI8)
*    OUTPUTS - YI1(1...7) =  0., 1., 2.,...,  6.     YI1 = 1.
*              YI2(1...7) =  0.,10.,20.,..., 60.     YI2 = 1.
*              YI3(1...7) =  0.,-2.,-4.,...,-12.     YI3 = 1.
*              YI4(1...7) =  1., 2., 3.,...,  7.     YI4 = 1.
*              YI5(1...2) = -1., 0.                  YI5 = 1.
*              YI6(1)     = -1.                      YI6 = 1.
*              YI7 = YI7 = YI8 = YI8 = -999.  (NO OUTPUT CASES)
*
* 2. MULTIPLE INTEGRATION WITH OUTPUT REPLACING INPUT
*    INPUTS  - Y(1...7) = 4.,4.,...,4.
*    USAGE   -        DO 10  I=1,3
*                  10  CALL INTGRA(0.,Y,7,1.,Y,YOFX1(I))
*    OUTPUTS - Y(1...7) = 0., 1., 6., 19., 44., 85., 146.
*              YOFX1(1...3) = 4., 0., 0.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR      0              XR4
       BCI      1,INTGRA
* ONLY ENTRY. INTGRA(CIGRTN, YOFX, LY, DELX, YIGRTD, YOFX1)
INTGRA SXD      INTGRA-2,4
* CHECK LY (AT LEAST =1) AND DELX (NON-ZERO UNLESS LY=0)
       CLA*     3,4            LY
       TMI      LEAVE          EXIT IF NEGATIVE,
       PDX      0,4
       TXL      LEAVE,4,0      OR ZERO.
       SXD      TXL,4          STORED IF OK.
       TXL      LXD4,4,1       AVOID DELX BUSINESS IF LY=1
       LXD      INTGRA-2,4
       CLA*     4,4            DELX
       TZE      LEAVE
       FDP      FL2            DELX /2.0
       STQ      DXOV2
* NOW SET OUTPUTS FOR LY AT LEAST =1
 LXD4  LXD      INTGRA-2,4
       CLA*     2,4            YOFX(1)
       STO      LASTY
       STO*     6,4            YOFX1
       CLA*     1,4            CIGRTN
       STO*     5,4            YIGRTD(1)
* THEN SET UP LOOP FOR LARGER LY
       CLA      2,4
       ADD      K1             A(YOFX)+1
       STA      GET
       CLA      5,4
       ADD      K1             A(YIGRTD)+1
       STA      STORE
       ADD      K1             A(YIGRTD)+2
       STA      ADD
```

```
* BUT BYPASS THE LOOP IF LY =1
      LXD    TXL,4
      TXL    LEAVE,4,1
* OTHERWISE ENTER LOOP
      AXT    2,4
* LOOP TO SET YIGRTD(2...LY)
 GET  LDQ    **,4           ** = A(YOFX)+1
      CLA    LASTY
      STQ    LASTY
      FAD    LASTY                               YOFX(K)+YOFX(K-1)
      XCA
      FMP    DXOV2                               TIMES DELX/2.0
 ADD  FAD    **,4           **=A(YIGRTD)+2       PLUS YIGRTD(K-1)
 STORE STO   **,4           **=A(YIGRTD)+1       BECOMES YIGRTD(K)
      TXI    *+1,4,1
 TXL  TXL    GET,4,**       **=LY
* EXIT
 LEAVE LXD   INTGRA-2,4
      TRA    7,4
* CONSTANTS, TEMPORARIES
 FL2  DEC    2.0
 K1   PZE    1
 LASTY PZE   **,**,**       = PREVIOUS YOFX VALUES, STARTS WITH YOFX(1)
 DXOV2 PZE   **,**,**       = DELX/2.0
      END
```

```
*      INTHOL (SUBROUTINE)               9/9/64   LAST CARD IN DECK IS NO. 0155
*      FAP
*INTHOL
      COUNT   200
      LBL     INTHOL
      ENTRY   INTHOL (NHOL,HOL,FMT,NDATAD,NDATAA,DATA)
*
*
*                    ----ABSTRACT----
*
* TITLE - INTHOL
*         INTERPRET HOLLERITH
*
*               SUBROUTINE INTHOL INTERPRETS A SERIES OF HOLLERITH WORDS
*               ACCORDING TO A GIVEN FORMAT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 72 REGISTERS
* SPEED      -
* AUTHOR     - R.A. WIGGINS   4/64
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - FNDFMT
*       AND FORTRAN SYSTEM ROUTINES - (IOH),(RTN)
*
* FORTRAN USAGE
*     CALL INTHOL(NHOL,HOL,FMT,NDATAD,NDATAA,DATA)
*
*
* INPUTS
*
*    NHOL       NUMBER OF HOLLERITH WORDS TO BE INTERPRETED
*               MUST BE GRTHN= 1, LSTHN= 22
*
*    HOL(I)     I=1,...,NHOL HOLLERITH WORDS (6 HOLLERITH CHARACTERS PER
*                 WORD) TO BE INTERPRETED.
*
*    FMT(I)     I=1,2,... IS EITHER A NORMAL FORMAT VECTOR, OR LITERAL
*                 HOLLERITH IN A CALLING SEQUENCE WHOSE CHARACTERS
*                 (READING LEFT TO RIGHT) ARE THE DESIRED FORMAT STRIPPED
*                 OF THE ENCLOSING PARENTHESES.  THE FIRST AND SECOND
*                 CHARACTERS MUST NOT BE EITHER  '('  OR  ')' .
*               IS USED TO INTERPRET HOL(I).
*               SHOULD BE A FORMAT FOR READING ONLY ONE LINE, I.E. IT
*                 SHOULD CONTAIN NO  '/' .
*
*    NDATAD     NUMBER OF DATA VALUES DESIRED FROM HOL ACCORDING TO FMT.
*               SHOULD NOT PROVIDE FOR A GREATER NUMBER OF VALUES THAN
*                 CAN BE INTERPRETED.
*
*
* OUTPUTS
*
*    NDATAA     NUMBER OF DATA VALUES ACTUALLY INTERPRETED.  INTHOL SCANS
*                 THE HOL AND FMT ONLY ONCE TO FIND THE DESIRED VALUES.
*
*    DATA(I)    I=1,...,NDATAA CONTAINS THE VALUES INTERPRETED.
*
*
* EXAMPLES
*
* 1. INPUTS  - NHOL = 1  HOL(1) = 6H-53.31   FMT(1) = 6H(F6.2)  NDATAD=1
*    OUTPUTS - NDATAA = 1  DATA(1) = -53.31
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NDATAD = 6
*    OUTPUTS - SAME AS EXAMPLE 1.
*
* 3. INPUTS  - NHOL = 2  HOL(1...2) = 3HXYZ,6H  5 -9  NDATAD = 3
*    USAGE   -       CALL INTHOL(NHOL,HOL, 6HA6,2I3 ,NDATAD,NDATAA,
*                  1             DATA)
```

****
*   INTHOL   *
****
(PAGE 2)


```
*     OUTPUTS - NDATAA = 3  DATA(1...3) = 3HXYZ,5,-9
*
*
* PROGRAM FOLLOWS BELOW
*
 XPR1   HPR     0
 XPR2   HPR     0
 XPR4   HPR     0
        BCI     1,INTHOL
INTHOL  SXD     XPR4,4          SAVE
        SXD     XPR2,2          INDEX
        SXD     XPR1,1          REGISTERS.
        AXT     BUFSIZ,2        PUT
        CLA     =0606060606060  BLANKS
        STO     -1,2            IN
        TIX     *-1,2,1         BUFFER
        CAL     2,4             BEFORE
        ADD     =1B35
        STA     HOL             TRANSFERRING
        CLA*    1,4             THE
        STD     NHOL
        AXT     1,1             HOLLERITH
        AXT     BUFSIZ,2
 HOL    CLA     **,1            WORDS
        STO     -1,2            TO
        TIX     *+2,2,1         THE
        TRA     NHOL+1          INPUT-
        TXI     *+1,1,1         OUTPUT
 NHOL   TXL     HOL,1,**        BUFFER.
        CLA*    4,4             SET UP
        STD     NDATA           INSTRUCTIONS
        CAL     6,4             FOR
        ADD     =1B35           STORING
        STA     DATA            DATA.
        CLA     3,4             SET
        STA     FMT1            UP
        TSX     $FNDFMT,4       FNDFMT
 FMT1   TSX     **,0            ARGUMENTS
        TSX     INUM,0
        CLA     =32562B17       CONVERT
        SUB     INUM            INDEX WITH RESPECT TO COMMON
        ARS     18              TO A
        STA     FMT             MACHINE ADDRESS.
        STZ     IFST            INITIALIZE ONE-PASS COUNTER.
        STZ     INUM            INITIALIZE DATA COUNTER.
*
* INITIALIZE (IOH)
*
        CLA     =2B17           DUMMY ITAPE
        AXC     FMT-1,4         LOAD IR4 FOR DUMMY READING SEQUENCE
        LDQ     NOP             GET INPUT (SSH) FLAG
        TRA*    $(IOH)        * GO INITIALIZE (IOH)
 NOP    NOP     SSH
*
* RETURN FROM (IOH) - GO BACK
*
 SSH    ZET     IFST            IS THIS FIRST RETURN
        TRA     RTN1            NO, GO EXIT
        SXD     IFST,4          YES, RESET IFST AND
        TRA     1,4           * RETURN TO (IOH)  (BUFFER WAS SET UP ABOVE)
*
* DUMMY READING SEQUENCE
*
 FMT    PZE     **              FORMAT ADDRESS
        AXT     1,1             SEQUENCE
 STR    STR                     FOR
 DATA   STQ     **,1            OBTAINING
        SXD     INUM,1         CONVERTED
        TXI     *+1,1,1         NUMBERS
 NDATA  TXL     STR,1,**        FROM
        TSX     $(RTN),4        (IOH).
 RTN1   LXD     XPR4,4          RESTORE IR 4
        CLA     INUM            OUTPUT ACTUAL
        STO*    5,4             NO. OF DATA VALUES STORED.
 RTN    LXD     XPR1,1          RESOTRE IR 1
```

INTHOL (PAGE 3)

```
        LXD     XPR2,2          RESOTRE IR 2
        TRA     7,4           * RETURN TO MAIN.
*
 IFST   PZE     0               ONE-CARD SWITCH.
 INUM   PZE     0               DATA COUNT.
BUFSIZ  EQU     22              BUFFER SIZE.
*
        END
```

```
************************
*    INTMSB            *
************************
 REFER TO
     TIMSUB
```


```
************************
*    INTMSB            *
************************
 REFER TO
     TIMSUB
```

```
*      INTOPR (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0250
*      FAP
*INTOPR
       COUNT   200
       LBL     INTOPR
       ENTRY   INTOPR (NDATA, XLO, DELX, X, OPER)
*
*
*                    ----ABSTRACT----
*
* TITLE - INTOPR
*        INTERPOLATION OPERATOR FOR 1 TO 4 EVENLY SPACED DATA VALUES
*
*              INTOPR  FINDS  OPER(1,2,...,N)  GIVEN  N,  X,  XLO,  DELX
*              SUCH THAT FOR ANY  N  EQUALLY SPACED DATA VALUES,
*              F(1),  F(2),  ...,  F(N),  WHERE  N  LSTHN=  4,
*
*                    OPER(1)*F(1) + ... + OPER(N)*F(N)  =  P(X)
*
*              WHERE  P  IS THE EXACT FITTING POLYNOMIAL (OF DEGREE
*              N-1),  TO THE DATA VALUES AS FOLLOWS.
*
*                       P(XLO)              =  F(1)
*                       P(XLO+DELX)         =  F(2)
*                                   ETC.
*                       P(XLO+(N-1)*DELX)   =  F(N)
*
*              THE DEGENERATE CASE OF  N = 1  YIELDS  OPER(1) = 1.0 .
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 111 REGISTERS
* SPEED      -  50 MC FOR N=1          WHERE  MC = MACHINE CYCLES
*               90 MC FOR N=2
*              205 MC FOR N=3
*              435 MC (709) OR 380 MC (7090) FOR N=4
* AUTHOR     - S.M. SIMPSON, MARCH 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL INTOPR(NDATA, XLO, DELX, X, OPER)
*
*
* INPUTS
*
*    NDATA     IS THE QUANTITY N OF THE ABSTRACT
*              MAY ONLY HAVE VALUE 1,2,3, OR 4
*
*    XLO       IS DEFINED IN ABSTRACT
*
*    DELX      IS DEFINED IN ABSTRACT
*              MUST BE NON-ZERO (MAY BE NEGATIVE)
*
*    X         IS DEFINED IN ABSTRACT
*
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF NDATA OR DELX ILLEGAL
*
*    OPER(I)   I=1...NDATA  IS THE OPERATOR AS DEFINED IN ABSTRACT
*
*
* EXAMPLES
*
* 1. THIS EXAMPLE FINDS THE OPERATORS FOR  X  VALUES CORRESPONDING
*              EXACTLY TO THE DATA POINTS, IN WHICH CASE THE
*              CORRESPONDING COEFFICIENT MUST BE UNITY AND THE OTHERS
*              MUST VANISH.
*
*    INPUTS  - NDATA1,NDATA2,NDATA3,NDATA4 = 1,2,3,4  XLO*0.0,  DELX=2.0
```

```
*    USAGE   -        CALL INTOPR(NDATA1,XLO,DELX,0.0,OPER11)
*                     CALL INTOPR(NDATA2,XLO,DELX,0.0,OPER21)
*                     CALL INTOPR(NDATA2,XLO,DELX,2.0,OPER22)
*                     CALL INTOPR(NDATA3,XLO,DELX,0.0,OPER31)
*                     CALL INTOPR(NDATA3,XLO,DELX,2.0,OPER32)
*                     CALL INTOPR(NDATA3,XLO,DELX,4.0,OPER33)
*                     CALL INTOPR(NDATA4,XLO,DELX,0.0,OPER41)
*                     CALL INTOPR(NDATA4,XLO,DELX,2.0,OPER42)
*                     CALL INTOPR(NDATA4,XLO,DELX,4.0,OPER43)
*                     CALL INTOPR(NDATA4,XLO,DELX,6.0,OPER44)
*    OUTPUTS - OPERNM  (N=1...4, M=1...N)  IS A VECTOR OF LENGTH  N,
*                 ALL OF WHOSE ELEMENTS VANISH EXCEPT  OPERNM(M) = 1.0 .
*
* 2. NON-TRIVIAL EXAMPLES
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  DELX = 1.0
*    USAGE   -        CALL INTOPR(NDATA2,XLO,DELX,0.5,OPER23)
*                     CALL INTOPR(NDATA3,XLO,DELX,0.5,OPER34)
*                     CALL INTOPR(NDATA4,XLO,DELX,0.5,OPER45)
*    OUTPUTS - OPER23(1...2) = 0.5,0.5
*              OPER34(1...3) = 0.375,0.750,-0.125
*              OPER45(1...4) = 0.3125,0.9375,-0.3125,0.0625
*
* 3. ERROR EXITS
*    INPUTS  - SAME AS EXAMPLE 1., EXCEPT SET  OPER(1...4) = -9999.
*    USAGE   -        CALL INTOPR(0,XLO,DELX,1.,OPER)
*                     CALL INTOPR(5,XLO,DELX,1.,OPER)
*                     CALL INTOPR(NDATA2,XLO,0.,1.,OPER)
*    OUTPUTS - OPER(1...4) = -9999.
*
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
*
* ONLY ENTRY.  INTOPR(NDATA, XLO, DELX, X, OPER)
*
        HTR     0               XR1
        HTR     0               XR4
        BCI     1,INTOPR
*
*ONLY  ENTRY.  INTOPR (NDATA,XLO,DELX,X,OPER)
*
INTOPR  SXD     INTOPR-2,4
        SXD     INTOPR-3,1
        NZT*    3,4             DELX ZERO CHECK
        TRA     LEAVE
        CLA*    1,4             NDATA CHECK
        TMI     LEAVE
        PDX     0,1             (STAYS IN XR1 TILL WINDUP)
        TXL     LEAVE,1,0
        TXH     LEAVE,1,4
        CLA     5,4             A(OPER)
        ADD     KA1
        STA     STOOPR
        CLA     K2L             (B) ANTICIPATE NDATA=4
        LDQ     K3L             (A)
        TXH     FORMY,1,3       AND TEST FOR IT
        CLA     K1L             (B OR K1) IF NOT, ANTICIPATE NDATA =3, OR 1
        LDQ     K1L             (A)
        TXH     FORMY,1,2       AND TEST FOR IT
        TXL     STOOPR,1,1      (IT MIGHT BE 2, IT MIGHT BE 1)
*
* NDATA=2
*
        CLA*    4,4
        FSB*    2,4
        FDP*    3,4
        STQ     K2              K2 = (X-XLO)/DELX
        CLA     K1L
        FSB     K2
        STO     K1              K1 = 1-K2
        TRA     WINDUP
*
* COMPUTE Y = NORMALIZED X, FOR NDATA = 3, 4
```

```
*       (A IS IN MQ, B IN AC)
* AND THEN BRANCH ON NDATA AGAIN
*
FORMY  STQ      K4               (K4 IS TEMP HERE, FOR A)
       FDP*     3,4
       STQ      K3               (K3 IS TEMP HERE, FOR B/DELX)
       CLA*     4,4              X
       FSB*     2,4              -XLO
       XCA
       FMP      K3               *B/DELX
       FSB      K4               -A
       STO      Y                Y = (X-XLO) *B/DELX - A
       XCA
       FMP      Y
       STO      K4               (Y-SQUARED WILL BE USEFUL)
       TXH      NDEQ4,1,3
*
* RETURN TO THE CASE NDATA=3 AND SET K1,K2,K3
*       (REMEMBER K4 AND AC HAVE Y-SQUARED)
*
       FSB      Y
       FDP      K2L
       STQ      K1               K1=(YSQR-Y)/2.0
       CLA      K1L
       FSB      K4
       STO      K2               K2=(1-YSQR)
       CLA      K4
       FAD      Y
       FDP      K2L
       STQ      K3               K3=(YSQR+Y)/2.0
       TRA      WINDUP
*
* SET K1,K2,K3,K4 FOR NDATA=4
*
 NDEQ4 CLS      Y
       FAD      K3L
       XCA
       FMP      Y
       FAD      K1L
       XCA
       FMP      Y
       FSB      K3L
       FDP      K48L
       STQ      K1               K1=(((-Y+3)Y+1)Y-3)/48
       CLA      Y
       FSB      K1L
       XCA
       FMP      Y
       FSB      K9L
       XCA
       FMP      Y
       FAD      K9L
       FDP      K16L
       STQ      K2               K2=(((Y-1)Y-9)Y+9)/16
       CLS      Y
       FSB      K1L
       XCA
       FMP      Y
       FAD      K9L
       XCA
       FMP      Y
       FAD      K9L
       FDP      K16L
       STQ      K3               K3=(((-Y-1)Y+9)Y+9)/16
       CLA      Y
       FAD      K3L
       XCA
       FMP      Y
       FSB      K1L
       XCA
       FMP      Y
       FSB      K3L
       FDP      K48L
       STQ      K4               K4=(((Y+3)Y-1)Y-3)/48
*
```

```
     * MOVE K1,K2,... TO OPER(1...NDATA)
     *
     WINDUP CLA     K1+1,1
     STOOPR STO     **,1            **=A(OPER)+1
            TIX     WINDUP,1,1
     *
     * EXIT
     *
      LEAVE LXD     INTOPR-3,1
            TRA     6,4
     *
     * CONSTANTS, TEMPORARIES
     *
      KA1   PZE     1
      K1L   DEC     1.0
      K2L   DEC     2.0
      K3L   DEC     3.0
      K9L   DEC     9.0
      K16L  DEC     16.0
      K48L  DEC     48.0
      K4    PZE     **,**,**
      K3    PZE     **,**,**
      K2    PZE     **,**,**
      K1    PZE     **,**,**
      Y     PZE     **,**,**
            END
```

```
*      INTSUM (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0109
*      FAP
*INTSUM
       COUNT   100
       LBL     INTSUM
       ENTRY   INTSUM ( X, LX,XISUMD)
       ENTRY   XNTSUM (IX,LIX,IXISMD)
*
*                    ----ABSTRACT----
*
* TITLE - INTSUM
*        INTEGRATED SUMMATION OF A FLOATING OR FIXED VECTOR
*
*              INTSUM FORMS A FLOATING VECTOR WHOSE I-TH ELEMENT IS THE
*              SUM, THROUGH ELEMENT I, OF THE ELEMENTS OF ANOTHER
*              FLOATING VECTOR.  OUTPUT MAY REPLACE INPUT.
*
*              XNTSUM DOES THE SAME THING FOR FIXED VECTORS.
*
*              INTSUM AND XNTSUM ARE THE EXACT INVERSE OPERATORS OF
*              SUBROUTINES DIFPRS AND XDFPRS RESPECTIVELY.
*
* LANGUAGE   - FAP SUBROUTINES  (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 27   REGISTERS
* SPEED      - INTSUM  35 + 12.4*LX  MACHINE CYCLES, LX= VECTOR LENGTH
*              XNTSUM  37 +  8.0*LX
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL INTSUM( X, LX,XISUMD)
*     CALL XNTSUM(IX,LIX,IXISMD)
*
* INPUTS
*
*    X(I)      I=1...LX  IS A FLOATING VECTOR INPUT TO INTSUM
*    LX        SHOULD EXCEED ZERO
*
*    IX(I)     I=1...LIX IS A FXD VECTOR INPUT TO XNTSUM. THE POSITION
*                  OF THE BINARY POINT IS ARBITRARY.
*    LIX       SHOULD EXCEED ZERO
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1
*
*    XISUMD(I) I=1...LX  IS  XISUMD(I) = SUM(FROM K=1 TO I) OF X(K)
*
*    IXISMD(I) I=1...LIX IS  IXISMD(I) = SUM(FROM K=1 TO I) OF IX(K),
*                  WITH SAME BINARY POINT AS IX.
*              DANGER OF FIXED POINT OVERFLOW NOT TESTED FOR BY XNTSUM.
*
*              EQUIVALENCE(XISUMD,X),(IXISMD,IX) IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4) = 1., 2., 3., 4.   IX(1...4) = 1,2,3,4  SUM3 =0.
*    USAGE   -        CALL INTSUM( X,4, SUM1)
*                     CALL XNTSUM(IX,4,ISUM1)
*                     CALL INTSUM( X,4, X)
*                     CALL INTSUM( X,1, SUM2)
*                     CALL INTSUM( X,0, SUM3)
*    OUTPUTS - SUM1(1...4) = 1., 3., 6., 10.   ISUM1(1...4) = 1,3,6,10
*                 X(1...4) = 1., 3., 6., 10.     SUM2 = 1.
*              SUM3 = 0.  (NO OUTPUT CASE)
*
* 2. INPUTS  - IX(1...3) = OCT 000000000001, 000000000002, 000000000003
*    USAGE   -        CALL XNTSUM(IX,3,IX)
*    OUTPUTS - IX(1...3) = OCT 000000000001, 000000000003, 000000000006
*
```

```
************************
*    INTSUM         *
************************
(PAGE  2)
```


```
      * PROGRAM FOLLOWS BELOW
      *
      *
      * NO TRANSFER VECTOR
             HTR     0             XR4
             BCI     1,INTSUM
      * PRINCIPAL ENTRY.   INTSUM(X,LX,XISUMD)
      INTSUM CLA     FAD
       SETUP STO     GET
             SXD     INTSUM-2,4
             CLA     1,4           A(X)
             STA     GET
             CLA     3,4           A(XISUMD)
             STA     STORE
             CLA*    2,4           LX
             TMI     LEAVE
             PDX     0,4
             TXL     LEAVE,4,0
             TXI     *+1,4,-1      LX-1
             SXD     TXL,4
             PXD     0,0           CLEAR AC
             PDX     0,4           AND XR4
      * LOOP
       GET   NOP                   FAD  **,4  OR  ADD  **,4  **=A(X)
       STORE STO     **,4          **=A(XISUMD)
             TXI     *+1,4,1
       TXL   TXL     GET,4,**      **=LX-1
      * EXIT
       LEAVE LXD     INTSUM-2,4
             TRA     4,4
      * SECOND ENTRY. XNTSUM (IX, LIX, IXISMD)
      XNTSUM CLA     ADD
             TRA     SETUP
      * CONSTANTS
       FAD   FAD     **,4
       ADD   ADD     **,4
             END
```

```
*     IPLYEV (SUBROUTINE)               10/2/64   LAST CARD IN DECK IS NO. 0083
*     LABEL
CIPLYEV
      SUBROUTINE IPLYEV(LA,A,X,Y,EVR,EVI)
C
C                      ----ABSTRACT----
C
C TITLE - IPLYEV
C       COMPLEX POLYNOMIAL EVALUATION
C
C              IPLYEV EVALUATES THE POLYNOMIAL WITH REAL COEFFICIENTS
C
C               EV = A(1)+A(2)*Z+A(3)*Z**2+....A(LA)*Z**(LA-1)
C
C              AT THE POINT   Z = X+IY.
C
C              IF X=COS(W) AND Y=SIN(W) FOR REAL W THEN THIS POLYNOMIAL
C              EVALUATION IS EQUIVALENT TO TAKING THE FOURIER TRANSFORM
C              OF A(I),I=1...LA AT THE FREQUENCY W.  (W=PI IS THE
C              FOLDING FREQUENCY).
C
C LANGUAGE  - FORTRAN II SUBROUTINE (USES COMPLEX ARITHMETIC)
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 98 REGISTERS
C SPEED     - ABOUT  126*LA + 86  MACHINE CYCLES ON THE 7090.
C AUTHOR    - R.A. WIGGINS, 9/26/62
C
C                      ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   NONE
C       AND FORTRAN SYSTEM ROUTINES -   (IFMP)
C
C FORTRAN USAGE
C     CALL IPLYEV(LA,A,X,Y,EVR,EVI)
C
C INPUTS
C
C    A(I)       I=1...LA ARE THE FLOATING POINT COEFFICIENTS OF THE
C                 POLYNOMIAL
C
C    LA         IS FORTRAN II INTEGER
C               MUST BE GRTHN= 2
C
C    X          IS THE REAL PART OF THE NUMBER AT WHICH THE POLYNOMIAL
C                 IS TO BE EVALUATED.
C
C    Y          IS THE IMAGINARY PART OF THE NUMBER AT WHICH THE
C                 POLYNOMIAL IS TO BE EVALUATED.
C
C OUTPUTS
C
C    EVR        IS THE REAL PART OF THE POLYNOMIAL EVALUATION.
C
C    EVI        IS THE COMPLEX PART OF THE POLYNOMIAL EVALUATION.
C
C EXAMPLES
C
C 1. INPUTS  - A(1...3)=3.,2.,1.  LA=3  X=1.  Y=0.
C    OUTPUTS - EVR=6.  EVI=0.
C
C 2. INPUTS  - A(1...3)=3.,2.,1.  LA=3   X=0.  Y=1.
C    OUTPUTS - EVR=2.  EVI=2.
C
C 3. INPUTS  - A(1...3)=3.,2.,1.  LA=3   X=1.  Y=1.
C    OUTPUTS - EVR=5.  EVI=4.
C
I     DIMENSION Z(1),A1(1),EV(1)
      DIMENSION A(5)
      Z(1)=X
      Z(2)=Y
      A1(2)=0.
      EV(1)=A(LA)
      EV(2)=0.
C*****
      DO 10  I=2,LA
```

```
          J=LA-I
          A1(1)=A(J+1)
     I    EV=A1+EV*Z
      10  CONTINUE
     C*****
          EVR=EV(1)
          EVI=EV(2)
          RETURN
          END
```

```
*      ITOMLI (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0097
*      FAP
*ITOMLI
       COUNT   100
       LBL     ITOMLI
       ENTRY   ITOMLI (IV,LIV,MLIV,IANS)
*
*                   ----ABSTRACT----
*
* TITLE - ITOMLI
*       FAST CONVERT FORTRAN INTEGER VECTOR TO MLI VECTOR
*
*             ITOMLI CONVERTS A FORTRAN INTEGER VECTOR TO A MACHINE
*             LANGUAGE INTEGER VECTOR.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 37 REGISTERS
* SPEED     - LENGTH OF VECTOR TIMES 8 MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON JR, MAY 1961
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL ITOMLI(IV,LIV,MLIV,IANS)
*
* INPUTS
*
*    IV(I)     I=1,2,...,LIV IS THE FORTRAN FIXED POINT VECTOR.
*
*    LIV       MUST EXCEED 0
*
* OUTPUTS
*
*    MLIV(I)   I=1,2,...,LIV IS THE MACHINE LANGUAGE FIXED POINT VECTOR.
*              MLIV MAY BE SET EQUIVALENT TO IV.
*
*    IANS      = 0 JOB DONE OK
*              =-1 LIV IS ILLEGAL
*
* EXAMPLES
*
* 1. INPUTS  - IV=1,-1,2,-2,10,-10  LIV=6
*    OUTPUTS - MLIV=OCT 1,400000000001,2,400000000002,12,400000000012
*              IANS=0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LIV=0
*    OUTPUTS - IANS=-1
*
* 3. INPUTS  - IV(1)=3  LIV=1
*    OUTPUTS - IANS=0   MLIV(1)=OCT3
*
       HTR     0
       BCI     1,ITOMLI
ITOMLI SXA     EXIT,1
       SXD     ITOMLI-2,4
       CLA     1,4            A(A(IV))
       ADD     K1
       STA     CLA
       CLA     2,4            A(A(LIV))
       STA     GET2
       CLA     3,4            A(A(MLIV))
       ADD     K1
       STA     STO
       CLA     4,4            A(A(IANS))
       STA     PUT4
* GET AND CHECK LIV.
       CLS     K1
       STO     IANS
  GET2 CLA     **             A(LIV)
       ARS     18
       STO     LIV
```

```
          TMI     LEAVE
          TZE     LEAVE
* LOOP
          CLA     K0
          STO     IANS
          LXA     LIV,1
     CLA  CLA     **,1           A(IV)+1
          ARS     18
     STO  STO     **,1           A(MLIV)+1
          TIX     CLA,1,1
* STORE IANS AND EXIT.
   LEAVE  CLA     IANS
          ALS     18
    PUT4  STO     **             A(IANS)
    EXIT  AXT     **,1
          TRA     5,4
* CONSTANTS
      K0  PZE     0
      K1  PZE     1
* VARIABLES
    IANS  PZE     **
     LIV  PZE     **
          END
```

```
*       IVTOHV (SUBROUTINE)              3/15/65   LAST CARD IN DECK IS NO. 0147
*       FAP
*IVTOHV
        COUNT   150
        LBL     IVTOHV
        ENTRY   IVTOHV (IV,LHV,HV)
*
*                    ----ABSTRACT----
*
* TITLE - IVTOHV
*         PACK UP FORTRAN INTEGER VECTOR AS HOLLERITH VECTOR
*
*               IVTOHV CONVERTS AN INTEGER VECTOR IV(I), I=1...6*LHV,
*               INTO A PACKED VECTOR HV(I), I=1...LHV.  THE BITS 12 THRU
*               17 OF EACH IV(I) ARE EXTRACTED (OTHER BITS ARE IGNORED).
*               6 GROUPS LIKE THIS FROM 6 SUCCESSIVE IV(I) REGISTERS
*               ARE PACKED INTO A SINGLE HV(I) WORD.
*
*               IVTOHV IS THE INVERSE OF SUBROUTINE HVTOIV
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 70 REGISTERS
* SPEED     - 59 + 67*LHV MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON, MARCH, 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*       CALL IVTOHV(IV,LHV,HV)
*
* INPUTS
*
*    IV(I)      I=1...6*LHV  IS AN ARBITRARY INTEGER VECTOR
*                   (ONLY BITS 12,...,17 ARE VISIBLE TO IVTOHV, I.E.
*                    POSITIVE FORTRAN INTEGERS LESS THAN= 63)
*
*    LHV        IS THE LENGTH OF THE OUTPUT HOLLERITH VECTOR
*               MUST EXCEED ZERO  (STRAIGHT RETURN IF NOT)
*
* OUTPUTS       (IVTOHV TURNS OFF THE AC OVERFLOW INDICATOR)
*
*    HV(I)      I=1...LHV  IS THE PACKED HOLLERITH
*                  E.G.  HV(1) CONTAINS  IV(1...6)  PACKED
*                        HV(2) CONTAINS  IV(7...12) PACKED
*                          ETC
*                  PACKING IS LEFT-TO-RIGHT (IV(1) OCCUPIES BITS 0...5)
*               EQUIVALENCE  (IV,HV)  IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - IV(1...18) = 19,24,17,41,17,19,51,21,41,50,
*                           48,51,38,48,39,17,19,34          LHV = 3
*    OUTPUTS - HV(1) = 6HCHARAC   (= OCT233021512123)
*              HV(2) = 6HTERS T   (= OCT632551626063)
*              HV(3) = 6HO PACK   (= OCT466047212342)
*
* 2. SHOWING MASKING BEHAVIOUR AND ILLEGAL LHV BEHAVIOUR
*    INPUTS  - IV(1...6) = -17,82,83,84,-85,22
*              (IE IV(1...6)=OCT400021000000,000122000000,000123000000,
*                             000124000000,400125000000,000026000000 )
*    USAGE   -       DIMENSION  HV(2), IV(6)
*                    CALL IVTOHV(IV,1,HV)
*                    CALL IVTOHV(IV,0,HV(2))
*    OUTPUTS - HV(1) = 6HABCDEF   (= OCT212223242526)
*              HV(2) IS UNDISTURBED
*
* PROGRAM FOLLOWS BELOW
        HTR     0
        HTR     0
        HTR     0
        BCI     1,IVTOHV
```

****
*   IVTOHV   *
****
(PAGE 2)

```
IVTOHV SXD     IVTOHV-2,4
       SXD     IVTOHV-3,2
       SXD     IVTOHV-4,1
* SETUP SEQUENCE
       CLA     1,4            A(IV)
       STA     C1
       SUB     K1
       STA     C2
       SUB     K1
       STA     C3
       SUB     K1
       STA     C4
       SUB     K1
       STA     C5
       SUB     K1
       STA     C6
       CLA*    2,4            LHV
       TMI     LEAVE
       TZE     LEAVE
       STD     TESTLH
       CLA     3,4            A(HV)
       ADD     K1
       STA     SLW
* (XR2) CONTROLS ACQUISITION, XR1 CONTROLS STORAGE)
       AXT     0,2
       AXT     1,1
* LOOP (STRAIGHT LINE PROGRAM FOR SPEED)
       LDQ     KO            (MUST BE ZERO FOR SHIFTS)
 NEXT6 STZ     WORD
    C1 CLA     **,2           **=A(IV)
       ANA     MSK
       LGL     12
       ACL     WORD
       SLW     WORD
    C2 CLA     **,2           **=A(IV)-1
       ANA     MSK
       LGL     6
       ACL     WORD
       SLW     WORD
    C3 CLA     **,2           **=A(IV)-2
       ANA     MSK
       ACL     WORD
       SLW     WORD
    C4 CLA     **,2           **=A(IV)-3
       ANA     MSK
       ARS     6
       ACL     WORD
       SLW     WORD
    C5 CLA     **,2           **=A(IV)-4
       ANA     MSK
       ARS     12
       ACL     WORD
       SLW     WORD
    C6 CLA     **,2           **=A(IV)-5
       ANA     MSK
       ARS     18
       ACL     WORD
   SLW SLW     **,1           **=A(HV)+1
* BUMP XRS AND CHECK COMPLETION
       TXI     *+1,2,6
       TXI     *+1,1,1
TESTLH TXL     NEXT6,1,**     **=LHV
       TOV     LEAVE         (C1+2 MAY CAUSE OVERFLOW)
* EXIT
 LEAVE LXD     IVTOHV-3,2
       LXD     IVTOHV-4,1
       TRA     4,4
* CONSTANTS, TEMPORARIES
    KO PZE     0
    K1 PZE     1
   MSK OCT     000077000000
  WORD PZE     **
       END
```

```
*     IXCARG (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0066
*     LABEL
CIXCARG
      SUBROUTINE IXCARG(ARG,IXCOM)
C
C                    ----ABSTRACT----
C
C TITLE - IXCARG
C       LOCATE ARGUMENT WITH RESPECT TO COMMON
C
C             IXCARG RETURNS THE LOCATION OF ITS FIRST ARGUMENT TO
C             THE CALLING PROGRAM, THE LOCATION BEING DETERMINED AS
C             THE INDEX OF THAT ARGUMENT WITH RESPECT TO THE FORTRAN
C             COMMON BLOCK.
C
C             THUS IXCARG PERMITS ACCESS TO LITERAL DATA IN A CALLING
C             SEQUENCE, A PRINCIPAL USE BEING TO LOCATE HOLLERITH DATA.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 35 REGISTERS
C SPEED     - 42 MACHINE CYCLES PLUS TWO CALLS OF XLOCF
C AUTHOR    - S.M. SIMPSON,  MARCH 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - XLOC
C
C FORTRAN USAGE
C     CALL IXCARG(ARG,IXCOM)
C
C INPUTS
C
C    ARG       IS THE ARGUMENT WHOSE LOCATION IS DESIRED
C
C OUTPUTS
C
C    IXCOM     IS THE INDEX OF ARG WITH RESPECT TO COMMON
C                 I.E. IF THE CALLING PROGRAM HAS THE FOLLOWING
C                 STATEMENTS
C                    DIMENSION CM(2)
C                    COMMON CM
C                 THEN CM(IXCOM) EQUALS ARG
C
C EXAMPLES
C
C 1. TYPICAL USE TO LOCATE LITERAL HOLLERITH DATA
C        (NOTE HOW OUTPUT SHOWS FAP-STYLE STORAGE WITH FENCE)
C    USAGE   -       DIMENSION CM(2)
C                    COMMON  CM
C                    CALL IXCARG(18HFIRST,SECOND,THIRD,IXCOM)
C    OUTPUTS - CM(IXCOM)     = 6HFIRST,
C              CM(IXCOM-1)   = 6HSECOND
C              CM(IXCOM-2)   = 6H,THIRD
C              CM(IXCOM-3)   = OCT 777777777777    (THIS IS THE FENCE)
C
C 2. LOCATION OF LITERAL CONSTANTS
C    USAGE   -       CALL IXCARG(3.14159265,IXCOM)
C    OUTPUTS - CM(IXCOM) = 3.14159265
C
C PROGRAM FOLLOWS BELOW
      DIMENSION CM(2)
      COMMON CM
      IXCOM = XLOCF(CM) - XLOCF(ARG) +1
      RETURN
      END
```

```
*     KIINT1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0128
*     LABEL
CKIINT1
      SUBROUTINE KIINT1 (CHISQ,NDF,PROB,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - KIINT1
C       PROBABILITY THAT A CHI-SQUARED VARIATE EXCEEDS A VALUE.
C
C             KIINT1 PRODUCES THE PROBABILITY THAT A CHI-SQUARED VARIATE
C             WILL EXCEED A GIVEN VALUE. THIS PROBABILITY IS COMPUTED BY
C             EQUATIONS GIVEN BY YULE AND KENDALL, 1950, THEORY OF
C             STATISTICS, PAGE 464 (FOOTNOTE) FOR NDF LESS THAN 31,
C             WHERE NDF = NO. DEGREES OF FREEDOM.
C             FOR HIGHER NDF THE NORMAL APPROXIMATION IS USED.
C             WHEN THE NORMAL APPROXIMATION IS USED A TABLE OF THE
C             NORMAL DISTRIBUTION WHICH APPEARS IN SUBROUTINE NOINT1 IS
C             USED AND, SINCE THIS TABLE HAS ONLY 201 VALUES
C             CORRESPONDING TO VALUES OF X (UNIT NORMAL) FROM
C             0.0 TO 4.0, PROBABILITIES LESS THAN .00032 ARE SET TO ZERO
C             AND THOSE GREATER THAN 99968 ARE SET EQUAL TO ONE. THIS
C             DOES NOT OCCUR IF THE EQUATIONS ARE USED.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 191 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NOINT1
C       AND FORTRAN SYSTEM ROUTINES - SQRT, EXP(3
C
C FORTRAN USAGE
C     CALL KIINT1(CHISQ,NDF,PROB,IANS)
C
C INPUTS
C
C   CHISQ     IS THE PARTICULAR VALUE OF A CHI-SQUARED VARIATE.
C             MUST BE GRTHN=0.
C
C   NDF       IS THE NUMBER OF DEGREES OF FREEDOM OF THE VARIATE.
C             MUST BE GRTHN 0.
C
C OUTPUTS
C
C   PROB      IS THE PROBABILITY THAT THE VARIATE GRTHN=CHISQ.
C
C   IANS      =0  NORMAL
C             =1  ILLEGAL CHISQ
C             =2  ILLEGAL NDF
C
C EXAMPLES
C
C      THE AGREEMENT BETWEEN THE PROB VALUE IN THE EXAMPLES AND THE
C     COMPUTED PROB VALUE IS TO 3 OR FOUR PLACES SINCE 4 PLACE TABLES
C     WERE USED TO MAKE UP THE EXAMPLES.
C
C 1. INPUTS  - NDF=1   CHISQ=-1.
C   OUTPUTS - ERROR   IANS=1
C
C 2. INPUTS  - NDF=0   CHISQ=1.
C   OUTPUTS - ERROR   IANS=2
C
C 3. INPUTS  - NDF=1   CHISQ=1.
C   OUTPUTS - PROB=.3179   IANS=0
C
C 4. INPUTS  - NDF=8   CHISQ=2.7330
C   OUTPUTS - PROB=.95   IANS=0
C
C 5. INPUTS  - NDF=21   CHISQ=38.932
C   OUTPUTS - PROB=.01   IANS=0
C
```

```
C 6. INPUTS  - NDF=30   CHISQ=43.773
C    OUTPUTS - PROB=.05   IANS=0
C
C 7. INPUTS  - NDF=31   CHISQ=17.
C    OUTPUTS - PROB=.98   IANS=0
C
C 8. INPUTS  - NDF=3   CHISQ=2.366
C    OUTPUTS - PROB=.50   IANS=0
C
C
C INITIALIZE AND CHECK IF NORMAL CURVE APPROXIMATION IS TO BE USED.
      IANS=1
      IF(CHISQ)9999,10,10
 10   IANS=2
      IF(NDF) 9999,9999,12
 12   IANS=0
   15 CHI=SQRTF(CHISQ)
      IF (NDF-30)  20,20,70
C PROB IS COMPUTED IN THE FORM PROB = P1+P2*P3. CHECK NDF FOR EVEN, ODD.
   20 P2=(2.71828183)**(-CHISQ/2.0)
      NDFH=NDF/2
      IF (NDF-2*NDFH)  25,25,30
C EVEN.  SET P1=0, AND P3=1.0 IF NDF=2.
   25 P1=0.0
      IF (NDF-2)  27,27,50
   27 P3=1.0
      GO TO 60
C ODD.  COMPUTE P1, MODIFY P2 AND SET P3=0.0 IF NDF=1.
   30 CALL NOINT1(CHI,P1)
      P1=2.0*(1.0-P1)
      P2=CHI*P2*.79788480
      IF (NDF-1)  35,35,50
   35 P3=0.0
      GO TO 60
C EVALUATE P3 AS A POLYNOMIAL FOR NDF GREATER THAN 2.
   50 NLOOPS=NDFH-1
      P3=1.0
C IF NDF=3 (NLOOPS=0), P3=1.
      IF(NLOOPS)  60,60,52
 52   DIV=NDF-2
      DO 55  I=1,NLOOPS
      P3=P3*CHISQ/DIV+1.0
   55 DIV=DIV-2.0
      GO TO 60
C COMBINE PIECES TO FORM PROB.
   60 PROB=P1+P2*P3
      GO TO 9999
C USE NORMAL APPROXIMATION FOR NDF GREATER THAN 30.
   70 CHIMOD=CHI*1.414214-SQRTF(FLOATF(NDF)*2.0-1.0)
      CALL NOINT1(CHIMOD,P1)
      PROB=1.0-P1
      GO TO 9999
 9999 RETURN
      END
```

```
*      KOLAPS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0218
*      FAP
*KOLAPS
       COUNT   200
       LBL     KOLAPS
       ENTRY   KOLAPS (XMID,M,TYPE,L,CMID,ERR)
*
*                    ----ABSTRACT----
*
* TITLE - KOLAPS
*       COLLAPSE ODD-LENGTHED VECTOR ABOUT ITS MIDPOINT
*
*             KOLAPS REDUCES A VECTOR X(I) I=-M,....,0,....,M TO ANOTHER
*             VECTOR C(I) I= -L,....,0,....,L BY THE OPERATION
*                  C(I) = X(I)+X(I+2*L)+X(I-2*L)+X(I+4*L)+X(I-4*L)+...
*                         FOR I= -(L-1),....,0,....,L-1
*                     WHERE SUMMATION TERMINATES AS X SERIES TERMINATES
*                  C(L) = C(-L) = ONE-HALF VALUE FROM ABOVE EXPRESSION
*             KOLAPS HANDLES BOTH FIXED AND FLOATING POINT VECTORS.
*             OUTPUT MAY BE STORED ON TOP OF INPUT.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 100 REGISTERS
* SPEED     - ABOUT 12*M  MACHINE CYCLES, FOR FIXED PT. DATA
*             ABOUT 21*M  MACHINE CYCLES, FOR FLOATING PT. DATA
* AUTHOR    - J. CLARK   10/61
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL KOLAPS(XMID,M,TYPE,L,CMID,ERR)
*
* INPUTS
*
*    XMID(I)   CONTAINS THE TWO-SIDED VECTOR X(J) J= -M,....,0,....,M
*                SUCH THAT XMID(1) = X(0) , I.E.
*                       XMID(I) = X(I-1)     I=-M+1,...,M+1
*              XMID MAY BE FLOATING POINT OR FIXED POINT
*
*    M         DEFINES LENGTH OF X TO BE 2*M+1
*              MUST NOT BE NEGATIVE
*
*    TYPE      = 0.0 SIGNIFIES X(I) IS FIXED POINT
*              NOT = 0.0 SIGNIFIES X(I) IS FLOATING POINT
*
*    L         DEFINES LENGTH OF COLLAPSED VECTOR TO BE 2*L+1
*              MUST EXCEED ZERO. MAY EXCEED M.
*
* OUTPUTS
*
*    CMID(I)   CONTAINS THE COLLAPSED VECTOR C(J) J= -L,...,L
*                SUCH THAT CMID(1) = C(0)  I.E.
*                       CMID(I) = C(I-1)    I = -L+1,...,L+1
*                WHERE C(J) IS DEFINED IN ABSTRACT ABOVE
*              EQUIVALENCE (XMID,CMID) IS PERMITTED
*
*    ERR       = 0.0 NORMALLY
*              = 1.0 IF L OR M IS ILLEGAL
*              = 2.0 IF OVERFLOW OCCURS
*
* EXAMPLES
*
*              IN ALL EXAMPLES, INPUTS ARE ASSUMED TO BE THE SAME AS
*              EXAMPLE 1. UNLESS OTHERWISE STATED
*
* 1. ORDINARY USAGE (FIXED OR FLOATING)
*    INPUTS  - XX(1...9) = 1.,3.,2.,1.,3.,5.,1.,1.,1.
*              IXX(1...9)= 10,30,20,10,30,50,10,10,10
*    USAGE   -       CALL KOLAPS(XX(5),4,1.0,2,CC(3),ERR1)
*                    CALL KOLAPS(IXX(5),4,0.,2,ICC(3),ERR2)
*    OUTPUTS - CC(1...5)  = 1.5,2.,5.,8.,1.5    ERR1=0.
```

```
*              ICC(1...5) = 15,20,50,80,15      ERR2=0.
*
* 2. STORAGE OF OUTPUT ON TOP OF INPUT (FIXED OR FLOATING)
*    USAGE   -          CALL KOLAPS(XX(5),4,1.0,2,XX(5),ERR1)
*                       CALL KOLAPS(IXX(5),4,0.,2,IXX(5),ERR2)
*    OUTPUTS - XX(1...9) = 1.,3.,1.5,2.,5.,8.,1.5,1.,1.
*              IXX(1...9) = 10,30,15,20,50,80,15,10,10
*
* 3. SPECIAL CASE - L=M   (FLOATING)
*    USAGE   -          CALL KOLAPS(XX(3),2,1.,2,CC(3),ERR)
*    OUTPUTS - CC(1...5) = 2.,3.,2.,1.,2.     ERR=0.
*
* 4. SPECIAL CASES - L EXCEEDS M AND M=0
*    USAGE   -          CALL KOLAPS(XX(3),2,1.,4,CC(5),ERR1)
*                       CALL KOLAPS(IXX,0,0.,2,ICC(3),ERR2)
*    OUTPUTS - CC(1...9) = 0.,0.,1.,3.,2.,1.,3.,0.,0.   ERR1 = 0.
*              ICC(1...5) = 0,0,10,0,0
*
* 5. ERROR CONDITIONS
*    USAGE   -          CALL KOLAPS(XX,-1,1.,2,CC,ERR1)
*              OR       CALL KOLAPS(XX,0,1.,0,CC,ERR2)
*    OUTPUTS - ERR1 = 1. (ILLEGAL M)
*              ERR2 = 1. (ILLEGAL L)
*
* 6. INPUTS  - IXX(1...5) = 90000,90000,90000,90000,90000
*    USAGE   -          CALL KOLAPS(IXX(3),2,0.,1,ICC(2),ERR)
*    OUTPUTS - ERR = 2. (OVERFLOW)
*
       HTR     0
       BCI     1,KOLAPS
KOLAPS SXD     KOLAPS-2,4
       SXA     LEAVE+1,1
       SXA     LEAVE+2,2
* GET L AND CHECK IT (MUST EXCEED ZERO)
       CLA     KF1
       STO     ERR
       CLA*    4,4
       TMI     LEAVE
       TZE     LEAVE
       STD     KL
* SET UP FOR FIXED OR FLOATING
       AXT     0,1
       ZET*    3,4
       AXT     -1,1
       CLA     KADD1,1
       STO     NOP2
       CLA     KADD2,1
       STO     NOP3
       CLA     KLRS,1
       STO     NOP4
* SET DECREMENTS ETC. DEPENDING ON L,M
       CLA     KL
       STD     TXI3            L
       PDC     0,1
       SXD     TXI4,1          -L
       ADD     KL
       STD     TXI1            2L
       PDC     0,1
       SXD     TXI2,1          -2L
       SUB     KD1
       STD     TXL1            2L-1
       CLA*    2,4
       TMI     LEAVE           (ILLEGAL M EXIT)
       STD     TXH1            M
       STD     KTXH
       ADD     KD1
       PDC     0,1             -M-1
       SXD     TXH2,1
       SXD     KTXL,1
       CLA     KTXL
       STO     NOP1
* SET ADDRESS XMID,CMID
       CLA     1,4
       STA     NOP2
       STA     NOP3
```

```
       CLA    5,4
       STA    STO
       STA    CLA
       STA    STQ1
       STA    STQ2
 * MAIN LOOP. SETS C(I) I=-L,...,L-1
 *  NOTES - XR4 CONTROLS I
 *        - XR1 CONTROLS X(I),X(I-2L),... (XR1 GETS BUMPED DOWN)
 *        - XR2 CONTROLS X(I+2L),X(I+4L),...  (XR2 GETS BUMPED UP)
 *SUMMATION IN PAIRS BY XR1,XR2 MEANS,WHEN XR1 EXCEEDS BOUNDS SO WILL XR2
       CLA    KF2
       STO    ERR
       TOV    *+1
       LDC    KL,4          START WITH I=-L (XR4=-L)
 * OUTER LOOP
  OUTR CLA    K0            CLEAR AC
       PXA    0,4
       PAX    0,1           INITIALIZE XR1 AND XR2 TO I
       PAX    0,2
 * (CHECK IF FIRST X(I) IS OUTSIDE RANGE.  IF SO, STORE ZERO)
  NOP1 NOP                  = TXL  STO,1,-M-1 FOR I=-L,...,-I
 *                          = TXH  STO,1,M FOR I=0,1,...,L-1
 * INNER LOOP
  TXI1 TXI    *+1,2,**      **=2L (BUMP XR2 UP, LOWEST=+L)
  NOP2 NOP                  ADD  **,1  OR  FAD  **,1   **=XMID
  TXH1 TXH    *+2,2,**      **=M
  NOP3 NOP                  ADD  **,2  OR  FAD  **,2   **=XMID
  TXI2 TXI    *+1,1,**      ** =-2L  (XR1 IS NEG FOR ALL TESTS)
  TXH2 TXH    TXI1,1,**     **=-M-1
 * STORE AND CHECK FOR MORE
  STO  STO    **,4          **=CMID
       TXI    *+1,4,1       BUMP XR4
       TXH    TXI3,4,0
       CLA    KTXH          SWITCH TEST ON FIRST X(I)
       STO    NOP1          FOR I=0 ON
  TXI3 TXI    *+1,4,**      **=+L      CHECK
  TXL1 TXL    *+2,4,**      **=2L-1    FOR
       TRA    DONE                     COMPLETION
  TXI4 TXI    OUTR,4,**     **=-L      BACK
 * PATCH UP ENDS
  DONE LDC    KL,1
       LXD    KL,4
  CLA  CLA    **,1          **=CMID
  NOP4 NOP                  =LRS 36  OR FDP KF2
  STQ1 STQ    **,1          **=CMID
  STQ2 STQ    **,4          **=CMID
       TOV    LEAVE
       STZ    ERR
 LEAVE LXD    KOLAPS-2,4
       AXT    **,1
       AXT    **,2
       CLA    ERR
       STO*   6,4
       TRA    7,4
  K0   PZE    0
  KF2  DEC    2.0
  KF1  DEC    1.0
  KL   PZE    0,0,**        **=L
  KD1  PZE    0,0,1
  KLRS LRS    36            THE STORAGE
       FDP    KF2           ORDER
  KADD1 ADD   0,1           OF
       FAD    0,1           THESE SIX
  KADD2 ADD   0,2           IS
       FAD    0,2           IMPORTANT
  KTXL TXL    STO,1,**      **=-M-1
  KTXH TXH    STO,1,**      **=M
  ERR  PZE    **            **=ERR SETTING = 1.0,2.0,0.0
       END
```

```
*     LIMITS (SUBROUTINE)               9/8/64   LAST CARD IN DECK IS NO. 0161
*     FAP
*LIMITS
      COUNT   150
      LBL     LIMITS
      ENTRY   LIMITS (IANSX1,IANS, X1,X1A,X1B, X2,X2A,X2B, ..J.,
*                     XN,XNA,XNB)
*
*
*                     ----ABSTRACT----
*
* TITLE - LIMITS
*       CHECK THAT VARIABLES FROM LIST FALL WITHIN GIVEN LIMITS
*
*             LIMITS  IS A VARIABLE-LENGTH-CALLING-SEQUENCE PROGRAM IN
*             WHICH THE ARGUMENTS BEYOND THE SECOND OCCUR IN TRIPLETS.
*             LIMITS  CHECKS TO SEE IF THE FIRST MEMBER OF EACH TRIPLET
*             LIES IN THE INCLUSIVE RANGE DEFINED BY THE NEXT TWO
*             MEMBERS.  IF THIS HOLDS FOR ALL TRIPLETS, THEN LIMITS
*             SETS ITS SECOND ARGUMENT EQUAL TO ZERO.  IF NOT, THEN ITS
*             SECOND ARGUMENT IS SET EQUAL TO THE FIXED POINT SUM OF
*             ITS FIRST ARGUMENT PLUS ONE LESS THAN THE INDEX OF THE
*             FIRST TRIPLET FOUND TO FAIL THE TEST.
*
*             THE TRIPLET ARGUMENTS MAY BE ANY MODE, AND +0 IS TREATED
*             EQUAL TO -0 IN THE COMPARISONS.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 44 REGISTERS
* SPEED      - ABOUT 26 + 43N MACHINE CYCLES
*                 WHERE N = NUMBER OF TRIPLETS
* AUTHOR     - S.M. SIMPSON,  JUNE 1964
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*      AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*    CALL LIMITS(IANSX1,IANS, X1,X1A,X1B, X2,X2A,X2B, ..J, XN,XNA,XNB)
*
*
* INPUTS
*
*   IANSX1    IS ANY FIXED POINT NUMBER, PREFERABLY GRTHN= 1, WHICH IS
*                TO BE THE OUTPUT VALUE OF  IANS  IN THE CASE THAT  X1
*                FAILS TO LIE WITHIN  X1A  TO  X1B.
*
*    X1       IS ANY MODE.
*    X1A      SHOULD BE SAME MODE AS  X1.
*    X1B      SHOULD BE SAME MODE AS  X1  (MAY BE GRTHN, LSTHN OR EQUAL
*                TO  X1A).
*
*    X2       IS ANY MODE, NOT NECESSARILY THE SAME AS  X1.
*    X2A      SAME MODE AS  X2.
*    X2B      SAME MODE AS  X2.
*
*          (ETC UP THRU XN,XNA,XNB WHERE N SHOULD EXCEED ZERO)
*
*
* OUTPUTS     ILLEGAL RETURN OCCURS IF ARGUMENT COUNT MINUS 2 IS NOT A
*                MULTIPLE OF 3.
*
*             LET  XJ,XJA,XJB  STAND FOR  J-TH  TRIPLET,  J=1,2,..J.,N
*             AND  LET  XJLO = MIN(XJA,XJB),  XJHI = MAX(XJA,XJB).
*             THEN
*
*   IANS      = 0            IF  XJLO LSTHN= XJ LSTHN= XJHI,  FOR ALL J.
*             = IANSX1+K-1   IF  XK  FAILS TO SATISFY ABOVE EQUATION,
*                  WHERE K IS THE LOWEST J VALUE FOR WHICH FAILURE
*                  OCCURS.
*
*
```

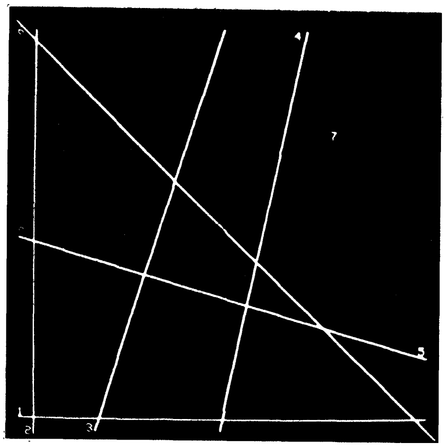
```
     * EXAMPLES
     *
     * 1. ZERO TESTS
     *    USAGE   -       CALL LIMITS(1,IANS, -0,-0,1, -0,+0,1, +0,-0,1,
     *                   1   +0,+0,1, -0,-1,-0, -0,-1,+0, +0,-1,-0, +0,-1,+0,
     *                   2   +0,+0,+0, +0,+0,-0, +0,-0,+0, +0,-0,-0,
     *                   3   -0,+0,+0, -0,+0,-0, -0,-0,+0, -0,-0,-0)
     *    OUTPUTS - IANS = 0
     *
     * 2. GENERAL TESTS
     *    USAGE   -       CALL LIMITS(1,IANS1, 1.0,2.0,3.0)
     *                    CALL LIMITS(21,IANS2, 3,1,4, 3.,1.,4., -3.,-4.,-1.,
     *                   1             1,1,4, 1,2,3, 4,1,4)
     *                    CALL LIMITS(31,IANS3, 0.,0.,0., 1,1,1, -1,-1,-1,
     *                   1             3,1,2, 0,1,2)
     *    OUTPUTS - IANS1 = 1,  IANS2 = 25,  IANS3 = 34
     *
     * 3. USAGE   - SAME AS EXAMPLE 2. BUT REVERSING THE ORDER OF THE SECOND
     *                 AND THIRD MEMBER OF EACH TRIPLET.
     *    OUTPUTS - SAME AS EXAMPLE 2.
     *
     *
     * PROGRAM FOLLOWS BELOW
     *
            BCI     1,LIMITS
     *
     * ONLY ENTRY.  LIMITS(IANSX1,IANS, X1,X1A,X1B, X2,X2A,X2B, ...)
     *
     LIMITS SXA     DONE,1
            CLA     1,4            A(IANSX1)
            STA     ADD
            CLA     2,4            A(IANS)
            STA     STO
            AXT     0,1            XR1 IS TRIPLET INDEX MINUS 1
            STZ*    2,4            (INITIALIZE IANS TO ZERO)
     *
     * CHECK FOR ANOTHER TRIPLET
     *
      CAL   CAL     3,4
            ANA     AMASK
            LAS     TSXZ           IS C(3,4) A TSX X,0 INSTRUCTION
            TRA     DONE           NO
            TRA     CHECK          YES
      DONE  AXT     **,1           NO   (** = XR1 INITIAL)
            TRA     3,4
     *
     * COMPARE X AND XLO, UNLESS WE HAVE ALREADY FOUND A DISCREPANCY.
     *
      CHECK ZET*    STO
            TRA     TXI
            CLA*    5,4            X1B
            LDQ*    4,4            X1A
            TLQ     *+2
            XCA
            STO     XHI
            STQ     XLO
            CLA*    3,4            X
            TNZ     CAS1
            SSP                    (BIG ZERO FOR LOW CHECK)
      CAS1  CAS     XLO            X AGAINST XLO
            TRA     OKLO           OK
            TRA     OKLO           OK
     *
     * SET IANS FOR DISCREPANCY
     *
      BAD   PXD     0,1
      ADD   ADD     **             ** = A(IANSX1)
      STO   STO     **             ** = A(IANS)
            TRA     TXI
     *
     * COMPARE X AND XHI
     *
      OKLO  TNZ     CAS2
            SSM                    (LITTLE ZERO FOR HI CHECK)
      CAS2  CAS     XHI            X AGAINST XHI
```

```
************************
*    LIMITS            *
************************
               (PAGE  3)
```


```
      TRA    BAD          X TOO BIG
      NOP                 OK
TXI   TXI    *+1,1,1      OK,
      TXI    CAL,4,-3     TRY ANOTHER.
*
* CONSTANTS
*
AMASK OCT    777777700000
TSXZ  TSX    0,0
XLO   PZE    **,**,**
XHI   PZE    **,**,**
      END
```

```
*       LINE  (709) (SUBROUTINE)          9/29/64   LAST CARD IN DECK IS NO. 0192
*       FAP
*LINE    (709)
        COUNT   150
        LBL     LINE
        ENTRY   LINE  (X1,Y1,X2,Y2)
*
*                     ----ABSTRACT----
*
* TITLE - LINE  (709)
*         FAST, ARBITRARY STRAIGHT LINE SEGMENT ON SCOPE
*
*               LINE PLOTS A STRAIGHT LINE FROM A POINT (X1,Y1) TO A
*               POINT (X2,Y2) ON THE SCOPE.  THE PLOTTING DENSITY IS
*               ADJUSTED SO THAT THE SEPARATION BETWEEN INDIVIDUAL POINTS
*               WILL BE LSTHN=2.0 AND GRTHN=1.414 SCOPE UNITS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709  (MAIN FRAME AND SCOPE UNIT)
* STORAGE   - 91 REGISTERS
* SPEED     - MAXIMUM
* AUTHOR    - S.M. SIMPSON
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL LINE  (X1,Y1,X2,Y2)
*
* INPUTS
*    X1         IS X COORDINATE OF 1 END OF LINE TO BE PLOTTED.
*    Y1         IS Y COORDINATE OF 1 END OF LINE TO BE PLOTTED.
*    X2         IS X COORDINATE OF 2 END OF LINE TO BE PLOTTED.
*    Y2         IS Y COORDINATE OF 2 END OF LINE TO BE PLOTTED.
*
*               NOTES
*                 X1,Y1,X2,Y2  ARE FLOATING POINT NUMBERS.
*                              MUST BE LSTHN 1024.  GRTHN=0.
*                              IF ARE GRTHN=1024. OR LSTHN 0 NO LINE
*                                IS PLOTTED.
*
* OUTPUTS       LINE PLOTTED ON THE SCOPE
*
* EXAMPLES
* 1. INPUTS  - X1=10.  Y1=50.  X2=1000.  Y2=50.
*
* 2. INPUTS  - X1=50.  Y1=10.  X2=50.  Y2=1000.
*
* 3. INPUTS  - X1=200.  Y1=20.  X2=500.  Y2=1000.
*
* 4. INPUTS  - X1=700.  Y1=1000.  X2=500.  Y2=20.
*
* 5. INPUTS  - X1=1000.  Y1=200.  X2=10.  Y2=500.
*
* 6. INPUTS  - X1=0.  Y1=1023.  X2=1023.  Y2=0.
*
* 7. INPUTS  - X1=750.  Y1=750.  X2=750.  Y2=750.
*
*    OUTPUTS -  THE NUMBERS ON THE DISPLAY CORRESPOND TO THE EXAMPLE
```

```
*                  NUMBER.  IT IS PLOTTED NEAR THE X1,Y1 POINT.
*
*
```

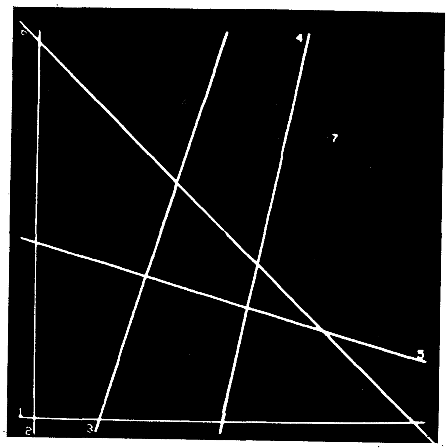


```
*
*
* 8. INPUTS  - X1=-4.  Y1=5.  X2=5.  Y2=5.
*
* 9. INPUTS  - X1=1024.15  Y1=5.  X2=5.  Y2=5.
*
*    OUTPUTS - NOTHING IS PLOTTED.
*
*SAVE INDEX REG AND CHECK LEGALITY OF ARGUMENTS
 LINE  SXA     LV,1
       CLA*    1,4
       TSX     CK,1
       CLA*    2,4
       TSX     CK,1
       CLA*    3,4
       TSX     CK,1
       CLA*    4,4
       TSX     CK,1
       TRA     SET           ALL OK
  CK   TMI     LV             BAD
       CAS     KL1024
       NOP                   BAD
       TRA     LV            BAD
       TRA     1,1            OK
*SET INITIAL X AND Y
  SET  CLA*    1,4            X1
       TSX     FX,1
       ALS     18
       STO     PTRND
       ALS     7
       STO     PTTRU
       CLA*    2,4             Y1
       TSX     FX,1
       ADD     PTRND
       STO     PTRND
       ANA     AN
       ALS     7
       ADD     PTTRU
       STO     PTTRU
*SET DELTA X, DELTA Y TIMES 2 EXP 7
       CLA*    3,4           X2
       FSB*    1,4           MINUS X1
       STO     DIFX
       CLA*    4,4           Y2
       FSB*    2,4           MINUS Y1
       STO     DIFY
*NO PTS PLOTTED WILL BE SET =(MAG(Y2-M)+MAG(X2-X1))/2+1
       CLA     DIFX
       SSP
       FAM     DIFY
```

```
        FDP     KL2
        XCA
        TSX     FX,1
        STO     NNCSX          STORE FXD PT NO. INCRS
        ORA     ORF
        FAD     ORF
        STO     NNCSL          STOR FLTG PT NO. INCRS
        LDQ     DIFX           FORM DELTA X * 2EXP7
        FMP     KL128
        FDP     NNCSL
        XCA
        TSX     FX,1
        ALS     18
        STO     XNTRU
        LDQ     DIFY           FORM DELTA Y * 2EXP7
        FMP     KL128
        FDP     NNCSL
        XCA
        TSX     FX,1
        STO     YNTRU
* SET FOR NO. PTS = = NO. INCRS PLUS 1
        LXA     NNCSX,1
        TXI     PLT,1,1
*PLOT LINE
 PLT    WTV
 CPY    CPY     PTRND
        CLA     PTTRU
        ADD     XNTRU
        ADD     YNTRU
        STO     PTTRU
        ARS     7
        ANA     AN2            GET RID OF EXTRA BITS
        STO     PTRND
        TIX     CPY,1,1
*EXIT
  LV    AXT     **,1
        TRA     5,4
 YNTRU  PZE     **,0,0          Y INC TIMES 2 EXP 7
 XNTRU  PZE     0,0,**          X INC TIMES 2 EXP 7
 PTTRU  PZE     **,0,**          X Y TIMES 2 EXP 7
 PTRND  PZE     **,0,**          X Y FOR SCOPE
  ORF   OCT     233000000000
  AN    OCT     000000777777
 DIFX   PZE     **              FLOATING POINT X2-X1
 DIFY   PZE     **              FLOATING POINT Y2 - Y1
 KL2    DEC     2.0
 NNCSL  PZE     **              = FLTG PT NO INCRS
 NNCSX  PZE     **              = FXD PT NO INCRS
 KL128  DEC     128.            = 2EXP7
KL1024  DEC     1024.0
  AN2   OCT     001777001777
  FX    UFA     ORF
        LRS
        ANA     AN
        LLS
        TRA     1,1
        END
```

```
*       LINE   (7090) (SUBROUTINE)         9/4/64   LAST CARD IN DECK IS NO. 0207
*       FAP
*LINE   (7090)
        COUNT   160
        LBL     LINE
        ENTRY   LINE (X1,Y1,X2,Y2)
*
*                    ----ABSTRACT----
*
* TITLE - LINE (7090)
*         FAST, ARBITRARY STRAIGHT LINE SEGMENT ON SCOPE
*
*               LINE PLOTS A STRAIGHT LINE FROM A POINT (X1,Y1) TO A
*               POINT (X2,Y2) ON THE SCOPE.  THE PLOTTING DENSITY IS
*               ADJUSTED SO THAT THE SEPARATION BETWEEN INDIVIDUAL POINTS
*               WILL BE LSTHN=2.0 AND GRTHN=1.414 SCOPE UNITS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090 (MAIN FRAME, DATA CHANNEL D AND SCOPE)
* STORAGE   - 95 REGISTERS
* SPEED     - HORIZONTAL LINE ACROSS ENTIRE SCOPE FACE TAKES ABOUT
*                                                  .13  SEC - 709
*                                                  .026 SEC - 7090
* AUTHOR    - S.M. SIMPSON
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*         AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*      CALL LINE  (X1,Y1,X2,Y2)
*
* INPUTS
*
*    X1         IS X COORDINATE OF 1 END OF LINE TO BE PLOTTED.
*
*    Y1         IS Y COORDINATE OF 1 END OF LINE TO BE PLOTTED.
*
*    X2         IS X COORDINATE OF 2 END OF LINE TO BE PLOTTED.
*
*    Y2         IS Y COORDINATE OF 2 END OF LINE TO BE PLOTTED.
*
*               NOTES
*                 X1,Y1,X2,Y2  ARE FLOATING POINT NUMBERS.
*                              MUST BE LSTHN 1024.  GRTHN=0.
*                              IF ARE GRTHN=1024. OR LSTHN 0 NO LINE
*                                IS PLOTTED.
*
* OUTPUTS       LINE PLOTTED ON THE SCOPE
*
* EXAMPLES
*
* 1. INPUTS  - X1=10.  Y1=50.  X2=1000.  Y2=50.
*
* 2. INPUTS  - X1=50.  Y1=10.  X2=50.  Y2=1000.
*
* 3. INPUTS  - X1=200.  Y1=20.  X2=500.  Y2=1000.
*
* 4. INPUTS  - X1=700.  Y1=1000.  X2=500.  Y2=20.
*
* 5. INPUTS  - X1=1000.  Y1=200.  X2=10.  Y2=500.
*
* 6. INPUTS  - X1=0.  Y1=1023.  X2=1023.  Y2=0.
*
* 7. INPUTS  - X1=750.  Y1=750.  X2=750.  Y2=750.
*    OUTPUTS -  THE NUMBERS ON THE DISPLAY CORRESPOND TO THE EXAMPLE
```

```
     *             NUMBER.  IT IS PLOTTED NEAR THE X1,Y1 POINT.
     *
```



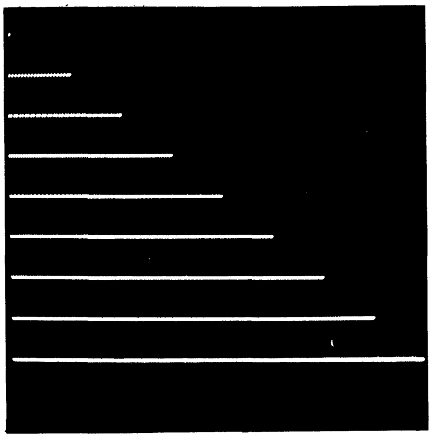
```
     *
     *
     *
     * 8. INPUTS  - X1=-4.  Y1=5.  X2=5.  Y2=5.
     *
     * 9. INPUTS  - X1=1024.15  Y1=5.  X2=5.  Y2=5.
     *    OUTPUTS - NOTHING IS PLOTTED.
     *
     *
     * PROGRAM FOLLOWS BELOW
     *
     * FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
     * TO CHANGE, ALTER THE LETTER DESIGNATION ONLY AND REASSEMBLE.
      X     TAPENO  D1
      SCPAD EQU     X-105
     *SAVE INDEX REG AND CHECK LEGALITY OF ARGUMENTS
            HTR     0
            BCI     1,LINE
      LINE  SXD     LINE-2,4
            SXA     LV,1
            CLA*    1,4
            TSX     CK,1
            CLA*    2,4
            TSX     CK,1
            CLA*    3,4
            TSX     CK,1
            CLA*    4,4
            TSX     CK,1
            TRA     SET           ALL OK
       CK   TMI     LV             BAD
            CAS     KL1024
            NOP                   BAD
            TRA     LV            BAD
            TRA     1,1            OK
     *SET INITIAL X AND Y
       SET  CLA*    1,4            X1
            TSX     FX,1
            ALS     18
            STO     PTRND
            ALS     7
            STO     PTTRU
            CLA*    2,4             Y1
            TSX     FX,1
            ADD     PTRND
            STO     PTRND
            ANA     AN
            ALS     7
            ADD     PTTRU
            STO     PTTRU
     *SET DELTA X, DELTA Y TIMES 2 EXP 7
            CLA*    3,4           X2
            FSB*    1,4            MINUS X1
```

****
* LINE (7090) *
****
(PAGE 3)

```
       STO     DIFX
       CLA*    4,4          Y2
       FSB*    2,4          MINUS Y1
       STO     DIFY
*NO PTS PLOTTED WILL BE SET =(MAG(Y2-M)+MAG(X2-X1))/2+1
       CLA     DIFX
       SSP
       FAM     DIFY
       FDP     KL2
       XCA
       TSX     FX,1
       STO     NNCSX        STORE FXD PT NO. INCRS
       ORA     ORF
       FAD     ORF
       STO     NNCSL        STOR FLTG PT NO. INCRS
       LDQ     DIFX         FORM DELTA X * 2EXP7
       FMP     KL128
       FDP     NNCSL
       XCA
       TSX     FX,1
       ALS     18
       STO     XNTRU
       LDQ     DIFY         FORM DELTA Y * 2EXP7
       FMP     KL128
       FDP     NNCSL
       XCA
       TSX     FX,1
       STO     YNTRU
* SET FOR NO. PTS = = NO. INCRS PLUS 1
       LXA     NNCSX,1
       TXI     PLT,1,1
*PLOT LINE
 PLT   WRS     SCPAD
       RCHX    IO
       CLA     PTTRU
       ADD     XNTRU
       ADD     YNTRU
       STO     PTTRU
       ARS     7
       ANA     AN2          GET RID OF EXTRA BITS
       STO     PTRND
       TIX     PLT,1,1
*EXIT
  LV   AXT     **,1
       TRA     5,4
 IO    IOCD    PTRND,0,1
 YNTRU PZE     **,0,0        Y INC TIMES 2 EXP 7
 XNTRU PZE     0,0,**        X INC TIMES 2 EXP 7
 PTTRU PZE     **,0,**        X Y TIMES 2 EXP 7
 PTRND PZE     **,0,**        X Y FOR SCOPE
  ORF  OCT     233000000000
  AN   OCT     000000777777
 DIFX  PZE     **            FLOATING POINT X2-X1
 DIFY  PZE     **            FLOATING POINT Y2 - Y1
 KL2   DEC     2.0
 NNCSL PZE     **            = FLTG PT NO INCRS
 NNCSX PZE     **            = FXD PT NO INCRS
 KL128 DEC     128.          = 2EXP7
KL1024 DEC     1024.0
  AN2  OCT     001777001777
  FX   UFA     ORF
       LRS     0
       ANA     AN
       LLS     0
       TRA     1,1
       END
```

```
*       LINEH (709) (SUBROUTINE)          9/29/64   LAST CARD IN DECK IS NO. 0157
*       FAP
*LINEH  (709)
        COUNT   150
        LBL     LINEH
        ENTRY   LINEH (NXLEFT, NYLEFT, NXRITE, NDELX)
*
*                   ----ABSTRACT----
*
* TITLE- LINEH (709)
*         PLOT FAST HORIZONTAL LINE ON SCOPE
*
*               LINEH HAS ARGUMENTS NXLEFT, NYLEFT, NXRITE, NDELX. IT
*               PLOTS A HORIZONTAL LINE ON THE SCOPE WITH LEFT END
*               COORDINATES (NXLEFT,NYLEFT), AND RIGHT END COORDINATES
*               (NXRITE,NYLEFT). THE SPACING OF THE POINTS WHICH COMPRISE
*               THE LINE IS NDLEX. THE LINE IS PLOTTED FROM LEFT TO RIGHT
*               BY PLOTTING THE POINTS (NXLEFT+K*NDELX,NYLEFT) FOR K=0,1,
*               2,...,M WHERE M*NDELX IS LESS THAN OR = TO NXRITE, AND
*               (K+1)*NDELX IS GREATER THAN NXRITE. IF =,LINE IS FINISHED.
*               IF LESS, ONE MORE POINT WILL BE PLOTTED WITH COORDINATES
*               (NXRITE,NYLEFT). NOTE INPUT VALUE RESTRICTIONS LISTED
*               UNDER INPUTS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 WITH SCOPE
* STORAGE   - 34 DECIMAL REGISTERS
* SPEED     - .5+.141*(LENGTH OF LINE/PLOTTING INCREMENT) MACHINE CYCLES
* AUTHOR    - J.N. GALBRAITH, MAY 10, 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL  LINEH(NXLEFT, NYLEFT, NXRITE, NDELX)
*
* INPUTS
*
*    NXLEFT    IS THE X COORDINATE OF THE LEFT END OF THE LINE.
*
*    NYLEFT    IS THE Y COORDINATE OF THE LEFT END OF THE LINE.
*
*    NXRITE    IS THE X COORDINATE OF THE RIGHT END OF THE LINE.
*                ABOVE COORDINATES ARE INTEGERS IN THE DECREMENT AND ARE
*                ASSUMED TO BE IN SCOPE UNITS (BETWEEN 0 AND 1023)
*
*    NDELX     IS THE PLOTTING INCREMENT. IT DETERMINES THE SPACING OF
*                THE POINTS WHICH MAKE THE LINE. A LARGE NDELX WILL
*                PLOT A DOTTED LINE. NDELX AN INTEGER IN THE DECREMENT.
*
*              NO POINT IS PLOTTED IF NXLEFT IS GREATER THAN NXRITE, AND
*              NO POINT IS PLOTTED IF NDELX=0 EXCEPT WHEN NXLEFT=NXRITE.
*              IN THIS CASE THE POINT (NXLEFT,NYLEFT) IS PLOTTED. NO
*              ERROR INDICATORS ARE SET FOR THESE CASES AND NO CHECK IS
*              MADE ON THE MAGNITUDES OF THE INPUT VALUES. QUANTITIES
*              GREATER THAN 1023 ARE PLOTTED MODULO 1024.
*
* OUTPUTS
*              HORIZONTAL LINE ON SCOPE.
*
* EXAMPLES
*
* 1. INPUTS  - NXLEFT=0, NYLEFT=0, NXRITE=1023, NDELX=1
*    OUTPUTS - LINE ON SCOPE (LOWER LINE IN PICTURE)
*
* 2. INPUTS  - NXLEFT=0, NYLEFT=100, NXRITE=900, NDELX=2
*    OUTPUTS - LINE ON SCOPE (SECOND LINE FROM BOTTOM IN PICTURE)
*
* 3. INPUTS  - NXLEFT=0, NYLEFT=200, NXRITE=775, NDELX=3
*    OUTPUTS - LINE ON SCOPE (THIRD FROM BOTTOM IN PICTURE)
*
* 4. INPUTS  - NXLEFT=0, NYLEFT=300, NXRITE=650, NDELX=4
```

```
*    OUTPUTS - LINE ON SCOPE (FOURTH FROM BOTTOM IN PICTURE)
*
* 5. INPUTS  - NXLEFT=0, NYLEFT=400, NXRITE=525, NDELX=5
*    OUTPUTS - LINE ON SCOPE (FIFTH FROM BOTTOM IN PICTURE)
*
* 6. INPUTS  - NXLEFT=0, NYLEFT=500, NXRITE=400, NDELX=6
*    OUTPUTS - LINE ON SCOPE (SIXTH FROM BOTTOM IN PICTURE)
*
* 7. INPUTS  - NXLEFT=0, NYLEFT=600, NXRITE=275, NDELX=7
*    OUTPUTS - LINE ON SCOPE (SEVENTH FROM BOTTOM IN PICTURE)
*
* 8. INPUTS  - NXLEFT=0, NYLEFT=700, NXRITE=150, NDELX=8
*    OUTPUTS - LINE ON SCOPE (EIGHTH FROM BOTTOM IN PICTURE)
*
* 9. INPUTS  - NXLEFT=0, NYLEFT=800, NXRITE=0, NDELX=0
*    OUTPUTS - POINT ON SCOPE (800 SCOPE UNITS UP IN PICTURE)
*
*10. INPUTS  - NXLEFT=0, NYLEFT=900, NXRITE=10, NDELX=0
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 900 SCOPE UNITS UP)
*
*11. INPUTS  - NXLEFT=100, NYLEFT=1000, NXRITE=10, NDELX=1
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 1000 SCOPE UNITS UP)
*
*     PICTURE OF SCOPE OUTPUT APPEARS BELOW.
*
```



```
*
*
*
      PZE
      BCI     1,LINEH
LINEH SXA     BACK,1
      SXA     BACK+1,2
      SXD     LINEH-2,4
      CLA*    2,4             Y COORD.
      ARS     18
      STA     POINT
      CLA*    3,4             X RIGHT
      SUB*    1,4             X LEFT
      TZE     LAST
      TMI     BACK
      PDX     ,1
      CLA*    4,4
      TZE     BACK
      STD     END
      STD     INCR
      CLA*    1,4
      STD     POINT
      PDX     ,2
LOOP  WTV
      CPY     POINT
INCR  TXI     *+1,2,**
      SXD     POINT,2
END   TIX     LOOP,1,**
      TXL     BACK,1,0
```

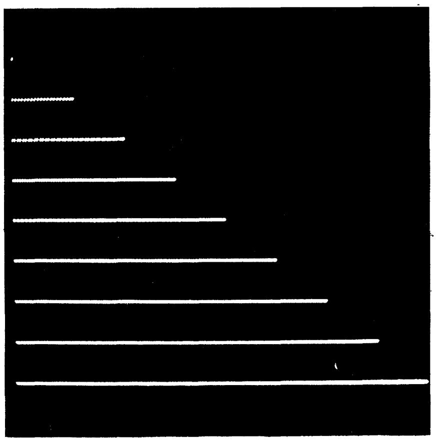
****
*   LINEH (709)   *
****
(PAGE 3)

```
      LAST  CLA*    3,4
            STD     POINT
            WTV
            CPY     POINT
      BACK  AXT     **,1
            AXT     **,2
            TRA     5,4
      POINT PZE     0
            END
```

```
*       LINEH (7090) (SUBROUTINE)          9/4/64   LAST CARD IN DECK IS NO. 0167
*       FAP
*LINEH (7090)
        COUNT   140
        LBL     LINEH
        ENTRY   LINEH (NXLEFT, NYLEFT, NXRITE, NDELX)
*
*                    ----ABSTRACT----
*
* TITLE - LINEH (7090)
*         PLOT FAST HORIZONTAL LINE ON SCOPE
*
*               LINEH HAS ARGUMENTS NXLEFT, NYLEFT, NXRITE, NDELX. IT
*               PLOTS A HORIZONTAL LINE ON THE SCOPE WITH LEFT END
*               COORDINATES (NXLEFT,NYLEFT), AND RIGHT END COORDINATES
*               (NXRITE,NYLEFT). THE SPACING OF THE POINTS WHICH COMPRISE
*               THE LINE IS NDLEX. THE LINE IS PLOTTED FROM LEFT TO RIGHT
*               BY PLOTTING THE POINTS (NXLEFT+K*NDELX,NYLEFT) FOR K=0,1,
*               2,...,M WHERE M*NDELX IS LESS THAN OR = TO (NXRITE-NXLEFT)
*               AND (M+1)*NDELX IS GREATER THAN (NXRITE-NXLEFT).  IF =,
*               LINE IS FINISHED.  IF LESS, ONE MORE POINT WILL BE PLOTTED
*               WITH COORDINATES (NXRITE,NYLEFT).  NOTE INPUT VALUE
*               RESTRICTIONS LISTED UNDER INPUTS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 7090 WITH SCOPE
* STORAGE   - 35 DECIMAL REGISTERS
* SPEED     - .5+.141*(LENGTH OF LINE/PLOTTING INCREMENT) MACHINE CYCLES
* AUTHOR    - J.N. GALBRAITH, MAY 10, 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL  LINEH(NXLEFT, NYLEFT, NXRITE, NDELX)
*
* INPUTS
*
*    NXLEFT    IS THE X COORDINATE OF THE LEFT END OF THE LINE.
*
*    NYLEFT    IS THE Y COORDINATE OF THE LEFT END OF THE LINE.
*
*    NXRITE    IS THE X COORDINATE OF THE RIGHT END OF THE LINE.
*                 ABOVE COORDINATES ARE INTEGERS IN THE DECREMENT AND ARE
*                 ASSUMED TO BE IN SCOPE UNITS (BETWEEN 0 AND 1023)
*
*    NDELX     IS THE PLOTTING INCREMENT. IT DETERMINES THE SPACING OF
*                 THE POINTS WHICH MAKE THE LINE. A LARGE NDELX WILL
*                 PLOT A DOTTED LINE. NDELX AN INTEGER IN THE DECREMENT.
*              NO POINT IS PLOTTED IF NXLEFT IS GREATER THAN NXRITE, AND
*              NO POINT IS PLOTTED IF NDELX=0 EXCEPT WHEN NXLEFT=NXRITE.
*              IN THIS CASE THE POINT (NXLEFT,NYLEFT) IS PLOTTED. NO
*              ERROR INDICATORS ARE SET FOR THESE CASES AND NO CHECK IS
*              MADE ON THE MAGNITUDES OF THE INPUT VALUES. QUANTITIES
*              GREATER THAN 1023 ARE PLOTTED MODULO 1024.
*
* OUTPUTS
*
*              HORIZONTAL LINE ON SCOPE.
*
* EXAMPLES
*
* 1. INPUTS  - NXLEFT=0, NYLEFT=0, NXRITE=1023, NDELX=1
*    OUTPUTS - LINE ON SCOPE (LOWER LINE IN PICTURE)
*
* 2. INPUTS  - NXLEFT=0, NYLEFT=100, NXRITE=900, NDELX=2
*    OUTPUTS - LINE ON SCOPE (SECOND LINE FROM BOTTOM IN PICTURE)
*
* 3. INPUTS  - NXLEFT=0, NYLEFT=200, NXRITE=775, NDELX=3
*    OUTPUTS - LINE ON SCOPE (THIRD FROM BOTTOM IN PICTURE)
*
* 4. INPUTS  - NXLEFT=0, NYLEFT=300, NXRITE=650, NDELX=4
```

LINEH (7090) (PAGE 2)

```
*    OUTPUTS - LINE ON SCOPE (FOURTH FROM BOTTOM IN PICTURE)
*
* 5. INPUTS  - NXLEFT=0, NYLEFT=400, NXRITE=525, NDELX=5
*    OUTPUTS - LINE ON SCOPE (FIFTH FROM BOTTOM IN PICTURE)
*
* 6. INPUTS  - NXLEFT=0, NYLEFT=500, NXRITE=400, NDELX=6
*    OUTPUTS - LINE ON SCOPE (SIXTH FROM BOTTOM IN PICTURE)
*
* 7. INPUTS  - NXLEFT=0, NYLEFT=600, NXRITE=275, NDELX=7
*    OUTPUTS - LINE ON SCOPE (SEVENTH FROM BOTTOM IN PICTURE)
*
* 8. INPUTS  - NXLEFT=0, NYLEFT=700, NXRITE=150, NDELX=8
*    OUTPUTS - LINE ON SCOPE (EIGHTH FROM BOTTOM IN PICTURE)
*
* 9. INPUTS  - NXLEFT=0, NYLEFT=800, NXRITE=0, NDELX=0
*    OUTPUTS - POINT ON SCOPE (800 SCOPE UNITS UP IN PICTURE)
*
*10. INPUTS  - NXLEFT=0, NYLEFT=900, NXRITE=10, NDELX=0
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 900 SCOPE UNITS UP)
*
*11. INPUTS  - NXLEFT=100, NYLEFT=1000, NXRITE=10, NDELX=1
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 1000 SCOPE UNITS UP)
*
*     PICTURE OF SCOPE OUTPUT APPEARS BELOW.
*
*
```



```
*
*
*
*
*
* PROGRAM FOLLOWS BELOW
*
* FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
* TO CHANGE, ALTER THE LETTER DESIGNATION ONLY, AND REASSEMBLE.
X     TAPENO  D1
SCPAD EQU     X-105
      PZE
      BCI     1,LINEH
LINEH SXA     BACK,1
      SXA     BACK+1,2
      SXD     LINEH-2,4
      CLA*    2,4          Y COORD.
      ARS     18
      STA     POINT
      CLA*    3,4          X RIGHT
      SUB*    1,4          X LEFT
      TZE     LAST
      TMI     BACK
      PDX     ,1
      CLA*    4,4
      TZE     BACK
      STD     END
      STD     INCR
```

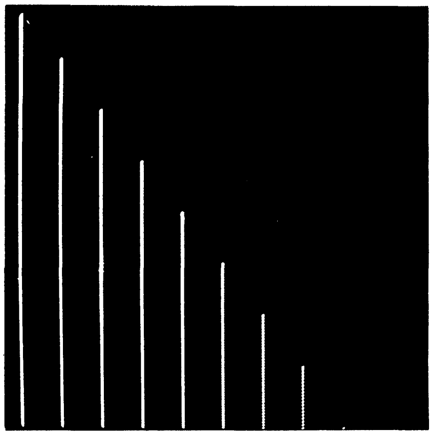
****
*   LINEH (7090)   *
****
(PAGE 3)

```
         CLA*    1,4
         STD     POINT
         PDX     ,2
LOOP     WRS     SCPAD
         RCHX    IO
INCR     TXI     *+1,2,**
         SXD     POINT,2
END      TIX     LOOP,1,**
         TXL     BACK,1,0
LAST     CLA*    3,4
         STD     POINT
         WRS     SCPAD
         RCHX    IO
BACK     AXT     **,1
         AXT     **,2
         TRA     5,4
POINT    PZE     0
IO       IOCD    POINT,0,1
         END
```

```
*      LINEV (709) (SUBROUTINE)         9/29/64   LAST CARD IN DECK IS NO. 0160
*      FAP
*LINEV (709)
       COUNT   150
       LBL     LINEV
       ENTRY   LINEV (NXBOT, NYBOT, NYTOP, NDELY)
*
*                    ----ABSTRACT----
*
* TITLE - LINEV (709)
*         PLOT FAST VERTICAL LINE ON SCOPE
*
*              LINEV HAS ARGUMENTS NXBOT, NYBOT, NYTOP, NDELY. IT PLOTS A
*              VERTICAL LINE ON THE SCOPE WITH BOTTOM COORDINATES
*              (NXBOT, NYBOT), AND TOP COORDINATES (NXBOT, NYTOP). THE
*              SPACING OF THE POINTS WHICH COMPRISE THE LINE IS NDELY.
*              THE LINE IS PLOTTED FROM BOTTOM TO TOP BY PLOTTING THE
*              POINTS (NXBOT, NYBOT+K*NDELY) FOR K=0,1,2,...,M WHERE
*              M*NDELY IS LESS THAN OR = TO NYTOP, AND (M+1)*NDELY IS
*              GREATER THAN NYTOP. IF =, LINE IS FINISHED. IF LESS, ONE
*              MORE POINT WILL BE PLOTTED WITH COORDINATES (NXBOT,NYTOP).
*              NOTE INPUT VALUE RESTRICTIONS LISTED UNDER INPUTS.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 WITH SCOPE
* STORAGE   - 34 DECIMAL REGISTERS
* SPEED     - .5+.141*(LENGTH OF LINE/PLOTTING INCREMENT) MILLISECONDS
* AUTHOR    - J.N. GALBRAITH , MAY 10, 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL  LINEV(NXBOT, NYBOT, NYTOP, NDELY)
*
* INPUTS
*
*    NXBOT     IS THE X COORDINATE OF THE BOTTOM OF THE LINE
*
*    NYBOT     IS THE Y COORDINATE OF THE BOTTOM OF THE LINE
*
*    NYTOP     IS THE Y COORDINATE OF THE TOP OF THE LINE
*                ABOVE COORDINATES ARE INTEGERS IN THE DECREMENT
*                AND ARE ASSUMED TO BE IN SCOPE UNITS (BETWEEN
*                           ZERO AND 1023 DECIMAL)
*
*    NDELY     IS THE PLOTTING INCREMENT. IT DETERMINES THE SPACING OF
*                THE POINTS WHICH MAKE THE LINE. A LARGE NDELY WILL PLOT
*                A DOTTED LINE. NDELY IS AN INTEGER IN THE DECREMENT.
*              NO POINT IS PLOTTED IF NYBOT IS GREATER THAN NYTOP, AND
*              NO POINT IS PLOTTED IF NDELY=0 EXCEPT WHEN NYTOP=NYBOT.
*              IN THIS CASE THE POINT (NXBOT,NYBOT) IS PLOTTED. NO ERROR
*              INDICATORS ARE SET FOR THESE CASES AND NO CHECK IS MADE
*              ON THE MAGNITUDES OF THE INPUT VALUES. QUANTITIES GREATER
*              THAN 1023 ARE PLOTTED MODULO 1024.
*
* OUTPUTS
*              VERTICAL LINE ON SCOPE.
*
* EXAMPLES
*
* 1. INPUTS  - NYBOT=0, NXBOT=0, NYTOP=1023, NDELY=1
*    OUTPUTS - LINE ON SCOPE (LEFT-MOST IN PICTURE)
*
* 2. INPUTS  - NYBOT=0, NXBOT=100, NYTOP=900, NDELY=2
*    OUTPUTS - LINE ON SCOPE (SECOND FROM LEFT IN PICTURE)
*
* 3. INPUTS  - NYBOT=0, NXBOT=200, NYTOP=775, NDELY=3
*    OUTPUTS - LINE ON SCOPE (THIRD FROM LEFT IN PICTURE)
*
* 4. INPUTS  - NYBOT=0, NXBOT=300, NYTOP=650, NDELY=4
*    OUTPUTS - LINE ON SCOPE (FOURTH FROM LEFT IN PICTURE)
*
```

****
*   LINEV (709)   *
****
(PAGE 2)

```
* 5. INPUTS  - NYBOT=0, NXBOT=400, NYTOP=525, NDELY=5
*    OUTPUTS - LINE ON SCOPE (FIFTH FROM LEFT IN PICTURE)
*
* 6. INPUTS  - NYBOT=0, NXBOT=500, NYTOP= 400, NDELY=6
*    OUTPUTS - LINE ON SCOPE (SIXTH FROM LEFT IN PICTURE)
*
* 7. INPUTS  - NYBOT=0, NXBOT=600, NYTOP=275, NDELY=7
*    OUTPUTS - LINE ON SCOPE (SEVENTH FROM LEFT IN PICTURE)
*
* 8. INPUTS  - NYBOT=0, NXBOT=700, NYTOP=150, NDELY=8
*    OUTPUTS - LINE ON SCOPE (EIGHTH FROM LEFT IN PICTURE)
*
* 9. INPUTS  - NYBOT=0, NXBOT=800, NYTOP=0, NDELY=0
*    OUTPUTS - POINT ON SCOPE (800 SCOPE UNITS FROM LEFT IN PICTURE)
*
*10. INPUTS  - NYBOT=0, NXBOT=900, NYTOP=10, NDELY=0
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 900 SCOPE UNITS FROM LEFT)
*
*11. INPUTS  - NYBOT=100, NXBOT=1000, NYTOP=10, NDELY=0
*    OUTPUTS - NO POINTS ON SCOPE(BLANK FILM 1000 SCOPE UNITS FROM LEFT)
*
*     PICTURE OF SCOPE OUTPUT APPEARS BELOW.
*
*
```



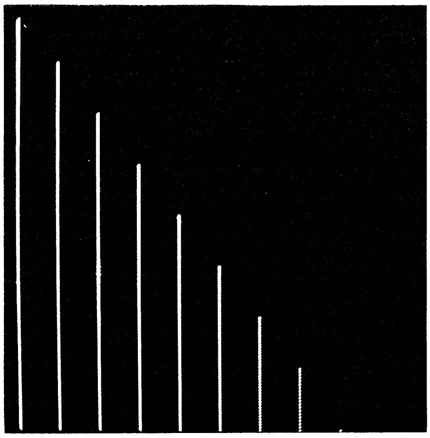
```
*
       PZE
       BCI     1,LINEV
 LINEV SXA     BACK,1
       SXA     BACK+1,2
       SXD     LINEV-2,4
       CLA*    1,4            X COORD.
       STD     POINT
       CLA*    3,4            YTOP
       SUB*    2,4            YBOT
       TZE     LAST
       TMI     BACK
       PDX     ,1
       CLA*    4,4            DELTA
       TZE     BACK
       STD     END
       STD     INCR
       CLA*    2,4
       PDX     ,2
       SXA     POINT,2
 LOOP  WTV
       CPY     POINT
 INCR  TXI     *+1,2,**
       SXA     POINT,2
 END   TIX     LOOP,1,**
```

****
*   LINEV (709)   *
****
(PAGE 3)

```
         TXL     BACK,1,0
LAST     CLA*    3,4
         ARS     18
         STA     POINT
         WTV
         CPY     POINT
BACK     AXT     **,1
         AXT     **,2
         TRA     5,4
POINT    PZE     0
         END
```

```
*      LINEV (7090) (SUBROUTINE)          9/4/64   LAST CARD IN DECK IS NO. 0168
*      FAP
*LINEV (7090)
       COUNT   140
       LBL     LINEV
       ENTRY   LINEV (NXBOT, NYBOT, NYTOP, NDELY)
*
*                   ----ABSTRACT----
*
* TITLE - LINEV (7090)
*       PLOT FAST VERTICAL LINE ON SCOPE
*
*             LINEV HAS ARGUMENTS NXBOT, NYBOT, NYTOP, NDELY. IT PLOTS A
*             VERTICAL LINE ON THE SCOPE WITH BOTTOM COORDINATES
*             (NXBOT, NYBOT), AND TOP COORDINATES (NXBOT, NYTOP). THE
*             SPACING OF THE POINTS WHICH COMPRISE THE LINE IS NDELY.
*             THE LINE IS PLOTTED FROM BOTTOM TO TOP BY PLOTTING THE
*             POINTS (NXBOT, NYBOT+K*NDELY) FOR K=0,1,2,...,M WHERE
*             M*NDELY IS LESS THAN OR = TO (NYTOP-NYBOT), AND
*             (M+1)*NDELY IS GREATER THAN (NYTOP-NYBOT).  IF =, LINE IS
*             FINISHED.  IF LESS, ONE MORE POINT WILL BE PLOTTED WITH
*             COORDINATES (NXBOT,NYTOP).  NOTE INPUT VALUE RESTRICTIONS
*             LISTED UNDER INPUTS.
*
* LANGUAGE   - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 7090 WITH SCOPE
* STORAGE    - 35 DECIMAL REGISTERS
* SPEED      - .5+.141*(LENGTH OF LINE/PLOTTING INCREMENT) MILLISECONDS
* AUTHOR     - J.N. GALBRAITH , MAY 10, 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL  LINEV(NXBOT, NYBOT, NYTOP, NDELY)
*
* INPUTS
*
*    NXBOT      IS THE X COORDINATE OF THE BOTTOM OF THE LINE
*
*    NYBOT      IS THE Y COORDINATE OF THE BOTTOM OF THE LINE
*
*    NYTOP      IS THE Y COORDINATE OF THE TOP OF THE LINE
*                 ABOVE COORDINATES ARE INTEGERS IN THE DECREMENT
*                 AND ARE ASSUMED TO BE IN SCOPE UNITS (BETWEEN
*                          ZERO AND 1023 DECIMAL)
*
*    NDELY      IS THE PLOTTING INCREMENT. IT DETERMINES THE SPACING OF
*                 THE POINTS WHICH MAKE THE LINE. A LARGE NDELY WILL PLOT
*                 A DOTTED LINE. NDELY IS AN INTEGER IN THE DECREMENT.
*               NO POINT IS PLOTTED IF NYBOT IS GREATER THAN NYTOP, AND
*               NO POINT IS PLOTTED IF NDELY=0 EXCEPT WHEN NYTOP=NYBOT.
*               IN THIS CASE THE POINT (NXBOT,NYBOT) IS PLOTTED. NO ERROR
*               INDICATORS ARE SET FOR THESE CASES AND NO CHECK IS MADE
*               ON THE MAGNITUDES OF THE INPUT VALUES. QUANTITIES GREATER
*               THAN 1023 ARE PLOTTED MODULO 1024.
*
* OUTPUTS
*
*               VERTICAL LINE ON SCOPE.
*
* EXAMPLES
*
* 1. INPUTS  - NYBOT=0, NXBOT=0, NYTOP=1023, NDELY=1
*    OUTPUTS - LINE ON SCOPE (LEFT-MOST IN PICTURE)
*
* 2. INPUTS  - NYBOT=0, NXBOT=100, NYTOP=900, NDELY=2
*    OUTPUTS - LINE ON SCOPE (SECOND FROM LEFT IN PICTURE)
*
* 3. INPUTS  - NYBOT=0, NXBOT=200, NYTOP=775, NDELY=3
*    OUTPUTS - LINE ON SCOPE (THIRD FROM LEFT IN PICTURE)
*
* 4. INPUTS  - NYBOT=0, NXBOT=300, NYTOP=650, NDELY=4
```

```
*    OUTPUTS - LINE ON SCOPE (FOURTH FROM LEFT IN PICTURE)
*
* 5. INPUTS  - NYBOT=0, NXBOT=400, NYTOP=525, NDELY=5
*    OUTPUTS - LINE ON SCOPE (FIFTH FROM LEFT IN PICTURE)
*
* 6. INPUTS  - NYBOT=0, NXBOT=500, NYTOP= 400, NDELY=6
*    OUTPUTS - LINE ON SCOPE (SIXTH FROM LEFT IN PICTURE)
*
* 7. INPUTS  - NYBOT=0, NXBOT=600, NYTOP=275, NDELY=7
*    OUTPUTS - LINE ON SCOPE (SEVENTH FROM LEFT IN PICTURE)
*
* 8. INPUTS  - NYBOT=0, NXBOT=700, NYTOP=150, NDELY=8
*    OUTPUTS - LINE ON SCOPE (EIGHTH FROM LEFT IN PICTURE)
*
* 9. INPUTS  - NYBOT=0, NXBOT=800, NYTOP=0, NDELY=0
*    OUTPUTS - POINT ON SCOPE (800 SCOPE UNITS FROM LEFT IN PICTURE)
*
*10. INPUTS  - NYBOT=0, NXBOT=900, NYTOP=10, NDELY=0
*    OUTPUTS - NO POINTS ON SCOPE (BLANK FILM 900 SCOPE UNITS FROM LEFT)
*
*11. INPUTS  - NYBOT=100, NXBOT=1000, NYTOP=10, NDELY=0
*    OUTPUTS - NO POINTS ON SCOPE(BLANK FILM 1000 SCOPE UNITS FROM LEFT)
*
*     PICTURE OF SCOPE OUTPUT APPEARS BELOW.
*
```



```
*
* PROGRAM FOLLOWS BELOW
*
* FOLLOWING CARD DESIGNATES THE DATA CHANNEL THAT CRT IS ATTACHED TO.
* TO CHANGE, ALTER THE LETTER DESIGNATION ONLY, AND REASSEMBLE.
X      TAPENO  D1
SCPAD EQU     X-105
      PZE
      BCI     1,LINEV
LINEV SXA     BACK,1
      SXA     BACK+1,2
      SXD     LINEV-2,4
      CLA*    1,4            X COORD.
      STD     POINT
      CLA*    3,4            YTOP
      SUB*    2,4            YBOT
      TZE     LAST
      TMI     BACK
      PDX     ,1
      CLA*    4,4            DELTA
      TZE     BACK
      STD     END
      STD     INCR
      CLA*    2,4
```

LINEV (7090)
(PAGE 3)

```
         PDX     ,2
         SXA     POINT,2
   LOOP  WRS     SCPAD
         RCHX    IO
   INCR  TXI     *+1,2,**
         SXA     POINT,2
   END   TIX     LOOP,1,**
         TXL     BACK,1,0
   LAST  CLA*    3,4
         ARS     18
         STA     POINT
         WRS     SCPAD
         RCHX    IO
   BACK  AXT     **,1
         AXT     **,2
         TRA     5,4
   POINT PZE     0
   IO    IOCD    POINT,0,1
         END
```

```
*     LINTR1 (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0092
*     LABEL
CLINTR1
      SUBROUTINE LINTR1(X,XLO,DELX,TABLE,NTABLE,YOFX)
C
C                    ----ABSTRACT----
C
C TITLE - LINTR1
C       LINEAR INTERPOLATION IN A TABLE
C
C             LINTR1 INTERPOLATES LINEARLY IN A TABLE TO FIND A VALUE
C             WHICH LIES BETWEEN THE TABULATED VALUES. XLO IS THE
C             ARGUMENT CORRESPONDING TO THE LOWEST TABULATED VALUE. DELX
C             IS THE ARGUMENT DIFFERENCE BETWEEN TABULAR VALUES.
C             THE TABLE IS LOCATED IN TABLE(I). X IS THE ARGUMENT AND
C             YOFX IS THE INTERPOLATED VALUE. HENCE
C
C                                                       XTRA
C             YOFX = TABLE(L) + (TABLE(L+1) - TABLE(L)) * ----
C                                                       DELX
C
C                  WHERE L IS SUCH THAT
C                       XLO+(L-1)*DELX LSTHN= X LSTHN XLO+L*DELX
C                  AND  XTRA = X-XLO-(L-1)*DELX
C
C             DELX IS CONSTRAINED TO BE POSITIVE
C             X MUST LIE IN THE ARGUMENT RANGE OF THE TABLE.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 96 REGISTERS
C SPEED     -
C AUTHOR    - S. M. SIMPSON
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL LINTR1(X,XLO,DELX,TABLE,NTABLE,YOFX)
C
C INPUTS
C
C    X          IS ARGUMENT FOR WHICH INTERPOLATION IS DESIRED.
C               XLO LSTHN OR = X LSTHN OR = XLO+(NTABLE-1)*DELX.
C
C    XLO        IS THE ARGUMENT CORRESPONDING TO THE FIRST TABULAR
C               ENTRY.
C
C    DELX       IS THE ARGUMENT DIFFERENCE BETWEEN TWO SUCCESSIVE
C               TABULAR ENTRIES.
C               MUST EXCEED 0.0, BUT THIS CONSTRAINT IS NOT CHECKED.
C
C    TABLE(I)   I=1...NTABLE IS A GIVEN ARRAY IN WHICH TABLE(J)
C                 CONTAINS Y(XLO+DELX*(J-1)).
C
C    NTABLE     IS THE LENGTH OF THE TABLE.
C
C OUTPUTS
C
C    YOFX       WILL CONTAIN THE LINEARLY INTERPOLATED VALUE
C
C EXAMPLES
C
C 1. INPUTS  - X=7.5     XLO=5.     DELX=2.5     TABLE(1...9)=1.,4.,9.,
C              16.,25.,36.,49.,64.,81.     NTABLE=9
C    OUTPUTS - YOFX=4.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT X=21.3
C    OUTPUTS - YOFX=56.8
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT X=25.
C    OUTPUTS - YOFX=81.
C
```

LINTR1
(PAGE 2)

```
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT X=13.
C    OUTPUTS - YOFX=17.8
C
      DIMENSION  TABLE(2)
C SET UP.
      XMXLO=X-XLO
   20 ILO=XMXLO/DELX+1.0
C INTERPOLATE ONLY IF ILO DOESNT CORRESPOND TO LAST TABULAR ENTRY.
      IF (ILO-NTABLE)  30,40,30
   30 FLILO=ILO-1
      DIFX=XMXLO-FLILO*DELX
      IHI=ILO+1
      YOFX=TABLE(ILO)+(TABLE(IHI)-TABLE(ILO))*DIFX/DELX
      GO TO 9999
   40 YOFX=TABLE(NTABLE)
      GO TO 9999
 9999 RETURN
      END
```

```
*     LISTNG (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0220
*     LABEL
CLISTNG
      SUBROUTINE LISTNG (ITAPE,JTAPE,DATA)
C
C                    ----ABSTRACT----
C
C TITLE - LISTNG
C        LIST AUXILIARY INFORMATION FOR A INDATA-OUDATA TYPE TAPE
C
C              LISTNG REWINDS A SPECIFIED TAPE, WRITES THE RECORD NUMBER,
C              LENGTH OF DATA, AND AUXILIARY INFORMATION, AND CHECKS THE
C              SUMCHECK FOR EACH RECORD ON THE TAPE, AND THEN REWINDS
C              THE TAPE AGAIN.  SEE THE WRITE-UP FOR OUDATA FOR A
C              DESCRIPTION OF THE FORMAT OF THE TAPE.
C
C              THE RECORD NUMBER AND AUXILIARY INFORMATION ARE
C              INTERPRETED AS FLOATING POINT, FIXED POINT, OCTAL, AND
C              HOLLERITH. THE HOLLERITH PRINT-OUT IS SUPPRESSED IF
C              ILLEGAL CHARACTERS ARE PRESENT.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME, DATA CHANNEL)
C STORAGE   - 755 REGISTERS
C SPEED     -
C AUTHOR    - R.A. WIGGINS   NOV., 1962
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FAPSUM, SAME, XSAME, FSKIP, SHFTR2
C        AND FORTRAN SYSTEM ROUTINES - (FIL),(RLR),(RWT),(SPH),(STH),
C                                      (TSB)
C
C FORTRAN USAGE
C     CALL LISTNG(ITAPE,JTAPE,DATA)
C
C INPUTS
C
C    ITAPE     IS LOGICAL TAPE NUMBER FOR THE TAPE THAT IS TO BE LISTED.
C              IS FORTRAN II INTEGER.
C
C    JTAPE     IS LOGICAL TAPE NUMBER FOR OUTPUT TAPE
C                (LISTNG DOES NOT REWIND THIS TAPE BEFORE OR AFTER
C                OUTPUT)
C
C    DATA(I)   I=1,N IS A BUFFER FOR TEMPORARY USE BY LISTNG.
C              N MUST BE GREATER THAN THE LONGEST DATA SERIES ON THE
C                TAPE.
C
C OUTPUTS  THE OUTPUT IS A LISTING OF THE TAPE AS SHOWN IN THE EXAMPLE
C          NOTE THAT ONLY THE FIRST 50 WORDS OF EACH AUXILIARY
C          INFORMATION IS PRINTED.
C
C EXAMPLES
C
C              EXAMPLES FOR OUDATA LOADED ON LOGICAL UNIT 9.
C    USAGE   -        DIMENSION DATA(10000)
C                     CALL LISTNG (9,2,DATA)
C    OUTPUTS - WRITTEN ON LOGICAL TAPE NO. 2
C
C111111111122222222223333333333444444444455555555556666666666777777777788
C12345678901234567890123456789012345678901234567890123456789012345678901
C
C  (PAGE 1 CONTAINS)
C
C THIS IS A LISTING OF THE AUXILIARY INFORMATION AND STATISTICS FOR AN
C   ((INDATA-OUDATA)) TYPE TAPE
C
C  (PAGE 2 CONTAINS)
C
C FILE   1 CONTAINS
C
C     RECORD NO.        -74852 (INTERPRETED AS AN INTEGER)
C                 -0.051516E 06 (INTERPRETED AS FLOATING POINT)
```

```
      C                 622144474325 (INTERPRETED AS OCTAL)
      C                       SAMPLE (INTERPRETED AS ALPHANUMERIC)
      C
      C LENGTH OF AUXILIARY INFORMATION BLOCK IS     2
      C NUMBER OF DATA POINTS IS     3
      C NUMBER OF DATA POINTS STORED PER REGISTER IS     1
      C
      C                       FLOATING       FIXED       OCTAL    ALPHANUMERIC
      C
      C
      C (PAGE 3 CONTAINS)
      C
      C FILE   2 CONTAINS
      C
      C     RECORD NO.                  3 (INTERPRETED AS AN INTEGER)
      C                 0.001722E-38 (INTERPRETED AS FLOATING POINT)
      C                 000003000000 (INTERPRETED AS OCTAL)
      C                       003000 (INTERPRETED AS ALPHANUMERIC)
      C
      C     WITH TITLE      SAMPLE INDATA-OUDATA TYPE TAPE RECORD
      C
      C LENGTH OF AUXILIARY INFORMATION BLOCK IS    21
      C NUMBER OF DATA POINTS IS     3
      C NUMBER OF DATA POINTS STORED PER REGISTER IS     2
      C
      C                       FLOATING       FIXED       OCTAL    ALPHANUMERIC
      C
      C    DELTAT          0.49999999E-01    63897   174631463146
      C
      C    RDAY            0.01721915E-38       30   000036000000
      C
      C    RUNITS         -0.16497062E-28   -18003   443123514645     MICRON
      C
      C    TITLE          -0.48000000E 02   -68992   606600000000      SAMPL
      C                    0.03516110E 14    89113   256031452421     E INDA
      C                   -0.12662410E 08   -78944   632140466424     TA-OUD
      C                    0.67124052E 04    72913   216321606370     ATA TY
      C                   -0.61013036E-21   -30064   472560632147     PE TAP
      C                    0.05680751E 14    89192   256051252346     E RECO
      C                   -0.33042351E-16   -38192   512460606060     RD
      C                   -0.06095237E 02   -68656   606060606060
      C
            DIMENSION DATA(10000)
      C     DEFINE THE MAXIMUM NUMBER OF ELEMENTS TO BE PRINTED IN AUX. INFO.
            MA = 50
      C     DEFINE THE PRINTED OUTPUT TAPE NO.
            N = JTAPE
            REWIND ITAPE
            ERR=0.
            WRITE OUTPUT TAPE N,10
       10   FORMAT(97H1THIS IS A LISTING OF THE AUXILIARY INFORMATION AND STAT
           1ISTICS FOR AN ((INDATA-OUDATA)) TYPE TAPE)
            IFILE=1
       15   READ TAPE ITAPE,IRECNO,LAUXBK,NOPTS,MODCOD,SCALE
            IF(IRECNO)  18,16,18
       16   CONTINUE
            REWIND ITAPE
            RETURN
       18   READ TAPE ITAPE,(DATA(I),I=1,LAUXBK)
            CALL FAPSUM (LAUXBK-1,DATA,SUMCK)
            IF(SUMCK-DATA(LAUXBK))  20,30,20
       20   ERR=1.
       30   CONTINUE
            J1 = 1
            TES = SAMEF(IRECNO)
            GO TO 5000
      200   WRITE OUTPUT TAPE N,210,IFILE,IRECNO,IRECNO,IRECNO,IRECNO
      210   FORMAT(5H1FILE,I4,9H CONTAINS//5X11HRECORD NO. I14,28H (INTERPRETE
           1D AS AN INTEGER)/16XE14.6,32H (INTERPRETED AS FLOATING POINT)/16XO
           214,23H (INTERPRETED AS OCTAL)/24XA6,30H (INTERPRETED AS ALPHANUMER
           3IC))
            GO TO 225
      215   WRITE OUTPUT TAPE N, 210,IFILE,IRECNO,IRECNO,IRECNO
      C     SCAN FOR TITLE
      225   T = 5HTITLE
```

```
       L=1
  35   IF(DATA(L))  40,80,40
  40   IF(DATA(L)-T)  50,60,50
  50   L=L+2+XSAMEF(DATA(L+1))
       GO TO 35
  60   IMIN=L+2
       IMAX=L+1+XSAMEF(DATA(L+1))
       WRITE OUTPUT TAPE N,70,(DATA(I),I=IMIN,IMAX)
  70   FORMAT(15H0    WITH TITLE5X12A6)
C      EITHER NO TITLE FOUND, OR TITLE FOUND AND PRINTED
  80   CONTINUE
C      PRINT INFORMATION ABOUT DATA
 220   WRITE OUTPUT TAPE N,230,LAUXBK,NOPTS,MODCOD
 230   FORMAT(41H0LENGTH OF AUXILIARY INFORMATION BLOCK IS,
      1I6/25H NUMBER OF DATA POINTS IS,I6/45H NUMBER OF DATA POINTS STORE
      2D PER REGISTER IS,I6//23X8HFLOATING7X5HFIXED7X5HOCTAL4X12HALPHANUM
      3ERIC)
C      PRINT AUX INFO.    (ONLY FIRST MA ELEMENTS IF A VECTOR)
       L=1
  90   IF(DATA(L))  100,150,100
 100   J1 = 2
       TES = DATA(L+2)
       GO TO 5000
 310   WRITE OUTPUT TAPE N, 101, DATA(L),DATA(L+2),DATA(L+2),DATA(L+2),
      1      DATA(L+2)
 101   FORMAT(1H03XA6,5XE18.8,I9,O15,4XA6)
       GO TO 330
 320   WRITE OUTPUT TAPE N, 101, DATA(L),DATA(L+2),DATA(L+2),DATA(L+2)
 330   IF (XSAMEF(DATA(L+1))-1) 140,140,340
 340   IMIN = L+3
       IMAX=XMINOF(MA,XSAMEF(DATA(L+1))) + L+1
       J1 = 3
       DO 108  J=IMIN,IMAX
       TES = DATA(J)
       GO TO 5000
 360   WRITE OUTPUT TAPE N,110,DATA(J),DATA(J),DATA(J),DATA(J)
 110   FORMAT(15XE18.8,I9,O15,4XA6)
       GO TO 108
 370   WRITE OUTPUT TAPE N,110,DATA(J),DATA(J),DATA(J)
 108   CONTINUE
       IF(XSAMEF(DATA(L+1))-MA)  140,140,120
 120   WRITE OUTPUT TAPE N,130,(DATA(L+1))
 130   FORMAT(1H025X4HETC,I6,8H IN ALL.)
 140   L=L+2+XSAMEF(DATA(L+1))
       GO TO 90
 150   CONTINUE
C      AUXILIARY INFO IS LISTED
       IF(ERR)  160,155,160
 155   NN=(NOPTS+MODCOD-1)/MODCOD +1
       READ TAPE ITAPE,(DATA(I),I=1,NN)
       CALL FAPSUM(NN-1,DATA,SUMCK)
       IF(SUMCK-DATA(NN))  160,180,160
 160   CONTINUE
       WRITE OUTPUT TAPE N,170,IFILE
       PRINT 170,IFILE
 170   FORMAT(    21H BAD SUMCHECK ON FILE,I6)
 180   IFILE=IFILE+1
       CALL FSKIP(ITAPE,+1)
       GO TO 15
C      THIS IS AN INTERNAL SUBROUTINE TO CHECK IF A DATA WORD HAS ILLEGAL
C      CHARACTERS FOR ALPHANUMERIC PRINTING.
5000   DO 1 I=1,6
       CALL SHFTR2(18-I*6,TES,1,TEZ,IANS)
B      TEZ=TEZ*000077000000
       K=XSAMEF(TEZ)+1
       GO TO(1,1,1,1,1,1,1,1,1,1,2,1,2,2,2,2,1,1,1,1,1,1,1,1,1,1,2,1,1,2,
      1      2,2,1,1,1,1,1,1,1,1,1,1,2,2,1,2,2,2,1,1,1,1,1,1,1,1,1,1,1,2,
      2      1,1,2,2,2),K
  1    CONTINUE
       GO TO (200,310,360),J1
  2    GO TO (215,320,370),J1
       END
```

```
*      LOC (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0053
*      FAP
*LOC
       COUNT   30
       LBL     LOC
       ENTRY   LOC  (VAR,IADD)
*
*                   ----ABSTRACT----
*
* TITLE - LOC
*       CORE LOCATION WITH INDEXABLE ARGUMENT
*
*             LOC GIVES THE CORE ADDRESS OF A VARIABLE.  THE VARIABLE
*             MAY BE SUBSCRIPTED.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 4 REGISTERS
* SPEED     - ABOUT 12 MACHINE CYCLES
* AUTHOR    - R.A. WIGGINS,   MAY, 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL LOC   (VAR,IADD)
*
* INPUTS
*
*    VAR       IS VARIABLE NAME (NEED NOT BE FLOATING POINT NAME).
*
* OUTPUTS
*
*    IADD      IS THE CORE ADDRESS FOR THE VARIABLE NAME.
*              IS FORTRAN II INTEGER
*
* EXAMPLES
*
* 1. INPUTS  - SUPPOSE VAR(1...5) IS STORED BEGINNING AT 77461 OCTAL
*    USAGE   -       CALL LOC (VAR,IADD)
*    OUTPUTS - IADD = 32561 (=OCTAL 77461)
*
* 2. INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   -       I=3
*                    CALL LOC (VAR(I),IADD)
*    OUTPUTS - IADD = 32559 (=OCTAL 77457)
*
 LOC   CAL     1,4
       ALS     18
       STD*    2,4
       TRA     3,4
       END
```

```
*     LOCATE (SUBROUTINE)             3/15/65   LAST CARD IN DECK IS NO. 2007
*     FAP
*LOCATE
*                     DEDICATED TO JACKIE
*
      COUNT   2000
      LBL     LOCATE
      ENTRY   LOCATE (SUBRU1,SUBRU2,...,SUBRUN)
      ENTRY   WHERE  (SUBRU,IANS,LOC,NARGS)
      ENTRY   CALL   (SUBRU,IANS,SPACER,ARG1,ARG2,...,ARGN)
      ENTRY   CALL2  (SUBRUV,IANS)
      ENTRY   SETSBV (SUBRU,SUBRUV,ARG1,ARG2,...,ARGN)
      ENTRY   SETUP  (LOCALL,NARGS,XR1,XR2)
      ENTRY   RETURN (LOCALL,XR1,XR2)
      ENTRY   XINDEX (LOCALL,NUMARG)            (FUNCTION)
      ENTRY   ARG    (LOCALL,NUMARG,IXVECT)     (FUNCTION)
      ENTRY   XARG   (LOCALL,NUMARG,IXVECT)     (FUNCTION)
      ENTRY   STORE  (ARGU,LOCALL,NUMARG,IXVECT)
      ENTRY   XNARGS (LOCALL)                   (FUNCTION)
      ENTRY   XNAME  (HNAME1,HNAME2)            (FUNCTION)
*
*                    ----ABSTRACT----
*
* TITLE - LOCATE , WITH SECONDARY ENTRIES WHERE,CALL,CALL2,SETSBV,SETUP,
*                                         RETURN,XINDEX(FUNCTION),
*                                         ARG(FUNCTION),XARG(FUNCTION),
*                                         STORE,XNARGS(FUNCTION),
*                                         AND XNAME(FUNCTION).
*
*       LOCATE AND OPERATE SUBROUTINES BY PROXY CALL STATEMENTS
*
*             LOCATE AND ITS ASSOCIATED ENTRIES ENABLE A FORTRAN II
*             PROGRAM (AT LEVEL 1) TO INDUCE A SUBROUTINE (AT LEVEL 2)
*             TO OPERATE, VIA PROXY CALL STATEMENTS (ENTRIES CALL AND
*             CALL2), ONE OR MORE STILL LOWER LEVEL SUBROUTINES,
*             WHERE THE SUBROUTINE AT LEVEL 2 NEED NOT KNOW IN
*             ADVANCE ANYTHING ABOUT THE LOWER LEVEL SUBROUTINES
*             (I.E., HOW MANY SUBROUTINES THERE ARE, WHAT THEIR
*             NAMES ARE, WHAT THEIR FUNCTIONS ARE, OR WHAT THE
*             NUMBER OF ARGUMENTS ASSOCIATED WITH EACH IS).
*
*             THE MOST SIGNIFICANT APPLICATION OF SUCH A FEATURE IS IN
*             THE CONSTRUCTION OF A CONTROL SUBROUTINE WHOSE FUNCTION
*             IS TO OPERATE, WITHIN THE FORMAL FRAMEWORK OF SOME GOAL,
*             A REPERTOIRE OF LOWER LEVEL SUBROUTINES WHICH IS VARIABLE
*             IN NUMBER AND NAMES AND PERHAPS EVOLVING WITH TIME.  THE
*             CONTROL SUBROUTINE CAN BE ISOLATED FROM SUCH CHANGES AND
*             REMAIN WITHIN THE FORTRAN-II SYSTEM, AND THE MACHINE
*             MEMORY REQUIREMENTS DURING ANY ONE EXECUTION ARE CONFINED
*             TO THOSE OF THE SPECIFIC SUBSET OF SUBROUTINES DESIRED
*             DURING THAT EXECUTION.
*
*             CHAINS OF SUCCESSIVE PROXY CALL STATEMENTS WILL WORK
*             PROPERLY, AND ONE OF THE ENTRIES (CALL) PERMITS PROXY
*             CALL STATEMENTS OF UNORTHODOX SUBROUTINES (SUCH AS DISPLA
*             AND GENHOL OR LOCATE ITSELF) WHICH UTILIZE INFORMATION
*             FROM THE STATEMENT(S) IMMEDIATELY FOLLOWING THEIR CALL
*             STATEMENT.
*
*             AS BY-PRODUCTS OF THE ABOVE FUNCTIONS, THE LOCATE GROUP
*             ALSO ENABLES
*               1. FORTRAN PROGRAMS TO FIND, AT EXECUTION TIME, THE
*                  ABSOLUTE MACHINE LOCATIONS OF THE ENTRY POINTS OF
*                  ANY SUBROUTINES WHOSE NAMES ARE KNOWN IN ADVANCE
*                  (ENTRIES LOCATE AND WHERE).
*               2. THE OPERATION OF A SUBROUTINE UNDER ONE OR MORE
*                  PSEUDONYMS, AND THE OPERATION OF DIFFERENT
*                  SUBROUTINES UNDER THE SAME NAME (ENTRY LOCATE).
*               3. FORTRAN SUBROUTINES TO BE WRITTEN WITH VARIABLE-
*                  LENGTH CALLING SEQUENCES (ENTRIES SETUP AND RETURN).
*               4. SUCH A VARIABLE-LENGTH-CALLING-SEQUENCE PROGRAM TO
*                  OBTAIN EASILY ANY OF ITS ARGUMENTS, EVEN IF ITS
*                  SUBROUTINE CARD LISTS NO ARGUMENTS AT ALL (ENTRIES
*                  XINDEX, ARG, AND XARG (ALL FORTRAN FUNCTIONS)).
*               5. THE OPERATION OF A SUBROUTINE WHOSE ARGUMENT COUNT
```

```
*                          IS LIMITED ONLY BY THE MEMORY SIZE (ENTRY CALL2).
*                       6. THE TRANSMISSION THRU SUBROUTINE LAYERS OF
*                          ARBITRARILY COMPLEX CALLING SEQUENCES VIA SINGLE
*                          NUMBERS (ENTRIES SETUP, XINDEX, ARG AND XARG,
*                          XNARGS AND STORE).
*                       7. DIRECT RETURN LINKAGES FROM LOW LEVEL SUBROUTINES
*                          TO HIGH LEVEL ROUTINES WHICH BYPASS INTERMEDIATE
*                          LEVELS AND STILL ALLOW RETURN MESSAGES (ENTRIES
*                          RETURN AND STORE).
*                       8. SUBROUTINE TRANSFER PATHS WHICH ARE CIRCULAR, E.G.,
*                          A SUBROUTINE CALLING ITSELF, OR CALLING A SUBROUTINE
*                          WHICH CALLS IT BACK (ENTRIES SETUP AND RETURN).
*
*                    THE LOCATE GROUP IS DEPENDENT ON THE SUBROUTINE LINKAGE
*                    CONCEPT (TRANSFER VECTOR AND ARGUMENT VECTOR (TSX ARG,0))
*                    OF THE FORTRAN SYSTEM AND MONITOR SYSTEM.  IT WILL NOT
*                    WORK ON SYSTEMS WITH DIFFERENT LINKAGE PATTERNS.  WITHIN
*                    THE FORTRAN II SYSTEM, HOWEVER, THESE PROGRAMS WORK
*                    INDEPENDENTLY OF THE INCLUSION OR EXCLUSION OF THE
*                    STANDARD ERROR PROCEDURE WHICH IS AN OPTIONAL FUNCTION
*                    AFFECTING LINKAGE FORMAT.
*
* LANGUAGE    - FAP SUBROUTINES (COMPATIBLE WITH FORTRAN-II AS
*                                 SUBROUTINES OR FUNCTIONS DEPENDING
*                                 ON THE ENTRY)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 512 REGISTERS
*               LOCATE IS A MINIMUM STORAGE PROGRAM
*
* SPEED       - VARIABLE DEPENDING ON ENTRY.  SINCE THE SYSTEM IS
*               DESIGNED FOR PRACTICAL APPLICATIONS INVOLVING NON-TRIVIAL
*               SUBROUTINES THE EXTRA TIME TAKEN BY THE LOCATE ROUTINES
*               TO ESTABLISH LINKAGE SHOULD NOT BE SIGNIFICANT.
*               APPROXIMATE TIMES (IN MACHINE CYCLES) ARE
*               LOCATE         - 160 (MINIMUM)
*               WHERE          - 400 (MINIMUM)
*               CALL           - 570 (MINIMUM)
*               CALL2          - 610 (MINIMUM)
*               SETSBV         - 330 (MINIMUM)
*               SETUP          - 430 (MINIMUM)
*               RETURN         - 250
*               XINDEX         - 44
*               ARG (OR XARG)  - 41
*               STORE          - 157
*               XNARGS         - 82 (MINIMUM)
*               XNAME          - 58 (MINIMUM)
*
* AUTHOR      - S.M.SIMPSON, MARCH 1963
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* PRELIMINARY NOTES -
*
*               1. ILLEGAL ARGUMENT COUNTS
*               IF ANY OF THE 8 SUBROUTINE ENTRIES IN THE LOCATE GROUP
*               ARE CALLED WITH AN INCORRECT ARGUMENT COUNT EXECUTION
*               STOPS ON AN HPR 77777(OCTAL) INSTRUCTION, WITH
*                 AC = 6H(NAME OF ENTRY CALLED ILLEGALLY)
*                 MQ = NO. OF ARGUMENTS (IN ADDRESS OF MQ)
*                 INDEX REGISTER 4 = ADDRESS OF ILLEGAL CALL STATEMENT
*                                   (NOT TWO*S COMPLEMENT OF ADDRESS)
*               IF COMPUTER IS RESTARTED CONTROL RETURNS IMMEDIATELY TO
*               THE STATEMENT FOLLOWING THE ILLEGAL CALL STATEMENT.
*
*               FOR THE VARIABLE-LENGTH-CALLING-SEQUENCE ENTRIES THE
*               ILLEGAL ARGUMENT COUNTS ARE
*                 LOCATE - ZERO
*                 CALL   - LESS THAN THREE
*                 SETSBV - LESS THAN TWO
*
```

```
*             NO ARGUMENT COUNT CHECK IS MADE FOR THE FUNCTION ENTRIES.
*
*             2. RESTORATION OF INDEX REGISTERS (PRIMARILY OF CONCERN
*                TO USERS MIXING FAP AND FORTRAN PROGRAMS)
*             ALL ENTRIES OF LOCATE RESTORE INDEX REGISTERS 1 AND 2
*             WITH PROVISOS IN THE CASES OF CALL, CALL2, AND RETURN.
*             INDEX REGISTER 4, USED FOR LINKAGE, IS SOMETIMES NOT
*             RESTORED.  CALL AND CALL2 DEFEND ON THE SUBROUTINE BEING
*             OPERATED TO RESTORE INDEX REGISTERS 1 AND 2 (THE CORRECT
*             VALUES ARE SET UP FOR THE SUBROUTINE BEFORE IT IS
*             ENTERED).  THE RESTORATION OF XR1 AND XR2 BY RETURN
*             DEPENDS ON THE EXISTENCE OF CONTIGUOUS SXD INSTRUCTIONS
*             AS THE VERY FIRST INSTRUCTIONS IN THE SUBROUTINE FROM
*             WHICH THE RETURN (OR THE APPARENT RETURN IN THE CASE OF
*             SKIP RETURNS) IS DESIRED.  FAP PROGRAMS USING ENTRY
*             RETURN SHOULD CONFORM TO THIS CONVENTION.
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY LOCATE
*
* FORTRAN USAGE OF LOCATE
*
*             SUBROUTINE LOCATE ESTABLISHES AN EQUIVALENCE BETWEEN
*             NAMES AND ABSOLUTE MACHINE LOCATIONS OF SUBROUTINES,
*             WHICH EQUIVALENCE IS UTILIZED IN SUBSEQUENT PROXY CALL
*             STATEMENTS.  AN APPROPRIATE CALL LOCATE STATEMENT MUST BE
*             EXECUTED PRIOR TO THE USE OF THE ENTRIES WHERE, CALL,
*             OR CALL2.  HOWEVER A CALL LOCATE STATEMENT NEED NOT BE
*             PRESENT IN A ROUTINE WHICH MAKES PROXY CALL STATEMENTS,
*             PROVIDED SOME OTHER ROUTINE HAS CALLED LOCATE PREVIOUSLY.
*             LOCATE IS CALLED WITH AN ARBITRARY NUMBER OF ARGUMENTS,
*             N, AS FOLLOWS.
*
*    CALL LOCATE(SUBRU1,SUBRU2,...,SUBRUN)
*  WHICH MUST BE FOLLOWED IMMEDIATELY BY N CALL STATEMENTS AS FOLLOWS
*    CALL SUBR1 (ARG11,ARG12,...,ARG1M1)
*    CALL SUBR2 (ARG21,ARG22,...,ARG2M2)
*         .
*         .
*         .
*    CALL SUBRN (ARGN1,ARGN2,...,ARGNMN)
*
*             THE N CALL STATEMENTS FOLLOWING THE CALL LOCATE STATEMENT
*             ARE NOT EXECUTED.  LOCATE RETURNS CONTROL TO THE
*             STATEMENT FOLLOWING CALL SUBRN.  (IT DOESNT MATTER IF THE
*             (N+1)TH STATEMENT FOLLOWING CALL LOCATE IS ALSO A CALL
*             STATEMENT.)  IN THE ABOVE ILLUSTRATION THE CALL SUBR
*             STATEMENTS ARE WRITTEN WITH INDIVIDUAL ARGUMENT LISTS.
*             THE OPERATION OF LOCATE, CALL AND CALL2 IS UNAFFECTED BY
*             THE LENGTHS OR CONTENTS OF THESE ARGUMENT LISTS.  THE
*             CALL SUBR STATEMENTS CAN BE WRITTEN EQUALLY AS WELL WITH
*             NO ARGUMENTS.
*
* INPUTS TO LOCATE
*
*   SUBRU1    IS 6 OR LESS HOLLERITH TO BE USED IN SUBSEQUENT WHERE,
*               CALL OR CALL2 STATEMENTS AS THE NAME OF THE FIRST
*               SUBROUTINE IN THE LIST OF N CALL SUBR STATEMENTS WHICH
*               IMMEDIATELY FOLLOW THE CALL LOCATE STATEMENT.  THE NAME
*               SUBRU1 DOES NOT HAVE TO BE IDENTICAL TO THE REAL
*               SUBROUTINE NAME, SUBR1, AS IT APPEARS IN THE CALL SUBR1
*               STATEMENT.  IF IT IS DIFFERENT FROM THE REAL NAME,
*               SUBRU1 DOES NOT HAVE TO CONFORM TO FORTRAN NAMING
*               CONVENTIONS (E.G., IT COULD BEGIN WITH A NUMBER,
*               INCLUDE SPECIAL CHARACTERS, OR EVEN BE 6 BLANKS).
*             SUBRU1 MUST BE IN FORMAT(1A6).  IF SUBR1 INVOLVES LESS
*               THAN 6 CHARACTERS THE POSITIONING OF THESE CHARACTERS
*               IS IMMATERIAL (SINCE WHERE, CALL AND CALL2 LEFT ADJUST
*               NAMES BEFORE COMPARING) BUT THE MISSING CHARACTERS MUST
*               BE BLANKS (OCT 60) AND EXTERNAL TO THE NAME.  IF SUBRU1
*               INVOLVES A BLANK CHARACTER BETWEEN TWO NON-BLANK
*               CHARACTERS THE BLANK CHARACTER IS CONSIDERED PART OF
*               THE NAME.
*
*   SUBRU2    IS 6 OR LESS HOLLERITH TO BE USED AS THE NAME OF THE
```

```
*                 SECOND SUBROUTINE IN THE LIST OF CALL SUBR STATEMENTS
*                 FOLLOWING THE CALL LOCATE STATEMENT.  IT HAS THE SAME
*                 CHARACTERISTICS AS SUBRU1.
*               SUBRU2 SHOULD NOT BE IDENTICAL TO SUBRU1 (I.E., IT IS
*                 IMPOSSIBLE IN THIS SYSTEM TO GIVE THE SAME NAME
*                 SIMULTANEOUSLY TO TWO DIFFERENT SUBROUTINES ALTHOUGH
*                 IT CAN BE DONE SEQUENTIALLY BY REPEATED CALL LOCATE
*                 STATEMENTS AS DESCRIBED BELOW).  IF THEY ARE IDENTICAL
*                 THEN SUBSEQUENT REFERENCE IN PROXY CALL STATEMENTS
*                 TO SUBRU2 = SUBRU1 WILL BE TRANSLATED AS A REFERENCE
*                 TO SUBR1, AND SUBR2 WILL NEVER BE OPERATED.  IN GENERAL
*                 IF SUBRUK IS IDENTICAL TO SUBRUL (K LESS THAN L) AND
*                 DIFFERENT FROM SUBRUJ (FOR ALL J LESS THAN K) THEN THE
*                 NAME SUBRUL=SUBRUK IMPLIES THE SUBROUTINE SUBRK.
*               HOWEVER THE CONVERSE WILL WORK, I.E., SUBR1 MAY BE THE
*                 SAME AS SUBR2, SO THAT SUBR1 CAN BE OPERATED EITHER
*                 UNDER THE NAME SUBRU1 OR UNDER THE NAME SUBRU2.
*
*       .             .
*       .             .
*       .             .
*
*    SUBRUN     IS THE NAME TO BE USED FOR THE LAST SUBROUTINE (SUBRN) OF
*                 THE CALL LIST.  THE NO. OF CALL STATEMENTS, NC,
*                 FOLLOWING THE CALL LOCATE STATEMENT SHOULD MATCH
*                 EXACTLY THE NO. OF ARGUMENTS, NA, IN THE CALL LOCATE
*                 STATEMENT.  IF NC IS LESS THAN NA CONTROL WILL BE
*                 RETURNED TO THE STATEMENT FOLLOWING THE CALL SUBRNC
*                 STATEMENT AND ARGUMENT NOS. NC+1,NC+2,...,NA OF CALL
*                 LOCATE WILL NOT BE RECOGNIZABLE IN PROXY CALL
*                 STATEMENTS.
*                 IF NC IS GREATER THAN NA, LOCATE RETURNS CONTROL TO THE
*                 STATEMENT FOLLOWING CALL SUBRNA, AND THE CALL
*                 SUBR(NA+1) STATEMENT WILL BE EXECUTED.
*
*               AS PRESENTLY WRITTEN LOCATE KEEPS RECORDS ONLY OF THE
*               LAST 14 CALL STATEMENTS MADE (AS EXECUTED FROM DIFFERENT
*               POSITIONS IN THE MEMORY, I.E., REPEATED EXECUTION OF THE
*               SAME CALL LOCATE STATEMENT, SAY IN A LOOP, COUNTS ONLY
*               ONCE).  LOCATE KEEPS ITS TABLES IN A REVOLVING FASHION
*               (CALL NO. 15 REPLACES CALL NO. 1) SO THAT ALL CALL LOCATE
*               STATEMENTS CAN BE ASSIGNED TABLE INDICES FROM 1 TO 14
*               (WHICH ARE NOT NECESSARILY IN THE SAME RELATIVE ORDER AS
*               THEIR TIME-OF-EXECUTION INDICES).  IF TWO DIFFERENT CALL
*               LOCATE STATEMENTS WITHIN THE LAST 14 EXECUTED ATTEMPT TO
*               DEFINE THE SAME PROXY NAME THEN THE DEFINITION ASSOCIATED
*               WITH THE LOWEST TABLE INDEX IS USED.
*
*               (THE LIMIT OF 14 ON LOCATE*S MEMORY CAN BE MOVED UP OR
*               DOWN BY MAKING THE FOLLOWING CHANGES TO THE PROGRAM (SEE
*               LISTING NEAR END)
*                 1. CHANGE THE CONSTANT KTABLE FROM 14 TO THE NEW VALUE
*                    DESIRED
*                 2. CHANGE THE BSS 12 INSTRUCTION (FIRST INSTRUCTION IN
*                    LIST OF VARIABLES) TO BSS (NEW VALUE - 2). )
*
*               A SCHEME SUCH AS THE FOLLOWING WILL ENABLE ONE TO
*               OPERATE A SINGLE SUBROUTINE UNDER DIFFERENT NAMES.
*
*                     ITIME=0
*                     SUBRU1=5HSUBRU
*                     SUBRU2=5HDUMMY
*                     GO TO 50
*                1    (OPERATE SUBR1 UNDER THE NAME SUBRU)
*                             .
*                             .
*                             .
*                     SUBRU1=5HDUMMY
*                     SUBRU2=5HSUBRU
*                     GO TO 50
```

```
*              2    (OPERATE SUBR2 UNDER THE NAME SUBRU)
*                           .
*                           .
*                           .
*                           .
*                           .
*              50   CALL LOCATE(SUBRU1,SUBRU2)
*                   CALL SUBR1
*                   CALL SUBR2
*                   ITIME=ITIME+1
*                   GO TO (1,2),ITIME
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY WHERE
*
* FORTRAN USAGE OF WHERE
*
*     CALL WHERE(SUBRU,IANS,LOC,NARGS)
*
* INPUTS TO WHERE
*
*    SUBRU     IS THE PROXY NAME, IN FORMAT(1A6), OF THE SUBROUTINE
*                TO BE FOUND ACCORDING TO THE DATA STORED BY LOCATE.
*                SUBRU SHOULD APPEAR AS ONE OF THE ARGUMENTS OF SOME
*                PRIOR CALL LOCATE STATEMENT.
*
* OUTPUTS FROM WHERE
*
*    IANS      =  0  MEANS SUBROUTINE WAS LOCATED.
*              = -1  MEANS NOT LOCATED, BUT THE TABLES ARE IN ORDER.
*              = -2  MEANS NOT LOCATED.  SUBRU WAS FOUND AS ONE OF THE
*                    ARGUMENTS OF A CALL LOCATE STATEMENT, BUT THE
*                    ASSOCIATED LIST OF CALL STATEMENTS WAS TOO SHORT
*                    TO EQUATE SUBRU WITH A REAL SUBROUTINE.
*              = -3  MEANS NOT LOCATED.  NO CALL LOCATE STATEMENTS HAVE
*                    BEEN MADE YET.
*              = -4  MEANS NOT LOCATED.  THE MEMORY CAPACITY (20) OF
*                    LOCATE HAS BEEN EXCEEDED AND THIS NAME MAY HAVE
*                    BEEN ASSOCIATED WITH A CALL LOCATE STATEMENT NOW
*                    FORGOTTEN.
*
*    LOC       IS UNDISTURBED UNLESS IANS=0.  IF IANS=0 THEN LOC GIVES
*                THE ABSOLUTE MACHINE ADDRESS OF THE ENTRY POINT OF THE
*                SUBROUTINE WHOSE PROXY NAME IS SUBRU (REAL NAME = SUBR)
*
*    NARGS     IS UNDISTURBED UNLESS IANS=0.  IF IANS=0 THEN NARGS IS
*                THE NO. OF ARGUMENTS OF SUBROUTINE SUBR AS WRITTEN DOWN
*                IN THE CALL SUBR STATEMENT IN THE LIST FOLLOWING THE
*                CALL LOCATE STATEMENT WHICH DEFINED SUBRU.  (THE PROXY
*                CALL STATEMENTS USE WHERE TO FIND SUBROUTINES BUT DO
*                NOT UTILIZE THE OUTPUT NARGS.)
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY CALL
*
* FORTRAN USAGE OF CALL
*
*              THE PROXY STATEMENT
*
*     CALL CALL(SUBRU,IANS,SPACER,ARG1,ARG2,...,ARGN)
*
*              IS FUNCTIONALLY EQUIVALENT TO THE STATEMENT
*
*     CALL SUBR(ARG1,ARG2,...,ARGN)
*
*              PROVIDED
*                1. THE NAME SUBRU HAS BEEN EQUATED TO SUBR BY A PRIOR
*                   CALL LOCATE STATEMENT
*                2. SUBR IS A SUBROUTINE WHICH ONLY USES INFORMATION
*                     A) FROM ITS ARGUMENTS ARG1...ARGN
*                   AND POSSIBLY
*                     B) FROM THE STATEMENTS FOLLOWING THE CALL CALL
*                        STATEMENT
*              SUBROUTINES WHICH UTILIZE INFORMATION PRIOR TO THEIR CALL
*              STATEMENTS MAY OR MAY NOT BE SUCCESSFULLY PROXIED (THE
*              SUBROUTINE SETUP DESCRIBED BELOW IS ONE WHICH CAN BE).
*              THE QUESTION MUST BE RESOLVED IN EACH CASE REFERENCING
```

```
*             THE LOGIC DETAILS OF SUBROUTINE CALL.
*
* INPUTS TO CALL
*
*    SUBRU     IS 6 HOLLERITH FOR THE PROXY NAME, IN FORMAT(1A6)
*
*    SPACER    IS ANY DUMMY VARIABLE (ITS VALUE IS NEITHER USED NOR
*                CHANGED)
*
*    ARG1...ARGN  ARE THE SUBROUTINE ARGUMENTS, N IN NUMBER WHERE N MAY
*                 BE ZERO.
*
* OUTPUTS FROM CALL
*
*    IANS      =  0  INDICATES NO TROUBLE, IN WHICH CASE THE IMPORTANT
*                    OUTPUTS ARE THOSE (IF ANY) OF SUBROUTINE SUBR
*              = -1 THRU -4 HAS SAME SIGNIFICANCE AS FOR SUBROUTINE
*                    WHERE, AND THE SUBROUTINE WAS NOT OPERATED.
*
*             THE SUBROUTINE CALL ITSELF MAY BE PROXIED.
*                    CALL CALL(4HCALL,IANS1,SPACER,SUBRU,IANS2,SPACER,
*                   1           ARG1,ARG2,...,ARGN)
*             IS EQUIVALENT TO THE ABOVE WHERE IANS1 NOW REFERS TO
*             WHETHER OR NOT CALL HAS APPEARED IN A CALL LOCATE
*             STATEMENT, AND IANS2 REFERS TO SUBRU.
*
*
* FAP DESCRIPTION OF THE FUNCTIONING OF CALL (BASIC OUTLINE)
*
*             THE STATEMENT CALL CALL(SUBRU,IANS,SPACER,ARG1,ARG2,...)
*             COMPILES TO
*
*               LOCALL  TSX  $CALL,4
*                 +1    TSX  A(SUBRU),4    A(C) STANDS FOR ADDRESS OF C
*                 +2    TSX  A(IANS),0
*                 +3    TSX  A(SPACER),0
*                 +4    TSX  A(ARG1),0
*                 +5    TSX  A(ARG2),0
*                             .
*                             .
*             CALL FINDS FROM WHERE THE ENTRY (LOC) OF THE DESIRED
*             SUBROUTINE, THEN CONVERTS LOCALL+3 TO READ TSX  LOC,4 ,
*             AND FINALLY TRANSFERS CONTROL TO LOCALL+3.
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY CALL2
*
* FORTRAN USAGE OF CALL2
*
*             CALL2 DIFFERS FROM CALL PRINCIPALLY IN THAT ALL THE
*             INFORMATION ABOUT THE SUBROUTINE BEING PROXIED IS PACKED
*             UP IN A SINGLE VECTOR, AND THAT THE CLASS OF PROXYABLE
*             SUBROUTINES IS SLIGHTLY SMALLER.
*
*             THE STATEMENT
*     CALL CALL2(SUBRUV,IANS)
*             IS FUNCTIONALLY EQUIVALENT TO THE STATEMENT
*     CALL SUBR(ARG1,ARG2,...,ARGN)
*             PROVIDED
*               1. THE SUBROUTINE VECTOR SUBRUV(1...N+4) IS SET UP IN
*                  ADVANCE AS DEFINED BELOW
*               2. SUBR IS A SUBROUTINE WHICH UTILIZES INFORMATION ONLY
*                  FROM ITS ARGUMENTS.
*
* INPUTS TO CALL2
*
*    SUBRUV(I) I=1,2,...,N+4  DEFINES THE SUBROUTINE DESIRED AND ITS
*             ARGUMENTS AS FOLLOWS
*               SUBRUV(1) = SUBRU = PROXY NAME OF SUBROUTINE, IN
*                                   FORMAT(1A6)
*                     (2) = N=NARGS = NO. OF ARGUMENTS OF THE
*                                     SUBROUTINE (FIXED POINT)
*                     (3) = FENCE = OCTAL 777777777777 (MUST BE PRESENT
*                                                  EVEN IF N=0)
*                     (4) = IXARG1    MUST
```

```
*                   (5) = IXARG2     BE
*                         .
*                         .          FIXED
*                         .
*                 (N+3) = IXARGN     POINT
*                 (N+4) = FENCE = OCTAL 777777777777 (MUST BE PRESENT
*                                                     EVEN IF N=0)
*               WHERE IXARG IS THE INDEX OF ARG WITH RESPECT TO THE
*               FORTRAN COMMON BLOCK, OBTAINABLE FOR EXAMPLE BY THE
*               STATEMENTS
*                   COMMON  COM
*                   IXARG = XLOCF(COM) - XLOCF(ARG) + 1
*
*             THE SUBROUTINE VECTOR, SUBRUV(I), IS A MIXED VECTOR WITH
*             A NAMING PROBLEM.  IT CAN BE CONSTRUCTED BY FORTRAN
*             STATEMENTS FOLLOWING AN EQUIVALENCE STATEMENT TO GIVE
*             SUBRUV(I) A FIXED POINT ALIAS, SAY ISUBRV(I), BUT THIS
*             PROCEDURE IS CUMBERSOME.  IT IS EASIER TO USE THE NEXT
*             ENTRY SETSBV TO CONSTRUCT SUBRUV(I) WITH A SINGLE CALL
*             STATEMENT.
*
* OUTPUTS     THE PRINCIPAL OUTPUTS ARE FROM THE SUBROUTINE OPERATED
*             (IF IT HAS OUTPUTS), BUT THE SUBROUTINE WILL BE OPERATED
*             ONLY IF THE IANS OUTPUT BELOW IS ZERO.
*
*    SUBRUV(I) CALL2 LEAVES SUBRUV(3,4,...,NARGS+4) MODIFIED, AND SOME
*               CAUTION MUST BE TAKEN IN REPEATED USE OF THE SAME CALL
*               CALL2 STATEMENT OR OF A SUBSEQUENT CALL CALL2 STATEMENT
*               INVOLVING THE SAME SUBROUTINE VECTOR SUBRUV(I).  THE
*               BASIC RULES ARE
*               1. REPEATED OR SUBSEQUENT USE WORKS PROPERLY IF
*                  A) SUBRUV(1...N+4) IS NOT DISTURBED BY THE CALLING
*                     PROGRAM FOLLOWING THE FIRST USE (I.E., LEFT THE
*                     WAY CALL2 MODIFIED IT).
*                     (NOTE THAT THIS ALLOWS ONE TO CHANGE ANY OR ALL
*                     OF THE VALUES OF THE ARGUMENTS ARG1,...,ARGN
*                     PROVIDED ONLY THAT THEIR LOCATIONS WITH RESPECT
*                     TO COMMON DONT CHANGE.)
*               OR B) SUBRUV(1...N+4) IS COMPLETELY RECONSTRUCTED
*                     BEFORE SUBSEQUENT USE
*               OR C) ONLY THE NAME OF THE SUBROUTINE IS CHANGED.
*               2. REPEATED USE WILL NOT WORK PROPERLY IF
*                  A) NARGS IS CHANGED IN SUBRUV(2), EVEN IF THE
*                     SUBROUTINE REQUESTED HAS A VARIABLE LENGTH
*                     CALLING SEQUENCE
*               OR B) SUBRUV(3...N+4) IS MODIFIED OR ONLY PARTIALLY
*                     RECONSTRUCTED.
*             REPEATED USE OF THE SAME SUBRUV VECTOR IN DIFFERENT
*             CALL STATEMENTS IS PERMITTED
*
*    IANS     =  0  INDICATES NO TROUBLE
*             = -1...-4 HAS SAME SIGNIFICANCE AS FOR SUBROUTINE WHERE
*                   AND THE SUBROUTINE WAS NOT OPERATED.
*             = -5  IF THIS IS THE FIRST CALL2 WITH THIS SUBRUV VECTOR
*                   (CALLED FIRST IF SUBRUV(3) = FENCE), BUT SOMETHING
*                   IS ILLEGAL ABOUT SUBRUV(I).
*             = -6  IF THIS IS A SECONDARY CALL2 WITH THIS SUBRUV
*                   VECTOR BUT THE VECTOR HAS BEEN ILLEGALLY CHANGED.
*
*             CALL2, LIKE CALL, IS ALSO PROXYABLE.  EITHER CALL OR
*             CALL2 CAN BE USED TO PROXY BOTH CALL AND CALL2.
*
*
* FAP DESCRIPTION OF THE FUNCTIONING OF CALL2 (BASIC OUTLINE)
*
*             THE STATEMENT CALL CALL2(SUBRUV,IANS) COMPILES TO
*
*             LOCALL TSX $CALL2,4   WITH  SUBRUV(N+4)=OCT  777777777777
*               +1   TSX A(SUBRUV,0       SUBRUV(N+3)=PZE  0,0,IXARG
*               +2   TSX A(IANS),0
```

```
*                +3                          .
*                                            .
*                                     SUBRUV(5)    PZE  0,0,IXARG2
*                                     SUBRUV(4)    PZE  0,0,IXARG1
*                                     SUBRUV(3)    OCT  777777777777
*                                     SUBRUV(2)    PZE  0,0,NARGS
*                                     SUBRUV(1)    BCI  1,SUBROU
*
*             CALL2 FINDS FROM WHERE THE ENTRY (LOC) OF THE DESIRED
*             SUBROUTINE AND SETS SUBRUV(N+4) TO READ TSX  LOC,4 .
*             SUBRUV(4...N+3) IS REVERSED AND CONVERTED TO READ
*             TSX A(ARGN),0 ... TSX A(ARG1),0  RESPECTIVELY.  SUBRUV(3)
*             IS REPLACED BY TRA  LOCALL+3, AND FINALLY CONTROL IS
*             TRANSFERRED TO SUBRUV(N+4).
*             ON A REPEAT USE OF THE SAME SUBRUV (DETECTED BY
*             SUBRUV(3) NOT = FENCE) SUBRUV(N+4) AND SUBRUV(3) ARE
*             RESET TO (POSSIBLY) NEW VALUES, BUT SUBRUV(4...N+3)
*             IS LEFT ALONE.
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY SETSBV
*
* FORTRAN USAGE OF SETSBV
*
*             SETSBV WILL CONSTRUCT A SUBROUTINE VECTOR SUBRUV(I) IN
*             THE FORMAT REQUIRED BY SUBROUTINE CALL2, BY THE SINGLE
*             STATEMENT
*
*    CALL SETSBV(SUBRU,SUBRUV,ARG1,ARG2,...,ARGN)
*
* INPUTS TO SETSBV
*
*   SUBRU     IS 6 HOLLERITH FOR THE SUBROUTINE NAME, IN FORMAT(1A6)
*
*   ARG1,...,ARGN  ARE THE ARGUMENTS, IF ANY, OF THE SUBROUTINE
*
* OUTPUTS FROM SETSBV
*
*   SUBRUV(I) I=1...N+4 IS THE SUBROUTINE VECTOR DESCRIBED ABOVE UNDER
*               CALL2 USAGE.  (SUBRUV(I) MUST BE DIMENSIONED AT LEAST
*               OF LENGTH N+4, WHERE N=0 IF THERE ARE NO ARGUMENTS)
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRIES SETUP AND RETURN
*
* FORTRAN USAGE OF SETUP AND RETURN
*
*             SETUP AND RETURN INSTRUMENT LINKAGE BETWEEN THE CALLING
*             PROGRAM AND A FORTRAN-II SUBROUTINE WHICH HAS VARIABLE
*             LENGTH CALLING SEQUENCE.  A CALL SETUP STATEMENT SHOULD
*             BE THE FIRST STATEMENT IN THE FORTRAN SUBROUTINE.  THE
*             CALL RETURN STATEMENT SHOULD APPEAR AT LEAST ONCE
*             ANYWHERE IN THE PROGRAM, FOR EXAMPLE -
*
*    SUBROUTINE SUBR(A,B,C)
*    (DIMENSION AND EQUIVALENCE STATEMENTS, ETC., IF ANY)
*    CALL SETUP(LOCALL,NARGS,XR1,XR2)
*     .
*     .
*     .
*    CALL RETURN(LOCALL,XR1,XR2)
*     .
*     .
*     .
*    CALL RETURN(LOCALL,XR1,XR2)
*     .
*     .
*     .
*
*             NOTE -
*               THE SUBROUTINE CARD NEED NOT HAVE ANY ARGUMENTS LISTED
*               ON IT, AND IF IT DOES THE NUMBER OF ARGUMENTS SO LISTED
*               NEED NOT CORRESPOND TO THE ACTUAL NO. (NARGS) OF
*               ARGUMENTS USED BY THE CALLING PROGRAM.  HOWEVER, IF THE
*               NUMBER OF ARGUMENTS LISTED ON THE SUBROUTINE CARD IS
*               LESS THAN THAT USED BY THE CALLING PROGRAM THE
*               SUBROUTINE HAS AN ACQUISITION AND/OR STORAGE PROBLEM
```

```
     *                CONCERNING SOME OR ALL OF ITS ARGUMENTS SINCE IT HAS NO
     *                SPECIFIC NAMES FOR THEM.  THIS PROBLEM IS SOLVED BY USE
     *                OF ONE OR MORE OF THE FIVE ENTRIES XINDEX, ARG, XARG,
     *                STORE, AND XNARGS WHICH ARE DISCUSSED BELOW.
     *
     * INPUTS TO SETUP
     *
     *              SETUP HAS NO INPUT ARGUMENTS EXCEPT INDEX REGISTER 4
     *              WHICH WAS SET BY THE CALL SETUP STATEMENT AND DEFINES
     *              THE LOCATION OF THAT STATEMENT.
     *              (SETUP PROCEEDS TO SCAN FROM THIS POINT BACKWARDS
     *              (LOWER ADDRESSES) TILL IT FINDS SXD X,Y INSTRUCTIONS
     *              (IN SUBROUTINE SUBR).)
     *
     * OUTPUTS FROM SETUP
     *
     *    LOCALL    IS THE ABSOLUTE MACHINE LOCATION OF THE CALL SUBR(ARG1,
     *                ...,ARGN) STATEMENT IN THE CALLING PROGRAM.  IN
     *                ORDINARY USAGE LOCALL SHOULD NOT BE CHANGED BY
     *                SUBROUTINE SUBR PRIOR TO THE CALL RETURN(LOCALL,
     *                XR1,XR2) STATEMENT.
     *
     *    NARGS     IS THE NO. OF ARGUMENTS ASSOCIATED WITH THE PRESENT CALL
     *                SUBR STATEMENT IN THE CALLING PROGRAM.
     *              (NARGS IS DETERMINED AS THE NO. OF SUCCESSIVE TSX Y,0
     *                INSTRUCTIONS WHICH IMMEDIATELY FOLLOW THE ADDRESS
     *                LOCALL).
     *
     *    XR1       IS THE VALUE TO WHICH INDEX REGISTER 1 MUST BE RESET
     *                (BY RETURN) BEFORE RETURNING TO THE CALLING PROGRAM.
     *
     *    XR2       IS THE ANALOGOUS VALUE FOR INDEX REGISTER 2.
     *
     *              IN ORDINARY USAGE XR1 AND XR2 SHOULD NOT BE CHANGED BY
     *              SUBROUTINE SUBR PRIOR TO THE CALL RETURN(LOCALL,XR1,XR2)
     *              STATEMENT.
     *
     *              CALL SETUP CAN NOT BE PROXIED BY A CALL2 STATEMENT.
     *
     * INPUTS TO RETURN
     *
     *    LOCALL    AS DEFINED UNDER OUTPUTS OF SETUP
     *
     *    XR1       AS DEFINED UNDER OUTPUTS OF SETUP
     *
     *    XR2       AS DEFINED UNDER OUTPUTS OF SETUP
     *
     * OUTPUTS FROM RETURN
     *
     *              RETURN RETURNS CONTROL PROPERLY TO THE CALLING PROGRAM
     *              AND RESTORES INDEX REGISTERS FOR THE CALLING PROGRAM.
     *
     *              BY USING RETURN IN A SLIGHTLY UNORTHODOX WAY IT IS
     *              POSSIBLE TO HAVE LOW LEVEL ROUTINES RETURN CONTROL
     *              DIRECTLY TO CALLING PROGRAMS 2 OR MORE LEVELS ABOVE THEM
     *              PROVIDED THE INTERMEDIATE ROUTINES PASS DOWN THE LOCALL,
     *              XR1, AND XR2 VALUES OF THE HIGH LEVEL ROUTINE.
     *              FOR EXAMPLE
     *
     *  LEVEL 1             LEVEL 2                  LEVEL 3
     *
     *      .               SUBROUTINE SUBA(...)     SUBROUTINE SUBB(LOC,X,Y)
     *      .               CALL SETUP(LOC,N,X,Y)        .
     *  CALL SUBA(...)          .                        .
     *      .                   .                        .
     *      .                   .                    CALL RETURN(LOC,X,Y)
     *      .               CALL SUBB(LOC,X,Y)           .
     *      .                   .                        .
     *      .                   .                        .
     *              (IN THIS USAGE IT IS IMMATERIAL WHETHER ANY OF THE
     *              ROUTINES INVOLVED HAVE A FIXED OR VARIABLE NO. OF
     *              ARGUMENTS.)
     *
     *
```

```
*             CALL RETURN CAN BE PROXIED BY CALL OR CALL2.
*
*             PROGRAMS USING LINKAGE BY SETUP AND RETURN CAN BE CALLED
*             BY PROXY, AND SKIP RETURNING TO PROGRAMS CALLED BY PROXY
*             IS PERMITTED.
*             SETUP AND RETURN PERMIT THE USER TO PROGRAM CIRCULAR
*             LOOPS BETWEEN SUBROUTINES.  A PROGRAM IN SUCH A LOOP MUST
*             SAVE LOCALL, XR1, XR2 IN SEPARATE LOCATIONS ACCORDING TO
*             THE SUBROUTINE WHICH CALLED IT (DETERMINED, FOR EXAMPLE,
*             FROM ONE OF ITS ARGUMENTS), AND THEN CALL RETURN WITH THE
*             APPROPRIATE VALUES OF LOCALL, XR1, AND XR2.  FOR
*             ILLUSTRATION SEE COMPUTATIONAL EXAMPLE 8.
*
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY XINDEX
*
*
* FORTRAN USAGE OF XINDEX (FUNCTION)
*
*             XINDEX FUNCTION ENABLES A VARIABLE-LENGTH-CALLING-
*             SEQUENCE SUBROUTINE TO LOCATE ABSOLUTELY ANY OF ITS
*             ARGUMENTS, AND REFER TO IT AS AN ORDINARY VECTOR (IN
*             COMMON), REGARDLESS OF WHETHER OR NOT ITS SUBROUTINE
*             CARD HAS NAMES FOR THE ARGUMENTS.  THE USAGE IS -
*
*     IXCOM=XINDEXF(LOCALL,NUMARG)
*
* INPUTS TO XINDEX
*
*    LOCALL    IS THE ABSOLUTE MACHINE ADDRESS OF THE CALLING STATEMENT
*                (AS PRODUCED BY SETUP)
*
*    NUMARG    IS THE ARGUMENT NUMBER DESIRED
*
* OUTPUTS FROM XINDEX
*
*    IXCOM     IS THE INDEX WITH RESPECT TO THE COMMON BLOCK OF ARGUMENT
*                NO. NUMARG.  FOR EXAMPLE -
*
*                    CALLING PROGRAM                SUBROUTINE
*
*                          .                     SUBROUTINE SUBR
*                          .                     DIMENSION COM(2)
*                          .                     COMMON COM
*                CALL SUBR(ARG1,ARG2,ARG3,...)   CALL SETUP(LOC,N,X1,X2)
*                (WHERE ARG3 IS SUPPOSED         IX=XINDEXF(LOC,3)
*                 TO BE A VECTOR)                A=COM(IX+4)
*                          .                     (THEN A = ARG3(5))
*                          .                             .
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRIES ARG AND XARG
*
* FORTRAN USAGE OF ARG AND XARG (FUNCTIONS)
*
*             ARG AND XARG FUNCTIONS GIVE A VARIABLE-LENGTH-CALLING-
*             SEQUENCE SUBROUTINE IMMEDIATE ACCESS TO ANY OF ITS
*             ARGUMENTS (INCLUDING ANY ELEMENT OF ANY VECTOR ARGUMENT).
*             THE FUNCTION IS SIMILAR TO THAT OF INDEX FUNCTION BUT
*             MORE DIRECT.  ARG AND XARG DIFFER ONLY ACCORDING TO
*             WHETHER THE USER WISHES TO GIVE THE ARGUMENT A FLOATING
*             OR FIXED POINT NAME RESPECTIVELY.  THE USAGE IS
*
*     ARGU  =  ARGF(LOCALL,NUMARG,IXVECT)
*           OR
*     IARGU = XARGF(LOCALL,NUMARG,IXVECT)
*
* INPUTS
*
*    LOCALL    IS THE ABSOLUTE MACHINE ADDRESS OF THE CALLING PROGRAM.
*
*    NUMARG    IS THE ARGUMENT NUMBER REQUIRED.  (EVERY ARGUMENT IS
*                CONSIDERED BY ARGF AND XARGF TO BE A VECTOR.)
*
*    IXVECT    IS THE SUBSCRIPT INDEX OF THE DESIRED ELEMENT WITHIN THE
```

```
*                REQUIRED ARGUMENT VECTOR.  (IF THE REQUIRED ARGUMENT
*                IS CONSIDERED BY THE SUBROUTINE TO BE A SINGLE
*                VARIABLE, NOT A VECTOR, THEN IXVECT SHOULD BE SET = 1)
*
* OUTPUTS
*
*    ARGU
*     OR
*    IARGU     IS THE DESIRED ELEMENT.
*
*              REFERRING TO THE EXAMPLE UNDER XINDEX, ARG3(5) COULD
*              EQUALLY WELL HAVE BEEN OBTAINED BY SUBR BY THE SINGLE
*              STATEMENT
*                    A = ARGF(LOCALL,3,5)
*              THUS BYPASSING THE NEED FOR THE DUMMY VECTOR, COM, AND
*              ITS ASSOCIATED DIMENSION AND COMMON STATEMENTS.
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY STORE
*
* FORTRAN USAGE OF STORE
*
*              STORE IS THE STORAGE COUNTERPART OF THE RETRIEVAL
*              FUNCTIONS ARG AND IXARG.  THE USAGE IS
*
*     CALL STORE (ARGU,LOCALL,NUMARG,IXVECT)
*
* INPUTS TO STORE
*
*    ARGU      IS THE QUANTITY TO BE STORED. (MAY HAVE FIXED POINT NAME)
*
*    LOCALL    SAME AS FOR ENTRY ARG
*    NUMARG        DITTO
*    IXVECT        DITTO
*
* OUTPUTS FROM STORE
*              ARGU IS STORED AS VECTOR ELEMENT NO. IXVECT, IN THE
*              VECTOR ARGUMENT NO. NUMARG, RELATIVE TO CALL STATEMENT
*              AT LOCALL.
*
*
*              AN IMPORTANT PROPERTY OF XINDEX, ARG, XARG, AND STORE
*              IS THAT THEIR PROCESSES ARE RELATIVE TO THE CONSTANT
*              LOCALL WHICH IS UNDER PROGRAM CONTROL.  LOCALL CAN BE
*              INITIALIZED BY ONE SUBROUTINE USING SETUP AND THEN PASSED
*              AS AN ARGUMENT UP OR DOWN THRU ARBITRARY SUBROUTINE
*              LEVELS PERMITTING THE ARGUMENTS OF THE INITIALIZING
*              SUBROUTINE TO BE TAPPED AS NEEDED BY THE OTHER ROUTINES.
*              THUS, IN THE EXAMPLE OF A SKIP RETURN GIVEN ABOVE, THE
*              SUBROUTINE SUBB AT LEVEL 3 COULD ACQUIRE DIRECTLY THE
*              ARGUMENTS PASSED TO SUBA FROM LEVEL 1 BY USING XINDEX,
*              ARG, OR XARG.  IF THE INITIAL ARGUMENT STRING IS VARIABLE
*              LENGTH IT MAY BE IMPORTANT THAT THE NUMBER OF ARGUMENTS
*              BE ACCESSIBLE AT ALL LEVELS.  THE INITIALIZING PROGRAM
*              CAN PASS THIS INFORMATION ALONG WITH LOCALL, BUT THIS
*              REQUIREMENT CAN BE SUPPRESSED BY REQUIRING THAT THE
*              ROUTINES OBTAIN THE ARGUMENT COUNT FROM THE NEXT ENTRY
*              XNARGS.
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY XNARGS
*
* FORTRAN USAGE OF XNARGS (FUNCTION)
*
*     NARGS=XNARGSF(LOCALL)
*
* INPUTS TO XNARGS
*
*    LOCALL    IS THE MACHINE ADDRESS OF ANY FORTRAN CALL STATEMENT
*                                        (ANY TSX X,4 INSTRUCTION)
*
* OUTPUTS FROM XNARGS
*
*    NARGS     = NUMBER OF ARGUMENTS ASSOCIATED WITH THE CALL STATEMENT
*                               EXCEPT
```

```
*              = -1 IF LOCALL IS NOT THE ADDRESS OF A CALL STATEMENT
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXSXXXXXXXXX ENTRY XNAME
*
* FORTRAN USAGE OF XNAME (FUNCTION)
*
*              XNAME FUNCTION IS A CONVENIENCE FOR MAKING IDENTITY
*              CHECKS BETWEEN TWO HOLLERITH NAMES OF 6 CHARACTERS OR
*              LESS, WHERE THE RELATIVE LEFT OR RIGHT ADJUSTMENT (IN
*              CASE OF LESS THAN 6 CHARACTERS) OF THE NAMES IS
*              CONSIDERED IMMATERIAL.  IT CAN BE USEFUL IN PROGRAM
*              SYSTEMS WHERE PROTOCOL IS BASED ON NAME EXCHANGING.  THE
*              USAGE IS
*
*     NEGDIF = XNAMEF(HNAME1,HNAME2)
*
* INPUTS TO XNAME
*
*    HNAME1    IS THE FIRST OF THE TWO HOLLERITH NAMES IN FORMAT(1A6).
*
*    HNAME2    IS THE SECOND OF THE TWO NAMES.
*
* OUTPUTS FROM XNAME
*
*    NEGDIF  (NEGATIVE IF DIFFERENT)  =  +0  IF THE NAMES MATCH
*                                     =  -1  IF THE NAMES DIFFER
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX COMPUTATIONAL EXAMPLES
*
*
* 1. EXAMPLES OF PAUSES ON ILLEGAL ARGUMENT COUNTS
*
*    IN THE FOLLOWING PROGRAM THE COMPUTER WILL STOP AFTER EACH
*    STATEMENT ON HPR 77777.  ASSUME THE OPERATOR RECORDS AC,MQ,XR4 AND
*    RESTARTS.
*    USAGE   -    1  CALL LOCATE
*                 2  CALL WHERE (A,B)
*                 3  CALL WHERE (A,B,C,D,E)
*                 4  CALL CALL(A)
*                 5  CALL CALL2
*                 6  CALL CALL2 (A,B,C)
*                 7  CALL SETSBV (A)
*                 8  CALL SETUP (A)
*                 9  CALL SETUP (A,B,C,D,E)
*                10  CALL RETURN
*                11  CALL RETURN(A,B,C,D)
*                12  CALL STORE(A,B,C,D,E)
*                13  CALL STORE(A,B)
*    OUTPUTS - PAUSE NO.       AC=              MQ=             XR4=
*
*                 1         434623216325  000000000000   ADDR. STATMNT 1
*                 2         663025512560  000000000002        PLUS 1
*                 3              DITTO    000000000005        PLUS 3
*                 4         232143436060  000000000001        PLUS 6
*                 5         232143430260  000000000000        PLUS 2
*                 6              DITTO    000000000003        PLUS 1
*                 7         622563622265  000000000001        PLUS 4
*                 8         622563644760  000000000001        PLUS 2
*                 9              DITTO    000000000005        PLUS 2
*                10         512563645145  0C0000000000        PLUS 6
*                11              DITTO    000000000004        PLUS 1
*                12         626346512560  000000000005        PLUS 5
*                13              DITTO    000000000002        PLUS 6
*                              (ADD 2 TO THE PLUS*S IF STD. ERR. PROC.)
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 2. EXAMPLE INVOLVING ONLY LOCATE AND WHERE, TO SHOW THE VARIOUS
*    CONDITIONS DISTINGUISHED BY WHERE
*
*    USAGE   -       DIMENSION  IANS(8), LOC(8), NARGS(8)
*                    DO 10  I=1,8
```

```
*                   IANS(I)=-99
*                   LOC(I) =-99
*              10   NARGS(I)=-99
*              1    CALL WHERE(6HANYSUB,IANS(1),LOC(1),NARGS(1))
*                   CALL LOCATE(6HLOCATE,6HLOCDUM)
*                   CALL LOCATE(A,B,C)
*                   CALL LOCATE
*              2    CALL WHERE(6HLOCATE,IANS(2),LOC(2),NARGS(2))
*              3    CALL WHERE(6HLOCDUM,IANS(3),LOC(3),NARGS(3))
*              4    CALL WHERE(6HANYSUB,IANS(4),LOC(4),NARGS(4))
*                   CALL LOCATE(5HWHERE,6H  DUM2)
*                   CALL WHERE(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)
*                   A=B
*              5    CALL WHERE(6H WHERE,IANS(5),LOC(5),NARGS(5))
*              6    CALL WHERE(4HDUM2,IANS(6),LOC(6),NARGS(6))
*                   CALL LOCATE(DUM)
*                   A=B
*                   CALL LOCATE(DUM)
*                   A=B
*                   CALL LOCATE(DUM)
*                   A=B
*                          (ETC,
*                           13 PAIRS IN TOTAL)
*                   CALL LOCATE(DUM)
*                   A=B
*              7    CALL WHERE(6HLOCATE,IANS(7),LOC(7),NARGS(7))
*              8    CALL WHERE(5HWHERE,IANS(8),LOC(8),NARGS(8))
*
*    OUTPUTS - I=   IANS(I)=              LOC(I)=               NARGS(I)=
*              1       -3                   -99                    -99
*              2        0             ADDRESS(LOCATE)                3
*              3        0              SAME AS LOC(2)                0
*              4       -1                   -99                    -99
*              5        0      ADDRESS(WHERE) (GRTHN LOC(2))        16
*              6       -2                   -99                    -99
*              7       -4                   -99                    -99
*              8        0              SAME AS LOC(5)               16
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 3. EXAMPLE OF PROXY CALL STATEMENTS BY CALL AND CALL2 (WITH AND
*    WITHOUT SETSBV)
*
*    INPUTS  - SUPPOSE THE SUBROUTINE TO BE PROXIED HAS THE FUNCTION OF
*              SETTING ITS SECOND ARGUMENT = TWICE ITS FIRST ARGUMENT
*              AS FOLLOWS
*                   SUBROUTINE DUBLER(I,K)
*                   K=2*I
*                   RETURN
*                   END
*
*    USAGE   -      DIMENSION SUBRUV(6),ISUBRV(6),COM(2),IANS(8),K(8)
*                   EQUIVALENCE (SUBRUV,ISUBRV)
*                   COMMON COM
*                   CALL LOCATE(6HDUBLER)
*                   CALL DUBLER
*              C FIRST TRY REPEATED USE OF CALL CALL
*                   DO 10 J=1,3
*                   I=J
*                   CALL CALL(6HDUBLER,IANS(I),SPACER,I,K(J))
*               10  CONTINUE
*              C NOW SET UP A SUBROUTINE VECTOR THE HARD WAY
*                   SUBRUV(1)=6HDUBLER
*                   ISUBRV(2)=2
*              B    SUBRUV(3)=777777777777
*                   ISUBRV(4)=XLOCF(COM)-XLOCF(I)+1
*                   ISUBRV(5)=XLOCF(COM)-XLOCF(KTEMP)+1
*              B    SUBRUV(6)=777777777777
*              C THEN TRY REPEATED USE OF CALL2 FROM THE SAME STATEMENT
*                   DO 20  J=4,6
*                   I=J
*                   CALL CALL2(SUBRUV,IANS(J))
*                   K(J)=KTEMP
*               20  CONTINUE
```

```
*              C NOW TRY IT FROM ANOTHER SPOT WITH THE SAME SUBRUV
*              C VECTOR
*                    I=7
*                    CALL CALL2(SUBRUV,IANS(7))
*                    K(7)=KTEMP
*              C NOW REESTABLISH SUBRUV THE EASY WAY AND RETRY CALL2
*                    CALL SETSBV(6HDUBLER,SUBRUV,8,K(8))
*                    CALL CALL2(SUBRUV,IANS(8))
*
*
*    OUTPUTS - IANS(1...8)=0
*              K(1...8)=2,4,6,8,10,12,14,16
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 4. EXAMPLES OF SETUP AND RETURN
*
*    INPUTS  - SUPPOSE COUNT1 IS A SUBROUTINE WHICH SETS ITS FIRST
*              ARGUMENT EQUAL TO ITS ARGUMENT COUNT (PROVIDED THE
*              COUNT IS NON-ZERO)
*                    SUBROUTINE COUNT1(ICOUNT)
*                    CALL SETUP(LOCALL,NARGS,XR1,XR2)
*                    IF (NARGS)  20,20,10
*               10   ICOUNT=NARGS
*               20   CALL RETURN(LOCALL,XR1,XR2)
*                    END
*
*    USAGE   -       DIMENSION ICOUNT(3)
*                    DO 10 I=1,3
*               10   ICOUNT(I)=0
*                    CALL COUNT1(ICOUNT(1),A,B,C,D,E,F)
*                    CALL COUNT1
*                    CALL COUNT1(ICOUNT(3))
*
*    OUTPUTS - ICOUNT(1...3) = 7,0,1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 5. EXAMPLE OF SETUP, RETURN, XINDEX, XARG AND STORE
*
*    INPUTS  - SUPPOSE A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE,
*              ADARGS, HAS THE FUNCTION OF SETTING THE FIRST ELEMENT OF
*              ITS FIRST ARGUMENT = NO. OF ITS ARGUMENTS, AND THE
*              SECOND ELEMENT = SUM OF THE REMAINING ARGUMENTS.  IT WILL
*              COMPUTE TWO WAYS, USING XARG AND XINDEX, AND COMPARE
*              BEFORE STORING. IT WILL EXIT FOR LESS THAN TWO ARGUMENTS.
*                    SUBROUTINE ADARGS
*                    DIMENSION  ICOM(2)
*                    COMMON     ICOM
*                    CALL SETUP(LOCALL,NARGS,XR1,XR2)
*                    IF (NARGS-2)  99,10,10
*              C FIRST COMPUTE USING XARG
*                10  ITEMP1=0
*                    DO 20  I=2,NARGS
*                20  ITEMP1=ITEMP1+XARGF(LOCALL,I,1)
*              C THEN USE XINDEX AND COMPARE RESULTS
*                    ITEMP2=0
*                    DO 30  I=2,NARGS
*                    IX=XINDEXF(LOCALL,I)
*                30  ITEMP2=ITEMP2+ICOM(IX)
*                    IF(ITEMP1-ITEMP2)  99,40,99
*              C SET OUTPUTS WITH STORE
*                40  CALL STORE(NARGS,LOCALL,1,1)
*                    CALL STORE(ITEMP1,LOCALL,1,2)
*               99   CALL RETURN(LOCALL,XR1,XR2)
*                    END
*
*    USAGE   -       DIMENSION IA(2),IB(2),IC(2)
*                    DO 10  I=1,2
*                    IA(I)=-99
*                    IB(I)=-99
*                10  IC(I)=-99
*                    CALL ADARGS(IA,1,2,3,4)
```

```
*                    CALL ADARGS(IB,1)
*                    CALL ADARGS(IC)
*                    CALL ADARGS
*
*   OUTPUTS - IA(1,2) = 5,10  IB(1,2)= 2,1  IC(1,2)= -99,-99
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 6. EXAMPLE OF TWO-WAY DIRECT COMMUNICATION THRU AN INTERMEDIATE
*    SUBROUTINE, USING SETUP,RETURN,XNARGS,XNAME,STORE
*
*    INPUTS  - SUPPOSE SUBROUTINE PASSER MERELY TRANSMITS ITS LOCALL,
*              XR1 AND XR2 VALUES TO NAMECK
*                    SUBROUTINE PASSER
*                    CALL SETUP(LOCALL,NARGS,XR1,XR2)
*                    CALL NAMECK(LOCALL,XR1,XR2)
*                    END
*              AND NAMECK HAS THE FUNCTION OF SETTING THE FIRST
*              ARGUMENT (RELATIVE TO LOCALL) EQUAL 1 OR -1 ACCORDING
*              TO WHETHER OR NOT THE REMAINING ARGUMENTS (ASSUMING AT
*              LEAST 3 TOTAL) ALL REPRESENT THE SAME HOLLERITH NAME,
*              AND THEN SKIP RETURNING.
*                    SUBROUTINE NAMECK(LOCALL,XR1,XR2)
*                    NARGS = XNARGSF(LOCALL)
*                    HNAME1 = ARGF(LOCALL,2,1)
*                    DO 10  I=3,NARGS
*                    HNAMET = ARGF(LOCALL,I,1)
*                    IF (XNAMEF(HNAME1,HNAMET))  20,10,10
*                 10 CONTINUE
*                    CALL STORE(1,LOCALL,1,1)
*                    GO TO 99
*                 20 CALL STORE(-1,LOCALL,1,1)
*                99  CALL RETURN(LOCALL,XR1,XR2)
*                    END
*
*   USAGE    -       CALL PASSER (IANS1,3H*A4,4H *A4,5H  *A4,6H   *A4)
*                    CALL PASSER (IANS2,6H   / /,3H/ /)
*                    CALL PASSER (IANS3,6HABCDEF,6HABCDEE)
*
*   OUTPUTS - IANS1 = 1  IANS2 = 1  IANS3 = -1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 7. EXAMPLE OF A GENERAL PURPOSE SUBROUTINE
*
* INPUTS     - SUPPOSE SUBROUTINE GNPURP MERELY OPERATES, USING CALL2,
*              AN ARBITRARY NUMBER OF SUBROUTINE VECTORS PASSED TO IT
*              AS ARGUMENT NOS. 2,3,... BUT REPORTS BACK, IN ITS FIRST
*              ARGUMENT (VECTOR) THE IANS RESULT OF CALL2 FOR EACH
*              SUBROUTINE
*                    SUBROUTINE GNPURP(IANS)
*                    DIMENSION IANS(2),COM(2)
*                    COMMON COM
*                    CALL SETUP(LOCALL,NARGS,XR1,XR2)
*                    IF (NARGS-1)  99,99,10
*                 10 DO 20  I=2,NARGS
*                    IX = XINDEXF(LOCALL,I)
*                    J=I-1
*                 20 CALL CALL2(COM(IX),IANS(J))
*                99  CALL RETURN(LOCALL,XR1,XR2)
*                    END
*
*              AND SUPPOSE SUBROUTINES DUBLER,COUNT1,ADARGS,PASSER AND
*              NAMECK ARE AS DEFINED PREVIOUSLY
*
* USAGE      -       DIMENSION SUBRVD(6),SUBRVC(9),SUBRVA(12),SUBRVP(7),
*                              ISUM(2),IANS(4)
*                    CALL LOCATE(6HDUBLER,6HCOUNT1,LHADARGS,6HPASSER)
*                    CALL DUBLER
*                    CALL COUNT1
*                    CALL ADARGS
*                    CALL PASSER
*              C (NOTE-NAMECK DOESNT NEED TO BE LOCATED, BUT IS NEEDED
```

```
*              C       FOR EXECUTION)
*                    CALL SETSBV(6HDUBLER,SUBRVD,16,IDUBL)
*                    CALL SETSBV(6HCOUNT1,SUBRVC,ICOUNT,A,B,C,D)
*                    CALL SETSBV(6HADARGS,SUBRVA,ISUM,3,5,7,11,13,17,19)
*                    CALL SETSBV(6HPASSER,SUBRVP,IANSP,4HSAME,6H  SAME)
*                    CALL GNPURP(IANS,SUBRVD,SUBRVC,SUBRVA,SUBRVP)
*
* OUTPUTS     - IANS(1...4) = 0  (MEANING ALL SUBROUTINES FOUND)  AND
*               IDUBL = 32  ICOUNT = 5  ISUM(1...2) = 8,75  IANSP = 1
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* 8. EXAMPLE OF USE OF SETUP AND RETURN FOR LOOP LOGIC BETWEEN
*    SUBROUTINES - A CALLS B, B CALLS C, C CALLS B, B RETURNS TO C,
*                  C CALLS B, B RETURNS TO A.
*
*    INPUTS  - SUPPOSE SUBROUTINES ADD1 AND ADD7 MERELY ADD 1 AND 7
*              RESPECTIVELY TO THEIR FIRST ARGUMENTS.  (ADD1 AND ADD7
*              WILL PLAY THE ROLES OF SUBROUTINES B AND C ABOVE
*              RESPECTIVELY.  ADD1 WILL HAVE THE PRIMARY LOGIC
*              RESPONSIBILITIES.)
*
*                    SUBROUTINE ADD1(ISUM,CALLER)
*                    CALL SETUP(LOCTMP,NARGS,XR1TMP,XR2TMP)
*                    ISUM=ISUM+1
*              C NOW FIND OUT WHO IS CALLING
*                    SUBAD7=4HADD7
*                    IF (XNAMEF(SUBAD7,CALLER))  10,20,20
*              C IF NOT ADD7 SET ASIDE LOCTMP, XR1TMP, XR2TMP,
*              C INITIALIZE COUNTER AND THEN CALL ADD7
*               10   LOCBAK=LOCTMP
*                    XR1BAK=XR1TMP
*                    XR2BAK=XR2TMP
*                    ICOUNT=1
*                    CALL ADD7(ISUM)
*              C IF IT IS ADD7, RETURN TO CALLING PROGRAM IF ICOUNT
*              C HAS REACHED 2.
*               20   IF (ICOUNT-2)  40,30,30
*               30   CALL RETURN(LOCBAK,XR1BAK,XR2BAK)
*              C IF ICOUNT IS STILL = 1, INCREASE IT BY 1 AND RETURN TO
*              C ADD7.
*               40   ICOUNT=ICOUNT+1
*                    RETURN
*              C NOTE - ABOVE STATEMENT IS EQUIVALENT TO
*              C CALL RETURN(LOCTMP,XR1TMP,XR2TMP)
*                    END
*
*                    SUBROUTINE ADD7(ISUM)
*                    ISUM=ISUM+7
*                    CALL ADD1(ISUM,6H  ADD7)
*                    ISUM=ISUM+7
*                    CALL ADD1(ISUM,5H ADD7)
*                    RETURN
*              C (NOTE - ABOVE RETURN STATEMENT WILL NEVER BE EXECUTED)
*                    END
*
*    USAGE   -       ISUM=0
*                    CALL ADD1(ISUM,4HMAIN)
*
*    OUTPUTS - ISUM=17   (I.E., 3 CALLS OF ADD1, 1 CALL OF ADD7 AND 1
*                         RETURN TO ADD7)
*
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
* PROGRAM FOLLOWS BELOW
*
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY LOCATE(SUBRU1,SUBRU2,...)
*
*
       HTR      0
       HTR      0
```

```
       HTR     0
       BCI     1,LOCATE
LOCATE SXD     LOCATE-4,1
       CLS     K1
       TSX     EXCHQR,1
* SEE IF THIS XR4 VALUE IS ALREADY IN TABLE
       PXA     0,4
       LXA     NKEYS,2
       TXL     NEWKEY,2,0    NOT IF NKEYS=0
 CASL  CAS     KEYS+1,2
       TRA     *+2
       TRA     LBACK         YES IT IS
       TIX     CASL,2,1
* PUT XR4 IN TABLE IF IT IS NEW,
* AND INDEX NKEYS AND NXKEY BY 1,
* BUT PREVENT NKEYS FROM EXCEEDING KTABLE (SET KOVER IF IT TRIES TO),
* AND RESET NXKEY TO 1 WHEN IT REACHES KTABLE+1
NEWKEY LXA     NXKEY,1
       STA     KEYS+1,1
       CLA     NXKEY
       ADD     K1
       STO     NXKEY
       LDQ     K1
       CAS     KTABLE
       STQ     NXKEY
       NOP
       CLA     NKEYS
       ADD     K1
       CAS     KTABLE
       STQ     KOVER
       TRA     *+3           KTABLE OR KTABLE+1
       STO     NKEYS
       TRA     LBACK
       NZT     KOVER         WHICH IS IT
       STO     NKEYS
* NOW GET INTERNAL SUBROUTINE SKIP TO HELP US GET BACK
* FIRST JUMP GIVES NO. ARGUMENTS OF THE TSX $LOCATE,4.
* IF MQ NEG, EXIIT TO 1,4. OTHERWISE, SKIP NARGS TIMES OR UNTIL
* MQ GOES NEGATIVE, WHICHEVER FIRST.
 LBACK TSX     SKIP,1
       TQP     *+2
       TRA     LEXIT
       PAX     0,2
 JUMP1 TXI     *+1,4,-1
       TSX     SKIP,1
       TQP     *+2
       TRA     LEXIT
       TIX     JUMP1,2,1
 LEXIT LXD     LOCATE-4,1
       LXD     LOCATE-3,2
       TRA     1,4
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY WHERE(SUBRU,IANS,LOC,NARGS)
*
*
* THE IANS OUTPUT OF WHERE IS
*      = 0     IF LOCATED OK.
*      =-1     NOT LOCATED, BUT THE TABLES ARE IN ORDER.
*      =-2     NOT LOCATED. IT WAS FOUND AS ONE OF THE ARGUMENTS OF
*                 A CALL LOCATE STATEMENT, BUT THE ASSOCIATED
*                 LIST OF CALL STATEMENTS WAS TOO SHORT.
*      =-3     NOT LOCATED, NO CALL LOCATE STATEMENT HAS BEEN MADE YET.
*      =-4     NOT LOCATED, BUT THE KEYS LIST HAS OVERFLOWED,
*                 AND IT MAY HAVE BEEN LOST FROM A PRIOR
*                 CALL LOCATE STATEMENT.
*
*
       BCI     1,WHERE
 WHERE SXD     LOCATE-4,1
       CLA     K4
       TSX     EXCHQR,1
* LEFT ADJUST THE NAME
       LDQ*    1,4           6HSUBROU
       TSX     ADJUST,1
```

```
       STQ     SUBROU
       CLA     3,4           A(LOC)
       STA     STOW1
       CLA     4,4           A(NARGS)
       STA     STOW2
* IMMEDIATE EXIT WITH IANS=-3 IF NKEYS=0
       ZET     NKEYS
       TRA     GOTAK
       CLS     KD3
* EXIT CHANNEL (ALSO USED BY CALL, CALL2)
LEAVEW LXD     LOCATE-2,4
       STO*    2,4
       TRA     SKIPEX
* OTHERWISE INITIALIZE IXKEY=0 AND START
 GOTAK STZ     IXKEY
* INDEX IXKEY AND CHECK IF IT EXCEEDS NKEYS
NXTKEY CLA     IXKEY
       ADD     K1
       STO     IXKEY
       CAS     NKEYS
       TRA     GIVUP
       NOP
       TRA     KEYOK
* IF SO SET IANS=-1 IF KOVER=0
*               =-4 IF KOVER=1, AND EXIT.
 GIVUP CLS     KD1
       NZT     KOVER
       TRA     LEAVEW
       CLS     KD4
       TRA     LEAVEW
* GET THIS KEY AND SET XR4 AND LARG WITH IT
 KEYOK PAX     0,2
       CLA     KEYS+1,2
       PAX     0,4           (SET FOR LATER SKIPPING)
       PAC     0,2
       SXA     LARG,2
* RUN DOWN THE ARGUMENTS OF LOCATE, UNTIL A NON-TSX X,0
* INSTRUCTION IS REACHED. XR2 KEEPS NEGATIVE TRACK OF ARG NO.
       AXC     1,2
 LARG  CAL     **,2          **=A(TSX $LOCATE,4)
       STA     GOTARG
       TSX     CKTSXZ,1
       TRA     NXTKEY
* FOR EACH TSX X,0 GET THE ARGUMENT AND COMPARE WITH SUBROU.
* BACK TO TRY NEXT ARGUMENT IF DOESNT MATCH
GOTARG LDQ     **            **=A(A(TSX $LOCATE,4)-XR2) XR2=-ARG NO.
       TSX     ADJUST,1
       STQ     TEMP
       CAL     TEMP
       LAS     SUBROU
       TRA     *+2
       TRA     MATCH
       TXI     LARG,2,-1
* WHEN IT MATCHES, -(XR2)=INDEX OF THE MATCHING ARG
* GET THIS INDEX AND USE SKIP THIS MANY TIMES
* TO LOCATE THE CORRESPONDING TSX $SUBROU,4
* BUT ERROR EXIT IF SKIP HITS A NON SPECIAL INSTRUCTION
 MATCH PXA     0,2
       PAC     0,2
 SKP   TSX     SKIP,1
       TQP     CONTIN
       CLS     KD2
       TRA     LEAVEW
CONTIN TXI     *+1,4,-1
       TIX     SKP,2,1
* FIGURE OUT THE SUBROUTINE ENTRY POINT AND SET LOC.
* ENTRY POINT = ADDRESS PORTION OF THAT REGISTER (IN THE TRANSFER
* VECTOR) WHOSE ADDRESS IS THE ADDRESS PORTION OF THE
* TSX $SUBROU,4 WHICH SKIP JUST STOPPED AT.
* (NOTE ADVANCE OF XR4 BY 1 AT CONTIN)
       CLA     0,4           PICKS UP TSX $SUBROU,4
       STA     *+1
       CAL     **
       ANA     MSK3
       ALS     18
```

LOCATE (PAGE 19)

```
 STOW1 SLW      **            **=A(LOC)
* USE SKIP AGAIN TO GET THE ARGUMENT COUNT.  SET IANS=0.  EXIT.
      TSX      SKIP,1
      ALS      18
 STOW2 STO      **            **=A(NARGS)
      CLA      K0            (SET IANS)
      TRA      LEAVEW
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY CALL(SUBRU,IANS,SPACER,ARG1,ARG2,...)
*
*
* CALL LEAVES IANS = 0 IF EVERYTHING OK
*                  = SAME AS FROM WHERE IF CANT FIND SUBROUTINE
*                            (-1,-2,-3,-4)
*
* SET WHICH CALL INDICATOR ICALL = 0 FOR CALL
*
      BCI      1,CALL
 CALL  SXD      LOCATE-4,1
* CALL MUST HAVE AT LEAST 2 ARGUMENTS AND 3,4 MUST BE SOME KIND OF A
* TSX (TSX X,4 ON REPEAT ENTRY)
      CLS      K2
CKCALL TSX      EXCHQR,1
      STZ      ICALL
      CAL      3,4
      TSX      CKTSXA,1
      TRA      CKCALL        (WITH AC HUGE,NEG OR POS)
      TRA      LXDCC2
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY CALL2 (SUBRUV,IANS)
*
*
* CALL2 LEAVES IANS = 0 IF EVERYTHING OK
*                   = SAME AS FROM WHERE IF SUBROUTINE NOT FOUND
*                             (-1,-2,-3,-4)
*                   = -5 IF FIRST CALL2 WITH THIS SUBRUV BUT SOMETHING
*                        IS ILLEGAL ABOUT SUBRUV
*                        (CONSIDERED FIRST CALL I.F.F. SUBRUV(3)=FENCE)
*                   = -6 IF SECONDARY CALL2 WITH THIS SUBRUV BUT
*                        SOMETHING IS ILLEGAL ABOUT SUBRUV
*
*
* SET ICALL NON-ZERO (=XR4) FOR CALL2
*
      BCI      1,CALL2
 CALL2 SXD      LOCATE-4,1
      CLA      K2
      TSX      EXCHQR,1
      SXA      ICALL,4
* THESE TWO CALLS RUN TOGETHER UNTIL THE SUBROUTINE IS FOUND
LXDCC2 LXD      LOCATE-4,1
      SXA      AXT4W,4
      CLA      1,4           A(6HSUBROU=SUBRUV(1))
      STA      TSXW1
      SUB      K1
      STA      NGET
      SUB      K1
      STA      GTFNCE
      STA      CALC2
      STA      CLAR
      STA      LDQR
      CLA      2,4           A(IANS)
      STA      ANSET
 NGET  CLS      **            **=A(NARGS) (CALL2)
      ARS      18
      ADD      1,4
      SUB      K3            A(SUBRUV(NARGS+4))
      STA      STTSX4
* USE WHERE TO FIND SUBROUTINE AND IANS (IT ALSO SAVES 6HSUBROU)
* AND THEN CHECK IANS
* (NOTE XR1 AND XR2 HAVE ORIGINAL VALUES AT THIS POINT.)
      TSX      WHERE,4
 TSXW1 TSX      **,0          **=A(6HSUBROU)
```

```
       TSX     TEMP2,0        IANS
       TSX     LOC,0
       TSX     TEMP3,0        (DUMMY FOR NARGS)
       CLA     TEMP2
 AXT4W AXT     **,4
       SXD     LOCATE-2,4
       TNZ     LEAVEW
* IF OK, FORK ON CALL TYPE
       ZET     ICALL
       TRA     NTCALL
* FOR CALL SET ADDRESS OF THE TSX LOC,4 = 3,4
       LDC     LOCATE-2,4
       TXI     *+1,4,3
       SXA     STTSX4,4
* THIS IS WHERE CALL AND CALL2 TRANSFER TO SUBROU (SETTING IANS=0)
 FUNEL LXD     LOCATE-4,1
       LXD     LOCATE-3,2
       CLA     K0
 ANSET STO     **             **=A(IANS)
       CLA     LOC
       ARS     18
       ADD     TSX4
STTSX4 STO     **             **=A(TSX LOC,4)
       TRA*    STTSX4
* IF SUBRUV(3) IS NOT 777... ASSUME THAT THIS IS A
* REPEAT ENTRY WITH THE SAME SUBRUV VECTOR AND
*      1. SUBRUV(3) = TRA X (BUT X MUST BE RESET)
*      2. SUBRUV(4,5,...,NARGS+3) IS STILL SET UP FROM LAST CALL2
*      3. SUBRUV(NARGS+4) IS STILL TSX Y,4 (BUT Y MUST BE RESET)
NTCALL CLA*    NGET           = NARGS
GTFNCE CAL     **             **=A(SUBRUV(3)) = FENCE OR TRA X
       LAS     FENCE
       HPR     CALL2-1        (MACHINE ERROR OR FENCE SMASHED)
       TRA     *+2            NEW
       TRA     REPEAT
* OTHERWISE WE WANT TO REVERSE AND CONVERT
* SUBRUV(4...,NARGS+3), PROVIDED NARGS ISNT ZERO,
* AND THEN SET LINKAGE IN SUBRUV(3)
* FIRST CHECK FENCE IN SUBRUV(NARGS+4)
       CAL*    STTSX4         = SUBRUV(NARGS+4)
       LAS     FENCE
       TRA     *+2
       TRA     *+2
       TRA     BADSBV
* IF OK CHECK FOR NEG OR ZERO NARGS
       CLA*    NGET
       TZE     RVOVR
       TPL     REV
* EXIT WITH IANS=-5 FOR ILLEGAL SUBRUV
BADSBV CLS     KD5
       TRA     LEAVEW
* FOR A REPEAT ENTRY, CHECK THAT WE STILL HAVE TRA X,
* NARGS IS NON-NEG, AND THE TSX*S ARE STILL THERE
REPEAT ANA     MSK1
       LAS     TRAZ
       TRA     *+2
       TRA     TRAOK
CHANGE CLS     KD6
       TRA     LEAVEW
 TRAOK CLA*    NGET
       TMI     CHANGE
       ADD     KD1            (ALWAYS HAVE A TSX Y,4)
       PDX     0,2
 CALC2 CAL     **,2           ** = A(SUBRUV(3))
       TSX     CKTSXA,1
       TRA     CHANGE
       TIX     CALC2,2,1
       TRA     RVOVR
* FORM (NARGS+1)/2 (TO CATCH MIDDLE TERM IF ODD)
* AND SET XR1 TO WORK ON HIGH END (LOW ADDRESSES)
*     AND XR2 TO WORK ON LOW END (HIGH ADDRESSES)
 REV   PDX     0,1            XR1=NARGS
       ADD     KD1
       ARS     1
       STD     TXLR
```

```
       AXT      1,2            XR2=1
* LOOP START
 CLAR  CLA      **,2           **=A(SUBRUV(3))
 LDQR  LDQ      **,1           **=A(SUBRUV(3))
* CONVERT AND EXCHANGE AC AND MQ
       TSX      CNVTAC,4
       XCA
       TSX      CNVTAC,4
* STORE THE VALUES AND INDEX FOR MORE
       STO*     CLAR
       STQ*     LDQR
       TXI      *+1,1,-1
       TXI      *+1,2,1
 TXLR  TXL      CLAR,2,**      **=(NARGS+1)/2
* FINISHED REVERSING.
* SET RETURN LINKAGE IN SUBRUV(3), AND GO ENTER SUBROUTINE
 RVOVR LDC      LOCATE-2,1
       TXI      *+1,1,3
       SXA      TRA,1
       CLA      TRA            TRA(A(TSX $CALL2,4)+3)
       STO*     GTFNCE
       TRA      FUNEL
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY SETSBV(SUBRU,SUBRUV,ARG1,ARG2,...)
*
*  SETS  SUBRUV(1) = 6HSUBROU
*              (2) = NARGS
*              (3) = OCT 777777777777 (= FENCE)
*              (4) = IXARG1
*              (5) = IXARG2
*                 ETC
*            (N+4) = OCT 777777777777
*
       BCI      1,SETSBV
SETSBV SXD      LOCATE-4,1
       CLS      K2
       TSX      EXCHQR,1
* COUNT ARGUMENTS (NARGS=COUNT-2).
       TSX      SKIP,1
       LXD      LOCATE-2,4
       SUB      K2
       PAX      0,2            XR2=NARGS
* SET SUBRUV(1...3)
       CLA*     1,4            6HSUBROU
       STO*     2,4
       CLA      2,4            A(SUBRUV(1))
       SUB      K1
       STA      STONRG         A(SUBRUV(2))
       SUB      K1
       STA      STOFNS         A(SUBRUV(3))
       STA      STOIXA
       CLA      FENCE
STOFNS STO      **             **=A(SUBRUV(3))
       PXD      0,2
STONRG STO      **             **=A(SUBRUV(2))=A(NARGS)
* THEN FILL IN SUBRUV(4...NARGS+3), IF NARGS NOT = 0.
       SXD      TXLCAL,2
       AXT      1,2            SR2 STORES SUBRUV(4,5,...)
       TZE      BAKFNS
       LDC      LOCATE-2,1     A(TSX $SETSBV,4)
       TXI      *+1,1,3
       SXA      CALTRG,1
       AXT      0,1            XR1 PICKS UP TSX ARG1, TSX ARG2, ...
CALTRG CAL      **,1           **=A(TSX ARG1,0)
       ANA      MSK3           EXTRACT ADDRESS
       SSM                     AND
       ADD      KCOMP1         CONVERT
       ALS      18
STOIXA STO      **,2           **=A(SUBRUV(3))
       TXI      *+1,1,-1
       TXI      *+1,2,1
TXLCAL TXL      CALTRG,2,**    **=NARGS
* ADD THE FINAL FENCE AT SUBRUV(NARGS+4) AND EXIT.
BAKFNS CLA      FENCE
```

```
       STO*    STOIXA
       TRA     SKIPEX
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY SETUP(LOCALL,NARGS,XR1,XR2)
*
*      SETUP SETS LOCALL, XR1, XR2 FROM FIRST SXD X,4, SXD Y,1, SXD Z,2
*              ENCOUNTERED IN FRONT OF THE TSX $SETUP,4
*
       BCI     1,SETUP
 SETUP SXD     LOCATE-4,1
       CLA     K4
       TSX     EXCHQR,1
* FIRST GET AND SET XR1, XR2, LOCALL=-XR4
       AXT     1,2
       TSX     FNDXRS,1
       STO*    3,4           XR1
       TSX     FNDXRS,1
       STO*    4,4           XR2
       TSX     FNDXRS,1
       PDC     0,1
       PXD     0,1
       STO*    1,4           LOCALL
       PDC     0,4           -LOCALL TO XR4 FOR NARGS COUNT
* THEN GET NARGS AND EXIT
       TSX     SKIP,1
       ALS     18            NARGS TO DECREMENT
       LXD     LOCATE-2,4
       STO*    2,4           NARGS
       TRA     SKIPEX
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY RETURN(LOCALL,XR1,XR2)
*
*      RETURNS CONTROL TO NEXT STATEMENT FOLLOWING TSX $SUB,4 AT LOCALL
*         RESTORES XR1 AND XR2 (FROM DECREMENTS)
*
       BCI     1,RETURN
RETURN SXD     LOCATE-4,1
       CLA     K3
       TSX     EXCHQR,1
       CLA*    2,4           XR1
       STD     LOCATE-4      (SAVED FOR SPCLEX TO RESTORE)
       CLA*    3,4           XR2
       PDX     0,2
       CLA*    1,4           LOCALL
       PDC     0,4
       TRA     SPCLEX
*
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY XINDEX(LOCALL,NUMARG)
*
* XINDEX FUNCTION SETS AC = INDEX WRT COMMON OF ARGUMENT NO. NUMARG
*     WHERE LOCALL = A(TSX $SUBROU,4).  NUMARG MAY BE NEGATIVE.
*
       BCI     1,XINDEX
XINDEX SXD     LOCATE-4,1
       TSX     GETSXZ,1
       CLA     TEMP
       SSM
       ADD     KCOMP1
       ALS     18
 EXITF LXD     LOCATE-4,1
       TRA     1,4
*
*
* XXXXX ENTRIES ARG(LOCALL,NUMARG,IXVECT) AND XARG(LOCALL,NUMARG,IXVECT)
*
* ARG FUNCTION AND XARG FUNCTION SET
*    AC = C(ADDRESS PORTION(LOCALL+NUMARG)-IXVECT+1)
*
       BCI     1,ARG
 ARG   EQU     *
 XARG  SXD     LOCATE-4,1
```

```
       TSX     GETSXZ,1
       STA     *+1
       CLA     **             **=A(ARG)
       TRA     EXITF
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY STORE(ARGU,LOCALL,NUMARG,IXVECT)
*
* STORE PUTS ARGU IN THE REGISTER WHOSE ADDRESS
*     = ADDRESS PORTION (LOCALL+NUMARG) -IXVECT +1
*
       BCI     1,STORE
 STORE SXD     LOCATE-4,1
       CLA     K4
       TSX     EXCHQR,1
* SET UP 77775, AC, MQ FOR GETSXZ
       CLA*    4,4            IXVECT
       STO     32765
       CLA*    2,4            LOCALL
       LDQ*    3,4            NUMARG
       TSX     GETSXZ,1
       STA     *+2
       CLA*    1,4            ARGU
       STO     **             **=STORAGE ADDRESS FOR ARGU
       TXI     EXITF,4,-4
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY XNARGS(LOCALL)
*
* XNARGS FUNCTION LEAVES
*         AC = NO. OF ARGUMENTS RELATIVE TO LOCALL
*            = -1 IF LOCALL NOT = TSX X,4
*
       BCI     1,XNARGS
XNARGS SXD     LOCATE-4,1
       SXD     LOCATE-2,4
       PDC     0,4
* CHECK FOR TSX X,4 AT LOCALL
       CAL     0,4
       TSX     CKTSX4,1
       TRA     CNTRGS
       CLS     KD1
 LXDN  LXD     LOCATE-2,4
       TRA     EXITF
* THEN COUNT ARGUMENTS AND LEAVE
CNTRGS TSX     SKIP,1
       TRA     LXDN
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY XNAME(HNAME1,HNAME2)
*
*
* FORTRAN FUNCTION COMPARING TWO HOLLERITH NAMES
*      AC=+0 IF SAME, =-1 IF DIFFERENT
*
* LEFT ADJUST THE TWO NAMES AND THEN COMPARE
       BCI     1,XNAME
 XNAME SXD     LOCATE-4,1
       STO     TEMP2
       TSX     ADJUST,1
       STQ     TEMP3
       LDQ     TEMP2
       TSX     ADJUST,1
       XCA
       CAS     TEMP3
       TRA     *+2
       TRA     SAME
* SET AC NEGATIVE IF DIFFERENT
       CLS     KD1
       TRA     *+2
* SET AC=0 IF THE SAME
 SAME  CLA     K0
       TRA     EXITF
*
*
```

```
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE EXCHQR
*
* SAVES INDEX REGISTERS 2 AND 4 AND CHECKS FOR LEGAL ARGUMENT COUNT
* FOR AN ILLEGAL COUNT IT
*       1.  SETS AC = 6H(NAME OF ENTRY INVOLVED)
*       2.  SETS MQ = NO. ARGUMENTS (IN ADDRESS)
*       3.  SETS XR4 = MACHINE ADDRESS OF ILLEGAL CALL STATEMENT
*       4.  STOPS ON HPR 77777
* IF RESTARTED IT WILL RETURN CONTROL TO CALLING PROGRAM
*
*       LINKAGE WITH XR1       (RETURN TO 1,1)
*       ASSUMES AC ADDRESS) = LEGAL COUNT IF POSITIVE (MAY BE ZERO)
*                           = MINIMUM LEGAL COUNT IF NEGATIVE
*       ASSUMES 0,4 = CALLING STATEMENT
*       ASSUMES -3,1 = BCI   1,(NAME OF ENTRY)
*       USES TEMP4, TEMP5, AND INTERNAL ROUTINE SKIP
*
EXCHQR SXD     LOCATE-3,2
       SXD     LOCATE-2,4
       SXA     SAV1Q,1
       LDQ     K1
       STZ     TEMP4           TEMP4 IS SWITCH
       TPL     *+2             (ZERO FOR EXACT COUNT)
       STQ     TEMP4
       SSP
       STO     TEMP5
       TSX     SKIP,1
       CAS     TEMP5
       ZET     TEMP4
       TRA     SAV1Q
* FOR ILLEGAL COUNT SET UP MQ, XR4, AC, AND PAUSE
       XCA
       LDC     LOCATE-2,4
       LXA     SAV1Q,1
       CLA     -3,1
       HPR     32767
* GENERAL EXIT FUNNEL USING SKIP
SKIPEX LXD     LOCATE-3,2
       LXD     LOCATE-2,4
SPCLEX TSX     SKIP,1          (USED BY RETURN)
       LXD     LOCATE-4,1
       TRA     1,4
* GOOD COUNT
 SAV1Q AXT     **,1
       LXD     LOCATE-2,4
       TRA     1,1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE SKIP
*
* SKIP TO NEXT TSX,4 OR TO NON-SPECIAL INSTRUCTION, WHICHEVER FIRST
*
*       LINKAGE WITH XR1, XR2 UNCHANGED
*       ASSUMES 1,4 IS FIRST LOCATION TO BE CHECKED
*       LEAVES AC = NO. TSX X,0 INSTRUCTIONS PASSED THRU (IN ADDRESS)
*       STOPS WHEN FINDS AN INSTRUCTION
*          1.      WHICH = TSX X,4
*       OR 2.      WHICH IS NOT=TSX X,0  (NTR,PZE PAIRS IGNORED)
*       LEAVES  MQ = PLUS FOR CASE 1., MINUS FOR CASE 2.
*       LEAVES  1,4 = TSX X,4 FOR CASE 1.
*                   = 1 BEYOND LAST TSX OF EITHER KIND FOR CASE 2.
*                     (BUT WONT BACK UP PAST ORIGINAL 1,4)
* EXAMPLES -
*       SUPPOSE ON INPUT 0,4=TSX A,B WHERE B IS ARBITRARY
*       LET NSI=NON SPECIAL INSTRUCTION,ANY=ANY INSTRUCTION,X,Y ARBITRARY
*       THEN SAMPLE OUTPUT SETTINGS OF XR4 ARE
*
*       CASES WITH NO TSX X,0 FOUND
*                                                  TSX A,B
*                                                  NTR
*         0,4        TSX A,B   TSX A,B   TSX A,B   PZE
*         1,4        NSI       NTR       TSX X,4   TSX X,4
*                              PZE       ANY       ANY
*                              NSI
*
```

```
     *      CASES WHERE ONE TSX X,0 FOUND
     *                                                       TSX A,B
     *                                                       TSX X,0
     *                  TSX A,B   TSX A,B   TSX A,B   NTR
     *        0,4       TSX X,0   TSX X,0   TSX X,0   PZE
     *        1,4       NSI       NTR       TSX Y,4   TSX Y,4
     *                            PZE       ANY       ANY
     *                            NSI
     *
      SKIP  SXA     SV2S,2
            SXA     SV1S,1
            TXI     *+1,4,-1
            SXD     TXLSKP,4
            TXI     *+1,4,1
            AXT     0,2           XR2 WILL COUNT NO. INSTRUCS=TSX X,0
     * FIRST CHECK FOR TSX X,0
      CALS  CAL     1,4
            TSX     CKTSXZ,1
            TRA     NOTSX
     * IF SO, INDEX, TSX X,0 COUNTER AND CONTINUE SCAN
            TXI     *+1,2,1
     * GO BACK
      BACKS TXI     CALS,4,-1
     * IF NOT TSX X,0 CHECK FOR NTR X,0,Y
     * GO BACK IF IT IS, BUT DONT INDEX COUNTER
      NOTSX ANA     MSK2          KNOCK OUT DECREMENT ALSO
            LAS     NTR
            TRA     ELSE
            TXI     BACKS,4,-1
     * CHECK FOR TSX X,4 WHEN ALL OTHER POSSIBLITIES FAIL
      ELSE  CAL     1,4           (WE LOST THE DECREMENT)
            TSX     CKTSX4,1
            TRA     T4            (YES)
     * FOR NON-SPECIAL INSTRUCTION BACK UP TILL 0,4=ANY KIND OF TSX
     * (FIRST SET MQ NEGATIVE)
            LDQ     MSK1
      BACK1 CAL     0,4
            TSX     CKTSXA,1
            TRA     *+2
            TRA     EXITS         GOT IT
     TXLSKP TXL     *+2,4,**      **=ORIG XR4-1
            TRA     EXITS
            TXI     BACK1,4,1
     * SET MQ POSITIVE FOR TSX X,4 AND EXIT
      T4    LDQ     K1
     * SET AC=NO. TSX X,0 INSTRUCTIONS AND RETURN
      EXITS PXA     0,2
      SV1S  AXT     **,1
      SV2S  AXT     **,2
            TRA     1,1
     *
     *
     * XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE ADJUST
     *
     *
     * LEFT ADJUST THE CONTENTS OF MQ AS HOLLERITH
     *      LINKAGE XR1
     *      RESULT IN MQ (=606060606060 IF ALL SPACES)
     *      AC LEFT=0
     *
     ADJUST SXA     ALEVE,2
            AXT     0,2           XR2 COUNTS NO. SPACES
     * CHECK SPACE
      CLAA  CLA     K0
            LGL     6
            CAS     K48
            TRA     *+2
            TRA     SPAFND
     * BACK PEDAL 1 AND LEAVE IF NOT SPACE
            LGR     6
     * LEAVE
      ALEVE AXT     **,2
            TRA     1,1
     * INSERT A SPACE IN MQ AND CHECK FOR 6
     SPAFND TXI     *+1,2,1
```

```
       STQ     TEMP
       CAL     TEMP
       ACL     K48
       LGR     36
       TXL     CLAA,2,4
       TRA     ALEVE
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE CNVTAC
*
* FORMS TSX A(ARG),0  IN AC  (FOR CALL2 ONLY)
*      LINKAGE WITH XR4 (RETURN TO 1,4)
*
CNVTAC ARS     18
       ANA     MSK3
       SSM
       ADD     KCOMP1
       ADD     TSX
       TRA     1,4
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE FNDXRS
*
*       FNDXRS FINDS FIRST SXD X,Y IN FRONT OF 0,4, AND THEN
*              LEAVES CONTENTS OF X IN AC
*         Y IS CONTROLLED BY XR2 (1 IMPLIES 1),(2 IMPLIES 2),
*              (3 IMPLIES 4)
*         LINKAGE WITH XR1
*         XR4 IS LEFT UNDISTURBED
*         XR2 IS LEFT INCREMENTED BY 1
*
FNDXRS SXA     SV4FXR,4
CALFXR CAL     -1,4           POSSIBLE SXD X,Y
       STA     CLAFXR
       ANA     MSK1
       LAS     SXD1+1,2
       TRA     *+2
       TXI     CLAFXR,2,1
       TXI     CALFXR,4,1
CLAFXR CLA     **             **=X
SV4FXR AXT     **,4
       TRA     1,1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINE GETSXZ
*
* GETSXZ SETS TEMP (ADDRESS)
*     = ADDRESS PORTION OF C(LOCALL+NUMARG)
*   SETS AC (ADDRESS) = DITTO-IXVECT+1
* LINKAGE WITH XR1
* ASSUMES AC = LOCALL, MQ = NUMARG, 77775 = IXVECT
*
GETSXZ STQ     TEMP
       ADD     TEMP
       ARS     18
       STA     *+1
       CAL     **             **=A(TSX X,0)
       ANA     MSK3
       STO     TEMP
       ALS     18
       SUB     32765
       ADD     KD1
       ARS     18
       TRA     1,1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXX INTERNAL SUBROUTINES CKTSX4, CKTSXA, CKTSXZ
*
* CHECK LOGICAL WORD IN AC, RETURNING AS FOLLOWS
*    CKTSX4-  1,1  IF AC = TSX X,4         ,  2,1  IF NOT
*    CKTSXA-  2,1  IF AC = TSX X,ANYTHING ,  1,1  IF NOT
*    CKTSXZ-  2,1  IF AC = TSX X,0         ,  1,1  IF NOT
*
CKTSX4 ANA     MSK1           KNOCK OUT ADDRESS
       LAS     TSX4
```

```
        TRA     2,1
        TRA     1,1
        TRA     2,1
CKTSXA  ANA     MSK4            KNOCK OUT ADDRESS AND TAG
        TRA     *+2
CKTSXZ  ANA     MSK1
        LAS     TSX
        TRA     1,1
        TRA     2,1
        TRA     1,1
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX TABLE OF CONSTANTS
*
*
KTABLE  PZE     14              USED ONLY BY LOCATE TO DIMENSION KEY TABLE
 K0     PZE     0
 K1     PZE     1
 K2     PZE     2
 K3     PZE     3
 K4     PZE     4
 K48    PZE     48             (= 1 SPACE IN BITS 30-35)
 KD1    PZE     0,0,1
 KD2    PZE     0,0,2
 KD3    PZE     0,0,3
 KD4    PZE     0,0,4
 KD5    PZE     0,0,5
 KD6    PZE     0,0,6
KCOMP1  OCT     000000077462    = ADDRESS OF COMMON BLOCK PLUS 1
KS2LXD  OCT     010000000000
 MSK1   OCT     777777700000
 MSK2   OCT     700000700000
 MSK3   OCT     000000077777
 MSK4   OCT     777777000000
 FENCE  OCT     777777777777
 TSX    TSX     0,0
 TSX4   TSX     0,4
        SXD     0,4
        SXD     0,2
 SXD1   SXD     0,1
 NTR    OCT     100000000000
 TRAZ   TRA     0,0,0
 TTR    TTR     0
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX TABLE OF VARIABLES
*
*
        BSS     12             (ETC UPWARDS)
        PZE     **             **=XR4 FROM SECOND CALL LOCATE
 KEYS   PZE     **             **=XR4 FROM FIRST CALL LOCATE (AND 15TH)
 NKEYS  PZE     0              **=NO. KEYS PRESENTLY IN TABLE
 NXKEY  PZE     1             **=INDEX OF NEXT KEY TO BE INSERTED IN TABLE
 KOVER  PZE     0              **=0(NORMAL),=1(MORE THAN 14 CALL LOCATES)
 IXKEY  PZE     **             **=INDEX OF KEY ABOUT TO BE USED
 TRA    TRA     **
SUBROU  PZE     **             **=6HSUBROU
 LOC    PZE     0,0,**
 TEMP   PZE     **
 TEMP2  PZE     **
 TEMP3  PZE     0,0,**
 TEMP4  PZE     **
 TEMP5  PZE     **
 ICALL  PZE     **
*
*
* XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   END
*
*
        END
```

```
*      LSHFT (FUNCTION)                 9/29/64   LAST CARD IN DECK IS NO. 0071
*      FAP
*LSHFT
       COUNT   100
       LBL     LSHFT
       ENTRY   LSHFT  (N,X)
       ENTRY   XLSHFT (N,X)
*
*                    ----ABSTRACT----
*
* TITLE - LSHFT
*         LOGICAL SHIFT FUNCTION
*
*              LSHFT RERFORMS A LOGICAL RIGHT OR LEFT SHIFT OF A WORD.
*
*              XLSHFT PERFORMS THE SAME OPERATION.
*
* LANGUAGE   - FAP FUNCTION  (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 12  REGISTERS
* SPEED      - ABOUT 25 MACHINE CYCLES.
* AUTHOR     - R.A. WIGGINS  JULY,1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     X1 =  LSHFTF(N,X)
*     I1 = XLSHFTF(N,X)
*
* INPUTS
*
*    N         IS THE NUMBER OF PLACES TO BE SHIFTED.
*              IF GRTHN 0 SHIFTING IS TO THE RIGHT.
*              IF LSTHN 0 SHIFTING IS TO THE LEFT.
*              MUST BE GRTHN= -35,  LSTHN= 35
*
*    X         IS WORD TO BE SHIFTED.
*              NEED NOT HAVE FLOATING POINT NAME.
*
* OUTPUTS
*
*    X1        IS THE SHIFTED WORD
*
*    I1        IS SAME AS X1.
*
* EXAMPLES
*
* 1. INPUTS  - N=6   X = OCT 774200011201
*    OUTPUTS - X1= OCT 007742000112
*
* 2. INPUTS  - N=-6  X = OCT 774200011201
*    OUTPUTS - X1= OCT 420001120100
*
* PROGRAM FOLLOWS BELOW
*
XLSHFT BSS     0
 LSHFT ARS     18
       STA     SFTR
       STA     SFTL
       CLM
       TPL     SFTR
 SFTL  LGL     **
       TRA     *+2
 SFTR  LGR     **
       LLS     0
       LGL     36
       TOV     *+1
       TRA     1,4
       END
```

```
*     LSLINE (SUBROUTINE)              10/1/64   LAST CARD IN DECK IS NO. 0081
*     LABEL
CLSLINE
      SUBROUTINE LSLINE (YY,LY,XMIN,XMAX,C0,C1)
C
C                    -----ABSTRACT----
C
C TITLE - LSLINE
C       LEAST-SQUARE LINE
C
C             LSLINE FITS A LINE TO AN EQUALLY SPACED INPUT SERIES BY
C             LEAST-SQUARES.  THAT IS, GIVEN AN EQUALLY SPACED DATA
C             SERIES Y(XMIN)...Y(XMAX), LSLINE FINDS THE COEFFICIENTS
C             C0 AND C1 SO THAT
C                                  2                               2
C                (Y(XMIN)-C0-C1*XMIN)  + ... + (Y(XMAX)-C0-C1*XMAX)
C
C             IS A MINIMUM.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 117 REGISTERS
C SPEED     - ABOUT 380 MACHINE CYCLES ON 709 (LESS ON 7090)
C AUTHOR    - R.A. WIGGINS
C
C                    -----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL LSLINE(YY,LY,XMIN,XMAX,C0,C1)
C
C INPUTS
C
C    YY(I)      I=1...LY CONTAINS THE DATA POINTS Y(XMIN)...Y(XMAX) FOR
C                 AN EQUALLY SPACED SERIES.
C
C    LY         MUST EXCEED ONE.
C
C    XMIN       IS THE X COORDINATE CORRESPONDING TO YY(1).
C
C    XMAX       IS THE X COORDINATE CORRESPONDING TO YY(LY).
C
C OUTPUTS
C
C    C0         IS THE FIRST COEFFICIENT FOR THE BEST LEAST-SQUARE LINE.
C
C    C1         IS THE SECOND COEFFICIENT.
C                 THUS, THE LINE IS GIVEN BY  C0 + C1*X.
C
C EXAMPLES
C
C 1. INPUTS  - LY = 5   YY(1...5) = 2.,3.,4.,5.,6.
C               XMIN = 2.  XMAX = 6.
C    OUTPUTS - C0 = 0.  C1 = 1.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT XMIN = 0.  XMAX = 4.
C    OUTPUTS - C0 = 2.  C1 = 1.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT XMIN = -1.  XMAX = 7.
C    OUTPUTS - C0 = 2.5   C1 = .5
C
      DIMENSION YY(10)
      XLY=LY
      DELX = (XMAX-XMIN)/(XLY-1.)
      X=XMIN-DELX
      SMX=0.
      SMXX=0.
      SMY=0.
      SMXY=0.
      DO 10 I=1,LY
      X=X+DELX
      SMX=SMX+X
      SMY=SMY+YY(I)
```

****
*   LSLINE   *
****
(PAGE 2)


```
      SMXX=SMXX+X*X
   10 SMXY=SMXY+X*YY(I)
      DEN = XLY*SMXX-SMX*SMX
      C0 = (SMY*SMXX-SMXY*SMX)/DEN
      C1 = (XLY*SMXY-SMX*SMY)/DEN
      RETURN
      END
```

```
*     LSSS1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0115
*     LABEL
CLSSS1
      SUBROUTINE LSSS1 (LL,AA,RR,GG,FF,ALP)
C
C                     ----ABSTRACT----
C
C TITLE - LSSS1
C        LEAST SQUARE SHAPER BY SIDEWAYS ITERATION
C
C              LSSS1 PERFORMS A SIDEWARDS ITERATION OF A SHAPER FILTER
C              F(K,L) (K REFERS TO THE K-TH ELEMENT IN A VECTOR OF
C              LENGTH L) TO CORRESPOND TO A SIMILAR ITERATION OF A
C              CROSSCORRELATION VECTOR G(K).  THAT IS, GIVEN A VECTOR
C              F(K,L) THAT SATISFIES THE EQUATIONS
C
C                F(L,L)*R(0)   + ... + F(1,L)*R(L-1)  = G(L-1)
C
C                F(L,L)*R(1)   + ... + F(1,L)*R(L-2)  = G(L-2)
C                .
C                .
C                F(L,L)*R(-L+1)+ ... + F(1,L)*R(0)    = G(0)
C
C              AND A(K,L-1) AND ALP(0,L-1) THAT CORRESPOND TO R(T) (AS
C              GIVEN BY RLSPR) THEN LSSS1 COMPUTES THE VECTOR F1(K,L)
C              WHICH SATISFIES
C
C                F1(L,L)*R(0)   + ... + F1(1,L)*R(L-1) = G(L-2)
C                .
C                .
C                .
C                F1(L,L)*R(-L+1)+ ... + F1(1,L)*R(0)   = G(-1)
C
C              SEE SUBROUTINE RLSSR FOR AN INTERPRETATION OF R(K) AND
C              G(K).
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 122 REGISTERS
C SPEED     - ABOUT .000210*L + .00019 SECONDS ON THE 7094 MOD 1 .
C AUTHOR    - R.A. WIGGINS   3/63
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINE - FDOT
C        AND FORTRAN SYSTEM ROUTINE - NONE
C
C FORTRAN USAGE
C     CALL LSSS1 (LL,AA,RR,GG,FF,ALP)
C
C INPUTS
C
C    LL         =L IS THE LENGTH OF A, R, AND F VECTORS.
C               MUST BE GRTHN=2
C
C    AA(I)      I=1,...,LL CONTAINS THE PREDICTION ERROR OPERATOR
C                 A(0,L-1) THROUGH A(L-1,L-1).
C
C    RR(I)      I=1,...,LL CONTAINS THE AUTOCORRELATION VECTOR R(0)
C                 THROUGH R(L-1).
C
C    GG(I)      I=1,...,LL+1 CONTAINS THE CROSSCORRELATION VECTOR G(-1)
C                 THROUGH G(L-1).
C
C    FF(I)      I=1,...,LL CONTAINS THE SHAPER FILTER F(1,L) THROUGH
C                 F(L,L).
C
C    ALP        CONTAINS THE ERROR COVARIANCE ALP(0,L-1)
C
C OUTPUTS
C
C    FF(I)      I=1,...,LL CONTAINS THE SHAPER FILTER F1(1,L) THROUGH
C                 F1(L,L).
C
```

```
C EXAMPLES
C
C 1. INPUTS  - LL=0  RR(1)=1.25,.5  GG(1...2)=0.,1.
C    USAGE   -       DIMENSION  FF(2)
C                    DO 10 I=1,2
C                    CALL RLSPR (LL,AA,RR,ALP)
C                 10 CALL RLSSR (LL,AA,RR,GG(2),FF,ALP)
C                    CALL LSSS1(LL,AA,RR,GG(1),FF,ALP)
C    OUTPUTS - LL=2  FF(1...2) = -0.381,0.9524
C
C 2. INPUTS  - LL=0  RR(1...5)=1.25,.5,0.,0.,0.
C                    GG(1...9)=0.,0.,0.,0.,1.,0.,0.,0.,0.
C    USAGE   -       DO 10  I=1,5
C                    CALL RLSPR (LL,AA,RR,ALP)
C                    CALL RLSSR (LL,AA,RR,GG(5),FF,ALP)
C                 10 CONTINUE
C                    DO 20  I=1,4
C                    J=5-I
C                    CALL LSSS1(LL,AA,RR,GG(J),FF,ALP)
C                 20 CONTINUE
C    OUTPUTS - LL=5  AA(1...5)=1.000,-0.498,0.246,-0.117,0.047
C                    FF(1...5)=0.047,-0.117,0.246,-0.498,0.999
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),RR(10),GG(10),FF(10)
      L1=LL
      L2=L1+2
      FL=FF(L1)
      DO 10  I=2,L1
      J=L2-I
      K=J-1
      FF(J)=FF(K)-FL*AA(I)
   10 CONTINUE
      CALL FDOT(L1-1,FF(2),RR(2),C1)
      F=(GG(1)-C1)/ALP
      FF(1)=F
      DO 20  I=2,L1
      FF(I)=FF(I)+F*AA(I)
   20 CONTINUE
      RETURN
      END
```

```
*     MATINV (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0078
*     LABEL
CMATINV
      SUBROUTINE MATINV (LA,A,B,SPACE,ERR)
C
C                    ----ABSTRACT----
C
C TITLE - MATINV
C          INVERSE OF A MATRIX
C
C              MATINV FINDS THE INVERSE OF AN N BY N DIMENSIONAL MATRIX.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 90 REGISTERS
C SPEED      - ABOUT 33*N*N*N + 110*N*N + 163*N + 209 MACHINE CYCLES
C                ON THE 7090.
C AUTHOR     - R.A. WIGGINS     2/63
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - SIMEQ
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MATINV(LA,A,B,SPACE,ERR)
C
C INPUTS
C
C    LA        IS THE DIMENSION OF ONE SIDE OF THE MATRIX.
C              MUST EXCEED ZERO.
C
C    A(I)      I=1,...LA*LA  IS A SQUARE MATRIX STORED CLOSELY PACKED
C                BY COLUMNS.  I.E.
C                       A(1...LA)      CONTAINS COLUMN 1.
C                       A(LA+1...2*LA) CONTAINS COLUMN 2.
C                       ETC.
C
C    SPACE(I)  I=1,...,LA*(LA+1)  IS COMPUTATIONAL SPACE NEEDED
C                BY MATINV.
C
C OUTPUTS
C
C    B(I)      I=1...LA*LA IS THE INVERSE OF A.  IT IS STORED BY
C                COLUMNS, CLOSELY PACKED.
C
C    ERR       =0.  IF SOLUTION WAS SUCCESSFUL.
C              =1.  IF OVERFLOW OCCURRED.
C              =2.  IF THE MATRIX A WAS SINGULAR.
C
C EXAMPLES
C
C 1. INPUTS  - LA = 2   A(1...4) =  3.00,  1.20,  2.50,  1.10
C    OUTPUTS -          B(1...4) =  3.67, -4.00, -8.33, 10.00
C
C 2. INPUTS  - LA = 2   A(1...4) =  3.00,  2.50,  1.20,  1.10
C    OUTPUTS -          B(1...4) =  3.67, -8.33, -4.00, 10.00
C
C PROGRAM FOLLOWS BELOW
C
C     DUMMY DIMENSION OF INPUT VARIABLES.
      DIMENSION A(9),B(9),SPACE(12)
      LAA=LA*LA
      LA1=LA+1
C
C     MOVE A INTO B, AND CLEAR PART OF SPACE
      DO 10  I=1,LAA
      B(I)=A(I)
 10   SPACE(I)=0.
C
C     SET UP A UNIT MATRIX IN SPACE.
      DO 20  I=1,LAA,LA1
 20   SPACE(I)=1.
C
C     INVERT A
```

MATINV
(PAGE 2)

```
      D=1.
      CALL SIMEQ (LA,LA,LA,B,SPACE,D,SPACE(LAA+1),ERR)
      RETURN
      END
```

```
*      MATML1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0136
*      FAP
*MATML1
       COUNT   100
       LBL     MATML1
       ENTRY   MATML1 (LA,A,B,C,M)
*
*                    ----ABSTRACT----
*
* TITLE - MATML1
*       SQUARE MATRIX MULTIPLICATION
*
*             MATML1 MULTIPLIES TWO SQUARE MATRICES, A AND B, TO OBTAIN
*             THE PRODUCT C.
*
*                     C = A * B
*
*             A, B AND C ARE ASSUMED TO BE CLOSELY PACKED BY COLUMNS.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - IBM 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 61 REGISTERS
* SPEED     - IF ZIFSTO=1.  ABOUT 18*N*N*N + 9*N*N + 10*N + 66
*               MACHINE CYCLES,
*             IF ZIFSTO=0.  ABOUT 18*N*N*N + 12*N*N + 10*N + 66
*               MACHINE CYCLES ON THE 7090,
*             WHERE ZIFSTO=0. IF THE PRODUCT IS STORED IN THE OUTPUT
*               AREA AND  =1. IF THE PRODUCT IS ADDED TO THE OUTPUT
*               AREA.
* AUTHOR    - R.A. WIGGINS      2/63
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MATML1(LA,A,B,C,M)
*
* INPUTS
*
*    LA         IS THE LENGTH OF THE COLUMNS (OR ROWS) OF A, B, AND C.
*               IS FORTRAN INTEGER
*               MUST BE GRTHN=1.
*
*    A(I)       I=1...LA*LA IS A SQUARE MATRIX STORED BY COLUMNS.
*                 I.E.  A(1...LA)       CONTAINS COLUMN 1.
*                       A(LA+1...2*LA) CONTAINS COLUMN 2.
*                       ETC.
*
*    B(I)       I=1...LA*LA IS A SQUARE MATRIX STORED BY COLUMNS.
*                 SEE ABOVE
*
*    M          =0 THE CONTENTS OF C ARE SET TO ZERO BEFORE
*                    MULTIPLICATION.
*               NOT =0 THE MULTIPLICATION IS ADDED TO THE PREVIOUS
*                    CONTENTS OF C.
*
* OUTPUTS
*
*    C(I)       I=1...LA*LA IS THE SQUARE MATRIX (STORED BY COLUMNS)
*                 THAT IS THE PRODUCT OF A AND B.
*
* EXAMPLES
*
* 1. INPUTS   - LA=2   A(1...4) = 1.,1.,2.,1.   B(1...4) = 2.,1.,3.,4.
*               M = 0
*
*    OUTPUTS - C(1...4) = 4.,3.,11.,7.
*
* 2. INPUTS   - SAME AS EXAMPLE 1. EXCEPT  M=1  C(1...4) = 1.,1.,1.,1.
*
*    OUTPUTS - C(1...4) = 5.,4.,12.,8.
*
```

*   MATML1   *
(PAGE 2)

```
      * PROGRAM FOLLOWS BELOW
      *
             HTR     0
             BCI     1,MATML1
      MATML1 SXD     *-2,4            SAVE
             SXA     EX,1             INDEX
             SXA     EX+1,2           REGISTERS.
             CAL     2,4              GET
             ADD     =1               ADDRESS
             STA     A                OF A.
             CAL     3,4              GET
             ADD     =1               ADDRESS
             STA     B                OF B.
             CAL     4,4              GET
             ADD     =1               ADDRESS
             STA     C                OF
             STA     C+1              C.
             STA     C1
             CLA*    1,4              GET
             STO     LA               LA
             STD     T10              AND SAVE.
             XCA                      DETERMINE
             MPY     LA
             ALS     17               LA*LA
             STO     LAA
             STO     LAA1
             STO     LAA2
             CLA*    5,4              DETERMINE
             TNZ     T1               MODE OF M.
             LXD     LAA,4            SET
       C1    STZ     **,4             C = 0.
             TIX     C1,4,1
       T1    LXD     LAA,1            LOAD
             LXD     LAA,2            INDEX
             LXD     LAA,4            REGISTERS.
       A     LDQ     **,1             CENTRAL
       B     FMP     **,2             LOOP.
       C     FAD     **,4
             STO     **,4
       T10   TIX     T30,1,**         A
             TIX     T15,1,1          INDEXED
             LXD     LAA,1            BY
             SXD     LAA1,1           ROWS.
             TIX     T13,2,1
       EX    AXT     **,1             EXIT
             AXT     **,2
             LXD     MATML1-2,4
             TRA     6,4
       T13   SXD     LAA2,2           B INDEXED
             TRA     T20              BY COLUMNS.
       T15   LXD     LAA1,1
             TIX     *+1,1,1
             SXD     LAA1,1
             LXD     LAA2,2
       T20   TIX     A,4,1
             TRA     EX
       T30   TIX     A,2,1
             TRA     EX
       LA    PZE
       LAA   PZE
       LAA1  PZE
       LAA2  PZE
             END
```

```
*     MATML3 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0104
*     LABEL
CMATML3
      SUBROUTINE MATML3 (N,M,L,AA,BB,TRAN,CC,M1)
C
C                     ----ABSTRACT----
C
C TITLE - MATML3
C       N X M MATRIX BY M X L MATRIX MULTIPLICATION
C
C             MATML3 MULTIPLIES AN N X M MATRIX A BY AN M X L MATRIX B
C             TO OBTAIN AN N X L PRODUCT MATRIX C.
C
C                        M        L          L
C                     (     )   ( )         ( )
C                   N (  A  ) * ( )    =  N (C)
C                     (     )   (B) M       ( )
C                               ( )
C                               ( )
C
C             A IS ASSUMMED TO BE STORED BY COLUMNS. B MAY BE STORED
C             BY COLUMNS OR ROWS.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 120 REGISTERS
C SPEED     - ABOUT  ((18*M + 98)*N + 14)*L + 108  MACHINE CYCLES ON
C               THE 7090.
C AUTHOR    - R.A. WIGGINS    3/63
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - DOTJ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL MATML3(N,M,L,AA,BB,TRAN,CC,M1)
C
C INPUTS
C
C    N          IS THE NUMBER OF ROWS IN A.
C               MUST BE GRTHN=1
C
C    M          IS NUMBER OF COLUMNS IN A.
C               IS NUMBER OF ROWS IN B (AFTER TRANSPOSITION)
C               MUST BE GRTHN=1
C
C    L          IS NUMBER OF ROWS IN B (AFTER TRANSPOSITION)
C               MUST BE GRTHN=1
C
C    AA(I)      I=1,...,N*M CONTAINS THE MATRIX A STORED BY COLUMNS.
C                 THAT IS
C                 AA(1...N)     CONTAINS COLUMN 1 OF A.
C                 AA(N+1...2*N) CONTAINS COLUMN 2 OF A.
C                 ETC.
C
C    BB(I)      I=1,...,M*L CONTAINS THE MATRIX B STORED BY EITHER ROWS
C                 OR COLUMNS.
C
C    TRAN       IF NCT = 0.  BB IS TRANSPOSED BEFORE MULTIPLICATION.
C               IF = 0.  THE MULTIPLICATION IS MADE WITH BB AS STORED.
C
C    M1         IF GRTHN 0 THE PRODUCT C IS ADDED TO THE VALUE OF C
C                 ON INPUT.
C               IF LSTHN=0 C IS CLEARED BEFORE MULTIPLICATION.
C
C OUTPUTS
C
C    CC(I)      I=1,...,N*L CONTAINS THE MATRIX C STORED BY COLUMNS.
C
C EXAMPLES
C
C 1. INPUTS  - N=1 M=1 L=1  AA(1)=2.  BB(1)=3.  M1=0 TRAN=0.
C    OUTPUTS - CC(1)=6.
C
```

```
C 2. INPUTS  - N=3 M=2 L=2  AA(1...6) = 1.,1.,3.,2.,7.,1.
C              TRAN=0.      BB(1...4) = 1.,5.,3.,7.
C                           CC(1...6) = 6.,0.,0.,0.,0.,0.  M1=1
C    OUTPUTS - CC(1...6) = 17.,36.,8.,17.,52.,16.
C
C 3. INPUTS  - SAME AS EXAMPLE 2. EXCEPT TRAN = 1.   M1 = 0
C    OUTPUTS - CC(1...6) = 7.,22.,6.,19.,54.,22.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),BB(10),CC(10)
      J3=0
      LI=L
      IF (TRAN) 5,6,5
 5    L1=LI
      L2=1
      L3=LI
      GO TO 7
 6    L1=LI*M
      L2=M
      L3=1
 7    DO 20 I=1,L1,L2
      J2=1
      DO 10  J=1,N
      J3=J3+1
      CALL DOTJ(M,N,AA(J),L3,BB(I),CC(J3),M1,1.)
 10   CONTINUE
 20   CONTINUE
      RETURN
      END
```

```
*       MATRA  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0176
*       FAP
*MATRA
        COUNT   100
        LBL     MATRA
        ENTRY   MATRA (A,N,M,ATRAN)
*
*                    ----ABSTRACT----
*
* TITLE - MATRA
*          MATRIX TRANSPOSE
*
*               MATRA FINDS THE MATRIX TRANSPOSE OF A MATRIX WHICH HAS
*               ITS ROWS CLOSELY PACKED. EQUIVALENCE OF INPUT AND OUTPUT
*               AREAS IS ALLOWED.  DURING THE PROCESS OF TRANSPOSITION,
*               THE LOW ORDER BIT (BIT 36) OF THE DATA WORDS IS SET TO
*               ZERO.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 92 REGISTERS
* SPEED      - ABOUT  (.000080 +OR- .000010)*N*M + .000200  SECONDS
*                 ON THE 7094 MOD 1 WHERE N*M IS THE TOTAL NUMBER OF
*                 ELEMENTS.
* AUTHOR     - R.A. WIGGINS AND S.M. SIMPSON,  3/63
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MATRA (A,N,M,ATRAN)
*
* INPUTS
*
*    A(I)       I=1,...,N*M  IS THE N X M MATRIX TO BE TRANSPOSED.  THE
*                 MATRIX IS ASSUMED TO BE STORED CLOSELY PACKED BY
*                 COLUMNS.
*
*    N          IS NUMBER OF ROWS IN THE INPUT MATRIX.
*               IS NUMBER OF COLUMNS IN THE OUTPUT MATRIX.
*               MUST BE GRTHN=1
*
*    M          IS NUMBER OF COLUMNS IN THE INPUT MATRIX.
*               IS NUMBER OF ROWS IN THE OUTPUT MATRIX.
*               MUST BE GRTHN=1
*
* OUTPUTS
*
*    ATRAN(I)  I=1,...,M*N  IS THE M X N TRANSPOSED MATRIX STORED
*                 CLOSELY PACKED BY COLUMNS.  THE LOW ORDER BIT
*                 HAS BEEN SET TO ZERO.
*               MAY BE EQUIVALENT TO A(I).
*
* EXAMPLES
*
* 1. INPUTS  - N=5  M=2  A(1...10)=1.,2.,3.,4.,5.,6.,7.,8.,9.,10.
*    OUTPUTS -         ATRAN(1...10)=1.,6.,2.,7.,3.,8.,4.,9.,5.,10.
*
* 2. INPUTS  - N=1  M=4  A(1...4)=1.,2.,3.,4.
*    OUTPUTS -         ATRAN(1...4)=1.,2.,3.,4.
*
* 3. INPUTS  - N=1  M=1  A(1) = 2.
*    OUTPUTS -         ATRAN(1) = 2.
*
* PROGRAM FOLLOWS BELOW
*
        HTR     0
        HTR     0
        HTR     0
        BCI     1,MATRA
MATRA   SXD     MATRA-4,1
        SXD     MATRA-3,2
        SXD     MATRA-2,4
```

```
     * GET ADDRESSES OF A AND ATRAN
            CAL     1,4
            ADD     =1B35
            STA     A
            CAL     4,4
            STA     AT1
            STA     AT2
            STA     AT3
            ADD     =1B35
            STA     AT
     * GET N AND M AND STORE IN NSCALD AND MM.
            CLA*    2,4
            TZE     EXIT
            TMI     EXIT
            ARS     18
            STO     NSCALD
            CLA*    3,4
            TZE     EXIT
            TMI     EXIT
            ARS     18
            STO     MM
            XCA
            MPY     NSCALD
            XCA
            PAX     ,1
            TIX     *+1,1,1
            SXD     ENDCC,1
            TXI     *+1,1,1
     * SET VDP,VLM, AND ALS INSTRUCTIONS, AND NSCALD
            CAL     =0777700777777
            ANS     VDP
            ANS     VLM
            CLA     MM
            ORA     =0233000000000
            FAD     =0233000000000
            ANA     =0077000000000
            ARS     9
            ORS     VDP
            ORS     VLM
            PDX     ,2
            SXA     ALS,2
            CLA     NSCALD
      ALS   ALS     **                   **=NO. BITS IN (M)
            STO     NSCALD
     * PUT A FLAG IN BIT 35 OF DATA AND MOVE TO OUTPUT AREA.
      A     CAL     **,1            **=ADR(A)+1
            ORA     =1B35
      AT    SLW     **,1            **=ADR(ATRAN)+1
            TIX     A,1,1
     *
     * TRANSPOSE THE OUTPUT
     * THE GENERAL SCHEME IS TO MOVE A WORD FROM I1 TO I2
     *
     *    I2 =  I1/N + XMODF(I1,N)*M
     *
     * THEN SET I1 = I2 AND REPEAT.
     *
     * WHEN WE LOOP BACK ON OURSELF WE SEARCH FOR A NEW BEGINNING POINT
     * UNTIL ALL TERMS ARE TRANSPOSED.
     *
     * BEGIN WITH I1=0 (FOR THE LOOP WE LET THE FIRST VALUE HAVE INDEX 0)
     *
            AXT     1,1             KEEPS TOTAL COUNT
            AXT     0,4             KEEPS PRESENT LOCATION
      AT3   CLA     **,4            **=ADR(ATRAN)
            LBT
            TXI     AT3,4,1
      NWI   STO     TEMP
            SXD     ECC,4
     * GET NEXT INDEX
      NXTI  LDQ     =0
            PXA     ,4
      VDP   VDP     NSCALD,0,**          **=NO. BITS IN (M)
            STO     T1
            STQ     T
```

```
        LDQ     MM
 VLM    VLM     T1,0,**          **= NO. BITS IN (M)
        ADD     T
        PAX     ,4
* EXCHANGE THE NUMBERS
 AT1    LDQ     **,4         **=ADR(ATRAN)
        CAL     TEMP
        ANA     =O777777777776
 AT2    SLW     **,4         **=ADR(ATRAN)
        STQ     TEMP
        TXI     *+1,1,1
 ECC    TXH     NXTI,4,**    **=FIRST LOOP VALUE.
* IF NOT ALL VALUES HAVE BEEN TRANSPOSED, SEARCH FOR NEXT STARTING POINT
 ENDCC  TXH     EXIT,1,**    **=N*M-1
 SRCH   TXI     AT3,4,1
 EXIT   LXD     MATRA-4,1
        LXD     MATRA-3,2
        LXD     MATRA-2,4
        TRA     5,4
*     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 T      PZE
 T1     PZE
 TEMP   PZE
 MM     PZE     **               **=M(IN ADDRESS)
NSCALD  PZE     **             **=N(IN ADDRESS)*2 EXP(NO.BITS IN(M))
*     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
        END
```

```
*      MATRA1 (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0094
*      FAP
*MATRA1
       COUNT   100
       LBL     MATRA1
       ENTRY   MATRA1 (LA,A)
*
*                        -----ABSTRACT-----
*
* TITLE - MATRA1
*        SQUARE MATRIX TRANSPOSE
*
*              MATRA1 TRANSPOSES A SQUARE, CLOSELY PACKED MATRIX.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - IBM 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 42 REGISTERS
* SPEED      - ABOUT  11*LA*LA + 9*LA + 50  MACHINE CYCLES ON THE 7090
*                WHERE LA IS NUMBER OF ROWS OR COLUMNS IN THE MATRIX.
* AUTHOR     - R.A. WIGGINS   2/63
*
*                        ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MATRA1(LA,A)
*
* INPUTS
*
*    A(I)       I=1...LA*LA IS A SQUARE, CLOSELY PACKED MATRIX.
*                 I.E.  A(1...LA)       CONTAINS COLUMN 1,
*                       A(LA+1...2*LA) CONTAINS COLUMN 2,
*                       ETC.
*
*    LA         MUST BE GRTHN=1.
*               IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*    A(I)       I=1...LA*LA IS THE MATRIX STORED BY ROWS.
*                 I.E.  A(1...LA)       CONTAINS ROW 1,
*                       A(LA+1...2*LA) CONTAINS ROW 2,
*                       ETC.
*
* EXAMPLES
*
* 1. INPUTS  - LA=2  A(1...4) = 1.,2.,3.,4.
*
*    OUTPUTS - A(1...4) = 1.,3.,2.,4.
*
* PROGRAM FOLLOWS BELOW
*
       HTR     0
       BCI     1,MATRA1
MATRA1 SXD     *-2,4
       SXA     EX,1
       SXA     EX+1,2
       CLA*    1,4           GET
       STO     LAA1          LA.
       STD     T20
       ADD     =1B17
       STD     T21
       LDQ     LAA1
       MPY     LAA1
       ALS     17
       STO     LAA1
       CAL     2,4           GET ADR(A)
       ADD     =1
       STA     A
       STA     A1
       STA     A2
       STA     A3
       LXD     LAA1,1        LOAD
```

****
* MATRA1 *
****
(PAGE 2)

```
        LXD     LAA1,2          INDICES.
        TRA     T20
A       CLA     **,1            LOOP.
A1      LDQ     **,2
A2      STQ     **,1
A3      STO     **,2
T20     TIX     T30,1,**        (LA)
        LXD     LAA1,1
T21     TIX     T25,1,**        (LA+1)
EX      AXT     **,1            EXIT.
        AXT     **,2
        LXD     MATRA1-2,4
        TRA     3,4
T25     SXD     LAA1,1
        LXD     LAA1,2
        TRA     T20
T30     TIX     A,2,1
        TRA     EX
LAA1    PZE
        END
```

```
************************
*    MAXAB             *
************************
 REFER TO
     MAXSN
```


```
************************
*    MAXAB             *
************************
 REFER TO
     MAXSN
```


```
************************
*    MAXABM            *
************************
 REFER TO
     MAXSNM
```


```
************************
*    MAXABM            *
************************
 REFER TO
     MAXSNM
```

```
*       MAXSN (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0169
*       FAP
*MAXSN
        COUNT   150
        LBL     MAXSN
        ENTRY   MAXSN (LX,X,XMAX1,I)
        ENTRY   MINSN (LX,X,XMIN1,I)
        ENTRY   MAXAB (LX,X,XMAX2,I)
        ENTRY   MINAB (LX,X,XMIN2,I)
*
*                    ----ABSTRACT----
*
* TITLE - MAXSN , WITH SECONDARY ENTRY POINTS MINSN, MAXAB, AND MINAB
*         FIND SIGNED OR UNSIGNED EXTREMAL VALUES OF A VECTOR.
*
*              MAXSN FINDS THE MAXIMUM SIGNED NUMBER, AND ITS INDEX, IN
*                A VECTOR OF NUMBERS (EITHER FIXED OR FLOATING POINT).
*
*              MINSN FINDS THE MINIMUM SIGNED NUMBER.
*
*              MAXAB FINDS THE MAXIMUM OF THE ABSOLUTE VALUES.
*
*              MINAB FINDS THE MINIMUM OF THE ABSOLUTE VALUES.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 54 REGISTERS
* SPEED     - APPROX. 14N MACHINE CYCLES, N = LENGTH OF VECTOR
* AUTHOR    - J.F. CLAERBOUT
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE FOR MAXSN
*     CALL MAXSN (LX,X,XMAX1,I)
*
* INPUTS
*
*    X(I)       I=1...LX IS A VECTOR OF NUMBERS.
*               MAY BE FIXED OR FLOATING POINT.
*
*    LX         IS FORTRAN II INTEGER.
*               MUST BE GRTHN=1.
*
* OUTPUTS
*
*    XMAX1      IS THE MAXIMUM SIGNED VALUE IN THE X VECTOR.
*
*    I          IS THE INDEX OF THE MAXIMUM SIGNED VALUE.
*                 I.E.   X(I) = XMAX1
*
* FORTRAN USAGE FOR MINSN
*     CALL MINSN (LX,X,XMIN1,I)
*
* INPUTS        SAME AS FOR MAXSN
*
* OUTPUTS
*
*    XMIN1      IS THE MINIMUM SIGNED VALUE IN THE X VECTOR
*
*    I          IS THE INDEX OF THE MINIMUM SIGNED VALUE.
*
* FORTRAN USAGE FOR MAXAB
*     CALL MAXAB (LX,X,XMAX2,I)
*
* INPUTS        SAME AS FOR MAXSN
*
* OUTPUTS
*
*    XMAX2      IS THE MAXIMUM ABSOLUTE VALUE IN THE X VECTOR.
*               NOTE THAT XMAX2 MAY BE NEGATIVE.
*
```

```
*    I          IS THE INDEX OF THE MAXIMUM ABSOLUTE VALUE.
*
* FORTRAN USAGE FOR MINAB
*       CALL MINAB (LX,X,XMIN2,I)
*
* INPUTS        SAME AS FOR MAXSN
*
* OUTPUTS
*
*    XMIN2      IS THE MINIMUM ABSOLUTE VALUE IN THE X VECTOR.
*               NOTE THAT XMIN2 MAY BE NEGATIVE.
*
*    I          IS THE INDEX OF THE MINIMUM ABSOLUTE VALUE.
*
* EXAMPLES
*
* 1. INPUTS   - X(1...10) = -11.,-8.,-5.,-2.,1.,4., 7.,10.,13.,16.
*               LX = 10
*    USAGE    -       CALL MAXSN (LX,X,XMAX1,I1)
*                     CALL MINSN (LX,X,XMIN1,I2)
*                     CALL MAXAB (LX,X,XMAX2,I3)
*                     CALL MINAB (LX,X,XMIN2,I4)
*    OUTPUTS -    XMAX1 = 16.     I1 = 10
*                 XMIN1 =-11.     I2 =  1
*                 XMAX2 = 16.     I3 = 10
*                 XMIN2 =  1.     I4 =  5
*
* 2. INPUTS   - X(1...10) = -16.,-13.,-10.,-7.,-4.,-1.,2.,5.,8.,11.
*               LX = 10
*    USAGE    - SAME AS EXAMPLE 1.
*    OUTPUTS -    XMAX1 = 11.     I1 = 10
*                 XMIN1 =-16.     I2 =  1
*                 XMAX2 =-16.     I3 =  1
*                 XMIN2 = -1.     I4 =  6
*
* 3. INPUTS   - X(1...10) = -16,-13,-10,-7,-4,-1,2,5,8,11   LX = 10
*    USAGE    - SAME AS EXAMPLE 1.
*    OUTPUTS      XMAX1 = 11     I1 = 10
*                 XMIN1 =-16     I2 =  1
*                 XMAX2 =-16     I3 =  1
*                 XMIN2 = -1     I4 =  6
*
       HTR     0
       BCI     1,MAXSN
 MAXSN CLA     MX
       STO     USE
       TRA     *+3
 MINSN CLA     MN
       STO     USE
       CLA     NOP
       STO     A-1
       CLA     SUB
       STO     A
       TRA     START
 MAXAB CLA     MX
       STO     USE
       TRA     *+3
 MINAB CLA     MN
       STO     USE
       CLA     SSP
       STO     A-1
       CLA     SBM
       STO     A
 START SXA     SV,1
       SXD     MAXSN-2,4
       CLA*    1,4
       PDX     ,1             ARRAY LENGTH TO IR1
       CLA     2,4
       ADD     =1
       STA     A+2
       STA     A
       CLA*    2,4            GET TRIAL
       STO*    3,4            EXTREMUM
       CLA     =1             SET CORRECT INDEX FOR TRIAL EXTREMUM
       ALS     18
```

****
*   MAXSN   *
****
(PAGE 3)


```
       STO     INDEX
LOOP   CLA*    3,4
       HTR     0          EITHER NOP OR SSP
 A     HTR     **,1       EITHER SUB OR SBM
 USE   HTR     B          EITHER TPL OR TMI
       CLA     **,1
       STO*    3,4
       SXD     INDEX,1
 B     TIX     LOOP,1,1
       CLA     INDEX
       STO*    4,4
 SV    AXT     **,1
       TRA     5,4
NOP    NOP
SUB    SUB     0,1
SSP    SSP
SBM    SBM     0,1
MX     TPL     B
MN     TMI     B
INDEX  BSS     1
       END
```

```
*      MAXSNM (SUBROUTINES)              9/4/64   LAST CARD IN DECK IS NO. 0246
*      FAP
*MAXSNM
       COUNT   200
       LBL     MAXSNM
       ENTRY   MAXSNM (FOFIJ,LI,LJ,IDIMEN,FMAXSN,IMAXSN,JMAXSN)
       ENTRY   MINSNM (FOFIJ,LI,LJ,IDIMEN,FMINSN,IMINSN,JMINSN)
       ENTRY   MAXABM (FOFIJ,LI,LJ,IDIMEN,FMAXAB,IMAXAB,JMAXAB)
       ENTRY   MINABM (FOFIJ,LI,LJ,IDIMEN,FMINAB,IMINAB,JMINAB)
*
*
*                    ----ABSTRACT----
*
* TITLE - MAXSNM, WITH SECONDARY ENTRIES MINSNM, MAXABM, AND MINABM
*        EXTREMAL VALUES OF MATRIX ELEMENTS
*
*              MAXSNM FINDS THE LARGEST ELEMENT OF A MATRIX.
*              MINSNM FINDS THE SMALLEST ELEMENT OF A MATRIX.
*              MAXABM FINDS THE ELEMENT WHOSE MAGNITUDE IS LARGEST.
*              MINABM FINDS THE ELEMENT WHOSE MAGNITUDE IS SMALLEST.
*
*              THE FORTRAN INDICES OF THE EXTREMAL VALUE ARE ALSO
*              GIVEN AS OUTPUTS.  THE MATRIX ELEMENTS MAY BE FIXED
*              OR FLOATING POINT.
*
* LANGUAGE    - FAP SUBROUTINE, FORTRAN II COMPATIBLE
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 61 REGISTERS
* SPEED       - APPROXIMATELY 14N MACHINE CYCLES, WHERE N IS THE
*               NUMBER OF ELEMENTS IN THE MATRIX.
* AUTHOR      - S.M. SIMPSON, MARCH 1964
*               (BASED ON THE VECTOR VERSION, MAXSN, BY J. CLAERBOUT)
*
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*        AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL MAXSNM(FOFIJ, LI, LJ, IDIMEN, FMAXSN, IMAXSN, JMAXSN)
*     CALL MINSNM(FOFIJ, LI, LJ, IDIMEN, FMINSN, IMINSN, JMINSN)
*     CALL MAXABM(FOFIJ, LI, LJ, IDIMEN, FMAXAB, IMAXAB, JMAXAB)
*     CALL MINABM(FOFIJ, LI, LJ, IDIMEN, FMINAB, IMINAB, JMINAB)
*
*
* INPUTS TO ALL ENTRIES
*
*    FOFIJ(I,J)  I=1...LI,  J=1...LJ  IS THE MATRIX TO BE SCANNED.  ITS
*                     MODE MAY BE EITHER FLOATING POINT OR FIXED POINT.
*
*    LI         MUST EXCEED ZERO (NOT CHECKED).
*
*    LJ         MUST EXCEED ZERO (NOT CHECKED).
*
*    IDIMEN     IS THE CALLER'S DIMENSION FOR THE INDEX  I  IN
*                  FOFIJ(I,J).
*               MUST BE  GRTHN=  LI  (NOT CHECKED).
*
*
* OUTPUTS FROM MAXSNM
*
*    FMAXSN     IS A VALUE SELECTED FROM THE MATRIX SUCH THAT
*                        FMAXSN GRTHN= FOFIJ(I,J)  OVER  I=1...LI, J=1...LJ.
*
*    IMAXSN, JMAXSN  ARE INDICES FOR WHICH
*                        FOFIJ(IMAXSN,JMAXSN) = FMAXSN.
*
*
* OUTPUTS FROM MINSNM
*
*    FMINSN     IS A VALUE SELECTED FROM THE MATRIX SUCH THAT
*                        FMINSN LSTHN= FOFIJ(I,J)  OVER  I=1...LI, J=1...LJ.
*
```

```
*    IMINSN, JMINSN  ARE INDICES FOR WHICH
*                    FOFIJ(IMINSN,JMINSN) = FMINSN.
*
*
* OUTPUTS FROM MAXABM
*
*    FMAXAB    IS A VALUE SELECTED FROM THE MATRIX SUCH THAT
*                    // FMAXAB // GRTHN=  FOFIJ(I,J)  OVER  I=1...LI,
*                                                            J=1...LJ
*                 WHERE  //  //  STANDS FOR ABSOLUTE VALUE.
*              NOTE THAT FMAXAB MAY BE NEGATIVE.
*
*    IMAXAB, JMAXAB  ARE INDICES FOR WHICH
*                    FOFIJ(IMAXAB,JMAXAB) = FMAXAB.
*
*
* OUTPUTS FROM MINABM
*
*    FMINAB    IS A VALUE SELECTED FROM THE MATRIX SUCH THAT
*                    // FMINAB //  LSTHN=  FOFIJ(I,J)  OVER  I=1...LI,
*                                                            J=1...LJ .
*              NOTE THAT  FMINAB  MAY BE NEGATIVE.
*
*    IMINAB, JMINAB  ARE INDICES FOR WHICH
*                    FOFIJ(IMINAB,JMINAB) = FMINAB.
*
*
*              CHOICE OF VALUES IN CASE OF DUPLICATE EXTREMALS
*
*    FOR ALL ENTRIES THE ORDER OF SCANNING IS, IN TERMS OF THE INDICES
*    I,J,      LI...1,1     LI...1,2     ETC.     LI...1,LJ  .
*    IF THE EXTREMAL VALUE OCCURS MORE THAN ONCE THE INDICES SELECTED
*    FOR OUTPUT CORRESPOND EITHER TO THE FIRST OR TO THE LAST OCCURRENCE
*    OF THE VALUE IN THIS SCAN ORDER, ACCORDING TO THE FOLLOWING TABLE.
*
*                     EXTREMAL POSITIVE    EXTREMAL NEGATIVE
*              MAXSNM        FIRST               LAST
*              MINSNM        LAST                FIRST
*              MAXABM        FIRST               FIRST
*              MINABM        LAST                LAST
*
*
* EXAMPLES
*
* 1. INPUTS   - F(1...3,1...3) = 5.,-2.,8.,, 3.,2.,4.,, -12.,-5.,-12.
*    USAGE    -        DIMENSION  F(5,3)
*                      DO 10  LJ=1,3
*                      DO 10  LI=1,3
*                      CALL MAXSNM(F, LI, LJ, 5, F1(LI,LJ), I1(LI,LJ),
*                    1              J1(LI,LJ))
*                      CALL MINSNM(F, LI, LJ, 5, F2(LI,LJ), I2(LI,LJ),
*                    1              J2(LI,LJ))
*                      CALL MAXABM(F, LI, LJ, 5, F3(LI,LJ), I3(LI,LJ),
*                    1              J3(LI,LJ))
*                 10   CALL MINABM(F, LI, LJ, 5, F4(LI,LJ), I4(LI,LJ),
*                    1              J4(LI,LJ))
*
*    OUTPUTS -         (FOR  LI  =  1   2   3,, 1   2   3,,   1    2    3)
*              F1(1...3,1...3) = 5., 5., 8.,,5., 5., 8.,,   5.,  5.,  8.
*              I1(1...3,1...3) =  1,  1,  3,, 1,  1,  3,,   1,   1,   3
*              J1(1...3,1...3) =  1,  1,  1,, 1,  1,  1,,   1,   1,   1
*              F2(1...3,1...3) = 5.,-2.,-2.,,3.,-2.,-2.,,-12.,-12.,-12.
*              I2(1...3,1...3) =  1,  2,  2,, 1,  2,  2,,   1,   1,   3
*              J2(1...3,1...3) =  1,  1,  1,, 2,  1,  1,,   3,   3,   3
*              F3(1...3,1...3) = 5., 5., 8.,,5., 5., 8.,,-12.,-12.,-12.
*              I3(1...3,1...3) =  1,  1,  3,, 1,  1,  3,,   1,   1,   3
*              J3(1...3,1...3) =  1,  1,  1,, 1,  1,  1,,   3,   3,   3
*              F4(1...3,1...3) = 5.,-2.,-2.,,3., 2., 2.,,   3.,  2.,  2.
*              I4(1...3,1...3) =  1,  2,  2,, 1,  2,  2,,   1,   2,   2
*              J4(1...3,1...3) =  1,  1,  1,, 2,  2,  2,,   2,   2,   2
*
* 2. INPUTS   - IF(1...3,1...3) = 5,-2,8,, 3,2,4,, -12,-5,-12
*    USAGE    - SIMILAR TO EXAMPLE 1., REPLACING  F BY IF,  F1 BY IF1,
*                 ETC.
*    OUTPUTS - SIMILAR TO EXAMPLE 1., EXCEPT  IF1, IF2, ...  WILL BE
```

```
*             FIXED POINT.
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR    0              XR4
       BCI    1,MAXSNM
*
* PRINCIPAL ENTRY. MAXSNM(FOFIJ,LI,LJ,IDIMEN,FMAXSN,IMAXSN,JMAXSN)
*
MAXSNM CLA    MX
       TRA    MINSNM+1
*
* SECOND ENTRY.  MINSNM(FOFIJ,LI,LJ,IDIMEN,FMINSN,IMINSN,JMINSN)
*
MINSNM CLA    MN
       STO    TEST
       CLA    NOP
       LDQ    SUB
       TRA    MERGE
*
* THIRD ENTRY.  MAXABM(FOFIJ,LI,LJ,IDIMEN,FMAXAB,IMAXAB,JMAXAB)
*
MAXABM CLA    MX
       TRA    MINABM+1
*
* FOURTH ENTRY.  MINABM(FOFIJ,LI,LJ,IDIMEN,FMINAB,IMINAB,JMINAB)
*
MINABM CLA    MN
       STO    TEST
       CLA    SSP
       LDQ    SBM
*
* FINISH SETTING UP THE LOOP INSTRUCTIONS.
*
 MERGE STO    SIGN
       STQ    DIFF
       SXD    MAXSNM-2,4
       SXA    LEAVE,1
       SXA    LEAVE+1,2
       CLA    1,4            A(FOFIJ)
       ADD    K1
       STA    DIFF
       CLA*   4,4            IDIMEN
       ARS    18
       STA    IDIM
*
* SET XR2 TO COUNT J = 1,2,...,LJ
*     XR1 TO COUNT I = LI,LI-1,...,1
* THEN ENTER LOOP SO THAT FIRST TRIAL IS SET = FOFIJ(LI,1)
*
       CLA*   2,4            LI
       PDX    0,1
       CLA*   3,4            LJ
       STD    TXL
       AXT    1,2
       TRA    CLADIF
 LOOP  CLA*   5,4            FMAXAB OR FMINAB OR ETC
 SIGN  NOP          NOP      OR SSP
 DIFF  NOP          SUB**,1 OR SBM**,1    ** = A(FOFIJ)+1  (-)DIM,...)
 TEST  NOP          TPL TIX OR TMI TIX
*
* REPLACE FMAXAB BY NEW TRIAL IF FALLS THRU TEST
*
CLADIF CLA*   DIFF
       STO*   5,4            NEW TRIAL STORED
       PXD    0,1
       STO*   6,4            NEW I STORED
       PXD    0,2
       STO*   7,4            NEW J STORED
 TIX   TIX    LOOP,1,1
*
* RESET FOR NEXT LINE
*
```

```
      CAL    DIFF        (NOTE DIFF MAY BE NEG OR POS)
      SUB    IDIM
      STA    DIFF
      CLA*   2,4         RESET XR1 TO LI
      PDX    0,1
      TXI    *+1,2,1
TXL   TXL    LOOP,2,**   ** = LJ
LEAVE AXT    **,1
      AXT    **,2
      TRA    8,4
*
* CONSTANTS, VARIABLES
*
NOP   NOP
SUB   SUB    **,1
SSP   SSP
SBM   SBM    **,1
MX    TPL    TIX
MN    TMI    TIX
K1    PZE    1
IDIM  PZE    **          ** = IDIMEN
      END
```

```
*     MDOT  (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0093
*     LABEL
CMDOT
      SUBROUTINE MDOT  (N,L,AA,BB,CC,ORDER)
C
C                      ----ABSTRACT----
C
C TITLE - MDOT
C        DOT PRODUCT OR REVERSED DOT PRODUCT OF VECTORS OF MATRICES
C
C               MDOT FINDS THE DOT PRODUCT
C
C                 C = A(1)*B(1) + ... + A(L)*B(L)
C
C               OR THE REVERSED DOT PRODUCT
C
C                 C = A(1)*B(L) + ... + A(L)*B(1)
C
C               OF TWO VECTORS OF N X N MATRICES A(K) AND B(K); THE
C               MATRICES ARE ASSUMED TO BE STORED BY COLUMNS AND
C               CLOSELY PACKED.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 109 REGISTERS
C SPEED      - ABOUT  (18*L)*N*N*N + (12*L-3)*N*N + (10*L+4)*N + 104*L
C                 + 117  MACHINE CYCLES ON THE 7090.
C AUTHOR     - R.A. WIGGINS   3/63
C
C                      ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MATML1
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL MDOT  (N,L,AA,BB,CC,ORDER)
C
C INPUTS
C
C    N          IS THE DIMENSION OF THE MATRICES IN THE A AND B VECTORS;
C               MUST BE GRTHN= 1
C
C    L          IS THE NUMBER OF MATRICES IN THE A AND B VECTORS;
C               MUST BE GRTHN= 1
C
C    AA(I)      I=1,...,L*N*N CONTAINS THE VECTOR OF MATRICES A(1)
C                 THROUGH A(L) STORED CLOSELY PACKED BY COLUMNS.
C
C    BB(I)      I=1,...,L*N*N CONTAINS THE VECTOR OF MATRICES B(1)
C                 THROUGH B(L) STORED CLOSELY PACKED BY COLUMNS.
C
C    ORDER      IF GRTHN= 0 THE DOT PRODUCT IS FOUND.
C               IF LSTHN  0 THE REVERSE DOT PRODUCT IS FOUND.
C
C OUTPUTS
C
C    CC(I)      I=1,...,N*M CONTAINS THE DOT PRODUCT C AS DESCRIBED
C                 IN THE ABSTRACT, STORED BY COLUMNS.
C
C EXAMPLES
C
C 1. INPUTS  - N=1  L=3  AA(1...3) = 1.,2.,3.  BB(1...3) = 1.,-1.,-4.
C               ORDER=1.
C    OUTPUTS - CC(1) = -13.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ORDER=-1.
C    OUTPUTS - CC(1) = -3.
C
C 3. INPUTS  - N=2  L=2  AA(1...8) = 1.,2.,1.,2.,3.,4.,3.,4.
C               ORDER=1.  BB(1...8) = 2.,2.,2.,2.,4.,4.,4.,4.
C    OUTPUTS - CC(1...4) = 28.,40.,28.,40.
C
C 4. INPUTS  - SAME AS EXAMPLE 3. EXCEPT ORDER=-1.
C    OUTPUTS - CC(1...4) = 20.,32.,20.,32.
C
```

```
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),BB(10),CC(10)
      M=0
      N1=N
      NN=N1*N1
      NN1=NN
      K=1
      J=1
      IF (ORDER) 10,20,20
 10   K=(L-1)*NN+1
      NN1=-NN
 20   DO 100 I=1,L
      CALL MATML1(N1,AA(J),BB(K),CC,M)
      J=J+NN
      K=K+NN1
 100  M=1
      RETURN
      END
```

```
*     MDOT3 (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0119
*     LABEL
CMDOT3
      SUBROUTINE MDOT3(N,M,L,LAB,AA,BB,TRAN,CC,ORDER)
C
C                     ----ABSTRACT----
C
C TITLE - MDOT3
C       DOT PRODUCT OR REVERSED DOT PRODUCT OF VECTORS OF MATRICES
C
C              MDOT FINDS THE DOT PRODUCT
C
C                 C = A(1)*B(1) + ... + A(LAB)*B(LAB)
C
C               OR
C                         ----                  ------   -
C                C = A(1)*B(1) + ... + A(LAB)*B(LAB)  B = B TRANSPOSE
C
C              OR THE REVERSED DOT PRODUCT
C
C                 C = A(I)*B(LAB) + ... + A(LAB)*B(1)
C
C               OR
C                         ------                  ----
C                C = A(1)*B(LAB) + ... + A(LAB)*B(1)
C
C              WHERE A(K) IS A VECTOR OF N X M MATRICES STORED BY
C              COLUMNS AND B(K) IS A VECTOR OF M X N MATRICES STORED
C              BY COLUMNS OR BY ROWS.  BOTH VECTORS ARE CLOSELY PACKED.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094 (MAIN FRAME ONLY)
C STORAGE    - 122 REGISTERS
C SPEED      - ABOUT  (((18*M+98)*N +14)*L + 147)*LAB + 133  MACHINE
C                 CYCLES ON THE 7094 IF THE VERSION OF MATML3 WRITTEN
C                 MARCH, 1963, IS USED.
C AUTHOR     - R.A. WIGGINS    3/63
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MATML3
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MDOT3 (N,M,L,LAB,AA,BB,TRAN,CC,ORDER)
C
C INPUTS
C
C    N          IS THE NUMBER OF ROWS IN THE MATRICES IN A.
C               MUST BE GRTHN=1
C
C    M          IS THE NUMBER OF COLUMNS IN THE MATRICES IN A AND B
C                 (AFTER TRANSPOSITION)
C               MUST BE GRTHN=1
C
C    L          IS THE NUMBER OF ROWS IN THE MATRICES IN B (AFTER
C                 TRANSPOSITION)
C               MUST BE GRTHN=1
C
C    LAB        IS THE NUMBER OF MATRICES IN THE A AND B VECTORS.
C               MUST BE GRTHN=1
C
C    AA(I)      I=1,...,LAB*N*M CONTAINS THE VECTOR OF MATRICES A(1)
C                 THROUGH A(L) STORED CLOSELY PACKED BY COLUMNS.
C
C    BB(I)      I=1,...,LAB*M*L CONTAINS THE VECTOR OF MATRICES B(1)
C                 THROUGH B(L) STORED CLOSELY PACKED.
C
C    TRAN       IF = 0. B IS ASSUMED TO BE STORED BY COLUMNS.
C               IF NOT= 0. B IS ASSUMED TO BE STORED BY ROWS AND THE DOT
C                 PRODUCT OF B TRANSPOSE IS FOUND.
C
C    ORDER      IF GRTHN=0 THE DOT PRODUCT IS FOUND.
C               IF LSTHN 0 THE REVERSE DOT PRODUCT IS FOUND.
C
```

```
C OUTPUTS
C
C    CC(I)      I=1,...,N*L CONTAINS THE DOT PRODUCT C AS DESCRIBED
C                 IN THE ABSTRACT, STORED BY COLUMNS.
C
C EXAMPLES
C
C 1. INPUTS  - N=1 M=1 L=1 LAB=3 AA(1...3)=1.,2.,3. B(1...3) =1.,-1.,-4.
C              ORDER=1.  TRAN=0.
C    OUTPUTS - CC(1) = -13.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ORDER=-1.
C    OUTPUTS - CC(1) = -3.
C
C 3. INPUTS  - LAB=2      AA(1...8) = 1.,2.,1.,2.,3.,4.,3.,4.  N=2  M=2
C              ORDER=1.  BB(1...8) = 2.,2.,3.,2.,4.,1.,4.,4. L=2 TRAN=0.
C    OUTPUTS - CC(1...4) = 19.,28.,29.,42.
C
C 4. INPUTS  - SAME AS EXAMPLE 3. EXCEPT ORDER = -1.
C    OUTPUTS - CC(1...4) = 17.,26.,23.,36.
C
C 5. INPUTS  - SAME AS EXAMPLE 3. EXCEPT TRAN = 1.
C    OUTPUTS - CC(1...4) = 29.,42.,19.,28.
C
C 6. INPUTS  - SAME AS EXAMPLE 3. EXCEPT ORDER = -1.  TRAN = 1.
C    OUTPUTS - CC(1...4) = 23.,36.,17.,26.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),BB(10),CC(10)
      M1=0
      IDA=N*M
      IDB=M*L
      J=1
      K=1
      IF(ORDER)  10,40,40
 10   K=(LAB-1)*IDB+1
      IDB=-IDB
 40   DO 100  I=1,LAB
      CALL MATML3 (N,M,L,AA(J),BB(K),TRAN,CC,M1)
      J=J+IDA
      K=K+IDB
 100  M1=M1+1
      RETURN
      END
```

```
*     MEMUSE (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0068
*     LABEL
CMEMUSE
      SUBROUTINE MEMUSE(ITPOUT)
C
C
C                     ----ABSTRACT----
C
C TITLE - MEMUSE
C        OFF-LINE PRINT OF MEMORY USAGE - PROGRAM AND COMMON
C
C              MEMUSE  PRINTS ONE LINE ON A SPECIFIED TAPE UNIT GIVING
C              TOTAL PROGRAM STORAGE, TOTAL DIMENSIONED COMMON SPACE,
C              AND REMAINING AVAILABLE COMMON SPACE.  IT IS APPLICABLE
C              ONLY UNDER THE FORTRAN MONITOR SYSTEM.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  (MAIN FRAME PLUS 1 TAPE DRIVE)
C STORAGE    - 71 REGISTERS
C SPEED      - TAKES TIME REQUIRED TO OUTPUT ONE 95 CHAR. BCD RECORD
C AUTHOR     - S.M. SIMPSON, JUNE 1964
C
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - XLCOMN
C       AND FORTRAN SYSTEM ROUTINES - (STH), (FIL)
C
C FORTRAN USAGE
C     CALL MEMUSE(ITPOUT)
C
C
C INPUTS
C
C    ITPOUT    IS THE LOGICAL TAPE NUMBER FOR OUTPUT
C
C
C OUTPUTS      A ONE-LINE MESSAGE IS PRINTED AS ILLUSTRATED BELOW
C
C
C EXAMPLES
C
C 1. USAGE    - SUPPOSE THAT THE FOLLOWING MAIN PROGRAM IS OPERATED,
C               AND THAT THE MAIN PROGRAM PLUS  MEMUSE,  EXIT  AND LOWER
C               LEVEL ROUTINES OCCUPY OCTAL LOCATIONS  144  THROUGH 4523.
C                     DIMENSION  C(2000)
C                     COMMON  C
C                     CALL MEMUSE(2)
C                     CALL EXIT
C                     END
C    OUTPUTS - ONE LINE (COLUMNS 2 THRU 95) IS FORMED ON LOGICAL 2 AS
C               FOLLOWS.
C                  MEMORY USAGE (DECIMAL) -  2288 FOR PROGRAM,
C                  2000 FOR DIMENSIONED COMMON,
C                  28174 UNUSED COMMON
C
C
C PROGRAM FOLLOWS BELOW
C
C NOTE - 32,462 DECIMAL = 100,000 - 144 -(77,777-77,461) OCTAL
C
      LDCOM = XLCOMNF(1.0)
      LUCOM = XLCOMNF(0.0) - LDCOM
      LPROG = 32462 - LUCOM - LDCOM
      WRITE OUTPUT TAPE ITPOUT,70,LPROG,LDCOM,LUCOM
 70   FORMAT(26H MEMORY USAGE (DECIMAL) - , I5, 13H FOR PROGRAM,, I6,
    1      24H FOR DIMENSIONED COMMON,, I6, 14H UNUSED COMMON)
 9999 RETURN
      END
```

```
*     MFACT  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0102
*     LABEL
CMFACT
      SUBROUTINE MFACT (N,AA,BB)
C
C                    ----ABSTRACT----
C
C TITLE - MFACT
C        FACTOR A SYMMETRIC POSITIVE DEFINITE MATRIX
C
C              MFACT FINDS A TRIANGULAR MATRIX B(I,J)  (I REFERS TO THE
C              ROW, J REFERS TO THE COLUMN) SUCH THAT
C
C                A = B * (B TRANSPOSE)
C
C              WHERE A IS A SQUARE, POSITIVE DEFINITE MATRIX.
C
C              THE FACTORIZATION IS MADE ACCORDING TO THE SCHEME
C
C                         ( 0                                 I GRTHN J
C                         (
C                         (           N
C                         ( (A(J,J)-SUM (B(I,K)*B(I,K)))**.5      I = J
C                B(I,J)=(          K=J+1
C                         (
C                         (           N
C                         ( (A(I,J)-SUM (B(I,K)*B(K,J)))/B(J,J)  I LSTHN J
C                         (          K=J+1
C
C              WHERE N IS THE NUMBER OF ROWS AND COLUMNS AND THE
C              CALCULATION PROCEEDS AS
C
C                 J=N    I=N,N-1,N-2,...,1
C                 J=N-1  I=N,N-1,N-2,...,1
C                 .
C                 .
C                 J=1    I=N,N-1,N-2,...,1
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT - 709, 7090, 7094 (MAIN FRAME ONLY)
C STORAGE    - 187 REGISTERS
C SPEED      - ABOUT  .0000060*N*N*N + .000102*N*N + .000302*N + .000110
C                SECONDS ON THE 7094 MOD 1 .
C AUTHOR     -  R.A. WIGGINS     3/63
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - DOTJ, STZ
C        AND FORTRAN SYSTEM ROUTINES - SQRT
C
C FORTRAN USAGE
C     CALL MFACT (N,AA,BB)
C
C INPUTS
C
C    N          IS THE NUMBER OF COLUMNS OR ROWS IN THE MATRICES A AND B.
C               MUST BE GRTHN=1
C
C    AA(I)      I=1,...,N*N CONTAINS THE MATRIX A(I,J) STORED CLOSELY
C                 PACKED.
C
C OUTPUTS
C
C    BB(I)      I=1,...,N*N CONTAINS THE MATRIX B(I,J) STORED CLOSELY
C                 PACKED BY COLUMNS.
C
C EXAMPLES
C
C 1. INPUTS   - N=1  AA(1) = 4.
C    OUTPUTS - BB(1)=2.
C
C 2. INPUTS   - N=2  AA(1...4) = 5.,6.,6.,9.
C    OUTPUTS - BB(1...4) = 1.,0.,2.,3.
C
C 3. INPUTS   - N=3
```

```
C              AA(1...9) = 4.69,4.74,2.40,4.74,6.13,3.40,2.40,3.40,4.00
C     OUTPUTS - BB(1...9) = 1.00,0.  ,0.  ,1.50,1.80,0.  ,1.20,1.70,2.00
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),BB(10)
      N1=N
 30   NN=N1*N1
      CALL STZ(NN,BB)
      J=NN
      DO 70  I=1,N1
      I1=I-1
      J=J-I1
      J1=J
      JN=J+N1
      CALL DOTJ (I1,N1,BB(JN),N1,BB(JN),DOT,0,1.)
      BB(J)=SQRTF(AA(J)-DOT)
      J=J-1
      IF(J)  100,100,50
 50   I2=I+1
      DO 60  K=I2,N1
      JN1=J+N1
      CALL DOTJ (I1,N1,BB(JN1),N1,BB(JN),DOT,0,1.)
      BB(J)=(AA(J)-DOT)/BB(J1)
 60   J=J-1
 70   CONTINUE
 100  RETURN
      END
```

```
*     MIFLS (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0166
*     LABEL
CMIFLS
      SUBROUTINE MIFLS (N,LL,BB,RR,GG,FF,C)
C
C                    ----ABSTRACT----
C
C TITLE - MIFLS
C       MULTI-INPUT FILTER BY LEAST SQUARES
C
C             MIFLS INCREASES THE LENGTH OF A MULTI-INPUT LEAST SQUARE
C             FORWARD SHAPER FILTER BY ONE. THAT IS, GIVEN THE VECTOR
C             OF MATRICES F(K,L) (K REFERS TO A PARTICULAR 1 X N MATRIX
C             ELEMENT IN A VECTOR OF L ELEMENTS) THAT SATISFIES THE
C             EQUATIONS
C
C               F(L,L)*R(0)   + ... + F(1,L)*R(L-1)  =  G(L-1)
C
C               F(L,L)*R(-1)  + ... + F(1,L)*R(L-2)  =  G(L-2)
C               .
C               .
C               .
C               F(L,L)*R(-L+1)+ ... + F(1,L)*R(0)    =  G(0)
C
C             AND BET(0,L) AND B(K,L) AS DESCRIBED IN MIPLS
C             THEN MIFLS INCREASES THE LENGTH OF F(K,L) BY ONE SO THAT
C             IT SATISFIES THE EQUATIONS
C
C               F(L+1,L+1)*R(0) + ... + F(1,L+1)*R(L) = G(L)
C               ETC.
C
C             IF R(K) REPRESENTS THE N X N MATRIX VALUED AUTOCORRELATION
C             OF AN N X M MATRIX VALUED WAVELET X(T)
C
C               R(K) = SUM (X(T+K)*X(T)TRANSPOSE)
C
C             AND G(K) REPRESENTS THE 1 X M MATRIX VALUED CROSS-
C             CORRELATION OF A DESIRED OUTPUT D(T) WITH THE WAVELET X(T)
C
C               G(K) = SUM (D(T)*X(T-K)TRANSPOSE)
C
C             THEN THE FIRST SET OF EQUATIONS ABOVE ARE THE NORMAL
C             EQUATIONS FOR A SHAPER FILTER
C
C               D(T) - (F(L,L)*X(T-L) + ... + F(1,L)*X(T-1)) = ZET(T,L)
C
C             WHERE ZET(T,L) IS AN 1 X M MATRIX VALUED ERROR SERIES.
C
C             SEE THE ABSTRACT OF MIPLS FOR THE INTERPRETATION OF
C             B(K,L) AND BET(0,L).
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 276 REGISTERS
C SPEED     - IF  L = 1,  THE TIME IS ABOUT
C                  36*N*N + 244*N + 744  MACHINE CYCLES,
C             OR IF  L GRTHN 1,  THE TIME IS ABOUT
C                  36*N*N*LL + 224*N*LL + 287*LL + 36*N*N - 204*N - 438
C             MACHINE CYCLES ON THE 7090, WHERE  LL = L+1 .
C             (THESE ESTIMATES ARE BASED ON THE VERSION
C             OF  MATML3  WRITTEN MARCH, 1963.)
C AUTHOR    - R.A. WIGGINS
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MATML3,MOVREV
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MIFLS (N,LL,BB,RR,GG,FF,C)
C
C INPUTS
C
C   N         IS THE DIMENSION OF THE MATRICES IN THE F, B, AND R
C               VECTORS.
```

```
C              MUST BE GRTHN=1
C
C    LL        =L+1 IS THE NUMBER OF MATRICES IN THE F VECTOR
C                AFTER THE PROGRAM HAS OPERATED.
C              MUST BE GRTHN=1
C
C    BB(I)     I=1,...,LL*N*N CONTAINS THE VECTOR OF
C                N X N MATRICES B(0,L) THROUGH B(L,L) AS DESCRIBED IN
C                THE ABSTRACT.
C                IF NN = N*N, THEN
C               BB(1...N)            CONTAINS COLUMN 1 OF B(0,L)
C               BB(N+1...2*N)        CONTAINS COLUMN 2 OF B(0,L)
C               .                    .
C               .                    .
C               BB((N+1)*N+1...NN)   CONTAINS COLUMN N OF B(0,L)
C               BB(NN+1...(N+1)*N-1) CONTAINS COLUMN 1 OF B(1,L)
C               ETC.
C
C    RR(I)     I=1,...,LL*N*N CONTAINS THE AUTOCORRELATION VECTOR OF
C                N X N MATRICES R(0) THROUGH R(L) AS DESCRIBED IN THE
C                ABSTRACT STORED SIMILARLY TO BB(I).
C
C    GG(I)     I=1,...,LL*N*N CONTAINS THE CROSSCORRELATION VECTOR OF
C                1 X N MATRICES G(0) THROUGH G(L) AS DESCRIBED IN THE
C                ABSTRACT, STORED SIMILARLY TO BB(I).
C
C    FF(I)     I=1,...,(LL-1)*N IS THE FILTER VECTOR OF 1 X N MATRICES
C                F(1,L) THROUGH F(L,L) AS DESCRIBED IN THE ABSTRACT,
C                STORED SIMILARLY TO BB.
C
C    C(I)      I=1,...,6*N*N IS COMPUTATION SPACE NEEDED BY MIFLS.
C              I=1,...,N*N CONTAINS ALP(0,L) AS DESCRIBED IN MIPLS.
C              I=N*N+1...2*N*N CONTAINS BET(0,L) AS DESCRIBED IN MIPLS.
C              I=2*N*N+1,...,3*N*N CONTAINS ALP(0,L) INVERSE
C              I=3*N*N+1,...,4*N*N CONTAINS BET(0,L) INVERSE
C                (THESE VALUES ARE UNDISTURBED BY MIFLS)
C
C OUTPUTS
C
C    FF(I)     I=1,...,LL*N CONTAINS THE NEW FILTER F(1,L+1) THROUGH
C                F(L+1,L+1)
C
C EXAMPLES
C
C 1. SINGLE-INPUT CASE
C    INPUTS  - N=1   LL=1  BB(1...2) = 1.,-.4  RR(1...3) = 1.25,.5,0.
C              GG(1...3) = 1.,0.,0.   CC(1...4) = 1.25,1.25,.8,.8
C              FF(1) = .8
C    OUTPUTS - FF(1...2) = 0.9524,-0.3810
C
C 2. MULT-INPUT CASE - USE OF MIPLS IN CONJUNCTION WITH MIFLS
C    INPUTS  - N=1   L=4
C              RR(1...20) = 1.89, 0.89, 0.89, 1.05, GG(1...8)=-1.20,-.55
C                           1.20, 0.60, 0.55,-0.18,          -0.50,-.50
C                           0.50, 0.10, 0.50, 0.01,            .00, .00
C                            .00,  .00,  .00,  .00             .00, .00
C    USAGE   -        LL=0
C                     DO 10 I=1,L
C                     CALL MIPLS (N,LL,AA,BB,RR,C,ERR)
C                  10 CALL MIFLS (N,LL,BB,RR,GG,FF,C)
C    OUTPUTS - FF(1...8) =-0.8564,  0.0288
C                         0.2117, -0.2008
C                         0.1531,  0.3455
C                        -0.3259,  0.1564
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION BB(10),RR(10),GG(10),FF(10),C(10)
      NI=N
      LI=LL-1
      NN=NI*NI
      NN2=NN+1
      NN4=NN2+NN+NN
      NN5=NN4+NN
      LN1=LI*NI+1
```

```
      LNN1=(LN1-1)*NI+1
   20 CALL MOVREV(NI,1,GG(LN1),1,C(NN5),-1.)
      IF (LI) 90,60,30
   30 CONTINUE
      J=LN1
      DO 50 I=NN2,LNN1,NN
      J=J-NI
      CALL MATML3 (1,NI,NI,FF(J),RR(I),0.,C(NN5),1)
   50 CONTINUE
   60 CALL MATML3 (1,NI,NI,C(NN5),C(NN4),0.,FF(LN1),0)
      CALL MOVREV(NI,1,FF(LN1),1,C(NN5),-1.)
      J=LN1
      DO 80 I1=1,LNN1,NN
      CALL MATML3 (1,NI,NI,C(NN5),BB(I1),0.,FF(J),I1-1)
   80 J=J-NI
   90 RETURN
      END
```

```
************************
*    MINAB             *
************************
 REFER TO
     MAXSN
```

```
************************
*    MINABM            *
************************
 REFER TO
     MAXSNM
```

```
************************
*    MINSN             *
************************
 REFER TO
     MAXSN
```

```
************************
*    MINSNM            *
************************
 REFER TO
     MAXSNM
```


```
************************
*    MINAB             *
************************
 REFER TO
     MAXSN
```

```
************************
*    MINABM            *
************************
 REFER TO
     MAXSNM
```

```
************************
*    MINSN             *
************************
 REFER TO
     MAXSN
```

```
************************
*    MINSNM            *
************************
 REFER TO
     MAXSNM
```

```
*     MIPLS  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0253
*     LABEL
CMIPLS
      SUBROUTINE MIPLS (N,LL,AA,BB,RR,C,ERR)
C
C                     ----ABSTRACT----
C
C TITLE - MIPLS
C        MULTI-INPUT PREDICTOR BY LEAST SQUARES
C
C              MIPLS INCREASES THE LENGTH OF MULTI-INPUT LEAST SQUARE
C              PREDICTION AND RETROSPECTIVE ERROR OPERATORS BY ONE.
C              THAT IS, GIVEN THE VECTOR OF MATRICES A(K,L) AND B(K,L)
C              (K REFERS TO A PARTICULAR N X N MATRIX ELEMENT IN A
C              VECTOR OF L+1 ELEMENTS) THAT SATISFY THE EQUATIONS
C
C                A(L,L)*R(0)    + ... + A(1,L)*R(L-1) + A(0,L)*R(L)  = 0
C
C                A(L,L)*R(-1)   + ... + A(1,L)*R(L-2) + A(0,L)*R(L-1)= 0
C                .
C                .
C                .
C                A(L,L)*R(-L+1)+ ... + A(1,L)*R(0)    + A(0,L)*R(-1) = 0
C
C              AND
C
C                B(0,L)*R(-1)   + B(1,L)*R(0)    + ... + B(L,L)*R(L-1)= 0
C
C                B(0,L)*R(-2)   + B(1,L)*R(-1)   + ... + B(L,L)*R(L-2)= 0
C                .
C                .
C                .
C                B(0,L)*R(-L)   + B(1,L)*R(-L+1)+ ... + B(L,L)*R(0)  = 0
C
C              WHERE A(0,L) AND B(0,L) ARE CONSTRAINED TO BE IDENTITY
C              MATRICES, THEN MIPLS INCREASES THE LENGTHS OF A AND B
C              BY ONE SO THAT THEY SATISFY THE EQUATIONS
C
C                A(L+1,L+1)*R(0)+...+A(1,L+1)*R(L-1)+A(0,L+1)*R(L)   = 0
C                ETC.
C              AND
C                B(0,L+1)*R(-1)+B(1,L+1)*R(0)+...+B(L+1,L+1)*R(L)    = 0
C                ETC.
C
C              IF R(K) REPRESENTS THE AUTOCORRELATION OF AN N X M MATRIX
C              VALUED TIME SERIES X(T)
C
C                R(K) = EXPECTED VALUE (X(T+K),X(T)TRANSPOSE)
C
C              THEN THE FIRST SET OF EQUATIONS ABOVE ARE THE NORMAL
C              EQUATIONS FOR THE PREDICTION ERROR OPERATOR
C
C                A(L,L)*X(T-L) +...+ A(1,L)*X(T-1)+A(0,L)*X(T) = EPS(T,L)
C
C              AND THE SECOND SET OF EQUATIONS ABOVE ARE THE NORMAL
C              EQUATIONS FOR THE RETROSPECTIVE ERROR OPERATOR
C
C                B(0,L)*X(T)+B(1,L)*X(T+1) +...+ B(L,L)*X(T+L) = ETA(T,L)
C
C              WHERE EPS AND ETA ARE THE N X M MATRIX ERROR SERIES.
C
C              AS A MATTER OF TERMINOLOGY, WE DEFINE
C
C                A(L,L)*R(1) +...+ A(1,L)*R(L)+A(0,L)*R(L+1) = ALP(L+1,L)
C                A(L,L)*R(-L)+...+ A(1,L)*R(-1)+A(0,L)*R(0)  = ALP(0,L)
C              AND
C                B(0,L)*R(0) + B(1,L)*R(1) +...+ B(L,L)*R(L) = BET(0,L)
C                B(0,L)*R(-L-1)+B(1,L)*R(-L)+...+B(L,L)*R(1) =BET(-L-1,L)
C
C              WHERE ALP(0,L) AND BET(0,L) ARE THE COVARIANCE MATRICES
C              OF EPS(T,L) AND ETA(T,L), RESPECTIVELY.  THAT IS
C
C                ALP(0,L) = EXPECTED VALUE (EPS(T,L)*EPS(T,L)TRANSPOSE)
C                BET(0,L) = EXPECTED VALUE (ETA(T,L)*ETA(T,L)TRANSPOSE)
C
```

```
C              MIPLS RETURNS THE VALUES OF ALP(0,L) AND BET(0,L) FOR
C              THE NEW OPERATORS OF LENGTH L+1.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 571 REGISTERS
C SPEED     - IF  LL=0  ABOUT  33*N*N*N + 148*N*N + 163*N + 1287
C                MACHINE CYCLES ON THE 7090.
C              IF  LL=1  ABOUT  156*N*N*N + 769*N*N + 369*N + 2369
C                MACHINE CYCLES ON THE 7090.
C              IF  LL GRTHN 1  ABOUT (36*LL+120)*N*N*N + (216*LL+563)*N*N
C                + (28*LL+368)*N + 396*LL + 1990  MACHINE CYCLES
C                ON THE 7090.
C AUTHOR    - R.A. WIGGINS
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - IXCARG,MATINV,MATML3,MATRA,
C                                      MDOT3,MOVREV,STZ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MIPLS (N,LL,AA,BB,RR,C,ERR)
C
C INPUTS
C
C    N          IS THE DIMENSION OF THE MATRICES IN THE A, B, AND R
C                 VECTORS.
C               MUST BE GRTHN=1
C
C    LL         =L+1 THE NUMBER OF MATRICES IN THE A AND B VECTORS
C                 WHEN THE PROGRAM IS ENTERED (THIS IS ALSO AN OUTPUT).
C               MUST BE GRTHN=0
C
C    AA(I)      I=1,...,LL*N*N CONTAINS THE VECTOR OF MATRICES A(0,L)
C                THROUGH A(L,L) AS DESCRIBED IN THE ABSTRACT.
C                IF NN = N*N THEN
C                 AA(1...N)          CONTAINS COLUMN 1 OF MATRIX A(0,L)
C                 AA(N+1...2N)       CONTAINS COLUMN 2 OF MATRIX A(0,L)
C                 .                  .
C                 .                  .
C                 AA((N-1)*N+1...NN) CONTAINS COLUMN N OF MATRIX A(0,L)
C                 AA(NN+1...NN+N-1)  CONTAINS COLUMN 1 OF MATRIX A(1,L)
C                 ETC.
C
C    BB(I)      I=1,...,LL*N*N CONTAINS THE VECTOR OF MATRICES B(0,L)
C                 THROUGH B(L,L) AS DESCRIBED IN THE ABSTRACT, STORED
C                 SIMILARLY TO AA(I).
C
C    RR(I)      I=1,...,(LL+1)*N*N CONTAINS THE CORRELATION VECTOR OF
C                 MATRICES R(0) THROUGH R(L+1) AS DESCRIBED IN THE
C                 ABSTRACT, STORED SIMILARLY TO AA(I).
C
C    C(I)       I=1,...,5*N*N+N IS COMPUTATION SPACE NEEDED BY MIPLS.
C               I=1,...,N*N       CONTAINS ALP(0,L)
C               I=N*N+1...2*N*N   CONTAINS BET(0,L)
C               I=2*N*N+1...3*N*N CONTAINS ALP(0,L)INVERSE
C               I=3*N*N+1...4*N*N CONTAINS BET(0,L)INVERSE
C
C OUTPUTS
C
C    LL         =L+2 INCREASED ONE FROM ITS INPUT VALUE.
C
C    AA(I)      I=1,...,LL*N*N (NEW LL) CONTAINS A(0,L+1) THROUGH
C                 A(L+1,L+1).
C
C    BB(I)      I=1,...,LL*N*N (NEW LL) CONTAINS B(0,L+1) THROUGH
C                 B(L+1,L+1).
C
C    C(I)       I=1,...,N*N       CONTAINS ALP(0,L+1)
C               I=N*N+1...2*N*N   CONTAINS BET(0,L+1)
C               I=2*N*N+1...3*N*N CONTAINS ALP(0,L+1)INVERSE
C               I=3*N*N+1...4*N*N CONTAINS BET(0,L+1)INVERSE
C
C    ERR        =0. IF SOLUTION WAS SUCCESSFUL.
```

```
C              =1. IF ANY MATRICES THAT WERE INVERTED WERE SINGULAR.
C              =2. IF OVERFLOW OR UNDERFLOW OCCURRED DURING INVERSION.
C              =3. IF THE VALUE OF L IS ILLEGAL
C
C EXAMPLES
C
C 1. SINGLE-INPUT EXAMPLE
C    INPUTS  - N=1  LL=0  RR(1...3) = 1.25,.50,0.
C    OUTPUTS - AA(1)=1.  BB(1)=1.  C(1...4) = 1.25,1.25,.8,.8  ERR=0.
C              LL=1
C
C 2. CONTINUATION OF ITERATION FROM EXAMPLE 1.
C    INPUTS  - SAME AS OUTPUTS FOR EXAMPLE 1.
C    OUTPUTS - AA(1...2) = 1.,-.4  BB(1...2) = 1.,-.4  LL=2  ERR=0.
C              C(1...4) = 1.05,1.05,0.9524,0.9524
C
C 3. MULTI-INPUT EXAMPLE
C    INPUTS  - N=1  L=5  LL=0
C              RR(1...24) = 1.89, 0.89, 0.89, 1.05,
C                           1.20, 0.60, 0.55,-0.18,
C                           0.50, 0.10, 0.50, 0.01,
C                           0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.
C    USAGE   -       DO 10 I=1,L
C                 10 CALL MIPLS (N,LL,AA,BB,RR,C,ERR)
C    OUTPUTS - AA(1...20) =  1.0000, 0.    , 0.    , 1.0000,
C                           -0.8564,-0.5639, 0.0288, 0.1852,
C                            0.2117, 0.6365,-0.2008,-0.4866,
C                            0.1531, 0.2426, 0.3455,-0.6331,
C                           -0.3259, 0.2593, 0.1564,-0.7539
C              BB(1...20) =  1.0000, 0.    , 0.    , 1.0000,
C                           -0.7383,-0.2250,-0.1260, 0.0671,
C                            0.0209,-0.2111, 0.1523,-0.3603,
C                            0.4137, 0.7391,-0.1588,-0.8445,
C                           -0.2272, 0.3056,-0.0745,-0.8026
C              C (1...16) =  0.8837, 0.3902, 0.3902, 0.7371,
C                            0.9542, 0.5186, 0.5186, 0.8050,
C                            1.4768,-0.7816,-0.7816, 1.7703,
C                            1.6128,-1.0390,-1.0390, 1.9116
C              LL = 5        ERR = 0.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),BB(10),RR(10),C(10),CM(10)
      COMMON CM
      NI=N
      LI=LL
      NN=NI*NI
      NN2=NN+1
      CALL IXCARG (C,IC1)
      IC2=IC1+NN
      IC3=IC2+NN
      IC4=IC3+NN
      IC45=IC4+NN
      IC5=IC45+NI
      LNN1=LI*NN+1
      LNN0=LNN1-NN
      CALL IXCARG (AA(LNN1),IAL)
      CALL IXCARG (BB(LNN1),IBL)
      IF (LI)  10,20,40
C ILLEGAL LL.
 10   ERR=3.
 15   RETURN
C SPECIAL CASE LL=0
 20   CALL STZ (NN,AA)
      CALL STZ (NN,BB)
      N1=NI+1
      DO 30  I=1,NN,N1
      AA(I)=1.
 30   BB(I)=1.
      CALL MOVREV (NN,1,RR,1,CM(IC1),1)
      CALL MOVREV (NN,1,RR,1,CM(IC2),1)
      CALL MATINV (NI,CM(IC1),CM(IC3),CM(IC4),ERR)
      CALL MOVREV (NN,1,CM(IC3),1,CM(IC4),1)
      LL=1
      GO TO 15
```

****
*   MIPLS   *
****
(PAGE 4)

```
      C GENERAL CASE. LL GRTHN 1.
      C
      C CONSTRUCT A(L+1,L+1)
       40   CALL MDOT3 (NI,NI,NI,LI,AA,RR(NN2),0.,CM(IC5),-1.)
            CALL MATML3(NI,NI,NI,CM(IC5),CM(IC4),0.,CM(IAL),0)
            CALL MOVREV(NN,1,CM(IAL),1,CM(IAL),-1.)
      C CONSTRUCT B(L+1,L+1)
            CALL MATRA (CM(IC5),N,N,CM(IC4))
            CALL MATML3(NI,NI,NI,CM(IC4),CM(IC3),0.,CM(IBL),0)
            CALL MOVREV(NN,1,CM(IBL),1,CM(IBL),-1.)
            IF (LI-1)  10,75,60
      C FILL IN OTHER TERMS OF A AND B.
       60   CONTINUE
            J=LNNO
            DO 70  I2=NN2,LNNO,NN
            CALL MOVREV (NN,1,AA(J),1,CM(IC3),1.)
            CALL MATML3 (NI,NI,NI,CM(IAL),BB(I2),0.,AA(J),1)
            CALL MATML3 (NI,NI,NI,CM(IBL),CM(IC3),0.,BB(I2),1)
       70   J=J-NN
      C GET NEW ALP AND BET
       75   CONTINUE
            CALL MATML3 (NI,NI,NI,CM(IAL),CM(IC4),0.,CM(IC1),1)
            CALL MATINV (NI,CM(IC1),CM(IC3),CM(IC4),ERR1)
            CALL MATML3 (NI,NI,NI,CM(IBL),CM(IC5),0.,CM(IC2),1)
            CALL MATINV (NI,CM(IC2),CM(IC4),CM(IC45),ERR2)
            ERR=ERR1+ERR2
            LL=LI+1
            GO TO 15
            END
```

```
*     MISS (SUBROUTINE)                   10/5/64   LAST CARD IN DECK IS NO. 0149
*     LABEL
CMISS
      SUBROUTINE MISS (N,L,AA,BB,RR,GG,FF,C)
C
C                    ----ABSTRACT----
C
C TITLE - MISS
C       MULTI-INPUT SIDEWARDS ITERATION
C
C             MISS PERFORMS A SIDEWARDS ITERATION OF A MULTI-INPUT
C             MATRIX VALUED FILTER F(K,L) (K REFERS TO THE K-TH 1 X N
C             MATRIX ELEMENT IN A VECTOR OF L ELEMENTS) TO CORRESPOND
C             TO A SIMILAR ITERATION OF A CROSSCORRELATION VECTOR G(K).
C             THAT IS, GIVEN A VECTOR F(K,L) THAT SATISFIES
C
C               F(L,L)*R(0)    + ... + F(1,L)*R(L-1) = G(L-1)
C
C               F(L,L)*R(-1)   + ... + F(1,L)*R(L-2) = G(L-2)
C               .
C               .
C               .
C               F(L,L)*R(-L+1)+ ... + F(1,L)*R(0)    = G(0)
C
C             AND THE ERROR OPERATORS A(K,L-1) AND B(K,L-1) WITH THEIR
C             RESPECTIVE COVARIANCE MATRICES ALP(0,L-1) AND BET(0,L-1)
C             THAT CORRESPOND TO THE R(T) ABOVE (SEE ABSTRACT OF
C             MIPLS FOR A DESCRIPTION OF THESE QUANTITIES)
C             THEN MISS COMPUTES THE VECTOR OF MATRICES F1(K,L)
C             WHICH SATISFY
C
C               F1(L,L)*R(0)    + ... + F1(1,L)*R(L-1)  = G(L-2)
C               .
C               .
C               .
C               F1(L,L)*R(-L+1)+ ... + F1(1,L)*R(0)     = G(-1)
C
C             SEE THE  ABSTRACT OF MIFLS FOR A DESCRIPTION OF R(T), G(T)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 335 REGISTERS
C SPEED     - ABOUT  (54*L-18)*N*N + (346*L-91)*N + 411*L + 522
C               MACHINE CYCLES ON THE 7090.
C AUTHOR    - R.A. WIGGINS     3/63
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MATML3,MDOT3,MOVREV
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MISS  (N,L,AA,BB,RR,GG,FF,C)
C
C INPUTS
C
C   N          IS THE DIMENSION OF THE MATRICES IN THE A, B, R, G, F,
C                AND F1 VECTORS.
C              MUST BE GRTHN=1
C
C   L          IS THE NUMBER OF MATRICES IN THE A, B, R, F, AND F1
C                VECTORS.
C              MUST BE GRTHN=1
C
C   AA(I)      I=1,...,L*N*N CONTAINS THE N X N MATRIX VALUED PREDICTION
C                ERROR OPERATOR A(0,L-1) THROUGH A(L-1,L-1) AS COMPUTED
C                BY MIPLS.
C
C   BB(I)      I=1,...,L*N*N CONTAINS THE N X N MATRIX VALUED
C                RETROSPECTIVE ERROR OPERATOR B(0,L-1) THROUGH
C                B(L-1,L-1) AS COMPUTED BY MIPLS.
C
C   RR(I)      I=1,...,L*N*N CONTAINS THE N X N MATRIX VALUED
C                AUTOCORRELATION VECTOR R(0) THROUGH R(L) STORED CLOSELY
```

```
C                 PACKED BY COLUMNS.
C
C    GG(I)      I=1,...,(L+1)*N CONTAINS THE 1 X N MATRIX VALUED
C                 CROSSCORRELATION VECTOR G(-1) THROUGH G(L-1) AS
C                 DESCRIBED IN THE ABSTRACT.
C
C    FF(I)      I=1,...,L*N CONTAINS THE 1 X N MATRIX VALUED SHAPER
C                 FILTER F(1,L) THROUGH F(L,L) AS COMPUTED BY MIFLS.
C
C    C(I)       I=1,...,6*N*N IS COMPUTATION SPACE NEEDED BY MISS.
C               I=1,...,N*N CONTAINS THE COVARIANCE MATRIX ALP(0,L)
C                 AS COMPUTED BY MIPLS.
C               I=N*N+1,...,2*N*N CONTAINS THE COVARIANCE MATRIX BET(0,L)
C                 AS COMPUTED BY MIPLS.
C
C OUTPUTS
C
C    FF(I)      I=1,...,L*N*N CONTAINS THE 1 X N MATRIX VALUED SHAPER
C                 FILTER F1(1,L) THROUGH F1(L,L) AS DESCRIBED IN THE
C                 ABSTRACT.
C
C EXAMPLES
C
C 1. INPUTS   - L=2  N=1  RR(1...2) = 1.25,.5  GG(1...2) = 0.,1.
C               IG1=2  IG2=1
C    USAGE    -       LL=0
C                     DO 10 I=1,L
C                     CALL MIPLS (N,LL,AA,BB,RR,C,ERR)
C                10   CALL MIFLS (N,LL,BB,RR,GG(IG1),FF,C)
C                     CALL MISS  (N,L,AA,BB,RR,GG(IG2),FF,C)
C    OUTPUTS - ERR=0.
C               FF(1...2) = -0.3810,0.9524
C
C 2. INPUTS   - L=2  N=2  RR(1...8) = 1.25,0.,0.,1.16,0.5,0.,0.,0.4
C               GG(1...4) = 1.,0.,0.,0.   IG1=3  IG2=1
C    USAGE    - SAME AS EXAMPLE 1.
C    OUTPUTS - ERR=0.
C               FF(1...4) = 0.9524,0.0,-0.3810,0.0
C
C 3. INPUTS   - L=2  N=2  RR(1...8) = 1.89,0.89,0.89,1.05,
C                                      1.20,0.60,0.55,-0.18
C               GG(1...4) = -1.2,-0.55,-0.5,-0.5  IG1=3  IG2=1
C    USAGE    - SAME AS EXAMPLE 1.
C    OUTPUTS - ERR=0.
C               FF(1...4) = -0.5050,-0.6413,0.6842,-0.9015
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(2),BB(2),RR(2),GG(2),FF(2),C(2)
      NI=N
      LI=L
      NN=NI*NI
      NN2=NN+1
      NN3=NN2+NN
      NN5=NN3+NN+NN
      NN6=NN5+NI
      LAB=LI*NN
      IF1=LI*NI-NI+1
      CALL MOVREV(NI,1,FF(IF1),1,FF(IF1),-1.)
      J=IF1-NI
      DO 10 I=NN2,LAB,NN
      CALL MATML3(1,NI,NI,FF(IF1),BB(I),0.,FF(J),1)
   10 J=J-NI
      CALL MDOT3 (1,NI,NI,LI-1,FF,RR(NN2),-1.,C(NN5),1.)
      CALL MOVREV(IF1-1,1,FF,1,FF(NI+1),1.)
      J=NN5
      DO 20 I=1,NI
      C(J)=GG(I)-C(J)
   20 J=J+1
      CALL MATML3(1,NI,NI,C(NN5),C(NN3),0.,C(NN6),0)
      J=1
      DO 30 I=1,LAB,NN
      CALL MATML3(1,NI,NI,C(NN6),AA(I),0.,FF(J),I-1)
   30 J=J+NI
      RETURN
      END
```

```
*       MLI2A6 (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0217
*       FAP
*MLI2A6
        COUNT   200
        LBL     MLI2A6
        ENTRY   MLI2A6 (MLI,MLIHOL,NCRS)
*
*                    ----ABSTRACT----
*
* TITLE - MLI2A6
*         CONVERT MACHINE LANGUAGE INTEGER TO EQUIVALENT HOLLERITH
*
*               MLI2A6 CONVERTS A MACHINE LANGUAGE INTEGER (CONSIDERED
*               DECIMAL) INTO A 2-REGISTER FORTRAN VECTOR OF EQUIVALENT
*               HOLLERITH (FORMAT(2A6)) WITH LEADING ZEROES SUPPRESSED,
*               PLUS SIGN SUPPRESSED, SIGNIFICANT DIGITS RIGHT ADJUSTED,
*               AND MINUS SIGN (IF PRESENT) RIGHT ADJUSTED AGAINST MOST
*               SIGNIFICANT DIGIT.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 128 REGISTERS
* SPEED     -
* AUTHOR    - S.M. SIMPSON JR., JUNE 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*         AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*       CALL MLI2A6(MLI,MLIHOL,NCRS)
*
* INPUTS
*
*     MLI       IS THE MACHINE LANGUAGE INTEGER.
*
* OUTPUTS
*
*     MLIHOL    IS THE HOLLERITH EQUIVALENT OF MLI IN 2A6 FORMAT.
*
*     NCRS      IS THE NO. OF NON-BLANK CHARACTERS INVOLVED (INCLUDING
*                 THE MINUS SIGN IF PRESENT).
*
* EXAMPLES
*
* 1. INPUTS  - MLI = OCT 173 (=DECIMAL 123)
*    OUTPUTS - MLIHOL(1,2) = OCT 606060606060,606060010203  NCRS=3
*
* 2. INPUTS  - MLI = OCT 400000000173 (=DECIMAL -123)
*    OUTPUTS - MLIHOL(1,2)= OCT 606060606060,606040010203  NCRS=4
*
* 3. INPUTS  - MLI = OCT 144 (= DECIMAL 100)
*    OUTPUTS - MLIHOL(1,2)= OCT 606060606060,606060010000  NCRS=3
*
* 4. INPUTS  - MLI = OCT 0 (= +0)
*    OUTPUTS - MLIHOL(1,2) = OCT 606060606060,606060606000  NCRS=1
*
* 5. INPUTS  - MLI = OCT 400000000000 (= -0)
*    OUTPUTS - MLIHOL(1,2) = OCT 606060606060,606060604000  NCRS=2
*
* 6. INPUTS  - MLI = OCT 400000000144 (=DECIMAL -100)
*    OUTPUTS - MLIHOL(1,2) = OCT 606060606060,606040010000  NCRS=4
*
* 7. INPUTS  - MLI = OCT 777777777777 (=DECIMAL -34359738367)
*    OUTPUTS - MLIHOL(1,2) = OCT 400304030511,070310030607  NCRS=12
*
        HTR     0
        BCI     1,MLI2A6
MLI2A6  SXA     EXIT,1
        SXA     EXIT+1,2
        SXA     EXIT+2,4
        SXD     MLI2A6-2,4
        CLA     1,4
        STA     GET1
```

```
************************
*    MLI2A6             *
************************
(PAGE  2)
```


```
        CLA     2,4
        STA     PUT2A
        SUB     K1
        STA     PUT2B
        CLA     3,4
        STA     PUT3
* STORE MAGN(MLI) AND SET FOR SIGN.
  GET1 CLA      **             A(MLI)
        TMI     NEG
        STO     MLI
        STZ     NCRS
        CLA     SPACE
 STOSN STO      SIGN
        TRA     CNVRT
   NEG SSP
        STO     MLI
        CLA     K1
        STO     NCRS
        CLA     MINUS
        TRA     STOSN
* NOW CONVERT THE MAGNITUDE INTO TWELVE REGISTERS.
* XR1 IS THE DIGIT INDEX =11,10,...,1
* XR2 IS THE DIGIT 0,1,...9
* XR4 IS A LEADING ZERO SUPPRESS INDICATOR
* XR4 = 1 MEANS CONVERT DIGIT=0 TO SPACE
*     = 0 MEANS DONT SUPPRESS A ZERO DIGIT.
 CNVRT LXA      K1,4
        CLA     SPACE
        STA     HOLV1-5
        CLA     MLI
        STO     TEMP
        LXA     K11,1
 GTEMP CLA      TEMP
        LXA     K0,2
* (NOTE - ZEROES ARE PLUS ZEROES IN SUBTRACTIONS BELOW)
   SUB SUB      POWRS+1,1
        TMI     ADD
        TXI     SUB,2,1
* (NOTE - ZEROES WILL BE MINUS ZERO IN ADDITION BELOW, BUT NO HARM.)
   ADD ADD      POWRS+1,1
        STO     TEMP
* AT THIS POINT XR2 CONTAINS DESIRED DIGIT.
* STORE DIGIT IF (SUPPRESS IS OFF) OR (SUPPRESS IS ON AND THIS IS DIGIT
* 1).
        TXL     PXA,4,0
        TXL     STNDGS,1,1
* OTHERWISE SUPPRESS DIGIT IF ZERO
        PXA     0,2
        TZE     GTSPA
* IF GETS HERE, THIS IS FIRST DIGIT.  TURN OFF SUPPRESS, SET NDIGS,
* SET NDIGS, SET SIGN CHARACTER, AND STORE DIGIT.
        LXA     K0,4
STNDGS SXA      NDIGS,1
        CLA     SIGN
        STA     HOLV2,1
        TRA     PXA
* THIS SETS SPACE IN PLACE OF LEADING ZERO.
 GTSPA CLA      SPACE
        TRA     STA
   PXA PXA      0,2
   STA STA      HOLV2+1,1
        TIX     GTEMP,1,1
* STORE NCRS.
        CLA     NDIGS
        ADD     NCRS
        STO     NCRS
* NOW FORM MLIH1 AND MLIH2 BY
* PACKING UP HOLV1 AND HOLV2 RESPECTIVELY.
        STZ     MLIH1
        STZ     MLIH2
        CLA     K30
        STO     LSHIFT
        LXA     K6,1
STSHFT CLA      LSHIFT
        STA     LGL1
```

```
         STA    LGL2
         CAL    HOLV1+1,1
   LGL1  LGL    **             30,24,....,0
         ORA    MLIH1
         SLW    MLIH1
         CAL    HOLV2+1,1
   LGL2  LGL    **             30,24,.......,0
         ORA    MLIH2
         SLW    MLIH2
         CLA    LSHIFT
         SUB    K6
         STO    LSHIFT
         TIX    STSHFT,1,1
         TRA    LEAVE
* LEAVE, STORING RESULTS
  LEAVE  CLA    NCRS
         ALS    18
   PUT3  STO    **             A(NCRS)
         CLA    MLIH1
  PUT2A  STO    **             A(MLIHOL(1))
         CLA    MLIH2
  PUT2B  STO    **             A(MLIHOL(2))
   EXIT  AXT    **,1
         AXT    **,2
         AXT    **,4
         TRA    4,4
* CONSTANTS
         DEC    10000000000
         DEC    1000000000
         DEC    100000000
         DEC    10000000
         DEC    1000000
         DEC    100000
         DEC    10000
         DEC    1000
         DEC    100
         DEC    10
  POWRS  DEC    1
  SPACE  OCT    60
  MINUS  OCT    40
     K0  PZE    0
     K1  PZE    1
     K6  PZE    6
    K10  PZE    10
    K11  PZE    11
    K30  PZE    30
* TEMPORARIES
    MLI  PZE    **             MAGNITUDE OF MLI
   SIGN  PZE    **             = OCT 60 OR OCT 40
         PZE    **             SPACE OR MINUS
         PZE    **             0,1,2,...,9 OR SPACE OR MINUS
         PZE    **             ETC
         PZE    **             ETC
         PZE    **             ETC
  HOLV1  PZE    **             ETC
         PZE    **             ETC
         PZE    **             ETC
         PZE    **             ETC
         PZE    **             ETC
         PZE    **             ETC
  HOLV2  PZE    **             0,1,2,....,9 (NOT SPACE OR MINUS)
  MLIH1  PZE    **             COMPLETED HOL FOR MLIHOL(1)
  MLIH2  PZE    **             COMPLETED HOL FOR MLIHOL(2)
  NDIGS  PZE    **             1,2,....,11
   NCRS  PZE    **
 LSHIFT  PZE    **             30,24,....,0
   TEMP  PZE    **
         END
```

```
*       MLISCL (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0114
*       FAP
*MLISCL
        COUNT   100
        LBL     MLISCL
        ENTRY   MLISCL (MLIV,LMLIV,ISCALE,MLIVSC,IANS)
*
*                      ----ABSTRACT----
*
* TITLE - MLISCL
*       MULTIPLY AN MLI VECTOR BY A FORTRAN FIXED POINT INTEGER
*
*              MLISCL MULTIPLIES EACH ELEMENT OF AN MLI VECTOR BY A
*              GIVEN FORTRAN FIXED POINT INTEGER, CHECKING FOR OVERFLOW.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 47 REGISTERS
* SPEED     - LENGTH OF VECTOR TIMES 25 MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON JR, JUNE 1962
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MLISCL(MLIV,LMLIV,ISCALE,MLIVSC,IANS)
*
* INPUTS
*
*    MLIV(I)   I=1...LMLIV IS THE MLI VECTOR
*
*    LMLIV     MUST EXCEED 0
*
*    ISCALE    IS THE FORTRAN FIXED POINT MULTIPLIER.
*
* OUTPUTS
*
*    MLIVSC(I) I=1...LMLIV =ISCALE*MLIV(1...LMLIV) AS A MLI VECTOR.
*                   (NOTE MLIVSC MAY BE EQUIVALENT TO MLIV).
*
*    IANS      = 0 MEANS JOB DONE OK
*              =-1 MEANS ILLEGAL LMLIV
*              =-2 MEANS OVERFLOW OCCURRED.
*
* EXAMPLES
*
* 1. INPUTS  - MLIV= OCT 1,2,3  LMLIV=3  ISCALE=4
*    OUTPUTS - IANS=0  MLIVSC= OCT 4,10,14
*
* 2. INPUTS  - SAME AS EXAMPLE 1 EXCEPT LMLIV=-2
*    OUTPUTS - IANS=-1
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT MLIV(1)= OCT 377777777777
*    OUTPUTS - IANS=-2
*
* 4. INPUTS  - MLIV(1)= OCT2  LMLIV=1  ISCALE=5
*    OUTPUTS - MLIVSC(1) = OCT12  IANS=0
*
        HTR     0
        BCI     1,MLISCL
MLISCL  SXA     EXIT,1
        SXD     MLISCL-2,4
        CLA     1,4             A(A(MLIV))
        ADD     K1
        STA     LDQ
        CLA     2,4             A(A(LMLIV))
        STA     GET2
        CLA     3,4             A(A(ISCALE))
        STA     GET3
        CLA     4,4             A(A(MLIVSC))
        ADD     K1
        STA     STQ
        CLA     5,4             A(A(IANS))
```

```
        STA     PUT5
* GET LMLIV,ISCALE, AND CHECK LMLIV.
        CLS     K1
        STO     IANS
  GET2  CLA     **            A(LMLIV)
        ARS     18
        STO     LMLIV
        TMI     LEAVE
        TZE     LEAVE
  GET3  CLA     **            A(ISCALE)
        ARS     18
        STO     ISCALE
* SET IANS FOR POSSIBLE OVERFLOW DURING LOOP.
        CLS     K2
        STO     IANS
* LOOP, CHECKING FOR OVERFLOW.
        LXA     LMLIV,1
   LDQ  LDQ     **,1          A(MLIV)+1
        MPY     ISCALE
   STQ  STQ     **,1          A(MLIVSC)+1
        TNZ     LEAVE
        TIX     LDQ,1,1
* ALL OK IF FALLS THRU LOOP.
        CLA     K0
        STO     IANS
* SET IANS AND EXIT.
 LEAVE  CLA     IANS
        ALS     18
  PUT5  STO     **            A(IANS)
  EXIT  AXT     **,1
        TRA     6,4
* CONSTANTS
    K0  PZE     0
    K1  PZE     1
    K2  PZE     2
* VARIABLES
 LMLIV  PZE     **
ISCALE  PZE     **
  IANS  PZE     **            -1,-2,0
        END
```

```
*      MONOCK (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0164
*      FAP
*MONOCK
       COUNT    100
       LBL      MONOCK
       ENTRY    MONOCK (X, LX, ZFNDCR, IANSNG, IANS)
*
*
*                    ----ABSTRACT----
*
* TITLE - MONOCK
*        CHECK VECTOR FOR MONOTONE INCREASING OR DECREASING BEHAVIOR
*
*               MONOCK  CHECKS EITHER THAT A GIVEN VECTOR IS MONOTONE
*               INCREASING (NON-DECREASING) OR THAT IT IS MONOTONE
*               DECREASING (NON-INCREASING).  MINIMUM VECTOR LENGTH IS
*               2 .  VECTOR MAY BE FIXED OR FLOATING POINT, AND  MONOCK
*               FORCES THE SIGN BIT POSITIVE FOR ALL VECTOR ELEMENTS OF
*               ZERO MAGNITUDE.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 48 REGISTERS
* SPEED      - (709,7090) 50+8*LX MACHINE CYCLES,  LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON,  JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL MONOCK(X, LX, ZFNDCR, IANSNG, IANS)
*
*
* INPUTS
*
*    X(I)       I = 1...LX  IS THE VECTOR, MAY BE FIXED OR FLOATING.
*
*    LX         SHOULD EXCEED  1 .
*
*    ZFNDCR     = 0     REQUESTS A CHECK FOR NON-DECREASING BEHAVIOUR.
*               NOT= 0  REQUESTS A CHECK FOR NON-INCREASING BEHAVIOUR.
*
*    IANSNG     IS THE DESIRED  IANS  OUTPUT IF THE CHECK FAILS.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LX  LSTHN=  0 .
*
*    X(I)       OUTPUT WILL EQUAL  X(I)  INPUT EXCEPT THAT THE SIGN BIT
*                  OF ALL ZERO MAGNITUDE VALUES, IF ANY, WILL BE POSITIVE
*                  ON OUTPUT.  THIS WILL BE TRUE REGARDLESS OF THE
*                  OUTCOME OF THE TEST.
*
*    IANS       = 0        IF  LX = 1,  OR IF  X(I)  PASSES THE TEST.
*               = IANSNG  IF  X(I)  FAILS THE TEST.
*
*
* EXAMPLES
*
* 1. INPUTS   - IX1(1...5) = 1,2,2,3,4   IX2(1...5) = 1,2,1,3,4
*               IANS7 = 43   X1(1...5) = 4.,3.,2.,2.,1.
*               X2(1...5) = 4.,3.,1.,2.,1.
*    USAGES  -        CALL MONOCK(IX1,5,0.,-1,IANS1)
*                     CALL MONOCK(IX2,5,0.,-2,IANS2)
*                     CALL MONOCK( X1,5,1.,-3,IANS3)
*                     CALL MONOCK( X2,5,1.,-4,IANS4)
*                     CALL MONOCK(IX2,2,0.,-5,IANS5)
*                     CALL MONOCK(IX2,1,0.,-6,IANS6)
*                     CALL MONOCK(IX2,0,0.,-7,IANS7)
*    OUTPUTS - IANS1,IANS2,...,IANS7 = 0,-2,0,-4,0,0,43
*
* 2. INPUTS   - IX3(1...3) = IX4(1...3) = +0,-0,+0
*               IX5(1...3) = IX6(1...3) = -0,+0,-0
```

```
*              IX7(1...3) = -0,+0,-1     IX8(1...3) = +0,-0,+1
*              IX9 = IX10 = -0
*   USAGES  -         CALL MONOCK(IX3, 3,0., -8,IANS8)
*                     CALL MONOCK(IX4, 3,1., -9,IANS9)
*                     CALL MONOCK(IX5, 3,0.,-10,IANS10)
*                     CALL MONOCK(IX6, 3,1.,-11,IANS11)
*                     CALL MONOCK(IX7, 3,0.,-12,IANS12)
*                     CALL MONOCK(IX8, 3,1.,-13,IANS13)
*                     CALL MONOCK(IX9, 1,0.,-14,IANS14)
*                     CALL MONOCK(IX10,1,1.,-15,IANS15)
*   OUTPUTS - IANS8...IANS15 = 0,0,0,0,-12,-13,0,0
*              IX3(1...3) = IX4(1...3) = IX5(1...3) = IX6(1...3)
*                     = +0,+0,+0
*              IX7(1...3) = +0,+0,-1     IX8(1...3) = +0,+0,+1
*              IX9 = IX10 = +0
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR     0                XR4
       BCI     1,MONOCK
*
* ONLY ENTRY.  MONOCK(X, LX, ZFNDCR, IANSNG, IANS)
*
MONOCK SXD     MONOCK-2,4
       CLA     1,4              A(X)
       STA     CAS1
       STA     CAS2
       ADD     K1               A(X)+1
       STA     CLA1
       STA     CLA2
       STA     CLA3
       CLA*    4,4              IANSNG
       STO     IANSNG           BROUGHT IN
       LDQ*    3,4              ZFNDCR  IN MQ FOR A WHILE
*
* CHECK OUT  LX  AND, IF OK, SET IT IN XR4.  THEN SET  X(LX)  =  +0  IF
* IT IS ZERO MAGNITUDE.  EXIT IF  LX  =  1,  OTHERWISE BRANCH TO LOOP.
*
       CLA*    2,4              LX
       TMI     TRA              (NO ACTION
       TZE     TRA               EXITS)
       PDX     0,4
 CLA1  CLA     **,4             ** = A(X)+1,    GIVES  X(LX)
       TNZ     *+2
       STZ*    CLA1
       PXD     0,0
       XCA                      ZFNDCR  IN AC,  +0  IN MQ
       TXI     *+1,4,-1         LX-1  IN XR4
       TXL     LEAVE,4,0
       TNZ     CLA3
*
* INCREASING CASE.  COMPARE  X(J)  IN AC AGAINST  X(J+1)  J = LX-1,...,1
*              (AND FORCE ZEROES POSITIVE)
*
 CLA2  CLA     **,4             ** = A(X)+1
       TNZ     CAS1
       SSP
       STO*    CLA2
 CAS1  CAS     **,4             ** = A(X)
       LDQ     IANSNG           NG
       TIX     CLA2,4,1         OK
       TIX     CLA2,4,1         OK
       TRA     LEAVE
*
* DECREASING CASE.  SAME COMPARISON SEQUENCE.
*
 CLA3  CLA     **,4             ** = A(X)+1
       TNZ     CAS2
       SSP
       STO*    CLA3
 CAS2  CAS     **,4             ** = A(X)
       TIX     CLA3,4,1         OK
```

```
       TRA    TIX          OK
       LDQ    IANSNG       NG
 TIX   TIX    CLA3,4,1
*
* EXIT SETTING  IANS  FROM MQ.
*
 LEAVE LXD    MONOCK-2,4
       STQ*   5,4
 TRA   TRA    6,4
*
* CONSTANTS, TEMPORARIES
*
 K1    PZE    1
IANSNG PZE    **,**,**
       END
```

```
*     MOUT (SUBROUTINE)                  9/8/64   LAST CARD IN DECK IS NO. 0100
*     LABEL
CMOUT
      SUBROUTINE MOUT (ITAPE,NSPACE,X,XNAME,NRX,NCX,LX)
C
C                    ----ABSTRACT----
C
C TITLE - MOUT
C        MATRIX OUTPUT IN G FORMAT
C
C              MOUT WRITES A VECTOR OF MATRICES ON AN OUTPUT TAPE.  THE
C              MATRICES ARE ASSUMED TO BE STORED TIGHTLY PACKED BY
C              COLUMNS.  THE MATRIX IS HEADED BY A LINE SUCH AS
C
C                   MATRIX  ( 1...    3,  1...    5,  1...    2 ) =
C
C              AND THEN EACH SUCCEEDING ROW IS PRINTED IN FORMAT(5G15.7)
C              WITH DOUBLE SPACES BETWEEN ROWS AND TRIPLE SPACES BETWEEN
C              MATRICES.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME AND TAPE UNIT)
C STORAGE    - 130  REGISTERS
C SPEED      -
C AUTHOR     - R.A. WIGGINS   3/64
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CARIGE
C        AND FORTRAN SYSTEM ROUTINES - (FIL),(STH)
C
C FORTRAN USAGE
C     CALL MOUT  (ITAPE,NSPACE,X,XNAME,NRX,NCX,LX)
C
C
C INPUTS
C
C    ITAPE     LOGICAL TAPE NUMBER FOR OUTPUT
C
C    NSPACE    NUMBER OF SPACES FOR CARRIAGE TO BE MOVED BEFORE PRINTING
C              IF LSTHN 0, THE PAGE IS RESTORED.
C
C    X(I)      I=1...NRX,1...NCX,1...LX IS THE VECTOR OF MATRICES TO BE
C                WRITTEN.  THE COLUMNS, AND MATRICES, MUST BE CLOSELY
C                SPACED.
C              NEED NOT BE FLOATING POINT.
C
C    XNAME     IS A 6, OR FEWER, CHARACTER HOLLERITH NAME FOR THE ARRAY.
C
C    NRX       NUMBER OF ROWS IN EACH X MATRIX.
C              MUST BE GRTHN= 1
C
C    NCX       NUMBER OF COLUMNS IN EACH X MATRIX.
C              MUST BE GRTHN= 1
C
C    LX        NUMBER OF MATRICES IN X.
C              MUST BE GRTHN= 1
C
C    NOTE - THE LEGALITY OF ITAPE, NRX, NCX, AND LX IS NOT CHECKED.
C
C
C OUTPUTS - ARRAY IS WRITTEN ON TAPE ITAPE.
C
C
C EXAMPLES
C
C 1. INPUTS  - ITAPE = 2     NSPACE = 1     XNAME = 1HX
C              X(1...2,1...2,1...2) = 1.,2.,3.,4.,6,7,8,9
C              NRX = 2     NCX = 2     LX = 2
C    OUTPUTS - THE FOLLOWING LINES ARE WRITTEN ON LOGICAL TAPE 2 .
C                 X      ( 1...    2,  1...    2,  1...    2 ) =
C
C                    1.0000000      3.0000000
C
```

```
      C                    2.0000000       4.0000000
      C
      C
      C                    6               8
      C
      C                    7               9
      C
      C PROGRAM FOLLOWS BELOW
      C
            DIMENSION X(2)
            CALL CARIGE (ITAPE,NSPACE)
            WRITE OUTPUT TAPE ITAPE, 10,XNAME,NRX,NCX,LX
         10 FORMAT(3XA6,7H ( 1...I5,7H,  1...I5,7H,  1...I5,5H ) = )
            LXM=NRX*NCX
            LXT=LXM*LX
            DO 40  I1=1,LXT,LXM
            J1=I1+NRX-1
            J2=I1+LXM-1
            DO 30  I2=I1,J1
            WRITE OUTPUT TAPE ITAPE, 20, (X(I),I=I2,J2,NRX)
         20 FORMAT(/(5X5G15.7))
         30 CONTINUE
            CALL CARIGE (ITAPE,1)
         40 CONTINUE
            RETURN
            END
```

```
*     MOUTAI (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0294
*     LABEL
CMOUTAI
      SUBROUTINE MOUTAI(ITAPE,NSPACE,FOFIJ,FNAME,LI,LJ,IDIMEN,
     1                  NDIGS,SCALE,SPACE)
C
C
C                    ----ABSTRACT----
C
C TITLE - MOUTAI
C        OUTPUT A MATRIX AS INTEGERS DENSELY PACKED OFF-LINE
C
C              MOUTAI IS DESIGNED FOR TIGHTLY PACKED PRINTED OUTPUT
C              OF RELATIVELY LOW ACCURACY MATRIX DATA.  MOUTAI FINDS OR
C              IS GIVEN A SUITABLE SCALE FACTOR TO CONVERT A FLOATING
C              POINT MATRIX TO FIXED POINT DATA WITH A SPECIFIED MAXIMUM
C              NO. (1 TO 5) OF DIGITS.  THESE ARE THEN PRINTED IN FORMAT
C              OF 60I2, OR 40I3, OR 30I4, OR 25I5, OR 20I6 RESPECTIVELY.
C
C              VARIOUS SCALING OPTIONS EXIST, BUT THE ORIGINAL MATRIX
C              ALWAYS REMAINS UNDISTURBED SINCE THE SCALING IS DONE ROW
C              BY ROW INTO A SCRATCH VECTOR PROVIDED BY THE USER.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS 1 TAPE DRIVE)
C STORAGE    - 357 REGISTERS
C SPEED      - A MATRIX  F  OF DIMENSION  F(50,20)  TAKES ABOUT
C              .48  SECONDS (ON THE 7094) IF  MOUTAI  DOES THE SCALING,
C              ABOUT  .42  SECONDS IF  F  IS ALREADY FIXED POINT.
C AUTHOR     - S.M. SIMPSON, MARCH 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CARIGE,GNHOL2,MAXABM,RND,MOVE,
C                                     MULPLY,FIXVR,SAME
C       AND FORTRAN SYSTEM ROUTINES - EXP(2,(FIL),LOG,(STH)
C
C FORTRAN USAGE
C     CALL MOUTAI (ITAPE, NSPACE, FOFIJ, FNAME, LI, LJ, IDIMEN,
C    1           NDIGS,SCALE,SPACE)
C
C
C INPUTS
C
C    ITAPE     IS LOGICAL TAPE NO. FOR OUTPUT
C              SHOULD LIE BETWEEN 1 AND 20 (NOT CHECKED)
C
C    NSPACE    IS DESIRED NO. OF INITIAL BLANK LINES BEFORE OUTPUT
C                BEGINS.  MAY BE ZERO.  IF LESS THAN ZERO A
C                PAGE RESTORE IS CREATED.
C
C    FOFIJ(I,J)  I=1...LI,  J=1...LJ  IS THE LI BY LJ MATRIX TO
C                BE PRINTED.  FOFIJ IS FLOATING POINT EXCEPT IN THE
C                CASE SCALE = 0.0 AS DESCRIBED BELOW.
C
C    FNAME     IS 6 HOLLERITH (FORMAT,(1A6)) TO BE USED AS A LABEL FOR
C                FOFIJ.
C
C    LI        SHOULD EXCEED ZERO (NOT CHECKED)
C
C    LJ        SHOULD EXCEED ZERO (NOT CHECKED)
C
C    IDIMEN    IS THE VALUE TO WHICH THE INDEX I IN FOFIJ (I,J) IS
C                DIMENSIONED IN THE CALLING PROGRAM
C
C    NDIGS     SPECIFIES THE MAXIMUM NUMBER OF DIGITS (EXCLUSIVE OF
C                SIGN) WHICH MAY BE USED TO EXPRESS THE SCALED VALUES OF
C                THE MATRIX. NDIGS MAY ONLY HAVE VALUES = 1,2,3,4, OR 5.
C                THE FIELD WIDTH IN PRINTING WILL BE ONE GREATER THAN
C                NDIGS.
C
C    SCALE     SPECIALIZES THE TYPE OF SCALING TO BE PERFORMED.
C                FOR SCALE=0.0, IT IS ASSUMED THAT FOFIJ IS ALREADY
C                IN INTEGER FORM (COMPATIBLE WITH NDIGS).  IN THIS CASE
```

```
C                MOUTAI DOES NO SCALING.
C
C                IF SCALE EXCEEDS 0.0, IT IS ASSUMED THAT SCALE IS A
C                SCALE FACTOR THE CALLING PROGRAM WISHES MOUTAI TO USE
C                AS FOLLOWS,
C                  OUTPUT INTEGER = (SCALE * FOFIJ) ROUNDED TO INTEGER.
C
C                IF SCALE=-1., MOUTAI FINDS ITS OWN SCALE FACTOR, THE
C                FACTOR RESTRICTED TO BEING A POWER OF 10, SUCH THAT
C                THE LARGEST OUTPUT MAGNITUDE, CALL IT MAXMAG, WILL
C                SATISFY  10EXP(NDIGS-1) LSTHN= MAXMAG LSTHN
C                10EXP(NDIGS).
C
C                IF SCALE=-2., MOUTAI FINDS ITS OWN SCALE SUCH THAT THE
C                LARGEST OUTPUT MAGNITUDE, MAXMAG, WILL =10EXP(NDIGS-1).
C
C    SPACE(I)  I=1...LI+1 MUST BE AVAILABLE FOR SCRATCH
C
C
C
C OUTPUTS      THE ONLY OUTPUT WILL BE ON LOGICAL  ITAPE.  VALUES ALONG
C              ROWS ON THE PRINTED PAGE WILL CORRESPOND TO INCREASING
C              VALUES OF THE INDEX  I  IN FOFIJ(I,J).  SEE EXAMPLES
C              BELOW FOR GENERAL FORMAT.
C
C
C EXAMPLES
C
C 1. INPUTS  - FOFIJ(1...9,1...2) = 11.,21.,....,91.,, -12.,-22.,....,-92.
C              IFOFIJ(1...75) = 1,2,...,75
C
C    USAGE   -       DIMENSION  FOFIJ(15,2),IFOFIJ(75,1),SPACE(76)
C                    DO 10  NDIGS=1,2
C                    CALL MOUTAI(2,2,FOFIJ,6HFOFIJ1,9,2,15,NDIGS,.1,
C                   1            SPACE)
C                    CALL MOUTAI(2,2,FOFIJ,6HFOFIJ2,9,2,15,NDIGS,-1.,
C                   1            SPACE)
C                    CALL MOUTAI(2,2,FOFIJ,6HFOFIJ3,9,2,15,NDIGS,-2.,
C                   1            SPACE)
C                10  CALL MOUTAI(2,2,IFOFIJ,6HIFOFIJ,75,1,75,NDIGS,0.,
C                   1            SPACE)
C
C    OUTPUTS - 64 OUTPUT LINES ARE CREATED ON LOGICAL  2 .  COLUMNS  2
C              THRU  51  ARE SHOWN BELOW (COLUMN  1  IS ALWAYS BLANK).
C
C
C               FOFIJ1(I,J) *   0.10000E 00 AND ROUNDED
C
C                  J                              I = 1 ...    9
C
C                  1/ 1 2 3 4 5 6 7 8 9
C                  2/-1-2-3-4-5-6-7-8-9
C
C
C               FOFIJ2(I,J) *   0.10000E 00 AND ROUNDED
C
C                  J                              I = 1 ...    9
C
C                  1/ 1 2 3 4 5 6 7 8 9
C                  2/-1-2-3-4-5-6-7-8-9
C
C
C               FOFIJ3(I,J) *   0.10870E-01 AND ROUNDED
C
C                  J                              I = 1 ...    9
C
C                  1/ 0 0 0 0 1 1 1 1 1
C                  2/-0-0-0-0-1-1-1-1-1
C
C
C               IFOFIJ(I,J)
C
C                  J                              I = 1 ...   75
C
C                  1/ 1 2 3 4 5 6 7 8 910111213141516171819202122
```

```
C                   6162636465666768697071727374 75
C
C
C              FOFIJ1(I,J) *   0.10000E 00 AND ROUNDED
C
                 J                                 I = 1 ...    9
C
C                1/  1  2  3  4  5  6  7  8  9
C                2/ -1 -2 -3 -4 -5 -6 -7 -8 -9
C
C
C              FOFIJ2(I,J) *   0.10000E 01 AND ROUNDED
C
C                J                                 I = 1 ...    9
C
C                1/ 11 21 31 41 51 61 71 81 91
C                2/-12-22-32-42-52-62-72-82-92
C
C
C              FOFIJ3(I,J) *   0.10870E 00 AND ROUNDED
C
C                J                                 I = 1 ...    9
C
C                1/  1  2  3  4  5  7  8  9 10
C                2/ -1 -2 -3 -4 -5 -7 -8 -9-10
C
C
C              IFOFIJ(I,J)
C
C                J                                 I = 1 ...   75
C
C                1/  1  2  3  4  5  6  7  8  9 10 11 12 13 14 1
C                   41 42 43 44 45 46 47 48 49 50 51 52 53 54 5
C
C
C
C PROGRAM FOLLOWS BELOW
C
C
C DUMMY DIMENSIONS
C
      DIMENSION FOFIJ(2),SPACE(2)
C
C TRUE DIMENSIONS
C
      DIMENSION FMT(5),NW(4)
C
C BRING IN SOME INPUTS, DO INITIAL CARIAGE SPACING
C
      ITP = ITAPE
      LX = LI
      LY = LJ
      IDIM = IDIMEN
      NDGS = NDIGS
      SCA = SCALE
      BIGEST = 10.0**NDGS
      LSP = LX+1
      CALL CARIGE(ITP,NSPACE)
C
C THE NO. WORDS PER ROW, NW(NDGS), IS
C     NW(1...5) = 60, 40, 30, 25, 20
C
      NW = 5*(9-NDGS)
      IF (NDGS-2)    35,30,40
 30   NW = 40
      GO TO 40
 35   NW = 60
C
C CONSTRUCT THE FORMAT FMT(1...5)  FOR OUTPUTING ROWS
C
 40   NW(2) = NDGS+1
      IF (NW-LX) 50,45,50
 45   NW=NW+1
 50   NW(3)=NW(1)
      NW(4)=NW(2)
```

```
      CALL GNHOL2 (NW,4,41H(9H(1XI4,1H/I2,1HII1,4H/(6XI2,1HII1,2H))),
     1   FMT,IDUM,IDUM,IDUM)
C
C NOW WE HAVE TO SET THE SCALE FACTOR SCL
C      (ANTICIPATE FOR THE CASE SCALE GRTHN= 0.)
C
      SCL = SCA
      IF (SCA)     100,200,200
C
C FOR THE NEGATIVE CASE WE ARE GOING TO NEED THE RESULT FROM
C MAXABM REGARDLESS.  TEST THE RESULT FOR ZERO, THEN BRANCH
C ON SCALE = -1. OR -2. (ANTICIPATING A -2.)
C
 100  CALL MAXABM(FOFIJ,LX,LY,IDIM,FMAXAB,TEMP,TEMP)
      SCL = 0.0
      X = ABSF(FMAXAB)
      IF (X)   120,200,120
 120  SCL = .1*BIGEST/X
      IF (SCA+1.0)    200,150,200
C
C SET SCALING FOR THE CASE SCALE = -1.0
C
C      THIS IS ALMOST THE SAME AS THE FOLLOWING PROBLEM
C GIVEN X GRTHN 0,   N = INTEGER GRTHN= 0,
C FIND M = POS. OR NEG. INTEGER SUCH THAT
C     10EXP(N)  LSTHN=  X*10EXP(M)  LSTHN  10EXP(N+1),
C WHOSE SOLUTION IS
C      M = N - (LOG(X))TRUNCATED + EPS
C      WHERE EPS = 0 IF X GRTHN=0,  = 1 IF X LSTHN 0.
C
C OUR CASE IS SLIGHTLY DIFFERENT DUE TO A TERMINAL ROUNDING
C OF X*10EXP(M), WHICH MAY ROUND IT UP INTO 10EXP(N+1).
C
C IN OUR CASE N = NDIGS-1,  X = ABSF(FMAXAB)
C
 150  M = NDGS-1-XFIXF(.43429448*LOGF(X))
      IF (X-1.0)  160,170,170
 160  M = M+1
 170  SCL = 10.0**M
      IF (RNDF(SCL*X)-BIGEST)  200,180,180
 180  SCL = SCL/10.0
C
C PROCEED WITH THE HEADING OUTPUT
C
 200  IF (SCL)  215,205,215
 205  WRITE OUTPUT TAPE ITP,210,FNAME,LX
 210  FORMAT(1X,A6,5H(I,J),//,4X,1HJ,30X,10HI = 1 ... ,I4,/,1H )
      GO TO 700
 215  WRITE OUTPUT TAPE ITP,220,FNAME,SCL,LX
 220  FORMAT(1X,A6,8H(I,J) * ,E13.5,12H AND ROUNDED,//,
     1  4X,1HJ,30X,10HI = 1 ... ,I4,/,1H )
C
C OUTPUT THE ROWS ONE BY ONE, FIRST MOVING THEM INTO SPACE(2...LX+1)
C      WITH SCALING AND CONVERSION IF NECESSARY
C
 700  DO 770   IXROW=1,LY
      SPACE(1) = SAMEF(IXROW)
      IXF = 1+(IXROW-1)*IDIM
      IF (SCL)   740,730,740
 730  CALL MOVE(LX,FOFIJ(IXF),SPACE(2))
      GO TO 760
 740  CALL MULPLY(FOFIJ(IXF),LX,SCL,SPACE(2))
      CALL FIXVR(SPACE(2),LX,SPACE(2))
 760  WRITE OUTPUT TAPE ITP,FMT,(SPACE(I),I=1,LSP)
 770  CONTINUE
C
C EXIT
C
 9999 RETURN
      END
```

```
*       MOVE (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0091
*       FAP
*MOVE
        COUNT   75
        LBL     MOVE
        ENTRY   MOVE (N,SOURCE,DEST)
*
*                   ----ABSTRACT----
*
* TITLE - MOVE
*        MOVE A VECTOR TO A DIFFERENT LOCATION
*
*              MOVE MOVES A VECTOR TO A DIFFERENT LOCATION.  OVERLAP
*              OF THE SOURCE AND DESTINATION VECTORS IS ALLOWED.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709, 7090, 7094 (MAIN FRAME ONLY)
* STORAGE   - 32 REGISTERS
* SPEED     - ABOUT  8*N + 34  MACHINE CYCLES ON THE 7094 WHERE N IS
*                THE LENGTH OF THE VECTOR.
* AUTHOR    - J.F. CLAERBOUT,  JUNE, 1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*      CALL MOVE  (N,SOURCE,DEST)
*
* INPUTS
*
*    SOURCE(I) I=1...N IS A VECTOR OF WORDS.
*              (NEED NOT HAVE FLOATING NAME)
*
*    N         IS FORTRAN II INTEGER.
*              MUST BE GRTHN=0.
* OUTPUTS
*
*    DEST(I)   I=1...N IS THE SOURCE VECTOR.
*              (NEED NOT HAVE FLOATING NAME)
*
* EXAMPLES
*
*              LET SOURCE(I),I=1...K BE EQUIVALENT TO DEST(J),J=1...K
* 1. INPUTS  - SOURCE(1...3) = 1.,2.,3.  N=3  I=1  J=5
*    OUTPUTS - DEST(5...7) = 1.,2.,3.
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  J=2
*    OUTPUTS - DEST(2...4) = 1.,2.,3.
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  J=1
*    OUTPUTS - DEST(1...3) = 1.,2.,3.
*
* 4. INPUTS  - SOURCE(3...5) = 1.,2.,3.  N=3  I=3  J=1
*    OUTPUTS - DEST(1...3) = 1.,2.,3.
*
* 5. INPUTS  - SOURCE(1...3) = 1.,2.,3.  N=0  I=1  J=5
*    OUTPUTS - DEST(5...7) IS UNCHANGED.
*
        HTR     0
        BCI     1,MOVE
 MOVE   SXD     *-2,4
        NZT*    1,4
        TRA     4,4
        CLA     2,4
        ADD     =1
        STA     SPA
        STA     SPB
        CLA     3,4
        ADD     =1
        STA     DPA
        STA     DPB
        CLA     2,4            DECIDE WHICH ONE OF THE TWO MOVING
        SUB     3,4            LOOPS IS BEST ( IN CASE OF OVERLAP)
```

```
       TMI     PB              DEST IS IN HIGHER MEMORY LOC
       CLA*    1,4             SOURCE IS IN HIGHER MEMORY LOC
       PDX     ,4
SPA    CLA     **,4
DPA    STO     **,4
       TIX     *-2,4,1
       TRA     SV4
PB     CLA*    1,4
       STD     PBL
       AXT     1,4
SPB    CLA     **,4
DPB    STO     **,4
       TXI     *+1,4,1
PBL    TXL     SPB,4,**
SV4    LXD     MOVE-2,4
       TRA     4,4
       END
```

```
*       MOVECS (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0105
*       FAP
*MOVECS
        COUNT   150
        LBL     MOVECS
        ENTRY   MOVECS (LXY1,X1,Y1,...,LXYN,XN,YN)
*
*                       ----ABSTRACT----
*
* TITLE - MOVECS
*         MOVE AN ARBITRARY SET OF VECTORS
*
*               MOVECS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE.
*               THE ARGUMENTS ARE CONSIDERED IN TRIPLETS, EACH TRIPLET
*               SPECIFYING THE MOVING OF ONE VECTOR (LENGTH, SOURCE,
*               DESTINATION).  THE OUTPUT VECTORS MAY OVERLAP THE SOURCE
*               VECTORS IN ANY MANNER.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 24 REGISTERS
* SPEED      - 5 + 28*N + 8*M  MACHINE CYCLES, WHERE N = NO. VECTORS
*                 MOVED, AND M = THEIR COMBINED LENGTH.
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                       ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - MOVE
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL MOVECS(LXY1,X1,Y1,LXY2,X2,Y2,...,LXYN,XN,YN)
*
*               THE ARGUMENT COUNT MUST BE A MULTIPLE OF 3 .  IF NOT,
*               AN IMPROPER RETURN WILL RESULT.
*
* INPUTS
*
*    X1(I)      I=1...LXY1  IS FIRST VECTOR (ANY MODE) TO BE MOVED
*
*    LXY1       EXCEEDS ZERO
*
*         ETC
*
*    XN(I)      I=1...LXYN  IS LAST VECTOR (ANY MODE) TO BE MOVED
*
*    LXYN       EXCEEDS ZERO
*
* OUTPUTS       IF ANY LXY  IS ZERO OR NEGATIVE THE CORRESPONDING
*               VECTOR IS NOT MOVED.
*
*    Y1(I)      I=1...LXY1  WILL = X1(1...LXY1)  IF LXY1 EXCEEDS 0
*
*         ETC
*
*    YN(I)      I=1...LXYN  WILL = XN(1...LXYN) IF LXYN EXCEEDS 0
*
*               EQUIVALENCE (Y(K),X(L)) PERMITTED FOR ANY K,L PAIR.
*               VECTORS ARE MOVED IN SAME ORDER AS THEY APPEAR IN THE
*                  ARGUMENT STRING
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3)=1.,2.,3.  IX(1...4)=1,2,3,4  U=0.0
*    USAGE   -       CALL MOVECS(3,X,Y,4,IX,IY,1,IX,IZ)
*                    CALL MOVECS(3,X,W,0,X,U,3,W,Z)
*    OUTPUTS - Y(1...3)=1.,2.,3.  IY(1...4)=1,2,3,4  IZ=1
*              W(1...3) = 1.,2.,3.  U = 0.0  (NO OUTPUT CASE)
*                Z(1...3) = 1.,2.,3.
*
* 2. INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   -       CALL MOVECS(2,X(2),X(1),3,IX(1),IX(2))
*    OUTPUTS - X(1...3)=2.,3.,3.  IX(1...4)=1,1,2,3
*
```

```
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR HAS MOVE(LXY,X,Y)
       HTR     0                XR4 ORIGINAL
       BCI     1,MOVECS
* ONLY ENTRY.  MOVECS(LXY1,X1,Y1,...,LXYN,XN,YN)
MOVECS SXD     MOVECS-2,4
* CHECK IF NEXT ARGUMENT IS TSX  A,0
GETLXY CAL     1,4
       STA     LXY              (START SETTING MOVE CALL SEQUENCE)
       ANA     MASK
       LAS     TSXZ
       TRA     LEAVE            NO
       TRA     MORE             YES
* EXIT WHEN FIRST OF TRIPLET IS NON-TSX  A,0
 LEAVE TRA     1,4              NO
* COMPLETE CALLING SEQUENCE FOR MOVE.  TSX$ .  BACK FOR NEXT THREE
 MORE  CLA     2,4
       STA     X
       CLA     3,4
       STA     Y
       SXA     SV4,4
       TSX     $MOVE,4
 LXY   TSX     **,0             **=A(NEXT LXY)
 X     TSX     **,0             **=A(NEXT X)
 Y     TSX     **,0             **=A(NEXT Y)
 SV4   AXT     **,4             **=XR4 VARIABLE
       TXI     GETLXY,4,-3
* CONSTANTS
 MASK  OCT     777777700000
 TSXZ  TSX     0,0
       END
```

```
*      MOVREV  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0155
*      FAP
*MOVREV
       COUNT   150
       LBL     MOVREV
       ENTRY   MOVREV  (LXY,IX,X,IY,Y,SIGN)
*
*                    ----ABSTRACT----
*
* TITLE - MOVREV
*        MOVE, REVERS, CHANGE SPACING, OR CHANGE SIGN OF A VECTOR
*
*              MOVREV MOVES A VECTOR X(I) TO Y(J) ACCORDING TO
*
*                    Y(1)         = SIGN*X(1)
*                    Y(1+IY)      = SIGN*X(1+IX)
*                    .
*                    .
*                    Y(1+LXY*IY) = SIGN*X(1+LXY*IX)
*
*              OR ACCORDING TO
*
*                    Y(1)         = SIGN*X(1+LXY*IX)
*                    Y(1+IY)      = SIGN*X(1+(LXY-1)*IX)
*                    .
*                    .
*                    Y(1+LXY*IY) = SIGN*X(1)
*
*              WHERE LXY, IX, IY, AND SIGN (=+1. OR -1.) ARE INPUT
*              PARAMETERS.  OVERLAP OF INPUT AND OUTPUT IS ALLOWED
*              EXCEPT WHEN THE REVERSE AND MOVE FEATURE ARE USED AT
*              THE SAME TIME.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 74 REGISTERS
* SPEED      - 100 + 10*LXY MACHINE CYCLES
* AUTHOR     - R.A. WIGGINS,  JULY,1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MOVREV(LXY,IX,X,IY,Y,SIGN)
*
* INPUTS
*
*    LXY       DEFINES NUMBER OF DATA VALUES TO BE MOVED.
*              MUST BE GRTHN=1
*
*    IX        DEFINES THE INCREMENT OF THE X INDEX.
*              MUST BE GRTHN=0
*
*    X(I)      I=1,1+IX,...,1+(LXY-1)*IX CONTAINS THE SERIES TO BE MOVED
*              NEED NOT BE FLOATING POINT.
*
*    IY        DEFINES THE INCREMENT OF THE Y INDEX.
*              IF NEGATIVE THE SERIES WILL BE REVERSED ON OUTPUT.
*
*    SIGN      IF GRTHN 0 THE SIGN IS NOT CHANGED
*              IF LSTHN=0 THE SIGN OF EACH TERM MOVED IS CHANGED.
*
* OUTPUTS
*
*    Y(I)      I=1,1+IY,...,1+(LXY-1)*IY CONTAINS THE MOVED SERIES
*              MAY OVERLAP X(I) WHEN IY=1
*
* EXAMPLES
*
* 1. INPUTS  - LXY=3  IX=1  IY=2  X(1...3) = 1.,2.,3.
*              SIGN=1.              Y(1...6) = .1,.1,...,.1
*    OUTPUTS - Y(1...6) = 1.,.1,2.,.1,3.,.1
*
```

```
      * 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT SIGN=-1  IX=0
      *    OUTPUTS - Y(1...6) = -1.,.1,-1.,.1,-1.,.1
      *
      * 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IY=-1
      *    OUTPUTS - Y(1...3) = 3.,2.,1.
      *
      * PROGRAM FOLLOWS BELOW
      *
       XR1    HTR     0
       XR2    HTR     0
       XR4    HTR     0
              BCI     1,MOVREV
      MOVREV  SXD     XR1,1              SAVE
              SXD     XR2,2              INDEX
              SXD     XR4,4              REGISTERS AND
              STI     SI                 INDICATORS.
              SIR     3                  LOAD INDICATORS.
       CLA    CLA     5,4                =ADR(X)
              STA     DPA
              CLA     3,4                =ADR(Y)
              STA     SPA
              SUB     5,4                DECIDE WHICH DIRECTION
              TMI     *+2                TO MOVE
              RIR     3                  SAVE IN LEFT HALF INDICATOR
              PXD     ,0                 CHECK FOR ILLEGAL VALUES OF
              CAS*    1,4                LXY
       CLS    CLS     *
              TRA     LV                *EXIT IF ZERO OR NEGATIVE*
              CAS*    2,4                AND IX
              TRA     LV                *EXIT IF NEGATIVE*
              NOP
              LDQ     CLA
              CAS*    6,4                CHECK SIGN CONVENTION
              NOP
              LDQ     CLS                ZERO OR NEGATIVE
              SLQ     SPA                POSITIVE.
              CAS*    4,4                CHECK SIGN OF IY
              SIR     1                  SET
              RIR     2                  INDICATORS
              ADM*    4,4
              RNT     2
              SUB     =32768B17
              STD     TXI2
              CLA*    2,4
              RNT     1
              SUB     =32768B17
              STD     TXI1
              RNT     1
              TRA     XIX
              CLM
       A1     PAX     ,1
              RNT     2
              TRA     XIY
              CLM
       A2     PAX     ,2
              CLA*    1,4
              PDX     ,4
       SPA    CLA     **,1
       DPA    STO     **,2
       TXI1   TXI     *+1,1,**
       TXI2   TXI     *+1,2,**
              TIX     SPA,4,1
       LV     LDI     SI
              LXD     XR1,1
              LXD     XR2,2
              LXD     XR4,4
              TRA     7,4
       XIX    CLA*    1,4
              SUB     =1B17
              XCA
              MPY*    2,4
              ARS     1
              TRA     A1
       XIY    CLA*    1,4
              SUB     =1B17
```

****
*   MOVREV   *
****
(PAGE 3)

```
        XCA
        MPY*    4,4
        ARS     1
        TRA     A2
   SI   PZE
        END
```

```
*      MPSEQ1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0196
*      FAP
*MPSEQ1
       COUNT   200
       LBL     MPSEQ1
       ENTRY   MPSEQ1 (X,LX,B,LB,IX,IXLO,IANS)
*
*                    ----ABSTRACT----
*
* TITLE - MPSEQ1
*       MAPS A SEQUENCE OF NUMBERS INTO AN INTEGER SERIES
*
*              MPSEQ1 MAPS A SEQUENCE X(I), I=1,...,LX INTO AN INTEGER
*              SEQUENCE IX(I), I=1,...,LX.  THE MAPPING IS CONTROLLED BY
*              A GIVEN VECTOR OF RANGE LIMITS B(I), I=1,...,LB, WHERE
*              B(I) IS MONOTONELY INCREASING FROM B(1) TO B(LB), THUS
*              SPECIFYING LB-1 SEPARATE RANGES.  EACH RANGE IS CONSIDERED
*              CLOSED ON THE LOWER END, OPEN ON THE HIGH END AND THE
*              RANGES ARE INDEXED FROM IXLO+1 TO IXLO+LB-1, WHERE IXLO
*              IS A PARAMETER.  IX(I) IS THEN SET EQUAL TO THE INDEX OF
*              THE RANGE TO WHICH X(I) BELONGS, WITH THE FOLLOWING
*              TREATMENT OF EXTREMAL X VALUES
*                IF X(I) IS LSTHN  B( 1),  IX(I) = IXLO
*                IF X(I) IS GRTHN= B(LB),  IX(I) = IXLO+LB
*              NOTE- THE LOGIC USED IS ALMOST IDENTICAL TO THAT OF FRQCT2
*
* LANGUAGE  - FAP SUBROUTINE WITH FORTRAN II CALLING SEQUENCE
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 110 REGISTERS
* SPEED     -
* AUTHOR    - J. N. GALBRAITH
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL MPSEQ1(X,LX,B,LB,IX,IXLO,IANS)
*
* INPUTS
*
*    X(I)       I=1...LX IS THE INPUT SERIES TO BE MAPPED.
*                 MAY BE FLOATING, FORTRAN INTEGER, OR MACHINE LANGUAGE
*                 INTEGER, BUT MUST BE THE SAME MODE AS B(J).
*
*    LX         IS LENGTH OF X VECTOR.
*               MUST BE GRTHN=1.
*
*    B(I)       I=1...LB GIVES INPUT RANGES OF MAPPING INTERVALS.
*               MUST BE SAME MODE AS X(I).
*               B(I) MUST INCREASE MONOTONELY, I.E. B(I+1) GRTHN B(I).
*
*    LB         IS LENGTH OF RANGE VECTOR.
*               MUST BE GRTHN=1.
*
*    IXLO       IS LOWER LIMIT OF OUTPUT MAPPING.  IXLO+1 = INDEX OF
*               LOWEST RANGE.
*
* OUTPUTS
*
*    IX(I)      I=1...LX IS THE INTEGER MAPPING OF X(I).
*
*    IANS       =0  NORMAL
*               =1  ILLEGAL LX
*               =2  ILLEGAL LB
*               =3  WEIRD ERROR
*
* EXAMPLES
*
* 1. INPUTS  - LX=0   X(1...16)=-5.,-4.,-3.2,-3.1,-2.,-2.1,0.,-1.1,
*               -.5,5.,4.,3.5,3.,2.9,1.1,1.   LB=16   B(1...9)=-4.,-3.,
*               -2.,-1.,0.,1.,2.,3.,4.,   IXLO=0
*    OUTPUTS - ERROR  IANS=1
*
```

```
* 2. INPUTS  - X AND B SAME AS EXAMPLE 1  LX=16  LB=0  IXLO=0
*    OUTPUTS - ERROR  IANS=2
*
* 3. INPUTS  - X AND B SAME AS EXAMPLE 1  LX=16  LB=9  IXLO=0
*    OUTPUTS - IX(1,...,16)=0,0,0,0,2,1,4,2,3,7,7,7,7,7,6,5,5   IANS=0
*
* 4. INPUTS  - X, B, LX, AND LB SAME AS EXAMPLE 3  IXLO=12
*    OUTPUTS - IX(1,...,16)=12,12,12,12,14,13,16,14,15,19,19,19,19,18,
*              17,17   IANS=0
*
        PZE    0
        BCI    1,MPSEQ1
MPSEQ1  SXA    RETURN,1
        SXA    RETURN+1,2
        SXA    RETURN+2,4
        SXD    MPSEQ1-2,4
        STZ*   7,4            IANS=0
        CLA*   2,4            GET LX
        TZE    ERR1
        TMI    ERR1
        STD    END
        CLA*   4,4            GET LB
        TZE    ERR2
        TMI    ERR2
        ARS    18             LB IN ADDRESS
        STO    LB
        ARS    1              LB/2 (IN ADDRESS)
        STO    LBHALF
        CLA    1,4            ADDRESS OF X
        ADD    K1MLI          A(X+1)
        STA    XADD
        STA    TESTLO
        CLA    3,4            ADDRESS OF B
        ADD    K1MLI          A(B+1)
        STA    BTEST1
        STA    BADD
        SUB    LB
        STA    TESTHI
        CLA*   6,4            GET IXLO
        SUB    K2FX           IXLO-2
        STO    XLOW
        CLA    5,4            ADDRESS OF IX
        ADD    K1MLI          A(IX+1)
        STA    IXSTO
        AXT    1,1
        AXT    1,4
 LOOP   CLA    K1MLI
        STO    LBLO           INITIAL LBLO=1
        CLA    LB
        STO    LBHI           INITIAL LBHI=LB
        CLA    LBHALF
        STO    LBCOM          INITIAL LBCOM=LB/2
        AXT    1,2
TESTLO  CLA    **,1           GET X.  (**=A(X+1))
BTEST1  CAS    **,4           B(1)  SEE IF IN LOWEST RANGE
        TRA    TESTHI
        TRA    NEXIND
        TRA    NEXIND
TESTHI  CAS    **             **=A(B(LB)). SEE IF IN HIGHEST RANGE
        TRA    HIEST
        TRA    HIEST
SEARCH  LXA    LBCOM,2
 XADD   CLA    **,1           GET X(IR1)
 BADD   CAS    **,2           COMPARE WITH B(LBCOM)
        TRA    GRATER         X GREATER, NEW LBLO (=LBCOM)
        TRA    NEXIND         GOT IT, SET IX(IR1+1)
 LESS   PXA    0,2            X LESS, NEW LBHI (=LBCOM)
        SUB    LBLO           LBCOM-LBLO=DIF
        CAS    K1MLI
        TRA    *+3            DIF GREATER THAN ONE
        TRA    EQUAL          DIF=1, GOT IT, SET IX(IR1+1)
        TRA    ERROR             IMPOSSIBLE
        ARS    1              DIF/2
        ADD    LBLO           NEW LBCOM
        LDQ    LBCOM
```

MPSEQ1
(PAGE 3)

```
          STQ     LBHI
          STO     LBCOM
          TRA     SEARCH
   GRATER PXA     0,2
          SUB     LBHI          LBCOM-LBHI=-DIF
          SSP                   DIF
          CAS     K1MLI
          TRA     *+3
          TRA     NEXIND        DIF=1, GOT IT, SET IX(IR1+1)
          TRA     ERROR         IMPOSSIBLE
          ARS     1
          ADD     LBCOM
          LDQ     LBCOM
          STO     LBCOM
          STQ     LBLO
          TRA     SEARCH
   NEXIND TXI     *+1,2,1
    EQUAL PXD     ,2
          ADD     XLOW
    IXSTO STO     **,1          **= ADDRESS OF IX+1
          TXI     *+1,1,1
    END   TXL     LOOP,1,**     **=LX
   RETURN AXT     **,1
          AXT     **,2
          AXT     **,4
          TRA     8,4
    HIEST LXA     LB,2
          TRA     EQUAL
    ERR1  CLA     K1FX
          STO*    7,4           STORE IANS
          TRA     8,4           RETURN
    ERR2  CLA     K2FX
          TRA     ERR1+1
    ERROR CLA     K3FX
          TRA     ERR1+1
   * CONSTANTS AND TEMPORARIES
    K1FX  PZE     0,0,1
    K2FX  PZE     0,0,2
    K3FX  PZE     0,0,3
    K1MLI PZE     1,0,0
    LB    PZE     0
   LBHALF PZE     0
   LBLO   PZE     0
   LBCOM  PZE     0
   LBHI   PZE     0
    XLOW  PZE
          END
```

```
*     MRVRS (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0066
*     LABEL
CMRVRS
      SUBROUTINE MRVRS (N,M,LA,AA)
C
C                       ----ABSTRACT----
C
C TITLE - MRVRS
C       REVERSE VECTOR OF MATRICES
C
C              MRVRS REVERSES THE ORDER OF MATRICES IN A VECTOR.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 61 REGISTERS
C SPEED     - 10*N*M*LA + 41*LA + 110  MACHINE CYCLES ON THE 7090
C               WHERE N*M IS THE NUMBER OF ELEMENTS IN A MATRIX AND
C               LA IS THE NUMBER OF MATRICES IN THE VECTOR.
C AUTHOR    - R.A. WIGGINS     3/63
C
C                       ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - REVERS
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C     CALL MRVRS (N,M,LA,AA)
C
C INPUTS
C
C    N          IS NUMBER OF ROWS IN A MATRIX IN AA.
C               MUST BE GRTHN=1
C
C    M          IS NUMBER OF COLUMNS IN A MATRIX IN AA.
C               MUST BE GRTHN=1
C
C    LA         IS NUMBER OF MATRICES IN THE VECTOR OF MATRICES AA.
C               MUST BE GRTHN=1
C
C    AA(I)      I=1,...,LA*N*M IS A VECTOR OF MATRICES STORED CLOSELY
C                 PACKED.
C
C OUTPUTS
C
C    AA(I)      I=1,...,LA*N*M IS THE REVERSED INPUT VECTOR.
C
C EXAMPLES
C
C 1. INPUTS  - N=1 M=1 LA=1  AA(1)=3.
C    OUTPUTS - A(1)=3.
C
C 2. INPUTS  - N=2 M=1 LA=3  AA(1...6)=1.,2.,3.,4.,5.,6.
C    OUTPUTS - AA(1...6)=5.,6.,3.,4.,1.,2.
C
C 3. INPUTS  - N=2 M=1 LA=4  AA(1...8)=1.,2.,3.,4.,5.,6.,7.,8.
C    OUTPUTS - AA(1...8)=7.,8.,5.,6.,3.,4.,1.,2.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION  AA(2)
      NM   = N*M
      NMLA = NM*LA
      CALL REVERS(NMLA, AA)
      DO 10  I=1,NMLA,NM
 10   CALL REVERS(NM, AA(I))
      RETURN
      END
```

```
*     MSCON1 (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0107
*     LABEL
CMSCON1
      SUBROUTINE MSCON1 (NORDER,P,PHI,DEPEND,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - MSCON1
C       MEAN SQUARE CONTINGENCY AND DEPENDENCY FROM PROBABILITY DENSITY.
C
C              MSCON1 COMPUTES THE MEAN SQUARE CONTINGENCY AND A
C              DEPENDENCY MEASURE AS DEFINED ON PAGE 282 OF CRAMER,
C              MATHEMATICAL METHODS OF STATISTICS, PRINCTON UNIV. PRESS,
C              1951. THE COMPUTATION REQUIRES THE SECOND PROBABILITY
C              DENSITY WHICH CAN BE COMPUTED WITH SUBROUTINE PROB2 (SEE
C              WRITE-UP OF PROB2). IF PHI IS THE MEAN SQUARE CONTINGENCY,
C              DEPEND IS THE DEPENDENCY MEASURE, AND NORDER IS THE ORDER
C              OF THE SECOND PROBABILITY MATRIX, P(I,J), THEN
C
C                          DEPEND = PHI/(NORDER-1)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709, 7090 (MAIN FRAME ONLY)
C STORAGE   - 238 REGISTERS
C SPEED     -
C AUTHOR    - J.N. GALBRAITH
C
C                     ----USAGE---
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C      CALL MSCON1(NORDER,P,PHI,DEPEND,IANS)
C
C INPUTS
C
C    NORDER     INTEGER. THE ORDER OF THE P(I,J) PROBABILITY DENSITY
C               MATRIX. GRTHN ONE, LSTHN OR EQUAL 25.
C
C    P(I,J)     I=1,..,NORDER, J=1,..,NORDER. PROBABILITY DENSITY MATRIX
C               NORMALIZED SUCH THAT THE SUM OVER I AND J IS = TO 1.
C               P(I,J) HAS DIMENSION (25,25), P(I,J) MUST NOT HAVE AN
C               ENTIRE ROW OR COLUMN SUM EQUAL TO ZERO, OR NEGATIVE.
C
C OUTPUTS
C
C    PHI        THE MEAN SQUARE CONTINGENCY.
C
C    DEPEND     THE DEPENDENCY MEASURE.
C
C    IANS       ERROR INDICATOR
C               =0  NORMAL
C               =-1 ILLEGAL NORDER. LSTHN 1 OR GRTHN 25
C               =-2 ILLEGAL P MATRIX. ROW OR COLUMN SUM ZERO OR NEGATIVE.
C
C EXAMPLES
C
C 1. INPUTS   - P(I,1)=.2 ,P(I,I),I=2,5 =.1, P(1,I),I=2,5 =.1
C               ALL OTHER P(I,J)=0.
C               NORDER=0
C    OUTPUTS - PHI=0.  DEPEND=0.  IANS=-1
C
C 2. INPUTS   - SAME AS EXAMPLE 1 EXCEPT
C               NORDER=26
C    OUTPUTS - PHI=0.  DEPEND=0.  IANS=-1
C
C 3. INPUTS   - SAME AS EXAMPLE 1 EXCEPT
C               NORDER=5
C    OUTPUTS - PHI=1.6666666  DEPEND=.41666666  IANS=0
C
C 4. INPUTS   - SAME AS EXAMPLE 1 EXCEPT
C               P(1,5)=0., P(5,1)=.1  NORDER=5
C    OUTPUTS - PHI=1.7333333   DEPEND=.43333333   IANS=0
C
```

```
C 5. INPUTS  - SAME AS EXAMPLE 4 EXCEPT
C                P(5,5)=0.
C    OUTPUTS - IANS=-2
C
      DIMENSION P(25,25),PSROW(25),PSCOL(25)
C     CHECK NORDER
      IANS=-1
      IF(NORDER-1) 9999,9999,5
 5    IF(NORDER-26) 6,9999,9999
C     FIND ROW AND COLUMN SUMS
 6    DO 10 J=1,NORDER
      PSROW(J)=0.
      PSCOL(J)=0.
      DO 10 I=1,NORDER
      PSROW(J)=PSROW(J)+P(J,I)
 10   PSCOL(J)=PSCOL(J)+P(I,J)
C     CHECK ROW AND COLUMN SUMS
      IANS=-2
      DO 15 I=1,NORDER
      IF(PSROW(I)) 9999,9999,12
 12   IF(PSCOL(I)) 9999,9999,15
 15   CONTINUE
C     COMPUTE MEAN SQUARE CONTINGENCY
      PHI=0.
      DO 20 I=1,NORDER
      DO 20 J=1,NORDER
 20   PHI=PHI+P(I,J)*P(I,J)/(PSROW(I)*PSCOL(J))
      PHI=PHI-1.
C     COMPUTE DEPENDENCY MEASURE
      DEPEND=PHI/(FLOATF(NORDER-1))
      IANS=0
 9999 RETURN
      END
```

```
************************
*     MULK             *
************************
 REFER TO
     ADDK
```


```
************************
*     MULK             *
************************
 REFER TO
     ADDK
```

```
*     MULK  -II (SUBROUTINE)           9/29/64   LAST CARD IN DECK IS NO. 0077
*     LABEL
CMULK  -II
      SUBROUTINE MULK (C)
C
C                    ----ABSTRACT----
C
C TITLE - MULK  -II
C        MULTIPLY ANY NO. OF VARIABLES BY A SINGLE FLTG. PT. CONSTANT
C
C              MULK IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
C              WHICH MULTIPLIES EACH OF ITS ARGUMENTS BEYOND THE FIRST
C              BY THE FIRST ARGUMENT, ASSUMED TO BE FLOATING POINT.
C
C              THIS SUBROUTINE IS THE FORTRAN EQUIVALENT OF THE
C              FAP SUBROUTINE OF THE SAME NAME.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 76 REGISTERS
C SPEED      -
C AUTHOR     - S.M. SIMPSON, AUGUST 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - SETUP, ARG, STORE, RETURN
C       AND FORTRAN SYSTEM ROUTINES - (NONE)
C
C FORTRAN USAGE
C     CALL MULK (C,X1,X2,...,XN)
C
C INPUTS
C
C    C         IS THE VALUE BY WHICH X1...XN  ARE TO BE MULTIPLIED
C              MUST BE FLTG. PT.
C
C OUTPUTS      NO OUTPUTS IF ARGUMENT COUNT IS LESS THAN 2 (PURE RETURN)
C
C    X1        = C * X1
C    X2        = C * X2
C           ETC.
C    XN        = C * XN
C
C              EQUIVALENCES OF ANY ARGUMENTS ARE PERMITTED, THE
C              BEHAVIOUR DEPENDS ON THE FACT THAT X1...XN ARE SET
C              IN THAT ORDER.
C
C    C         IS AN OUTPUT IF ANY X IS EQUIVALENT TO C.  IF SO
C              SUCCEEDING ARGUMENTS ARE MULTIPLIED BY THE NEW C.
C
C EXAMPLES
C
C 1. INPUTS  - X=1., Y=2., Z=3., U=4., V=5., W=6.
C    USAGE   -       CALL MULK( 2.,X,Y,Z,U)
C                    CALL MULK(V,W)
C                    CALL MULK(V)
C                    CALL MULK
C    OUTPUTS -       X=2., Y=4., Z=6., U=8.
C                    W=30.,
C                    V=5.
C                        NO OUTPUTS FROM LAST TWO CALLS
C
C 2. INPUTS  - SAME AS EXAMPLE 1  EXCEPT C = 2.
C    USAGE   -       CALL MULK (C, X, Y, Z, C, U, V, V, W)
C    OUTPUTS -       X=2., Y=4., Z=6., C=4., U=16., V=80., W=24.
C
C PROGRAM FOLLOWS BELOW
C
C GET ARGUMENT COUNT AND CHECK IT
      CALL SETUP(LOCALL,NARGS,XR1,XR2)
      IF (NARGS-2)  9999,10,10
C LOOP TO SET X1...XN
 10   DO 20  NUMARG=2,NARGS
      XOUT=C*ARGF(LOCALL,NUMARG,1)
 20   CALL STORE(XOUT,LOCALL,NUMARG,1)
```

```
*************************
*    MULK  -II          *
*************************
                 (PAGE  2)
```


```
C EXIT
 9999 CALL RETURN(LOCALL,XR1,XR2)
      END
```

```
************************
*    MULKS             *
************************
 REFER TO
     ADDK
```


```
************************
*    MULKS             *
************************
 REFER TO
     ADDK
```

```
*     MULLER (SUBROUTINE)              9/9/64   LAST CARD IN DECK IS NO. 0231
*     LABEL
CMULLER
      SUBROUTINE MULLER (COE,N1,ROOTR,ROOTI)
C
C
C                     ----ABSTRACT----
C
C TITLE - MULLER
C        POLYNOMIAL ROOT FINDER
C
C               MULLER FINDS THE REAL AND COMPLEX ROOTS OF A POLYNOMIAL
C               WITH REAL COEFFICIENTS.  THE METHOD USED IS TAKEN FROM
C               MULLER, 'A METHOD OF SOLVING ALGEBRAIC EQUATIONS USING
C               AN AUTOMATIC COMPUTER', MTAC (1956), 280-215.
C
C               ALL ARITHMETIC IS DONE IN THE COMPLEX MODE.  THEREFORE
C               ALL ROOTS FOUND WILL HAVE REAL AND IMAGINARY PARTS.
C               REAL ROOTS WILL HAVE SMALL IMAGINARY PARTS ON THE ORDER
C               OF 7 DECIMAL PLACES LESS THAN THE REAL PARTS.
C
C               THE PROGRAM WILL FIND MULTIPLE ROOTS BUT THE ACCURACY
C               DECREASES AS THE MULTIPLICITY INCREASES.  A NON-MULTIPLE
C               ROOT IS NORMALLY ACCURATE FROM 6 TO 8 DECIMAL PLACES.
C               WHEN THE MULTIPLICITY IS 4, THE ACCURACY DECREASES 2
C               DECIMAL PLACES.
C
C LANGUAGE    - FORTRAN II SUBROUTINE
C EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE     - 757 REGISTERS
C SPEED       - ABOUT  .010*N*N  SECONDS ON THE 7094 MOD 1,
C               WHERE  N  IS THE DEGREE OF THE POLYNOMIAL.
C AUTHOR      - IRA HANSON, LMSN, SUNNYVALE CAL.     JUNE,1960
C
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
C        AND FORTRAN SYSTEM ROUTINES - SQRT
C
C FORTRAN USAGE
C     CALL MULLER (COE,N1,ROOTR,ROOTI)
C
C
C INPUTS
C
C    COE(I)     I=1,...,N1+1 IS THE ARRAY OF POLYNOMIAL COEFFICIENTS.
C
C    N1         IS THE DEGREE OF THE POLYNOMIAL
C               MUST BE GRTHN=1
C
C
C OUTPUTS
C
C    ROOTR(I)  I=1,...,N1 IS THE REAL PARTS OF THE COMPLEX ROOTS.
C
C    ROOTI(I)  I=1,...,N1 IS THE CORRESPONDING IMAGINARY PARTS OF THE
C                 COMPLEX ROOTS.
C
C
C EXAMPLES
C
C 1. INPUTS  - N1 = 2  COE(1...3) = 2.21,-1.00,1.00
C    OUTPUTS - ROOTR(1...2) = 0.5, 0.5
C              ROOTI(1...2) = 1.4,-1.4
C
C 2. INPUTS  - N1 = 10  COE(1...11) = 1332.5,-7690.8,26130.,-46510.,
C                 51730.,-38520.,19350.,-6153.9,968.28,-4.2000,1.0000
C    OUTPUTS - ROOTR(1...10) = 0.2, 0.2, 1.5, 1.5, 1.0, 1.0, 0.5, 0.5,
C                             -1.1,-1.1
C              ROOTI(1...10) =-0.3, 0.3, 0.4,-0.4,-1.0, 1.0,-1.4, 1.4,
C                             -31.0, 31.0
C
C PROGRAM FOLLOWS BELOW
C
```

```
      DIMENSION COE(16),ROOTR(15),ROOTI(15)
      N=N1
      N2=N+1
      N4=0
      I=1
   19 IF(COE(I))9,7,9
    7 N4=N4+1
      ROOTR(N4)=0.
      ROOTI(N4)=0.
      I=I+1
      IF(N4-N )19,37,19
    9 CONTINUE
   10 AXR=0.8
      AXI=0.
      L=1
      N3=1
      ALP1R=AXR
      ALP1I=AXI
      M=1
      GOTO99
   11 BET1R=TEMR
      BET1I=TEMI
      AXR=0.85
      ALP2R=AXR
      ALP2I=AXI
      M=2
      GOTO99
   12 BET2R=TEMR
      BET2I=TEMI
      AXR=0.9
      ALP3R=AXR
      ALP3I=AXI
      M=3
      GOTO99
   13 BET3R=TEMR
      BET3I=TEMI
   14 TE1=ALP1R-ALP3R
      TE2=ALP1I-ALP3I
      TE5=ALP3R-ALP2R
      TE6=ALP3I-ALP2I
      TEM=TE5*TE5+TE6*TE6
      TE3=(TE1*TE5+TE2*TE6)/TEM
      TE4=(TE2*TE5-TE1*TE6)/TEM
      TE7=TE3+1.
      TE9=TE3*TE3-TE4*TE4
      TE10=2.*TE3*TE4
      DE15=TE7*BET3R-TE4*BET3I
      DE16=TE7*BET3I+TE4*BET3R
      TE11=TE3*BET2R-TE4*BET2I+BET1R-DE15
      TE12=TE3*BET2I+TE4*BET2R+BET1I-DE16
      TE7=TE9-1.
      TE1=TE9*BET2R-TE10*BET2I
      TE2=TE9*BET2I+TE10*BET2R
      TE13=TE1-BET1R-TE7*BET3R+TE10*BET3I
      TE14=TE2-BET1I-TE7*BET3I-TE10*BET3R
      TE15=DE15*TE3-DE16*TE4
      TE16=DE15*TE4+DE16*TE3
      TE1=TE13*TE13-TE14*TE14-4.*(TE11*TE15-TE12*TE16)
      TE2=2.*TE13*TE14-4.*(TE12*TE15+TE11*TE16)
      TEM=SQRTF(TE1*TE1+TE2*TE2)
      IF(TE1)113,113,112
  113 TE4=SQRTF(.5*(TEM-TE1))
      TE3=.5*TE2/TE4
      GO TO 111
  112 TE3=SQRTF(.5*(TEM+TE1))
      IF(TE2)110,200,200
  110 TE3=-TE3
  200 TE4=.5*TE2/TE3
  111 TE7=TE13+TE3
      TE8=TE14+TE4
      TE9=TE13-TE3
      TE10=TE14-TE4
      TE1=2.*TE15
      TE2=2.*TE16
      IF(TE7*TE7+TE8*TE8-TE9*TE9-TE10*TE10)204,204,205
```

```
  204 TE7=TE9
      TE8=TE10
  205 TEM=TE7*TE7+TE8*TE8
      TE3=(TE1*TE7+TE2*TE8)/TEM
      TE4=(TE2*TE7-TE1*TE8)/TEM
      AXR=ALP3R+TE3*TE5-TE4*TE6
      AXI=ALP3I+TE3*TE6+TE4*TE5
      ALP4R=AXR
      ALP4I=AXI
      M=4
      GO TO 99
   15 N6=1
   38 IF(ABSF(HELL)+ABSF(BELL)-1.E-20)18,18,16
   16 TE7=ABSF(ALP3R-AXR)+ABSF(ALP3I-AXI)
      IF(TE7/(ABSF(AXR)+ABSF(AXI))-1.E-7)18,18,17
   17 N3=N3+1
      ALP1R=ALP2R
      ALP1I=ALP2I
      ALP2R=ALP3R
      ALP2I=ALP3I
      ALP3R=ALP4R
      ALP3I=ALP4I
      BET1R=BET2R
      BET1I=BET2I
      BET2R=BET3R
      BET2I=BET3I
      BET3R=TEMR
      BET3I=TEMI
      IF(N3-100)14,18,18
   18 N4=N4+1
      ROOTR(N4)=ALP4R
      ROOTI(N4)=ALP4I
      N3=0
   41 IF(N4-N )30,37,37
 37   CONTINUE
      DO 380 I=1,N
      IF (ABSF(ROOTI(I))-1.E-5)  370,370,380
 370  ROOTI(I)=0.
 380  CONTINUE
      RETURN
   30 IF(ABSF(ROOTI(N4))-1.E-5)10,10,31
 31   L=L
      GO TO (32,10),L
   32 AXR=ALP1R
      AXI=-ALP1I
      ALP1I=-ALP1I
      M=5
      GO TO 99
   33 BET1R=TEMR
      BET1I=TEMI
      AXR=ALP2R
      AXI=-ALP2I
      ALP2I=-ALP2I
      M=6
      GO TO 99
   34 BET2R=TEMR
      BET2I=TEMI
      AXR=ALP3R
      AXI=-ALP3I
      ALP3I=-ALP3I
      L=2
      M=3
   99 TEMR=COE(N2)
      TEMI=0.0
      DO 100 I=1,N
      I1=N2-I
      TE1=TEMR*AXR-TEMI*AXI
      TEMI=TEMI*AXR+TEMR*AXI
  100 TEMR=      TE1+COE(I1)
      HELL=TEMR
      BELL=TEMI
   42 IF(N4)102,103,102
  102 DO101I=1,N4
      TEM1=AXR-ROOTR(I)
      TEM2=AXI-ROOTI(I)
```

****
*   MULLER   *
****
(PAGE 4)

```
      TE1=TEM1*TEM1+TEM2*TEM2
      TE2=(TEMR*TEM1+TEMI*TEM2)/TE1
      TEMI=(TEMI*TEM1-TEMR*TEM2)/TE1
  101 TEMR=TE2
 103  M=M
      GO TO(11,12,13,15,33,34),M
      END
```

```
*      MULPLY (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0113
*      FAP
*MULPLY
       COUNT    150
       LBL      MULPLY
       ENTRY    MULPLY ( X, LX,XMLPLR,XMLPLD)
       ENTRY    XMLPLY (IX,LIX,IXMPLR,IXMPLD)
*
*                     ----ABSTRACT----
*
* TITLE - MULPLY WITH SECONDARY ENTRY XMLPLY
*        MULTIPLY VECTOR BY FLOATING OR FIXED CONSTANT
*
*              MULPLY SETS A VECTOR EQUAL TO A GIVEN VECTOR TIMES A
*              FLTG CONSTANT, OUTPUT MAY REPLACE INPUT.
*
*              XMLPLY SETS A VECTOR EQUAL TO A GIVEN VECTOR TIMES A
*              FXD CONSTANT.  OUTPUT MAY REPLACE INPUT.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34 REGISTERS
* SPEED      -                7090    709
*              MULPLY   40 + (19    OR 22.2)*LX  MACHINE CYCLES,
*              XMLPLY   42 + (20.6 OR 24.8)*LX    LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL MULPLY( X, LX,XMLPLR,XMLPLD)
*     CALL XMLPLY(IX,LIX,IXMPLR,IXMPLD)
*
* INPUTS
*
*    X(I)      I=1...LX  IS A FLTG VECTOR
*
*    LX        SHOULD EXCEED ZERO
*
*    XMLPLR    IS A FLTG VARIABLE. EQUIVALENCE (XMLPLR, SOME X(I)) IS OK
*
*    IX(I)     I=1...LIX IS A FXD VECTOR
*
*    LIX       SHOULD EXCEED ZERO
*
*    IXMPLR    IS A FXD VARIABLE. EQUIVALENCE (IXMPLR, SOME IX(I)) IS OK
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1.
*
*    XMLPLD(I) I=1...LX  HAS VALUES  XMLPLR * X(I)
*              EQUIVALENCE (XMLPLD,X) IS PERMITTED.
*
*    IXMPLD(I) I=1...LIX  HAS VALUES  IXMPLR * IX(I)
*              EQUIVALENCE (IXMPLD,IX) IS PERMITTED.
*
*              THE INITIAL VALUE OF THE MULTIPLIER IS USED THRUOUT.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)=1.,2.,3.,4.  IX(1...4)=1,2,3,4  U=0.0
*    USAGE   -        CALL MULPLY(X,4,2.,Y)
*                     CALL XMLPLY(IX,4,2,IY)
*                     CALL MULPLY(X,1,2.,Z)
*                     CALL MULPLY(X,0,2.,U)
*                     CALL MULPLY(X,4,X(3),X)
*    OUTPUTS - Y(1...4)=2.,4.,6.,8.  IY(1...4)=2,4,6,8
*              Z=2.  U=0.0 (NO OUTPUT CASE)  X(1...4)=3.,6.,9.,12.
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR      0                  XR4
```

```
       BCI    1,MULPLY
* PRINCIPAL ENTRY.  MULPLY(X,LX,XMLPLR,XMLPLD)
MULPLY CLA    FMP
       LDQ    NOP
 SETUP STO    MLPLY
       STQ    VARY
       SXD    MULPLY-2,4
 K1    CLA    1,4
       ADD    K1                 A(X)+1
       STA    GET
       CLA    4,4
       ADD    K1                 A(XMLPLD)+1
       STA    STORE
       CLA*   3,4                XMLPLR
       STO    TEMP
       CLA*   2,4                LX
       TMI    LEAVE
       PDX    0,4
       TXL    LEAVE,4,0
* MULTIPLICATION LOOP
 GET   LDQ    **,4               **=A(X)+1
 MLPLY NOP                       =FMP  TEMP  OR MPY  TEMP
 VARY  NOP                       = NOP  OR ALS  17
 STORE STO    **,4               **=A(XMLPLD)+1
       TIX    GET,4,1
* EXIT
 LEAVE LXD    MULPLY-2,4
       TRA    5,4
* SECOND ENTRY.  XMLPLY(IX,LIX,IXMPLR,IXMPLD)
XMLPLY CLA    MPY
       LDQ    ALS
       TRA    SETUP
* CONSTANTS, VARIABLES
 FMP   FMP    TEMP
 NOP   NOP
 MPY   MPY    TEMP
 ALS   ALS    17
 TEMP  PZE    **,**,**           MULTIPLIER
       END
```

```
*      MUVADD (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0244
*      FAP
*MUVADD
       COUNT   200
       LBL     MUVADD
       ENTRY   MUVADD (IV,ILO,IHI,LADD,MUVSUM,NSUMS,IANS)
*
*                  ----ABSTRACT----
*
* TITLE - MUVADD
*       FAST MOVING SUMMATION OF A FIXED POINT VECTOR
*
*             MUVADD MAKES A MOVING SUMMATION (OVER A SPECIFIED SUMMING
*             LENGTH) OF A FIXED POINT FORTRAN VECTOR WITHIN A
*             SPECIFIED RANGE OF THE VECTOR.  OVERFLOW CHECK IS MADE.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 129 REGISTERS
* SPEED      - FOR VECTORS LONG WITH RESPECT TO SUMMING LENGTH TIME
*                IS LENGTH OF RANGE TIMES 10 MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON JR, MAY 1962
*
*                  ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*    CALL MUVADD(IV,ILO,IHI,LADD,MUVSUM,NSUMS,IANS)
*
* INPUTS
*
*    IV(I)     I=ILO...IHI IS THE SPECIFIED VECTOR RANGE.
*
*    ILO       MUST EXCEED 0
*
*    IHI       MUST EQUAL OR EXCEED ILO
*
*    LADD      IS THE SUMMING LENGTH.  IT MUST EXCEED 0.
*
* OUTPUTS
*
*    MUVSUM(I) I=1,2,...,NSUMS CONTAINS THE MOVING SUMS
*              WHERE
*                   MUVSUM(1)=IV(ILO)+IV(ILO+1)+...+IV(ILO+LADD-1)
*                   MUVSUM(2)=IV(ILO+1)+IV(ILO+2)+...+IV(ILO+LADD)
*                               ETC.
*                   MUVSUM(NSUMS)=IV(IHI-LADD+1)+...+IV(IHI-1)+IV(IHI)
*                              NOTE - SEE EXCEPTION BELOW UNDER IANS.
*
*    NSUMS     = IHI-ILO+2-LADD OR ONE, WHICHEVER IS LARGER.
*
*    IANS      = 0 MEANS JOB IS DONE
*              = 1 MEANS LADD EXCEEDED LENGTH OF RANGE.  IN THIS CASE
*                  NSUMS IS SET =1.
*              =-1 MEANS ILLEGAL SPECIFICATION OF ILO, IHI, OR LADD
*                  (NSUMS WILL =0).
*              =-2 MEANS OVERFLOW OCCURRED BUT ALL SUMS COMPUTED.
*
* EXAMPLES
*
* 1. INPUTS  - IV(1...10)=1,2,4,8,16,32,10,9,8,7  ILO=2, IHI=8, LADD=2
*    OUTPUTS - IANS=0, NSUMS=6, MUVSUM(1...6)=6,12,24,48,42,19
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LADD = 1
*    OUTPUTS - IANS=0, NSUMS=7, MUVSUM(1...7)=2,4,8,16,32,10,9
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LADD=12
*    OUTPUTS - IANS=1, NSUMS=1, MUVSUM(1)=81
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=2
*    OUTPUTS - IANS=1,NSUMS=1,MUVSUM(1)=2
*
* 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=2 AND LADD=1
```

```
*    OUTPUTS - IANS=0, NSUMS=1, MUVSUM(1)=2
*
* 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LADD=7
*    OUTPUTS - IANS=0, NSUMS=1, MUVSUM(1)=81
*
* 7. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IV(2)=131068  (THIS PRODUCES
*              OVERFLOW ON FIRST SUM ONLY SINCE 131068+4=2**17)
*    OUTPUTS - IANS=-2, NSUMS=6, MUVSUM(1...6)=0,12,24,48,42,19
*
* 8. EXAMPLES 8., 9., AND 10. TEST ILLEGALITY CHECKS.
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT ILO=0
*    OUTPUTS - IANS=-1, NSUMS=0
*
* 9. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=1
*    OUTPUTS - IANS=-1, NSUMS=0
*
*10. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LADD=-50
*    OUTPUTS - IANS=-1, NSUMS=0
*
       HTR     0
       HTR     0
       HTR     0
       BCI     1,MUVADD
MUVADD SXD     MUVADD-4,1
       SXD     MUVADD-3,2
       SXD     MUVADD-2,4
* ADDRESS SETUP.
       CLA     1,4
       STA     IV
       CLA     2,4
       STA     GET2
       CLA     3,4
       STA     GET3
       CLA     4,4
       STA     GET4
       CLA     5,4
       STA     MUVSUM
       CLA     6,4
       STA     PUT6
       CLA     7,4
       STA     PUT7
* INPUT CHANNEL FOR ILO, IHI, LADD
  GET2 CLA     **            A(ILO)
       ARS     18
       STO     ILO
  GET3 CLA     **            A(IHI)
       ARS     18
       STO     IHI
  GET4 CLA     **            A(LADD)
       ARS     18
       STO     LADD
* CHECK ILO, IHI, LADD
   CK2 CLA     ILO
       CAS     K0
       TRA     CK3
       NOP                   ILO MUST
       TRA     ILEGL         EXCEED 0
   CK3 CLA     IHI
       CAS     ILO
       NOP
       TRA     CK4           IHI MUST = OR EXCEED ILO.
       TRA     ILEGL
   CK4 CLA     LADD
       CAS     K0
       TRA     STNSM
       NOP                   LADD MUST
       TRA     ILEGL         EXCEED 0
* NOW SET NSUMS AND CHECK IF ZERO OR NEGATIVE, WHICH MEANS LADD IS
* LONGER THAN RANGE.
 STNSM CLA     IHI
       SUB     ILO
       ADD     K2
       SUB     LADD
       STO     NSUMS
       CAS     K0
```

```
       TRA    NORML
       NOP
       TRA    SHORT
* IF RANGE SHORT SET LFRST=LENGTH OF RANGE AND NSUMS=1 AND NMORE=0 AND
* IANS=1.
 SHORT CLA    K1
       STO    NSUMS
       STO    IANS
       ADD    IHI
       SUB    ILO
       STO    LFRST
       STZ    NMORE
       TRA    SETUP
* NORMALLY LFRST=LADD, NMORE=NSUMS-1, IANS=0
 NORML SUB    K1
       STO    NMORE
       CLA    LADD
       STO    LFRST
       STZ    IANS
       TRA    SETUP
* NOW SETUP THE TWO LOOPS AND THEN GO TO FIRST ONE AFTER TURNING OFF
* OVERFLOW IF ON.
 SETUP CLA    IV
       SUB    ILO
       ADD    K2
       STA    L1ADD
       STA    L2SUB
       SUB    LADD
       STA    L2ADD
       CLA    MUVSUM
       STA    L1STO
       STA    L2STO
       CLA    NMORE
       ALS    18
       STD    TXL
       STD    L2TXL
       TOV    L1
       TRA    L1
* FIRST LOOP FORMS FIRST SUM.
    L1 LXA    LFRST,1
       CLA    K0
 L1ADD ADD    **,1            A(IV)-ILO+2
       TIX    L1ADD,1,1
* STORE FIRST SUM
 L1STO STO    **              A(MUVSUM)
* THEN CHECK IF MORE SUMS ARE TO BE DONE (KEEP FIRST IN AC).
* IF NOT GO CHECK FOR OVERFLOW AND LEAVE.
       LXA    K1,1
   TXL TXL    L2ADD,1,**      **=NMORE
       TRA    CKOV
* SECOND LOOP FORMS REST OF SUMS BY ADDING ONE, SUBTRACTING ONE.
 L2ADD ADD    **,1            A(IV)-ILO-LADD+2
 L2SUB SUB    **,1            A(IV)-ILO+2
 L2STO STO    **,1            A(MUVSUM)
       TXI    *+1,1,1
 L2TXL TXL    L2ADD,1,**      (NMORE)
* WHEN DONE, GO CHECK OVERFLOW AND LEAVE.
       TRA    CKOV
* EXIT FOR ILLEGAL INPUTS.
 ILEGL CLS    K1
       STO    IANS
       STZ    NSUMS
       TRA    LEAVE
* CHECK FOR OVERFLOW BEFORE LEAVING.
  CKOV TOV    OVSET
       TRA    LEAVE
 OVSET CLS    K2
       STO    IANS
       TRA    LEAVE
* STORE IANS, NSUMS, AND EXIT.
 LEAVE CLA    IANS
       ALS    18
  PUT7 STO    **              A(IANS)
       CLA    NSUMS
       ALS    18
```

```
      PUT6 STO      **           A(NSUMS)
      EXIT LXD      MUVADD-4,1
           LXD      MUVADD-3,2
           LXD      MUVADD-2,4
           TRA      8,4
    * CONSTANTS
        K0 PZE      0
        K1 PZE      1
        K2 PZE      2
    * VARIABLES
       ILO PZE      **           MUST BE
       IHI PZE      **           MOVED
      LADD PZE      **           FROM DECREMENTS
     NSUMS PZE      **           MUST BE MOVED
      IANS PZE      **           TO DECREMENTS.
        IV PZE      **           **=A(IV)
    MUVSUM PZE      **           **=A(MUVSUM)
     NMORE PZE      **           NSUMS-1
     LFRST PZE      **           IS NO. OF ELEMENTS IN FIRST SUM.
           END
```

```
*       MVBLOK (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0082
*       FAP
*MVBLOK
        COUNT   75
        LBL     MVBLOK
        ENTRY   MVBLOK (NN,ISORCE,IDEST)
*
*                    ----ABSTRACT----
*
* TITLE - MVBLOK
*         MOVE DATA BLOCK
*
*              MVBLOK MOVES A DATA SERIES FROM ONE AREA IN CORE TO
*              ANOTHER AREA.  THE TWO AREAS MAY NOT OVERLAP, UNLESS
*              THE SOURCE AREA HAS A HIGHER CORE ADDRESS THAN THE
*              DESTINATION AREA.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 19 REGISTERS
* SPEED     - 28 + 6N MACHINE CYCLES WHERE N=LENGTH OF DATA SERIES.
* AUTHOR    - S.M. SIMPSON,  NOVEMBER, 1961
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL MVBLOK(NN,ISORCE,IDEST)
*
* INPUTS
*
*    NN         IS THE LENGTH OF THE DATA BLOCK.
*               IS FORTRAN II INTEGER.
*               MUST BE GRTHN=1.
*
*    ISORCE     IS THE CORE ADDRESS OF THE SOURCE DATA BLOCK.
*               IS FORTRAN II INTEGER.
*
*    IDEST      IS THE CORE ADDRESS OF THE DESTINATION DATA BLOCK.
*               IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*               THE CONTENTS OF ISORCE THRU ISORCE-N+1 REPLACES THE
*                   CONTENTS OF IDEST THRU IDEST-N+1.
*
* EXAMPLES
*
*               LET SORCE AND DEST BE THE TWO DATA AREAS,  THEN THE
*               PROGRAMMING SEQUENCE
*
*                     ISORCE = XLOCF(SORCE)
*                     IDEST  = XLOCF(DEST)
*                     CALL MVBLOK (NN, ISORCE, IDEST)
*
*               IS EQUIVALENT TO
*
*                     DO 10 I=1,NN
*                     J = NN-I+1
*                10   DEST(J) = SORCE(J)
*
        HTR     0
        BCI     1,MVBLOK
MVBLOK  SXD     *-2,4
        CLA*    2,4
        ARS     18
        ADD     K1            ISRCE+1
        STA     MOV
        CLA*    3,4
        ARS     18
        ADD     K1            IDST+1
        STA     MOV+1
        CLA*    1,4           N
```

```
        PDX    0,4            TO XR4
  MOV   CLA    **,4           **=ISRCE+1
        STO    **,4           **=IDST+1
        TIX    MOV,4,1
        LXD    MVBLOK-2,4
        TRA    4,4
  K1    PZE    1
        END
```

```
*     MVINAV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0115
*     LABEL
CMVINAV
      SUBROUTINE MVINAV (REC,LREC,K,RECAV,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - MVINAV
C       MOVING AVERAGE OF A VECTOR
C
C              MVINAV FINDS THE MOVING AVERAGE, RECAV(I), OF
C              A FLOATING POINT VECTOR, REC(I) I=1,...,LREC, ACCORDING
C              TO THE EQUATION
C
C                                    1     I+K
C                  RECAV(I)  =  -----  SUM  ( REC (J) )
C                               2K+1  J=I-K
C
C                    FOR I = 1,2,...,LREC
C                    WHERE K AND LREC ARE INPUT PARAMETERS,
C                      AND THE COMPUTATIONS ARE MADE AS THOUGH REC(J)
C                      WERE ZERO FOR J LESS THAN 1 AND GREATER THAN LREC
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 221 REGISTERS
C SPEED     - 65*LREC MACHINE CYCLES FOR LARGE LREC
C AUTHOR    - S.M. SIMPSON, MARCH 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MVINAV(REC,LREC,K,RECAV,IANS)
C
C INPUTS
C
C    REC(I)     I=1,...,LREC IS A FLOATING POINT VECTOR
C
C    LREC       MUST EXCEED ZERO
C
C    K          SPECIFIES THE AVERAGING LENGTH AS 2K+1 POINTS
C               MUST BE NON-NEGATIVE, AND
C                    2*K+1 MUST BE LESS THAN LREC  (UNLESS K=0)
C
C OUTPUTS
C
C    RECAV(I)  I=1,...,LREC  IS THE MOVING AVERAGE GIVEN IN ABSTRACT
C
C    IANS       =  0  NORMALLY
C               = -2  FOR ILLEGAL LREC (NO OTHER OUTPUT IN THIS CASE)
C               = -3  FOR ILLEGAL K    (NO OTHER OUTPUT IN THIS CASE)
C
C
C EXAMPLES
C
C 1. INPUTS  - REC(1...6)  = 9.,9.,0.,36.,36.,9.
C               LREC=6   K1=0   K2=1   K3=2
C    USAGE   -          CALL MVINAV(REC,LREC,K1,RECAV1,IANS1)
C                       CALL MVINAV(REC,LREC,K2,RECAV2,IANS2)
C                       CALL MVINAV(REC,LREC,K3,RECAV3,IANS3)
C    OUTPUTS - IANS1=0  RECAV1(1...6) = 9.,9.,0.,36.,36.,9.
C              IANS2=0  RECAV2(1...6) = 6.,6.,15.,24.,27.,15.
C              IANS3=0  RECAV3(1...6) = 3.6,10.8,18.,18.,16.2,16.2
C 2. ILLEGAL CASES
C    USAGE   -          CALL MVINAV(REC,0,0,RECAV,IANS1)
C                       CALL MVINAV(REC,3,-1,RECAV,IANS2)
C                       CALL MVINAV(REC,7,3,RECAV,IANS3)
C    OUTPUTS - IANS1 = -2  (ILLEGAL LREC)
C              IANS2 = IANS3 = -3  (ILLEGAL K)
C
C DUMMY DIMENSIONS
      DIMENSION REC(2),RECAV(2)
```

```
C CHECK INPUTS.  LREC GRTHN 0,  K GRTHN=0,  2*K+1  LSTHN LREC
      IANS=-2
      IF(LREC)  9999,9999,10
   10 IANS=-3
      IF(K)  9999,30,20
   20 IF(2*K+1-LREC)  40,9999,9999
C SPECIAL TREATMENT FOR K=0
   30 IANS=0
      DO 35  I=1,LREC
   35 RECAV(I)=REC(I)
      GO TO 9999
C TREAT LEFT EDGE EFFECT - SET RECAV(1...K+1)
C FIRST SET RECAV(1)
   40 IANS=0
      LEND=K+1
      RECAV(1)=0.
      DO 50  I=1,LEND
   50 RECAV(1)=RECAV(1)+REC(I)
C THEN SET RECAV(2...K+1)
      DO 60  I=2,LEND
      IADD=I+K
   60 RECAV(I)=RECAV(I-1)+REC(IADD)
C NOW TREAT CENTRAL TERMS - SET RECAV(K+2,...,LREC-K)
      IXLO=LEND+1
      IXHI=LREC-K
      DO 70  I=IXLO,IXHI
      IADD=I+K
      ISUB=I-LEND
   70 RECAV(I)=RECAV(I-1)+REC(IADD)-REC(ISUB)
C NEXT TREAT RIGHT EDGE EFFECT - SET RECAV (LREC-K+1,...,LREC)
      IXLO=LREC-K+1
      DO 80  I=IXLO,LREC
      ISUB=I-LEND
   80 RECAV(I)=RECAV(I-1)-REC(ISUB)
C FINALLY AVERAGE THE RESULT
      SCALE=1./FLOATF(2*K+1)
      DO 90  I=1,LREC
   90 RECAV(I)=RECAV(I)*SCALE
C EXIT
 9999 RETURN
      END
```

```
*       MVNSUM (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0201
*       FAP
*MVNSUM
        COUNT   250
        LBL     MVNSUM
        ENTRY   MVNSUM (X, LX, LSUM, DVSR, SUMOVD, LSUMOD)
*
*
*                   ----ABSTRACT----
*
* TITLE - MVNSUM
*        MOVING SUMMATION WITH DIVISION BY A CONSTANT
*
*               MVNSUM  COMPUTES
*
*                            1  I+L-1
*                 S(I)  =  ---  SUM  X(J) ,  I = 1,2,...,N=LX-L+1
*                            D   J=I
*
*               GIVEN  X(1...LX),  LX,  L,  AND  D.
*
*               COMPUTATIONS ARE SPED UP FOR D = 1.0 .  THE OUTPUT VECTOR
*               MAY REPLACE THE INPUT VECTOR.  THE LENGTH  N  IS AN
*               ADDITIONAL OUTPUT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 71 REGISTERS
* SPEED      - ON THE 7090, MVNSUM TAKES
*                 78.6 + 8.4*L + 22.8*N  MACHINE CYCLES IF  D = 1.0
*                 74.6 + 8.4*L + 39.8*N  MACHINE CYCLES IF  D NOT= 1.0
*                 WHERE  L,  N  AND  D  ARE DEFINED ABOVE.
* AUTHOR     - S.M. SIMPSON,   JULY 1964
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*      AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL MVNSUM(X, LX, LSUM, DVSR, SUMOVD, LSUMOD)
*
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLOATING POINT VECTOR.
*
*    LX         MUST EXCEED ZERO.
*
*    LSUM       IS THE SUMMING LENGTH,  L  OF THE ABSTRACT.
*               MUST EXCEED ZERO AND BE LSTHN=  LX.
*
*    DVSR       IS THE DIVISOR,  D,  OF THE ABSTRACT.
*               MUST BE NON-ZERO.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LX,  LSUM,  OR  DVSR
*                  ILLEGAL.
*
*    SUMOVD(I) I=1...LSUMOD  IS THE MOVING SUM  S(1...N)  OF THE
*                  ABSTRACT.
*
*    LSUMOD     WILL  =  LX-LSUM+1 .
*
*
*
* EXAMPLES
*
* 1. UNITY DIVISOR CASES
*    INPUTS  - X(1...3) = 1.,2.,4.   DVSR = 1.0
*              S(1...3,1...3,1...3) = -9.,-9.,...
*              LS(1...3,1...3) = -9,-9,...
*    USAGE   -        DO 10  LX=1,3
*                     DO 10  LSUM=1,LX
```

```
*               10   CALL MVNSUM(X,LX,LSUM,DVSR,S(1,LSUM,LX),
*                  1                 LS(LSUM,LX))
*     OUTPUTS - S(1...3,1...3,1) = 1.,-9.,-9.,,-9.,-9.,-9.,,-9.,-9.,-9.
*               S(1...3,1...3,2) = 1., 2.,-9.,, 3.,-9.,-9.,,-9.,-9.,-9.
*               S(1...3,1...3,3) = 1., 2., 4.,, 3., 6.,-9.,, 7.,-9.,-9.
*               LS(1...3,1...3)  = 1,-9,-9,,2,1,-9,,3,2,1
*
*  2. NON-UNITY DIVISOR CASES
*     INPUTS  - SAME AS EXAMPLE 1. EXCEPT  DVSR = 0.5
*     USAGE   - SAME AS EXAMPLE 1.
*     OUTPUTS - SAME AS EXAMPLE 1. EXCEPT THAT ALL VALUES OF  S  WHICH DO
*                     NOT EQUAL -9. WILL BE DOUBLED.
*
*  3. CASE WHERE OUTPUT REPLACES INPUT
*     INPUTS  - SAME AS EXAMPLE 1.
*     USAGE   -        CALL MVNSUM(X,3,2,DVSR,X,LSUMOD)
*     OUTPUTS - X(1...3) = 3.,6.,4.  LSUMOD = 2
*
*  4. ILLEGAL USAGES
*     INPUTS  - SAME AS EXAMPLE 1.
*     USAGE   -        CALL MVNSUM(X,0, 2,1.0,S,LS)
*                      CALL MVNSUM(X,2,-1,1.0,S,LS)
*                      CALL MVNSUM(X,2, 3,1.0,S,LS)
*                      CALL MVNSUM(X,3, 2,0.0,S,LS)
*     OUTPUTS - S = -9.  LS = -9
*
*
*  PROGRAM FOLLOWS BELOW
*
*  NO TRANSFER VECTOR
*
       HTR      0                XR4
       BCI      1,MVNSUM
*
*  ONLY ENTRY.  MVNSUM(X, LX, LSUM, DVSR, SUMOVD, LSUMOD)
*
MVNSUM SXD      MVNSUM-2,4
*
*  CHECK OUT  LSUM, DVSR,  LX.  SET  LSUMOD.
*
       CLA*     3,4              LSUM
       ARS      18               IN ADDRESS
       TMI      LEAVE
       TZE      LEAVE
       STO      LSUM             STORED
       CLA*     4,4              DVSR
       STO      DVSR
       TZE      LEAVE
       FSB      K1L              SET SWITCH
       STO      ZFD1             FOR DVSR = 1.0
       CLA*     2,4              LX
       TMI      LEAVE
       TZE      LEAVE
       SUB*     3,4              LX-LSUM
       ADD      KD1              LX-LSUM+1 = LSUMOD
       TMI      LEAVE
       TZE      LEAVE
       STO*     6,4              LSUMOD STORED
       STD      TXL1
       STD      TXL2
*
*  THEN SET ADDRESSES AND DVSR
*
       CLA      1,4              A(X)
       ADD      K1               A(X)+1
       STA      FAD1
       STA      LDQ1
       STA      LDQ2
       ADD      K1               A(X)+2
       SUB      LSUM             A(X)+2-LSUM
       STA      FAD2
       STA      FAD3
       CLA      5,4              A(SUMOVD)
       ADD      K1               A(SUMOVD)+1
       STA      STO
```

```
        STA     STQ
*
* FORM  X(LSUM)+X(LSUM-1)+...+X(1),  THEN BRANCH TO PROPER LOOP
*
        LXA     LSUM,4          LSUM TO XR4
        PXD     0,0             ZERO TO AC
 FAD1   FAD     **,4            ** = A(X)+1
        TIX     FAD1,4,1
        ZET     ZFD1            (NOTE XR4 IS NOW = 1)
        TRA     LDQ2
        TRA     LDQ1
*
* LOOP WITHOUT DIVISION
*
 FAD2   FAD     **,4            ** = A(X)+2-LSUM
        FSB     LEND
 LDQ1   LDQ     **,4            ** = A(X)+1
        STQ     LEND            (SET ASIDE LEFT END ELEMENT)
 STO    STO     **,4            ** = A(SUMOVD)+1
        TXI     *+1,4,1
 TXL1   TXL     FAD2,4,**       ** = LX-LSUM+1 = LSUMOD
        TRA     LEAVE
*
* LOOP WITH DIVISION BY DVSR
*
 CLA    CLA     TEMP
 FAD3   FAD     **,4            ** = A(X)+2-LSUM
        FSB     LEND
 LDQ2   LDQ     **,4            ** = A(X)+1
        STQ     LEND
        STO     TEMP
        FDP     DVSR
 STQ    STQ     **,4            ** = A(SUMOVD)+1
        TXI     *+1,4,1
 TXL2   TXL     CLA,4,**        ** = LX-LSUM+1 = LSUMOD
*
* EXIT
*
 LEAVE LXD      MVNSUM-2,4
        TRA     7,4
*
* CONSTANTS, TEMPORARIES
*
 K1     PZE     1
 KD1    PZE     0,0,1
 K1L    DEC     1.0
 LSUM   PZE     **,0,0          ** = LSUM
 DVSR   PZE     **,**,**        INPUT DVSR
 TEMP   PZE     **,**,**
 LEND   PZE     **,**,**        X(1),X(2),...
 ZFD1   PZE     **,**,**        = 0.0 IF DVSR = 1.0, NOT = 0.0 OTHERWISE
        END
```

```
*      MVNTIN (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0233
*      FAP
*MVNTIN
       COUNT   250
       LBL     MVNTIN
       ENTRY   MVNTIN (X, LX, DEL, LINT,  XMI,  LXMI)
       ENTRY   MVNTNA (X, LX, DEL, LINT, XAMI, LXAMI)
*
*
*                    ----ABSTRACT----
*
* TITLE - MVNTIN, WITH SECONDARY ENTRY MVNTNA
*        MOVING TRAPEZOIDAL INTEGRAL OR ABSOLUTE VALUE INTEGRAL
*
*              MVNTIN  COMPUTES
*
*                                 X(I)   I+LINT-2         X(I+LINT-1)
*                    XMI(I) = DEL * (---- +   SUM    X(J) + ----------- )
*                                   2     J=I+1              2
*
*                    FOR  I = 1,2,...,LXMI=LX-LINT+1
*
*              GIVEN THE VECTOR  X(1...LX), THE LENGTH  LX,  THE
*              INCREMENT  DEL,  AND THE INTEGRATING LENGTH  LINT.  THE
*              LENGTH,  LXMI,  OF THE MOVING INTEGRAL IS AN ADDITIONAL
*              OUTPUT.  THE OUTPUT VECTOR MAY REPLACE THE INPUT VECTOR.
*
*              MVNTNA  COMPUTES THE SAME EXPRESSION AS DOES  MVNTIN
*              EXCEPT THAT THE MAGNITUDES OF THE  X(I)  VALUES ARE USED.
*
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 88 REGISTERS
* SPEED      - ON THE 7090 EITHER ENTRY TAKES ABOUT
*                 50 + 18.8*LINT + 52.6*LXMI  MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, AUGUST 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*        AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL MVNTIN(X, LX, DEL, LINT,  XMI,  LXMI)
*     CALL MVNTNA(X, LX, DEL, LINT, XAMI, LXAMI)
*
* INPUTS
*
*    X(I)      I=1...LX  IS THE VECTOR TO BE INTEGRATED.
*
*    LX        MUST BE GRTHN=  2 .
*
*    DEL       IS THE INCREMENT BETWEEN  X(I)  VALUES.  MAY BE NEGATIVE
*                 OR ZERO.
*
*    LINT      IS THE INTEGRATING LENGTH.
*              MUST BE GRTHN=  2  AND LSTHN=  LX.
*
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF  LX  OR  LINT
*              IS ILLEGAL.
*
*    XMI(I)    I=1...LXMI  IS OUTPUT FROM  MVNTIN  AS GIVEN IN ABSTRACT.
*                 EQUIVALENCE  (X,XMI)  IS PERMITTED.
*
*    LXMI      =  LX-LINT+1  IS OUTPUT FROM  MVNTIN.
*                 EQUIVALENCE  (LX,LXMI)  IS PERMITTED.
*
*    XAMI(I)   I=1...LXAMI  IS OUTPUT FROM  MVNTNA  AS GIVEN IN
*              ABSTRACT.  EQUIVALENCE  (LX,LXAMI)  IS PERMITTED.
*
*    LXAMI     =  LX-LINT+1  IS OUTPUT FROM MVNTNA.
```

```
*                 EQUIVALENCE  (LX,LXAMI)  IS PERMITTED.
*
* EXAMPLES
*
* 1. TESTING EXTREMAL VALUES OF  LX,  LINT  (INCLUDING ILLEGAL VALUES)
*    INPUTS  - X(1...4) = -1.,2.,-4.,8.      DEL = 2.0
*              XMI(1...3,1...4,1...4) = XAMI(1...3,1...4,1...4)
*                    = -99.,-99.,...
*              LXMI(1...4,1...4) = LXAMI(1...4,1...4) = -9,-9,...
*    USAGE   -        DO 10  LX=1,4
*                     DO 10  LI=1,4
*                     CALL MVNTIN(X,LX,DEL,LI, XMI(1,LI,LX), LXMI(LI,LX))
*               10    CALL MVNTNA(X,LX,DEL,LI,XAMI(1,LI,LX),LXAMI(LI,LX))
*    OUTPUTS - ALL  XMI  VALUES  =  -99.  AND ALL  LXMI  VALUES  =  -9
*              EXCEPT AS FOLLOWS.
*              XMI(1...1,2,2) =  1.0              LXMI(2,2) = 1
*              XMI(1...2,2,3) =  1.0,-2.0         LXMI(2,3) = 2
*              XMI(1...1,3,3) = -1.0              LXMI(3,3) = 1
*              XMI(1...3,2,4) =  1.0,-2.0,4.0     LXMI(2,4) = 3
*              XMI(1...2,3,4) = -1.0, 2.0         LXMI(3,4) = 2
*              XMI(1...1,4,4) =  3.0              LXMI(4,4) = 1
*
*              ALL  XAMI  VALUES  =  -99.  AND ALL  LXAMI  VALUES  =  -9
*              EXCEPT AS FOLLOWS.
*              XAMI(1...1,2,2) = 3.0              LXAMI(2,2) = 1
*              XAMI(1...2,2,3) = 3.0,6.0          LXAMI(2,3) = 2
*              XAMI(1...1,3,3) = 9.0              LXAMI(3,3) = 1
*              XAMI(1...3,2,4) = 3.0,6.0,12.0     LXAMI(2,4) = 3
*              XAMI(1...2,3,4) = 9.0,18.0         LXAMI(3,4) = 2
*              XAMI(1...1,4,4) =21.0              LXAMI(4,4) = 1
*
* 2. CASE WHERE OUTPUTS REPLACE INPUTS
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LX = 4
*    USAGE   -        CALL MVNTIN(X,LX,DEL,2,X,LX)
*    OUTPUTS - X(1...4) = 1.0,-2.0,4.0,8.         LX = 3
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR      0                XR4
       BCI      1,MVNTIN
*
* FIRST ENTRY.  MVNTIN(X, LX, DEL, LINT, XMI, LXMI)
*
MVNTIN CLA      FAD4             CHOOSE INSTRUCTIONS
       LDQ      FSB4             FOR SIGNED ADDITION, SUBTRACTION.
       TRA      MERGE
*
* SECOND ENTRY. MVNTNA(X,LX,DEL,LINT,XAMI,LXAMI)
*
MVNTNA CLA      FAM4             CHOOSE ABSOLUTE VALUE
       LDQ      FSM4             INSTRUCTIONS.
 MERGE SXD      MVNTIN-2,4
       STO      FAZ
       STO      FA1
       STO      FA2
       STQ      FS1
       STQ      FS2
*
* SET  DEL/2.0,  CHECK  LX,  LINT.
*
       CLA*     3,4              DEL
       FDP      K2L              DEL/2.0
       STQ      DELHAF
       CLA*     2,4              LX
       CAS      KD1
       TRA      LXOK             MUST EXCEED 1
       TRA      LEAVE
       TRA      LEAVE
  LXOK CLA*     4,4              LINT
       SUB      KD1              LINT-1
       TMI      LEAVE            CHECK FOR
       TZE      LEAVE            UNDERSIZED LINT
```

MVNTIN (PAGE 3)


```
        STD     TXL1
        SUB*    2,4             LINT-1-LX
        CHS                     LX-LINT+1 = LXMI
        TMI     LEAVE           CHECK FOR
        TZE     LEAVE           OVERSIZED LINT
*
* INPUTS OK.
*
        STO*    6,4             LXMI STORED
        STD     TXL3
        SUB     KD1             LXMI-1
        STD     TXL2
        CLA*    4,4             LINT
        ARS     18
        STO     LINT
        CLA     1,4             A(X)
        STA     FS2
        ADD     K1              A(X)+1
        STA     FAZ
        STA     FS1
        SUB     LINT            A(X)+1-LINT
        STA     FA2
        ADD     K1              A(X)+2-LINT
        STA     FA1
        CLA     5,4             A(XMI)
        ADD     K1              A(XMI)+1
        STA     STO
*
* START COMPUTATIONS BY FORMING
*               S(1)  =  X(1) + 2*X(2) + ... + 2*X(LINT-1) + X(LINT),
* AVOIDING MIDDLE TERMS IF  LINT = 2,  AND STORE IN  SLAST.
*
        AXT     1,4             XR4 = I = 1...LINT
        PXD     0,0
 FAZ    NOP                     FAD (FAM) **,4      ** = A(X)+1
        TXI     TXL1,4,1
 XEC    XEC     FAZ
        XEC     FAZ
        TXI     *+1,4,1
 TXL1   TXL     XEC,4,**        ** = LINT-1
        XEC     FAZ
        STO     SLAST
*
* MAIN LOOP FORMS S(I+1) = S(I)-X(I)-X(I+1)+X(I+LINT-1)+X(I+LINT)
*                        = SNEXT
*       SETS  XMI(I) = (DEL/2.0)*SLAST
*       SETS  SLAST  = SNEXT
* (NOTE FORMULA IS OK FOR  LINT = 2)
*
        AXT     0,4
        TRA     TXI2
 CLA    CLA     SLAST           (I  OF ABOVE FORMULA IS NOW IN XR4)
 FS1    NOP                     FSB (FSM) **,4      ** = A(X)+1
 FS2    NOP                     FSB (FSM) **,4      ** = A(X)
 FA1    NOP                     FAD (FAM) **,4      ** = A(X)+2-LINT
 FA2    NOP                     FAD (FAM) **,4      ** = A(X)+1-LINT
        STO     SNEXT
 LDQ    LDQ     SLAST
        FMP     DELHAF
 STO    STO     **,4            ** = A(XMI)+1
        CLA     SNEXT
        STO     SLAST
 TXI2   TXI     *+1,4,1         XR4 = 1,2,...,LXMI+1
 TXL2   TXL     CLA,4,**        ** = LXMI-1
 TXL3   TXL     LDQ,4,**        ** = LXMI      (AVOIDS FORMING S(LXMI+1))
*
* EXIT
*
 LEAVE LXD      MVNTIN-2,4
        TRA     7,4
*
* CONSTANTS, TEMPORARIES
*
 K1     PZE     1
 KD1    PZE     0,0,1
```

****
*   MVNTIN   *
****
(PAGE 4)

```
     K2L    DEC     2.0
     FAD4   FAD     **,4
     FSB4   FSB     **,4
     FAM4   FAM     **,4
     FSM4   FSM     **,4
     LINT   PZE     **,0,0          ** = LINT
     DELHAF PZE     **,**,**        DEL/2.0
     SNEXT  PZE     **,**,**        S(I+1)     I=1,2,...,LXMI-1
     SLAST  PZE     **,**,**        S(I)       I=1,2,...,LXMI
            END
```

```
*************************
*    MVNTNA             *
*************************
 REFER TO
     MVNTIN
```



```
*************************
*    MVNTNA             *
*************************
 REFER TO
     MVNTIN
```

```
*     MVSQAV (SUBROUTINE)                  9/29/64   LAST CARD IN DECK IS NO. 0115
*     LABEL
CMVSQAV
      SUBROUTINE MVSQAV (REC,LREC,K,RECAV,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - MVSQAV
C       MOVING MEAN SQUARE AVERAGE OF A VECTOR
C
C              MVSQAV FINDS THE MOVING SQUARE AVERAGE, RECAV(I), OF
C              A FLOATING POINT VECTOR, REC(I) I=1,...,LREC, ACCORDING
C              TO THE EQUATION
C
C                                   1     I+K              2
C                    RECAV(I)  =  -----  SUM  ( REC (J) )
C                                 2K+1  J=I-K
C
C                    FOR I = 1,2,...,LREC
C                    WHERE K AND LREC ARE INPUT PARAMETERS,
C                      AND THE COMPUTATIONS ARE MADE AS THOUGH REC(J)
C                      WERE ZERO FOR J LESS THAN 1 AND GREATER THAN LREC
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 236 REGISTERS
C SPEED     - 97*LREC MACHINE CYCLES FOR LARGE LREC
C AUTHOR    - S.M. SIMPSON, MARCH 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL MVSQAV(REC,LREC,K,RECAV,IANS)
C
C INPUTS
C
C    REC(I)    I=1,...,LREC IS A FLOATING POINT VECTOR
C
C    LREC      MUST EXCEED ZERO
C
C    K         SPECIFIES THE AVERAGING LENGTH AS 2K+1 POINTS
C              MUST BE NON-NEGATIVE, AND
C                   2*K+1 MUST BE LESS THAN LREC  (UNLESS K=0)
C
C OUTPUTS
C
C    RECAV(I)  I=1,...,LREC  IS THE MOVING AVERAGE GIVEN IN ABSTRACT
C
C    IANS      =  0  NORMALLY
C              = -2  FOR ILLEGAL LREC (NO OTHER OUTPUT IN THIS CASE)
C              = -3  FOR ILLEGAL K    (NO OTHER OUTPUT IN THIS CASE)
C
C
C EXAMPLES
C
C 1. INPUTS  - REC(1...6)  = 3.,-3.,0.,6.,-6.,3.
C              LREC=6   K1=0   K2=1   K3=2
C    USAGE   -        CALL MVSQAV(REC,LREC,K1,RECAV1,IANS1)
C                     CALL MVSQAV(REC,LREC,K2,RECAV2,IANS2)
C                     CALL MVSQAV(REC,LREC,K3,RECAV3,IANS3)
C    OUTPUTS - IANS1=0  RECAV1(1...6) = 9.,9.,0.,36.,36.,9.
C              IANS2=0  RECAV2(1...6) = 6.,6.,15.,24.,27.,15.
C              IANS3=0  RECAV3(1...6) = 3.6,10.8,18.,18.,16.2,16.2
C 2. ILLEGAL CASES
C    USAGE   -        CALL MVSQAV(REC,0,0,RECAV,IANS1)
C                     CALL MVSQAV(REC,3,-1,RECAV,IANS2)
C                     CALL MVSQAV(REC,7,3,RECAV,IANS3)
C    OUTPUTS - IANS1 = -2  (ILLEGAL LREC)
C              IANS2 = IANS3 = -3  (ILLEGAL K)
C
C DUMMY DIMENSIONS
      DIMENSION REC(2),RECAV(2)
```

```
C CHECK INPUTS.  LREC GRTHN 0,  K GRTHN=0,  2*K+1  LSTHN LREC
      IANS=-2
      IF(LREC)  9999,9999,10
 10   IANS=-3
      IF(K)  9999,30,20
 20   IF(2*K+1-LREC)  40,9999,9999
C SPECIAL TREATMENT FOR K=0
 30   IANS=0
      DO 35  I=1,LREC
 35   RECAV(I)=REC(I)*REC(I)
      GO TO 9999
C TREAT LEFT EDGE EFFECT - SET RECAV(1...K+1)
C FIRST SET RECAV(1)
 40   IANS=0
      LEND=K+1
      RECAV(1)=0.
      DO 50  I=1,LEND
 50   RECAV(1)=RECAV(1)+REC(I)*REC(I)
C THEN SET RECAV(2...K+1)
      DO 60  I=2,LEND
      IADD=I+K
 60   RECAV(I)=RECAV(I-1)+REC(IADD)*REC(IADD)
C NOW TREAT CENTRAL TERMS - SET RECAV(K+2,...,LREC-K)
      IXLO=LEND+1
      IXHI=LREC-K
      DO 70  I=IXLO,IXHI
      IADD=I+K
      ISUB=I-LEND
 70   RECAV(I)=RECAV(I-1)+REC(IADD)*REC(IADD)-REC(ISUB)*REC(ISUB)
C NEXT TREAT RIGHT EDGE EFFECT - SET RECAV (LREC-K+1,...,LREC)
      IXLO=LREC-K+1
      DO 80  I=IXLO,LREC
      ISUB=I-LEND
 80   RECAV(I)=RECAV(I-1)-REC(ISUB)*REC(ISUB)
C FINALLY AVERAGE THE RESULT
      SCALE=1./FLOATF(2*K+1)
      DO 90  I=1,LREC
 90   RECAV(I)=RECAV(I)*SCALE
C EXIT
 9999 RETURN
      END
```

```
*     MXRARE (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0249
*     LABEL
CMXRARE
      SUBROUTINE MXRARE(DN,DD,LD,DNFRAC,DDFRAC,MNREWI,RAMX,ILO,IHI,IANS)
C
C                    -----ABSTRACT----
C
C TITLE - MXRARE
C       FINDS REGION TO MAXIMIZE RATIO OF TWO DISTRIBUTION FUNCTIONS
C
C             MXRARE FINDS A REGION (SUBJECT TO CONSTRAINTS), IN TERMS
C             OF THE INDICES ILO AND IHI, WHICH MAXIMIZES THE
C             FOLLOWING RATIO
C
C                                  DN(IHI) - DN(ILO)
C                    RATIO   =  --------------------
C                                  DD(IHI) - DD(ILO)
C
C               WHERE
C                   DN(1...LD)  IS ANY DISTRIBUTION FUNCTION
C                   DD(1...LD)  IS ANY OTHER DISTRIBUTION FUNCTION
C
C                        AND
C                         BOTH DISTRIBUTION FUNCTIONS MUST SATISFY
C                            1)  D(I+1)  EQUALS OR EXCEEDS D(I)
C                            2)  D(LD)   EXCEEDS  D(1)
C                         THE LENGTH LD IS ARBITRARY
C                   IHI-ILO  , THE WIDTH OF THE MAXIMIZING REGION, IS
C                         CONSTRAINED BY THE USER IN THREE WAYS
C
C                                    DN(IHI)-DN(ILO)
C                             1)  --------------- MUST BE GRTHN= DNFRAC
C                                     DN(LD)-DN(1)
C                           AND
C                                    DD(IHI)-DD(ILO)
C                             2)  --------------- MUST BE GRTHN= DDFRAC
C                                     DD(LD)-DD(1)
C                           AND
C                             3)  IHI-ILO         MUST BE GRTHN= MNREWI
C                         WHERE DNFRAC, DDFRAC, AND MNREWI ARE INPUTS
C
C             IF ZERO DENOMINATORS OCCUR THEY ARE TREATED AS FOLLOWS-
C                  0/0  IS TAKEN TO HAVE VALUE ZERO, AND A FLAG IS SET.
C                                                                   35
C                  K/0  WITH K GRTHN 0, IS TAKEN TO HAVE VALUE 10
C                       AND IS CHOSEN AS THE MAXIMUM RATIO.  (A FLAG IS
C                       ALSO SET IN THIS CASE.)
C
C
C             IF SEVERAL REGIONS HAVE THE SAME MAXIMUM RATIO THE ONE
C             WITH MINIMUM ILO IS CHOSEN (IF SEVERAL HAVE THE SAME
C             MINIMUM ILO, THEN THE ONE OF THIS SUBSET WITH MINIMUM
C             IHI IS CHOSEN)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 302 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON, MARCH 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - EXP(2
C
C FORTRAN USAGE
C     CALL MXRARE(DN,DD,LD,DNFRAC,DDFRAC,MNREWI,RAMX,ILO,IHI,IANS)
C
C INPUTS
C
C    DN(I)      I=1...LD  IS THE NUMERATOR DISTRIBUTION FUNCTION
C
C    DD(I)      I=1...LD  IS THE DENOMINATOR DISTRIBUTION FUNCTION
C                 (SEE ABSTRACT FOR CONDITIONS ON DN(I) AND DD(I).)
C
```

```
C    LD         MUST EXCEED 1
C
C    DNFRAC     IS THE CONSTRAINT ON DN(IHI)-DN(ILO)  (SEE ABSTRACT).
C               MUST BE GRTHN= 0. AND LSTHN= 1.
C
C    DDFRAC     IS THE CONSTRAINT ON DD(IHI)-DD(ILO)  (SEE ABSTRACT).
C               MUST BE GRTHN= 0. AND LSTHN= 1.
C
C    MNREWI     IS THE CONSTRAINT ON IHI-ILO          (SEE ABSTRACT).
C               MUST EXCEED ZERO AND BE LESS THAN LD
C
C OUTPUTS
C
C    RAMX       IS THE MAXIMUM VALUE FOUND FOR RATIO
C
C    ILO        IS THE LOW INDEX OF THE MAXIMIZING REGION
C
C    IHI        IS THE HIGH INDEX OF THE MAXIMIZING REGION
C
C    IANS       =  0  NORMALLY
C               = -1  FOR ILLEGAL DN      (NO OTHER OUTPUT IN THIS CASE)
C               = -2  FOR ILLEGAL DD             (DITTO)
C               = -3  FOR ILLEGAL LD             (DITTO)
C               = -4  FOR ILLEGAL DNFRAC         (DITTO)
C               = -5  FOR ILLEGAL DDFRAC         (DITTO)
C               = -6  FOR ILLEGAL MNREWI         (DITTO)
C               =  1  IF A  0/0 RATIO WAS FOUND
C               =  2  IF A  K/0 RATIO WAS FOUND (SUPERSEDES IANS=1 CASE)
C EXAMPLES
C
C 1. BEHAVIOUR WITH REGION UNCONSTRAINED (IN THIS CASE IHI-ILO WILL
C           ALWAYS COME OUT = 1)
C    INPUTS  - DN(1...10) = 1.,8.,16.,26.,36.,37.,38.,58.,59.,74.
C              DD(1...10) = -1.,0.,1.,2.,3.,4.,5.,6.,7.,8.
C              LD = 10   DNFRAC = 0.  DDFRAC = 0.  MNREWI = 1
C    OUTPUTS - IANS = 0     RAMX = 20.    ILO = 7    IHI = 8
C
C 2. BEHAVIOUR WITH REGION WIDTH CONSTRAINED TO BE 1,2,...,6
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT MNREWI IS SET IN USAGE
C    USAGE   -       DIMENSION  RAMX(6),ILO(6),IHI(6),IANS(6)
C                    DO 10    IR=1,6
C                10  CALL MXRARE(DN,DD,LD,DNFRAC,DDFRAC,IR,
C                   1                RAMX(IR),ILO(IR),IHI(IR),IANS(IR))
C    OUTPUTS - IANS(1...6) =      0,    0,    0,    0,    0,    0
C              RAMX(1...6) =   20.0, 12.0, 12.0, 9.25,8.400,8.333
C               ILO(1...6) =      7,    7,    7,    6,    3,    2
C               IHI(1...6) =      8,   10,   10,   10,    8,    8
C
C 3. BEHAVIOUR WITH CONSTRAINT ON NUMERATOR ONLY
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT DNFRAC(1...3) = .35,.70,.80
C    USAGE   -       DO 10  I=1,3
C                10  CALL MXRARE(DN,DD,LD,DNFRAC(I),DDFRAC,MNREWI,
C                   1                RAMX(I),ILO(I),IHI(I),IANS(I))
C    OUTPUTS - IANS(1...3) =      0,    0,    0
C              RAMX(1...3) =   12.0,8.286,8.250
C               ILO(1...3) =      7,    3,    2
C               IHI(1...3) =     10,   10,   10
C
C 4. BEHAVIOUR WITH CONSTRAINT ON DENOMINATOR ONLY
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT DDFRAC (1..3) = .25,.70,.80
C    USAGE   -       DO 10   I=1,3
C                10  CALL MXRARE(DN,DD,LD,DNFRAC,DDFRAC(I),MNREWI,
C                   1                RAMX(I),ILO(I),IHI(I),IANS(I))
C    OUTPUTS - SAME AS EXAMPLE 3.
C
C 5. CASES INVOLVING ZERO/ZERO RATIO
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT DD(1)=0.  AND  DN(1) = 8.
C                    AND MNREWI IS SET IN USAGE
C    USAGE   -        CALL MXRARE(DN,DD,LD,DNFRAC,DDFRAC,1,RAMX1,
C                   1                ILO1,IHI1,IANS1)
C                     CALL MXRARE(DN,DD,LD,DNFRAC,DDFRAC,2,RAMX2,
C                                    ILO2,IHI2,IANS2)
C    OUTPUTS - IANS1=  1    RAMX1=20.0    ILO1=7      IHI1=8
C              IANS2=  0    RAMX2=12.0    ILO2=7      IHI2=10
C
```

```
C 6. CASES INVOLVING K/O RATIO
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT DD(1) = 0.
C                  AND MNREWI IS SET IN USAGE
C    USAGE   - SAME AS EXAMPLE  5.
C    OUTPUTS - IANS1= 2    RAMX1=10**35  ILO1=1     IHI1=2
C              IANS2= 0    RAMX2=15.0    ILO2=1     IHI2=3
C
C 7. ILLEGAL CASES
C    INPUTS  - SAME AS EXAMPLE 1 EXCEPT AS MODIFIED IN USAGE
C                     CALL MXRARE(DN,DD,1,0.,0.,1,RAMX,ILO,IHI,IANS1)
C                     CALL MXRARE(DN,DD,2,2.,0.,1,RAMX,ILO,IHI,IANS2)
C                     CALL MXRARE(DN,DD,2,0.,-1.,1,RAMX,ILO,IHI,IANS3)
C                     CALL MXRARE(DN,DD,2,0.,0.,2,RAMX,ILO,IHI,IANS4)
C                     DN(1) = 8.
C                     CALL MXRARE(DN,DD,2,0.,0.,1,RAMX,ILO,IHI,IANS5)
C                     DD(2) = -2.
C                     CALL MXRARE(DN,DD,3,0.,0.,1,RAMX,ILO,IHI,IANS6)
C    OUTPUTS - IANS1= -3  IANS2= -4  IANS3= -5  IANS4= -6
C              IANS5= -1  IANS6= -2
C
C PROGRAM FOLLOWS BELOW
C DUMMY DIMENSIONS
      DIMENSION DN(2),DD(2)
C CHECK INPUTS IN THE ORDER LD, (DN,DD), DNFRAC,DDFRAC,MNREWI
C
      IANS=-3
      IF(LD-1)  9999,9999,10
 10   DNTOTL=DN(LD)-DN(1)
      DDTOTL=DD(LD)-DD(1)
      IF(DNTOTL)  30,30,15
 15   IF(DDTOTL)  35,35,20
 20   DO 25  I=2,LD
      IF(DN(I)-DN(I-1))  30,23,23
 23   IF(DD(I)-DD(I-1))  35,25,25
 25   CONTINUE
      GO TO 40
 30   IANS=-1
      GO TO 9999
 35   IANS=-2
      GO TO 9999
 40   IANS=-4
      IF(DNFRAC)    9999,45,45
 45   IF(DNFRAC-1.0)  50,50,9999
 50   IANS=-5
      IF(DDFRAC)  9999,55,55
 55   IF(DDFRAC-1.0)  60,60,9999
 60   IANS=-6
      IF(MNREWI)  9999,9999,65
 65   IF(MNREWI-LD)  70,9999,9999
C ALL OK
C IANS WILL BE ZERO NOW UNLESS SPECIAL CASES ENCOUNTERED
  70  IANS =0
      ILOT=0
      DNAMNT=DNFRAC*DNTOTL
      DDAMNT=DDFRAC*DDTOTL
      RAMX=0.
C START NEW LOW INDEX LOOP BY INCREASING ILOT BY 1, AND SETTING
C IHIT=ILOT+MNREWI THEN CHECK FOR COMPLETION UNDER EACH OF THE
C THREE CONSTRAINTS
 100  ILOT=ILOT+1
      IHIT=ILOT+MNREWI
      DNILOT=DN(ILOT)
      DDILOT=DD(ILOT)
C CHECK FOR COMPLETION WHEN HIGH INDEX RUNS OFF
 110  IF(IHIT-LD)  130,130,9999
C IF INDEX OK, CHECK NUMERATOR AND DENOMINATOR CONDITIONS
 130  IF(DN(IHIT)-DNILOT-DNAMNT)  150,140,140
 140  IF(DD(IHIT)-DDILOT-DDAMNT)  150,170,170
C IF CONDITIONS ON NUM AND DENOM NOT MET INCREASE IHIT BY 1
C AND GO RECHECK INDEX
 150  IHIT=IHIT+1
      GO TO 110
C IF ALL CONSTRAINTS SATISFIED, LOOP ON HIGH INDEX
 170  DO 250  IXHI=IHIT,LD
C CHECK FOR ZEROES
```

****
*   MXRARE   *
****
(PAGE 4)

```
      TOP=DN(IXHI)-DNILOT
      BOT=DD(IXHI)-DDILOT
      IF(BOT)  180,180,220
 180  IF(TOP)  190,190,200
C SPECIAL IANS SETTING FOR 0/0 CASE
 190  IANS=1
      GO TO 250
C SPECIAL EXIT FOR K/0 CASE
 200  IANS=2
      RAMX=10.**35
      ILO=ILOT
      IHI=IXHI
      GO TO 9999
C CHECK RATIO FOR BOT NOT ZERO
 220  IF (TOP/BOT-RAMX)  250,250,230
C RESET TRIAL RATIO AND INDICES
 230  RAMX=TOP/BOT
      ILO=ILOT
      IHI=IXHI
 250  CONTINUE
C WHEN FALL THRU, GO BACK FOR NEXT ILOT
      GO TO 100
C EXIT
 9999 RETURN
      END
```

```
***********************
*    NEXCOS           *
***********************
 REFER TO
     SEQSAC
```


```
***********************
*    NEXCOS           *
***********************
 REFER TO
     SEQSAC
```


```
***********************
*    NEXSIN           *
***********************
 REFER TO
     SEQSAC
```


```
***********************
*    NEXSIN           *
***********************
 REFER TO
     SEQSAC
```

```
*      NMZMG1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0096
*       FAP
*NMZMG1
       COUNT   75
       LBL     NMZMG1
       ENTRY   NMZMG1 (LX,X,XMAX,SCALE)
*
*                    ----ABSTRACT----
*
* TITLE - NMZMG1
*       NORMALIZE A VECTOR TO GIVEN MAXIMUM VALUE
*
*             NMZMG1 NORMALIZES A FLOATING POINT SERIES TO A SPECIFIED
*             MAXIMUM ABSOLUTE VALUE AND RETURNS THE SCALING FACTOR USED
*             IN THE NORMALIZATION.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 34 REGISTERS
* SPEED     - (LENGTH OF SERIES)*33 MACHINE CYCLES
* AUTHOR    - R.A. WIGGINS,   17/9/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL NMZMG1(LX,X,XMAX,SCALE)
*
* INPUTS
*
*    X(I)       I=1...LX IS A FLOATING POINT SERIES
*
*    LX         MUST BE GRTHN=1
*
*    XMAX       MAXIMUM VALUE WHICH THE X SERIES IS TO ATTAIN
*
* OUTPUTS
*
*    X(I)       I=1...LX IS THE NORMALIZED (TO THE VALUE OF XMAX) SERIES
*
*    SCALE      IS THE SCALING FACTOR THAT THE ORIGINAL SERIES WAS
*                 DIVIDED BY TO OBTAIN THE NORMALIZED SERIES
*               SCALE = MAXIMUM ABSOLUTE VALUE IN XSERIES/XMAX
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5)=1.,3.,-2.,.5,0.  LX=5  XMAX=6.
*    OUTPUTS - X(1...5)=2.,6.,-4.,1.,0.  SCALE=.5
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT XMAX=1.
*    OUTPUTS - X(1...5)=.333,1.,-.667,.1667,0.  SCALE=3.
*
* 3. INPUTS  - X(1...5)=1.,-4.,2.,0.2,0.01  LX=5 XMAX=1.
*    OUTPUTS - X(1...5)=0.25,-1.,0.5,0.05,0.0025  SCALE=4.
*
* 4. INPUTS  - SAME AS EXAMPLE 3. EXCEPT XMAX=-1.
*    OUTPUTS - X(1...5)=-0.25,1.,-0.5,-0.05,-0.0025  SCALE=-4.
*
* 5. INPUTS  - SAME AS EXAMPLE 3. EXCEPT XMAX=0.
*    OUTPUTS - X(1...5)=0.,-0.,0.,0.,0.  SCALE=0.
*
       HTR     0
       BCI     1,NMZMG1
NMZMG1 SXD     *-2,4
       SXA     ADR,1
       CLA*    1,4
       PDX     ,1
       CLA     2,4
       ADD     =1B35
       STA     A
       STA     A+2
       STA     C
       STA     C+2
```

****
*   NMZMG1   *
****
(PAGE 2)

```
         CAL*     2,4
         STO      MAX
         CLA      MAX
A        SBM      **,1
         TPL      B
         CAL      **,1
         STO      MAX
B        TIX      A-1,1,1
         CLA      MAX
         FDP*     3,4
         STQ*     4,4
         STQ      MAX
         CLA*     1,4
         PDX      ,1
C        CLA      **,1
         FDP      MAX
         STQ      **,1
         TIX      C,1,1
ADR      AXT      **,1
         TRA      5,4
MAX      PZE
         END
```

```
*      NOINT1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0374
*      FAP
*NOINT1
       COUNT    370
       LBL      NOINT1
       ENTRY    NOINT1 (X,PROB)
       ENTRY    NOINT2 (XMEAN,XSD,NDIV,XDIV,IANS)
*
*                   ----ABSTRACT----
*
* TITLE - NOINT1 WITH SECONDARY ENTRY NOINT2
*       NORMAL DISTRIBUTION AND DIVISION INTO EQUALLY LIKELY SECTIONS
*
*             NOINT1 FINDS THE INTEGRAL OF THE ZERO MEAN, UNIT VARIANCE,
*             NORMAL PROBABILITY DENSITY FUNCTION FROM MINUS INFINITY
*             TO X. THIS IS DONE BY TABLE LOOK UP IN A TABLE OF 201
*             VALUES OF THE NORMAL DISTRIBUTION WHICH CORRESPOND
*             TO VALUES OF X FROM 0.0 TO 4.0 IN INCREMENTS OF .02
*             LINEAR INTERPOLATION IS USED FOR VALUES OF X LYING
*             BETWEEN TABULATED VALUES. THE PROGRAM RETURNS ZERO FOR X
*             VALUES LESS THAN -4.0, AND RETURNS 1.0 FOR X VALUES
*             GREATER THAN 4.0.
*
*             NOINT2 DIVIDES UP THE ENTIRE X AXIS INTO AN ARBITRARY
*             NUMBER, NDIV, OF RANGES WHICH ARE EQUALLY LIKELY WITH
*             RESPECT TO A GIVEN NORMAL DISTRIBUTION SPECIFIED BY
*             ITS MEAN AND STANDARD DEVIATION.
*
*             THE INTEGRAL OF THE NORMAL DISTRIBUTION GIVES THE
*             PROBABILITY THAT X LIES IN A CERTAIN RANGE.  NOINT2
*             REVERSES THE PROCESS BY FINDING THE X RANGES WITH
*             A GIVEN PROBABILITY.  1/NDIV = PROBABILITY FOR EACH
*             DIVISION.  FOR K-TH DIVISION, XAXIS LIMITS CORRESPOND
*             TO THE PROBABILITIES (K-1)/NDIV, K/NDIV.  STORED VALUES
*             OF THE ANTISYMMETRIC INTEGRAL OF THE UNIT NORMAL
*             DISTRIBUTION FOR X VALUES ZERO TO 4 IN INCREMENTS OF .02
*             ARE SEARCHED FOR PROBABILITY VALUES GIVEN BY K/NDIV.
*             INTERPOLATION WHERE NECESSARY IS LINEAR. I.E. FIND NEAREST
*             VALUE OF X TO CORRESPONDING TO P WHEN P DOES NOT APPEAR
*             IN TABLE EXACTLY.  IF R-TH VALUE IN TABLE IS LESS THAN P,
*             AND (R+1) TH VALUE IS GREATER, THEN X VALUE = ((P-RTH
*             VALUE)/((R+1)TH-RTH VALUE))*.02+R*.02.  THIS VALUE IS
*             THEN SCALED FOR THE PARTICULAR NORMAL DISTRIBUTION SUCH
*             THAT THE OUTPUT X = X*XSD+MEAN.  SINCE ONLY HALF OF THE
*             NORMAL INTEGRAL IS STORED, THE X VALUES CORRESPONDING TO
*             P1 GREATER THAN .5 ARE COMPUTED FIRST AND THE VALUES
*             FOR P2 LESS THAN .5 ARE SYMMETRIC AND EQUAL TO 1-P1.
*
*             NOTE - NOINT1 AND NOINT2 ARE INDEPENDENT EXCEPT FOR
*             THEIR MUTUAL NEED OF THE DISTRIBUTION FUNCTION TABLE.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 369 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON AND J.N. GALBRAITH
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - LINTR1
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE OF NOINT1
*     CALL NOINT1(X,PROB)
*
* INPUTS TO NOINT1
*
*    X          = UPPER LIMIT OF THE INTEGRAL (FLT PT.).
*
* OUTPUTS FROM NOINT1
*
*                    1          X          2
*    PROB      = ---------  INTEGRAL (EXP(-X /2)DX).
*                SQRT(2PI) -INFINITY
*
```

```
*               IS FLOATING POINT
*
* FORTRAN USAGE OF NOINT2
*     CALL NOINT2(XMEAN,XSD,NDIV,XDIV,IANS)
*
* INPUTS TO NOINT2
*
*    XMEAN      = MEAN OF X SERIES
*
*    XSD        = STANDARD DEVIATION OF X SERIES.
*               MUST BE GRTHN 0.
*
*    NDIV       = NUMBER OF EQUALLY LIKELY DIVISIONS INTO WHICH X SERIES
*                 IS TO BE PLACED.
*               MUST BE GRTHN 1
*
* OUTPUTS FROM NOINT2
*
*    XDIV(I)    I=1...NDIV-1 ARE THE X VALUES FOR EQUALLY LIKELY
*                 DIVISIONS. FIRST DIVISION IS FROM -INFINITY TO XDIV(1),
*                 THE SECOND IS FROM XDIV(1) TO XDIV(2) ETC. THE LAST
*                 DIVISION IS FROM XDIV(NDIV-1) TO +INFINITY.
*
*    IANS       =0  NORMAL
*               =1  ILLEGAL XSD
*               =2  ILLEGAL NDIV
*
* EXAMPLES OF NOINT1
*
* 1. INPUTS  - X=-5.
*    OUTPUTS - PROB=0.
*
* 2. INPUTS  - X=-4.
*    OUTPUTS - PROB=.32 E-04
*
* 3. INPUTS  - X=.013
*    OUTPUTS - PROB=.5052
*
* 4. INPUTS  - X=4.
*    OUTPUTS - PROB=.999968
*
* 5. INPUTS  - X=4.1
*    OUTPUTS - PROB=1.
*
* EXAMPLES OF NOINT2
*
* 1. INPUTS  - XMEAN=0.       XSD=1.        NDIV=3
*    OUTPUTS - XDIV(1)=-.430722     XDIV(2)=.430722       IANS=0
*
* 2. INPUTS  - XMEAN=0.       XSD=2.        NDIV=3
*    OUTPUTS - XDIV(1)=-.861444     XDIV(2)=.861444       IANS=0
*
* 3. INPUTS  - XMEAN=1.       XSD=2.        NDIV=3
*    OUTPUTS - XDIV(1)=.1385185     XDIV(2)=1.861444      IANS=0
*
* 4. INPUTS  - XMEAN=0.       XSD=1.        NDIV=2
*    OUTPUTS - XDIV(1)=0.           IANS=0
*
* 5. INPUTS  - XMEAN=3.5      XSD=1.        NDIV=2
*    OUTPUTS - XDIV(1)=3.5          IANS=0
*
* 6. INPUTS  - XMEAN=3.5      XSD=1.        NDIV=1
*    OUTPUTS - ERROR  IANS=2
*
* 7. INPUTS  - XMEAN=3.5      XSD=0.        NDIV=2
*    OUTPUTS - ERROR  IANS=1
*
* 8. INPUTS  - XMEAN=0.       XSD=1.        NDIV=4
*    OUTPUTS - XDIV(1...3)=-.674602,0.,+.674602      IANS=0
*
* 9. INPUTS  - XMEAN=0.       XSD=1.        NDIV=5
*    OUTPUTS - XDIV(1...4)=-.8417856,-.253334,.253334,.8417856     IANS=0
*
*INITIALIZE.
      PZE     0
```

```
        BCI     1,NOINT1
NOINT1  SXA     LV,4
        SXD     NOINT1-2,4
        CLA     1,4
        STA     GETX
        CLA     2,4
        STA     STORE
*GET,STORE X AND ITS SIZE. COMPARE SIZE WITH 4.0.
 GETX   CLA     **            **=ADDRESS OF X
        STO     XX
        SSP
        STO     SX
        CAS     K4FL
        TRA     BIGGER
        TRA     INTRP
        TRA     INTRP
*(OR ZERO FOR NEG X).
BIGGER  CLA     K1FL
        STO     TEMP
        TRA     CHECK
*INTERPOLATE IF SIZE LESS THAN OR = 4.0.
*NOTE LINTR1 MUST BE USED BACKWARDS SINCE OUR
*TABLE IS FORWARDS.
 INTRP  CLA     K4FL
        FSB     SX
        STO     SXMOD
        TSX     $LINTR1,4
        TSX     SXMOD         SXMOD=4.0-SX
        TSX     K0            XLO=0.0
        TSX     KDELX         KDELX=0.02
        TSX     Y+200         TABLE IS FORTRAN VECTOR
        TSX     KD201         NTABLE=201
        TSX     TEMP          ANSWER
*IF X WAS MINUS WE NEED 1.0 MINUS THE INTERPOLATED
*VALUE.
 CHECK  CLA     XX
        TPL     STORE-1
        CLA     K1FL
        FSB     TEMP
        TRA     STORE
        CLA     TEMP
 STORE  STO     **            **=ADDRESS OF PROB
    LV  AXT     **,4          **=XR4
        TRA     3,4
*TEMPORARIES
    XX  PZE     **             **=X
    SX  PZE     **             **=MAGNITUDE OF X
 SXMOD  PZE     **             **=4.0-SX
  TEMP  PZE     **             **=OUTPUT FROM LINTR1
*CONSTANTS
    K0  PZE     0
 KD201  PZE     0,0,201
  K1FL  DEC     1.0
  K4FL  DEC     4.0
 KDELX  DEC     0.02
*       ENTRY   NOINT2 (XMEAN,XSD,NDIV,XDIV,IANS)
*       SAVE IRS AND INITIALIZE IANS
        PZE     0
        BCI     1,NOINT2
NOINT2  SXA     RETURN,1
        SXA     RETURN+1,2
        SXA     RETURN+2,4
        SXD     NOINT2-2,4
        STZ*    5,4             IANS=0
*       CHECK XSD AND NDIV.
        CLA*    2,4             GET XSD
        TZE     ERR1            TRANSFER IF ILLEGAL
        TMI     ERR1            TRANSFER IF ILLEGAL
        CLA*    3,4             GET NDIV
        SUB     K1FX            NDIV-1
        TZE     ERR2            TRANSFER IF ILLEGAL
        TMI     ERR2            TRANSFER IF ILLEGAL
*       PARAMETERS OK. SET UP MEAN LOOP AND GET XSD AND XMEAN ADDRESSES.
        STD     END2            SET UP MEAN LOOP
        CLA     4,4             ADDRESS OF XDIV
```

```
          ADD     KMLI1
          STA     LOOP2
          STA     MEAN+1
          CLA     1,4            ADDRESS OF XMEAN
          STA     MEAN
          LDQ*    2,4
          FMP     KDELX
          STO     SCALE
          CLA     4,4            A(XDIV)
          CLA*    3,4            GET NDIV
          LRS     18             FLOAT IT
          ORA     CONST
          FAD     CONST
          STO     NDIVFL         NDIVFL=FLOATF(NDIV)
          CLA     K1FL
          FDP     NDIVFL
          STQ     DELP
          CLA*    3,4            GET NDIV
          LGR     19
*         NDIV/2  WITH REMAINDER IN SIGN OF MQ
          PAX     ,1
          SXD     END,1
          SSM
          ADD     4,4            (ADDRESS OF XDIV)-NDIV/2
          ADD     KMLI1          ADDRESS OF XDIV(NDIV/2)
          STA     STO1
          STA     STO2
          TQP     EVEN           TRANSFER IF NDIV EVEN
          CLA     DELP
          FDP     K2FL
          XCA
          FAD     Y              P=(.5+DELP/2)
          STO     P
          AXT     0,1
          AXT     1,2
          AXT     0,4
          TRA     SEARCH
   EVEN   AXT     0,2
          CLA     Y              .5
          STO     P
          STZ*    STO1
          AXT     1,2
          AXT     -1,4
          AXT     0,1
   LOOP   CLA     P
          FAD     DELP
          STO     P
  SEARCH  CAS     Y,1            P IS IN AC
          TXI     SEARCH,1,-1     TRY AGAIN
          TRA     SKINT          GOT IT. SKIP INTERPOLATION
          FSB     Y-1,1          INTERPOLATE. P-RTH VALUE
          STO     XTEMP1
          CLA     Y,1            (R+1)TH
          FSB     Y-1,1          RTH
          STO     XTEMP2
          CLA     XTEMP1
          FDP     XTEMP2
          FMP     SCALE
          STO     XTEMP1
          TRA     SKINT+1
   SKINT  STZ     XTEMP1         ZERO INTERPOLATION
          TXI     *+1,1,1        COMPLEMENT OF INDEX OF RTH VALUE IN IR1
          SXA     XTEMP2,1
          PXA     ,1             GET IR1
          PAC     ,1             2 COMPLEMENT
          PXA     ,1             INDEX FOR RTH VALUE =N
          ORA     CONST          FLOAT
          FAD     CONST
          XCA                    FLOATF(N)=FLN IN MQ
          FMP     SCALE          FLN*.02*XSD=X
          FAD     XTEMP1
   STO1   STO     **,2           **=A(XDIV)-NDIV/2+1
          SSM
   STO2   STO     **,4           **=A(XDIV)-NDIV/2+1
          LXA     XTEMP2,1
```

```
        TXI    *+1,4,-1
        TXI    *+1,2,1
  END   TXL    LOOP,2,**      **=NDIV/2  ROUNDED DOWN
 *      FINISHED SEARCH AND SCALING FOR ALL BLOCKS. ADD MEAN
        AXT    1,2
  LOOP2 CLA    **,2           **=A(XDIV)+1
  MEAN  FAD    **             XMEAN
        STO    **,2
        TXI    *+1,2,1
  END2  TXL    LOOP2,2,**     **=NDIV-1
 RETURN AXT    **,1
        AXT    **,2
        AXT    **,4
        TRA    6,4
  ERR1  CLA    K1FX
        STO*   5,4
        TRA    6,4
  ERR2  CLA    K2FX
        STO*   5,4
        TRA    6,4
  CONST OCT    233000000000
  K1FX  PZE    0,0,1
  K2FX  PZE    0,0,2
  KMLI1 PZE    1
  K2FL  DEC    2.0
 XTEMP1 PZE    0
 XTEMP2 PZE    0
   P    PZE    0
  DELP  PZE    0
 NDIVFL PZE
  SCALE PZE    0
 *TABLE (YULE AND KENDALL, THEORY OF STATISTICS,
 *1950, PAGE 664.)
      Y DEC .5000,.5080,.5160,.5239,.5319
        DEC .5398,.5478,.5557,.5636,.5714
        DEC .5793,.5871,.5948,.6026,.6103
        DEC .6179,.6255,.6331,.6406,.6480
        DEC .6554,.6628,.6700,.6772,.6844
        DEC .6915,.6985,.7054,.7123,.7190
        DEC .7257,.7324,.7389,.7454,.7517
        DEC .7580,.7642,.7704,.7764,.7823
        DEC .7881,.7939,.7995,.8051,.8106
        DEC .8159,.8212,.8264,.8315,.8365
        DEC .8413,.8461,.8508,.8554,.8599
        DEC .8643,.8686,.8729,.8770,.8810
        DEC .8849,.8888,.8925,.8962,.8997
        DEC .9032,.9066,.9099,.9131,.9162
        DEC .9192,.9222,.9251,.9279,.9306
        DEC .9332,.9357,.9382,.9406,.9429
        DEC .9452,.9474,.9495,.9515,.9535
        DEC .9554,.9573,.9591,.9608,.9625
        DEC .9641,.9656,.9671,.9686,.9699
        DEC .9713,.9726,.9738,.9750,.9761
        DEC .9772,.9783,.9793,.9803,.9812
        DEC .9821,.9830,.9838,.9846,.9854
        DEC .9861,.9868,.9875,.9881,.9887
        DEC .9893,.9898,.9904,.9909,.9913
        DEC .9918,.9922,.9927,.9931,.9934
        DEC .99379,.99413,.99446,.99477,.99506
        DEC .99534,.99560,.99585,.99609,.99632
        DEC .99653,.99674,.99693,.99711,.99728
        DEC .99744,.99760,.99774,.99788,.99801
        DEC .99813,.99825,.99836,.99846,.99856
        DEC .99865,.99874,.99882,.99889,.99897
        DEC .99903,.99910,.99916,.99921,.99926
        DEC .99931,.99936,.99940,.99944,.99948
        DEC .99952,.99955,.99958,.99961,.99964
        DEC .99966,.99969,.99971,.99973,.99975
        DEC .99977,.99978,.99980,.99981,.99983
        DEC .99984,.99985,.99986,.99987,.99988
        DEC .99989,.99990,.999908,.999915,.999922
        DEC .999928,.999933,.999939,.999943,.999948
        DEC .999952,.999956,.999959,.999963,.999966
        DEC .999968
        END
```

```
************************
*    NOINT2            *
************************
REFER TO
    NOINT1
```


```
************************
*    NOINT2            *
************************
REFER TO
    NOINT1
```

```
*     NRMVEC  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0099
*     LABEL
CNRMVEC
      SUBROUTINE NRMVEC (JOB,SCALE,X,LX,XMEAN,XMAX,XNRM)
C
C                     ----ABSTRACT----
C
C TITLE - NRMVEC
C        NORMALIZE AND CHANGE MEAN OF A VECTOR
C
C              NRMVEC NORMALIZES A VECTOR X SO THAT EITHER ITS RMS VALUE
C              OR THE ABSOLUTE MAXIMUM IS EQUAL TO A GIVEN VALUE. AFTER
C              THE NORMALIZATION IS PERFORMED, A SPECIFIED NUMBER IS
C              ADDED TO EACH TERM OF THE SERIES.  THUS IF EITHER
C
C                           1           LX
C                    XMAX  = ---- SQRTF ( SUM X(I)*X(I) )           (1)
C                           LX          I=1
C
C              OR
C                    XMAX  = ABSF ( MAX (X(I)) )   I=1,...,LX       (2)
C
C              THEN NRMVEC EVALUATES
C
C                    XNRM(I) = X(I)*SCALE/XMAX + XMEAN              (3)
C
C              WHERE SCALE AND XMEAN ARE INPUT PARAMETERS AND THE CHOICE
C              OF NORMALIZATION ALSO DEPENDS ON AN INPUT PARAMETER.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 111  REGISTERS
C SPEED      -
C AUTHOR     - R.A. WIGGINS     JUNE, 1963
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MAXAB
C       AND FORTRAN SYSTEM ROUTINES - SQRT
C
C FORTRAN USAGE
C     CALL NRMVEC(JOB,SCALE,X,LX,XMEAN,XMAX,XNRM)
C
C INPUTS
C
C    JOB       =0. IMPLIES NORMALIZATION IS TO BE MADE ON THE RMS VALUE
C                OF THE SERIES (FORMULA  (1) OF THE ABSTRACT).
C              NOT=0. IMPLIES NORMALIZATION IS TO BE MADE ON THE
C                ABSOLUTE MAXIMUM OF THE SERIES (FORMULA  (2) OF THE
C                ABSTRACT).
C
C    SCALE     IS THE VALUE THAT THE SERIES IS NORMALIZED TO.
C
C    X(I)      I=1,...,LX IS THE SERIES TO BE NORMALIZED.
C
C    LX        IS THE LENGTH OF X.
C              MUST BE GRTHN=1
C
C    XMEAN     IS THE VALUE TO BE ADDED TO THE NORMALIZED SERIES.
C
C OUTPUTS
C
C    XMAX      IS THE MAXIMUM FOUND (BY EITHER FORMULA (1) OR FORMULA
C                (2)).
C
C    XNRM(I)   I=1,...,LX IS THE SERIES NORMALIZED ACCORDING TO FORMULA
C                (3) OF THE ABSTRACT.
C              MAY BE EQUIVALENT OF X.
C
C EXAMPLES
C
C 1. INPUTS  - JOB=1  SCALE=1.  X(1...3)=1.,2.,-4. LX=3  XMEAN=0.
C    OUTPUTS - XNRM(1...3) = .2500,.5000,-1.0000  XMAX=4.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  JOB=0
```

```
C     OUTPUTS - XNRM(1...3) = .6546,1.3093,-2.6816  XMAX=1.5275
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  SCALE=2.
C     OUTPUTS - XNRM(1...3) = .5000,1.0000,-2.0000  XMAX=4.
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  XMEAN=-1.
C     OUTPUTS - XNRM(1...3)= -.7500,-.5000,-2.0000  XMAX=4.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION X(2),XNRM(2)
      IF(LX)  70,70,10
   10 IF(JOB)  40,20,40
   20 XMAX=0.
      DO 30  I=1,LX
   30 XMAX=XMAX+(X(I)*X(I))
      XMAX=SQRTF(XMAX)/FLOATF(LX)
      GO TO 50
   40 CALL MAXAB(LX,X,XMAX,I)
      XMAX=ABSF(XMAX)
   50 SCL=SCALE/XMAX
      DO 60  I=1,LX
   60 XNRM(I)=X(I)*SCL+XMEAN
   70 RETURN
      END
```

```
*      NTHA (FUNCTION)                   10/6/64   LAST CARD IN DECK IS NO. 0092
*      FAP
*NTHA
       COUNT   100
       LBL     NTHA
       ENTRY   NTHA   F(N, A1, A2, .... AN, ...)
       ENTRY   XNTHA  F(N, IA1, IA2, ..., IAN, ...)
*
*
*                     ----ABSTRACT----
*
* TITLE - NTHA  WITH SECONDARY ENTRY XNTHA
*        RETURN N-TH ARGUMENT BEYOND THE FIRST
*
*              NTHA  IS A FUNCTION WITH A VARIABLE NUMBER OF ARGUMENTS,
*              BUT A MINIMUM OF TWO.  THE FIRST ARGUMENT IS AN INTEGER,
*              N,  EXCEEDING ZERO, AND THE VALUE OF THE FUNCTION IS THE
*              N-TH  ARGUMENT BEYOND THE FIRST.
*
*              XNTHA  IS THE FIXED POINT PSEUDONYM FOR  NTHA.
*
* LANGUAGE   - FAP FUNCTION  (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 11 REGISTERS
* SPEED      -  6 MACHINE CYCLES  (ON 7090)  FOR  N = 1
*              14 MACHINE CYCLES IF  N  EXCEEDS  1
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     ARG  =   NTHAF(N,A1,A2,...,AN,...)
*     IARG =   XNTHAF(N,IA1,IA2,...,IAN,...)
*
*
* INPUTS
*
*    N         IS ANY INTEGER EXCEEDING ZERO
*
*    A1,A2,...    ARE FLOATING POINT ARGUMENTS FOR  NTHA
*
*    IA1,IA2,... ARE FIXED POINT ARGUMENTS FOR  XNTHA
*
*
*
* OUTPUTS
*
*    ARG OR IARG  WILL EQUAL THE  N-TH  ARGUMENT BEYOND  N,  EXCEPT THAT
*              IF  N  IS  LSTHN  1  THE VALUE IS  N,  AND IF  N  EXCEEDS
*              THE NUMBER OF ARGUMENTS WHICH FOLLOW, THE VALUE IS
*              UNPREDICTABLE.
*
*
* EXAMPLES
*
* 1. USAGE   -        ARG1 = NTHAF(1, 3.)
*                     DO 10  N=1,4
*                     ARG(N)  =  NTHAF(N, 4., 3., 2., 1.)
*                10   IARG(N) = XNTHAF(N, 1, 2, 3, 4)
*
*    OUTPUTS - ARG1 = 3.   ARG(1...4) = 4.,3.,2.,1.
*              IARG(1...4) = 1,2,3,4
*
*
* PROGRAM FOLLOWS BELOW
*
       BCI     1,NTHA
*
* EQUIVALENT ENTRIES.  NTHAF(N, A1, A2, ...)
*                AND  XNTHAF(N, IA1, IA2, ...)
*
```

NTHA

(PAGE 2)

```
 NTHA  BSS     0
 XNTHA CAS     KD1
       TRA     NBGR1
       XCA                   (A1 IN MQ)
       TRA     1,4
*
* TREATMENT FOR  N  GRTHN 1
*
 NBGR1 SXA     SV4,4
       PDX     0,4
       CLA     32767,4       32767 = OCTAL 77775 + 2
 SV4   AXT     **,4
       TRA     1,4
*
* CONSTANT
*
 KD1   PZE     0,0,1
       END
```

```
*      NURINC (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0326
*      FAP
*NURINC
       COUNT   300
       LBL     NURINC
       ENTRY   NURINC (YOFX, LY, XLO, XHI, LYNU, XLONU, XHINU, IERR1,
*                      YOFXNU, IANS)
*
*
*                    ----ABSTRACT----
*
* TITLE - NURINC
*        CREATE ONE VECTOR FROM ANOTHER WITH NEW RANGE AND NEW INCREMENT
*
*              NURINC  TAKES A FUNCTION, SPECIFIED BY EVENLY SPACED
*              VALUES
*
*                 Y(X)  FOR  X = XLO, XLO+DX, XLO+2DX, ...,
*                              XHI=XLO+(LY-1)DX
*
*              AND PRODUCES, BY LINEAR INTERPOLATION, ANOTHER SET OF
*              EVENLY SPACED VALUES OF THE FUNCTION
*
*                 Y(X)  FOR  X = XLONU, XLONU+DXNU, ...,
*                              XHINU=XLONU+(LYNU-1)DXNU
*
*              WHERE THE PROGRAM INPUTS ARE THE FIRST SET OF VALUES OF
*              Y,  LY,  XLO,  XHI,  LYNU,  XLONU,  AND  XHINU.
*
*              THE OUTPUT MAY REPLACE THE INPUT IN RESTRICTED CASES.
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 121 REGISTERS
* SPEED      - ON THE 7090  NURINC  TAKES ABOUT
*                 200 + 13LY + 72LYNU MACHINE CYCLES
*                 WHERE  LY  AND  LYNU  ARE DEFINED ABOVE.
* AUTHOR     - S.M. SIMPSON,  JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL NURINC(YOFX, LY, XLO, XHI, LYNU, XLONU, XHINU, IERR1,
*    1            YOFXNU, IANS)
*
*
* INPUTS
*
*    YOFX(I)   I = 1...LY  ARE THE INPUT SET OF VALUES OF  Y(X)  OF THE
*                 ABSTRACT.
*
*    LY        MUST BE  GRTHN=  2 .
*
*    XLO       IS THE FIRST INPUT  X  VALUE, I.E.,  YOFX(1) = Y(XLO).
*
*    XHI       IS THE LAST INPUT  X  VALUE, I.E.,  YOFX(LY) = Y(XHI).
*              XHI  MUST EXCEED  XLO.
*
*    LYNU      IS THE DESIRED NUMBER OF OUTPUT VALUES.
*              MUST BE  GRTHN=  1 .
*
*    XLONU     IS THE FIRST OUTPUT  X  VALUE, I.E.,  YOFXNU(1)
*                 WILL  =  Y(XLONU).
*              MUST SATISFY  XLO  LSTHN=  XLONU  LSTHN=  XHI.
*
*    XHINU     IS THE LAST OUTPUT  X  VALUE, I.E.,  YOFXNU(LYNU)
*                 WILL  =  Y(XHINU).
*              HOWEVER,  XHINU  IS NOT REFERRED TO IF  LYNU = 1 .  IF
*                 LYNU  EXCEEDS  1  THEN  XHINU  MUST SATISFY
```

```
*                 XLONU  LSTHN  XHINU  LSTHN=  XHI.
*
*    IERR1     +1  IS THE DESIRED  IANS  VALUE FOR ILLEGAL  LY.
*              SHOULD EXCEED ZERO.
*
*
* OUTPUTS
*
*    YOFXNU(I) I = 1...LYNU  ARE THE LINEARLY INTERPOLATED VALUES,
*                 COMPUTED ONLY IF  IANS = 0 .  EQUIVALENCE(YOFXNU,YOFX)
*                 IS PERMITTED UNDER A NUMBER OF CIRCUMSTANCES, THE
*                 SIMPLEST OF WHICH REQUIRES SIMULTANEOUSLY
*                     A)  XLO = XLONU = 1.0,
*                     B)  XHI = XHINU = FLOATF(LY),
*                 AND C)  LYNU  IS LESS THAN OR EQUAL TO  LY.
*                 IN GENERAL, THE OUTPUTS ARE STORED IN ORDER OF
*                 INCREASING  I  VALUES.  ANY EQUIVALENCE WHICH OVERLAPS
*                 YOFXNU  AND  YOFX  MUST ASSURE THAT THE OUTPUT STORAGE
*                 DOES NOT DESTROY AN INPUT VALUE NEEDED IN A SUBSEQUENT
*                 INTERPOLATION.  NO CHECK FOR THIS CONDITION IS MADE
*                 BY  NURINC.
*
*    IANS       = 0            IF ALL OK.  OTHERWISE
*               = IERR1+K-1  WHERE K = ARGUMENT NUMBER OF AN ILLEGAL
*                  INPUT ARGUMENT,
*                 K = 2  (LY),   = 4  (XHI),   = 5  (LYNU),   = 6  (XLONU),
*                   = 7  (XHINU)
*
*
* EXAMPLES
*
* 1. MISCELLANEOUS INTERPOLATIONS OF A LINEAR VECTOR
*
*    INPUTS  - YOFX(1...10) = 1.,2.,...,10.   LY=10.  XLO=1.  XHI=10.
*              LYNU(1...7)  = 2,3,3,3,11,1,1
*              XLONU(1...7) = 1.,1.,8.5,3.1416,2.,1.,10.
*              XHINU(1...7) = 10.,10.,9.5,3.9,3.,43.,-17.  IERR1 = 1
*    USAGE   -        DIMENSION YOFXNU(11,7)
*                     DO 10  I=1,7
*               10    CALL NURINC(YOFX,LY,XLO,XHI,LYNU(I),XLONU(I),
*                  1            XHINU(I),IERR1,YOFXNU(1,I),IANS(I))
*    OUTPUTS - YOFXNU(1...2,1)  = 1.,10.
*              YOFXNU(1...3,2)  = 1.,5.5,10.
*              YOFXNU(1...3,3)  = 8.5,9.,9.5
*              YOFXNU(1...3,4)  = 3.1416,3.5208,3.9000
*              YOFXNU(1...11,5) = 2.0,2.1,2.2,...,2.9,3.0
*              YOFXNU(1,6)      = 1.,  (XHINU NOT USED)
*              YOFXNU(1,7)      = 10., (XHINU NOT USED)
*              IANS(1...7)      = 0,0,...,0
*
* 2. SHORTEST VECTOR
*
*    INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   -        CALL NURINC(YOFX(3),2,3.,4.,3,3.1416,3.9,1,YOFXNU,
*                  1            IANS)
*    OUTPUTS - YOFXNU(1...3) = 3.1416,3.5208,3.9000   IANS = 0
*
* 3. OVERLAP OF OUTPUT ON TOP OF INPUT
*
*    INPUTS  - Y1(1...10) = Y2(1...10) = 1.,2.,...,10.
*    USAGE   -        CALL NURINC(Y1,10,1.0,10.0,4,1.0,10.0,1,Y1,IANS1)
*                     CALL NURINC(Y2,10,1.0,10.0,10,1.0,10.0,1,Y2,IANS2)
*    OUTPUTS - Y1(1...10) = 1.0,4.0,7.0,10.0,5.,6.,7.,8.,9.,10.
*              Y2(1...10) = 1.,2.,...,10.   IANS1 = IANS2 = 0
*
* 4. ILLEGAL CASES
*
*    USAGE   -        CALL NURINC(YOFX,1,1.,10.,2,1.,10.,1,YNU,IANS2)
*                     CALL NURINC(YOFX,2,1.,1.,2,1.,10.,1,YNU,IANS4)
*                     CALL NURINC(YOFX,2,1.,10.,0,1.,10.,1,YNU,IANS5)
*                     CALL NURINC(YOFX,2,1.,10.,2,0.,10.,1,YNU,IANS6A)
*                     CALL NURINC(YOFX,2,1.,10.,1,11.,10.,1,YNU,IANS6B)
*                     CALL NURINC(YOFX,2,1.,10.,2,1.,1.,1,YNU,IANS7A)
*                     CALL NURINC(YOFX,2,1.,10.,2,1.,11.,1,YNU,IANS7B)
```

```
*    OUTPUTS - IANS2,4,5,6A,6B,7A,7B = 2,4,5,6,6,7,7
*
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR     0              XR1
       HTR     0              XR2
       HTR     0              XR4
       BCI     1,NURINC
*
* ONLY ENTRY.  NURINC(YOFX,LY,XLO,XHI,LYNU,XLONU,XHINU,IERR1,YOFXNU,
*                      IANS)
*
NURINC SXD     NURINC-4,1
       SXD     NURINC-3,2
       SXD     NURINC-2,4
*
* SET ADDRESSES
*
       CLA     1,4            A(YOFX)
       ADD     K1             A(YOFX)+1
       STA     CLA1
       STA     FSB1
       STA     FAD1
       ADD     K1             A(YOFX)+2
       STA     CLA2
       CLA     9,4            A(YOFXNU)
       ADD     K1             A(YOFXNU)+1
       STA     STO2
*
* CHECK LY GRTHN= 2, XHI GRTHN XLO, SET DELX
*
       CLA*    8,4            IERR1
       ADD     KD1
       PDX     0,1            IERR1+2-1 TO XR1
       CLA*    2,4            LY
       CAS     KD1
       TRA     SUBK1
       NOP                    MUST EXCEED 1
       TRA     LEAVE
 SUBK1 SUB     KD1            LY-1
       STD     TXL1
       LRS     18
       ORA     OCTK
       FAD     OCTK
       STO     TEMP           (LY-1) FLOATED
       TXI     *+1,1,2        IERR1+4-1
       CLA*    4,4            XHI
       STO     XHI
       FSB*    3,4            XHI-XLO
       TMI     LEAVE
       TZE     LEAVE
       FDP     TEMP
       STQ     DELX
*
* CHECK LYNU GRTHN= 1, FORM (LY-1) FLOATED IF LYNU GRTHN 1 .
*
       TXI     *+1,1,1        IERR1+5-1
       CLA*    5,4            LYNU
       SUB     KD1            LYNU-1
       STD     TXL2
       TMI     LEAVE          NG
       TNZ     *+2            NORMAL
       TXI     GETXLU,1,1     OK BUT BYPASS XHINU CHECK, DELXNU SET
       SUB     KD1            LYNU-2
       STD     TXL3
       ADD     KD1            LYNU-1
       LRS     18
       ORA     OCTK
       FAD     OCTK
       STO     TEMP
*
```

```
* CHECK XHINU, SET DELXNU
*
       TXI     *+1,1,2         IERR1+7-1
       CLA*    7,4             XHINU
       STO     XHINU
       CAS*    4,4             XHINU AGAINST XHI
       TRA     LEAVE           NG
       NOP                     OK   XHINU-XLONU
       FSB*    6,4             OK
       TMI     LEAVE
       TZE     LEAVE
       FDP     TEMP
       STQ     DELXNU
*
* CHECK XLONU AND SET IT
*
       TXI     *+1,1,-1        IERR1+6-1
GETXLU CLA*    6,4             XLONU
       STO     XNUNXT
       CAS*    4,4             AGAINST XHI
       TRA     LEAVE
       NOP                     (OK FOR LYNU = 1, IMPOSSIBLE OTHERWISE)
       CAS*    3,4             AGAINST XLO
       TRA     START           OK
       TRA     START           OK
       TRA     LEAVE           NG
*
* LOOP STARTS AT CAS1.  INITIALIZING REQUIRED IS
*    1.  XNXT   = XLO IN AC,  XNUNXT = XLONU  IN VARIABLES TABLE
*    2.  IXNXT  =  1  IN XR1
*    3.  IXNUNX =  1  IN XR2
*    4.  XHI, XHINU, DELX, DELXNU IN VARIABLES TABLE
*
* (YOFX, YOFXNU ARE SOMETIMES CALLED Y, YNU BELOW)
*
 START AXT     1,3             XRS SET
       CLA*    3,4             XLO IN AC
       TRA     CAS1
*
* INCREMENT XNXT BUT FORCE EQUALITY WITH XHI FOR IXNXT = LY
*
XLSXNU FAD     DELX
 TXL1  TXL     CAS1,1,**       ** = LY-1
       CLA     XHI
*
* XNXT IS IN AC, XR1 HAS IXNXT, XR2 HAS IXNUNX
*
 CAS1  CAS     XNUNXT          XNXT AGAINST XNUNXT
       TRA     STO1            OK, GO INTERPOLATE
       TRA     EQUAL           OK, GO SET
       TXI     XLSXNU,1,1      GO JUMP XNXT AND TRY AGAIN
 EQUAL STO     XNXT
 CLA1  CLA     **,1            ** = A(YOFX)+1    (GIVES Y(IXNXT))
       TRA     STO2
*
* FORM YNU(IXNUNX) = Y(IXNXT)-(XNXT-XNUNXT)(Y(IXNXT)-Y(IXNXT-1))/DELX
*
 STO1  STO     XNXT
       FSB     XNUNXT
       STO     TEMP
 CLA2  CLA     **,1            ** = A(YOFX)+2    GIVES Y(IXNXT-1)
 FSB1  FSB     **,1            ** = A(YOFX)+1
       FDP     DELX            -(Y(IXNXT)-Y(IXNXT-1))/DELX
       FMP     TEMP                              TIMES (XNXT-XNUNXT)
 FAD1  FAD     **,1            ** = A(YOFX)+1    PLUS Y(IXNXT)
 STO2  STO     **,2            ** = A(YOFXNU)+1  (YNU(IXNUNX))
*
* INCREMENT XNUNXT, FORCING EQUALITY WITH XHINU FOR IXNUNX = LYNU
*
       CLA     XNUNXT
       FAD     DELXNU
 TXL2  TXL     TXL3,2,**       ** = LYNU-1       COMPLETION CHECK
       TRA     WINDUP
 TXL3  TXL     STO3,2,**       ** = LYNU-2       LAST IXNUNX CHECK
       CLA     XHINU
```

NURINC
(PAGE 5)

```
 STO3  STO    XNUNXT
       CLA    XNXT
       TXI    CAS1,2,1
WINDUP AXT    0,1               IANS FOR OK
*
* EXIT
*
 LEAVE PXD    0,1
       STO*   10,4              IANS
       LXD    NURINC-4,1
       LXD    NURINC-3,2
       TRA    11,4
*
* CONSTANTS
*
 K1    PZE    1
 KD1   PZE    0,0,1
 OCTK  OCT    233000000000
*
* VARIABLES
*
 XHI   PZE    **,**,**          INPUT
 XHINU PZE    **,**,**          INPUT
 XNXT  PZE    **,**,**          = (XLO),XLO+DELX,...
XNUNXT PZE    **,**,**          = XLONU,XLONU+DELXNU,...
 DELX  PZE    **,**,**          (XHI-XLO)/(LY-1)
DELXNU PZE    **,**,**          (XHINU-XLONU)/(LYNU-1)
 TEMP  PZE    **,**,**
       END
```

```
*     NXALRM (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0177
*     LABEL
CNXALRM
      SUBROUTINE NXALRM(JOB,MLIV,ILO,IHI,LEVEL,LTENSE,IBGIN,IEND,
     1                  ISUM,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - NXALRM
C       SCAN VECTOR FOR POSSIBLE BLOCK OF VALUES ALL ABOVE GIVEN LEVEL
C
C              NXALRM SCANS A GIVEN RANGE OF A FIXED POINT VECTOR TO
C              FIND THE NEXT BLOCK OF VALUES  A) WHICH EQUAL OR EXCEED
C              A GIVEN LEVEL, AND B) WHOSE BLOCK LENGTH EQUALS OR
C              EXCEEDS A GIVEN LENGTH.  SCANNING IS FROM LOW INDICES
C              TO HIGH INDICES.  OUTPUT IS FIRST AND LAST INDICES OF
C              BLOCK (IF ONE IS FOUND).  OPTIONAL OUTPUT IS SUM OF
C              VALUES IN BLOCK.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 243 REGISTERS
C SPEED     - FOR LONG SCANS BEFORE HITTING BLOCK, SPEED IS SAME AS THAT
C                OF SUBROUTINE FASCN1
C AUTHOR    - S.M. SIMPSON JR, JUNE 1962
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FASCN1
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL NXALRM(JOB,MLIV,ILO,IHI,LEVEL,LTENSE,IBGIN,IEND,
C    1            ISUM,IANS)
C
C INPUTS
C
C    JOB        =0 PERFORM ORDINARY COMPUTATIONS AS INDICATED BELOW.
C               =1 IS A HIGH SPEED BYPASS WHICH ELIMINATES THE
C                  COMPUTATION OF IEND AND ISUM.
C
C    MLIV(I)    I=ILO,...,IHI IS THE VECTOR RANGE FOR STUDY (FIXED POINT,
C                 WHERE THE BINARY POINT IS ARBITRARY)
C
C    ILO        (EXCEEDS ZERO)
C
C    IHI        (EQUALS OR EXCEEDS ILO)
C
C    LEVEL      IS THE GIVEN LEVEL (FIXED POINT, SAME BINARY POINT
C                 AS MLIV)
C
C    LTENSE     (EXCEEDS ZERO) IS THE MINIMUM BLOCK LENGTH.
C
C OUTPUTS
C
C    IBGIN      MLIV(IBGIN) IS FIRST VALUE IN BLOCK FOUND, IF ANY.
C               IS SET =0 IF NO BLOCK FOUND.
C
C    IEND       MLIV(IEND) IS LAST VALUE IN BLOCK FOUND (MLIV(IEND+1) IS
C                 LESS THAN LEVEL).
C               IS SET =0 IF NO BLOCK FOUND.
C               (IF JOB=1 AND BLOCK IS FOUND, IEND=IBGIN+LTENSE-1)
C
C    ISUM       IS SUM OF MLIV(I) FROM I=IBGIN TO IEND (FIXED POINT, SAME
C                 BINARY POINT AS MLIV). NO OVERFLOW CHECK IS MADE.
C               =0 IF NO BLOCK FOUND.
C               (ISUM IS MEANINGLESS IF JOB=1)
C
C    IANS       =0 MEANS NO BLOCK FOUND.
C               =1 MEANS BLOCK FOUND AND SPECIFIED
C               =2 MEANS POSSIBLE BLOCK STARTED BUT RAN OFF MLIV VECTOR
C                  BEFORE END WAS REACHED.
C                  NOTE - IN THIS CASE IEND=IHI,ISUM=SUM OF MLIV(I) FROM
C                         IBGIN TO IHI.
```

```
C                  NOTE - BLOCK IS DEFINITE IF IHI-IBGIN+1 EQUALS OR
C                         EXCEEDS LTENSE.
C                =-1  MEANS ILLEGAL SPECIFICATION OF ILO,IHI, OR LTENSE.
C                =-99 MEANS UNEXPECTED ERROR RETURN FROM FASCN1.
C
C EXAMPLES (MLI USED BELOW STANDS FOR MACHINE LANGUAGE INTEGER)
C
C 1. INPUTS  - JOB=0, MLIV(1...50)=MLI10,20,30,40,50,40,0,0,0,0,1,2,3,4,
C              5,6,7,8,9,10,9,8,7,6,5,4,3,2,1,0,10,10,10,10,0,0,...
C              ILO=1, IHI=50, LEVEL=MLI30, LTENSE=1
C    OUTPUTS - IANS=1, IBGIN=3, IEND=6, ISUM=MLI160
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LTENSE=2
C    OUTPUTS - SAME AS EXAMPLE 1.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LTENSE=5
C    OUTPUTS - IANS=0, IBGIN, IEND, AND ISUM NOT AFFECTED.
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ILO=8, IHI=22 AND LEVEL=MLI6
C    OUTPUTS - IANS=2, IBGIN=16, IEND=22 AND ISUM=MLI57
C
C 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=5 AND LTENSE=5
C    OUTPUTS - IANS=2, IBGIN=3, IEND=5, ANS ISUM=MLI120
C
C 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LEVEL=MLI50
C    OUTPUTS - IANS=1, IBGIN=5, IEND=5, ISUM=MLI50
C
C 7. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LEVEL =MLI6
C    OUTPUTS - IANS=1, IBGIN=1, IEND=6, ISUM=MLI190
C
C 8. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=5, LEVEL=MLI50
C    OUTPUTS - IANS=2, IBGIN=5, IEND=5, ISUM=MLI50
C
C 9. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ILO=10, LEVEL=MLI9, LTENSE=4
C    OUTPUTS - IANS=1, IBGIN=31, IEND=34, ISUM=MLI40
C
C10. INPUTS  - SAME AS EXAMPLE 1, EXCEPT JOB=1
C    OUTPUTS - IANS=2, IBGIN=3, IEND=3
C
      DIMENSION MLIV(2)
C INITIALIZE, CHECKING ILO, IHI, LTENSE.
      IANS=-1
      IF (ILO-1)  9999,10,10
   10 IF (IHI-ILO)  9999,20,20
   20 IF (LTENSE-1)  9999,50,50
C SET UP FOR FIRST SCAN TRIAL.
   50 IMIN=ILO
      IEND=IHI
C CLEAR ISUM AND BEGIN NEW SCAN.
  100 ISUM=0
      CALL FASCN1(MLIV,IMIN,IHI,LEVEL,I,IANSR)
      IF (IANSR)  9900,120,200
  120 CONTINUE
C NO ALARM FOUND IF FALLS THRU 120.
      IANS=0
      IBGIN=0
      IEND=0
      GO TO 9999
C LEVEL REACHED.  CHECK FOR TENSE LOOP.
  200 IANS=2
      IBGIN=I
      JLO=I+1
      IF (I-IHI)  210,400,400
  210 IF (LTENSE-1)  300,300,220
C TENSE LOOP CHECKS FOR DEFINITE ALARM IN THE CASE LTENSE IS 2 OR GRTR.
  220 JHI=JLO+LTENSE-2
      IF (JHI-IHI)  230,230,225
  225 JHI=IHI
  230 DO 240  J=JLO,JHI
      ISUM=ISUM+MLIV(J-1)
      IF (MLIV(J)-LEVEL)  250,240,240
  240 CONTINUE
C DEFINITE ALARM IF FALLS THRU 240
C CHECK FOR J=IHI BEFORE GOING TO PULLOUT.
      JLO=JHI+1
```

```
      IF (JHI-IHI)  300,400,400
C CANCEL ALARM IF JUMPS HERE FROM LOOP AND THEN RETURN TO SCAN MODE.
  250 IMIN=J
      GO TO 100
C DEFINITE ALARM. PULL OUT OF IT IF CAN BUT FIRST CHECK FOR HIGH SPEED
C EXIT WHICH BYPASSES PULLOUT.
  300 IEND=JLO-1
      IF (JOB)  310,310,400
  310 IEND=IHI
      DO 320  J=JLO,IHI
      ISUM=ISUM+MLIV(J-1)
      IF (MLIV(J)-LEVEL)  340,320,320
  320 CONTINUE
C BOX INCOMPLETE IF FALLS THRU 320 (ADD IN LAST SUM).
      GO TO 400
C CASE WHERE CANT PULL OUT OR SPECIAL CASE WHEN LEVEL FIRST REACHED
C AT IHI.
  400 ISUM=ISUM+MLIV(IHI)
      GO TO 9999
C BOX COMPLETE IF JUMPS FROM HERE TO LOOP
  340 IANS=1
      IEND=J-1
      GO TO 9999
C FASCN1 ERROR EXIT
 9900 IANS=-99
      GO TO 9999
C EXIT
 9999 RETURN
      END
```

```
*       ONLINE (SUBROUTINE)              4/14/65   LAST CARD IN DECK IS NO. 0190
*       FAP
*ONLINE
        COUNT   200
        SST
        LBL     ONLINE
        ENTRY   ONLINE (ISENSE)
        ENTRY   (STH)              (STORAGE TO TAPE HOLLERITH)
        ENTRY   (STHM)             (STORAGE TO TAPE HOLLERITH / MONITOR)
        ENTRY   (STHD)             (STORAGE TO TAPE HOLLERITH / DEBUG)
*
*
*
*                    ----ABSTRACT----
*
* TITLE - ONLINE, WITH SECONDARY ENTRY POINTS (STH), (STHM), (STHD)
*        OPTIONAL ONLINE MONITOR OF BCD TAPE WRITING.
*
*               SUBROUTINE ONLINE IS A MODIFICATION OF (STH) TO ALLOW
*               MONITORING OF BCD TAPE OUTPUT ON THE ONLINE PRINTER.
*               THIS IS ACCOMPLISHED BY SPECIFYING A SENSE SWITCH TO
*               SUBROUTINE ONLINE.  THEN, WHILE THIS SWITCH IS DOWN, ALL
*               BCD MATERIAL THAT IS WRITTEN ON A TAPE IS ALSO PRINTED
*               ON-LINE.  THE SWITCH USED MAY BE ALTERED DURING THE
*               PROGRAM AT WILL.
*
*               THE SECONDARY ENTRIES  (STH),  (STHM),  AND  (STHD)
*               FUNCTION IDENTICALLY TO THE STANDARD FORTRAN-II SYSTEM
*               ENTRIES OF THE SAME NAMES, EXCEPT FOR THIS MONITORING
*               FEATURE.
*
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME, TAPE UNIT, AND ONLINE PRINTER)
* STORAGE    - 134  REGISTERS
* SPEED      -
* AUTHOR     - R.A. WIGGINS   4/64
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (FIL),(IOH),(RCH),(SPH),(TES),
*                                     (WER),(WRS),(WTC)
*
* FORTRAN USAGE
*     CALL ONLINE(ISENSE)
*
*
* INPUTS
*
*    ISENSE     IS THE SENSE SWITCH NUMBER WHICH MUST BE DOWN TO
*                 ACTIVATE PRINTING.
*               IF LSTHN= 0 OR GRTHN 6 NO SWITCH IS ACTIVATED.
*
*
* OUTPUTS       IF THE ISENSE SENSE SWITCH IS DOWN, ALL BCD MATERIAL THAT
*                 IS WRITTEN ON TAPE IS ALSO PRINTED ONLINE.
*
*
* PROGRAM FOLLOWS BELOW
*
 XR4    HTR     0
        BCI     1,ONLINE
BUFSIZ EQU      22            RECORD BUFFER SIZE
*
 (STH) LDQ      *+2           PICKUP SWITCH SETTING, AND
        TRA     TRAIO         GO INITIALIZE (IOH).
        TRA     STH           OUTPUT / STORAGE TO TAPE HOLLERITH.
*
* STANDARD  STORAGE TO TAPE  ENTRY
*
*
(STHM) LDQ      *+2           PICKUP SWITCH SETTING, AND
        TRA     ONQ           GO INITIALIZE (IOH).
```

```
       TRA     STHM          OUTPUT / MONITOR
*
* IF THIS IS ONLINE, SCAN MAIN FOR TSX $(FIL),4 AND ALTER
*   TO TSX $ONLINE,4
*
 ONQ   NZT     ISENSE        IS ISENSE SWITCH TURNED ON
       TRA     TRAIO         NO, WRITE OUT NORMALLY.
 PSE   PSE     **            YES, IS THE SENSE SWITCH ON
       TRA     TRAIO         NO,  WRITE OUT NORMALLY.
       SXD     XR4,4         SAVE IR 4 AND
       STO     FMTLOC        SAVE ACCUMULATOR.
 SRCH  CAL     1,4           YES, SEARCH
       STA     *+1           FOR
       CAL     **            LOCATION ADDRESSING
       LAS     $(FIL)        TTR  (FIL).
       TRA     *+2
       TRA     *+2           FOUND IT,
       TIX     SRCH,4,1
       CAL     1,4           SAVE FOR
       SLW     TSXFIL        FUTURE USE.
       CAL     TRAON2        REPLACE WITH
       STA     1,4           TSX $ONLI2,4
       LXD     XR4,4         RESTORE IR 4 AND
       CLA     FMTLOC        ACCUMULATOR.
       TRA     TRAIO         GO INITIALIZE (IOH).
*
(STHD) LCQ     TRAD          PICKUP SWITCH SETTING, AND
 TRAIO TRA*    $(IOH)      * GO INITIALIZE (IOH).
TRAD   TRA     STHD          OUTPUT / DEBUG
*
STHDA  LDI     SIND          RESTORE INDICATORS.
       AXT     0,4           COUNT OF DEBUG LINES PRINTED.
       TXH     STHX,4,1000   LEAVE IF NUMBER EXCEEDED.
       TXI     *+1,4,1       UPDATE LINE COUNT
       SXA     *-3,4         AND SAVE.
       LXA     STHX,4        RESTORE RETURN INDEX.
*
 STHM  CAL     LINECT        INCREASE
       ADM     WDCNT         LINE COUNT
       STA     LINECT        BY 1.
*
 STH   SXA     STHX,4        SAVE RETURN INDEX.
*
 TES   TSX     $(WER),4    * GO CHECK PREVIOUS WRITE.
       LXA     STHX,4        SET
 WDCNT CAL     1,4           WORD COUNT
       STD     STHC          OF WRITE COMMAND.
       AXT     0,4           MOVE
       SXA     *+6,2         RECORD
       PCX     ,2            INTO
       CAL     REC,4         OUTPUT
       SLW     OUTPUT,4      BUFFER
       TXI     *+1,4,-1      ..
       TIX     *-3,2,1       ..
       AXT     ..,2          ..
       CAL     TES           SET SWITCH FOR
       SLW*    $(TES)        WRITE OVERLAP.
       XEC*    $(WRS)        SELECT CURRENT UNIT.
       AXC     STHC,4        INITIALIZE
       PXA     ,4            FOR
       STA*    $(WTC)        WRITE CHECKING.
       XEC*    $(RCH)        WRITE ONE TAPE RECORD.
 STHX  AXT     ..,4          RESTORE RETURN INDEX.
       TRA     2,4         * EXIT TO (IOH).
STHD   STI     SIND          SAVE INDICATORS.
       LDI     BLKS          LOAD INDICATORS WITH BLANKS.
       SXA     STHX,4        SAVE RETURN INDEX.
       CAL     1,4           PUT 2S COMPLEMENT OF NUMBER OF
       PCC     0,4           WORDS IN OUTPUT RECORD INTO IR 4
       TXI     *+1,4,3       AND REDUCE BY 3.
       ONT     REC+1,4       CHECK FOR NON ZERO AND NON BLANK.
       TRA     STHDA         FOUND (PRINT THIS LINE).
       TXI     *+1,4,1       EXAMINE NEXT WORD
       TXH     *-3,4,0       OF OUTPUT RECORD.
       LDI     SIND          ENTIRE LINE HAS ONLY ZERO NUMERICS.
```

```
       TRA     STHX           CONSEQUENTLY DO NOT PRINT.
*
* ONLINE ENTRY
*
CNLINE CLA*    1,4            INITIAL ONLINE ENTRANCE.  GET ISENSE,
       TMI     PXD            TEST
       TZE     PXD            FOR
       ARS     18             ILLEGAL
       SUB     =7B35          VALUES OF
       TPL     PXD            ISENSE.
       ADD     =119B35        OK, SET
       STA     PSE            UP PSE.
       CLA     =1B17          TURN ON
 STOIS STO     ISENSE         ISENSE SWITCH
       TRA     2,4          * RETURN TO MAIN.
 PXD   PXD     0,0            TURN OFF
       TRA     STOIS          ISENSE
*
* IF ISENSE SWITCH ON, CONTROL COMES HERE AFTER WRITING TAPE
*
 ONLI2 SXA     XR4,4          SAVE LOCATION OF TSX $(FIL),4
       TSX     $(FIL),4     * CALL (FIL) FROM HERE
       LXA     XR4,4          RESET IR 4
       CAL     TSXFIL         RESET TSX $(FIL),4
       STA     0,4            TO OLD DEFINITION, AND
       LXD     XR4,4          RESET IR4 TO INITIAL ADDRESS OF WRITING
       TRA     $(SPH)       * LIST AND CALL PRINTING ROUTINE.
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
ISENSE PZE     0
TSXFIL PZE     0
TRAON2 PZE     ONLI2
FMTLOC
 SIND  PZE
 BLKS  BCI     1,
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 STHC  IOST    OUTPUT,,..     WRITE COMMAND.
OUTPUT BSS     BUFSIZ         OUTPUT BUFFER.
       COMMON  -206+BUFSIZ
 REC   COMMON  1
 ..    EQU     0
       END
```

```
*     OUDATA (SUBROUTINE)              3/15/65   LAST CARD IN DECK IS NO. 0268
*     LABEL
COUDATA
      SUBROUTINE OUDATA(ITAPE,IRECNO,NOPTS,DATA,MODCOD)
C
C                     ----ABSTRACT----
C
C TITLE - OUDATA
C        FAST AND CONVENIENT DATA STORAGE ON TAPE
C
C              OUDATA WRITES DATA AND OTHER INFORMATION ABOUT THE DATA
C              ON A TAPE IN BINARY IN A FORM THAT CAN BE INTERPRETED BY
C              INDATA.
C
C              THE DATA AND ITS AUXILIARY INFORMATION ARE STORED IN 3
C              LOGICAL BLOCKS AS FOLLOWS
C
C                  BLOCK 1 IS AN INDEXING BLOCK CONTAINING 5 WORDS.
C                      WORD 1 IS THE RECORD NUMBER IN FIXED POINT,
C                               FLOATING POINT, OCTAL OR ALPHANUMERIC,
C                      WORD 2 IS THE LENGTH OF THE AUXILIARY INFORMATION
C                               BLOCK IN FIXED POINT,
C                      WORD 3 IS THE NUMBER OF DATA VALUES IN THE DATA
C                               BLOCK IN FIXED POINT.
C                      WORD 4 IS THE NUMBER OF DATA VALUES PACKED PER
C                               DATA WORD IN THE DATA BLOCK IN FIXED PT.
C                      WORD 5 IS THE SCALE FACTOR THAT THE DATA VALUES
C                               WERE MULTIPLIED BY BEFORE PACKING
C                               - FLOATING POINT.
C
C                  BLOCK 2 IS THE AUXILIARY INFORMATION BLOCK DIVIDED
C                    INTO GROUPS OF 3.  WORD 1 OF A GROUP CONTAINS THE
C                    ALPHANUMERIC NAME ASSOCIATED WITH A PARTICULAR PIECE
C                    OF INFORMATION.  WORD 2 CONTAINS A FIXED POINT
C                    NUMBER TELLING THE LENGTH OF THE INFORMATION (N).
C                    THE FOLLOWING N WORDS CONTAIN THE AUXILIARY INFOR-
C                    MATION IN WHATEVER MODE THAT IS ASSOCIATED WITH
C                    THIS INFORMATION.  AN ARBITRARY NUMBER OF THESE
C                    INFORMATION GROUPS MAY BE WRITTEN.  A BLANK WORD
C                    AND A SUM-CHECK WORD (AS COMPUTED BY FAPSUM) FOLLOW
C                    THE FINAL GROUP.
C
C                  BLOCK 3 CONTAINS THE DATA VALUES THAT OUDATA WILL
C                    SCALE, FIX, AND PACK BEFORE WRITING.  THE FINAL
C                    DATA WORD IS FOLLOWED BY A SUM-CHECK WORD (AS
C                    COMPUTED BY FAPSUM)
C
C              THE FINAL BLOCK IS FOLLOWED BY AN END FILE
+
C              THE FINAL FILE OF DATA ON A DATA TAPE MUST BE FOLLOWED BY
C              A DUMMY FILE WITH RECORD NO. = 0 TO IDENTIFY THE END OF
C              THE DATA.
C
C              THE OPERATIONS OF WRITING ARE ALL CONTROLLED BY INPUT
C              PARAMETERS IN THE CALLING SEQUENCE.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - IBM 709 OR 7090 (MAIN FRAME, DATA CHANNEL AND TAPE UNIT)
C STORAGE    - 495 REGISTERS
C SPEED      -
C AUTHOR     - J.F. CLAERBOUT  AUGUST, 1962
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - VARARG,LOC,MVBLOK,PAKN,FAPSUM
C        AND FORTRAN SYSTEM ROUTINES - (STB),(WLR),(EFT).
C
C FORTRAN USAGE
C     CALL OUDATA(ITAPE,IRECNO,NOPTS,DATA,MODCOD,
C    1           NAME1,LAUX1,AUX1,
C    2           NAME2,LAUX2,AUX2,
C    .           .
C    .           .
C    N           NAMEN,LAUXN,AUXN)
C              THE NUMBER OF ARGUMENTS IS VARIABLE DEPENDING UPON THE
```

```
C              AUXILIARY INFORMATION TO BE WRITTEN. WITH THE PRESENT
C              DIMENSIONING, OUDATA IS LIMITED TO 100 ARGUMENTS.
C
C INPUTS
C
C    ITAPE     IS THE LOGICAL TAPE NUMBER TO BE WRITTEN ON.
C              MUST BE FIXED POINT.
C
C    IRECNO    IS A RECORD NUMBER (SYMBOL) WHICH IS ASSOCIATED WITH
C                THE DATA.
C              MAY BE FIXED POINT, FLOATING POINT, OCTAL OR ALPHANUMERIC
C
C    NOPTS     IS NUMBER OF DATA POINTS.
C              MUST BE GRTHN= 1 .
C
C    DATA(I)   I=1,NOPTS IS THE DATA TO BE WRITTEN.  IF THE DATA IS TO
C                BE PACKED (I.E. MODCOD GRTHN=2) IT MUST BE FLOATING
C                POINT.  IF IT IS NOT TO BE PACKED, THEN THE DATA MAY
C                BE IN ANY MODE.  (SEE MODCOD)
C
C    MODCOD    IS THE NUMBER OF DATA VALUES TO BE PACKED PER WORD.
C              MUST BE GRTHN= 1, LSTHN= 18 .
C              IS FIXED POINT.
C
C              NOTE THAT THE DATA IS SCALED TO THE MAXIMUM ACCURACY
C                AVAILABLE FOR ANY SPECIFIC MODCOD (UNLESS MODCOD=1 IN
C                WHICH CASE THE DATA IS NOT TOUCHED) AND THEN FIXED
C                BEFORE PACKING.  THE MAXIMUM ACCURACY IS GIVEN BY
C
C                          36/MODCOD
C                  MAXX = 2.         -1.
C
C                IF THE INPUT SERIES CONSISTS OF FLOATING POINT INTEGERS
C                ALL OF WHICH ARE LSTHN= MAXX, THEN THE ORIGINAL
C                INTEGER VALUES MAY BE OBTAINED (WHEN THE TAPE IS READ
C                BY SUBROUTINE INDATA) BY ROUNDING THE OUTPUT VALUES
C                TO THE NEAREST INTEGER.
C
C      THE FOLLOWING N GROUPS OF INPUT ARGUMENTS CONTROL THE
C        CONSTRUCTION OF THE AUXILIARY INFORMATION BLOCK.  THESE
C        ARGUMENTS OCCUR IN GROUPS OF 3 FOR EACH PIECE OF INFORMATION.
C        ONLY THE N-TH CASE IS DESCRIBED. THE TOTAL LENGTH OF THE
C        AUXILIARY INFORMATION BLOCK LAUXBK MUST BE LSTHN = 200
C        WHERE
C                                    N
C             LAUXBK = 2 + 2N + SUM LAUXN
C                                    1
C
C    NAMEN     IS A WORD THAT NAMES THE N-TH INFORMATION.
C              IS GENERALLY HOLLERITH BUT MAY ALSO BE FIXED POINT,
C                FLOATING POINT, OR OCTAL.
C
C    LAUXN     IS THE NUMBER OF WORDS IN THE N-TH INFORMATION.
C              MUST BE GRTHN= 1 .
C              IS FORTRAN II INTEGER.
C
C    AUXN(I)   I=1,LAUXN IS THE N-TH AUXILIARY INFORMATION.
C              MAY BE IN ANY MODE.
C              IS NEVER PACKED.
C
C OUTPUTS
C
C    DATA(I)   I=1,NOPTS IS DESTROYED.  THE PROGRAM CAN BE ALTERED
C                TO RESTORE THE DATA (EXCEPT FOR ROUND-OFF) AS EXPLAINED
C                IN A COMMENT NEAR THE END OF THE PROGRAM.
C
C EXAMPLES
C
C 1. USAGE   -       CALL OUDATA(9,62,NOPTS,DATA,MODCOD)
C    OUTPUTS - THE DATA IS WRITTEN WITH NO AUXILIARY INFORMATION ON
C                TAPE 9.
C
C 2. INPUTS  - DT = .05
C              DATE(1...3) = 30, 6, 1962
C    USAGE   -       CALL OUDATA(9,63,NOPTS,DATA,MODCOD,
```

```
C                   1     6HDELTAT, 1, DT,
C                   2     4HDATE,   3, DATE)
C     OUTPUTS - THE DATA IS WRITTEN
C
C 3. NUMERICAL EXAMPLES
C
C     INPUTS  - IRECNO(1...2) = 6HSAMPLE, 3
C               DT=.05  RDAY=30  RUNITS=6HMICRON
C               TITLE (1...8) = 6H SAMPLE INDATA-OUDATA TYPE TAPE RECORD
C     USAGE   - C     CONSTRUCT A TYPICAL OUDATA TYPE TAPE
C                     DATA(1)=1.
C                     DATA(2)=2.
C                     DATA(3)=3.
C                     CALL OUDATA(ITAPE,IRECNO(1),3,DATA,2)
C                     DATA(1)=1.
C                     DATA(2)=2.
C                     DATA(3)=3.
C                     CALL OUDATA(ITAPE,IRECNO(2),3,DATA,3,
C                    1             6HDELTAT,1,DT,
C                    2             4HRDAY  ,1,RDAY,
C                    3             6HRUNITS,1,RUNITS,
C                    4             5HTITLE ,8,TITLE)
C               C
C               C     DO A ZERO RECORD NO. TO INDICATE END OF TAPE
C               C
C                     CALL OUDATA (ITAPE,0,1,DATA,1)
C     OUTPUTS - THIS IS AN OCTAL LISTING OF THE BINARY TAPE FORMED.
C
C
C     RECORD 1 OF FILE   1
C
C               622144474325   000002000000   000003000000   000002000000
C               220525251252
C
C     RECORD 2 OF FILE   1
C
C               000000000000   000000000000
C
C     RECORD 3 OF FILE   1
C
C               252525125252   000000377777   252525525251
C
C     RECORD 1 OF FILE   2
C
C               000003000000   000025000000   000003000000   000003000000
C               212525125252
C
C     RECORD 2 OF FILE   2
C
C               242543632163   000001000000   174631463146   512421706060
C               000001000000   000036000000   516445316362   000001000000
C               443123514645   633163432560   000010000000   606221444743
C               256031452421   632140466424   216321606370   472560632147
C               256051252346   512460606060   606060606060   000000000000
C               615376046704
C
C     RECORD 3 OF FILE   2
C
C               377725251252   377725251252
C
C     RECORD 1 OF FILE   3
C
C               000000000000   000002000000   000001000000   000001000000
C               212525125252
C
C     RECORD 2 OF FILE   3
C
C               000000000000   000000000000
C
C     RECORD 3 OF FILE   3
C
C               377725251252   377725251252
C
C
      DIMENSION DATA(5000),LOCS(100),IBLOK(200),BLOK(200)
```

OUDATA
(PAGE 4)

```
      EQUIVALENCE (BLOK,IBLOK)
      CALL VARARG(LOCS)
      GO TO 20
   10 RETURN
   20 CONTINUE
C     SET UP AUXILIARY INFORMATION BLOCK AT BLOK
      L=1
      J=6
   30 CONTINUE
C     IS AUX INFO BLOCK ALL SET UP
      IF(LOCS(J))40,50,40
   40 CONTINUE
C      TACK ON ANOTHER REQUEST
      CALL LOC(BLOK(L),LB)
      CALL MVBLOK(1,LOCS(J),LB)
      CALL MVBLOK(1,LOCS(J+1),LB-1)
      CALL MVBLOK(IBLOK(L+1),LOCS(J+2),LB-2)
      L=L+2+IBLOK(L+1)
      J=J+3
      GO TO 30
   50 CONTINUE
      BLOK(L)=0.
C     APPEND SUM CHECK
      CALL FAPSUM(L,BLOK,SUMCK)
      BLOK(L+1)=SUMCK
      LAUXBK=L+1
C     PACK AND SCALE DATA, APPEND SUMCK
      CALL PAKN(MODCOD,NOPTS,DATA,SCALE)
      NN=(NOPTS+MODCOD-1)/MODCOD+1
      NN1=NN-1
      CALL FAPSUM(NN1,DATA,SUMCK)
C     WRITE FMT BLOCK
      WRITE TAPE ITAPE,IRECNO,LAUXBK,NOPTS,MODCOD,SCALE
C     WRITE AUX BLOCK
      WRITE TAPE ITAPE,(BLOK(I),I=1,LAUXBK)
C     WRITE DATA
      WRITE TAPE ITAPE,(DATA(I),I=1,NN1),SUMCK
      END FILE ITAPE
   60 CONTINUE
C     BY REMOVING ((C)) FROM THE NEXT CARD, DATA WILL BE RESTORED
C     EXCEPT FOR ROUND-OFF ERROR.
C     CALL UNPAKN(MODCOD,NOPTS,DATA,SCALE)
      GO TO 10
      END
```

```
*      PACDAT (SUBROUTINE)              9/9/64   LAST CARD IN DECK IS NO. 0258
*      FAP
*PACDAT
       COUNT   200
       LBL     PACDAT
       ENTRY   PACDAT (ITAPE,NWORDS,IFW,IFOLD,DATA,LDATA,IANS)
*
*                    ----ABSTRACT----
*
* TITLE - PACDAT
*       READ EVERY N-TH WORD FROM BINARY TAPE
*
*              PACDAT READS A SPECIFIED NUMBER OF WORDS FROM A BINARY
*              RECORD OR SEQUENCE OF RECORDS, STARTING WITH A WORD WITH
*              GIVEN INDEX IN THE RECORD, AND OBTAINING SUCCESSIVE WORDS
*              BY SKIPPING A SPECIFIED NUMBER OF WORDS BETWEEN
*              EACH READING.  IT RETURNS WITH AN ERROR FLAG IF AN
*              END-OF-FILE OR A REDUNDANCY CHECK IS ENCOUNTERED.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 152 REGISTERS
* SPEED      -
* AUTHOR     - R.A. WIGGINS   JULY,1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*      AND FORTRAN SYSTEM ROUTINES - (IOS),(TCO),(RDS),(RCH),(ETT)
*
* FORTRAN USAGE
*     CALL PACDAT(ITAPE,NWORDS,IFW,IFOLD,DATA,LDATA,IANS)
*
* INPUTS
*
*    ITAPE     INPUT TAPE NUMBER.
*              MUST BE GRTHN 0
*
*    NWORDS    IS THE TOTAL NUMBER OF WORDS THAT IS TO BE READ.
*              MUST BE GRTHN 0
*              MAY BE GRTHN LENGTH OF THE RECORDS.
*
*    IFW       IS THE INDEX OF THE FIRST WORD TO BE READ FROM THE BLOCK.
*              MUST BE GRTHN 0
*
*    IFOLD     IS ONE GRTHN THE NUMBER OF WORDS SKIPPED. I.E. THIS IS
*                THE INCREMENT IN THE READING INDEX.
*              MUST BE GRTHN 0
*
* OUTPUTS
*
*    DATA(I)   I=1,...,LDATA CONTAINS EVERY IFOLD-TH VALUE READ
*                BEGINNING WITH THE IFW-TH VALUE.
*
*    LDATA     GIVES THE NUMBER OF DATA POINTS READ. IF AN END OF FILE
*                OR REDUNDANCY IS ENCOUNTERED THIS NUMBER MAY BE ONE
*                LESS THEN THE ACTUAL NUMBER READ.
*
*    IANS      = 0  NORMALLY
*              = 1  IF END OF FILE IS ENCOUNTERED.
*              = 2  IF A REDUNDANCY CHECK IS ENCOUNTERED.
*              =-1  IF ILLEGAL ITAPE
*              =-2  IF ILLEGAL NWORDS
*              =-3  IF ILLEGAL IFW
*              =-4  IF ILLEGAL IFOLD
*
* NOTE - IF AN END FILE IS ENCOUNTERED. THE TAPE IS POSITIONED
*          IMMEDIATELY AFTER THE END FILE MARK.
*        IF A REDUNDANCY IS ENCOUNTERED, THE TAPE IS POSITIONED AFTER
*          THE RECORD IN WHICH THE REDUNDANCY WAS FOUND.
*        AFTER A NORMAL EXIT THE TAPE IS POSITIONED AFTER THE LAST
*          RECORD READ FROM.
*
* EXAMPLES
*
```

```
*    NOTE THAT FORTRAN WRITES AN EXTRA WORD AT THE BEGINNING OF EACH
*    BINARY RECORD.  THIS WORD APPEARS AS A FLOATING POINT ZERO IN THESE
*    EXAMPLES.
*
* 1. GENERAL APPLICATION - READING WORDS FROM ONE RECORD.
*    INPUTS  - ITAPE = 5  NWORDS = 5  IFW = 3  IFOLD = 2
*              DATA1(1...20) = 1.,2.,...20.
*    USAGE   -       WRITE TAPE ITAPE, (DATA1(I),I=1,20)
*                    REWIND ITAPE
*                    CALL PACDAT (ITAPE,NWORDS,IFW,IFOLD,DATA,LDATA,
*                   1   IANS)
*    OUTPUTS - DATA(1...5) = 2.,4.,6.,8.,10.  LDATA = 5  IANS = 0
*
* 2. READING ACROSS RECORD GAPS, TERMINATION BY AN END-FILE.
*    INPUTS  - SAME AS EXAMPLE 1.  EXCEPT NWORDS = 7
*    USAGE   -       WRITE TAPE ITAPE, (DATA1(I),I=1,3)
*                    WRITE TAPE ITAPE, (DATA1(I),I=4,6)
*                    WRITE TAPE ITAPE, (DATA1(I),I=7,8)
*                    END FILE ITAPE
*                    REWIND ITAPE
*                    CALL PACDAT (ITAPE,NWORDS,IFW,IFOLD,DATA,LDATA,
*                   1   IANS)
*    OUTPUTS - DATA(1...5) = 2.,0.,5.,0.,8.  LDATA = 5  IANS = 1
*
* 3. READING ALL VALUES, TERMINATED BY A REDUNDANCY.
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT IFW = 1  IFOLD = 1  NWORDS = 10
*    USAGE   -       WRITE TAPE ITAPE, (DATA1(I),I=1,3)
*                    WRITE TAPE ITAPE, (DATA1(I),I=3,6)
*                    WRITE OUTPUT TAPE ITAPE, 10, DATA1(1)
*                 10 FORMAT(F6.2)
*                    REWIND ITAPE
*                    CALL PACDAT (ITAPE,NWORDS,IFW,IFOLD,DATA,LDATA,
*                   1   IANS)
*    OUTPUTS - DATA(1...8) = 0.,1.,2.,3.,0.,3.,4.,5.,6.
*              LDATA = 9   IANS = 2
*
* 4. ERROR CONDITIONS
*    USAGE   -       CALL PACDAT (0,5,1,1,DATA,LDATA,IANS1)
*                    CALL PACDAT (2,0,1,1,DATA,LDATA,IANS2)
*                    CALL PACDAT (2,5,0,1,DATA,LDATA,IANS3)
*                    CALL PACDAT (2,5,1,0,DATA,LDATA,IANS4)
*    OUTPUTS - IANS1 = -1   IANS2 = -2   IANS3 = -3   IANS4 = -4
*
* PROGRAM FOLLOWS BELOW
*
 XR4   HTR     0
       BCI     1,PACDAT
PACDAT SXD     XR4,4          SAVE
       SXA     XR1,1          INDEX
       SXA     XR2,2          REGISTERS
       CLA     11             SAVE
       STO     NIF1           CHANNEL
       CLA     13             TRAPPING
       STO     NIF2           INSTRUCTIONS
       STZ*    7,4            RESET IANS TO ZERO
       CLA*    2,4            (=NWORDS)
       TZE     IANM2
       TMI     IANM2
       STO*    6,4
       PDX     ,1
       CAL     5,4            =ADR(DATA)
       STA     IN
       CLA*    3,4            =IFW
       SUB     =1B17
       TMI     IANM3
       STD     SKM
       TZE     A3
       LDQ     NOP
       CLA     NOP
       TRA     A4
 A3    LDQ     TRA2
       CLA     LCHA1
 A4    STA     RCHA
       STQ     RCHA+1
       CLA*    4,4            =IFOLD
```

```
        SUB     =1B17
        TMI     IANM4
        STD     SKN
        LDQ     ALCH2
        TNZ     A2
        LDQ     ALCH1
A2      XCA
        STA     TIX
        CLA*    1,4             SET
        TZE     IANM1           UP
        TMI     IANM1           TAPE
        ADD     =020            HANDLING
        TSX     $(IOS),4        INSTRUCTIONS.
        LXD     XR4,4
        LDQ*    $(TCO)          TCO
        SLQ     TCOA
        SLQ     TCOA1
        LDQ*    $(RDS)          RTD
        STQ     RDSA
        LDQ*    $(RCH)          RCH
        SLQ     RCHA
        XCL                     LCH
        ADD     =0000400000000
        XCL
ALCH1   SLQ     LCHA1
ALCH2   SLQ     LCHA2
        CAL*    $(ETT)          LOCATION
        ANA     =03000          OF
        ARS     9               TRAP
        PAX     ,2
        SXA     ENBIN,2
        SXD     ENBIN,2
        ALS     1
        ADD     =8B35           INDICATOR.
        STA     TRAP
        CLA     TRA
        STO     11
        STO     13
TCOA    TCOA    *               DELAY IF CHANNEL IN OPERATION.
RDSA    RTBA    **              READ SELECT.
        ENB     ENBIN           ENABLE BOTH CHANNELS.
RCHA    RCHA    SKM             RESET AND LOAD CHANNEL.
        NOP
LCHA1   LCHA    IN              READ ONE WORD.
        CAL     IN
        SUB     =1B35
        SLW     IN
TIX     TIX     LCHA2,1,1       COUNT IT.
        TRA     LVO
LCHA2   LCHA    SKN             SKIP IFOLD-1 WORDS
        TRA     LCHA1           CYCLE.
SKM     IOCTN   **,,**          **=IFW-1
IN      IOCT    **,,1           **=ADR(DATA)
SKN     IOCTN   **,,**          **=IFOLD-1
LVO     STZ*    7,4
LV      CLA     NIF1            LEAVE.
TCOA1   TCOA    *
        IOT
        NOP
        ENB     =0
        STO     11
        CLA     NIF2
        STO     13
        LXD     XR4,4
XR1     AXT     **,1
XR2     AXT     **,2
        TEFA    *+1
        TEFB    *+1
        TRCA    *+1
        TRCB    *+1
        TRA     8,4
TRA     TRA     TRAP
ENBIN   PZE     3,,3
TRAP    LXD     **,2
        TRA     TRA1+1,2
```

****
* PACDAT *
****
(PAGE 4)

```
         NOP
         NOP
         NOP
         TRA   ENDFL
   NOP   NOP   SKM
         TRA   REDUN
   TRA1  TRA   LVO          NO LCH WAITING
   ENDFL CLA   =1B17
         STO*  7,4
   A1    PXD   ,1
         SUB*  6,4
         SSP
         STO*  6,4
         TRA   LV
   REDUN CLA   =2B17
         STO*  7,4
         TRA   A1
   IANM1 CLS   =1B17
         STO*  7,4
         TRA   LV
   IANM2 CLS   =2B17
         STO*  7,4
         TRA   LV
   IANM3 CLS   =3B17
         STO*  7,4
         TRA   LV
   IANM4 CLS   =4B17
         STO*  7,4
         TRA   LV
   TRA2  TRA   LCHA1+1
  *      . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   NIF1  PZE
   NIF2  PZE
         END
```

```
*       PAKN (SUBROUTINE)                  9/29/64   LAST CARD IN DECK IS NO. 0146
*       FAP
*PAKN
        COUNT   150
        LBL     PAKN
        ENTRY   PAKN  (N,LD,D,SCALE)
*
*                     ----ABSTRACT----
*
* TITLE - PAKN
*         SCALE AND FIX DATA VECTOR, PACK N DATA POINTS PER REGISTER
*
*               PAKN SCALES FLOATING POINT DATA TO THE LARGEST VALUE
*               COMMENSURATE WITH THE NUMBER OF DATA POINTS PER WORD,
*               ROUNDS AND FIXES THE DATA AND PACKS IT FROM RIGHT TO LEFT.
*               THE ORIGINAL DATA IS DESTROYED BY THE PROGRAM.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 78 REGISTERS
* SPEED     - ABOUT (LENGTH OF ORIGINAL VECTOR)*60 MACHINE CYCLES
* AUTHOR    - J.F. CLAERBOUT
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - FXDATA
*         AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*       CALL PAKN  (N,LD,D,SCALE)
*
* INPUTS
*
*    N          IS NUMBER OF DATA POINTS PACKED PER REGISTER.
*               MUST BE GRTHN=1, LSTHN=18.
*               IF =1 THE DATA IS NOT SCALED OR FIXED.
*               IS FORTRAN II INTEGER
*
*    D(I)       I=1...LD IS THE FLOATING POINT DATA SERIES TO BE PACKED.
*
*    LD         IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*    D(I)       I=1...(LD+N-1)/N  IS THE PACKED DATA SERIES.
*
*    SCALE      IS THE SCALE FACTOR USED ON DATA
*                  DATA*SCALE = SCALED DATA FOR PACKING
*
* EXAMPLES
*
* 1. INPUTS  - D(1...6) = 1.,4.,8.,-7.,5.,2.  LD=6  N=1
*    OUTPUTS - D(1...6) = 1.,4.,8.,-7.,5.,2.  SCALE NOT CHANGED
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  N=2
*    OUTPUTS - D(1...3) = OCT 200000040000,  737773777777,  100000237777
*               SCALE = 16383.87
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  N=5
*    OUTPUTS - D(1...2) = OCT 237567720020,  000000000040
*               SCALE = 7.875
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  N=7
*    OUTPUTS - D(1) = OCT 002117275004   SCALE = 1.875
*
* 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  N=18
*    OUTPUTS - D(1) = OCT 000000000724   SCALE = 0.125
*
        HTR     0
        BCI     1,PAKN
 PAKN   SXD     *-2,4
        SXA     SV2,2
        SXA     SV1,1
        CLA*    1,4
        ARS     18
```

```
        STO     N
        SUB     =1
        TZE     5,4
        CLA     2,4
        STA     TSXMX+1
        CLA*    2,4
        STD     L
        CLA     3,4
        STA     TSXMX+2
        ADD     =1
        STA     SS
        STA     SL
        STA     SD
        STA     SF
        CLA     =0
        LDQ     =36
        DVP     N
        STQ     NB          NB = NUMBER OF BITS IN PACKED WORD
        CLA     NB
        SUB     =1
        STO     NBM
*       FORM  ((2**(NB-1))-1) = WANTED MAX OF DATA
        CLA     NBM
        STA     *+2
        CLA     =1
        ALS     **
        STO     NB2
        SUB     =1
        ALS     18
        STO     MAX1
*       SCALE AND FIX
 TSXMX  TSX     $FXDATA,4
        TSX     **              =L
        TSX     **              =D
        TSX     MAX1
        TSX     SCALE
*       REPLACE THE SIGN
        LXD     L,1
 SF     CLA     **,1
        TPL     *+3
        SUB     NB2
 SS     STO     **,1
        TIX     SF,1,1
        CLA     NB
        STA     IRS
*       PACK
        AXT     1,2             FOR PICKUP
        AXT     1,4             FOR STORAGE
 NXW    LXA     N,1             FOR PACK COUNT
 SL     CAL     **,2            ** = DATA+1
 IRS    LGR     **              ** = BITS/PACKED NUMBER
        TXI     *+1,2,1
        TIX     SL,1,1
 SD     STQ     **,4            ** = DATA+1
        TXI     *+1,4,1
 L      TXL     NXW,2,**         **=L
        CLA     SCALE
 SV4    LXD     PAKN-2,4
        STO*    4,4
 SV1    AXT     **,1
 SV2    AXT     **,2
        TRA     5,4
 MAX1   PZE
 NB     PZE                     NUMBER OF BITS IN PACKED WORD
 NBM    PZE                     NUMBER OF BITS IN PACKED WORD MINUS ONE
 N      PZE                     NUMBER OF NUMBERS PACKED IN ONE REGISTER
 NB2    PZE                     2**NBM
 MAX    PZE                     MAXIMUM OF DATA
 SCALE
 ORF    OCT     233000000000
 AN     OCT     000007777777
        END
```

```
*     PLANSP (SUBROUTINE)                9/9/64   LAST CARD IN DECK IS NO. 0382
*     LABEL
CPLANSP
      SUBROUTINE PLANSP (JOB,NRA,NCA,AA,MRS,JMAXR,MCS,JMAXC,SPT,
     1                  SPACE1,SPACE2,IANS)
C
C                 ----ABSTRACT----
C
C TITLE - PLANSP
C        FAST TWO-DIMENSIONAL SPATIAL SPECTRUM
C
C              PLANSP FINDS THE TWO-DIMENSIONAL SPECTRUM OF EITHER A
C              CENTRO-SYMMETRIC OR A CENTRO-ANTISYMMETRIC RECTANGULAR
C              ARRAY.  THAT IS, GIVEN AN ARRAY OF NUMBERS
C
C                    A(X,Y)  X=-XL,-XL+1.,...,XL
C                            Y=-YL,-YL+1.,...,YL
C
C              (WHERE  XL  AND  YL  ARE EITHER INTEGERS OR HALF-
C              INTEGERS) THAT IS CENTRO-SYMMETRIC,
C
C                    A(X,Y)  =  A(-X,-Y),
C
C              THEN PLANSP EVALUATES THE FORMULA
C
C                          XL     YL
C               SP(I,J) = SUM   SUM(A(X,Y)*COS(I*X*PI/MR + J*Y*PI/MC))
C                         X=-XL Y=-YL
C
C              OR, IF THE ARRAY IS ANTISYMMETRIC,
C
C                    A(X,Y)  = -A(-X,-Y),
C
C              PLANSP EVALUATES THE FORMULA
C
C                          XL     YL
C               SP(I,J) = SUM   SUM(A(X,Y)*SIN(I*X*PI/MR + J*Y*PI/MC))
C                         X=-XL Y=-YL
C
C                    FOR  I = -JMAXR,...,JMAXR
C                         J =    0  ,...,JMAXC
C
C              WHERE  PI = 3.14159265,
C                      XL, YL, MR, MC, JMAXR, AND JMAXC ARE RELATED TO
C                        INPUT PARAMETERS,
C                      0 LSTHN JMAXR LSTHN= MR,
C                      0 LSTHN JMAXC LSTHN= MC.
C
C              SPEED IS OBTAINED BY
C
C                1. ROTATING THE INPUT ARRAY TO MINIMIZE THE TOTAL
C                     NUMBER OF MULTIPLICATIONS.
C                2. COLLAPSING AND SPLITTING A(X,Y) (WHERE POSSIBLE).
C                3. USING THE HIGH-SPEED LOOPING LOGIC OF SUBROUTINE
C                     COSISP TO PERFORM THE TRANSFORM OF THE REDUCED
C                     PARTS.
C
C              TEMPORARY REGISTERS ARE REQUIRED FOR THE COMPUTATIONS.
C              A SPECIAL ENTRY TO PLANSP IS PROVIDED WHICH WILL GIVE
C              THE SIZE NEEDED FOR A PARTICULAR SET OF INPUT PARAMETERS.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 1169 REGISTERS
C SPEED      - IF MR LSTHN= XL AND 2*XL IS ODD AND IF
C                 MC LSTHN= YL AND 2*YL IS ODD, THE TIME IS ABOUT
C                    40*(JMAXR+1)*MC*(MR+JMAXC+1)     OR
C                    40*(JMAXC+1)*MR*(MC+JMAXR+1)
C                 MACHINE CYCLES ON THE 7090, WHICHEVER IS SMALLER.
C              IF MR GRTHN XL AND 2*XL IS ODD, SUBSTITUTE 2*XL FOR MR.
C              IF MC GRTHN YL AND 2*YL IS ODD, SUBSTITUTE 2*YL FOR MC.
C              IF 2*XL IS EVEN SUBSTITUTE 2*MR (IF MR LSTHN= XL)
C                 OR 4*XL+1 (IF MR GRTHN XL) FOR MR.
C              IF 2*YL IS EVEN SUBSTITUTE 2*MC (IF MC LSTHN= YL)
C                 OR 4*YL+1 (IF MC GRTHN YL) FOR MC.
```

PLANSP (PAGE 2)


```
C AUTHOR     - R.A. WIGGINS,  JULY, 1964
C
C                        ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CHOOSE,COSIS1,COSTBL,IXCARG,STZ,
C                                     LIMITS,KOLAPS,MOVREV,ROAR2,SETKS,
C                                     SINTBL,XADDK,XADDKS,XOOZE,MATRA
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C      CALL PLANSP(JOB,NRA,NCA,AA,MRS,JMAXR,MCS,JMAXC,SPT,
C     1                SPACE1,SPACE2,IANS)
C
C INPUTS
C
C     JOB       =1  IMPLIES  A  IS CENTRO-SYMMETRIC.  THE FIRST FORMULA
C                      DESCRIBED IN THE ABSTRACT IS EVALUATED.
C               =-1 IMPLIES  A  IS CENTRO-ANTISYMMETRIC.  THE SECOND
C                      FORMULA DESCRIBED IN THE ABSTRACT IS EVALUATED.
C               =0  IMPLIES THAT THE USER ONLY DESIRES TO KNOW THE
C                      LENGTHS OF THE SPACE VECTORS.  NO OUTPUTS ARE
C                      GIVEN OTHER THAN IANS.
C
C     NRA       = 2*XL AS USED IN THE ABSTRACT.
C               IS THE TOTAL NUMBER OF ROWS  IN  A .
C               MUST EXCEED 0 .
C
C     NCA       = 2*YL AS USED IN THE ABSTRACT.
C               IS THE TOTAL NUMBER OF COLUMNS IN  A .
C               MUST EXCEED 0 .
C
C
C     AA(I)     I=1,...,NRA*((NCA+1)/2) CONTAINS A(X,Y)  X=-XL,-XL+1.,
C                 ...,XL,  Y=YZ,YZ+1.,...,YL  WHERE  YZ=0.  IF  NCA  IS
C                 ODD,  YZ=.5  IF  NCA  IS EVEN.  AA IS STORED BY ROWS
C                 CLOSELY SPACED.
C
C     MRS       = MR  IN THE ABSTRACT.
C               DEFINES THE FUNDAMENTAL FREQUENCY OF THE TRANSFORM ACROSS
C                 THE ROWS TO HAVE A PERIOD OF 2*MRS+1 .
C               MUST EXCEED 0 .
C
C     JMAXR     DEFINES THE HIGHEST MULTIPLE OF THE FUNDAMENTAL FREQUENCY
C                 DESIRED ACROSS THE ROWS.
C               MUST BE GRTHN 0  LSTHN= MRS .
C
C     MCS       = MC  IN THE ABSTRACT.
C               DEFINES THE FUNDAMENTAL FREQUENCY OF THE TRANSFORM ACROSS
C                 THE COLUMNS TO HAVE A PERIOD OF 2*MCS+1 .
C               MUST EXCEED 0 .
C
C     JMAXC     DEFINES THE HIGHEST MULTIPLE OF THE FUNDAMENTAL FREQUENCY
C                 DESIRED ACROSS THE COLUMNS.
C               MUST BE GRTHN 0  LSTHN= MCS .
C
C     SPACE1(I) I=1,...,LSP1  IS TEMPORARY COMPUTATION SPACE, WHERE
C                 LSP1 = 3+MAX(LSR1*NC1+MAX(NRNR1,NCNC1+LSC2),
C                        NRNR1*NC1+LSR1)+2*MAX(MRS,MCS)
C                        ( MIN (NRA/2,MRS)    IF NRA ODD
C                   NR = (
C                        ( MIN (NRA-1,2*MRS) IF NRA EVEN
C
C                        ( MIN (NCA/2,MCS)    IF NCA ODD
C                   NC = (
C                        ( MIN (NCA-1,2*MCS) IF NCA EVEN
C
C                   NR1=NR+1
C                   NRNR1=NR+NR+1
C                   NC1=NC+1
C                   NCNC1=NC+NC+1
C                   LSR1=JMAXR+1
C                   LSR2=JMAXR+JMAXR+1
C                   LSC2=JMAXC+JMAXC+1
C               MAY BE EQUIVALENT TO AA
C
```

```
C     SPACE2(I) I=1,...,LSP2 IS TEMPORARY COMPUTATION SPACE, WHERE
C                 LSP2 = MAX(LSR1*NC1,LSC2*LSR1,NRA*((NCA+1)/2))
C                 (SEE SPACE1 FOR A DEFINITION OF THE TERMS)
C               MAY BE EQUIVALENT TO SPT.
C
C OUTPUTS
C
C     SPT(I)    I=1,...,(2*JMAXR+1)*(JMAXC+1) CONTAINS SP(I,J)
C                 I = -JMAXR,...,JMAXR,  J = 0,...,JMAXC  AS DEFINED IN
C                 THE ABSTRACT.  IS STORED CLOSELY SPACED, BY COLUMNS.
C
C     IANS(I)   (I=1) = 0  NORMALLY
C                     = 1  IF JOB ILLEGAL
C                     = 2  IF NRA LSTHN= 0
C                     = 3  IF NCA LSTHN= 0
C                     = 4  IF MRS LSTHN= 0
C                     = 5  IF MCS LSTHN= 0
C                     = 6  IF JMAXR LSTHN= 0, GRTHN MRS
C                     = 7  IF JMAXC LSTHN= 0, GRTHN MCS
C               (I=2) = LSP1  AS DEFINED UNDER SPACE1.
C               (I=3) = LSP2  AS DEFINED UNDER SPACE2.
C               IS THE ONLY OUTPUT IF JOB=0
C
C
C EXAMPLES
C
C 1. INPUTS   - JOB = 0  NRA = 4  NCA = 4
C              AA(1...8) = 1.0  -2.0                 (STORED BY COLUMNS)
C                          2.0  -1.0
C                          2.0   1.0
C                          1.0   2.0
C              MRS = 2  JMAXR = 2  MCS = 2  JMAXC = 2
C    OUTPUTS - IANS(1...3) = 0,44,15
C
C 2. INPUTS   - SAME AS EXAMPLE 1. EXCEPT  JOB = 1
C    OUTPUTS - IANS(1...3) = 0,44,15
C              SPT(1...15) = 0.000  -2.828   4.000   (STORED BY COLUMNS)
C                            2.828   8.000  -8.485
C                           12.000   8.485   0.000
C                            2.828  -4.000   8.485
C                            0.000   2.828  -4.000
C
C 3. INPUTS   - JOB = 1  NRA = 7  NCA = 7
C              AA(1...28) = 0.0  1.0  0.0  -2.0      (STORED BY COLUMNS)
C                           0.0  0.0  0.0   0.0
C                           0.0  2.0  0.0  -1.0
C                           0.0  0.0  0.0   0.0
C                           0.0  2.0  0.0   1.0
C                           0.0  0.0  0.0   0.0
C                           0.0  1.0  0.0   2.0
C              MRS = 4  JMAXR = 2  MCS = 4  JMAXC = 2
C    OUTPUTS - IANS(1...3) = 0,44,28
C              SPT(1...15) SAME AS EXAMPLE 2.
C
C 4. INPUTS   - JOB = -1  NRA = 5  NCA = 5
C              AA(1...15) = 0.1  0.2  0.1            (STORED BY COLUMNS)
C                           0.4  0.3  0.2
C                           0.0  0.4  0.1
C                          -0.4  0.5  0.0
C                          -0.1  0.6 -0.1
C              MRS = 2  JMAXR = 2  MCS = 2  JMAXC = 2
C    OUTPUTS - IANS(1...3) = 0,28,15
C              SPT(1...15) = 0.0  0.8  0.0           (STORED BY COLUMNS)
C                            0.8 -0.4  1.6
C                            0.0  4.0  0.0
C                           -0.8 -1.2 -1.6
C                            0.0  0.8  0.0
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(2),SPT(2),IANS(3),CM(2)
      COMMON CM
C
```

****
* PLANSP *
****
(PAGE 4)

```
C BRING IN VARIABLES
C
      CALL SETKS (JOB,JB,NRA,NRT,NCA,NCT,MRS,MR,MCS,MC,JMAXR,JXR,
     1 JMAXC,JXC)
      CALL LIMITS (1,IANS, JB,-1,1, NRT,1,32768,  NCT,1,32768,
     1  MR,1,32767,  MC,1,32767,  JXR,1,MR,  JXC,1,MC)
      IF (IANS) 10,10,1010
 10   CONTINUE
      CALL IXCARG (SPT,ISPT)
      CALL IXCARG (SPACE1,ISP1)
      CALL IXCARG (SPACE2,ISP2)
C
C MOVE XX TO SPACE1 VIA SPACE2, INSERTING ZEROS IF NEEDED
C
      CALL CHOOSE (XOOZEF(NRT),MR,MR+MR,MR,NR,NRT-1,NRT/2,IS,2,1,
     1 NRTA,NRT+NRT-1,NRT)
      CALL CHOOSE (XOOZEF(NCT),MC,MC+MC,MC,NC,NCT-1,NCT/2,NRTB,
     1 NRTA+NRTA,NRTA)
     NCTA=(NCT+1)/2
     LX=NCTA*NRT
     IF (JB) 3000,3010,3000
 3000 CONTINUE
     ISP2A=LX-NRT+ISP2
     LX1=NRTB*NCTA
     ISP1A=LX1-NRTA+ISP1
      CALL MOVREV (LX,1,AA,1,CM(ISP2),1)
      CALL STZ (LX1,CM(ISP1))
     DO 20 I=1,NCTA
      CALL MOVREV (NRT,1,CM(ISP2A),IS,CM(ISP1A),1)
     ISP2A=ISP2A-NRT
 20   ISP1A=ISP1A-NRTB
 2000 Q=1.
 3010 CONTINUE
C DECIDE WHICH ORDER OF COMPUTATION IS FASTEST
     NR1=XMINOF(NR,MR)
     NC1=XMINOF(NC,MC)
     ICH=(JXR-JXC)*NR1*NC1+(NC1-NR1)*JXR*JXC
      IF (ICH)  100,120,120
C PRESENT ORDER IS FASTEST, SHOULD WE ROTATE
 100  CONTINUE
      IF (MC-NC)  110,130,130
C ROTATE AND GO COLLAPSE
 110  CONTINUE
      IF (JB) 3020,3030,3020
 3020 CONTINUE
      CALL ROAR2 (JB,SPACE1,NR,NC,SPACE1)
 3030 CONTINUE
      GO TO 140
C OPPOSITE ORDER IS FASTEST, ROTATE PARAMETERS
 120  CONTINUE
      CALL SETKS (NR,I,NC,NR,I,NC,MR,I,MC,MR,I,MC,
     1 JXR,I, JXC,JXR, I,JXC)
 130  CONTINUE
 140  CONTINUE
      CALL XADDKS (1,NC,NC1, NC,NC1,NCNC1, 1,NR,NR1, NR,NR1,NRNR1,
     1 MR,MR+1,MRMR1, MC,MC+1,MCMC1, 1,JXR,LSR1, JXR,LSR1,LSR2,
     2 1,JXC,LSC1, JXC,LSC1,LSC2)
C COLLAPSE ROWS IF VALID
      IF (MC-NC)  160,150,150
C IT IS NOT VALID
 150  CONTINUE
C SHOULD WE ROTATE
      IF (ICH)  190,180,180
C IT IS VALID TO COLLAPSE
 160  CONTINUE
      IF (JB) 3040,3050,3040
 3040 CONTINUE
      CALL XADDKS (NC,ISP1,ISM, 0,ISP1,IS1, -MC,ISM,IS2)
      DO 170  I1=1,NR1
      CALL KOLAPS (CM(ISM),NC,1.,MC,CM(ISM),ERR1)
      CALL MOVREV(MCMC1,1,CM(IS2),1,CM(IS1),1)
     ISM=ISM+NCNC1
     IS1=IS1+MCMC1
 170  IS2=IS2+NCNC1
 3050 CONTINUE
```

```
      CALL XADDKS (0,MC,NC, 1,MC,NC1, NC,NC1,NCNC1)
C REROTATE BACK AGAIN
 180  CONTINUE
      IF (JB) 3060,3070,3060
 3060 CONTINUE
      CALL ROAR2 (JB,SPACE1,NC,NR,SPACE1)
 3070 CONTINUE
 190  CONTINUE
C COLLAPSE COLUMNS IF VALID TO DO SO
      IF (MR-NR)  200,220,220
C IT IS VALID
 200  CONTINUE
      IF (JB) 3080,3090,3080
 3080 CONTINUE
      CALL XADDKS (NR,ISP1,ISM, 0,ISP1,IS1, -MR,ISM,IS2)
      DO 210  I2=1,NC1
      CALL KOLAPS (CM(ISM),NR,1.,MR,CM(ISM),ERR2)
      CALL MOVREV (MRMR1,1,CM(IS2),1,CM(IS1),1)
      ISM=ISM+NRNR1
      IS1=IS1+MRMR1
 210  IS2=IS2+NRNR1
 3090 CONTINUE
      CALL XADDKS (0,MR,NR, 1,NR,NR1, NR,NR1,NRNR1)
 220  CONTINUE
C TRANSFORM COLUMNS, BEGINNING WITH LAST ONE.
      IC=ISP1+3+XMAXOF(LSR1*NC1+XMAXOF(NRNR1,NCNC1+LSC2),NRNR1*NC1+LSR1)
      IS=ISP2+XMAXOF(LSR1*NC1,LSC2*LSR1)
      IX=ISP1+NC*NRNR1
      ICOS=IC
      ISIN=ICOS+XMAXOF(MR,MC)+1
      IF (JB) 3100,1000,3100
 3100 CONTINUE
      CALL COSTBL (MR,CM(ICOS))
      CALL SINTBL (MR,CM(ISIN))
 2100 Q=1.
      DO 260  I3=1,NC1
      CALL XADDK (-LSR1,IC,IS)
      CALL COSIS1 (3,CM(IX),NRNR1,CM(ICOS),CM(ISIN),MR,0,JXR,
     1             CM(IC),CM(IS),0.,CM(IX),IAN1)
 260  IX=IX-NRNR1
C TRANSPOSE THE SINE AND COSINE VALUES
      CALL MATRA (CM(IC),LSR1,NC1,CM(IC))
      CALL MATRA (CM(IS),LSR1,NC1,CM(IS))
 2110 Q=1.
C TRANSFORM THE SINES, THEN THE COSINES.
      IF (MR-MC)  270,280,270
 270  CONTINUE
      CALL COSTBL (MC,CM(ICOS))
      CALL SINTBL (MC,CM(ISIN))
 280  CONTINUE
      CALL XADDKS (LSC2,ISP1,IS1, NC,IS1,IS2, 0,ISPT,ISA, JXC,ISP1,ICA1,
     1 0,(JB+3)/2,JB2, 0,-(JB-3)/2,JB3)
CXXXXXXXXX
      DO 325  I4=1,LSR1
      CALL SETKS (-1,M1, -JB,M2, JB2,JB1, ISA+JXC,ISC1, ISA,ISC, IS,ICS)
CXXXXXXXXXXXXXXXXXXX
      DO 310  I5=1,2
      CALL MOVREV (NC1,1,CM(ICS),1,CM(IS2),M1)
      CALL MOVREV (NC,1,CM(IS2+1),-1,CM(IS1),M2)
      CALL COSIS1 (JB1,CM(IS1),NCNC1,CM(ICOS),CM(ISIN),MC,0,JXC,
     1             CM(ISC1),CM(ISC1),0.,CM(IS1),IAN2)
      CALL MOVREV (JXC,1,CM(ISC1+1),-1,CM(ISC),M2)
      CALL SETKS (JB,M1, JB,M2, JB3,JB1, ICA1,ISC1, ISP1,ISC, IC,ICS)
 310  CONTINUE
CXXXXXXXXXXXXXXXXXXX
CXXXXX
      DO 320  I6=1,LSC2
      CM(ISA)=CM(ISA)+CM(ISC)
 320  CALL XADDK (1,ISA,ISC)
CXXXXX
 2200 Q=1.
 325  CALL XADDK (NC1,IC,IS)
CXXXXXXXXX
C ANSWERS NEED TRANSPOSING IF INITIAL DATA WAS IN PROPER ORDER
      IF (ICH)  330,340,340
```

****
*   PLANSP   *
****
(PAGE 6)

```
  330 CONTINUE
      CALL ROAR2 (JB,CM(ISPT),JXC,JXR,CM(ISPT))
  340 CONTINUE
 1000 CONTINUE
      IANS(2)=ISIN+ISIN-ICOS-ISP1
      IANS(3)=XMAXOF(IS-ISP2,LX)
 1010 CONTINUE
      RETURN
      END
```

```
*     PLOTVS (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0260
*     LABEL
CPLOTVS
      SUBROUTINE PLOTVS(ITAPE,ISENSE,LOCYV,YSMBV,LYV,IXSTRV,NY,ARGLO,
     1  ARGDEL,ZFAFXD,FMTARG,NCOLS,YBOT,YTOP,HLINV,HLSMBV,NHL)
C
C                     ----ABSTRACT----
C
C TITLE - PLOTVS
C        PRINTER-PLOT OF ARBITRARY SET OF VECTORS
C
C              PLOTVS MAKES A SIMULTANEOUS PRINTER PLOT, OFF-LINE AND/OR
C              ON-LINE, OF AN ARBITRARY NUMBER OF VECTORS, FOR VIEWING
C              WITH THE PAGE ROWS VERTICAL USING ONE ROW FOR EACH VECTOR
C              INDEX.  EACH VECTOR HAS ITS OWN LENGTH AND THE USER
C              FURTHER CONTROLS WHICH ROW THE PLOTTING OF EACH VECTOR IS
C              TO BEGIN IN, WHAT CHARACTER IS TO BE USED FOR EACH
C              VECTOR, THE NUMERICAL LABELLING OF EACH ROW, THE NUMBER
C              OF COLUMNS THE PLOT IS TO OCCUPY, THE VECTOR VALUES
C              ASSOCIATED WITH THE FIRST AND LAST COLUMNS, AND THE
C              POSITIONS AND CHARACTERS OF ANY HORIZONTAL LINES WHICH
C              MAY BE DESIRED.
C
C              THE ON-LINE PLOT OPTION MAY BE EITHER DEFINITE OR IN
C              THE FORM OF MONITORING UNDER SENSE SWITCH CONTROL.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094 MAIN FRAME PLUS ONE TAPE DRIVE (OPTIONAL),
C                PLUS ON-LINE PRINTER (OPTIONAL),
C                PLUS ONE SENSE SWITCH (OPTIONAL)
C STORAGE    - 494 REGISTERS
C SPEED      - THE PLOT, OVER  100  COLUMNS, OF  8  VECTORS OF LENGTH
C                 300  WITH  4  HORIZONTAL LINES TAKES ABOUT  3.2
C                 SECONDS ON THE  7094 .
C AUTHOR     - S.M. SIMPSON, MARCH 1964
C
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - RND,SETKS,SETKV,SETVEC,SWITCH
C       AND FORTRAN SYSTEM ROUTINES - (FIL),(SPH),(STH)
C
C FORTRAN USAGE
C     CALL PLOTVS(ITAPE,ISENSE,LOCYV,YSMBV,LYV,IXSTRV,NY,ARGLO,ARGDEL,
C    1            ZFAFXD,FMTARG,NCOLS,YBOT,YTOP,HLINV,HLSMBV,NHL)
C
C
C INPUTS
C
C    ITAPE     IS OUTPUT TAPE NUMBER.  = 0 IF NO TAPE OUTPUT.
C
C    ISENSE    IS A SENSE SWITCH NO. WHICH, IF ON, WILL GIVE ON LINE
C                MONITORING OF OUTPUT WHILE DEPRESSED.
C              IF = 0 OR NEGATIVE NO SENSE SWITCH TEST IS MADE, NO
C                ON LINE OUTPUT.
C                 = 7 OR GREATER GIVES FULL ON LINE OUTPUT, WHETHER OR
C                NOT THERE IS TAPE OUTPUT.
C
C    LOCYV(I)  I = 1...NY  IS A VECTOR OF MACHINE LOCATIONS OF THE Y
C                SERIES TO BE PLOTTED.
C
C    YSMBV(I)  I = 1...NY  IS A VECTOR GIVING SYMBOLS (EACH IN FORMAT
C                (A1)) FOR PLOTTING CORRESPONDING Y VALUES.
C
C    LYV(I)    I = 1...NY  ARE THE VECTOR LENGTHS.
C
C    IXSTRV(I) I = 1...NY GIVES THE ROW INDEX AT WHICH THE PLOTTING OF
C                THE CORRESPONDING VECTOR IS TO START.
C
C    NY        IS THE NO. OF VECTORS.
C
C    ARGLO     IS AN ARGUMENT VALUE ASSOCIATED WITH Y(1).
C
C    ARGDEL    IS AN INCREMENT SUCH THAT ARG(Y(K+1))=ARG(Y(K))+ARGDEL
C
```

```
C    ZFAFXD    = 0.0 IF ARGLO AND ARGDEL ARE FIXED POINT.
C              NOT= 0.0  IF ARGLO AND ARGDEL ARE FLOATING PUINT.
C
C    FMTARG    IS 6 HOLLERITH FORMAT (WITHOUT PARENTHESES) FOR PRINTING
C                ARGUMENTS.  THESE WILL BE PRINTED STARTING IN COLUMN 2.
C
C    NCOLS     IS WIDTH OF PLOTTING FIELD IN COLUMNS.  THE FIRST COLUMN
C                IMMEDIATELY FOLLOWS THE LAST ONE USED IN PRINTING ARG,
C                THE LAST COLUMN USED IS NCOLS-1 BEYOND THE FIRST COLUMN
C
C    YBOT      IS VALUE OF Y ASSOCIATED WITH FIRST COLUMN OF PLOTTING
C                FIELD.
C
C    YTOP      IS VALUE OF Y ASSOCIATED WITH LAST COLUMN OF PLOTTING
C                FIELD.
C                VALUES OF Y OUTSIDE THESE LIMITS ARE IGNORED.
C                YTOP MAY BE LESS THAN OR GREATER THAN YBOT.
C
C    HLINV(I)  I = 1...NHL  IS A VECTOR OF Y VALUES AT WHICH HORIZONTAL
C                LINES ARE TO BE DRAWN  (HORIZONTAL WHEN PAGE IS VIEWED
C                WITH COLUMNS HORIZONTAL).
C
C    HLSMBV(I) I = 1...NHL IS VECTOR OF SYMBOLS (EACH IN FORMAT (A1)) OF
C                THE CORRESPONDING HORIZONTAL LINES.
C
C    NHL       IS NO. OF DESIRED HORIZONTAL LINES.  NHL MAY BE ZERO, IN
C                WHICH CASE BOTH HLINV AND HLSMBV ARE IGNORED.
C
C              IN CASE OF CRISS-CROSS OF TWO OR MORE Y VECTORS THE
C              ASTERISK (*) SYMBOL IS USED AT THE INTERSECTION.  IF A Y
C              VECTOR INTERSECTS A HORIZONTAL LINE THE Y VECTOR SYMBOL
C              IS USED.
C
C
C EXAMPLES
C
C 1. INPUTS  - Y1(1...13) = 130.,140.,...,250.
C              Y2(1...8) = 50.,80.,...,260.
C              Y3(1...4) = 280.,240.,...,160.
C              LOCYV(1...3) = XLOCF(Y1),XLOCF(Y2),XLOCF(Y3)
C              YSMBV(1...3) = 1HA,1HB,1HC     LYV(1...3) = 13,8,4
C              IXSTRV(1...3) = 3,5,18      NY = 3     IARGLO = 0
C              IARGDL = 1      ZFAFXD = 0.0     FMTARG = 2HI2
C              NCOLS = 31    YBOT = 0.0     YTOP = 300.0
C              HLINV(1...4) = 0.,100.,200.,300.
C              HLSMBV(1...4) = 1HZ,1H.,1H.,1H.     NHL = 4
C
C    USAGE   -        CALL PLOTVS(2,7,LOCYV,YSMBV,LYV,IXSTRV,NY,IARGLO,
C                    1           IARGDL,ZFAFXD,FMTARG,NCOLS,YBOT,YTOP,
C                    2           HLINV,HLSMBV,NHL)
C    OUTPUTS - 21  ROWS ARE PRINTED ON-LINE AND OFF-LINE FROM LOGICAL 2,
C              OCCUPYING COLUMNS  1  THRU  34  (COLUMN  1  IS BLANK), AS
C              SHOWN BEGINNING  2  LINES BELOW.
C
C               0Z         .         .         .
C               1Z         .         .         .
C               2Z         .  A      .         .
C               3Z         .   A     .         .
C               4Z    B    .    A    .         .
C               5Z       B .     A   .         .
C               6Z         .B     A  .         .
C               7Z         .   B   A .         .
C               8Z         .      B A.         .
C               9Z         .         *         .
C              10Z         .         .A B      .
C              11Z         .         . A   B   .
C              12Z         .         .  A      .
C              13Z         .         .   A     .
C              14Z         .         .    A    .
C              15Z         .         .         .
C              16Z         .         .         .
C              17Z         .         .       C .
C              18Z         .         .   C     .
C              19Z         .         C         .
```

```
C              20Z        .    C    .         .
C
C
C PROGRAM FOLLOWS BELOW.
C
      DIMENSION LOCYV(2),YSMBV(2),LYV(2),IXSTRV(2),HLINV(2),HLSMBV(2)
      DIMENSION FMT(4),STAGE(131),COM(2)
      EQUIVALENCE (ARG,IARG)
      COMMON COM
C
C SET PLOTTING FORMAT,ARG,IARGD,FLOATF(NCOLS),IXROW=1,SPACES,NROWS=0
C
      CALL SETVEC(FMT,4H(1X,,FMTARG,6H,131A1,1H))
      CALL SETKS(ARGLO,ARG,  ARGDEL,IARGD, FLOATF(NCOLS),FNC,
     1   1,IXROW,  6H      ,SPACES,  0,NROWS)
      SCALE = (FNC-1.0)/(YTOP-YBOT)
C
C FIGURE OUT THE TOTAL NO. OF ROWS TO BE PLOTTED
C
      DO 50   IXY=1,NY
 50   NROWS = XMAXOF(NROWS,LYV(IXY)+IXSTRV(IXY)-1)
      IF (NROWS)   9999,9999,100
C
C BEGIN PROCESSING FOR NEXT LINE OF OUTPUT
C START BY CLEARING THE STAGING AREA
C
 100  CALL SETKV(6H      ,131,STAGE)
C
C THEN SET UP CHARACTERS FOR THE YS
C
      DO 170   IXY=1,NY
C
C CHECK IF THIS ROW CONTAINS THE VECTOR
C
      IXSTRT = IXSTRV(IXY)
      IF (IXSTRT-IXROW)  110,120,170
 110  IF (IXROW-IXSTRT-LYV(IXY)+1)   120,120,170
C
C OK, IT DOES
C
C (32561=COMMON BASE 10, =77461 BASE 8)
C
 120  IXCOM = 32562-LOCYV(IXY)-IXSTRT+IXROW
      Y = COM(IXCOM)
      ASSIGN 130 TO IEXCON
      GO TO 700
 130  IF (IXSTAG)   170,170,140
C
C SET THE CHARACTER (OR * IF A REPEAT) BUT IGNORE IF CHARACTER IS BLANK
C
 140  IF (YSMBV(IXY)-SPACES)   145,170,145
 145  IF (STAGE(IXSTAG)-SPACES)   150,160,150
 150  STAGE(IXSTAG) = 1H*
      GO TO 170
 160  STAGE(IXSTAG) = YSMBV(IXY)
 170  CONTINUE
C
C BEGIN PROCESSING OF HORIZONTAL LINES IF ANY
C
 200  IF  (NHL)   400,400,210
 210  DO 260   IXHL=1,NHL
      Y = HLINV(IXHL)
      ASSIGN 230 TO IEXCON
      GO TO 700
 230  IF (IXSTAG)   260,260,240
C
C DONT DISTURB A PREVICUS SETTING
C
 240  IF (STAGE(IXSTAG)-SPACES)   260,250,260
 250  STAGE(IXSTAG) = HLSMBV(IXHL)
 260  CONTINUE
C
C FINALLY PRINT THE LINE
C
```

****
* PLOTVS *
****
(PAGE 4)

```
C PRINT FIRST OFF-LINE IF REQUESTED
C
 400  IF (ITAPE)   420,420,410
 410  WRITE OUTPUT TAPE ITAPE,FMT,ARG,(STAGE(I),I=1,NCOLS)
C
C AND THEN ON-LINE IF REQUESTED
C
 420  IF (ISENSE-6)              430,430,450
 430  IF (SWITCHF(ISENSE))       460,460,450
 450  PRINT FMT,ARG,(STAGE(I),I=1,NCOLS)
C
C INCREMENT ARG AND GO BACK FOR MORE UNLESS DONE
C
 460  IF (ZFAFXD)   480,470,480
 470  IARG = IARG+IARGD
      GO TO 490
 480  ARG = ARG+ARGDEL
 490  IXROW = IXROW+1
      IF (IXROW-NROWS)  100,100,9999
C
C INTERNAL ROUTINE TO CONVERT A Y TO AN IXSTAG
C
C ENTER AT 700,  LEAVE THRU IEXCON
C
C     Y = YBOT GIVES IXSTAG = 1
C     Y = YTOP GIVES IXSTAG = NCOLS
C
 700  IXSTAG = 0
      FIXSTG = 1.0+(Y-YBOT)*SCALE
      IF (FIXSTG)  730,730,710
 710  IF (FIXSTG-.5-FNC)  720,730,730
 720  IXSTAG = RNDF(FIXSTG)
 730  GO TO IEXCON,(130,230)
C
C EXIT
C
 9999 RETURN
      END
```

```
*     PLTVS1 (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0392
*     LABEL
CPLTVS1
      SUBROUTINE PLTVS1(ITAPE,ISENSE,ARGLO,ARGDEL,ZFAFXD,NCOLS,ZFZERS,
     1                 RMSSEP,S,LX,ZFLIST,VMATRX,IDIMEN,NX)
C
C
C
C                   ----ABSTRACT----
C
C TITLE - PLTVS1
C        PRINTER PLOT OF A SET OF EQUAL LENGTH VECTORS
C
C               PLTVS1  MAKES A SIMULTANEOUS PRINTER PLOT, OFF-LINE WITH
C               ON-LINE OPTION, OF UP TO  35  EQUAL LENGTH VECTORS, THE
C               VECTORS BEING ALIGNED AND ORIENTED FOR VIEWING WITH THE
C               PAGE ROWS VERTICAL, USING ONE ROW FOR EACH VECTOR INDEX.
C               THE VECTORS ARE SPECIFIED EITHER BY A LIST OR AS THE
C               COLUMNS OF A MATRIX.  PRIOR TO PLOTTING  PLTVS1  SCALES
C               ALL THE VECTORS SO THAT EACH HAS UNIT RMS VALUE, SO THAT
C               THEIR ZERO LEVELS WILL BE EQUALLY SPACED ON THE PRINTED
C               PAGE, THE SPACING BEING CONTROLLED BY USER (MAY BE ZERO),
C               AND SO THAT THE COMPLETED GRAPH WILL JUST FIT INTO A
C               SPECIFIED NUMBER OF COLUMNS WITHOUT LOSS OF ANY DATA
C               VALUES.  THE VECTORS ARE UN-SCALED AFTER THE PLOTTING.
C               A TABLE GIVING CHARACTER ASSIGNMENTS AND EXTREMAL VALUES
C               OF EACH VECTOR IS PRINTED PRIOR TO THE PLOT.  FURTHER
C               CONTROLS BY THE USER INCLUDE THE NUMERICAL LABELLING OF
C               THE ROWS, THE INCLUSION OR EXCLUSION OF ZERO-LEVEL LINES,
C               THE OMISSION OF THE PLOTTING OF SELECTED DUMMY VECTORS
C               FOR SPACING PURPOSE, AND THE OPTION OF ON-LINE MONITORING
C               UNDER SENSE SWITCH CONTROL.
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  MAIN FRAME PLUS ONE TAPE DRIVE AND/OR THE
C                   ON-LINE PRINTER, PLUS ONE SENSE SWITCH (OPTIONAL)
C STORAGE    - 817 REGISTERS
C SPEED      - PLOTS  5  VECTORS OF LENGTH  33  OVER  50  COLUMNS IN
C                   ABOUT  .42  SECONDS ON THE  7094 .  FOR  15  VECTORS
C                   OF LENGTH  50  OVER  100  COLUMNS IT TAKES ABOUT  1.1
C                   SECONDS.
C AUTHOR     - S.M.SIMPSON, MARCH 1964
C
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -  BOOST,DPRESS, MAXSN, MINSN,
C                                      MULPLY,PLOTVS,RMSDEV, SETKS,
C                                      SETKVS,SETVEC,VARARG, XSAME,XSTLIN
C        AND FORTRAN SYSTEM ROUTINES -  (FIL), (STH),XLOC
C
C
C FORTRAN USAGE
C      CALL PLTVS1(ITAPE, ISENSE, ARGLO, ARGDEL, ZFAFXD, NCOLS, ZFZERS,
C     1              RMSSEP, S, LX, ZFLIST, VMATRX, IDIMEN, NX)
C OR
C      CALL PLTVS1(ITAPE, ISENSE, ARGLO, ARGDEL, ZFAFXD, NCOLS, ZFZERS,
C     1              RMSSEP, S, LX, ZFLIST, X1, X2, X3, ...., XNX)
C
C
C INPUTS
C
C    ITAPE      IS LOGICAL TAPE NUMBER FOR OUTPUT.
C               SHOULD LIE IN CLOSED RANGE  1  TO  20  (NOT CHECKED).
C
C    ISENSE     GRTHN= 7  GIVES CONTINUOUS ON-LINE OUTPUT
C               LSTHN= 0  SUPPRESSES ALL ON-LINE OUTPUT.
C               = 1,2,3,4,5, OR 6  RESULTS IN ON-LINE OUTPUT WHILE AND
C                  AND ONLY WHILE THE CORRESPONDING SENSE SWITCH IS
C                  DEPRESSED (ON).
C
C    ARGLO      IS THE NUMERICAL VALUE BY WHICH TO LABEL THE FIRST OUTPUT
C                  ROW.
C
```

```
C    ARGDEL    IS THE NUMERICAL INCREMENT BETWEEN LABELLINGS OF
C                 SUCCESSIVE ROWS.
C
C    ZFAFXD    =    0.0  IMPLIES  ARGLO  AND  ARGDEL  ARE FIXED POINT.
C              NOT= 0.0  IMPLIES THEY ARE FLOATING POINT.
C                 FIXED    LABELS ARE PRINTED IN COLUMNS  2  THRU   5
C                    FORMAT(I4)).
C                 FLOATING LABELS ARE PRINTED IN COLUMNS  2  THRU  13
C                    (FORMAT(E12.5)).
C
C    NCOLS     SPECIFIES THAT THE PLOTTING FIELD SHALL OCCUPY COLUMNS
C                 L+1  THRU  L+NCOLS  WHERE  L = 5  OR  13  ACCORDING TO
C                 ZFAFXD.
C              SHOULD NOT EXCEED NO. COLUMNS ON PRINTER  - L  (NOT
C                 CHECKED).
C
C    ZFZERS    =    0.0  REQUESTS THAT ZERO LEVELS OF THE VECTORS BE
C                 PLOTTED AS STRAIGHT LINES.
C              NOT= 0.0  REQUESTS SUPPRESSION OF ZERO LEVEL PLOTS.
C
C    RMSSEP    SPECIFIES THE PLOTTING SEPARATION BETWEEN ZERO LEVELS OF
C                 THE SUCCESSIVE VECTORS IN UNITS OF THEIR COMMON
C                 (AFTER SCALING) ROOT-MEAN-SQUARE VALUE OF  1.0 .
C                 EXAMPLE  -  AN  RMSSEP  OF  2/.707  WOULD CAUSE THE
C                                 PLOTS OF  COS(W)  AND  -COS(W)  TO GRAZE
C                                 EACH OTHER AT ODD MULTIPLES OF  PI.
C              MUST BE  GRTHN=  0.0  (NOT CHECKED).
C
C    S(I)       I=1...300  MUST BE AVAILABLE FOR SCRATCH.
C
C    LX        IS THE COMMON LENGTH OF THE VECTORS.
C              MUST EXCEED ZERO (NOT CHECKED).
C
C    ZFLIST    NOT= 0.0  SIGNIFIES THAT THE CALLER IS USING THE FIRST OF
C                 THE TWO POSSIBLE FORMS OF CALLING SEQUENCE (INVOLVING
C                 VMATRX,  IDIMEN,  NX)  IN WHICH THE VECTORS TO BE
C                 PLOTTED ARE THE COLUMNS OF A MATRIX.
C              =    0.0  SIGNIFIES THE USE OF THE SECOND FORM OF CALLING
C                 SEQUENCE IN WHICH THE VECTORS ARE SPECIFIED BY A LIST.
C
C    VMATRX(I,J)  I=1...LX,  J=1...NX  CONTAINS, FOR  ZFLIST  NOT=  0.0,
C                 THE  NX  VECTORS
C                     X1(1...LX)  =  VMATRX(1...LX,1)
C                     X2(1...LX)  =  VMATRX(1...LX,2)
C                                  ETC.
C                    XNX(1...LX)  =  VMATRX(1...LX,NX)
C
C    IDIMEN    IS THE CALLER'S DIMENSION OF THE INDEX  I  IN
C                 VMATRX(I,J).
C              MUST BE  GRTHN=  LX  (NOT CHECKED).
C
C    NX        MUST BE  GRTHN=  1  AND  LSTHN=  35  (NOT CHECKED).
C
C    X1,X2,...,XNX  ARE, FOR  ZFLIST =0.0,  THE  NX  VECTORS TO BE
C                 PLOTTED.
C
C
C OUTPUTS      A TABLE GIVING MAXIMA AND MINIMA OF THE VECTORS PLUS
C              THEIR PLOTTING CHARACTERS IS PRINTED (OFF-LINE ONLY)
C              FOLLOWED BY A PAGE RESTORE AND THE PLOT PROPER.  THE
C              VECTORS ARE PLOTTED WITH ZERO LEVELS SEPARATED (IF
C              RMSSEP  GRTHN  0.0) SO THAT  X1  IS CLOSEST TO TOP OF
C              PLOT (HIGH COLUMN NUMBERS) AND  XNX  CLOSEST TO BOTTOM.
C              THE CHARACTERS FOR  X1,X2,...  ARE TAKEN SUCCESSIVELY
C              FROM THE LIST  1,2,...,9,A,B,...,Z.  IF ANY  X  VECTOR
C              HAS NOTHING BUT ZERO ELEMENTS ITS PLOTTING IS SUPPRESSED
C              BUT THE SUCCEEDING VECTOR IS SPACED AS THOUGH THE ZERO
C              VECTOR WERE PRESENT.  ZERO LEVEL LINES (IF REQUESTED) ARE
C              PLOTTED WITH PERIODS, AND INTERSECTIONS OF CURVES ARE
C              INDICATED BY ASTERISKS.
C
C
C EXAMPLES
C
C 1. INPUTS  - THE FOLLOWING SEQUENCE SETS UP A MATRIX OF  5  VECTORS
```

```
C             WHICH ARE COSINE WAVES OF VARIOUS FREQUENCIES, EXCEPT
C             THAT TWO SPIKES ARE THROWN IN THE SECOND AND THIRD
C             VECTORS, AND THAT THE FOURTH VECTOR IS ZEROED TO
C             ILLUSTRATE SPACING.
C
C                   DIMENSION    V(50,5),X1(50),X2(50),X3(50),X4(50),
C                  1             X5(50),S(300)
C                   EQUIVALENCE  (X1,V),(X2,V(51)),(X3,V(101)),
C                  1             (X4,V(151)),X5,V(201))
C                   DO 10  J=1,5
C                   TENTHJ = .1*FLOATF(J)
C                   DO 10  I=1,33
C              10   V(I,J) = COSF(TENTHJ*FLOATF(I-1))
C                   V(5,3) = 7.0
C                   V(15,2) = -10.0
C                   DO 20  I=1,33
C              20   V(I,4) = 0.0
C
C    USAGE    -     CALL PLTVS1(2,1,0,1,0,50,1.,1.5,S,33,1.,V,50,5)
C                   CALL PLTVS1(2,1,0,1,0,50,0.,1.5,S,33,0.,X1,X2,X3,
C                  1            X4,X5)
C
C    OUTPUTS - THE TWO CALLS LEAD TO IDENTICAL OUTPUT ON LOGICAL  2
C             (ON-LINE MONITORING WITH SENSE SWITCH 1), EXCEPT THAT
C             ZERO LEVELS ARE PLOTTED ONLY FOR THE SECOND  CALL.  (IN
C             EACH CASE A 5-LINE TABLE, GIVING VECTOR NUMBER,
C             CHARACTER, MAX VALUE AND ITS INDEX, MIN VALUE AND ITS
C             INDEX, IS PRINTED IN COLUMNS  2  THRU  83  PRECEDING THE
C             PLOT.  THE GRAPH INCLUDING THE ZERO LEVELS IS SHOWN
C             BELOW.
C
C               0        .      5       .   3   .  2   .       1
C               1        .      5.      .   3   . 2    .       1
C               2        .  5   .       .  3    . 2    .      1
C               3        5      .       . 3     . 2    .      1
C               4     5  .      .       .       . 2    .      1  3
C               5  5     .      .       3       .2     .      1
C               6 5      .      .      3.       .2     .     1
C               7 5      .      .     3 .       2      .     1
C               8   5    .      .    3  .       2      .    1
C               9      5 .      .    3  .      2.      .    1
C              10        . 5    .   3   .      2.      .   1
C              11        .    5 .   3   .     2 .      .   1
C              12        .     5.    3  .     2 .      .  1
C              13        .      5    3  .    2  .      . 1
C              14   2    .    5 .     3 .       .      . 1
C              15        . 5    .      3.    2  .      .1
C              16       5.      .       3    2  .      1
C              17   5    .      .       .3   2  .      1
C              18 5      .      .       . 3  2  .     1.
C              19 5      .      .       .  3  2 .    1 .
C              20  5     .      .       .   3 2 .    1 .
C              21    5   .      .       .   3 2 .   1  .
C              22        5      .       .   3  2.  1   .
C              23        .  5   .       .  3   2.  1   .
C              24        .     5.       . 3     2 1    .
C              25        .      5       .3      .21    .
C              26        .     5.       3       .*     .
C              27        .   5  .      3.       .12    .
C              28        .5     .     3 .       .12    .
C              29     5  .      .    3  .       .12    .
C              30  5     .      .    3  .       1 2    .
C              31 5      .      .   3   .       1 2    .
C              32 5      .      .   3   .       1 2    .
C
C
C PROGRAM FOLLOWS BELOW
C
C
C
C USE OF SPACE VECTOR
C
C    S(1...35) = S(JYSM...) =
C              = 1H1,1H2,...,1H9,1HA,1HB,...,1HZ
C    S(40...50) = XLOCF(ITAPE),(ISENSE),...,(ZFLIST)
```

```
C     S(51...85) = S(JLOC...) = XLOCF(X1),(X2),...,(XN),0...  IF ZFLIST=0
C                             = XLOCF(VMATRX),(IDIMEN),0,... IF ZFLIST
C                                 NOT = 0
C     S(91...125) = S(JLYV...) = LX,LX,...
C     S(131...165) = S(JSTR...) = 1,1,1,...
C     S(171...205) = S(JHLN)...) = ZER(X1),ZER(X2),...,ZER(XN) TRANSLATED
C     S(211...245) = S(JRMS...) = RMS(X1),RMS(X2),...,RMS(XN)  ORIGINAL
C     S(251...275) = S(JHSM...) = 1H., 1H., ...
C
      DIMENSION S(2),C(2)
      COMMON C
C
C SET UP A LOCATION VECTOR
C
      CALL VARARG(S(40))
      GO TO 10
 9999 RETURN
C
C SET UP INDICES IN SPACE VECTOR  S,  SET UP SYMBOL VECTOR,
C      LENGTH VECTOR, STARTING POINT VECTOR,  L,  SEP .
C
 10   CALL SETKS(1,JYSM, 251,JHSM, 51,JLOC, 91,JLYV, 131,JSTR,
     1   171,JHLN, 211,JRMS, LX,L, RMSSEP, SEP)
      CALL SETVEC(S(JYSM),1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HA,1HB,
     1  1HC,2HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,
     2  1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ)
      CALL SETKVS(L,35,S(JLYV),  1,35,S(JSTR),  1H.,35,S(JHSM))
C
C SET N FOR THE CASE OF A LIST
C
      N = -1
 30   N = N+1
      IF (S(N+51))   30,40,30
C
C RECOMPUTE THE LOCATION VECTOR AND RESET N FOR A MATRIX INPUT
C
 40   IF (ZFLIST)  50,60,50
 50   N = NX
      CALL XSTLIN(XLOCF(VMATRX),-IDIMEN,N,S(JLOC))
C
C CHECK SOME ITEMS
C      (DUMMY AT PRESENT)
C
 60   CONTINUE
C
C NOW BEGIN SCAN OF VECTORS
C
 100  BORLO = 0.0
      BORHI = 0.0
      EDGE = .1
      DO 150  IXX=1,N
C
C FIND IXC = INDEX WRT COMMON OF X SUB IXX
C
      ITEMP = JLOC+IXX-1
      TEMP = S(ITEMP)
      IXC = 32562-XSAMEF(TEMP)
C
C FIND RMS OF X AND STORE IN S(JRMS+IXX-1)
C
      CALL RMSDEV(C(IXC),L,0.,RMS)
      ITEMP = JRMS+IXX-1
      S(ITEMP) = RMS
C
C FIND THE CHARACTER USED FOR THIS VECTOR
C
      ITEMP = JYSM+IXX-1
      CHAR = S(ITEMP)
C
C IF THE RMS VALUE IS ZERO MAKE A
C SPECIAL COMMENT AND SKIP TO NEXT VECTOR CHANGING CHARACTER TO BLANK
C
      IF (RMS)   120,120,130
 120  WRITE OUTPUT TAPE ITAPE, 122,IXX
 122  FORMAT(8H VECTOR ,I2,39H IS IDENTICALLY ZERO, PLOTTING OMITTED.)
```

PLTVS1
(PAGE 5)

```
B     S(ITEMP) = 606060606060
      GO TO 150
C
C OTHERWISE FIND EXTREMAL VALUES AND PRINT THEM
C
 130  CALL MAXSN(L,C(IXC),XMAX,IXMAX)
      CALL MINSN(L,C(IXC),XMIN,IXMIN)
      WRITE OUTPUT TAPE ITAPE,135,IXX,CHAR,IXMAX,XMAX,IXMIN,XMIN
 135  FORMAT(8H VECTOR ,I2,12H, CHARACTER ,A1,13H, HAS MAX AT ,I4,
     1  3H = ,E12.5,9H, MIN AT ,I4,3H = ,E12.5)
C
C THEN SCALE XMAX XMIN AND THE VECTOR TO HAVE UNIT RMS
C
      SCALE = 1.0/RMS
      XMAX = SCALE*XMAX
      XMIN = SCALE*XMIN
      CALL MULPLY(C(IXC),L,SCALE,C(IXC))
C
C UPDATE THE TRIAL VALUES OF BORLO,BORHI
C
      TEMP = MAX1F(0.,-XMIN)
      BORLO = MAX1F(BORLO,TEMP-SEP*FLOATF(N-IXX))
      TEMP = MAX1F(0.,XMAX)
      BORHI = MAX1F(BORHI,TEMP-SEP*FLOATF(IXX-1))
C
C END OF FIRST VECTOR SCAN
C
 150  CONTINUE
C
C WHEN DONE MAKE A CHECK THAT ALL VECTORS ARENT ZERO
C
      IF (BORLO+BORHI)  160,160,170
 160  WRITE OUTPUT TAPE ITAPE,165
 165  FORMAT(43H ALL VECTORS VANISH, NO PLOTTING WILL OCCUR)
      GO TO 9999
C
C NOW THAT WE HAVE BORLO AND BORHI, THE MEANS CAN BE ADJUSTED,
C AND THE ADDED CONSTANTS INSERTED INTO S(JHLN...)
C
 170  DO 180  IXX = 1,N
      ITEMP = JLOC+IXX-1
      TEMP = S(ITEMP)
      IXC = 32562-XSAMEF(TEMP)
      CONST = EDGE+BORLO+SEP*FLOATF(N-IXX)
      CALL BOOST(C(IXC),L,CONST,C(IXC))
      ITEMP = JHLN+IXX-1
      S(ITEMP) = CONST
 180  CONTINUE
C
C YTOP AND YBOT CAN BE SET NOW
C
      YBOT = 0.0
      YTOP = EDGE+BORLO+SEP*FLOATF(N-1)+BORHI+EDGE
C
C SKIP TO NEW PAGE
C
      WRITE OUTPUT TAPE ITAPE,200
 200  FORMAT(1H1)
C
C SET NHL ACCORDING TO ZFZERS
C
      NHL = 0
      IF (ZFZERS)  220,210,220
 210  NHL = N
C
C SET FMTARG ACCORDING TO ZFAFXD
C
 220  FMTARG = 2HI4
      IF (ZFAFXD)  230,240,230
 230  FMTARG = 5HE12.5
C
C THEN GO PLOT
C
 240  CALL PLOTVS(ITAPE,ISENSE,S(JLOC),S(JYSM),S(JLYV),S(JSTR),N,
```

PLTVS1
(PAGE 6)

```
     1  ARGLO,ARGDEL,ZFAFXD,FMTARG,NCOLS,YBOT,YTOP,S(JHLN),S(JHSM),NHL)
C
C THEN RESCALE THE VECTORS TO THEIR ORIGINAL VALUES
C
      DO 250  IXX=1,N
      ITEMP = JLOC+IXX-1
      TEMP = S(ITEMP)
      IXC = 32562-XSAMEF(TEMP)
      CONST = EDGE+BORLO+SEP*FLOATF(N-IXX)
      CALL DPRESS(C(IXC),L,CONST,C(IXC))
      ITEMP = JRMS+IXX-1
      RMS = S(ITEMP)
      CALL MULPLY(C(IXC),L,RMS,C(IXC))
 250  CONTINUE
C
C GO EXIT
C
      GO TO 9999
      END
```

```
************************
*    PLURAL            *
************************
 REFER TO
     SEVRAL
```


```
************************
*    PLURAL            *
************************
 REFER TO
     SEVRAL
```

```
*       PLURNS (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0246
*       FAP
*PLURNS
        COUNT   200
        LBL     PLURNS
        ENTRY   PLURNS (A1,A2,...,AN,B1,B2,...,BN,.......,Z1,Z2,...,ZN)
*                    OR
*                       (A1,A2,...,ANA,STOP,B1,B2,...,BNB,STOP,.......,
*                        Z1,Z2,...,ZNZ)
*
*                           ----ABSTRACT----
*
* TITLE - PLURNS
*         PLURALIZE THE NEXT SUBROUTINE
*
*                PLURNS IS A VARIABLE-LENGTH-CALLING SEQUENCE SUBROUTINE
*                WHOSE ARGUMENTS ARE DIVIDED INTO EQUAL LENGTH GROUPS OR
*                INTO ARBITRARY LENGTH GROUPS SEPARATED BY A FENCE-TYPE
*                ARGUMENT.  EACH SUCH GROUP REPRESENTS A SET OF ARGUMENTS
*                TO BE ASSOCIATED WITH THE SUBROUTINE WHOSE NAME APPEARS
*                IN A CALL STATEMENT IMMEDIATELY FOLLOWING THE CALL PLURNS
*                STATEMENT.  PLURNS THEN CALLS THAT SUBROUTINE ONCE FOR
*                EACH OF THESE GROUPS.  THE CALL SUBROUTINE STATEMENT HAS
*                EITHER ONE ARGUMENT OR NO ARGUMENTS.  IF ONE, AND
*                GREATER THAN ZERO, EQUAL LENGTH GROUPS ARE ASSUMED, WITH
*                LENGTH = ARGUMENT.  IF NONE, OR ONE WITH VALUE 0, THE
*                FENCE FORMAT IS ASSUMED.
*
*                LIMITATION  -  NONE OF THE ARGUMENTS IN A GROUP MAY BE
*                EXPRESSIONS INVOLVING OUTPUTS OF A PREVIOUS GROUP
*                EXCEPT FOR PURE EQUIVALENCES.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 73 REGISTERS
* SPEED       -
* AUTHOR      - S.M. SIMPSON,   OCTOBER 1963
*
*                        ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE FOR SUBROUTINES WITH FIXED, NON-ZERO ARGUMENT COUNT = N
*
*     CALL PLURNS(A1,A2,...,AN,B1,B2,...,BN,.......,Z1,Z2,...,ZN)
*     CALL SUBRU(N)
*
* IS EQUIVALENT TO
*
*     CALL SUBRU(A1,A2,...,AN)
*     CALL SUBRU(B1,B2,...,BN)
*          ETC
*     CALL SUBRU(Z1,Z2,...,ZN)
*
* FORTRAN USAGE FOR SUBROUTINES WITH VARIABLE OR ZERO ARGUMENT COUNTS
* EITHER
*     CALL PLURNS(A1,A2,...,ANA,STOP,B1,B2,...,BNB,STOP,.......,Z1,Z2,
*               ...,ZNZ)
*     CALL SUBRU(0)
* OR
*     CALL PLURNS(A1,A2,...,ANA,STOP,B1,B2,...,BNB,STOP,.......,Z1,Z2,
*               ...,ZNZ)
*     CALL SUBRU
*
* WHERE STOP = OCT 777777712345
*
* IS EQUIVALENT TO
*
*     CALL SUBRU(A1,A2,...,ANA)
*     CALL SUBRU(B1,B2,...,BNB)
*          ETC
*     CALL SUBRU(Z1,Z2,...,ZNZ)
*
```

```
* WHERE ONE OR MORE OF THE ARGUMENT COUNTS NA,NB,...,NZ MAY BE ZERO.
*
* IMPORTANT LIMITATION  -  IF X IS AN OUTPUT FROM ONE CALL SUBRU AND Y
* IS AN INPUT TO A SUCCEEDING CALL SUBRU, EQUIVALENCE (Y,X) IS THE ONLY
* MANNER IN WHICH THE Y EXPRESSION MAY INVOLVE THE X EXPRESSION.  THUS,
* IF X=I THEN Y=A(I) IS ILLEGAL, Y=2*I IS ILLEGAL, ETC.
*
*
* EXAMPLES
*              FOR ILLUSTRATION WE USE THE FOLLOWING SIMPLE SUBROUTINES
*                    SUBROUTINE ADD(X,Y,SUMXY)
*                    SUMXY=X+Y
*                    RETURN
*                    END
*              AND
*                    SUBROUTINE TELL
*                    WRITE OUTPUT TAPE 2,5
*                  5 FORMAT(21H TELL HAS BEEN CALLED)
*                    RETURN
*                    END
*
* 1. WITH SPECIFIED ARGUMENT COUNT
*    INPUTS  - X1,X2,X3,X4 = 1.,2.,3.,4.   Y1,Y2,Y3,Y4 = 2.,4.,6.,8.
*    USAGE   -       CALL PLURNS(X1,Y1,SUM1,X2,Y2,SUM2,X3,Y3,SUM3)
*                    CALL ADD(3)
*                    CALL PLURNS(X4,Y4,SUM4)
*                    CALL ADD(3)
*    OUTPUTS - SUM1,SUM2,SUM3,SUM4 = 3.,6.,9.,12.
*
* 2. WITH ZERO OR UNSPECIFIED ARGUMENT COUNT
*    INPUTS  - SAME AS EXAMPLE 1, PLUS STOP=OCT 777777712345
*    USAGE   -       CALL PLURNS(X1,Y1,S1A,STOP,X2,Y2,S2A,STOP,
*                              X3,Y3,S3A)
*                    CALL ADD(0)
*                    CALL PLURNS(X4,Y4,S4A)
*                    CALL ADD(0)
*                    CALL PLURNS(X1,Y1,S1B,STOP,X2,Y2,S2B,STOP,
*                              X3,Y3,S3B)
*                    CALL ADD
*                    CALL PLURNS(X4,Y4,S4B)
*                    CALL ADD
*    OUTPUTS - S1A=S1B=3.   S2A=S2B=6.   S3A=S3B=9.   S4A=S4B=12.
*
* 3. BEHAVIOUR ON A NO-ARGUMENT SUBROUTINE
*    INPUTS  - STOP = SAME AS EXAMPLE 2.
*    USAGE   -       CALL PLURNS(STOP,STOP)
*                    CALL TELL(0)
*                    CALL PLURNS
*                    CALL TELL(0)
*    OUTPUTS - THE FOLLOWING 4 LINES
*              TELL HAS BEEN CALLED
*              TELL HAS BEEN CALLED
*              TELL HAS BEEN CALLED
*              TELL HAS BEEN CALLED
*              WILL BE PRINTED OFF-LINE FROM LOGICAL 2
*
* 4. USAGE   -       CALL PLURNS(STOP,STOP)
*                    CALL TELL
*                    CALL PLURNS
*                    CALL TELL
*    OUTPUTS - SAME AS EXAMPLE 3.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
      HTR     0             XR1
      HTR     0             XR2
      HTR     0             XR4
      BCI     1,PLURNS
* ONLY ENTRY.  PLURNS(A1,A2,...,AN, B1,B2,...,BN,.......,Z1,Z2,...,ZN)
*    FOLLOWED BY CALL SUBRU(N)    N=NARGS
*
* OR (CASE 2)  PLURNS(A1,A2,...,ANA,STOP,B1,B2,...,BNB,STOP,.......,
*                    Z1,Z2,...,ZNZ)
```

```
*    FOLLOWED BY CALL SUBRU(O) OR CAL SUBRU  -  NO ARGUMENTS
PLURNS SXD     PLURNS-2,4
       SXD     PLURNS-3,2
       SXD     PLURNS-4,1
* SCAN DOWN FOR TSX $SUBRU,4
 CAL   CAL     1,4
       STA     TRAOUT         (ANTICIPATORY SETTING)
       ANA     AMASK
       LAS     TSXZ4
       TRA     *+2
       TXI     GOTSUB,4,-1   GOT IT (NOW AT 0,4)
* NOT YET
       TXI     CAL,4,-1
* THEN CHECK FOR CASE 1, OR CASE 2
GOTSUB TSX     TSXZCK,1       CHECK LOC(TSX $SUBRU,4)+1
       TRA     SXAZIF         CASE 2 IF NARGS UNSPECIFIED
* CASE 1.  CLEAR ZIFCA1, SET NARGS BUMPER, SET EXIT TO
*          LOC(TSX $SUBRU,4)+2  (HOWEVER, SWITCH TO CASE 2 IF NARGS=0)
       STZ     ZIFCA1
       TXI     *+1,4,-1       (ANTICIPATE A SWITCH)
       CLA*    0,4            NARGS
       PDC     0,2            -NARGS
       TXL     SXAZIF,2,0     (SWITCH TO CASE 2)
       SXD     TXI1,2         OK, NON-ZERO NARGS
       TRA     SXAAXT
* CASE 2.  SET ZIFCA1 NON ZERO, SET EXIT (VARIABLE)
SXAZIF SXA     ZIFCA1,4
SXAAXT SXA     AXTX,4
* INITIALIZE LOOP BY RESTORING ORIGINAL XR4
       LXD     PLURNS-2,4
* LOOP BEGINS.  XR4 IS USED TO FOOL THE SUBROUTINE.  XR2 IS USED TO FIND
* THE END OF ITS CALLING SEQUENCE, FOR LINKAGE.
 NEXT  ZET     ZIFCA1
       TRA     SXA2
* FOR CASE 1.  MOVE XR4 TO XR2 AND BUMP IT BY NARGS.  GO TO SUBROUTINE.
       PXA     0,4
       PAX     0,2
 TXI1  TXI     SETLNK,2,**    ** = -NARGS
* FOR CASE 2, USE XR4 TO SCAN FOR STOP OR END OF SEQUENCE, BUT SAVE XR4
* FOR LATER RESTORATION.
 SXA2  SXA     AXT2,4
TSXZC2 TSX     TSXZCK,1       FIRST CHECK FOR ANOTHER ARGUMENT
       TRA     PXA2           NO, TERMINATE SCAN
* IF 1,4 IS AN ARGUMENT, CHECK TO SEE IF THE ARGUMENT IS STOP.
       CAL*    1,4
       LAS     STOP
       TRA     *+2
-      TRA     PXA2           YES, TERMINATE SCAN
       TXI     TSXZC2,4,-1    NO, TRY AGAIN
* AFTER SCAN, MOVE NEW XR4 TO XR2 AND RESTORE OLD XR4
 PXA2  PXA     0,4
       PAX     0,2
 AXT2  AXT     **,4           ** = XR4 TO FOOL SUBROUTINE
* SET RETURN LINKAGE IN 1,2
SETLNK CLA     1,2
       STO     SAVNXT
       CLA     TRABAK
       STO     1,2
* GO OPERATE THE SUBROUTINE
       SXA     BACK,2
TRAOUT TRA     **             ** = A(TTR SUBRU)
* AFTERWARDS, RESTORE XR4 TO OLD XR2, RESTORE 1,4
 BACK  AXT     **,4           ** = XR2 BEFORE SUBROUTINE
       CLA     SAVNXT
       STO     1,4
* EXIT IF 1,4 IS NOT AN ARGUMENT
       TSX     TSXZCK,1
       TRA     LEAVE          NO
* IF MORE TO DO, INDEX XR4 BY 0(CASE 1) OR -1(CASE 2)
* AND RETURN FOR NEXT ARGUMENT SEQUENCE.
       ZET     ZIFCA1
       TXI     NEXT,4,-1      CASE2
       TRA     NEXT           CASE1
* EXIT
 LEAVE LXD     PLURNS-3,2
```

PLURNS

(PAGE 4)


```
        LXD     PLURNS-4,1
 AXTX   AXT     **,4            ** = -(A(TSX $SUBRU,4)+1) OR
*                                     -A(TSX $SUBRU,4)
        TRA     1,4
* INTERNAL SUBROUTINE TO CHECK IF 1,4 IS A TSX X,0
*       LINKAGE WITH XR1   DESTROYS AC
*     RETURNS TO 1,1 IF NOT
*                2,1 IF SO
TSXZCK  CAL     1,4
        ANA     AMASK
        LAS     TSXZ
        TRA     *+2
        TRA     2,1             YES
        TRA     1,1             NO
* CONSTANTS, TEMPORARIES
 AMASK  OCT     777777700000
 TSXZ   TSX     0,0
TRABAK  TRA     BACK
 TSXZ4  TSX     0,4
 STOP   OCT     777777712345
SAVNXT  PZE     **,**,**        TEMP FOR INSTRUCTION
ZIFCA1  PZE     **,0            ** = 0 IF CASE 1, NOT=0 IF CASE 2
        END
```

```
*     PLYSYN (SUBROUTINE)              10/5/64   LAST CARD IN DECK IS NO. 0161
*     LABEL
CPLYSYN
      SUBROUTINE PLYSYN(SCALES,RADII,DGREES,NROOTS,PLYCOS,NCOFS,SPACE)
C
C                    ----ABSTRACT----
C
C TITLE - PLYSYN
C       POLYNOMIAL SYNTHESIZED FROM ITS REAL AND COMPLEX ROOTS
C
C              GIVEN REAL ROOTS X(I) WITH REAL SCALE FACTORS U(I) WHERE
C              I RUNS FROM 1 TO M AND GIVEN COMPLEX ROOTS Y(J) WITH REAL
C              SCALE FACTORS V(J) WHERE J RUNS FROM 1 TO N, SUBROUTINE
C              POLYSYN COMPUTES THE REAL POLYNOMIAL COEFFICIENTS
C                  A(0),A(1),...,A(N+2M)
C              ACCORDING TO THE FORMULA
C
C                  A(0) + A(1)Z + A(2)Z**2 + ... + A(M+2N)Z**(M+2N) =
C
C                    M                     N
C                  PRODUCT U(I)(Z-X(I)) PRODUCT V(J)(Z-Y(J))(Z-Y(J)BAR)
C                   I = 1                 J = 1
C
C              WHERE Y(J)BAR IS THE COMPLEX CONJUGATE OF Y(J).
C
C              NOTE - N OR M MAY BE ZERO BUT NOT BOTH.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 172 REGISTERS
C SPEED     -
C AUTHOR    - E.A. ROBINSON
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CONVLV
C       AND FORTRAN SYSTEM ROUTINES - COS
C
C FORTRAN USAGE
C      CALL PLYSYN(SCALES,RADII,DGREES,NROOTS,PLYCOS,NCOFS,SPACE)
C
C INPUTS
C
C    SCALES(I) I=1...NROOTS  IS THE NUMERICAL VALUE OF EACH OF THE SCALE
C               FACTORS U(I), V(J) LISTED IN ANY ORDER
C
C    RADII(I)  I=1...NROOTS  IS THE ABSOLUTE VALUE OR THE NEGATIVE OF
C               THE ABSOLUTE VALUE OF EACH OF THE ROOTS X(I), Y(J)
C               LISTED IN THE SAME ORDER AS SCALES(I)
C
C    DGREES(I) I=1...NROOTS  IS THE ANGLE IN DEGREES OF EACH OF THE
C               ROOTS, LISTED IN THE SAME ORDER.  THE ANGLE IS
C               DETERMINED BY THE EQUATION
C
C                    ROOT = RADII * EXP(SQUAREROOT(-1)*ANGLE*PI/180)
C
C               FOR REAL ROOTS, THE ANGLES WILL BE ZERO OR MULTIPLES
C               OF 180.  PLYSYN CONSIDERS THE ROOT TO BE REAL ONLY IF THE
C               ANGLE IS EXACTLY ZERO OR AN EXACT MULTIPLE OF 180.
C
C    NROOTS     INTEGER EQUAL TO M+N
C               MUST EXCEED ZERO
C
C
C OUTPUTS
C
C    PLYCOS(I) I=1...NCOFS  IS THE POLYNOMIAL COEFFICIENTS, WHERE A(0)
C               IS PLYCOS(1),  A(1) IS PLYCOS(2), ...,
C               A(M+2N) IS PLYCOS(NCOFS)
C
C    NCOFS      IS THE NUMBER OF POLYCOEFFICIENTS, WHICH IS = TO M+2N+1
C
C    SPACE(I)  I=1...NCOFS MUST BE AVAILABLE FOR TEMPORARY STORAGE
C
```

```
C EXAMPLES
C
C 1. CASE OF ONE REAL ROOT OUTSIDE UNIT CIRCLE
C     INPUTS  - SCALES(1) = 1.0    RADII(1) = 1.25    DGREES(1) = 720.
C               NROOTS = 1
C     OUTPUTS - PLYCOS(1...2) = -1.25, 1.0     NCOFS =2
C
C 2. CASE OF ONE REAL ROOT INSIDE UNIT CIRCLE WHICH IS RECIPROCAL
C               TO ROOT ABOVE
C     INPUTS  - SCALES(1) = -1.25    RADII(1) = .8    DGREES(1) = -720.
C               NROOTS =1
C     OUTPUTS - PLYCOS(1...2) = 1.0,-1.25     NCOFS = 2
C                 (NOTE - THIS PLYCOS IS THE REVERSE OF THE PLYCOS ABOVE)
C
C 3. CASE OF ONE COMPLEX ROOT OUTSIDE THE UNIT CIRCLE
C     INPUTS  - SCALES(1) = 1.0   RADII(1) = 1.25    DGREES(1) = 45.
C               NROOTS = 1
C     OUTPUTS - PLYCOS(1...3) = 1.5625, -1.767767, 1.0     NCOFS = 3
C
C 4. CASE OF ONE COMPLEX ROOT INSIDE THE UNIT CIRCLE WHICH IS RECIPROCAL
C               OF ROOT ABOVE
C     INPUTS  - SCALES(1) = 1.5625    RADII(1) = .8    DGREES(1) = -45.
C               NROOTS = 1
C     OUTPUTS - PLYCOS(1...3) = 1.0, -1.7677669, 1.5625    NCOFS = 3
C                 (NOTE - THIS PLYCOS IS THE REVERSE OF THE PLYCOS ABOVE)
C
C 5. CASE OF TWO REAL AND ONE COMPLEX ROOTS OUTSIDE THE UNIT CIRCLE
C     INPUTS  - SCALES(1...3) = 1.,1.,1.  RADII(1...3) = 1.25,1.25,-1.25
C               DGREES(1...3) = 0.,90.,720.  NROOTS = 3
C     OUTPUTS - PLYCOS(1...5) = -2.4414,0.,0.,0.,1.   NCOFS = 5
C
C 6. CASE OF TWO REAL AND ONE COMPLEX ROOTS INSIDE THE UNIT CIRCLE
C               WHICH ARE RECIPROCALS OF ROOTS ABOVE
C     INPUTS  - SCALES(1...3) = -1.25,1.5625,1.25  RADII(1...3) = -.8,
C               -.8,.8  DGREES(1...3) = -540.,-270.,-180.  NROOTS = 3
C     OUTPUTS - PLYCOS(1...5) = 1.,0.,0.,0.,-2.4414
C                 (NOTE - THIS PLYCOS IS THE REVERSE OF THE PLYCOS ABOVE)
C
C 7. CASE OF AUTOCORRELATION POLYNOMIAL
C     INPUTS  - SCALES(1,2) = 1.,1.5625,    RADII(1,2) = 1.25,-.8
C               DGREES(1,2) = 90.,270.  NROOTS=2
C     OUTPUTS - PLYCOS(1...5) = 1.5625,0.,3.4414,0.,1.5625   NCOFS = 5
C
C 8. CASE OF ANOTHER AUTOCORRELATION POLYNOMIAL
C     INPUTS  - SCALES(1...4)=1.,4.,1.,16.  RADII(1...4)=2.,.5,4.,.25
C               DGREES(1...4)=32.,-32.,199.,199.
C     OUTPUTS - PLYCOS(1...9)=64.,242.,-468.,-1420.,4723.,-1420.,-468.,
C                              242.,64.    NCOFS =9
C               (THE VALUES OF PLYCOS GIVEN HERE ARE TRUNCATED TO WHOLE
C               NUMBERS.)
C
      DIMENSION SCALES(2),RADII(2),DGREES(2),PLYCOS(2),SPACE(2),T(3)
C CHECK FOR ILLEGAL NROOTS BEFORE ENTERING LOOP
      IF (NROOTS) 9999,9999,10
 10   DO 200  I=1,NROOTS
      SCALE = SCALES(I)
      RADIUS = RADII(I)
 15   ANGLE = MODF(DGREES(I),360.)
      IF (ABSF(ANGLE)-180.)  20,50,20
C ANGLE NOT = 180 IN MAGNITUDE
 20   IF (ANGLE)  100,60,100
C ANGLE DOES = 180
 50   RADIUS = -RADII(I)
C SET UP T(1) T(2) FOR CASE ANGLE = 180 OR ZERO
 60   T(1) = -RADIUS*SCALE
      T(2) = SCALE
      NT = 2
      GO TO 150
C COMPLEX ROOTS CASE
 100  NT = 3
      T(1) = SCALE*RADIUS*RADIUS
      T(2) = -2.*RADIUS*COSF(ANGLE*3.14159265/180.)*SCALE
      T(3) = SCALE
C CHECK FOR FIRST ROOT
 150  IF (I-1)  180,160,180
```

```
      C IF FIRST ROOT MOVE T(I) INTO PLYCOS(I) AND NT INTO NCOFS
  160 NCOFS = NT
      DO 170  J=1,NT
  170 PLYCOS(J) = T(J)
      GO TO 200
      C CONVOLVE IF NOT FIRST ROOT AND RESET NCOFS AND PLYCOS(I)
  180 CALL CONVLV(NCOFS,PLYCOS,NT,T,SPACE)
      NCOFS = NCOFS+NT-1
      DO 190  J=1,NCOFS
  190 PLYCOS(J) = SPACE(J)
  200 CONTINUE
 9999 RETURN
      END
```

```
*     POKCT1 (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0133
*     LABEL
CPOKCT1
      SUBROUTINE POKCT1 (IX,NHANDS,ICT,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - POKCT1
C       EVALUATION OF INTEGER SEQUENCE IN GROUPS OF FIVE AS POKER HANDS.
C
C             POKCT1 BREAKS UP A FORTRAN II INTEGER SEQUENCE INTO NON-
C             OVERLAPPING GROUPS OF FIVE DIGITS WHICH IT TREATS AS POKER
C             HANDS. THE HANDS ARE EVALUATED AND A TABULATION OF THE
C             NUMBER OF DIFFERENT TYPES OF HANDS IS PRODUCED. THE A
C             PRIORI PROBABILITIES OF DIFFERENT HAND TYPES ARE KNOWN FOR
C             THE CASE OF INDEPENDENT EQUALLY LIKELY DIGITS FROM ZERO TO
C             NINE. HENCE A POKER COUNT IS USEFUL IN DETERMINING THE
C             INDEPENDENCE OF A SEQUENCE. THE A PRIORI PROBABILITIES,
C             DUE TO DAVID DURAND OF M.I.T., ARE GIVEN BELOW AND ARE
C             EXACT. THE DECIMALS TERMINATE AT THE FOURTH PLACE.
C
C                 BUST                         .2952
C                 1 PAIR                       .5040
C                 2 PAIR                       .1080
C                 3 OF A KIND                  .0720
C                 FULL HOUSE                   .0090
C                 STRAIGHT                     .0072
C                 4 OF A KIND                  .0045
C                 5 OF A KIND                  .0001
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 219 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FRQCT1
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL POKCT1(IX,NHANDS,ICT,IANS)
C
C INPUTS
C
C    IX(I)      I=1...5*NHANDS IS THE DIGIT SEQUENCE
C               ZERO LESS THAN OR = IX LESS THAN OR = 9
C
C    NHANDS     IS THE NUMBER OF HANDS TO BE FORMED FROM THE IX SEQUENCE.
C               NHANDS MUST BE GREATER THAN ZERO.
C
C OUTPUTS
C
C    ICT(I)     I=1...8 IS THE COUNT OF TYPES OF HANDS FOUND WHERE
C               ICT(1) = NO. OF HANDS OF NO VALUE
C               ICT(2) = NO. OF HANDS WITH 1 PAIR
C               ICT(3) = NO. OF HANDS WITH 2 PAIRS
C               ICT(4) = NO. OF HANDS WITH 3 OF A KIND
C               ICT(5) = NO. OF STRAIGHTS
C               ICT(6) = NO. OF FULL HOUSES
C               ICT(7) = NO. OF HANDS WITH 4 OF A KIND
C               ICT(8) = NO. OF HANDS WITH 5 OF A KIND
C               WHERE HAND NO. 1 =(IX(1),IX(2),IX(3),IX(4),IX(5))
C                     HAND NO. 2 =(IX(6),IX(7),IX(8),IX(9),IX(10))
C                            ETC.
C               AND SUM OF ICT(I) = NHANDS.
C
C    IANS       =0  NORMAL
C               =1  ILLEGAL NHANDS
C               =3  ERROR RETURN FROM FRQCT1
C
C EXAMPLES
C
C 1. INPUTS  - NHANDS = 0
```

POKCT1
(PAGE 2)

```
C              IX(I)  I=1,280
C                 (BROKEN INTO GROUPS OF FIVE FOR EASY CHECKING)
C              40123  43125  23456  52643  76543  87654  95867
C              97654  02345  98762  14327  02678  86430  63142
C              01230  18741  32024  99413  08628  54531  07499
C              01220  42246  45999  94977  82238  77335  55060
C              10020  23334  06033  88381  74877  06006  15113
C              11222  21212  80808  94449  55454  61116  06006
C              90000  66866  44644  88883  21111  00700  09999
C              99999  00000  11111  22222  66666  33333  36410
C    OUTPUTS - ICT(1...8) = 0,0,0,0,0,0,0,0    IANS=1
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NHANDS=56
C    OUTPUTS - ICT(1...8) = 8,7,7,6,7,8,7,6  IANS=0
C
      DIMENSION  IX(2),ICT(2),IC1(10),IC2(6)
C CLEAR THE OUTPUT VECTOR.  THEN WORK THRU DATA HAND BY HAND.
      IANS=1
      IF(NHANDS) 9999,9999,10
 10   IANS=0
      DO 15 I=1,8
   15 ICT(I)=0
      DO 90  II=1,NHANDS
C FOR EACH HAND FIRST MAKE A FREQUENCY COUNT OF THE DIGITS (VALUES 0-9).
C NOTE RESTRICTION 1 VIOLATION IS CAUGHT BY FRQCT1.
      J=(II-1)*5+1
      CALL FRQCT1(IX(J),5,0,9,IC1,IANS)
      IF(IANS)  9991,21,9991
C AND THEN MAKE A FREQUENCY COUNT OF THE FREQUENCY COUNT (VALUES 0 TO 5,
 21   CALL FRQCT1(IC1,10,0,5,IC2,IANS)
      IF(IANS)  9991,22,9991
C THE HAND VALUE, IVAL (1 TO 8), IS DETERMINABLE FROM IC2(1),IC2(3),
C IC2(2) EXCEPT FOR STRAIGHTS.
 22   IVAL=1
      IF (IC2(1)-6)  60,92,50
   50 IF (IC2(3)-1)  55,96,93
   55 IF (IC2(2)-1)  98,97,94
C CHECK FOR POSSIBLE STRAIGHT WHEN ALL DIGITS ARE DIFFERENT.
   60 I=0
   62 I=I+1
      IF (IC1(I))  70,62,70
   70 IF (IC1(I+1))  71,91,71
   71 IF (IC1(I+2))  72,91,72
   72 IF (IC1(I+3))  73,91,73
   73 IF (IC1(I+4))  95,91,95
C SET THE HAND VALUE.
   98 IVAL=IVAL+1
   97 IVAL=IVAL+1
   96 IVAL=IVAL+1
   95 IVAL=IVAL+1
   94 IVAL=IVAL+1
   93 IVAL=IVAL+1
   92 IVAL=IVAL+1
   91 ICT(IVAL)=ICT(IVAL)+1
   90 CONTINUE
 9999 RETURN
 9991 IANS=3
      GO TO 9999
      END
```

```
*     POLYDV (SUBROUTINE)                9/9/64   LAST CARD IN DECK IS NO. 0101
*     LABEL
CPOLYDV
      SUBROUTINE POLYDV (N,DVS,M,DVD,L,Q)
C
C
C                    ----ABSTRACT----
C
C TITLE - POLYDV
C       PERFORM LONG DIVISION OF TWO POLYNOMIALS
C
C             POLYDV COMPUTES THE FIRST L COEFFICIENTS OF THE QUOTIENT
C             OF TWO POLYNOMIALS.  THE POLYNOMIALS ARE SPECIFIED BY
C             THEIR COEFFICIENTS.  SOME OF THE LAST COEFFICIENTS MAY
C             TURN OUT TO BE ZERO IF THE QUOTIENT IS AN EXACT
C             POLYNOMIAL OF ORDER LESS THAN L.  THE REMAINDER IS NOT
C             COMPUTED.  THE COMPUTATION IS
C
C                                2       3          (L-1)
C             Q(1)+Q(2)*X+Q(3)*X +Q(4)*X +...+Q(L)*X     +REMAINDER
C
C                                           (M+1)                  N-1
C                  =DVD(1)+DVD(2)*X+...DVD(M)*X    /DVS(1)+...DVS(N)*X
C
C             WHERE X IS UNSPECIFIED SINCE ALL OPERATIONS ARE ON THE
C                      COEFFICIENTS,
C                   Q IS THE QUOTIENT VECTOR,
C                   DVD IS THE DIVIDEND VECTOR,
C                   DVS IS THE DIVISOR VECTOR.
C
C LANGUAGE    - FORTRAN-II SUBROUTINE
C EQUIPMENT   - 709, 7090, 7094  (MAIN FRAME ONLY)
C STORAGE     - 130 REGISTERS
C SPEED       - TAKES ABOUT  .0006*L*MINIMUM(L,N)  SECONDS ON THE
C               7094 MOD 1.  (ESTIMATE IS CONSERVATIVE - IN SOME
C               CASES IT MAY BE 50 PERCENT HIGH.)
C AUTHORS     - J. CLAERBOUT AND R.A. WIGGINS
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MOVE, STZ
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C       CALL POLYDV(N,DVS,M,DVD,L,Q)
C
C
C INPUTS
C
C    N          NUMBER OF COEFFICIENTS IN DIVISOR POLYNOMIAL
C               MUST BE GRTHN= 1 .
C
C    DVS(I)     I=1,...,N COEFFICIENTS OF DIVISOR POLYNOMIAL
C               DVS(1) MUST BE NON ZERO
C
C    M          NUMBER OF COEFFICIENTS IN DIVIDEND POLYNOMIAL
C               MUST BE GRTHN= 1 .
C
C    DVD(I)     I=1,....M COEFFICIENTS OF DIVIDEND POLYNOMIAL
C
C    L          NUMBER OF COEFFICIENTS IN QUOTIENT POLYNOMIAL
C               MUST BE GRTHN= 1 .
C
C
C OUTPUTS
C
C    Q(I)       I=1,...,L COEFFICIENTS IN QUOTIENT POLYNOMIAL
C               EQUIVALENCE  (Q,DVD)  ALLOWED.
C
C
C EXAMPLES
C
C 1. INPUTS  - M=1    DVD(1)=1.
C              N=2    DVS(1...2)=1.,-.5
```

POLYDV (PAGE 2)

```
C               L=4
C    OUTPUTS - Q(1...4)=1.,.5,.25,.125
C
C 2. INPUTS  - M=3 , DVD(1...3)= 1.,2.,1.
C               N=2 , DVS(1...2)= 1.,1.
C               L=10
C    OUTPUTS - Q(1...10)=1.,1.,0.,0.,0.,0.,0.,0.,0.,0.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION DVS(2),DVD(2),Q(2)
      MINML=XMINOF(M,L)
      CALL MOVE (MINML,DVD,Q)
      CALL STZ  (L-MINML,Q(MINML+1))
      NM1=N-1
      DO 50  I=1,L
      Q(I)=Q(I)/DVS(1)
      LSUB=XMINOF (NM1,L-I)
      IF (LSUB) 50,50,10
   10 CONTINUE
      K=I
      DO 20  J=1,LSUB
      K=K+1
   20 Q(K)=Q(K)-Q(I)*DVS(J+1)
   50 CONTINUE
      RETURN
      END
```

```
*      POLYEV (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0061
*      LABEL
CPOLYEV
       SUBROUTINE POLYEV(N,C,X,A)
C
C                     ----ABSTRACT----
C
C TITLE - POLYEV
C        EVALUATE A POLYNOMIAL WITH REAL COEFFICIENTS FOR REAL ARGUMENT
C
C              POLYEV EVALUATES A POLYNOMIAL.  THAT IS, GIVEN THE
C              POLYNOMIAL COEFFICIENTS C(1...N), POLYEV FINDS THE VALUE,
C              A, OF THE POLYNOMIAL FOR A GIVEN X, (C AND X REAL)
C
C                                        2       3            N-1
C                  A = C(1)+C(2)*X+C(3)*X +C(4)*X +....+X(N)*X
C
C              SPEED IS GAINED BY GROUPING THE POLYNOMIAL AS (FOR N=5)
C
C                  A = C(1)+X*(C(2)+X*(C(3)+X*(C(4)+X*C(5)))))
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 54 REGISTERS
C SPEED      - ABOUT 35 TIMES NUMBER OF COEFFICIENTS  MACHINE CYCLES .
C AUTHOR     - J.F. CLAERBOUT
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C       CALL POLYEV(N,C,X,A)
C
C INPUTS
C
C    C(I)        I=1,...,N ARE THE COEFFICIENTS OF THE POLYNOMIAL.
C                IS FLOATING POINT.
C
C    N           IS FORTRAN INTEGER
C                MUST BE GRTHN=1.
C
C    X           IS THE VALUE FOR WHICH THE POLYNOMIAL IS TO BE EVALUATED.
C                MUST BE FLOATING POINT.
C
C OUTPUTS
C
C    A           IS THE VALUE OF THE POLYNOMIAL.
C
C EXAMPLES
C
C 1. INPUTS  - N = 3   C(1...3) = 1.,2.,3.    X=2.
C    OUTPUTS - A = 17.
C
      DIMENSION C(100)
      A=0.
      DO 10 I=1,N
      J=N-I
 10   A=X*A+C(J+1)
      RETURN
      END
```

```
*     POLYSN (SUBROUTINE)               9/8/64   LAST CARD IN DECK IS NO. 0166
*     LABEL
CPOLYSN
      SUBROUTINE POLYSN (SCALE,NOZ,ZRE,ZIM,ZIFCOM,ZIFCNJ,LPOLY,POLY,
     1                   SPACE)
C
C
C                    ----ABSTRACT----
C
C TITLE - POLYSN
C        POLYNOMIAL SYSTHESIS FROM REAL AND COMPLEX ROOTS
C
C              SUBROUTINE POLYSN SYNTHESIZES A POLYNOMIAL WITH REAL
C              COEFFICIENTS FROM REAL AND COMPLEX ROOTS.  NECESSARILY,
C              THE COMPLEX ROOTS OCCUR IN COMPLEX CONJUGATE PAIRS.
C              POLYSN ALLOWS OPTIONS FOR THE USER TO SPECIFY EITHER ONE,
C              OR BOTH, OF THE ROOTS IN THESE PAIRS.  ALSO, THE COMPLEX
C              ROOTS MAY BE SPECIFIED BY THEIR REAL AND IMAGINARY PARTS,
C              OR BY THEIR MAGNITUDE AND ARGUMENT (IN DEGREES).
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094  (MAIN FRAME ONLY)
C STORAGE    - 256 RE ISTERS
C SPEED      - TAKES ABOUT  .0010 + .00011*N*N SECONDS ON THE
C              7094 MOD I, WHERE N IS THE NUMBER OF ROOTS.
C AUTHOR     - R.A. WIGGINS    4/64
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CONVLV, MOVE
C       AND FORTRAN SYSTEM ROUTINES - COS, SQRT
C
C FORTRAN USAGE
C     CALL POLYSN(SCALE,NOZ,ZRE,ZIM,ZIFCOM,ZIFCNJ,LPOLY,POLY,SPACE)
C
C
C INPUTS
C
C    SCALE     IS A SCALE VALUE THAT POLYNOMIAL IS MULTIPLIED BY.
C              IF = 0., THE POLYNOMIAL IS SCALED SO THAT POLY(1)=1.
C
C    NOZ       NUMBER OF ZEROES GIVEN.
C
C    ZRE(I)    I=1...NOZ GIVES THE REAL PART IF ZIFCOM = 0., GIVES THE
C                MAGNITUDE IF ZIFCOM NOT = 0.
C
C    ZIM(I)    I=1...NOZ GIVES THE IMAGINARY PART OF THE ZERO IF
C                ZIFCOM = 0., GIVES THE ARGUMENT (IN DEGREES) IF
C                ZIFCOM NOT = 0.
C
C    ZIFCOM    = 0.  IF ZEROES SPECIFIED BY REAL AND IMAGINARY PARTS.
C              NOT= 0.  IF ZEROES SPECIFIED BY MAGNITUDE AND ARGUMENT.
C
C    ZIFCNJ    = 0. IF POLYSN MUST FIND THE CONJUGATE OF ALL NON-REAL
C                     ROOTS.  I.E. ONLY ONE OF EACH PAIR OF COMPLEX
C                     CONJUGATES IS SPECIFIED IN ZRE AND ZIM.
C              NOT= 0.  IF CONJUGATE OF EACH NON-REAL ROOT IS ACTUALLY
C                     CONTAINED IN ZRE AND ZIM.  POLYSN ASSUMES THAT
C                     THESE CONJUGATE PAIRS ARE STORED SEQUENTIALLY.
C
C    SPACE(I)  I=1...2*NOZ IS TEMPORARY COMPUTATION SPACE.
C
C
C OUTPUTS
C
C    LPOLY     LENGTH OF POLYNOMIAL FORMED.
C
C    POLY(I)   I=1...LPOLY CONTAINS THE COEFFICIENTS OF THE POLYNOMIAL
C                     IN ORDER OF INCREASING POWERS OF Z.
C
C
C EXAMPLES
C
C 1. INPUTS  - SCALE = 1.  NOZ = 1  ZRE(1) = -.5  ZIM(1) = 0.  ZIFCOM=0.
```

```
C              ZIFCNJ = 0.
C    OUTPUTS - LPOLY = 2  POLY(1...2) = .333,.667
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT SCALE = 0.
C    OUTPUTS - LPOLY = 2  POLY(1...2) = 1.,2.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ZIM(1) = 180.  ZIFCOM = 1.
C    OUTPUTS - LPOLY = 2  POLY(1...2) = -.333,.667
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  ZRE(1) = .5  ZIM(1) = .5
C    OUTPUTS - LPOLY = 3  POLY(1...3) = .172,-.343,.343
C
C 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ZIM(1) = 45.  ZIFCOM = 1.
C    OUTPUTS - LPOLY = 3  POLY(1...3) = .111,-.314,.444
C
C 6. INPUTS  - SCALE = 1.  NOZ = 2  ZIFCOM = 0.  ZIFCNJ = 0.
C              ZRE(1...2) = .5,2.  ZIM(1...2) = 0.,1.
C    OUTPUTS - LPOLY = 4  POLY(1...4) = -.159,.446,-.286,.0637
C
C 7. INPUTS  - SCALE = 1.  NOZ = 3  ZIFCOM = 0.  ZIFCNJ = 1.
C              ZRE(1...3) = .5,2.,2.  ZIM(1...3) = 0.,1.,-1.
C    OUTPUTS - LPOLY = 4  POLY(1...4) = -.159,.446,-.286,.0637
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION ZRE(2),ZIM(2),POLY(2),SPACE(2)
      DIMENSION T(3)
      IF (NOZ) 999,999,10
 10   CONTINUE
      CONV=3.14159265/180.
      LPLY =1
      POLY(1)=1.
      IFST=0
      DO 120 I=1,NOZ
      ZR=ZRE(I)
      ZI=ZIM(I)
      IF (ZIFCNJ) 12,18,12
 12   CONTINUE
      IF (IFST) 18,18,14
 14   CONTINUE
      IFST=0
      GO TO 120
 18   CONTINUE
      IF (ZIFCOM) 50,20,50
C ZEROES ARE EXPRESSED BY THEIR REAL AND IMAGINARY PARTS
 20   CONTINUE
      IF (ZI)  40,30,40
C SINGLE ZERO
 30   CONTINUE
      T(2)=1./(1.+ABSF(ZR))
      T(1)=-ZR*T(2)
      NT=2
      GO TO 100
C DOUBLE ZERO
 40   CONTINUE
      T(1)=ZR*ZR+ZI*ZI
      T(3)=1./((1.+SQRTF(T(1)))*(1.+SQRTF(T(1))))
      T(1)=T(1)*T(3)
      T(2)=-2.*ZR*T(3)
      IFST=1
      NT=3
      GO TO 100
C ZEROES ARE EXPRESSED BY MAGNITUDE AND PHASE
 50   CONTINUE
      ZI =MODF(ZI,360.)
      IF (ABSF(ZI)-180.)  70,60,70
 60   ZR=-ZR
      GO TO 30
 70   IF (ZI) 80,30,80
C DOUBLE ZERO
 80   CONTINUE
      T(3)=1./((1.+ABSF(ZR))*(1.+ABSF(ZR)))
      T(1)=ZR*ZR*T(3)
      T(2)=-2.*ZR*COSF(ZI*CONV)*T(3)
```

****
*   POLYSN   *
****
(PAGE 3)

```
      IFST=1
      NT=3
C CONVLV AND RESET LPOLY AND PLYCOS(I)
  100 CONTINUE
      CALL CONVLV (LPLY ,POLY,NT,T,SPACE)
      LPLY =LPLY +NT-1
      CALL MOVE (LPLY ,SPACE,POLY)
  120 CONTINUE
      SC=SCALE
      IF (SC) 140,130,140
  130 SC=1./POLY(1)
  140 CONTINUE
      DO 150 I=1,LPLY
  150 POLY(I)=SC*POLY(I)
      LPOLY=LPLY
  999 RETURN
      END
```

```
*      POWER  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0129
*      FAP
*POWER
       COUNT   150
       LBL     POWER
       ENTRY   POWER  (X,LX,N,X2NTH)
       ENTRY   SMPRDV (X,LX,N,XBASE,SXMB2N)
*
*                    ----ABSTRACT----
*
* TITLE - POWER , WITH SECONDARY ENTRY SMPRDV
*        RAISE VECTOR TO POWER OR SUM POWER OF DEVIATIONS FROM BASE
*
*              POWER RAISES ELEMENTS OF A VECTOR TO A POSITIVE OR
*              NEGATIVE INTEGER POWER.  OUTPUT MAY REPLACE INPUT.
*
*              SMPRDV SUMS THE N-TH POWER OF THE DEVIATIONS OF A
*              VECTOR FROM A CONSTANT, WHERE  N  IS POSITIVE OR
*              NEGATIVE.
*
* LANGUAGE   - FAP SUBROUTINES, FORTRAN-II COMPATIBLE
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 50   REGISTERS
* SPEED      -  7090   709
*              (46  OR  50) + (70 TO 270)*LX  MACHINE CYCLES,
*                   DEPENDING ON ARGUMENTS
* AUTHOR     - S.M.SIMPSON, SEPTEMBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - EXP(2
*
* FORTRAN USAGE
*     CALL POWER (X,LX,N,X2NTH)
*     CALL SMPRDV(X,LX,N,XBASE,SXMB2N)
*
* INPUTS
*
*    X(I)      I=1...LX   IS A FLTG PT  VECTOR
*
*    LX        MUST EXCEED ZERO.
*
*    N         IS THE DESIRED POWER (POS OR NEG OR ZERO)
*
*    XBASE     IS THE BASE (FOR SMPRDV)
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF LX LSTHN 1, EXCEPT
*              THAT SXMB2N WILL BE SET = 0.
*
*    X2NTH(I)  I=1...LX   IS  X2NTH(I) = (X(I))**N
*
*              EQUIVALENCE(X2NTH,X)  IS PERMITTED
*
*    SXMB2N    IS = SUM(FROM I=1 TO LX) OF (X(I)-XBASE)**N
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4) = 1., 2., 3., -4.            XN4 = -999.
*    USAGE   -        CALL POWER (X, 4,  3, XN1)
*                     CALL POWER (X, 4, -2, XN2)
*                     CALL POWER (X, 1,  3, XN3)
*                     CALL SMPRDV(X, 4,  3, 1., SXN1)
*                     CALL POWER (X, 4,  3, X)
*                     CALL POWER (X, 0,  3, XN4)
*    OUTPUTS - XN1(1...4) =  1., 8., 27., -64.
*              XN2(1...4) =  1., .25, .1111111, .0625,
*              XN3 = 1.
*              SXN1 = -116.
*              X(1...4)  =  1., 8., 27., -64.
*              XN4 = -999. (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
```

****
*   POWER   *
****
(PAGE 2)

```
* TRANSFER VECTOR CONTAINS EXP(2
       HTR    0             XR1
       HTR    0             XR4
       BCI    1,POWER
* PRINCIPAL ENTRY. POWER(X,LX,N,X2NTH)
 POWER CLA    4,4
       ADD    K1            A(X2NTH)+1
       STA    STO
       STZ    BASE          CLEAR BASE
       STZ    ZIFPOW        AND ENTRY INDICATOR
       CLA    TRAP
       TRA    SETUP
* SECONDARY ENTRY. SMPRDV(X,LX,N,XBASE,SXMB2N)
SMPRDV CLA*   4,4           XBASE
       STO    BASE
       CLA    5,4           A(SMXNTH)
       STA    FAD
       STZ*   5,4           CLEAR SUM
       CLA    TRAS
       SXD    ZIFPOW,4      (ZIFPOW NON-ZERO)
 SETUP SXD    POWER-2,4
       SXD    POWER-3,1
       STA    TRA
 K1    CLA    1,4
       ADD    K1            A(X)+1
       STA    CLA
       CLA*   3,4           N
       STO    POWR
       CLA*   2,4           LX
       TMI    LEAVE
       PDX    0,1
       TXL    LEAVE,1,0
* LOOP
 CLA   CLA    **,1          **=A(X)+1
       FSB    BASE
       LDQ    POWR
       TSX    $EXP(2,4
 TRA   TRA    **            **=STO OR FAD
 STO   STO    **,1          **=A(X2NTH)+1
       TRA    TIX
 FAD   FAD    **            **=A(SMXNTH)
       STO*   FAD
 TIX   TIX    CLA,1,1
* EXIT
 LEAVE LXD    POWER-2,4
       LXD    POWER-3,1
       ZET    ZIFPOW
       TRA    6,4
       TRA    5,4
 TRAP  PZE    STO
 TRAS  PZE    FAD
 BASE  PZE    **,**,**      =  0.0 OR XBASE
 POWR  PZE    0,0,**        = N
ZIFPOW PZE    0,0,**        **=0 IF POWER,=NON ZERO IF SMPRDV
       END
```

```
*     PRBFIT (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0186
*     LABEL
CPRBFIT
      SUBROUTINE PRBFIT(NOR,XMOM,NOUT,X,F,PHI,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - PRBFIT
C       GENERATE PROBABILITY DISTRIBUTION WITH SPECIFIED MOMENTS
C
C             PRBFIT GENERATES A ZERO-MEAN DISTRIBUTION FUNCTION, F(X),
C             WHOSE HIGHER MOMENTS (2ND,3RD,...,NTH  WHERE N IS LESS
C             THAN OR EQUAL 6) ASSUME GIVEN VALUES.  F(X) HAS THE FORM
C             OF A NORMAL DISTRIBUTION TIMES A POLYNOMIAL IN X, AND
C             CONSEQUENTLY IS USEFUL FOR APPROXIMATING EMPIRICAL
C             DISTRIBUTIONS WHICH ARE ROUGHLY NORMAL IN APPEARANCE,
C             BUT FOR WHICH THE NORMAL APPROXIMATION IS INADEQUATE.
C             IT SHOULD BE NOTED THAT THE PROCEDURE CAN YIELD NEGATIVE
C             VALUES FOR THE DISTRIBUTION IN CASES WHERE THE DEVIATION
C             FROM NORMALITY IS SEVERE.
C             AN ANALYSIS OF THE PROCEDURE USED MAY BE FOUND IN
C             CRAMER, H., 1951, MATHEMATICAL METHODS OF STATISTICS,
C             PRINCETON UNIVERSITY PRESS, PRINCETON, PAGE 222.
C
C             THE FORM OF THE CALCULATION IS
C
C                               C(3)       D  D  D(PHI(U))
C                F(X) = PHI(U) + ----- * (--*--*---------)
C                               1*2*3      DU DU DU
C
C                        C(4)       D  D  D  D(PHI(U))
C                     + ------- * (--*--*--*---------) +...+
C                       1*2*3*4    DU DU DU DU
C
C                      C(NOR)           D       D(PHI(U))
C                     +----------- * (--*...*---------)
C                      1*2*...*NOR    DU      DU
C
C             EVALUATED FOR A GIVEN SET OF X VALUES
C                 X=X(1),X(2),...,X(NOUT)
C             WHERE
C                     D
C                     --   DENOTES DIFFERENTIATION WITH RESPECT TO U
C                     DU
C
C                      U = X/SIG
C
C                  PHI(U) = EXP(-.5*U*U)/(SQUARE ROOT(2*PI))
C                           (I.E. NORMAL CURVE)
C
C                     PI = 3.14159265
C
C                           K    XMOM(L)
C                    C(K) = SUM ( ------- * A(K,L) )
C                           L=0     SIG
C
C                  A(K,L) = COEFFICIENT OF LTH POWER OF X IN THE KTH
C                           HERMITE POLYNOMIAL (X)
C
C                 XMOM(L) = LTH PROBABILITY MOMENT
C                           (INPUT PARAMETER VECTOR)
C
C                     SIG = SQUARE ROOT(XMOM(2))
C                           I.E. STANDARD DEVIATION
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709, 7090 (MAIN FRAME ONLY)
C STORAGE    - 373 REGISTERS
C SPEED      -
C AUTHOR     - R.J. GREENFIELD,  JAN 1963
C
C                    ----USAGE----
C
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
```

PRBFIT (PAGE 2)

```
C       AND FORTRAN SYSTEM ROUTINES - SQRT, EXP(2, EXP
C
C FORTRAN USAGE
C    CALL PRBFIT(NOR,XMOM,NOUT,X,F,PHI,IANS)
C
C INPUTS
C
C    NOR       IS THE ORDER OF THE HIGHEST ORDER MOMENT GIVEN
C              MUST BE GRTHN= 2 AND LSTHN = 6
C
C    XMOM(I)   I=1...NOR  CONTAINS THE MOMENTS WHICH WILL BE USED TO
C                DEVELOP THE EXPANSION.  THE FIRST MOMENT, XMOM(1),
C                IS NOT ACTUALLY USED, BUT IS ASSUMED TO BE =0.
C                (I.E. ZERO MEAN ASSUMPTION).
C
C    NOUT      IS THE NUMBER OF X VALUES AT WHICH THE EXPANSION WILL BE
C                EVALUATED
C
C    X(I)      I=1...NOUT IS THE LIST OF VALUES AT WHICH THE EXPANSION
C                WILL BE EVALUATED
C
C    PHI(I)    USED FOR STORAGE
C              MUST BE DIMENSIONED AT LEAST AS LARGE AS NOUT
C
C OUTPUTS
C
C    F(I)      I=1...NOUT  ARE THE VALUES OF THE EXPANSION FOR THE
C                NOUT VALUES OF X, I.E. F(I) = F(X(I)) AS DEFINED
C                IN ABSTRACT
C
C    IANS      = 0  NORMAL
C              = 1  ILLEGAL NOR
C
C
C EXAMPLES
C
C 1. (NORMAL APPROXIMATION)
C    INPUTS  - NOR = 2    XMUD(1...4) = 0.,4.,8.,10.     NOUT = 4
C              X(1...4)= 0.,5.,.8,-.8
C    OUTPUTS - F(1...4)= .39894,.017528,.36828,.36828     IANS= 0
C
C 2. INPUTS    SAME AS IN EXAMPLE 1. EXCEPT NOR= 3
C    OUTPUTS - F(1...4)= .39894,.041265,.29854,.43800     IANS= 0
C
C 3. INPUTS  - SAME AS IN EXAMPLE 1. EXCEPT NOR= 4
C    OUTPUTS - F(1...4)= .28051,.0333501,.22328,.36272    IANS= 0
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NOR= 0
C    OUTPUTS - ERROR  IANS= 1
C
C 5. INPUTS  - SAME AS IN EXAMPLE 1. EXCEPT NOR=10
C    OUTPUTS - ERROR   IANS = 1
C
      DIMENSION A(7,7),C(7),PHI(100),XMOM(7),X(100),XMUD(7)
      DIMENSION XMU(7),F(2)
      NORDER = NOR +1
C  TEST INPUT DATA
      IF (NORDER-2) 31,31,32
 31   IANS=1
      RETURN
 32   IF(NORDER-7)  33,33,31
 33   IANS=0
      XMU(1)= 1.
      XMU(2)= 0.
      DO 50 K=2,NOR
 50   XMU(K+1)=XMOM(K)
C  SET UP A TABLE
      DO 1  J=1,7
 1    A(J,J)=1.
      A(3,1)=-1.
      A(4,2)=-3.
      A(5,1)=3.
      A(5,3)=-6.
      A(6,2)=15.
      A(6,4)=-10.
```

****
* PRBFIT *
****
(PAGE 3)

```
         A(7,1)=-15.
         A(7,3)=45.
         A(7,5)=-15.
C  ALL SUBSCRIPTS ADVANCED BY 1
C  X(I) INPUT NORMALIZED BY CALLING PROG (ZERO MEAN)
C  XMU ARE NOT NORMALIZED BUT ARE FOR ZERO MEAN
C  SEC TO COMP C
         SIG= SQRTF(XMU(3))
         DO 51 I=1,NOUT
  51     X(I)= X(I)/SIG
         FACT=1.
         DO 5 K=1,NORDER
         C(K)=0.
         IF(K-1) 41,41,40
  40     FACT=FACT*FLOATF(K-1)
  41     DO 4  L=1,K
   4     C(K)=C(K)+(XMU(L)/(SIG**(L-1)))*A(K,L)
   5     C(K)=C(K)/FACT
C  SET UP TABLE OF PHI
         DC 6  I=1,NOUT
   6     PHI(I)=EXPF(-X(I)*X(I)*.5)*.3989423
C  COMPUTE F(I) FOR NCRMAL DISTRIBUTICN
         DO 7  I=1,NOUT
   7     F(I)=C(1)*PHI(I)
         IF(NORDER-4)  99,8,8
C  COMPUTES OTHER ORDER F
   8     DO 19  K=4,NORDER
         DO 12  I=1,NOUT
         HER=A(K,1)
         DO 10  L=2,K
  10     HER=HER+A(K,L)*X(I)**(L-1)
  12     F(I)=F(I)+PHI(I)*C(K)*HER
  19     CONTINUE
  99     DO 98 I=1,NOUT
  98     X(I)= X(I)*SIG
         RETURN
         END
```

```
*     PROB2 (SUBROUTINE)                 10/6/64   LAST CARD IN DECK IS NO. 0174
*     LABEL
CPROB2
      SUBROUTINE PROB2 (IX,LX,N,IP,P,IXHI,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - PROB2
C       SECOND PROBABILITY DENSITY OF INTEGER SERIES AT GIVEN LAG.
C
C             PROB2 COMPUTES THE SECOND PROBABILITY DENSITY FOR AN
C             INTEGER SERIES BY A FREQUENCY COUNT METHOD. THE SECOND
C             PROBABILITY DENSITY, P(M,L), OF A SERIES IX(K) IS THE
C             PROBABILITY THAT X(K) = M AND X(K+N)=L, WHERE N IS THE
C             LAG. PROB2 COMPUTES THIS QUANTITY FOR A GIVEN N. THE
C             INTEGER SERIES MUST BE SCALED SUCH THAT THE LOWEST VALUE
C             OF IX(K) =1 AND THE HIGHEST VALUE IS IXHI. IXHI MUST BE
C             LESS THAN OR EQUAL TO THE DIMENSION OF THE P(I,J) MATRIX.
C             THE PROGRAM BELOW DIMENSIONS P(I,J) TO P(25,25).
C
C             PROB2 COUNTS INTO AN INTEGER MATRIX, IP(I,J), THE NUMBER
C             OF TIMES IX(K)=M AND IX(K+N)=L OVER ALL INDEX PAIRS
C             K, K+N  SUCH THAT BOTH K AND K+N LIE IN THE INCLUSIVE
C             RANGE 1 TO LX WHERE LX IS THE SERIES LENGTH.  N MAY
C             BE NEGATIVE.
C
C             THE INTEGER FREQUENCY COUNT MATRIX IS FLOATED INTO P(I,J)
C             AND NORMALIZED SUCH THAT SUM OVER I AND J OF P(I,J) IS 1.
C             THIS IS DONE BY DIVIDING EACH ELEMENT BY R, WHERE
C             R=LX-XABSF(N). P(I,J) AND IP(I,J) MAY BE EQUIVALENT IF THE
C             FREQUENCY COUNT IS NOT NEEDED. (THIS CAN BE RECONSTRUCTED
C             SINCE LX AND N ARE KNOWN.)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709,7090 (MAIN FRAME ONLY)
C STORAGE   - 229 DECIMAL REGISTERS
C SPEED     -
C AUTHOR    - J.N. GALBRAITH
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL PROB2 (IX,LX,N,IP,P,IXHI,IANS)
C
C INPUTS
C
C    IX(I)      I=1,...,LX  INTEGER SERIES. IX(I) GRTHN 0, LSTHN OR = IXHI
C
C    LX         INTEGER. LENGTH OF IX SERIES. GRTHN ZERO
C
C    N          INTEGER. LAG OR SEPARATION FOR COUNT. CAN BE +,-, OR 0.
C               XABS(N) LSTHN OR = LX
C
C    IP(I,J)    I=1,...,IXHI,J=1,...,IXHI  SPACE FOR COMPUTATION OF
C               FREQUENCY RATIOS. MAY BE EQUIVALENT TO P(I,J). WILL
C               CONTAIN FREQUENCY RATIOS WHEN RETURN IS MADE IF NO
C               EQUIVALENCE HAS BEEN MADE.
C
C    IXHI       INTEGER. LARGEST VALUE IX TAKES ON. PROGRAM ASSUMES
C               IXHI LSTHN OR = 25. MUST BE LSTHN OR EQUAL DIMENSION OF
C               P(I,J) MATRIX.
C
C OUTPUTS
C
C    P(I,J)     I=1,...,IXHI,J=1,...,IXHI. PROBABILITY DENSITY FOR LAG OF N
C               NORMALIZED SUCH THAT  SUM OVER I AND J OF P(I,J) IS 1.
C
C    IANS       INTEGER.  ERROR INDICATOR
C               =0 NORMAL
C               =-1 ILLEGAL IX VALUE. SOME IX LSTHN 1 OR GRTHN IXHI.
C               =-2 ILLEGAL LX. LX LSTHN 1
C               =-3 ILLEGAL N.  XABSF(N) GRTHN LX.
```

```
C              =-6 ILLEGAL IXHI. IXHI GRTHN 26 OR LSTHN 1.
C              =3  JOB DONE BUT N=0 AND ONLY CONTRIBUTIONS TO P(I,J) ARE
C                  ON THE DIAGONAL.
C
C EXAMPLES
C
C 1. INPUTS  - IX(I)=0, LX=5, N=1, IXHI=5
C    OUTPUTS - IP(I,J)=0 , P(I,J)=0 , IANS=-1
C
C 2. INPUTS  - SAME AS EXAMPLE 1  EXCEPT  IX(I)=1,2,3,4,6
C    OUTPUTS - SAME AS EXAMPLE 1
C
C 3, INPUTS  - SAME AS EXAMPLE 2 EXCEPT LX=0
C    OUTPUTS - IANS=-2
C
C 4. INPUTS  - SAME AS EXAMPLE 2 EXCEPT IXHI=0
C    OUTPUTS - IANS=-6
C
C 5. INPUTS  - SAME AS EXAMPLE 4 EXCEPT IXHI=26
C    OUTPUTS - IANS=-6
C
C 6. INPUTS  - SAME AS EXAMPLE 2 EXCEPT IX(5)=5, N=-6
C    OUTPUTS - IANS=-3
C
C 7. INPUTS  - IX(I)=1,1,2,2,3,3,4,4,5,5,1,2,2,3,4,5,5,1,1,1,1,1,1,1
C              IXHI=5, LX=21, N=1
C    OUTPUTS - IANS=0
C               4  2  0  0  0                 .2  .1  .0   .0   .0
C               0  2  2  0  0                 .0  .1  .1   .0   .0
C     IP(I,J)=  0  0  1  2  0        P(I,J)=  .0  .0  .05  .1   .0
C               0  0  0  1  2                 .0  .0  .0   .05  .1
C               2  0  0  0  2                 .1  .0  .0   .0   .1
C
C 8. INPUTS  - SAME AS EXAMPLE 7 EXCEPT N=-1
C    OUTPUTS - IANS=0
C
C               4  0  0  0  2                 .2  .0  .0   .0   .1
C               2  2  0  0  0                 .1  .1  .0   .0   .0
C     IP(I,J)=  0  2  1  0  0        P(I,J)=  .0  .1  .05  .0   .0
C               0  0  2  1  0                 .0  .0  .1   .05  .0
C               0  0  0  2  2                 .0  .0  .0   .1   .1
C
C 9. INPUTS  - SAME AS EXAMPLE 7 EXCEPT LX=24, N=3
C    OUTPUTS - IANS=0
C               4  1  2  0  0                 .19  .05  .1   .0   .0
C               0  0  1  2  1                 .0   .0   .05  .1   .05
C     IP(I,J)=  0  0  0  1  2        P(I,J)=  .0   .0   .0   .05  .1
C               2  0  0  0  1                 .1   .0   .0   .0   .05
C               2  2  0  0  0                 .1   .1   .0   .0   .0
C
C10. INPUTS  - SAME AS EXAMPLE 7 EXCEPT LX=20, N=0
C    OUTPUTS - IANS=3
C
C              6  0  0  0  0                  .3  .0  .0   .0   .0
C              0  4  0  0  0                  .0  .2  .0   .0   .0
C     IP(I,J)= 0  0  3  0  0         P(I,J)=  .0  .0  .15  .0   .0
C              0  0  0  3  0                  .0  .0  .0   .15  .0
C              0  0  0  0  4                  .0  .0  .0   .0   .2
C
      DIMENSION IX(1000),IP(25,25),P(25,25)
C     CHECK LX
      IANS=-2
      IF(LX)  9999,9999,2
 2    IANS=-6
C     CHECK IXHI
      IF(IXHI)  9999,9999,3
 3    IF(IXHI-25)  4,4,9999
C     CHECK IX SERIES
 4    IANS=-1
      DO 1  I=1,LX
      IF(IX(I)) 9999,9999,11
 11   IF(IX(I)-IXHI) 1,1,9999
 1    CONTINUE
      IANS=-3
C     CHECK N
```

****
*   PROB2   *
****
(PAGE 3)

```
      IF(XABSF(N)-LX) 41,9999,9999
   41 IANS=0
C     CLEAR IP(I,J)
      DO 5  I=1,25
      DO 5 J=1,25
    5 IP(I,J)=0
      IF(N) 6,7,8
    6 LFRST=-N+1
      LLAST=LX
      GO TO 9
    7 IANS=3
    8 LFRST=1
      LLAST=LX-N
    9 DO 10 I=LFRST,LLAST
      J=IX(I)
      KK=I+N
      K=IX(KK)
   10 IP(J,K)=IP(J,K)+1
      L=LLAST-LFRST+1
      TOTAL=L
      DO 15  I=1,IXHI
      DO 15 J=1,IXHI
   15 P(I,J)=FLOATF(IP(I,J))/TOTAL
 9999 RETURN
      END
```

```
*       PROCOR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 1498
*       FAP
*PROCOR
        COUNT   1500
        LBL     PROCOR
        ENTRY   PROCOR (X,LX,MAXX,PROG1,PROG2,ERR)
        ENTRY   FASCOR (Y,KMIN,KMAX,CORZER,ERROR)
        ENTRY   FASEPC (Y,KMIN,KMAX,CORZER,ERROR)
        ENTRY   FASCR1 (Y,KMIN,KMAX,CORZER,ERROR)
        ENTRY   FASEP1 (Y,KMIN,KMAX,CORZER,ERROR)
*
*                    ----ABSTRACT----
*
* TITLE - PROCOR WITH SECONDARY ENTRY POINTS FASCOR,FASEPC,FASCR1,FASEP1
*       FAST CORRELATIONS FOR LONG SERIES OF FIXED POINT INTEGERS
*
*              PROCOR WRITES A MACHINE LANGUAGE PROGRAM DESIGNED TO
*              COMPUTE AT HIGH SPEED A SINGLE FIXED POINT CROSS PRODUCT
*              OF A GIVEN SERIES, X(1...LX), WITH AN ARBITRARY SERIES,
*              Y(1...LX), WHERE THE Y SERIES CAN BE LAGGED ARBITRARILY.
*              SPEED OF ONE CROSS PRODUCT APPROACHES 2LX MACHINE CYCLES
*              AS LX GETS LARGE WITH RESPECT TO MAXIMUM MAGNITUDE OF X
*              (CONSIDERED AS 35 BIT-PLUS SIGN INTEGERS).  USER PROVIDES
*              SPACE FOR OBJECT PROGRAM WHICH IS SOMEWHAT LONGER THAN
*              X SERIES.  ONCE THE PROGRAM IS GENERATED X(I) IS NO
*              LONGER NEEDED AND THE PROGRAM IS REUSABLE.  HIGH SPEED
*              IS ATTAINED BY GROUPING MULTIPLIERS SO AS TO SUBSTITUTE
*              SUMMATION FOR MULTIPLICATION AND BY CARRYING OUT THE
*              SUMMATION BY A STRAIGHT LINE PROGRAM.  FOR EXAMPLE IF
*                 X(1...8)  = 1, 2,-1, 0,-2, 0, 1, 2
*                 Y(1...8)  = 2,-1, 2, 0, 1, 2,-2, 1
*              THE CROSS PRODUCT
*                 1*2 + 2*(-1) - 1*2 + 0*0 - 2*1 + 0*2 + 1*(-2) + 2*1
*              WOULD BE COMPUTED BY THE OBJECT PROGRAM IN THE FORM
*                      (2 - 2 - 2)*1  +  (- 1 - 1 + 1)*2
*
*              FASCOR SUCCESSIVELY OPERATES THE PROGRAM GENERATED BY
*              PROCOR TO PRODUCE A SPECIFIED TRANSIENT CORRELATION
*              FUNCTION, XP(K), BETWEEN X(I) AND Y(I)
*
*                                 LX
*                    XP(K)  =  SUM ( X(I)*Y(I+K) )
*                                 I=1
*
*                      FOR K= KMIN,KMIN+1,...,0,1,...,KMAX
*                      WHERE
*                             KMIN = NEGATIVE OR ZERO INPUT PARAMETER
*                             KMAX = POSITIVE OR ZERO INPUT PARAMETER
*                             Y(L) IS, FOR PURPOSES OF THE ABOVE EQUATION,
*                                  CONSIDERED = 0 WHENEVER L FALLS OUTSIDE
*                                  OF THE INCLUSIVE RANGE 1 TO LX (THIS
*                                  IS THE TRANSIENT ASSUMPTION).
*                             X(I) IS RESTRICTED TO HAVE MAGNITUDES NOT
*                                  EXCEEDING VALUE 1000
*
*              FASEPC IS IDENTICAL TO FASCOR EXCEPT THAT IT DOES
*              NOT MAKE THE TRANSIENT ASSUMPTION ABOUT Y(I).
*
*              FASCR1 IS IDENTICAL TO FASCOR EXCEPT THAT XP(K) IS ADDED
*              INTO THE OUTPUT AREA RATHER THAN BEING STORED INTO IT.
*              (THIS IS A SPACE-SAVING FEATURE OF VALUE, FOR INSTANCE,
*              WHEN DOING AVERAGED CORRELATIONS ON SEGMENTED SERIES.)
*
*              FASEP1 IS IDENTICAL TO FASEPC EXCEPT THAT XP(K) IS ADDED
*              INTO THE OUTPUT AREA RATHER THAN BEING STORED INTO IT.
*
*
*              PROCOR IS SEPARATED FROM THE OTHER ENTRY POINTS TO PERMIT
*              EFFICIENT COMPUTATION OF SUCCESSIVE CORRELATIONS OF X(I)
*              WITH A SET OF OTHER SERIES, SAY Y(I),Z(I),AND W(I). IN
*              THIS CASE PROCOR IS CALLED JUST ONCE TO ESTABLISH THE
*              OBJECT PROGRAM SPECIALIZED TO THE X SERIES.  THEN
*              ANOTHER ENTRY IS CALLED SUCCESSIVELY FOR EACH OF THE
*              OTHER SERIES.  (AFTER THE CALL OF PROCOR Y(I) CAN BE
```

```
*             MOVED INTO THE X(I) AREA TO SAVE SPACE IF NECESSARY.)
*
*             BY PROPER SEQUENCES OF CALLS THESE PROGRAMS CAN BE USED
*             TO PRODUCE AUTOCORRELATIONS, CROSS CORRELATIONS, AND
*             CONVOLUTIONS, FOR EITHER EQUAL LENGTH OR UNEQUAL LENGTH
*             SERIES, AND FOR EITHER THE TRANSIENT OR FOR THE EQUAL-
*             PRODUCTS ASSUMPTION.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709, OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 770 REGISTERS
* SPEED     - PROCOR TAKES ABOUT
*                    64*LX + 78*MAXX  MACHINE CYCLES
*                    WHERE MAXX = MAXIMUM MAGNITUDE OF X(I)
*             FASCOR TAKES ABOUT
*                120*MAXX + (KMAX+1)*(2*LX - KMAX + 20*MAXX)
*                      + KMN*(2*LX - KMN + 20*MAXX)  MACHJ CYCLES
*                    WHERE KMN = MAGNITUDE OF KMIN
*             FASEPC TAKES ABOUT
*               120*MAXX + (KMAX+KMN+1)*(2*LX+9*MAXX) MACH. CYCLES
*             FASCR1 TAKES THE SAME TIME AS FASCOR
*             FASEP1 TAKES THE SAME TIME AS FASEPC
*
* AUTHOR    - S.M. SIMPSON JR,  10/15/62
*
*
*
*                ----USAGE OF PROCOR-FASCOR-FASEPC-FASCR1-FASEP1----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*      AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE OF PROCOR
*     CALL PROCOR(X,LX,MAXX,PROG1,PROG2,ERR)
*
* INPUTS TO PROCOR
*
*    X(I)      I=1,2,...,LX IS A SERIES OF MACHINE LANGUAGE INTEGERS
*                 (BINARY POINT BEYOND BIT 35).
*              ALL HAVE MAGNITUDES LESS THAN OR = MAXX
*
*    LX        IS A FORTRAN INTEGER GREATER THAN OR = 1
*
*    MAXX      IS A FORTRAN INTEGER = UPPER BOUND TO X(I) SERIES.
*              MUST LIE BETWEEN 1 AND 1000 INCLUSIVELY. FOR MAXIMUM
*                 SPEED MAXX SHOULD BE MADE AS SMALL AS POSSIBLE
*
*    PROG1     WILL BE THE FIRST INSTRUCTION OF THE OBJECT PROGRAM
*                 PROG1 TO PROG2 IS TO BE MADE AVAILABLE FOR THE
*                 PROGRAM WHOSE LENGTH DEPENDS ON BOTH LX AND MAXX
*
*    PROG2     DEFINES HIGH ADDRESS END OF SPACE BLOCK AVAILABLE
*                 FOR PROGRAM. PROG1 AND PROG2 MUST SATISFY
*                 XLOCF(PROG2) - XLOCF(PROG1) EQUALS OR EXCEEDS
*                                                LX + 10*(MAXX+1)
*
* OUTPUTS FROM PROCOR
*
*              THE PRINCIPLE OUTPUT IS THE PROGRAM STORED IN
*              MACHINE ADDRESSES  PROG1, PROG1+1, ...
*
*    ERR       = 0.0 IF NO TROUBLE ARISES
*              = 1.0 IF OBJECT PROGRAM HAS INADEQUATE SPACE
*              = 2.0 IF ILLEGAL LX
*              = 3.0 IF SOME X(I) EXCEEDS MAXX
*              = 4.0 IF MAXX IS ILLEGAL
*
*
* FORTRAN USAGE OF FASCOR
*     CALL FASCOR(Y,KMIN,KMAX,CORZER,ERROR)
*
* INPUTS TO FASCOR
*              FASCOR ASSUMES PROCOR HAS ESTABLISHED ITS OBJECT PROGRAM
*
*    Y(I)      I=1...LX IS A SERIES OF MACHINE LANGUAGE INTEGERS TO BE
```

PROCOR (PAGE 3)

```
*                 CORRELATED WITH X(I).  Y(I) DOES NOT HAVE TO BE
*                 BOUNDED BY MAXX AS X(I) IS, BUT FIXED POINT OVERFLOW
*                 IS POSSIBLE.
*
*    KMIN      IS LARGEST NEGATIVE LAG DESIRED IN CORRELATION
*              IS A FORTRAN INTEGER EXCEEDING -LX AND LSTHN= 0
*
*    KMAX      IS LARGEST POSITIVE LAG DESIRED IN CORRELATION
*              IS A FORTRAN INTEGER GRTHN=0 AND LSTHN LX
*
* OUTPUTS FROM FASCOR
*
*    CORZER(I) I= -KMN+1,-KMN+2,...,0,1,...,KMAX+1 WILL CONTAIN
*                 THE CROSS PRODUCTS XP(KMIN,...,KMAX) WHERE XP(K)
*                 IS DEFINED IN THE ABSTRACT ABOVE,
*                 AND KMN = MAGNITUDE OF KMIN.
*                (THIS STORAGE FORMAT PLACES XP(0) IN CORZER(1).)
*                 THE CROSS-PRODUCTS ARE MACHINE LANGUAGE INTEGERS
*                 AS ARE X AND Y. OVERFLOW IS POSSIBLE IF Y DOES NOT
*                 HAVE REASONABLE BOUNDS. (PROGRAM EXITS IMMEDIATELY
*                 WHEN AN OVERFLOW IS DETECTED.) OVERFLOW IS IMPOSSIBLE
*                 IF LX*MAXX*MAXY IS LESS THAN 2EXP35 (APPROX 3*10EXP10)
*                 WHERE MAXY = MAXIMUM Y MAGNITUDE.
*
*    ERROR     = 0.0  NORMALLY
*              = 1.0  IF OBJECT PROGRAM NOT YET WRITTEN
*              = 2.0  FOR ILLEGAL KMIN OR KMAX  (NO COMPUTATIONS MADE)
*              = 3.0  IF OVERFLOW OCCURS AT SOME LAG.  (IF THIS
*                       HAPPENS PROCOR MUST BE OPERATED AGAIN
*                       BEFORE CALLING FASCOR AGAIN.  (FASCOR FAILS TO
*                       DETECT ONE KIND OF OVERFLOW - SEE NOTES ON
*                       EXAMPLES 1., 19., AND 20. BELOW)
*
* FORTRAN USAGE OF FASEPC
*     CALL FASEPC(Y,KMIN,KMAX,CORZER,ERROR)
*
* INPUTS TO FASEPC
*
*              IDENTICAL TO THOSE OF FASCOR EXCEPT THAT THE
*              MAGNITUDES OF KMIN AND KMAX ARE NOT RESTRAINED BY
*              ANY UPPER BOUND.
*
* OUTPUTS FROM FASEPC
*
*              IDENTICAL TO THOSE OF FASCOR EXCEPT THAT THE
*              COMPUTATION OF XP(K) DOES NOT ASSUME  THAT Y(L) = 0
*              WHEN L IS OUTSIDE THE INCLUSIVE RANGE 1 TO LX .
*
* FORTRAN USAGE OF FASCR1
*     CALL FASCR1(Y,KMIN,KMAX,CORZER,ERROR)
*
* INPUTS TO FASCR1
*
*              IDENTICAL TO THOSE OF FASCOR
*
* OUTPUTS FROM FASCR1
*
*              IDENTICAL TO THOSE OF FASCOR EXCEPT THAT XP IS ADDED TO
*              CORZER, IE   CORZER(I) = CORZER(I) + XP(I-1)  .
*
* FORTRAN USAGE OF FASEP1
*     CALL FASEP1(Y,KMIN,KMAX,CORZER,ERROR)
*
* INPUTS TO FASEP1
*
*              IDENTICAL TO THOSE OF FASEPC.
*
* OUTPUTS FROM FASEP1
*
*              IDENTICAL TO THOSE OF FASEPC EXCEPT THAT XP IS ADDED TO
*              CORZER, IE   CORZER(I) = CORZER(I) + XP(I-1)  .
*
* EXAMPLES
*
*              THE NOTATION MLI, USED BELOW, STANDS FOR MACHINE LANGUAGE
```

```
*              INTEGERS, I.E. FIXED POINT INTEGERS WITH BINARY POINT TO
*              RIGHT OF BIT 35.  OTHERWISE FORTRAN II CONVENTIONS ARE
*              USED WITH RESPECT TO NUMERICAL CONSTANT REPRESENTATION,
*              TO INDEXING OF VARIABLES, ETC.
*              THE OUTPUTS ERR AND ERROR ARE =0.0 IN THE EXAMPLES BELOW
*              UNLESS OTHERWISE STATED.
*              THE INPUTS IN ALL EXAMPLES ARE THE SAME AS
*              THOSE OF EXAMPLE 1. UNLESS OTHERWISE STATED.
*
* 1. ILLUSTRATION OF OBJECT PROGRAM FORMAT-(THIS EXAMPLE ONLY FOR THOSE
*              INTERESTED IN PROCOR LOGIC)
*    INPUTS  - SET X(1...20)=   MLI 1,0,-3,3,-3,-0,1,0,-3,-3,0,0,...
*                  Y(1...20)=   MLI 1,1,1,1,1,0,0,0,0,0,0,1,0,0,0,...
*                  Z(1...20)=   MLI 10,10,10,10,10,0,0,0,0,0,0,0,10,0,0,...
*                  COR(1...50)= MLI 0,0,...      (I.E. CLEAR OUTPUT AREA)
*    USAGE   -       CALL PROCOR(X,10,3,SPACE(100),SPACE(1),ERR)
*    OUTPUTS - THE FAP OBJECT PROGRAM BELOW
*                  SPACE(100)   PZE   3            N+1=11
*                    .          PZE   2
*                    .          PZE   1
*                    .          PZE   0
*                  SPACE(96)    CLA   9,1
*                    .          SUB   5,1
*                    .          ADD   3,1
*                    .          PZE   0
*                    .          PZE   0
*                    .          PZE   0
*                    .          PZE   0
*                    .          PZE   0
*                  SPACE(88)    CLA   10,1         (SPACE(88) IS ENTRY PT.
*                    .          ADD   4,1            TO OBJECT RROGRAM)
*                    .          XCA
*                    .          MPY   SPACE(98)
*                    .          XCA
*                    .          ADD   SUM          (SUM IS AN INTERNAL
*                    .          STO   SUM            ADDRESS IN PROCOR)
*                  SPACE(81)    CLS   8,1
*                    .          ADD   7,1          (NOTE-NO BLOCK EXISTS
*                    .          SUB   6,1            FOR MAGNITUDES X(I)=
*                    .          SUB   2,1            2)
*                    .          SUB   1,1
*                    .          XCA
*                    .          MPY   SPACE(100)
*                    .          XCA
*                    .          ADD   SUM
*                    .          STO   SUM
*                  SPACE(71)    TRA   1,4
*                  SPACE(70) THRU SPACE(66) = 0 SINCE NO X = 2 OR -2
*                  SPACE(65) THRU SPACE(49) IS TABLE SPACE FOR FASCOR.
*                  SPACE(48) THRU SPACE(1) IS EXTRA SPACE NOT USED.
*              (NOTE THAT IF THE RESULT OF AN MPY INSTRUCTION EXCEEDS
*              35 BITS THIS SHOULD BE CONSIDERED AN OVERFLOW BUT IT
*              WILL NOT BE CAUGHT.)
*
* 2. COMPLETE TRANSIENT CROSS-CORRELATION OF X(1...5) WITH Y(1...5)
*    USAGE   -       CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                    CALL FASCOR(Y,-4,4,COR(5),ERROR)
*    OUTPUTS - COR(1...19) = MLI -3,0,-3,-3,-2,1,-2,1,1,0,0,...
*
* 3. COMPLETE TRANSIENT AUTO-CORRELATION OF X(1...5)
*    USAGE   -       CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                    CALL FASCOR(X,0,4,COR(1),ERROR)
*    OUTPUTS - COR(1...19) = MLI 28,-18,6,3,-3,0,0,...
*
* 4. PARTIAL TRANSIENT AUTO-CORRELATION OF X(1...5)
*    USAGE   -       CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                    CALL FASCOR(X,0,2,COR(1),ERROR)
*    OUTPUTS - COR(1...19) = MLI 28,-18,6,0,0,...
*
* 5. PARTIAL TRANSIENT CROSS-CORRELATION WITH REUSE OF X(1...5), AFTER
*              PROCOR, TO STORE CORRELATION
*    USAGE   -       CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                    CALL FASCOR(Y,-2,2,X(3),ERROR)
*    OUTPUTS - X(1...15) = MLI -3,-3,-2,1,-2,-0,1,0,-3,-3,0,0,...
*
```

PROCOR

(PAGE 5)


```
* 6. REPEATED PARTIAL TRANSIENT CROSS-CORRELATION OF X(1...5) WITH
*              Y(1...5) AND WITH Z(1...5)
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASCOR(Y,-2,2,COR(5),ERROR)
*                       CALL FASCOR(Z,-2,2,COR(15),ERROR)
*    OUTPUTS - COR(1...20) = MLI 0,0,-3,-3,-2,1,-2,0,0,0,0,0,-30,-30,
*                              -20,10,-20,0,0,0
*
* 7. REPEATED PARTIAL TRANSIENT CROSS-CORRELATION OF X(1...5) WITH
*              Y(1...5) AND WITH Z(1...5) WITH SUMMATION OF OUTPUT
*              CORRELATIONS.
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASCOR(Y,-2,2,COR(3),ERROR)
*                       CALL FASCR1(Z,-2,2,COR(3),ERROR)
*    OUTPUTS - COR(1...20) = MLI -33,-33,-22,11,-22,0,0,...
*
* 8. EQUI-PRODUCTS CORRELATION OF X(1...5) WITH Y(I) SUCH THAT I STAYS
*              IN THE RANGE 1...20
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASEPC(Y(16),-15,0,COR(16),ERROR)
*    OUTPUTS - COR(1...16) = MLI -2,1,-2,1,1,0,0,-3,3,-3,0,1,0,0,0,0
*
* 9. EQUI-PRODUCTS CORRELATION OF X(1...5) WITH Y(I) AND WITH Z(I),
*              I IN THE RANGE 1...20, WITH SUMMATION OF OUTPUT
*              CORRELATIONS.
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASEPC(Y(16),-15,0,COR(16),ERROR)
*                       CALL FASEP1(Z(16),-15,0,COR(16),ERROR)
*    OUTPUTS - COR(1...16) = MLI -22,11,-22,11,11,0,0,-33,33,-33,0,11,0,
*                              0,0,0
*
*10. COMPLETE TRANSIENT CROSS-CORRELATIONS OF UNEQUAL LENGTH SERIES,
*              X(1...5) WITH Y(1...12), BY INSERTION OF LEADING AND
*              TERMINAL ZEROES AND USING FASEPC
*    INPUTS  - X(1...5) AND Y(1...12) AS IN EXAMPLE 1. INSERT 4 ZEROES
*              AT EACH END OF Y(I) BY LETTING W(1...4)=0 W(5...16)=
*              Y(1...12)  W(17...20)=0
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASEPC(W(16),-15,0,COR(16),ERROR)
*    OUTPUTS - COR(1...16) = MLI -3,0,-3,-3,-2,1,-2,1,1,0,0,-3,3,-3,0,1
*
*11. COMPLETE TRANSIENT CROSS-CORRELATIONS OF UNEQUAL LENGTH SERIES,
*              X(1...5) WITH Y(1...12) USING FASCOR FOR END EFFECTS AND
*              FASEPC FOR CENTRAL VALUES
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                       CALL FASCOR(Y,-4,0,COR(5),ERROR)
*                       CALL FASEPC(Y(7),-5,0,COR(11),ERROR)
*                       CALL FASCOR(Y(8),0,4,COR(12),ERROR)
*    OUTPUTS - COR(1...16) = MLI -3,0,-3,-3,-2,1,-2,1,1,0,0,-3,3,-3,0,1
*              NOTE- THE GENERAL FORMAT IN THIS CASE FOR
*              X(1...LX) Y(1...LY) WITH LX LSTHN LY IS
*                              N1=-LX+1
*                              N2= LY-LX
*                              N3=-(LY-LX-2)
*                              N4= LY-1
*                              N5= LY-LX+1
*                              N6= LX-1
*                              CALL PROCOR(X,LX,MAXX,PROG1,PROG2,ERR)
*                              CALL FASCOR(Y,N1,0,COR(LX),ERROR)
*                 (OMIT IF N3=1) CALL FASEPC(Y(N2),N3,0,COR(N4),ERROR)
*                              CALL FASCOR(Y(N5),0,N6,COR(LY),ERROR)
*                 WHICH LEAVES CORRELATION IN COR(1,2,...,LX+LY-1)
*
*12. COMPLETE CONVOLUTION OF TWO UNEQUAL LENGTH SERIES X(1...5) WITH
*              Y(1...12)
*    INPUTS  - SAME AS EXAMPLE 1. EXCEPT REVERSE X(I) I.E. X(1...5)=-3,
*                                              3,-3,0,1
*    USAGE   - SAME AS EXAMPLE 11.
*    OUTPUTS - COR(1...16) = MLI 1,1,-2,1,-2,-3,-3,0,-3,0,0,1,0,-3,3,-3
*
*13. EXAMPLES 13. THROUGH 20. ILLUSTRATE ERROR CONDITIONS.
*    USAGE   -          CALL PROCOR(X,5,3,SPACE(45),SPACE(1),ERR)
*    OUTPUTS - ERR=1.0 (SPACE BLOCK TOO SMALL)
*
*14. USAGE   -          CALL PROCOR(X,0,3,SPACE(100),SPACE(1),ERR)
```

```
*               OR     CALL PROCOR(X,15000,3,SPACE(20000),SPACE(1),ERR)
*     OUTPUTS - ERR=2.0  (ILLEGAL LX)
*
*15. USAGE   -         CALL PROCOR(X,5,2,SPACE(100),SPACE(1),ERR)
*     OUTPUTS - ERR=3.0  (X(I) FOUND GREATER THAN MAXX)
*
*16. USAGE   -         CALL PROCOR(X,5,0,SPACE(100),SPACE(1),ERR)
*               OR     CALL PROCOR(X,5,1100,SPACE(5000),SPACE(1),ERR)
*     OUTPUTS - ERR=4.0  (ILLEGAL MAXX)
*
*17. USAGE   -         CALL PROCOR(X,0,3,SPACE(100),SPACE(1),ERR)
*                      CALL FASCOR(Y,-4,4,COR(5),ERROR)
*     OUTPUTS - ERR=2.0 (SAME AS EXAMPLE 14.)
*               ERROR = 1.0 (NO OBJECT PROGRAM)
*
*18. USAGE   -         CALL PROCOR(X,5,3,SPACE(100),SPACE(1),ERR)
*                      CALL FASCOR(Y,-5,4,COR(10),ERROR)
*               OR     CALL FASCOR(Y, 2,4,COR(10),ERROR)
*               OR     CALL FASCOR(Y,-4,6,COR(10),ERROR)
*               OR     CALL FASCOR(Y,-4,-1,COR(10),ERROR)
*     OUTPUTS - ERR=0.0
*               ERROR=2.0  (ILLEGAL KMIN OR KMAX)
*
*19. INPUTS  - X(1...5) = MLI 20,40,60,80,100
*               Y(1...5) = MLI 200000000,200000000,...  =OCT 001372741000
*     USAGE   -        CALL PROCOR(X,5,100,SPACE(1016),SPACE(1),ERR)
*                      CALL FASCOR(Y,0,0,COR,ERROR)
*               OR     CALL FASEPC(Y,0,0,COR,ERROR)
*               OR     CALL FASCR1(Y,0,0,COR,ERROR)
*               OR     CALL FASEP1(Y,0,0,COR,ERROR)
*     OUTPUTS - ERR=0.0  ERROR=3.0  (OVERFLOW CAUGHT, COR(1)=6*10EXP10)
*
*20. INPUTS  - SAME AS EXAMPLE 19. EXCEPT  X(1...5) = 60,60,60,60,60
*     USAGE   - SAME AS EXAMPLE 19.
*     OUTPUTS - ERR=0.0  ERROR=0.0  (OVERFLOW NOT CAUGHT)
*
*21. SPECIAL CASE TEST - ALL X(I) ARE ZERO (PROGRAM HAS BYPASS FOR THIS)
*     INPUTS  - X(1...5) = MLI 0,0,...
*               Y(1...10) = MLI 5,4,3,2,1,0,-1,-2,-3,-4
*               COR(1...) = MLI 1,2,3,4,5,6,7,8,...
*     USAGE   -        CALL PROCOR(X,5,10,SPACE(200),SPACE,ERR)
*                      CALL FASCOR(Y,0,4,COR,ERROR)
*               OR     CALL FASEPC(Y(5),-4,0,COR(5),ERROR)
*     OUTPUTS - COR(1...) = MLI 0,0,0,0,0,6,7,8,...
*
*22. INPUTS  - SAME AS EXAMPLE 21.
*     USAGE   -        CALL PROCOR(X,5,10,SPACE(200), SPACE,ERR)
*                      CALL FASCR1(Y,0,4,COR,ERROR)
*               OR     CALL FASEP1(Y(5),-4,0,COR(5),ERROR)
*     OUTPUTS - COR(1...) = MLI 1,2,3,4,5,6,7,8,...
*
*23. SPECIAL CASE TEST -  X SERIES HAS LENGTH = 1 .
*     INPUTS  - SAME AS EXAMPLE 21. EXCEPT X(1) = MLI 2
*     USAGE   -        CALL PROCOR(X,1,10,SPACE(200),SPACE,ERR)
*                      CALL FASCOR(Y,0,0,COR,ERROR)
*     OUTPUTS - COR(1...) = MLI 10,2,3,4,5,6,7,8,...
*
*24. INPUTS  - SAME AS EXAMPLE 23 .
*     USAGE   -        CALL PROCOR(X,1,10,SPACE(200),SPACE,ERR)
*                      CALL FASEPC(Y(5),-4,2,COR(5),ERROR)
*     OUTPUTS - COR(1...) = MLI 10,8,6,4,2,0,-2,8,9,...
*
*
* PROGRAM FOLLOWS BELOW
*               NOTATION DIFFERENCES BELOW ARE
*                   N = LX
*                   POSMAX = KMAX
*                   NEGMAX = KMIN
*                   T = K = LAG NUMBER
*                   S=MAXX
*
        HTR     0
        HTR     0
        HTR     0
        BCI     1,PROCOR
```

```
PROCOR SXD      PROCOR-4,1
       SXD      PROCOR-3,2
       SXD      PROCOR-2,4
*GET X ADDRESS AND SET X+1 ADDRESSES
       CLA      1,4
       STA      T1
       ADD      K1
       STA      X20
       STA      X41
       STA      X44
       STA      X48
* GET N VALUE AND SET DECREMENTS, ETC (AFTER CHECKING N)
       CLA      K5             K5=2.0
       STO      T27
       CLA*     2,4            =N
       STO      T2
       TMI      X65
       TZE      X65
       CAS      K8             K8= 0,0,10000
       TRA      X65
       NOP
       STD      X24
       STD      X53
       ARS      18
       STA      T7
       STA      T11
* GET S VALUE AND SET DECREMENTS (AFTER CHECKING S)
       CLA      K29            K29= 4.0
       STO      T27
       CLA*     3,4            =S
       STO      T3
       TMI      X65
       TZE      X65
       CAS      K30            K30= 0,0,1000
       TRA      X65
       NOP
       STD      X36
       STD      X63
*MAKE OBJECT PROGRAM SIZE CHECK
       CLA      K4             K4= 1.0
       STO      T27
       CLA      4,4
       STA      T4
       CAL      5,4            FORM
       ANA      K20
       SUB      T4             PROG2-PROG1
       STA      T17
       CLA      T3             GET S (IN DECR)
       ADD      K7             S+1
       XCA
       MPY      K31            K31= 10
       XCA                     10(S+1)
       ADD      T2             N+10(S+1)
       ARS      18             MOVE TO ADDRESS
       CAS      T17            COMPARE WITH PROG2-PROG1
       TRA      X65
       TRA      *+1            OK
* CLEAR PROG1 THRU PROG1+N+10(S+1)
* (LOOP TAKES 3(N+10(S+1)+1) HI SPEED INSTRUCTIONS)
       ADD      K1             N+10(S+1)+1
       ADD      T4             PROG1 + DITTO
       STA      X2
       SUB      T4
       ALS      18             N+10(S+1)+1 IN DECR
       STD      X3
       LXD      K7,1           K7= 0,0,1
 X2    STZ      **,1       **=PROG1+N+10(S+1)+1
       TXI      *+1,1,1
 X3    TXL      X2,1,**   **=N+10(S+1)+1
*SET ADDRESSES=PROG1+S AND PROG1+S+1
       CLA      T3
       ARS      18
       ADD      T4
       STA      X62
       STA      T9             DEFINES TABLE
```

```
        ADD     K1
        STA     T12
*SET ADDRESSES=TABLE,TABLE+1,TABLE-S
        CLA     T9              TABLE
        STA     X25A
        STA     X25
        STA     X21
        STA     X22
        STA     X24A
        STA     X26
        STA     X29
        STA     X30
        STA     X31
        STA     X42
        STA     X43
        STA     X47
        ADD     K1              TABLE+1
        STA     X28
        STA     X32
        CLS     T3
        ARS     18
        ADD     T9              TABLE-S
        STA     X37
* FORM ADDRESSES OF TABL1,TABL2,TABL3,TABL4
        CLA     T3              SET S+1
        ADD     K7
        ARS     18
        STA     T18
        ALS     2               4(S+1)
        ADD     T18             5(S+1)
        ADD     T18             6(S+1)
        ADD     T12             PROG1+S+1+6(S+1)
        ADD     T11             PROG1+S+1+N+6(S+1)=TABL1
        STA     T13
        ADD     T18             TABL2
        STA     T14
        ADD     T18             TABL3
        STA     T15
        ADD     T18             TABL4
        STA     T16
        CLA     T13             SET TABL1 ADDRESSES
        STA     X22A
        STA     X28A
        STA     X26A
        STA     X33B
        SUB     K1
        STA     X37A
*SCAN X VALUES TO MAKE FREQUENCY COUNT OF MAGNITUDES IN
*DECREMENTS OF TABLE AND TABL1, CHECKING FOR EXCESSIVE X MAGNITUDES
*(THIS LOOP TAKES 10N HI SPEED INSTRUCTIONS)
  X19   AXT     1,2
        CLA     K6              K6= 3.0
        STO     T27
  X20   CLA     **,2      (**=X+1)           GET MAGNITUDE OF NEXT X VALUE
        SSP                                  AND INCREMENT
        CAS     T18             T18= S+1
        NOP
        TRA     X65
        PAX     0,4                          THE
  X21   CLA     **,4      (**=TABLE)         PROPER
        ADD     K7       (K7=PZE0,0,1)       COUNTER
  X22   STO     **,4      (**=TABLE)         BY ONE
 X22A   STO     **,4            (**=TABL1)
  X23   TXI     *+1,2,1
  X24   TXL     X20,2,**        (**=N)
        STZ     T27             ALL OK
*CHECK IF ALL X(I) = 0  .  IF SO SET FASCOR BYPASS SWITCH AND EXIT
        CLA     K1
        STO     T29
 X24A   CLA     **        **=TABLE     N(0)
        CAS     T2              T2= N
        HPR     *               IMPOSSIBLE
        TRA     X65             BYPASS
        STZ     T29             OK
*NOW SET UP ADDRESS TABLE FOR BLOCKS IN OBJECT PROG. AND INSERT 5-GROUPS
```

```
*SEE FORMAT OF THIS TABLE IN NOTES BELOW
* (NOTE XCA(0) = LOC(0) + N(0)  IF N(0) NON ZERO
*               = LOC(0) - 5 IF N(0) = 0
*         LOC(0) = PROG1 + S + 1 )
 X25   CLA     **             (**=TABLE)      GET N(0)
       ARS     18
       TZE     X26B
 X25B  ADD     T12            N(0)+PROG1+S+1
 X25A  ADD     **             (**=TABLE)
       SSM                    MZE  XCA(0),0,N(0)
 X26   STO     **             (**=TABLE)
       CLA     T12                           SET LOC(0)
 X26A  STA     **             (**=TABL1)      IN TABL1
       TRA     X26C
 X26B  CLS     K9             -5 IF N(0)=0
       TRA     X25B
 X26C  AXT     1,4            SET I=1
*NOW FORM XCA(I) AND LOC(I) RECURSIVELY BY
*         LOC(I) = XCA(I-1) + 5
*         XCA(I) = XCA(I-1) + 5 + N(I)  IF N(I) NOT =0
*                = XCA(I-1)             IF N(I) = 0
*(LOOP TAKES ABOUT 25S HI SPEED INSTRUCTIONS)
*   SET LOC(I)
 X28   CLS     **,4      **=TABLE+1      PZE XCA(I-1),0,N(I-1)
       ADD     K9        K9=5            PZE   LOC(I),0,N(I-1)
 X28A  STA     **,4      **=TABL1    SET LOC(I)=XCA(I-1)+5
*NOW CHECK N(I)
 X29   CLA     **,4      **=TABLE        PZE          0,0,N(I)
       TNZ     X30A
*SET XCA(I) FOR N(I) = 0
 X32   CAL     **,4      **=TABLE+1      MZE XCA(I-1),0,N(I-1)
       ANA     KMSK2                     MZE XCA(I-1),0,0
 X30   SLW     **,4      **=TABLE      = MZE   XCA(I),0,N(I)
       TRA     X35
*SET XCA(I) FOR N(I) NOT = 0
 X30A  ARS     18                        PZE N(I),0,0
 X33B  ADD     **,4      **=TABL1        PZE LOC(I)+N(I),0,N(I)
       SSM                               MZE   XCA(I),0,N(I)
 X31   STO     **,4      **=TABLE
*SET STORAGE ADDRESS IN XR2 FOR XCA,MPY,XCA,ADD,STO GROUP
       PAC     0,2            -XCA(I) IN XR2
*SET ADDRESS OF THE MPY INSTRUCTION IN THIS GROUP
       SXA     T5,4
       CLA     X62
       SUB     T5
       STA     K14            PROG1+S-I IN ADDR. OF K14
*NOW MOVE THE GROUP INTO POSITION IN OBJECT PROGRAM
*(NOTE THAT NO GROUP IS INSERTED FOR I=0 BLOCK )
       CLA     K11       K11=XCA
       STO     0,2
       STO     2,2
       CLA     K14       K14=MPY PROG1+S-I
       STO     1,2
       CLA     K12       K12=ADD SUM
       STO     3,2
       CLA     K13       K13= STO SUM
       STO     4,2
*CHECK COMPLETION
  X35  TXI     *+1,4,1        INCREASE I BY 1
 X36   TXL     X28,4,**       **=S
*WHEN DONE FILL IN TRA 1,4 INSTRUCTION AND SET ENTRY ADDRESS
  X37  LAC     **,4           (**=TABLE-S)
       CLA     K15            K15=TRA 1,4
       STO     5,4
 X37A  CLA     **        **=TABL1-1
       STA     T8             ENTRY = LOC(1)
*MAIN LOOP, SCANS X(I) AGAIN AND FILLS IN REMAINDER OF OBJECT PROGRAM
*(THIS LOOP TAKES 19N HI SPEED INSTRUCTIONS)
*SET X INDEX I =1
  X40  AXT     1,1
*GET NEXT X(I) VALUE
  X41  CLA     **,1           (**=X+1)
*SET MAGN(X) IN XR4 AND GET TABLE ENTRY (MAGN(X))
       PAX     0,4
 X42   CLA     **,4      **=TABLE
```

PROCOR
(PAGE 10)

```
*SET FOR STORAGE OF NEXT INSTRUCTION OF OBJECT PROGRAM
       STA     X51
       PDX     0,2
*IF TABLE ENTRY NEG, THIS IS FIRST X OF THIS MAGNITUDE
       TMI     X46
*IF POSITIVE WE WANT ADD OR SUB INSTRUCTION
* FIRST REDUCE DECR OF TABLE, THEN CHECK SIGN OF X(I)
       SUB     K7             K7=PZE0,0,1
  X43  STO     **,4           (**=TABLE)
  X44  CLA     **,1           (**=X+1)
       TMI     X45
*ADD  INSTRUCTION NEEDED
       CLA     K17
       TRA     X50
*SUB INSTRUCTION NEEDED
  X45  CLA     K19
       TRA     X50
*FOR FIRST X OF THIS MAGN WE WANT CLA OR CLS INSTRUCTION
*FIRST CHANGE SWITCH, REDUCE DECR OF TABLE, THEN CHECK SIGN OF X(I)
  X46  SSP
       SUB     K7             K7=PZE0,0,1
  X47  STO     **,4           (**=TABLE)
  X48  CLA     **,1           (**=X+1)
       TMI     X49
*CLA INSTRUCTION NEEDED
       CLA     K16
       TRA     X50
*CLS INSTRUCTION NEEDED
  X49  CLA     K18
       TRA     X50
*SUPPLY ADDRESS TO INSTRUCTION AND STORE IT
*NOTE CLA,ADD,CLS,SUB, ARE ALL POSITIVE NUMBERS
  X50  ADD     T7             T7=N+1-I
 X51   STO     **,2       **=XCA(MAGN(X(I)))
*INCREMENT I,N+1-I, AND CHECK FOR FINISH
  X52  CLA     T7
       SUB     K1
       STO     T7
       TXI     *+1,1,1
  X53  TXL     X41,1,**       (**=N)
*END OF MAIN LOOP
*NOW FILL IN INTEGER TABLE  IN PROG1 TO PROG1+S
*(LOOP TAKES 4(S+1)  HI SPEED INSTRUCTIONS)
 X60   AXT     0,4
  X61  PXA     0,4
   X62 STO     **,4           (**=PROG1+S)
       TXI     *+1,4,1
   X63 TXL     X61,4,**       (**=S)
*RESTORE INDEX REGISTERS, SET ERR, AND EXIT
 X65   LXD     PROCOR-4,1
       LXD     PROCOR-3,2
       LXD     PROCOR-2,4
       CLA     T27
       STO*    6,4
       TRA     7,4
*
*CONSTANTS FOR PROCOR, FASCOR
  K1   PZE     1
  K2   PZE     10000
  K3   PZE     500
  K4   DEC     1.0
  K5   DEC     2.0
  K6   DEC     3.0
  K7   PZE     0,0,1
  K8   PZE     0,0,10000
  K9   PZE     5
  K10  PZE     0,0,2
  K11  XCA
  K12  ADD     SUM
  K13  STO     SUM
  K14  MPY     **             (**=PROG1+S-I)
 K15   TRA     1,4
  K16  CLA     0,1
  K17  ADD     0,1
  K18  CLS     0,1
```

```
  K19  SUB    0,1
  K20  OCT    77777
 K21   TRA    0
 K22   TRA    **            (**=LOC(B)+N(B))
 K23   OCT    +050100000000    SEPARATES CLA FROM CLS
 K24   OCT    +040100000000   SEPARATES ADD FROM SUB
 K25   ADD    0,1           =040000100000
 K26   SUB    0,1           =040200100000
 K27   CLA    0,1           =050000100000
 K28   CLS    0,1           =050200100000
 K29   DEC    4.0
 K30   PZE    0,0,1000
 K31   PZE    10
  KMSK OCT    77777
 KMSK2 OCT    400000077777
*TEMPORARIES
  T1   PZE    **       **=X
  T2   PZE    0,0,**   **=N
  T3   PZE    0,0,**   **=S
 T4    PZE    **            **=PROG1
  T5   PZE    **       **=I DURING TABLE FORMING LOOP
  T6   MZE    **       **=XCA(I-1) DURING DITTO
  T7   PZE    **       **=N+1-I FOR MAIN LOOP (INITIAL=N)
  T8   PZE    **       **=ENTRY TO OBJECT PROGRAM FOR FASCOR = LOC(1)
  T9   PZE    **       **= TABLE = PROG1 + S
 T10   PZE    **            SPARE
 T11   PZE    **            **=N FOR FASCOR
 T12   PZE    **            **=PROG1+S+1
 T13   PZE    **            **=TABL1
 T14   PZE    **            **=TABL2
 T15   PZE    **            **=TABL3
 T16   PZE    **            **=TABL4
 T17   PZE    **            **=PROG2-PROG1
 T18   PZE    **            **=S+1
 T19   PZE    0,0,**        **=POSMAX
 T20   PZE    0,0,**     **=MAGN OF NEGMAX
 T21   PZE    **         **=CORZER
 T22   PZE    **         **=Y
 T23   PZE    **         **=Y-N
 T24   PZE    **         **=LOC(I) DURING RESTORATION AFTER NEG LAGS
 T25   PZE    **         **=N(I)   DURING RESTORATION AFTER NEG LAGS
 T26   PZE    **         **=MAGN OF NEGMAX
  T27  PZE    **       **= PROCOR ERR SETTING
  T28  PZE    **       **= N+10(S+1)+1
  T29  PZE    **       **= 0 (OK), = 1 (BYPASS)
  T30  PZE    **       **= FASCOR ERROR SETTING
  SUM  PZE    **       **=CROSS PRODUCTS SUM
 NXTXI PZE    **         **=N+1-I WHERE I=INDEX OF NEXT X TO DELETE
*                           (INIT =1 (POS LAGS), = N (NEG LAGS))
PNEWBT PZE    **         **=INSTR INDEX IN BLOCK NEWB FOR LAG T
NEWINS NOP                  HOLDS INSTRUCTION TO BE SET ASIDE
LOCNWB PZE    **         **=ADDRESS OF NEW BLOCK = LOC(NEWB)
 NNEWB PZE    **         **N VALUE OF NEW BLOCK = N(NEWB)
LOCOLX PZE    **       **= -(TABL2-LASTB)   (INIT = -(TABL2-S/2)
   LAG PZE    **         **=LAG T
*
*
*FORMAT OF FINAL OBJECT PROGRAM IS
*
*PROG1 PZE    S
*      PZE    S-1
*       ETC
* +S+1 CLA OR CLS   N+1-**,1       ZEROTH           THE ** ARE
*      ADD OR SUB   N+1-**,1       BLOCK            FORTRAN CONVENTION
*        ETC                       (NOT ACTUALLY    INDICES OF THE X(I)
*      ADD OR SUB   N+1-**,1       OPERATED, BUT    SERIES.  THE FIRST
*      PZE 0                       NEEDED FOR       BLOCK CONTAINS ALL
*      PZE 0                       SUBSTITUTION     INDICES FOR WHICH
*      PZE                         LOGIC OF         THE MAGNITUDE OF X(I)=1
*      PZE 0                       FASCOR)          THE SECOND BLOCK
*      PZE 0                                        ALL INDICES FOR WHICH
*LOC(1)CLA OR CLS   N+1-**,1       FIRST            THE MAGNITUDE OF X(I)=2
*      ADD OR SUB   N+1-**,1       BLOCK            ETC.  THE SIGN OF
*       ETC                        (ENTER HERE)     X(I) DETERMINES
*      ADD OR SUB   N+1-**,1                        WHETHER ADD (CLA) OR
```

```
*      XCA                                          SUB (CLS) IS USED.
*      MPY      PROG1+S-1                           BLOCKS WILL BE MISSING
*      XCA                                          IF THERE ARE NO X(I)
*      ADD      SUM                                 OF CORRESPONDING
*      STO      SUM                                 MAGNITUDE
*        ETC                                        NOTE THAT THE
*        ETC                                        TOTAL NUMBER OF
*LOC(S)CLA OR CLS   N+1-**,1       LAST            ADD, CLA, SUB, CLS
*      ADD OR SUB   N+1-**,1       BLOCK           INSTRNS OPERATED WILL
*        ETC                                        BE LESS THAN N BY
*      ADD OR SUB   N+1-**,1                       AN AMOUNT EQUAL TO
*      XCA                                          THE NUMBER OF ZEROES
*      MPY      PROG1                               IN THE X SERIES.
*      XCA
*      ADD      SUM                                 FASCOR SETS XR1=-(Y-N),
*      STO      SUM                                 Y IS SECOND SERIES,
*      TRA      1,4                EXIT            BEFORE ENTRY, AND
*                                                   INCREMENTS XR1 FOR LAGS
* HOWEVER EACH TIME FASCOR INCREMENTS FOR LAGS IT MUST DELETE AN
* INSTRUCTION IN THE OBJECT PROGRAM (BY INSERTING A TRA) TO
* PRODUCE THE TRANSIENT EFFECT.  WHEN IT FINISHES IT MUST RESTORE
* THE OBJECT PROGRAM FOR REUSE ON ANOTHER CALL OF FASCOR.
*
*PROCOR SETS UP TWO TABLES, TABLE(I) AND TABL1(I) AS BELOW
* (IT USES TABLE(I) ITSELF. TABL1 IS A SERVICE TO FASCOR)
*
*     TABLE-I = MZE XCA(I),0,N(I)    I=0,1,...,S
* AND
*     TABL1-I = PZE LOC(I),0,N(I)    I=0,1,...,S
* WHERE N(I) = NO. OF X VALUES HAVING MAGNITUDE I
*       XCA(I) = ADDRESS OF FIRST XCA INSTRUCTION IN BLOCK NO. I
*       LOC(I) = ADDRESS OF FIRST INSTRUCTION IN BLOCK NO. I
* RELATIONS BETWEEN LOC(I), XCA(I), AND N(I) NEEDED BY PROCOR LOGIC ARE
*       LOC(0) = PROG1+S+1
*       XCA(0) = LOC(0)+N(0)  IF N(0) NOT = 0
*              = LOC(0) - 5  IF N(0) = 0
*   RECURSION FORMULA FOR LOC(I)  I=1,2,...,S
*       LOC(I) = LOC(I-1)+N(I-1)+5  IF N(I-1) NOT  = 0
*              = LOC(I-1)  IF N(I-1) = 0
*   RECURSION FORMULA FOR XCA(I)  I=1,2,...,S
*       XCA(I) = XCA(I-1)+N(I)+5   IF N(I) NOT = 0
*              = XCA(I-1)    IF N(I) = 0
* IT CAN BE SHOWN BY INDUCTION THAT THE ABOVE LEADS TO
*       LOC(I) = XCA(I-1)+5 IN ALL CASES
* SO THAT
*       XCA(I) = LOC(I)+N(I)    IF N(I) NOT = 0
*              = LOC(I)-5   IF N(I) =0
*
* FASCOR USES TABL1 WITHOUT MODIFYING IT IN ANY WAY. FASCOR ALSO
* SETS UP THREE OTHER TABLES- TABL2, TABL3 AND TABL4.  THESE TABLES
* ARE DESIGNED TO SPEED UP THE SUBSTITUTION LOGIC INVOLVED IN
* THE SUCCESSIVE REPLACEMENT OF INSTRUCTIONS BY TRANSFERS.
*
* DEFINITIONS INVOLVED IN TABLE DESCRIPTIONS
*      B = AN ARBITRARY BLOCK NO. IN OBJECT PROGRAM
*      AD(Y) = ADDRESS PORTION OF INSTRUCTION AT LOCATION Y
*      P(B,Y) = INDEX OF THE INSTRUCTION AT LOCATION Y RELATIVE TO
*               BLOCK B WHERE P(B,LOC(B)) = 1  P(B,LOC(B)+1) = 2 ETC
*      OP(Y) = CONTENTS OF LOCATION Y
*      NXT(B,T) = LOCATION, JUST BEFORE COMPUTATION AT LAG T, OF THE
*                 NEXT INSTRUCTION IN BLOCK B WHICH IS TO BE CONSIDERED
*                 FOR POSSIBLE SUBSTITUTION
*
* TABL2 - ADDRESS PORTIONS OF NEXT POSSIBLE INSTRUCTION TO SET ASIDE
*         FROM BLOCK B
*    THE FOLLOWING FACTS ARE NEEDED TO UNDERSTAND HOW TABL2 WORKS
*      1. AD(Y) IS MONOTONE DECREASING FOR Y=LOC(B),...,LOC(B)+N(B)-1
*      2. OP(Y) CORRESPONDS TO MONOTONE INCREASING X(I) INDICES
*              FOR Y = LOC(B),...,LOC(B)+N(B)-1
*      3. THE VALUE OF AD(Y) MUST LIE BETWEEN 1 AND N INCLUSIVE
*              FOR Y = LOC(B),...,LOC(B)+N(B)-1
*    TABL2 MEANING JUST PRIOR TO SUBSTITUTION ANALYSIS FOR LAG T
*      TABL2-B = PZE AD(NXT(B,T))  IF B STILL HAS UNDELETED ELEMENTS
*              = OCT 000000077777  IF B HAS NO MORE DITTO (POS LAGS)
*              = OCT 000000000000  IF B HAS NO MORE DITTO (NEG LAGS)
```

```
* INITIAL SETTING OF TABL2 FOR POSITIVE LAGS
*      TABL2-B = PZE AD(LOC(B)+N(B)-1)  IF N(B) NOT = 0
*              = OCT 000000077777  IF N(B) = 0 (THIS IS BYPASS SWCH)
* INITIAL SETTING OF TABL2 FOR NEGATIVE LAGS
*      TABL2-B = PZE AD(LOC(B)) IF N(B) NOT  = 0
*              = PZE 0   IF N(B) = 0 ( BYPASS SWITCH)
*
* TABL3 - INDICES OF NEXT POSSIBLE INSTRUCTION TO SET ASIDE FROM BLOCK B
*    MEANING JUST PRIOR TO SUBSTITUTION ANALYSIS FOR LAG T IS
*      TABL3-B = PZE P(B,NXT(B,T))
*    INITIAL SETTING OF TABL3 FOR POSITIVE LAGS
*      TABL3-B = PZE N(B)
*    INITIAL SETTING OF TABL3 FOR NEGATIVE LAGS
*      TABL3-B = 1
*
* TABL4 - STORAGE FOR INSTRUCTION SET ASIDE FROM BLOCK B
*    MEANING JUST PRIOR TO SUBSTITUTION ANALYSIS FOR LAG T IS
*      TABL4-B = 0 IF NO INSTRUCTIONS HAVE BEEN TAKEN FROM BLOCK B YET
*              = OP(NXT(B,T)+1)  FOR POSITIVE LAGS
*              = OP(LOC(B))   NEGATIVE LAGS
*    INITIAL STTING OF TABL4  FOR POSITIVE OR NEGATIVE LAGS
*      TABL4-B = PZE 0
*
       HTR     0
       HTR     0
       HTR     0
       BCI     1,FASCOR
FASCOR SXD     FASCOR-4,1
       SXD     FASCOR-3,2
       SXD     FASCOR-2,4
*CHECK FOR EXISTENCE OF OBECT PROGRAM. (NO IF N FROM PROCOR = 0
*   OR IF ERR FROM PROCOR IS NOT ZERO)
       CLA     K4              K4= 1.0 ERROR SET
       STO     T30
       CLA     T2              N CHECK
       TMI     Y96
       TZE     Y96
       CLA     T27             ERR CHECK
       TNZ     Y96
*NOW CHECK LEGALITIES OF POSMAX, NEGMAX
       CLA     K5              K5= 2.0 ERROR SET
       STO     T30
       CLA*    3,4             POSMAX CHECK
       TZE     *+2
       TMI     Y96
       CAS     T2
       NOP
*(NEXT INSTRUCTION = NOP FOR FASEPC OR FASEP1)
 Y2    TRA     Y96             CANT = N
       STD     T19
       CLS*    2,4             NEGMAX CHECK
       TZE     *+2
       TMI     Y96
       CAS     T2
       NOP
*(NEXT INSTRUCTION = NOP FOR FASEPC OR FASEP1)
 Y3    TRA     Y96
       STD     T20
*SET OUTPUT CLEAR ROUTINE AND CHECK FOR BYPASS ON X(I) ALL ZERO
       ADD     T19
       ADD     K7              POSMAX+MAGN(NEGMAX)+1
       STD     Y95B
       CLA     T20
       ARS     18
       ADD     K1
       ACL     4,4             CORZER+MAGN(NEGMAX)+1
       STA     Y95A
       CLA     T29             BYPASS SWITCH
*(NEXT INSTRUCTION  = TNZ Y95C  FOR FASCR1, FASEP1)
 Y4    TNZ     Y95
*TURN OFF OVERFLOW AND SET ERRORS FOR  POSSIBLE OVERFLOW
*   (OBJECT PROGRAM IS NOT RESTORED IN THIS CASE)
       TOV     *+1
       CLA     K6              K6= 3.0
       STO     T27        (SO THAT PROCOR MUST BE USED AGAIN IF OVERFLOW)
```

```
        STO     T30
   *SET ENTRIES TO OBJECT PROGRAM, CORZER ADDRESSES, Y, Y-N
        CLA     T8
        STA     Y6
        STA     Y25
        STA     Y81
        CLA     4,4
        STA     Y7
        STA     Y26
        STA     Y82
        STA     T21
        CLA     1,4
        STA     T22
        SSP
        SUB     T11
        STA     T23
   *NOW COMPUTE ZERO LAG CORRELATION AND STORE IT
        PAC     0,1             -(Y-N) TO IR1
        STZ     SUM
    Y6  TSX     **,4            **=ENTRY TO OBJECT PROGRAM
        CLA     SUM
   *(NEXT INSTRUCTION =  ADD CORZER  FOR FASCR1 OR FASEP1)
        NOP
    Y7  STO     **              **=CORZER
        TOV     Y96
   *SET DECREMENTS = S
        CLA     T3
        STD     Y8G
        STD     Y33
        STD     Y43
        STD     Y94
   *SET TABL1 ADDRESSES BELOW
        CLA     T13
        STA     Y8A
        STA     Y20
        STA     Y29
        STA     Y38
        STA     Y60
        STA     Y86
   *SET TABL2 ADDRESSES BELOW
        CLA     T14
        STA     Y8D
        STA     Y21B
        STA     Y23A
        STA     Y40
        STA     Y72
        STA     Y78
        ADD     K1              TABL2+1
        ALS     18              IN DECR
        PDC     0,2
        SXD     Y10,2           -(TABL2+1) STORED
        SXD     Y50,2
        SUB     T3              TABL2+1-S
        SUB     K7
        PDC     0,2
        SXD     Y11,2           -(TABL2-S) STORED
        SXD     Y51,2
   *SET TABL3 ADDRESSES BELOW
        CLA     T15
        STA     Y8E
        STA     Y20A
        STA     Y21C
        STA     Y21D
        STA     Y30
        STA     Y42
        STA     Y61
        STA     Y73
        STA     Y74
   *SET TABL4 ADDRESSES BELOW
        CLA     T16
        STA     Y8F
        STA     Y20B
        STA     Y21A
        STA     Y28
        STA     Y31
```

```
        STA     Y41
        STA     Y76
        STA     Y85
        STA     Y87
*SET DECREMENTS DEPENDING ON POSMAX,NEGMAX BELOW
        CLA     T19           POSMAX IN DECR
        STD     Y27
        CLA     T20           MAGN OF NEGMAX IN DECR
        ARS     18
        STA     T26
*ARE POSITIVE LAGS WANTED
        CLA     T19           GET POSMAX
        CAS     K7
        TRA     Y8            YES
        TRA     Y8            YES
        TRA     Y35           NO
*MAKE INITIAL SETTINGS FOR POSITIVE LAGS
 Y8     CLA     K1
        STO     NXTXI         NXTXI=1
        STO     LAG           T=1
        CLA     T3            S
        ARS     19            S/2 IN ADDRESS
        SSM                   -S/2
        ADD     T14           PLUS TABL2
        PAC     0,1
        SXA     LOCOLX,1
*SET UP TABL2,TABL3, AND TABL4 FOR POSITIVE LAGS
*(LOOP TAKES 15(S+1) HI SPEED INSTRUCTIONS)
        AXT     0,1           I=0
 Y8A    CLA     **,1          (**=TABL1)
        PDC     0,2           -N(I) TO XR2
        TXH     Y8B,2,0
        CAL     K20           K20=OCT77777
        TRA     Y8D           SET THIS CONSTANT FOR N(I)=0
 Y8B    SUB     K1
        STA     *+1
 Y8C    CAL     **,2     **=LOC(I)-1
        ANA     K20           EXTRACT AD(LOC(I)+N(I)-1)
 Y8D    SLW     **,1     **=TABL2
        PXA     0,2
        PAC     0,2
        PXA     0,2
 Y8E    STO     **,1          (**=TABL3)   STORE N(I)
 Y8F    STZ     **,1          (**=TABL4)          CLEAR
        TXI     *+1,1,1
 Y8G    TXL     Y8A,1,**      (**=S)
*NOW LOOP THRU ALL POSITIVE LAGS
*THIS LOOP SCANS THE INDICES IN TABL2 TO FIND NEWB, I.E.,
*TO FIND FROM WHICH BLOCK THE NEXT INSTRUCTION IS TO BE DELETED.
*IT STARTS SCANNING FROM INSIDE TABL2 AT THE SAME BLOCK AS THAT
*OF THE PREVIOUS DELETION AND PROCEEDS OUTWARDS IN BOTH DIRECTIONS
*TO MINIMIZE NUMBER OF TRIAL COMPARISONS
*(LOOP Y9 TO Y22 TAKES ABOUT POSMAX*(53+(3 TO 3*(S+1))) HI SPEED
*   INSTRUCTIONS EXCLUSIVE OF OBJECT PROGRAM)
*(NEXT INSTRUCTION  = TRA Y24  FOR FASEPC OR FASEP1)
 Y9     LXA     LOCOLX,1      SET TO EXAMINE NEXT INDEX
        LXA     LOCOLX,2      IN OLD BLOCK FIRST
        CLA     NXTXI         GET INDEX TO COMPARE
*SCAN DOWNWARDS
 Y10    TXL     Y11,1,**      (**=-(TABL2+1))    AVOID OVERSHOOT
        CAS     0,1
        HPR     *             IMPOSSIBLE
        TRA     Y13           GOT IT
        TXI     *+1,1,-1
*AND UPWARDS
 Y11    TXH     Y10,2,**      **=-(TABL2-S)   AVOID UNDERSHOOT
        CAS     0,2
        HPR     *             IMPOSSIBLE
        TRA     Y12
        TXI     Y10,2,1
*WHEN INDEX FOUND REPLACE THE OLD BLOCK ADDRESS BY NEW ONE
*FIND NEWB AND SET XR1 TO NEWB
 Y12    SXA     LOCOLX,2
        TRA     Y14
 Y13    SXA     LOCOLX,1
```

```
 Y14   LAC     LOCOLX,1
       PXA     0,1             TABL2-NEWB IN AC NOW
       SUB     T14             T14=PZE TABL2
       SSP                     NEWB IN AC
       PAX     0,1             SET XR1 FOR FUTURE LOOK-UPS
*GET LOC(NEWB) AND N(NEWB)
 Y20   CLA     **,1            (**=TABL1)
       STA     LOCNWB
       ARS     18
       STA     NNEWB
*GET P(NEWB,NXT(NEWB,T)) AND THE NEW INSTRUCTION TO BE REPLACED
 Y20A  CLA     **,1            (**=TABL3)
       STA     PNEWBT
       ADD     LOCNWB
       SUB     K1
       PAC     0,2             -(ADDRESS OF INSTR TO DELETE)
       CLA     0,2
       STO     NEWINS
*CHECK,IS THIS FIRST REPLACEMENT IN THIS BLOCK
       CLA     PNEWBT
       CAS     NNEWB
       HPR     *               IMPOSSIBLE
       TRA     Y22             YES
*IF NOT, MOVE TRA INSTRUCTION UP ONE NOTCH, AND RESTORE OLD INSTRUCTION
       CLA     1,2                                MOVE UP
       STO     0,2                                TRA
 Y20B  CLA     **,1            (**=TABL4)         RESTORE OLD
       STO     1,2                                INSTRUCTION
*THEN SAVE NEW INSTRUCTION AND CHECK IF TRA IS NOW AT TOP OF BLOCK
Y21    CLA     NEWINS
 Y21A  STO     **,1            (**=TABL4)
       CLA     K1              (K1=1)
       CAS     PNEWBT
       HPR     *               IMPOSSIBLE
       TRA     Y23             YES, SINCE P(NEWB,NXT(NEWB,T)) = 1
*IF NOT AT TOP SET NEW X INDEX TO BE CHECKED(FROM BLOCK NEWB) INTO TABL2
       CLA     -1,2
 Y21B  STA     **,1            (**=TABL2)
*REDUCE INSTRUCTION INDEX FOR BLOCK NEWB IN TABL3
 Y21C  CLA     **,1            (**=TABL3)
       SUB     K1
 Y21D  STO     **,1            (**=TABL3)
       TRA     Y24             ON TO COMPUTE CORRELATION
*FOR FIRST REPLACEMENT FORM TRA LOC(NEWB)+N(NEWB) AND
*INSERT IN LOC(NEWB)+N(NEWB)-1
 Y22   CLA     LOCNWB
       ADD     NNEWB
       STA     K22             K22=TRA **
       CLA     K22
       STO     0,2
       TRA     Y21             BACK TO SAVE INSTRUCTION
*IF TRA AT TOP OF BLOCK ADD 5 TO ITS ADDRESS AND SET
*TABL2 SO THIS BLOCK IS INVISIBLE TO FUTURE SCANS
 Y23   CLA     0,2
       ADD     K9              K9=5
       STA     0,2
       CLA     K20             K20=77777
 Y23A  STO     **,1            (**=TABL2)
       TRA     Y21C            BACK TO UPDATE TABL3 FOR LAST TIME
*NOW COMPUTE CORRELATION
 Y24   CLS     LAG             -T                 SET FOR
       PAX     0,2             T IN IR2           STORAGE
       ADD     T23             Y-N-T
       PAC     0,1             -(Y-N-T) TO IR1
       TPL     *+2
       PAX     0,1             (FOR CASE Y-N-T NEGATIVE)
       STZ     SUM
 Y25   TSX     **,4            (**=ENTRY POINT TO OBJECT PROGRAM)
       CLA     SUM
*(NEXT INSTRUCTION  =  ADD CORZER,2 FOR FASCR1, FASEP1)
       NOP
 Y26   STO     **,2            (**=CORZER)
       TOV     Y96             ERROR EXIT
       CLA     NXTXI
       ADD     K1
```

```
       STO    NXTXI
       TXI    *+1,2,1
       SXA    LAG,2
 Y27   TXL    Y9,2,**        (**=POSMAX)
*WHEN DONE, RÉSTORE INSTRUCTIONS WHICH HAVE BEEN REPLACED BY TRA
*(LOOP TAKES ABOUT 24*(S+1) HI SPEED INSTRUCTIONS)
*(NEXT INSTRUCTION  =  TRA Y35  FOR FASEPC OR FASEP1)
       AXT    0,1            I=0
 Y28   CLA    **,1           (**=TABL4)
       TZE    Y32            ZERO MEANS NO REPLACEMENT IN BLOCK
 Y29   CLA    **,1           (**=TABL1)
 Y30   ADD    **,1           (**=TABL3)
       PAC    0,2            -(LOC(I)+P(I,LAST SUBS)-1)
 Y31   CLA    **,1           (**=TABL4)
       STO    0,2            RESTORE INSTRUCTION
 Y32   TXI    *+1,1,1
 Y33   TXL    Y28,1,**       (**=S)
*ARE NEGATIVE LAGS WANTED
 Y35   CLA    T20
       CAS    K7
       TRA    Y36            YES
       TRA    Y36            YES
       TRA    Y95C           NO-GO EXIT
*MAKE INITIAL SETTINGS FOR NEGATIVE LAGS
 Y36   CLA    T11
       STO    NXTXI
       CLA    K1
       STO    LAG
       CLA    T3             S
       ARS    19             S/2
       SSM                   -S/2
       ADD    T14            PLUS TABL2
       PAC    0,1
       SXA    LOCOLX,1
*SET UP TABL2, TABL3, TABL4 FOR NEG LAGS
*(LOOP TAKES 12*(S+1) HI SPEED INSTRUCTIONS)
  Y37  AXT    0,1            I = 0
  Y38  CLA    **,1           (** = TABL1)
       PDX    0,2            -N(I) TO XR2
       TXH    Y39,2,0
       CLM
       TRA    Y40
  Y39  PAC    0,2
       CAL    0,2
       ANA    K20            K20 = 77777
 Y40   SLW    **,1      **=TABL2
  Y41  STZ    **,1           (** = TABL4)
       CLA    K1
  Y42  STO    **,1           (** = TABL3)
       TXI    *+1,1,1
  Y43  TXL    Y38,1,**       (** = S)
*NOW LOOP THROUGH CORRELATIONS
*THIS LOOP SCANS TABL2 AS DOES LOOP AT Y9, BUT
*DOES IT FOR NEGATIVE LAGS (INEQUALITIES WORK OPPOSITELY)
*(LOOP Y49 TO Y84-1 TAKES ABOUT NEGMAX*(61+(3 TO 3*(S+1))) HI SPEED
*   INSTRUCTIONS EXCLUSIVE OF OBJECT PROGRAM)
*(NEXT INSTRUCTION  =  TRA Y80  FOR FASEPC OR FASEP1)
  Y49  LXA    LOCOLX,1
       LXA    LOCOLX,2
       CLA    NXTXI
 Y50   TXL    Y51,1,**       (**=-(TABL2+1))
       CAS    0,1
       TXI    Y51,1,-1      NO GOOD
       TRA    Y53           GOT IT
       HPR    *             IMPOSSIBLE
 Y51   TXH    Y50,2,**       **= -(TABL2-S)
       CAS    0,2
       TXI    Y50,2,1        NO GOOD
       TRA    Y52            GOT IT
       HPR    *              IMPOSSIBLE
  Y52  SXA    LOCOLX,2
       TRA    Y54
  Y53  SXA    LOCOLX,1
  Y54  LAC    LOCOLX,1
       PXA    0,1
```

```
        SUB     T14            T14 = PZE TABL2
        SSP
        PAX     0,1            NEWB IN XR1
*GET LOC(NEWB) AND N(NEWB)
  Y60   CLA     **,1           (** = TABL1)
        STA     LOCNWB
        PAC     0,4            -LOC(NEWB) TO XR4
        ARS     18
        STA     NNEWB
*GET P(NEWB,NXT(NEWB,T))
  Y61   CLA     **,1           (** = TABL3)
        STA     PNEWBT
        ADD     LOCNWB
        SUB     K1
        PAC     0,2            -(LOC(NEWB) + P(NEWB(T) - 1) TO XR2
*IS THIS THE FIRST REPLACEMENT OF BLOCK
        CLA     K1
        CAS     PNEWBT
        HPR     *              IMPOSSIBLE
        TRA     Y75            YES
*IF NOT, ADD 1 TO ADDRESS OF TRA INSTRUCTION
  Y62   CLA     0,4
        ADD     K1
        STO     0,4
*WAS OLD MODIFIED INSTRUCTION CLA (0500) OR CLS (0502)
  Y63   CLA     0,2
        CAS     K23            K23 = 0501
        TRA     Y65            IT IS CLS
        HPR     *
*IF IT WAS CLA RECONVERT TO ADD
  Y64   ANA     K20
        ADD     K25            K25 = ADD 0,1
        TRA     Y66
*IF IT WAS CLS RECONVERT TO SUB
  Y65   ANA     K20
        ADD     K26            K26 = SUB 0,1
  Y66   STO     0,2
*IS THIS LAST INSTRUCTION IN BLOCK TO BE DELETED
  Y67   CLA     PNEWBT
        CAS     NNEWB
        HPR     *              IMPOSSIBLE
        TRA     Y77            YES
*IF NOT, CONVERT THE ADD (0400) OR SUB (0402) TO CLA OR CLS
  Y68   CLA     1,2
        CAS     K24            K24 = 0401
        TRA     Y70            IT WAS SUB
        HPR     *              IMPOSSIBLE
*IF IT WAS ADD, CONVERT TO CLA
  Y69   ANA     K20
        ADD     K27            K27 = CLA 0,1
        TRA     Y71            GO ADJUST TABL2
*IF IT WAS SUB, CONVERT TO CLS
  Y70   ANA     K20
        ADD     K28            K28 = CLS 0,1
  Y71   STO     1,2
*SET NEW X INDEX
  Y72   STA     **,1           (** = TABL2)
*INCREASE TRIAL INSTRUCTION INDEX BY 1 IN TABL3
  Y73   CLA     **,1           (** = TABL3)
        ADD     K1
  Y74   STO     **,1           (** = TABL3)
        TRA     Y80
*IF FIRST REPLACEMENT, SAVE INSTRUCTION, FORM AND INSERT
*TRA LOC(NEWB) + 1  INTO LOC(NEWB)
  Y75   CLA     0,4
  Y76   STO     **,1           (** = TABL4)
        CLA     LOCNWB
        ADD     K1
        ADD     K21            K21 = TRA 0
        STO     0,4
        TRA     Y67       BACK TO CHECK IF IT IS LAST INSTRUCTION ALSO
*IF LAST REPLACEMENT, ADD 5 TO TRA INSTR ADDR AND SET
*TABL2 TO MAKE THIS BLOCK INVISIBLE TO FURTHER SCANNING
  Y77   CLA     0,4
        ADD     K9             K9 = 5
```

```
       STO     0,4
   Y78 STZ     **,1           (** = TABL2)
       TRA     Y73            BACK FOR ADJUSTMENT OF TABL3
*NOW COMPUTE CORRELATICN
   Y80 CLA     LAG            T          SET FOR
       PAC     0,2            -T IN IR2   STORAGE
       ADD     T23            Y-N+T
       PAC     0,1        -(Y-N+T) TO XR1   (Y-N+T  IS ALWAYS POSITIVE)
       STZ     SUM
   Y81 TSX     **,4           (** = ENTRY POINT TO OBJECT PROGRAM)
       CLA     SUM
*(NEXT INSTRUCTION = ADD CORZER,2  FOR FASCR1, FASEP1)
       NOP
   Y82 STO     **,2           (** = CORZER)
       TOV     Y96
       CLA     NXTXI
       SUB     K1
       STO     NXTXI
       CLA     LAG
       ADD     K1
       STO     LAG
       CAS     T26
       TRA     Y84            GO WINDUP
       TRA     Y49            GO BACK
       TRA     Y49            GO BACK
*WHEN DONE, RESTORE INSTRUCTIONS WHICH HAVE BEEN REPLACED
*BY TRA AND RECONVERT TO ADD OR SUB ANY OTHER
*INSTRUCTIONS WHICH HAVE BEEN CHANGED AND NOT RECHANGED
*(LOOP TAKES ABOUT 24*(S+1) HI SPEED INSTRUCTIONS)
*(NEXT INSTRUCTION =  TRA Y95C  FOR FASEPC OR FASEP1)
   Y84 AXT     0,1
   Y85 CLA     **,1           (** = TABL4)
       TZE     Y93            IGNORE IF ZERO
   Y86 CLA     **,1           (** = TABL1)
       PAC     0,2            -(LOC(I)) TO IR2
       STA     T24            SAVE LOC(I)
       ARS     18             AND N(I)
       STA     T25
       CLA     0,2            GET TRA INSTRUCTION FROM BLOCK I
   Y87 LDQ     **,1           (** = TABL4)      REPLACE BY
       STQ     0,2                              OLD INSTRUCTION
       ANA     K20            EXTRACT ADDRESS OF TRA
       SUB     T24            ADDRESS MINUS LOC(I)
       CAS     T25            COMPARE WITH N(I)
       TRA     Y93            BIGGER, SO MUST BE TRA LOC(I+1)
       HPR     *              IMPOSSIBLE
*GET INSTRUCTION TO BE RECONVERTED. IS IT CLA OR CLS
       ADD     T24
   Y89 STA     Y90
       STA     Y92            SET FOR RESTORAGE
  Y90  CLA     **             (**=ADDRESS OF INSTR BEING RECONVERTED)
       CAS     K23
       TRA     Y91            IT IS CLS
       HPR     *
*IF CLA, CHANGE TO ADD
       ANA     K20
       ADD     K25
       TRA     Y92
*IF CLS, CHANGE TO SUB
   Y91 ANA     K20
       ADD     K26
*RESTORE
   Y92 STO     **             (** = ADDRESS OF INSTR BEING RECONVERTED)
*CHECK END OF LOOP
   Y93 TXI     *+1,1,1
   Y94 TXL     Y85,1,**       (** = S)
       TRA     Y95C           GO EXIT
*CLEAR OUTPUT WHEN PROCOR SIGNALS ALL X(I) = 0, AND EXIT
*(NOT USED BY FASCR1, FASEP1)
  Y95  AXT     1,4
  Y95A STZ     **,4      **=CORZER+MAGNITUDE(NEGMAX)+1
       TXI     *+1,4,1
  Y95B TXL     Y95A,4,**      **=POSMAX+MAGNITUDE(NEGMAX)+1
*GENERAL EXIT FOR ALL OK
  Y95C STZ     T30
```

```
        STZ     T27
*RESTORE XRS AND SET ERROR
 Y96    LXD     FASCOR-4,1
        LXD     FASCOR-3,2
        LXD     FASCOR-2,4
        CLA     T30
        STO*    5,4
*NOW RESTORE FASCOR PROGRAM AND EXIT
        CLA     K50A
        STO     Y2
        STO     Y3
        CLA     K51A
        STO     Y9
        STO     Y49
        CLA     K52A
        STO     Y28-1
        STO     Y84
        CLA     K64A
        STO     Y4
        CLA     K62A
        STO     Y7-1
        STO     Y26-1
        STO     Y82-1
        TRA     6,4
*
*
*FASEPC, FASCR1, FASEP1  SUBSTITUTION CONSTANTS
*(SERIES A FOR FASCOR, SERIES B FOR FASEPC)
 K50A   TRA     Y96             FOR Y2 AND Y3
 K50B   NOP                     FOR Y2 AND Y3
 K51A   LXA     LOCOLX,1        FOR Y9 AND Y49
 K51B   TRA     Y24             FOR Y9
 K52A   AXT     0,1             FOR Y28-1 AND Y84
 K52B   TRA     Y35             FOR Y28-1
 K53B   TRA     Y80             FOR Y49
 K54B   TRA     Y95C            FOR Y84
 K60    ADD     **              **=CORZER
 K61    PZE     0,2,0
 K62A   NOP                     FOR Y7-1,Y26-1,Y82-1
 K63    TNZ     Y95C            FOR Y4
 K64A   TNZ     Y95             FOR Y4
*
*MODIFY FASCOR TO BYPASS SUBSTITUTION LOGIC AND TO ELIMINATE
*   MAXIMUM LAG LIMIT CHECK
FASEPC CLA      K50B
        STO     Y2
        STO     Y3
        CLA     K51B
        STO     Y9
        CLA     K52B
        STO     Y28-1
        CLA     K53B
        STO     Y49
        CLA     K54B
        STO     Y84
        TRA     FASCOR
*
*MODIFY FASCOR SO CORRELATIONS ADD TO CORZER(I)
FASCR1 LDQ      K1              K1 IS POSITIVE
        CLA     4,4
        STA     K60             K60= ADD **
        CLA     K60
        STO     Y7-1
        ADD     K61             K61= PZE 0,2,0
        STO     Y26-1
        STO     Y82-1
        CLA     K63
        STO     Y4
        TQP     FASCOR
        TRA     FASEPC
*
*SAME AS FASCR1 BUT FOR FASEPC
FASEP1 LDQ      KMSK2           KMSK2 IS NEGATIVE
        TRA     FASCR1+1
        END
```

```
*     PSQRT (SUBROUTINE)              10/5/64   LAST CARD IN DECK IS NO. 0090
*     LABEL
CPSQRT
      SUBROUTINE PSQRT (N,C,M,A)
C
C                    ----ABSTRACT----
C
C TITLE - PSQRT
C         FIND THE POWER SERIES SQUARE ROOT OF A POLYNOMIAL
C
C              GIVEN THE POLYNOMIAL
C
C                                        2       3          (N-1)
C              P(X)= C(1)+C(2)*X+C(3)*X +C(4)*X +...+C(N)*X
C
C                    WHERE C(1) IS GREATER THAN ZERO
C
C              FIND THE FIRST M COEFFICIENTS OF THE SQUARE ROOT POWER
C              SERIES.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - IBM  709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 155 REGISTERS
C SPEED     - 3 + M + .06*M*M    MILLISEC ON IBM 709
C AUTHOR    - J. CLAERBOUT
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C         AND FORTRAN SYSTEM ROUTINES - SQRT
C
C FORTRAN USAGE
C     CALL PSQRT (N,C,M,A)
C
C INPUTS
C
C    N          THE NUMBER OF COEFFICIENTS IN THE POLYNOMIAL TO BE ROOTED
C                MUST BE GREATER THAN OR =4(THIS IS NO SEVERE RESTRICTION
C                HOWEVER SINCE ANY OF C(2),C(3),AND/OR C(4) MAY = ZERO.)
C
C    C(I)       I=1,N   IS THE VECTOR OF COEFFICIENTS OF THE POLYNOMIAL
C                WHICH IS TO BE ROOTED.  C(1) MUST EXCEED ZERO.
C
C    M          THE NUMBER OF COEFFICIENTS DESIRED IN THE SQUARE ROOT
C                POWER SERIES.
C
C OUTPUTS
C
C    A(I)       I=1,M  THE VECTOR OF THE FIRST M COEFFICIENTS IN THE
C                      SQUARE ROOT POWER SERIES.
C
C EXAMPLES
C
C 1. INPUTS  - N=4   M=6
C              C(1...4) =1.,-4.,4.,0.  (NOTICE C(4) MUST BE DEFINED
C                                       EVEN THOUGH IT IS ZERO.)
C    OUTPUTS - A(1...6)=1.,-2.,0.,0.,0.,0.
C                       (THE POWER SERIES DEGENERATES TO POLYNOMIAL)
C
C 2. INPUTS  - N=4   M=15
C              C(1...4)=1.,2.,0.,0.
C    OUTPUTS - A(1...15)=1.,1.,-.5,.5,-.625,.875,-1.31,2.06,-3.35,5.58,
C                        -9.49,16.4,-28.7,50.7,-90.
C                        (IN THIS EXAMPLE THE RADIUS OF CONVERGENCE OF
C                         THE POWER SERIES IS .5, THEREFORE THE COEFS.
C                         TEND TO INCREASE. OVERFLOW WOULD OCCUR SOME-
C                         WHERE AROUND THE 120 TH COEFFICIENT.)
C
      DIMENSION A(100),C(100)
      A(1)=SQRTF(C(1))
      TA=2.*A(1)
      A(2)=C(2)/TA
      A(3)=(C(3)-A(2)*A(2))/TA
      DO 100  I=4,M
      IF(I-N)  20,20,10
```

****
*   PSQRT   *
****
(PAGE 2)

```
   10   PA=0.
        GO TO 30
   20   PA=C(I)
   30   CONTINUE
        PS=0.
        IH=I/2
        DO 40 J=2,IH
        K=I-J
   40   PS=PS+A(J)*A(K+1)
        PA=PA-2.*PS
        IF(2*IH-I)  50,60,50
   50   PA=PA-A(IH+1)*A(IH+1)
   60   A(I)=PA/TA
  100   CONTINUE
        RETURN
        END
```

```
*     PWMLIV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0141
*     LABEL
CPWMLIV
      SUBROUTINE PWMLIV(JOB,ITAPE,MLIV,LMLIV,IANS)
C
C                       ----ABSTRACT----
C
C TITLE - PWMLIV
C        PRINT OR WRITE OUTPUT TAPE A MACHINE LANGUAGE INTEGER VECTOR
C
C              PWMLIV IS AN ELEMENTARY OUTPUT PROGRAM FOR MLI VECTORS,
C              PRINTING 1 TO 10 WORDS PER LINE AND ALWAYS 12 CHARACTERS
C              PER WORD.  OUTPUT IS EITHER ON-LINE OR ON LOGICAL TAPE
C              AS SPECIFIED BY AN ARGUMENT.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090, MAIN FRAME PLUS 1 TAPE AND/OR ON-LINE PRINTER
C STORAGE   - 300 REGISTERS
C SPEED     -
C AUTHOR    - S.M. SIMPSON JR, JULY 1961
C
C                       ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - ML12A6
C        AND FORTRAN SYSTEM ROUTINES - (FIL), (SPH), (STH)
C
C FORTRAN USAGE
C     CALL PWMLIV(JOB,ITAPE,MLIV,LMLIV,IANS)
C
C INPUTS
C
C    JOB        = +N OR -N WHERE N = DESIRED NO. OF WORDS/LINE
C               AND + MEANS OFF-LINE OUTPUT, - MEANS ON-LINE OUTPUT.
C               AND 1 LSTHN= N LSTHN= 10
C
C    ITAPE      IS NOT USED FOR JOB NEGATIVE
C               IS LOGICAL NO. OF DESIRED OUTPUT TAPE FOR JOB POSITIVE
C                 IN THIS CASE ITAPE MUST HAVE VALUE 1...20 INCLUSIVE
C
C    MLIV(I)    I=1...LMLIV IS THE MLI VECTOR.
C
C    LMLIV      GRTHN= 1
C
C OUTPUTS
C
C               PRINCIPAL OUTPUT IS PRINTED COPY
C
C    IANS       = 0 JOB DONE
C               =-1 ILLEGAL JOB NO.
C               =-2 ILLEGAL ITAPE
C               =-4 ILLEGAL LMLIV
C
C EXAMPLES
C
C 1. INPUTS  - JOB=1  MLIV(1...15) = OCT 377777777777,777777777777,0,1,
C               2,3,4,5,6,7,10,11,12,400000000013,400000000000   LMLIV=15
C               ITAPE = 2
C    OUTPUTS - IANS=0  AND WE SHOULD GET THE FOLLOWING LIST OFF-LINE
C               IN FORMAT(2A6)    34359738367,-34359738367,0,1,2,3,4,5,6,
C               7,8,9,10,-11,-0
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=-1
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT OUTPUT SHOULD BE ON-LINE.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=10
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT FORMAT SHOULD BE 20A6),
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=-10
C    OUTPUTS - SAME AS EXAMPLE 3. EXCEPT OUTPUT SHOULD BE ON-LINE.
C
C 5. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LMLIV=1
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT LIST HAS ONLY FIRST TERM.
C
C 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=5
```

```
C     OUTPUTS - SAME AS EXAMPLE 1. EXCEPT FORMAT(10A6)
C
C 7. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=7
C     OUTPUTS - SAME AS EXAMPLE 1. EXCEPT FORMAT(14A6)
C
C 8. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=0
C     OUTPUTS - IANS=-1
C
C 9. INPUTS  - SAME AS EXAMPLE 1. EXCEPT JOB=11
C     OUTPUTS - IANS=-1
C
C10. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LMLIV=0
C     OUTPUTS - IANS=-4
C
C11. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ITAPE = 0
C     OUTPUTS - IANS=-2
C
      DIMENSION MLIV(2) , BUF(20)
C CHECK INPUTS JOB, ITAPE AND LMLIV
      IANS=-1
      IF (JOB)  10,9999,20
   10 IF (10+JOB)  9999,30,30
   20 IF (JOB-10)  30,30,9999
 30   IANS = -2
      IF (ITAPE-1)  9999,40,40
 40   IF (ITAPE-20)  50,50,9999
 50   IANS = -4
      IF (LMLIV)  9999,9999,100
C SET UP NRGLNS,NWREG,NWLST,NXMLI
  100 NWREG=JOB
      IF (JOB)  110,120,120
  110 NWREG=-JOB
  120 NRGLNS=LMLIV/NWREG
      NWLST=LMLIV-NRGLNS*NWREG
      NXMLI=1
      NRREG=2*NWREG
      IANS=0
      GO TO 200
C TREAT REGULAR LINES IF THERE ARE ANY
  200 IF (NRGLNS)  300,300,220
  220 DO 280  I=1,NRGLNS
      DO 240  J=1,NWREG
      K=2*J-1
      CALL   MLI2A6(MLIV(NXMLI),BUF(K),NCRS)
  240 NXMLI=NXMLI+1
      IF (JOB)  270,270,260
  260 WRITE OUTPUT TAPE ITAPE,700,(BUF(J),J=1,NRREG)
      GO TO 280
  270 PRINT 700,(BUF(J),J=1,NRREG)
  280 CONTINUE
      GO TO 300
C WORK ON LAST LINE OF OUTPUT IF ANY
  300 IF (NWLST)  9999,9999,320
  320 DO 330 I=1,NWLST
      J=2*I-1
      CALL   MLI2A6(MLIV(NXMLI),BUF(J),NCRS)
  330 NXMLI=NXMLI+1
      NRLST=2*NWLST
      IF (JOB)  350,350,340
  340 WRITE OUTPUT TAPE ITAPE,700,(BUF(J),J=1,NRLST)
      GO TO 9999
 350  PRINT 700,(BUF(J),J=1,NRLST)
      GO TO 9999
C ONLY FORMAT STATEMENT
  700 FORMAT(1H ,20A6)
C EXIT
 9999 RETURN
      END
```

```
*     QACORR (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0183
*     LABEL
CQACORR
      SUBROUTINE QACORR (X,LX,MXACC,MXLAG,SPACE,ACOR,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - QACORR
C        FAST AUTOCORRELATIONS FOR LONG, LIMITED ACCURACY SERIES
C
C              QACORR COMPUTES THE UNNORMALIZED AUTOCORRELATION
C              FUNCTION, AC(L), OF A LIMITED ACCURACY SERIES, X(I) OF
C              LENGTH LX, ACCORDING TO THE TRANSIENT FORMULA
C
C                              1    LX-L
C                   AC(L)  =  --- * SUM  ( X(I)*X(I+L) )
C                              LX   I=1
C
C                         FOR L = 0,1,...,MXLAG
C
C                              WHERE MXLAG AND LX ARE INPUT PARAMETERS
C
C              SPEED DEPENDS ON LX AND ACCURACY OF X.  FOR VERY LONG
C              SERIES A COMPLETE AUTOCORRELATION (MXLAG = LX-1) CAN BE
C              COMPUTED IN SLIGHTLY MORE THAN (LX)SQUARED MACHINE CYCLES.
C              QACORR OBTAINS THIS SPEED PRIMARILY BY CONVERTING X(I) TO
C              AN INTEGER SEQUENCE IX(I) WHOSE MAGNITUDES HAVE UPPER
C              LIMIT AS SPECIFIED BY AN INPUT PARAMETER MXACC, AND THEN
C              REGROUPS THE ABOVE EQUATION (FOR EACH LAG) SO AS TO
C              PERFORM LX-L ADDITIONS PLUS MXACC (OR FEWER) MULTIPLI-
C              CATIONS (RATHER THAN LX-L ADDITIONS PLUS LX-L MULTIPLI-
C              CATIONS).  THE RESULTS ARE THEN RECONVERTED TO FLOATING
C              POINT FORM WITH CORRECT SCALE. (SEE SUBROUTINE PROCOR,
C              FASCOR FOR LOGIC DETAILS.) IX(I) IS ALSO REFLOATED.
C
C              USER MUST PROVIDE QACORR WITH A BLOCK OF TEMPORARY
C              REGISTERS OF LENGTH LX + 10*(MXACC+1) + 1 .
C
C              X(I) IS LEFT SLIGHTLY MODIFIED BY THE FIXING, REFLOATING
C              PROCESS.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT - 709, 7090 (MAIN FRAME ONLY)
C STORAGE    - 207 REGISTERS
C SPEED      - FOR LONG SERIES QACORR TAKES ABOUT
C              (MXLAG+1)*(2*LX-MXLAG+20*MXACC) MACHINE CYCLES
C AUTHOR     - S. M. SIMPSON JR, 10/5/62
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   FXDATA, PROCOR, FASCOR, FLDATA
C        AND FORTRAN SYSTEM ROUTINES -   NONE
C
C FORTRAN USAGE
C     CALL QACORR(X,LX,MXACC,MXLAG,SPACE,ACOR,IANS)
C
C INPUTS
C
C    X(I)       I=1,2,...LX  IS A FLOATING POINT VECTOR
C
C    LX         MUST EXCEED ZERO AND BE LSTHN= 10000
C
C    MXACC      DEFINES ACCURACY OF X(I). X(I) WILL BE FIXED TO HAVE
C                  VALUES LYING BETWEEN -MXACC AND +MXACC INCLUSIVE.
C               MUST LIE BETWEEN 1 AND 1000 INCLUSIVE. (SMALLER VALUES
C                  YIELD HIGHER SPEEDS, AND REQUIRE FEWER TEMPORARIES.)
C
C    MXLAG      IS HIGHEST LAG NO. DESIRED IN AUTOCORRELATION
C               MUST BE NON-NEGATIVE
C
C    SPACE(I)  I=1,...,LSPACE MUST BE AVAILABLE AS TEMPORARIES, WHERE
C                         LSPACE = LX + 10*(MXACC+1) +1
C
C OUTPUTS
C
```

```
C    X(I)       I=1,2,....,LX  CONTAINS THE ROUNDED SERIES XX(I)
C                          XX(I) = FLOATF(IX(I))/SCALE
C                     WHERE
C                          IX(I) = XFIXF(X(I)*SCALE)
C                     WITH
C                         SCALE = FLOATF(MXACC)/XMAX
C                           XMAX = LARGEST X MAGNITUDE
C                    (NOTE- XFIXF IN ABOVE EXPRESSION IMPLIES ROUNDING
C                          TO NEAREST INTEGER, NOT TRUNCATION)
C               X(I) WILL BE LEFT = 0.0 IF XMAX = 0.0
C
C    ACOR(I)    I=1,2,....,(MXLAG+1)  WILL CONTAIN AC(L), L=0,1,....,MXLAG
C                  COMPUTED ON THE ROUNCED SERIES XX(I)
C
C                              1     LX-L
C                    AC(L)  =   --- * SUM  ( XX(I)*XX(I+L) )
C                              LX    I=1
C
C                            = 0.0  FOR ALL L GRTHN LX-1 IF ANY
C
C               ACOR(I) WILL BE IDENTICALLY 0.0 WHENEVER X(I) IS
C
C    IANS       = 0   IF NO TROUBLE ARISES
C               = -2  IF LX IS ILLEGAL
C               = -3  IF MXACC IS ILLEGAL
C               = -4  IF MXLAG IS ILLEGAL
C               = -98 IF UNEXPLAINED ERROR RETURN FROM PROCOR OCCURS
C               = -99 IF UNEXPLAINED ERROR RETURN FROM FASCOR OCCURS
C
C EXAMPLES      THE FIRST 3 EXAMPLES ARE CHOSEN SO THAT THE ROUNDOFF
C               EFFECT IS NOT PRESENT
C
C               CALL QACORR (X,LX,MXACC,MXLAG,SPACE,ACOR,IANS) IS THE
C               ASSUMED USAGE IN ALL EXAMPLES
C
C 1. COMPLETE AUTOCORRELATION
C    INPUTS  - X(1...5) = 10.,20.,10.,10.,5. , LX=5, MXACC=20, MXLAG=4
C    OUTPUTS - X(1...5) = SAME AS INPUTS   IANS=0
C              ACOR(1...5) = 145.,110.,70.,40.,10.
C
C 2. PARTIAL AUTOCORRELATION
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT MXLAG = 2
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT ACOR(4..5) NOT DEFINED
C
C 3. AUTOCORRELATION BEYOND END OF SERIES
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT MXLAG=7
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT
C              ACOR(1...8) = 145.,110.,70.,40.,10.,0.,0.,0.
C                (I.E. TERMINAL ZEROES SUPPLIED)
C
C 4. PARTIAL AUTOCORRELATION SHOWING ROUNDOFF
C    INPUTS  - X(1...3) = 23.8,148.0,20.3  LX=3  MXACC=100  MXLAG=0
C    OUTPUTS - X(1...3) = 23.68,148.0,20.72  IANS=0
C              ACOR(1) = 7631.354
C
C 5. THE NEXT 3 EXAMPLES SHOW ERROR CONDITIONS
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX=0
C    OUTPUTS - IANS =-2  (ILLEGAL LX)
C
C 6. INPUTS  - SAME AS EXAMPLE 1. EXCEPT MXACC=5000
C    OUTPUTS - IANS =-3  (ILLEGAL MXACC)
C
C 7. INPUTS  - SAME AS EXAMPLE 1. EXCEPT MXLAG=-2
C    OUTPUTS - IANS =-4  (ILLEGAL MXLAG)
C
C 8. (SPECIAL TEST FOR BYPASS IN CASE ALL X(I)=0.)
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT X(1...5)=0.,0.,...
C    OUTPUTS - SAME AS EXAMPLE 1. EXCEPT ACOR(1...5)=0.,0.,...
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION X(2),SPACE(2),ACOR(2)
C CHECK INPUTS
      IANS=-2
      IF (LX) 9999,9999,5
```

```
 5    IF (LX-10000) 10,10,9999
 10   IANS=-3
      IF (MXACC) 9999,9999,20
 20   IF (MXACC-1000) 30,30,9999
 30   IANS=-4
      IF (MXLAG) 9999,40,40
C CLEAR OUTPUT AREA
 40   NNLAGS=MXLAG+1
      DO 50 I=1,NNLAGS
 50   ACOR(I)=0.0
C SET NO. LAGS = MIN(MXLAG,LX-1)
      NLAGS=MXLAG
      IF (MXLAG-LX+1) 70,70,60
 60   NLAGS=LX-1
C SET SPACE CONSTANT FOR PROCOR AND FIX X. EXIT IF X=ZERO VECTOR.
 70   LSPACE=LX+10*(MXACC+1)+1
      CALL FXDATA(LX,X,MXACC,SCALE)
      IANS=0
      IF (SCALE) 9999,9999,80
C THEN COMPUTE AUTOCORRELATIONS
 80   IANS=-98
      CALL PROCOR(X,LX,MXACC,SPACE(LSPACE),SPACE(1),ANSR)
      IF (ANSR) 900,100,900
 100  IANS=-99
      CALL FASCOR(X,0,NLAGS,ACOR,ANSR)
      IF (ANSR) 900,120,900
C FLOAT AND SCALE ACOR
 120  IANS=0
      SCSQ=SCALE*SCALE*FLOATF(LX)
      CALL FLDATA(NNLAGS,ACOR,SCSQ)
C REFLOAT X SERIES
 900  CALL FLDATA(LX,X,SCALE)
 9999 RETURN
      END
```

```
*     QCNVLV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0293
*     LABEL
CQCNVLV
      SUBROUTINE QCNVLV (XX,LXX,YY,LYY,MXACC,LCC,SPACE,CC,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - QCNVLV
C
C        FAST CONVOLUTIONS FOR LONG, LIMITED ACCURACY SERIES
C
C              QCNVLV COMPUTES THE CONVOLUTION, C(J), OF TWO LIMITED
C              ACCURACY SERIES, X(I) I=0...LX  AND Y(I) I=0...LY,
C              ACCORDING TO THE FORMULA
C
C                                   LX
C                         C(J)  =   SUM ( X(I)*Y(J-I) )
C                                   I=0
C
C                        FOR J= 0,1,...,LC
C
C                         WHERE Y(K) IS ASSUMED = 0.0 WHENEVER K IS
C                                    OUTSIDE OF THE RANGE 0 TO LY
C                               LX,LY,AND LC  ARE INPUT PARAMETERS
C
C              TO OBTAIN HIGH SPEED THE X AND Y SERIES ARE CONVERTED TO
C              INTEGER SEQUENCES WHOSE MAGNITUDES HAVE UPPER LIMIT AS
C              SPECIFIED BY AN INPUT PARAMETER MXACC AND THEN REGROUPS
C              THE ABOVE EQUATION SO AS TO SUBSTITUTE ADDITIONS FOR
C              MULTIPLICATIONS (SEE PROCOR-FASCOR-FASEPC FOR LOGIC).
C              THE RESULTS ARE THEN RECONVERTED TO FLOATING POINT FORM
C              WITH CORRECT SCALE.  THE INTEGER SEQUENCES FOR X AND Y
C              ARE ALSO REFLOATED.
C
C              USER MUST PROVIDE QCNVLV WITH A BLOCK OF TEMPORARY
C              REGISTERS OF LENGTH  LMIN + 10*(MXACC+1) + 1
C                WHERE LMIN = MINIMUM(LX,LY) + 1
C
C              X(I) AND Y(I) ARE LEFT SOMEWHAT MODIFIED BY THE FIXING;
C              REFLOATING PROCESS.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 569 REGISTERS
C SPEED     - COMPLETE CONVOLUTIONS (LC=LX+LY) CAN BE COMPUTED IN ABOUT
C                2*(LX+1)*(LC+1) MACHINE CYCLES IF  LX MUCH LSTHN LY
C              1.2*(LX+1)*(LC+1) MACHINE CYCLES IF  LX ABOUT = LY
C              FOR LONG SERIES.
C AUTHOR    - S.M. SIMPSON, 10/18/62
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   FXDATA, PROCOR, FASCOR, FASEPC,
C                                       FLDATA
C        AND FORTRAN SYSTEM ROUTINES -  XLOC
C
C FORTRAN USAGE
C     CALL QCNVLV(XX,LXX,YY,LYY,MXACC,LCC,SPACE,CC,IANS)
C
C INPUTS
C
C     XX(I)      I=1,2,...,LXX  CONTAINS X(I) I=0,1,...,LX ; LX=LXX-1
C
C     LXX        MUST EXCEED ZERO
C
C     YY(I)      I=1,2,...,LYY  CONTAINS Y(I) I=0,1,...,LY ; LY=LYY-1
C                EQUIVALENCE(XX,YY) IS PERMITTED. HOWEVER NO PARTIAL
C                  CVERLAP OF XX AND YY IS ALLOWED.
C
C     LYY        MUST EXCEED ZERO
C
C     MXACC      SPECIFIES ACCURACY OF  X AND  Y.  THESE WILL BE FIXED SO
C                  AS TO HAVE INTEGER VALUES FROM -MXACC TO +MXACC DURING
C                  THE COMPUTATIONS.
```

```
C               MUST EXCEED ZERO AND BE LSTHN= 1000
C
C     LCC       IS NUMBER OF TERMS DESIRED IN OUTPUT CONVOLUTION
C               MUST EXCEED ZERO
C
C     SPACE(I)  I=1,2,...,LSPACE MUST BE AVAILABLE FOR TEMPORARY USE
C                   WHERE LSPACE = LMIN + 10*(MXACC+1) + 1
C                         LMIN = LXX OR LYY WHICHEVER IS SMALLER
C                         LMIN MUST NOT EXCEED 10000
C
C OUTPUTS
C
C     XX(I)     I=1...LXX WILL CONTAIN THE REFLOATED X SERIES
C
C     YY(I)     I=1...LYY WILL CONTAIN THE REFLOATED Y SERIES
C                 (IN THE FIXING PROCESS ROUNDING IS USED RATHER THAN
C                  TRUNCATION SO THE REFLOATED SERIES SHOULD NOT SHOW
C                  SYSTEMATIC DISCREPANCIES FROM THE ORIGINALS)
C
C     CC(I)     I=1...LCC IS THE CONVOLUTION C(J) J=0,1,...,LC , LC=LCC-1
C                 AS DEFINED IN THE ABSTRACT, AND AS COMPUTED ON THE
C                 FIXED VERSION OF THE X AND Y SERIES FOLLOWED BY
C                 FLOATING AND PROPER SCALING.
C
C     IANS      =  0  NORMALLY
C               = -2  IF LXX IS ILLEGAL
C               = -3  IF YY PARTIALLY OVERLAPS XX
C               = -4  IF LYY IS ILLEGAL OR IF LMIN EXCEEDS 10000
C               = -5  IF MXACC IS ILLEGAL
C               = -6  IF LCC IS ILLEGAL
C               =-99  IF UNEXPLAINED ERROR RETURN OCCURS FROM PROCOR
C                         FASCOR  OR  FASEPC
C
C EXAMPLES
C
C               THE FIRST 5 EXAMPLES ARE CHOSEN TO ELIMINATE THE ROUNDOFF
C               EFFECT.
C
C               INPUTS TO ALL EXAMPLES ARE ASSUMED THOSE OF EXAMPLE 1.
C               EXCEPT AS NOTED.
C               THE OUTPUT IANS IS EQUAL TO ZERO EXCEPT AS NOTED.
C
C 1. COMPLETE CONVOLUTION OF XX(1...3) WITH YY(1...7)
C     INPUTS  - XX(1...3)=10.,20.,20.  YY(1...7)=1.,10.,1.,1.,1.,1.,1.
C     USAGE   -       CALL QCNVLV (XX,3,YY,7,10,9,SPACE,CC,IANS)
C     CUTPUTS - XX(1...3) AND YY(1...7) SAME AS INPUT (NO ROUNDOFF
C               BECAUSE OF CHOICE OF XX YY)
C               CC(1...9)=10.,120.,230.,230.,50.,50.,50.,40.,20.
C               (IN THIS CASE LMIN=3, MXACC=10, SO SPACE(1) THRU SPACE
C               (114) IS USED AS TEMPORARY)
C
C 2. REVERSED ORDER CONVOLUTION
C     USAGE   -       CALL QCNVLV (YY,7,XX,3,10,9,SPACE,CC,IANS)
C     CUTPUTS - SAME AS EXAMPLE 1. (I.E. ORDER OF INPUTS XX AND YY
C                                  IMMATERIAL)
C
C 3. CONVOLUTION BEYOND END OF SERIES
C     USAGE   -       CALL QCNVLV (XX,3,YY,7,10,12,SPACE,CC,IANS)
C     OUTPUTS - SAME AS EXAMPLE 1. EXCEPT TERMINAL ZEROES ARE ADDED
C               TO CC, I.E.
C               CC(1..12)=10.,120.,230.,230.,50.,50.,50.,40.,20.,0.,0.,0.
C
C 4. PARTIAL CONVOLUTION
C     USAGE   -       CALL QCNVLV (XX,3,YY,7,10,3,SPACE,CC,IANS)
C     OUTPUTS - CC(1...3)=10.,120.,230.
C
C 5. COMPLETE AUTOCONVOLUTION
C     USAGE   -       CALL QCNVLV (XX,3,XX,3,10,5,SPACE,CC,IANS)
C     OUTPUTS - CC(1...5)=100.,400.,800.,800.,400.
C
C 6. PARTIAL AUTOCONVOLUTION SHOWING ROUNDOFF EFFECT WITH MXACC=100
C     INPUTS  - XX(1...3)=14.75,9.41,-20.0
C     USAGE   -       CALL QCNVLV (XX,3,XX,3,100,2,SPACE,CC,IANS)
C     CUTPUTS - XX(1...3)=14.80,9.40,-20.00
```

```
C              CC(1...2)=219.04,278.24
C
C 7. THE NEXT 5 EXAMPLES SHOW ERROR CONDITIONS
C    USAGE   -        CALL QCNVLV (XX,0,YY,2,10,4,SPACE,CC,IANS)
C    OUTPUTS - IANS=-2  (ILLEGAL LXX)
C
C 8. USAGE   -        CALL QCNVLV (XX,10,XX(3),3,10,9,SPACE,CC,IANS)
C              OR     CALL QCNVLV (YY(2),3,YY,5,10,4,SPACE,CC,IANS)
C    OUTPUTS - IANS=-3  (XX AND YY PARTIALLY OVERLAP)
C
C 9. USAGE   -        CALL QCNVLV (XX,10,YY,-3,10,4,SPACE,CC,IANS)
C              OR     CALL QCNVLV (XX,10100,YY,15000,10,SPACE)
C    OUTPUTS - IANS=-4  (ILLEGAL LYY OR LMIN)
C
C10. USAGE   -        CALL QCNVLV (XX,10,YY,3,1500,4,SPACE,CC,IANS)
C              OR     CALL QCNVLV (XX,10,YY,3,0,4,SPACE,CC,IANS)
C    OUTPUTS - IANS=-5  (ILLEGAL MXACC)
C
C11. USAGE   -        CALL QCNVLV (XX,10,YY,3,10,0,SPACE,CC,IANS)
C    OUTPUTS - IANS=-6
C
C12. SPECIAL CASE TEST - XX OR YY ALL ZERO
C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT XX(1...3)=0.,0.,0.
C    USAGE   -        CALL QCNVLV (XX,3,YY,7,10,9,SPACE,CC,IANS)
C              OR     CALL QCNVLV (YY,7,XX,3,10,9,SPACE,CC,IANS)
C    OUTPUTS - CC(1...9)=0.,0.,...
C
C13. SPECIAL CASE TEST - UNIT LENGTH XX
C    USAGE   -        CALL QCNVLV (XX,1,YY,7,10,7,SPACE,CC,IANS)
C    OUTPUTS - CC(1...7)=10.,100.,10.,10.,10.,10.,10.
C
C14. SPECIAL CASE TEST - LYY=LXX-1  (NO MIDDLE TERMS IN CONVOLUTION)
C    USAGE   -        CALL QCNVLV (XX,3,YY,2,10,4,SPACE,CC,IANS)
C    OUTPUTS - CC(1...4)=10.,120.,220.,200.
C
      DIMENSION XX(2),YY(2),CC(2),CM(2),SPACE(2)
      COMMON CM
C BRING IN LENGTHS, MXACC AND CHECK
      LX=LXX
      LY=LYY
      LC=LCC
      MAX=MXACC
      IANS=-2
      IF(LX)  9999,9999,30
 30   IANS=-4
      IF(LY)  9999,9999,40
 40   IANS=-5
      IF(MAX)  9999,9999,50
 50   IF(MAX-1000)  60,60,9999
 60   IANS=-6
      IF(LC)  9999,9999,80
C FIND LONGEST, SHORTEST SERIES AND INDICES W.R.T. COMMON
 80   LSHORT=XMINOF(LX,LY)
      LLONG=XMAXOF(LX,LY)
      LOCCM=XLOCF(CM)
      IX=LOCCM-XLOCF(XX)+1
      IY=LOCCM-XLOCF(YY)+1
      ISHORT=IX
      ILONG=IY
      IF(LX-LY)  100,100,90
 90   ISHORT=IY
      ILONG=IX
C CHECK FOR OVERLAP (ONLY PERMIT IDENTITY, BUT PERMIT UNEQUAL LENGTHS IF
C     IDENTICAL)
 100  IANS=-3
      IDIFF=IX-IY
      IF(IDIFF)  120,130,110
 110  IF(IDIFF-LY)  9999,130,130
 120  IF(-IDIFF-LX)  9999,130,130
C CLEAR OUTPUT AREA, FIX LONGEST AND (IF NOT IDENTICAL) SHORTEST.
C     REVERSE SHORTEST
 130  IANS=0
      DO 140  I=1,LC
 140  CC(I)=0.0
      CALL FXDATA(LLONG,CM(ILONG),MAX,SLONG)
```

```
      SSHORT=SLONG
      IF(SLONG)  9999,9999,150
 150  IF(IDIFF)  160,170,160
 160  CALL FXDATA (LSHORT,CM(ISHORT),MAX,SSHORT)
      IF (SSHORT) 910,910,170
 170  ASSIGN 300 TO IREV
      GO TO 200
C INTERNAL SUBROUTINE TO REVERSE SHORTEST
 200  LHAF=LSHORT/2
      IF(LHAF)  230,230,210
 210  DO 220  I=1,LHAF
      J=ISHORT+I-1
      K=ISHORT+LSHORT-I
      TEMP=CM(J)
      CM(J)=CM(K)
 220  CM(K)=TEMP
 230  GO TO IREV,(300,704,910)
C FIND NO. TERMS TO BE COMPUTED BY EACH OF THE THREE FASCOR,FASEPC,CALLS
C NCL=LEFT TERMS (FASCOR), NCM=MID TERMS (FASEPC), NCR=RIGHT TERMS
C SET FIRST LLC=ACTUAL NO. TERMS WHICH NEED TO BE COMPUTED.
 300  LLC=XMINOF(LC,LX+LY-1)
      NCL=XMINOF(LLC,LSHORT)
      NCM=0
      IF(LLC-NCL)  320,320,310
 310  NCM=XMINOF(LLC-NCL,LLONG-NCL-1)
 320  NCR=0
      IF (NCL+NCM-LLC) 330,700,700
 330  NCR=LLC-LLONG+1
C SET UP PROGRAM FOR SHORTEST
 700  IANS=-99
      LSPACE=NCL+10*(MAX+1)+1
      INCL=ISHORT+LSHORT-NCL
      CALL PROCOR(CM(INCL),NCL,MAX,SPACE(LSPACE),SPACE(1),ERR1)
C REREVERSE SHORTEST IF AUTOCONVOLUTION
      IF (IDIFF) 704,702,704
 702  ASSIGN 704 TO IREV
      GO TO 200
 704  IF (ERR1) 900,710,900
C CONVOLVE UP TO DISTANCE OF SHORTEST
 710  MINLAG=-NCL+1
      CALL FASCOR(CM(ILONG),MINLAG,0,CC(NCL),ERR2)
      IF(ERR2)  900,720,900
C CONVOLVE MIDDLE TERMS IF ANY
 720  IF(NCM) 740,740,730
 730  ICCM=NCL+NCM
      MINLAG=-NCM+1
      ILONGM=ILONG+NCM
      CALL FASEPC(CM(ILONGM),MINLAG,0,CC(ICCM),ERR3)
      IF(ERR3)  900,740,900
C CONVOLVE TAIL TERMS IF ANY
 740  IF(NCR)  760,760,750
 750  ICCR=NCL+NCM+1
      MAXLAG=NCR-1
      ILONGR = ILONG+NCM+1
      CALL FASCOR(CM(ILONGR),0,MAXLAG,CC(ICCR),ERR4)
      IF(ERR4)  900,760,900
C FLOAT CONVOLUTION
 760  IANS=0
      SCONV=SSHORT*SLONG
      CALL FLDATA(LLC,CC,SCONV)
C RE-REVERSE SHORTEST, REFLOAT LONGEST AND (MAYBE) SHORTEST
C (BUT AVOID REREVERSE FOR AUTO-CONVOLUTION)
 900  IF (IDIFF) 902,910,902
 902  ASSIGN 910 TO IREV
      GO TO 200
 910  CALL FLDATA(LLONG,CM(ILONG),SLONG)
      IF(IDIFF)  920,9999,920
 920  CALL FLDATA(LSHORT,CM(ISHORT),SSHORT)
C EXIT
 9999 RETURN
      END
```

```
*     QFURRY (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0180
*     LABEL
CQFURRY
      SUBROUTINE QFURRY(X,LX,IXZER,M,JMIN,JMAX,SPACE,CSP,SSP,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - QFURRY
C        FAST FOURIER TRANSFORM OF TRANSIENT WITH ARBITRARY TIME ORIGIN
C
C              QFURRY USES SUBROUTINE XSPECT TO OBTAIN A HIGH SPEED
C              FOURIER TRANSFORM OF THE TIME SERIES X(I), I=1...LX,
C              BASED ON THE ASSUMPTION THAT THE INDEX I=IXZER IS TO
C              CORRESPOND TO THE ZERO TIME ORIGIN, WHERE IXZER IS
C              ARBITRARILY SPECIFIED (IT MAY BE NEGATIVE). THE OUTPUTS
C              ARE THE REAL AND IMAGINARY PARTS OF THE FOURIER TRANSFORM
C              AND ARE EVALUATED OVER AN ARBITRARILY SPECIFIED FREQUENCY
C              RANGE WITH AN ARBITRARY FREQUENCY INCREMENT.
C
C              THE COMPUTATION IS AS FOLLOWS.  THE ORIGINAL SERIES
C                         X(I)  I=1...LX
C              UNDERGOES A TRANSLATION OF ORIGIN TO BECOME
C
C                        XT(I)  I= L,L+1,...,N
C                   WHERE
C                        L =  1 - IXZER      (NOTE L AND POSSIBLY N
C                        N = LX - IXZER       MAY BE NEGATIVE)
C
C              THE REAL AND IMAGINARY PARTS ARE THEN COMPUTED ON THE
C              TRANSLATED SERIES XT(I) AS FOLLOWS
C
C                           N
C                CS(J)  =   SUM ( XT(I)*COS(I*J*PI/M) )
C                           I=L
C
C                           N
C                SS(J)  =   SUM ( XT(I)*SIN(I*J*PI/M) )
C                           I=L
C
C                     FOR J = JMIN,JMIN+1,...,JMAX
C                WHERE
C                      PI = 3.14159265
C                      M, JMIN AND JMAX ARE INPUT PARAMETERS
C                          WITH THE RESTRAINT THAT
C                        0 LSTHN= JMIN LSTHN JMAX LSTHN= M
C
C              A BLOCK OF TEMPORARY REGISTERS IS REQUIRED
C                    OF LENGTH = LSPACE = 2*(M+K) + 6
C                    WHERE
C                         M IS DEFINED ABOVE
C                         K = GREATEST DISTANCE FROM THE ZERO INDEX
C                               TO THE TWO ENDS OF THE X SERIES,
C                               I.E. K = MAGNITUDE OF L OR N, AS
C                               DEFINED ABOVE, WHICHEVER IS GREATER
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 244 REGISTERS
C SPEED      - (CONTROLLED PRIMARILY BY SUBROUTINE XSPECT)
C              FOR  M LSTHN= K - 36*(JMAX-JMIN+1)*M MACHINE CYCLES
C              FOR  M GRTHN  K - 72*(JMAX-JMIN+1)*M MACHINE CYCLES
C AUTHOR     - S.M. SIMPSON JR., JUNE 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - STZ,MOVE,COSTBL,SINTBL,XSPECT
C       AND FORTRAN SYSTEM ROUTINES - (NONE)
C
C FORTRAN USAGE
C     CALL QFURRY(X,LX,IXZER,M,JMIN,JMAX,SPACE,CSP,SSP,IANS)
C
C INPUTS
C
C    X(I)      I=1...LX  LS THE INPUT TIME SERIES
C
```

```
C     LX          MUST EXCEED 1
C
C     IXZER       IS THE ZERO TIME INDEX.  MAY BE POSITIVE OR NEGATIVE
C                   AND MAY EXCEED LX.
C
C     M           CONTROLS THE FUNDAMENTAL FREQUENCY INCREMENT DESIRED AS
C                   SHOWN IN ABSTRACT.
C                 MUST EXCEED ZERO.
C
C     JMIN        CONTROLS THE LOWEST MULTIPLE OF THE FUNDAMENTAL
C                    FREQUENCY INCREMENT DESIRED, AS SHOWN IN ABSTRACT.
C                 MUST BE NON-NEGATIVE
C
C     JMAX        CONTROLS THE GREATEST MULTIPLE OF THE FUNDAMENTAL
C                   FREQUENCY INCREMENT DESIRED, AS SHOWN IN ABSTRACT.
C                 MUST EXCEED JMIN AND BE LESS THAN OR EQUAL TO M
C
C     SPACE(I)    I=1...LSPACE MUST BE AVAILABLE FOR SCRATCH, WHERE
C                   LSPACE IS DEFINED IN ABSTRACT.
C
C OUTPUTS
C
C     CSP(I)      I=1,2,...,JMAX-JMIN+1  CONTAINS CS(J), J=JMIN,...,JMAX ,
C                   AS DEFINED IN ABSTRACT.
C
C     SSP(I)      I=1,2,...JMAX-JMIN+1  CONTAINS SS(J), J=JMIN,...,JMAX ,
C                   AS DEFINED IN ABSTRACT.
C
C     IANS        =  0  NORMALLY
C                 = -1  IF LX IS ILLEGAL     (NO OTHER OUTPUT IN THIS CASE)
C                 = -2  IF M IS ILLEGAL                DITTO
C                 = -3  IF JMIN OR JMAX IS ILLEGAL      DITTO
C
C EXAMPLES
C
C 1. SIMPLE TIME SERIES WITH VARIOUS IXZER VALUES
C     INPUTS  - X(1...3) = 1.,1.,1.   LX=3,  M=10,  JMN=0,  JMX=2
C     USAGE   -        CALL QFURRY(X,LX,1,M,JMN,JMX,SPA,CSP1,SSP1,IANS1)
C                      CALL QFURRY(X,LX,2,M,JMN,JMX,SPA,CSP2,SSP2,IANS2)
C                      CALL QFURRY(X,LX,3,M,JMN,JMX,SPA,CSP3,SSP3,IANS3)
C                      CALL QFURRY(X,LX,0,M,JMN,JMX,SPA,CSP4,SSP4,IANS4)
C                      CALL QFURRY(X,LX,4,M,JMN,JMX,SPA,CSP5,SSP5,IANS5)
C     OUTPUTS - IANS1=IANS2=IANS3=IANS4=IANS5=0   AND
C               CSP1(1...3) =  3.00000, 2.76008, 2.11804
C               SSP1(1...3) =  0.00000, 0.89681, 1.53885
C               CSP2(1...3) =  3.00000, 2.90212, 2.61804
C               SSP2(1...3) =  0.00000, 0.00000, 0.00000
C               CSP3(1...3) =  3.00000, 2.76008, 2.11804
C               SSP3(1...3) =  0.00000,-0.89681,-1.53885
C               CSP4(1...3) =  3.00000, 2.34787, 0.80902
C               SSP4(1...3) =  0.00000, 1.70583, 2.48991
C               CSP5(1...3) =  3.00000, 2.34787, 0.80902
C               SSP5(1...3) =  0.00000,-1.70583,-2.48991
C
C 2. ILLEGAL CONDITIONS
C     USAGE   -        CALL QFURRY(X,1,1,2,0,2,SPACE,CSP,SSP,IANS1)
C                      CALL QFURRY(X,3,1,0,0,2,SPACE,CSP,SSP,IANS2)
C                      CALL QFURRY(X,3,1,2,-1,2,SPACE,CSP,SSP,IANS3)
C                      CALL QFURRY(X,3,1,2,0,3,SPACE,CSP,SSP,IANS4)
C     OUTPUTS - IANS1 = -1  (ILLEGAL LX)
C               IANS2 = -2  (ILLEGAL  M)
C               IANS3 = -3  (ILLEGAL JMIN)
C               IANS4 = -3  (ILLEGAL JMAX)
C
C PROGRAM FOLLOWS BELOW
C
C DUMMY DIMENSIONS
      DIMENSION  X(2), SPACE(2)
C CHECK LX AND M
      IANS = -1
      IF (LX-1) 9999,9999,10
 10   IANS = -2
      IF (M)    9999,9999, 20
C MAKE SETTINGS FOR SINE AND COSINE TABLE AND START OF CORR. BLOCK
 20   IXCSTB = 1
```

```
         IXSNTB = M+2
         IXCRNG = 2*M+3
   C FORM MAGNITUDES OF L AND N, AND CHECK LARGEST
         MAGL = XABSF ( 1 - IXZER)
         MAGN = XABSF (LX-IXZER)
         IF (MAGN-MAGL)  120,100,100
   C SET UP CONSTANTS FOR POS. BRANCH OF XT(I) LONGEST
    100  K = MAGN
         IXXMOV = IXCRNG + 2*K+1 - LX
         GO TO 130
   C SET UP CONSTANTS FOR NEG. BRANCH OF XT(I) LONGEST
    120  K = MAGL
         IXXMOV = IXCRNG
   C MAKE OTHER SETTINGS DEPENDENT ON K ALONE
    130  IXXCOR = IXCRNG + K
         LCR = 2*K+1
   C CLEAR THE CORRELATION AREA, THEN MOVE IN THE X SERIES
         CALL STZ(LCR, SPACE (IXCRNG))
         CALL MOVE (LX,X,SPACE(IXXMOV))
   C NOW SET UP THE COSINE AND SINE TABLES
         CALL COSTBL (M, SPACE(IXCSTB) )
         CALL SINTBL (M, SPACE (IXSNTB) )
   C FINALLY USE XSPECT, CHECKING FOR ILLEGAL JMIN, JMAX
         IANS = -3
         CALL XSPECT (SPACE (IXXCOR), K, SPACE (IXCSTB), SPACE (IXSNTB),
        1             M, JMIN, JMAX, CSP, SSP, SPACE (IXXCOR), ERR)
         IF (ERR)  9999,777,9999
    777  IANS = 0
   C EXIT
    9999 RETURN
         END
```

```
*     QIFURY (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0205
*     LABEL
CQIFURY
      SUBROUTINE QIFURY(FTREAL,FTIMAJ,MFREQ,LX,IXZER,SPACE,X,IANS)
C
C                     ----ABSTRACT----
C
C TITLE - QIFURY
C         QUICK INVERSE FOURIER TRANSFORM WITH ARBITRARY TIME ORIGIN
C
C               QIFURY USES SUBROUTINE COSISP TO OBTAIN A TIME SERIES
C               X(I), I=1...LX , HAVING ITS TIME ORIGIN AT ARBITRARY
C               INDEX IXZER,  FROM THE REAL AND IMAGINARY PARTS OF THE
C               FOURIER TRANSFORM OF X.  THE INPUT FOURIER TRANSFORM
C               IS GIVEN BY
C                      FTREAL(1...MFREQ+1) = FTR(0...MFREQ)
C                 AND  FTIMAJ(1...MFREQ+1) = FTI(0...MFREQ)
C               WHERE
C                   FTR(J) = REAL PART OF FOURIER TRANSFORM EVALUATED
C                            AT ANGULAR FREQUENCY  W = J*PI/MFREQ
C                   FTI(J) = IMAGINARY PART OF FOURIER TRANSFORM
C                            EVALUATED AT THE SAME FREQUENCY.
C
C               THE COMPUTATION IS
C
C                  X(1...LX)  =  XT(L,L+1,...N)
C                      WHERE  L =  1 - IXZER
C                             N = LX - IXZER
C
C               WHERE
C                            1   W=+PI
C                 XT(I)  = --- INTEGRAL ( FTR(W)*COS(I*W)  +
C                          2PI  W=-PI
C                                          FTI(W)*SIN(I*W) ) DW
C
C               WHERE THE INTEGRAL IS PERFORMED BY TRAPEZOIDAL
C                  APPROXIMATION AND ASSUMES FTR AND FTI ARE EVEN
C                  AND ODD FUNCTIONS.
C
C               A BLOCK OF 4*(MFREQ+1) TEMPORARY REGISTERS IS REQUIRED.
C
C               QIFURY IS AN APPROXIMATE INVERSE OPERATOR TO QFURRY.
C               THE INVERSE IS EXACT IF QFURRY AND QIFURY WERE
C               CALLED WITH THE SAME MFREQ AND IXZER PROVIDED THE
C               COMPLETE SPECTRUM (JMIN=0 JMAX=MFREQ) WAS COMPUTED
C               BY QFURRY AND THAT LX WAS LSTHN= 2*MFREQ-1, EXCEPT
C               THAT THE OUTPUT FROM QIFURY IS PERIODIC WITH PERIOD
C               2*MFREQ
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 280  REGISTERS
C SPEED      -         7090  709
C               ABOUT  (65  OR 72)*(MFREQ+1)*(MFREQ+1)  MACHINE CYCLES
C                  FOR LARGE MFREQ
C AUTHOR     - S.M. SIMPSON,  AUGUST 1963
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - COSTBL, SINTBL,COSISP
C       AND FORTRAN SYSTEM ROUTINES - XLOC
C
C FORTRAN USAGE
C     CALL QIFURY(FTREAL,FTIMAJ,MFREQ,LX,IXZER,SPACE,X,IANS)
C
C INPUTS
C
C    FTREAL(I) I=1...MFREQ+1  IS THE REAL PART OF THE FOURIER TRANSFROM
C
C    FTIMAJ(I) I=1...MFREQ+1  IS THE IMAGINARY PART OF THE FOURIER
C                 TRANSFORM
C
C    MFREQ     SHOULD EXCEED 0
C
```

```
C     LX         SHOULD EXCEED 0
C
C     IXZER      SPECIFIES THE ZERO TIME INDEX OF X(I).
C                MAY BE ANY VALUE, POSITIVE OR NEGATIVE (MAY EXCEED LX)
C
C     SPACE(I)   I=1...LSPACE MUST BE AVAILABLE FOR SCRATCH,
C                  WHERE  LSPACE = 4*M + 4
C
C OUTPUTS        STRAIGHT RETURN WITH NO OUTPUTS IF  LX OR MFREQ LSTHN 1
C
C     X(I)       I=1...LX  IS THE INVERSE TRANSFORM DETERMINED AS FOLLOWS.
C                LET  PI=3.14159265
C                     M=MFREQ
C                     S(J) = FTREAL(J+1) FOR J = 0, 1,...,M-1
C                     S(M) = FTREAL(M+1)/2
C                     S(J) = S(-J)       FOR J = -1,-2,...,-M
C                     A(J) = FTIMAJ(J+1) FOR J =  0, 1,..., M
C                     A(J) = -A(-J)      FOR J = -1,-2,...,-M
C
C                THE TRAPEZOIDAL APPROXIMATION USED FOR COMPUTING X(I)
C                IS THEN GIVEN BY
C
C                           1     M                               PI
C                  X(I) =  ---   SUM  ( S(J) * COS(J*(I-IXZER)*-----) +
C                          2*M   J=-M                              M
C
C                                                                 PI
C                                             A(J) * SIN(J*(I-IXZER)*-----)  )
C                                                                  M
C
C
C                    FOR  I = 1,2,...,LX
C
C                EQUIVALENCE(X,FTREAL OR FTIMAJ) IS PERMITTED
C
C     IANS       =  0  NORMALLY
C                = -1  MEANS ILLEGAL MFREQ
C                = -2  MEANS ILLEGAL LX
C
C EXAMPLES
C
C 1. INPUTS  - FTR(1...5) =  8., 0., -4., 0.,-16.
C              FTI(1...5) =  0., 0., 4., 0., 0.       M=4
C    USAGE   -        CALL QIFURY(FTR,FTI,M, 8, 1,SPACE,X1, IANS1)
C                     CALL QIFURY(FTR,FTI,M,16, 1,SPACE,X2, IANS2)
C                     CALL QIFURY(FTR,FTI,M, 8, 2,SPACE,X3, IANS3)
C                     CALL QIFURY(FTR,FTI,M, 8,-5,SPACE,X4, IANS4)
C                     CALL QIFURY(FTR,FTI,M, 4,27,SPACE,FTR,IANS5)
C                     CALL QIFURY(FTR,FTI,0, 8, 1,SPACE, X6,IANS6)
C                     CALL QIFURY(FTR,FTI,M,-1, 1,SPACE,X7, IANS7)
C    OUTPUTS - IANS1=IANS2=...=IANS5 = 0,  IANS6 = -1 IANS7 = -2
C              X1(1...8)  = -2., 4., 0., 2., -2., 4., 0., 2.
C              X2(1...8) = X1(1...8)  AND X2(9...16)=X1(1...8)
C              X3(1...8) = 2., -2., 4., 0., 2., -2., 4., 0.
C              X4(1...8) = 0., 2., -2., 4., 0., 2., -2., 4.
C              FTR(1...4) = 0., 2., -2., 4.
C
C 2. FOURIER TRANSFORM BY QFURRY WITH INVERSION BY QIFURY
C    INPUTS  - X(1...8) = -2.,4.,0.,2.,-2.,4.,0.,2.
C    USAGE   -        CALL QFURRY(X,8,1,4,0,4,SPACE,FTR1,FTI1,IANS8)
C                     CALL QIFURY(FTR1,FTI1,4,8,1,SPACE,X5,IANS9)
C
C                     CALL QFURRY(X,8,1,6,0,6,SPACE,FTR2,FTI2,IANS10)
C                     CALL QIFURY(FTR2,FTI2,6,11,1,SPACE,X6,IANS11)
C
C    OUTPUTS - IANS8 = IANS9 = IANS10 = IANS11 = 0
C              X5(1...8)  = -2.,4.,0.,2.,-2.,4.,0.,2.
C              X6(1...11) = -2.,4.,0.,2.,-2.,4.,0.,2.,0.,0.,0.
C                 (NOTE WE NEED TO DIMENSION FTR1(5),FTI1(5),FTR2(7),
C                  FTI2(7) )
C
C PROGRAM FOLLOWS BELOW
C
C DUMMY DIMENSIONS
      DIMENSION SPACE(2),X(2),FTREAL(2),FTIMAJ(2)
C BRING IN AND TEST MFREQ , LX
```

```
      M=MFREQ
      NX=LX
      IANS=-1
      IF (M)   9999,9999,10
 10   IANS=-2
      IF (NX)   9999,9999,20
C OK, END ADJUST FTREAL.  ( THE ENDS OF FTIMAJ SHOULD BE ZERO)
 20   IANS=0
      FTREAL(1)=FTREAL(1) /2.0
      FTREAL(M+1)=FTREAL(M+1) /2.0
C NOW COMPUTE THE TWO PARTS OF THE INVERSE TRANSFORM
C INTO SPACE (1) AND SPACE (M+2) WITH TABLES IN SPACE (2M+3) AND
C  SPACE (3M+4)
      ISANT=M+2
      ISCTAB=ISANT+M+1
      ISSTAB=ISCTAB+M+1
      CALL COSTBL(M,SPACE (ISCTAB))
      CALL SINTBL(M, SPACE (ISSTAB))
      CALL COSISP (FTREAL,FTREAL,FTIMAJ,FTIMAJ,M,SPACE(ISCTAB),
     1            SPACE(ISSTAB),M,0,M,1.0,SPACE(1),SPACE(ISANT))
C THE STARTING INDEX FOR XT IS XT(L)=XT(1-IXZER).
C WE HAVE TO FIND L AND PUT IT ,INCREMENTING BY 2M , IN THE
C INCLUSIVE RANGE  -M+1,-M+2,...,0,...M. THIS MODIFIED
C VALUE OF L WILL BE CALLED INEXT.  START BY MUDULO 2*M
      MDUBL=M+M
      L=1-IXZER
      INEXT = XMODF (L,MDUBL)
C THE MOD FUNCTION PUTS INEXT IN  -2M+1,...,2M-1
      IF (INEXT-M)   50,50,40
 40   INEXT=INEXT-MDUBL
      GO TO 70
 50   IF (INEXT+M)  60,60,70
 60   INEXT=INEXT+MDUBL
 70   FM=FLOATF(M)
C LOOP TO FORM X(1...LX)
      DO  120  IX=1,NX
C REDUCE INEXT BY 2*M WHENEVER IT INCREMENTS BEYOND M
      IF (INEXT-M)    90,90,80
 80   INEXT=INEXT-MDUBL
C INEXT IS NOW IN LEGAL RANGE
 90   MAGI=XABSF(INEXT)
      IANT=MAGI+ISANT
      TEMP= SPACE(IANT)
C REVERSE SIGN OF TEMP FOR NEGATIVE INEXT
       IF (INEXT)  100,110,110
 100  TEMP= -TEMP
C STORE X AND INCREMENT INEXT
 110  X(IX)= (SPACE(MAGI+1)+TEMP)/FM
      INEXT = INEXT+1
 120  CONTINUE
C RESCALE FTREAL UNLESS IT IS EQUIVALENT TO X
      IF  (XLOCF(FTREAL) - XLOCF(X))   130,9999,130
 130  FTREAL(1)= FTREAL(1)*2.0
      FTREAL(M+1)=FTREAL(M+1)*2.0
C EXIT
 9999 RETURN
      END
```

```
*     QINTR1 (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0191
*     LABEL
CQINTR1
      SUBROUTINE QINTR1 (X,XLO,DELX,TABLE,NTABLE,YOFX)
C
C
C                      ----ABSTRACT----
C
C TITLE - QINTR1
C        QUADRATIC INTERPOLATION IN A TABLE
C
C               QINTR1 USES QUFIT1 TO INTERPOLATE FOR A VALUE, WHICH LIES
C               AMONG THREE SUCCESSIVE TABULATED VALUES, BY FITTING A
C               PARABOLA.  LINEAR INTERPOLATION OCCURS IF THERE ARE ONLY
C               TWO TABLE VALUES.  XLO IS THE ARGUMENT CORRESPONDING TO
C               THE LOWEST TABLE VALUE.  DELX IS THE ARGUMENT DIFFERENCE
C               BETWEEN TABLE VALUES.  THE FORMULA
C
C                     YOFX = COEFS(1)+COEFS(2)*XREL+COEFS(3)*XREL**2
C
C               IS USED TO FIND THE INTERPOLATED VALUE,  WHERE
C                     YOFX IS THE INTERPOLATED VALUE,
C                     COEFS(1...3) ARE COEFFICIENTS COMPUTED BY QUFIT1,
C                       BASED ON THREE TABLE VALUES CHOSEN BY QINTR1,
C                     XREL IS A FRACTIONAL ARGUMENT VALUE RELATIVE TO
C                       THE MIDDLE CHOSEN TABLE VALUE.
C
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709/7090/7094 (MAIN FRAME ONLY)
C STORAGE    - 229 REGISTERS
C SPEED      - ABOUT  .6  TO  .7  MILLISECONDS ON THE  7094 .
C AUTHOR     - J. PROCITO,   MAY 1964
C
C
C                      ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - RNDUP, QUFIT1
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C     CALL QINTR1(X,XLO,DELX,TABLE,NTABLE,YOFX)
C
C
C INPUTS
C
C    X          IS THE ARGUMENT FOR WHICH INTERPOLATION IS DESIRED
C               SHOULD BE GRTHN= XLO AND LSTHN= XLO+(NTABLE-1)*DELX
C
C    XLO        IS THE ARGUMENT CORRESPONDING TO THE FIRST TABLE ENTRY
C
C    DELX       IS THE ARGUMENT DIFFERENCE BETWEEN TABLE ENTRIES
C               MUST EXCEED ZERO
C
C    TABLE(I)   I=1...NTABLE  IS THE TABLE OF VALUES TO BE USED FOR
C                 INTERPOLATING
C
C    NTABLE     IS LENGTH OF  TABLE
C               MUST BE GRTHN= 2
C
C
C OUTPUTS       STRAIGHT RETURN IF  DELX  OR  NTABLE  ILLEGAL.
C
C    YOFX       THE INTERPOLATED VALUE DESIRED CORRESPONDING TO X.
C               EXCEPT = 0.0 IF X IS ILLEGAL.
C
C
C EXAMPLES
C
C 1. INPUTS  - TABLE(1) = 1.  NTABLE=1  X=2.  XLO=0.  DELX=1.  YOFX=-99.
C    OUTPUTS - YOFX=-99.
C
C 2. INPUTS  - TABLE(1...2) = 0.,12.  NTABLE=2  X=1.5  XLO=1.  DELX=1.
C    OUTPUTS - YOFX=6.0
C
```

```
C 3. INPUTS  - TABLE(1...3) = 0.,6.,12.  NTABLE=3  X=2.333  XLO=1.
C               DELX=1.
C    OUTPUTS - YOFX=8.
C
C 4. INPUTS  - TABLE(1...4) = 0.,6.,12.,18,  NTABLE=4  X=2.3  XLO=1.
C               DELX=1.
C    OUTPUTS - YOFX=7.8
C
C 5. INPUTS  - SAME AS EXAMPLE 4. EXCEPT X=2.8
C    OUTPUTS - YOFX=10.8
C
C 6. INPUTS  - TABLE(1...4) = 0.,3.333,6.667,10.  X=1.4   REST = EX. 4.
C    OUTPUTS - YOFX=1.3333
C
C 7. INPUTS  - TABLE(1...8)= 0.,3.333,6.6667,10.,13.333,16.667,20.,23.33
C               NTABLE=8  X=6.7  XLO=1.  DELX=2.
C    OUTPUTS - YOFX=9.5
C
C 8. INPUTS  - SAME AS EXAMPLE 7. EXCEPT X=4.
C    OUTPUTS - YOFX=5.0
C
C 9. INPUTS  - SAME AS EXAMPLE 7. EXCEPT X=0.
C    OUTPUTS - YOFX=0.
C
C10. INPUTS  - SAME AS EXAMPLE 7. EXCEPT X=7.
C    OUTPUTS - YOFX=10.0
C
C11. INPUTS  - SAME AS EXAMPLE 7. EXCEPT DELX=1.   X=4.
C    OUTPUTS - YOFX=10.0
C
C12. INPUTS  - SAME AS EXAMPLE 11. EXCEPT XLO=2.
C OUTPUTS - YOFX=6.6667
C
C13. INPUTS  - SAME AS EXAMPLE 12. EXCEPT X=2.3
C    OUTPUTS - YOFX=1.0
C
C14. INPUTS  - SAME AS EXAMPLE 12. EXCEPT X=2.8
C    OUTPUTS - YOFX=2.666
C
C15. INPUTS  - SAME AS EXAMPLE 12. EXCEPT X=1.4
C    OUTPUTS - YOFX=0.
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION TABLE(3),COEFS(3)
C
C CHECK FOR ILLEGAL  NTABLE,  DELX.
C
      IF (NTABLE-1)  9991,9991,1
 1    IF (DELX)  9991,9991,2
 2    IF (X-XLO)  999,5,5
C
C COMPUTE ILO,IHI,XREL (XREL= X VALUE RELATIVE TO ILO, IHI)
C
 5    XREL=(X-XLO)/DELX+1.
      ILO=XREL
      IHI= RNDUPF(XREL)
      IF (IHI-NTABLE)  7,7,999
C
C BEGIN CHECKS FOR UPPER AND LOWER LIMITS.
C
 7    IF (IHI-ILO)  9991,200,10
 10   IF (ILO-1)  999,20,40
C
C ILO = 1 .  NOW BRANCH ON TABLE LENGTH.
C
 20   IF (NTABLE-2)  9991,30,50
C
C NTABLE = 2. SINCE THERE ARE ONLY 2 POINTS, INTERPOLATE LINEARLY, EXIT.
C
 30   YOFX=TABLE(1)+((X-XLO)*(TABLE(2)-TABLE(1)))/DELX
      GO TO 9991
C
C ILO GRTHN 1, NOW CHECK IHI.
C
```

```
 40   IF (IHI-NTABLE)  70,60,999
C
C NTABLE IS GRTHN 2 AND ILO = 1, SO SET IMID = 2 AND FIT PARABOLA
C
 50   IMID=2
      GO TO 90
C
C IHI = NTABLE, ILO GRTHN 1, SO IMID = NTABLE-1
C
 60   IMID=NTABLE-1
      GO TO 90
C
C TEST FRACTIONAL PART OF X VALUE AND FIND OUT WHETHER IT IS CLOSER TO
C     ILO OR IHI.  SET IMID = TO THE INDEX THE VALUE IS CLOSEST TO.
C
 70   XFRAC=XREL-FLOATF(ILO)
      IF (XFRAC-.5)  75,80,80
 75   IMID=ILO
      GO TO 90
 80   IMID=IHI
C
C FIND COEFFICIENTS FOR EQUATION
C
 90   CALL QUFIT1(TABLE(IMID-1),-1.,1.,COEFS)
C
C COMPUTE YOFX WITH COEFS AND XREL
C
      XREL=XREL-FLOATF(IMID)
      YOFX=COEFS(1)+COEFS(2)*XREL+COEFS(3)*XREL*XREL
      GO TO 9991
C
C IF ILO=IHI YOFX=TABLE(ILO)
C
 200  IF (ILO-NTABLE) 300,300,999
 300  YOFX=TABLE(ILO)
      GO TO 9991
C
C ALL X VALUES OUT OF TABLE RANGE =0.0
C
 999  YOFX=0.0
 9991 RETURN
      END
```

```
*       QUFIT1 (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0199
*       FAP
*QUFIT1
       COUNT   200
       LBL     QUFIT1
       ENTRY   QUFIT1 (FOFX,XLO,DELX,COEFS)
*
*
*                    ----ABSTRACT----
*
* TITLE - QUFIT1
*       FIND QUADRATIC WHICH EXACTLY FITS 3 EQUALLY SPACED POINTS
*
*              QUFIT1 FINDS C0,C1, AND C2 SUCH THAT THE QUADRATIC
*              POLYNOMIAL
*                                             2
*                    F(X) = C0+C1*X+C2*X
*
*              TAKES ON SPECIFIED VALUES AT X=XLO,XLO+DELX, AND
*              XLO+2*DELX, WHERE XLO AND DELX ARE PARAMETERS.
*
*              QUFIT1 HAS A HIGH SPEED AUTOMATIC BYPASS FOR THE CASE
*              THAT XLO=-1.0 AND DELX=1.0
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 79 REGISTERS
* SPEED      - ABOUT 250 MACHINE CYCLES IN GENERAL (ON THE 7090)
*                     79 MACHINE CYCLES IF XLO=-1.0, AND DELX=1.0
* AUTHOR     - S.M.SIMPSON, MARCH 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL QUFIT1(FOFX,XLO,DELX,COEFS)
*
*
* INPUTS
*
*    FOFX(I)   I=1,2,3  ARE THE VALUES WHICH THE POLYNOMIAL MUST ASSUME.
*
*    XLO       IS STARTING VALUE OF ARGUMENT X.
*
*    DELX      IS ARGUMENT INCREMENT.
*                    IF DELX=0.0 QUFIT1 COMPUTES AND EXITS AS THOUGH
*                    USER HAD SPECIFIED XLO=-1.0 (THE ACTUAL XLO IS NOT
*                    USED) AND DELX=1.0 .  THIS CASE TAKES 79 MACHINE
*                    CYCLES.  IF DELX AND XLO ARE ACTUALLY SPECIFIED TO
*                    BE 1.0 AND -1.0 RESPECTIVELY, ABOUT 91 MACHINE
*                    CYCLES ARE TAKEN.
*
*
* OUTPUTS
*
*    COEFS(I)  I=1,2,3  WILL CONTAIN C0,C1, AND C2, RESPECTIVELY, SUCH
*                    THAT THE POLYNOMIAL F(X) GIVEN IN THE ABSTRACT WILL
*                    SATISFY
*                            F(XLO)          =  FOFX(1)
*                            F(XLO+DELX)     =  FOFX(2)
*                            F(XLO+2*DELX)   =  FOFX(3)
*
*
* EXAMPLES
*
* 1. INPUTS  - FOFX(1...3) = 2.0,3.0,6.0     XLO=-1.0,  DELX=1.0
*    USAGE   -         CALL QUFIT1(FOFX,XLO,DELX,COEFS1)
*                      CALL QUFIT1(FOFX,3.0, 0.0,COEFS2)
*    OUTPUTS - COEFS1(1...3) = COEFS2(1...3) = 3.0,2.0,1.0
*
* 2. INPUTS  - FOFX(1...3) = 3.0,3.0,11.0     XLO=-2.0,  DELX=2.0
*    USAGE   -         CALL QUFIT1(FOFX,XLO,DELX,COEFS3)
```

```
*    OUTPUTS - COEFS3(1...3) = 3.0,2.0,1.0
*
* 3. INPUTS  - FOFX(1...3) = 1.0,2.0,3.0    XLO=1.0,  DELX=1.0
*    USAGE   -       CALL QUFIT1(FOFX,XLO,DELX,COEFS4)
*    OUTPUTS - COEFS4(1...3) = 0.0,1.0,0.0
*
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
*
       HTR     0              XR4
       BCI     1,QUFIT1
*
* ONLY ENTRY.  QUFIT1(FOFX,XLO,DELX,COEFS)
*
QUFIT1 SXD     QUFIT1-2,4
       CLA     1,4            A(FOFX)
       SUB     K1
       STA     CLAF2
       SUB     K1
       STA     CLAF3
       CLA     4,4   A(COEFS)
       SUB     K1
       STA     STOC1
       SUB     K1
       STA     STOC2
*
* TRIAL SETTINGS
*
* COEFS(1) = C0 = FOFX(2).  AND FORM 2C0 .
*
 CLAF2 CLA     **             **=A(FOFX)-1
       STO*    4,4
       TMI     *+3
       ADD     KDUBL
       TRA     *+2
       SUB     KDUBL
       STO     TWOCZ
*
* COEFS(2) = C1 = (FOFX(3)-FOFX(1))/2.0
*
 CLAF3 CLA     **             **=A(FOFX)-2
       FSB*    1,4
       TMI     *+3
       SUB     KDUBL
       TRA     *+2
       ADD     KDUBL
 STOC1 STO     **             **=A(COEFS)-1
*
* COEFS(3) = C2 = (FOFX(3)-2*FOFX(2)+FOFX(1))/2.0
*
       CLA*    CLAF3
       FSB     TWOCZ
       FAD*    1,4
       TMI     *+3
       SUB     KDUBL
       TRA     *+2
       ADD     KDUBL
 STOC2 STO     **             **=A(COEFS)-2
*
* QUIT IF DELX=0
*
       CLA*    3,4
       TZE     5,4
*
* ALL DONE IF DELX=1.0 AND XLO=-1.0
*
       CAS     K1L
       TRA     REVISE         FAIL
       TRA     *+2            MAYBE
       TRA     REVISE         FAIL
       CLS*    2,4            XLO CHECK
       CAS     K1L
```

```
***********************
*    QUFIT1           *
***********************
(PAGE  3)
```


```
       TRA     REVISE
       TRA     5,4             EXIT
*
* FOR REVISION SET G = -(XLO/DELX + 1)
*      AND
*              C0 = (G**2)*C2+G*C1+C0
*              C1 = (2*G*C2 +C1)/DELX
*              C2 = C2/(DELX**2)
*
REVISE CLA*    STOC2           C2
       STO     C2              (PUT ASIDE)
       FDP*    3,4             C2/DELX
       XCA
       FDP*    3,4             /DELX AGAIN
       STQ*    STOC2           = C2
       CLA*    STOC1
       STO     C1
       CLS*    2,4             -XLO
       FDP*    3,4             -XLO/DELX
       XCA
       FSB     K1L             -(XLO/DELX+1.0)
       STO     G
       FAD     G               2G
       XCA
       FMP     C2
       FAD     C1
       FDP*    3,4
       STQ*    STOC1
       LDQ     C2
       FMP     G
       FAD     C1
       XCA
       FMP     G
       FAD*    4,4
       STO*    4,4
*
* EXIT
*
       TRA     5,4
*
* CONSTANTS, TEMPORARIES
*
 K1    PZE     1
 K1L   DEC     1.0
 KDUBL OCT     001000000000
 TWOCZ PZE     **,**,**        2*C0
 G     PZE     **,**,**        -(XLO/DELX + 1)
 C2    PZE     **,**,**        TRIAL C2
 C1    PZE     **,**,**        TRIAL C1
       END
```

```
*     QXCORR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0248
*     LABEL
CQXCORR
      SUBROUTINE QXCORR (X,Y,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - QXCORR
C        FAST CROSS-CORRELATIONS FOR LONG, LIMITED ACCURACY SERIES
C
C              QXCORR COMPUTES THE UNNORMALIZED CROSS-CORRELATION
C              FUNCTION, XC(L), OF TWO LIMITED ACCURACY SERIES, X(I)
C              AND Y(I) BOTH OF LENGTH LXY, ACCORDING TO THE TRANSIENT
C              FORMULA
C
C                                1     LXY
C                   XC(L)       --- *  SUM  ( X(I)*Y(I+L) )
C                               LXY    I=1
C
C                   FOR L = -MXLAG,-MXLAG+1,...,-1,0,1,...,MXLAG
C
C                      WHERE  Y(K) IS ASSUMED = 0.0 WHENEVER K IS
C                                   OUTSIDE OF THE RANGE  1 TO LXY
C                             MXLAG AND LXY ARE INPUT PARAMETERS
C
C              SPEED IS CONTROLLED BY THE SERIES LENGTH AND THE
C              SERIES ACCURACY. FOR VERY LONG SERIES A COMPLETE CROSS-
C              CORRELATION (MXLAG = LXY-1) CAN BE COMPUTED IN SLIGHTLY
C              MORE THAN 2*(LXY(SQUARED)) MACHINE CYCLES.  QXCORR OBTAINS
C              THIS SPEED PRIMARILY BY CONVERTING X(I) AND Y(I) TO
C              INTEGER SEQUENCES IX(I) AND IY(I) WHOSE MAGNITUDES HAVE
C              UPPER LIMIT AS SPECIFIED BY AN INPUT PARAMETER MXACC, AND
C              THEN REGROUPS THE ABOVE EQUATION (FOR EACH LAG) SO AS TO
C              PERFORM 2*(LXY-L)-1 ADDITIONS PLUS MXACC (OR FEWER) MULTI-
C              PLICATIONS (RATHER THAN 2*(LXY-L)-1 ADDITIONS PLUS
C              2*(LXY-L)-1 MULTIPLICATIONS).(SEE SUBROUTINE PROCOR-FASCOR
C              FOR LOGIC DETAILS.)  THE RESULTS ARE THEN RECONVERTED TO
C              FLOATING POINT FORM WITH CORRECT SCALE.  IX(I) AND IY(I)
C              ARE ALSO REFLOATED.
C
C              USER MUST PROVIDE QXCORR WITH A BLOCK OF TEMPORARY
C              REGISTERS OF LENGTH  LXY + 10*(MXACC+1) + 1  .
C
C              X(I) AND Y(I) ARE LEFT SLIGHTLY MODIFIED BY THE FIXING,
C              REFLOATING PROCESS
C
C              IF QXCORR DETECTS THAT THE X AND Y SERIES ARE THE SAME
C              IT COMPUTES AND STORES XC(L) ONLY FOR POSITIVE LAGS SO
C              THAT QXCORR CAN BE USED FOR EFFICIENT AUTOCORRELATIONS
C              AS WELL AS CROSS-CORRELATIONS.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - IBM 709, 7090 (MAIN FRAME ONLY)
C STORAGE   - 283 REGISTERS
C SPEED     - FOR LONG SERIES QXCORR TAKES ABOUT
C             (2*MXLAG+1)*(2*LX-MXLAG+20*MXACC) MACHINE CYCLES
C               (DIVIDE THIS BY 2 IF X AND Y ARE EQUIVALENT)
C AUTHOR    - S. M. SIMPSON JR, 10/10/62
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   FXDATA, PROCOR, FASCOR, FLDATA
C        AND FORTRAN SYSTEM ROUTINES -   XLOC
C
C FORTRAN USAGE
C       CALL QXCORR(X,Y,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)
C
C INPUTS
C
C    X(I)       I=1...LXY IS THE FIRST SERIES
C
C    Y(I)       I=1...LXY IS THE SECOND SERIES
C               EQUIVALENCE (X,Y) IS PERMITTED  (GIVING AUTO COREL.)
C               NO OTHER OVERLAP OF X AND Y IS PERMITTED
C
```

```
C     LXY         MUST EXCEED ZERO AND BE LSTHN= 10000
C
C     MXACC       DEFINES ACCURACY OF THE TWO SERIES. X(I) AND Y(I) WILL
C                    BE FIXED SO AS TO HAVE VALUES LYING BETWEEN -MXACC
C                    AND +MXACC INCLUSIVE.
C                 MUST LIE BETWEEN 1 AND 1000 INCLUSIVE. (SMALLER VALUES
C                    YIELD HIGHER SPEEDS AND REQUIRE FEWER TEMPORARIES.)
C
C     MXLAG       IS THE HIGHEST LAG NO. DESIRED IN THE CROSS-CORRELATION
C
C     SPACE(I)    I=1...LSPACE MUST BE AVAILABLE AS TEMPORARIES, WHERE
C                        LSPACE = LX + 10*(MXACC+1) + 1
C
C OUTPUTS
C
C     X(I)        I=1...LXY CONTAINS THE ROUNDED SERIES XX(I)
C                           XX(I) = FLOATF(IX(I))/SCALEX
C                     WHERE
C                           IX(I) = XFIXF(X(I)*SCALEX)
C                          SCALEX = FLOATF(MXACC)/XMAX
C                            XMAX = LARGEST X MAGNITUDE
C                 X(I) IS LEFT = 0.0 IF XMAX = 0.0
C
C     Y(I)        I=1...LXY CONTAINS THE ROUNDED SERIES YY(I)
C                           YY(I) = FLOATF(IY(I))/SCALEY
C                     WHERE
C                           IY(I) = XFIXF(Y(I)*SCALEY)
C                          SCALEY = FLOATF(MXACC)/YMAX
C                            YMAX = LARGEST Y MAGNITUDE
C                 Y(I) IS LEFT = 0.0 IF YMAX = 0.0
C                    (NOTE- XFIXF IN ABOVE EXPRESSIONS IMPLIES ROUNDING
C                       TO NEAREST INTEGER, NOT TRUNCATION)
C
C     XCOR(I)     I=1,... CONTAINS THE CORRELATION FUNCTION
C                 IF X AND Y ARE DIFFERENT SERIES
C                    XCOR(1,2,...,2*MXLAG+1) CONTAINS THE CROSS-
C                    CORRELATION FUNCTION FROM NEGATIVE TO POSITIVE LAGS
C                 AS COMPUTED ON THE ROUNDED SERIES
C                    I.E.  XCOR(I) = XC(I-1-MXLAG)  I=1,...,2*MXLAG+1
C                    WHERE
C                                   1    LXY
C                       XC(L)  =  --- * SUM ( XX(I)*YY(I+L) )
C                                  LXY   I=1
C
C                         FOR L = -MXLAG,-MXLAG+1,...,MXLAG
C                         AND YY(K) ASSUMED = 0.0 WHENEVER K IS OUTSIDE
C                                 THE RANGE 1 TO LXY
C                 IF X AND Y ARE EQUIVALENT (XLOCF(X)=XLOCF(Y))
C                    XCOR(1,2,...,MXLAG+1) CONTAINS THE AUTOCORRELATION
C                    FUNCTION FROM LAG ZERO TO LAG MXLAG
C                    I.E.     XCOR(I) = XC(I-1)    I = 1,...,MXLAG+1
C                 XCOR(I) WILL BE IDENTICALLY ZERO IF X(I) OR Y(I) IS
C
C     IANS        =   0  IF NO TROUBLE ARISES
C                 =  -2  IF Y PARTIALLY OVERLAPS X
C                 =  -3  IF LXY IS ILLEGAL
C                 =  -4  IF MXACC IS ILLEGAL
C                 =  -5  IF MXLAG IS ILLEGAL
C                 = -98  IF UNEXPLAINED ERROR RETURN FROM PROCOR OCCURS
C                 = -99  IF UNEXPLAINED ERROR RETURN FROM FASCOR OCCURS
C
C EXAMPLES        THE FIRST 4 EXAMPLES ARE CHOSEN SO THAT THE ROUNDOFF
C                 EFFECT IS NOT PRESENT
C
C                 CALL QXCORR(X,Y,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)  IS
C                 THE ASSUMED USAGE IN ALL EXAMPLES UNLESS OTHERWISE STATED
C
C 1. COMPLETE CROSS CORRELATION
C     INPUTS  - X(1...5) = 10.,20.,10.,10.,5.   Y(1...5)=1.,1.,1.,1.,1.
C               LXY=5  MXACC=20  MXLAG=4
C     OUTPUTS - X(1...5) AND Y(1...5) = INPUT VALUES   IANS=0
C               XCOR(1...9)=1.,3.,5.,9.,11.,10.,8.,6.,2.
C
C 2. PARTIAL CROSS-CORRELATION
C     INPUTS  - SAME AS EXAMPLE 1 EXCEPT MXLAG=2
```

```
C     OUTPUTS - SAME AS EXAMPLE 1 EXCEPT XCOR(1...5)=5.,9.,11.,10.,8.
C
C 3. CORRELATION BEYOND END OF SERIES
C     INPUTS  - SAME AS EXAMPLE 1 EXCEPT MXLAG=6
C     OUTPUTS - SAME AS EXAMPLE 1 EXCEPT TERMINAL ZEROES ADDED TO XCOR
C               IE  XCOR(1...13)=0.,0.,1.,3.,5.,9.,11.,10.,8.,6.,2.,0.,0.
C
C4. COMPLETE AUTOCORRELATION
C     INPUTS  - SAME AS EXAMPLE 1
C     USAGE   -       CALL QXCORR(X,X,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)
C     OUTPUTS - SAME AS EXAMPLE 1 EXCEPT
C               XCOR(1...5) = 145.,110.,70.,40.,10.
C
C 5. COMPLETE CROSS-CORRELATION SHOWING ROUNDOFF
C     INPUTS  - X(1...3)=23.8,148.0,20.3   Y(1...3)=1.,1.,1.
C               LXY=3  MXACC=100  MXLAG=2
C     OUTPUTS - X(1...3)=23.68,148.0,20.72   Y(1...3)=1.,1.,1.  IANS=0
C               XCOR(1...5)=6.90667,56.2400,64.1333,57.2267,7.8933
C
C 6. THE NEXT FOUR EXAMPLES SHOW ERROR CONDITIONS
C     INPUTS  - SAME AS EXAMPLE 1
C     USAGE   -       CALL QXCORR(X,X(2),LXY,MXACC,MXLAG,SPACE,XCOR,IANS)
C               OR    CALL QXCORR(X(2),X,LXY,MXACC,MXLAG,SPACE,XCOR,IANS)
C     OUTPUTS - IANS = -2  (X AND Y PARTIALLY OVERLAP)
C
C 7. INPUTS  - SAME AS EXAMPLE 1 EXCEPT LXY=0
C     OUTPUTS - IANS=-3  (ILLEGAL LXY)
C
C 8. INPUTS  - SAME AS EXAMPLE 1 EXCEPT MXACC=1500
C     OUTPUTS - IANS=-4  (ILLEGAL MXACC)
C
C 9. INPUTS  - SAME AS EXAMPLE 1 EXCEPT MXLAG=-2
C     OUTPUTS - IANS=-5  (ILLEGAL MXLAG)
C
C10. (SPECIAL TEST FOR BYPASS ON X(I) ALL ZERO)
C     INPUTS  - SAME AS EXAMPLE 1 EXCEPT X(1...5)=0.,0.,...
C     OUTPUTS - XCOR(1...9)=0.,0.,...
C
C11. INPUTS  - SAME AS EXAMPLE 1 EXCEPT Y(1...5)=0.,0.,...
C     OUTPUTS - XCOR(1...9)=0.,0.,...
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION X(2),Y(2),SPACE(2),XCOR(2)
C CHECK INPUTS
      IANS=-3
      IF (LXY) 9999,9999,5
 5    IF (LXY-10000) 10,10,9999
 10   IANS=-4
      IF (MXACC) 9999,9999,20
 20   IF (MXACC-1000) 30,30,9999
 30   IANS=-5
      IF (MXLAG) 9999,40,40
C SET ACTUAL NO. OF LAGS TO BE COMPUTED
 40   NLAGS = XMINOF(MXLAG,LXY-1)
C SET SWITCH = 0 FOR CASE X EQUIV Y AND CHECK PARTIAL OVERLAP
      IANS=-2
      IDIFF = XABSF(XLOCF(X)-XLOCF(Y))
      IF (IDIFF)  55,55,52
 52   IF (IDIFF-LXY)  9999,60,60
C SET FOR X AND Y EQUIVALENT
 55   NCORS = MXLAG+1
      KSTORE = 1
      LMIN = 0
      GO TO 65
C SET FOR X AND Y DIFFERENT
 60   NCORS = 2*MXLAG+1
      KSTORE = MXLAG+1
      LMIN = -NLAGS
C CLEAR OUTPUT AREA
 65   DO 66  I=1,NCORS
 66   XCOR(I) = 0.0
C SET SPACE CONSTANT FOR PROCOR AND FIX X AND Y. EXIT FOR ZERO VECTORS.
 70   LSPACE=LXY+10*(MXACC+1)+1
      IANS=0
```

```
      CALL FXDATA(LXY,X,MXACC,SCALEX)
      IF (SCALEX)  9999,9999,80
 80   SCALEY=SCALEX
      IF (IDIFF)  85,90,85
 85   CALL FXDATA(LXY,Y,MXACC,SCALEY)
      IF (SCALEY)  900,900,90
C COMPUTE CROSS CORRELATION
 90   IANS=-98
      CALL PROCOR(X,LXY,MXACC,SPACE(LSPACE),SPACE(1),ANSR)
      IF (ANSR) 900,100,900
 100  IANS=-99
      CALL FASCOR(Y,LMIN,NLAGS,XCOR(KSTORE),ANSR)
      IF (ANSR) 900,120,900
C NOW FLOAT AND SCALE XCOR
 120  IANS=0
      SCXC=SCALEX*SCALEY*FLOATF(LXY)
      CALL FLDATA(NCORS,XCOR,SCXC)
C REFLOAT X AND Y
 900  CALL FLDATA(LXY,X,SCALEX)
      IF (IDIFF)  905,9999,905
 905  CALL FLDATA(LXY,Y,SCALEY)
C EXIT
 9999 RETURN
      END
```

```
*     QXCOR1 (SUBROUTINE)               3/15/65   LAST CARD IN DECK IS NO. 0197
*     LABEL
CQXCOR1
      SUBROUTINE QXCOR1 (LXX,XX,LYY,YY,MXACC,ILAG,NLAGS,CORR,IAD,
     1                  LSPACE,SPACE,IANS)
C
C                  ----ABSTRACT----
C
C TITLE - QXCOR1
C        QUICK CROSSCORRELATION OF MLI TRANSIENTS
C
C              QXCOR1 FINDS THE CROSSCORRELATION OF TWO MLI (MACHINE
C              LANGUAGE INTEGER) TRANSIENTS X(I) I=1,...,LX  Y(J)
C              J=1,...,LY ACCORDING TO THE FORMULA
C                              M
C                    C(L)  =  SUM  ( X(I+L) * Y(I) )
C                             I=-M
C
C                    FOR L = ILAG,...,ILAG+NLAGS-1
C              WHERE
C                    M IS GRTHN LX + LY  (X(I) AND Y(I) ARE ASSUMED TO
C                        BE ZERO OUTSIDE THE LIMITS OF THEIR DEFINITION)
C                    LX,LY,ILAG, AND NLAGS ARE INPUT PARAMETERS
C                    ADDITION INTO THE OUTPUT AREA IS MADE AT THE
C                        OPERATORS DISCRETION.
C
C              QXCOR1 OBTAINS ITS SPEED BY OPERATING SUBROUTINES PROCOR,
C              FASCR1, AND FASEP1 ON THE INPUT SERIES.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 502 REGISTERS
C SPEED      - CASE 1.  WHEN ONE SERIES IS VERY LONG AND ALSO CON-
C                 SIDERABLY LONGER THAN THE OTHER THE 7090 TIME
C                 APPROACHES  2*NLAGS*L  MACHINE CYCLES ,  WHERE
C                 NLAGS = DESIRED NO. OF OUTPUT CORRELATIONS AND
C                 L = LENGTH OF SHORTER SERIES.
C              CASE 2.  WHEN CROSS-CORRELATING TWO LONG, EQUAL-LENGTH
C                 SERIES FOR LAGS OF  -MXLAG TO +MXLAG  THE 7090
C                 TIME APPROACHES  2*MXLAG*(2*L-MXLAG)  MACHINE CYCLES
C                 WHERE L IS THE COMMON LENGTH.  AUTOCORRELATIONS TAKE
C                 HALF AS LONG.
C AUTHOR     - R.A. WIGGINS   JUNE,1963
C
C                  ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FASCR1,FASEP1,IXCARG,LIMITS,
C                                     PROCOR,REVERS,SETKS,STZ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL QXCOR1(LXX,XX,LYY,YY,MXACC,ILAG,NLAGS,CORR,IAD,
C    1             LSPACE,SPACE,IANS)
C
C INPUTS
C
C    LXX       =LX IS THE LENGTH OF X(I).
C              MUST BE GRTHN= 1
C
C    XX(I)     I=1,...,LXX CONTAINS THE MLI VECTOR X(I).
C
C    LYY       =LY IS THE LENGTH OF Y(I).
C              MUST BE GRTHN= 1
C
C    YY(I)     I=1,...,LYY CONTAINS THE MLI VECTOR Y(I).
C
C    MXACC     DEFINES THE ACCURACY OF THE VECTORS XX(I) AND YY(I).
C                ALL VALUES OF XX(I) AND YY(I) MUST LIE BETWEEN -MXACC
C                AND +MCACC INCLUSIVELY.
C              MUST BE GRTHN=1, LSTHN= 1000
C
C    ILAG      IS THE INITIAL LAG AT WHICH THE CORRELATION IS BEGUN.
C
C    NLAGS     IS THE NUMBER OF LAGS FOR WHICH THE CORRELATION IS FOUND.
```

```
C              MUST BE GRTHN= 1
C
C    IAD       =0     IMPLIES CORRELATION REPLACES OUTPUT VECTOR.
C              NOT= 0 IMPLIES CORRELATION IS ADDED TO THE OUTPUT VECTOR.
C
C    LSPACE    IS THE LENGTH OF TEMPORARY COMPUTATION SPACE AVAILABLE TO
C                QXCOR1.
C              MUST BE GRTHN= MIN(LXX,LYY) + 1 + 10*(MXACC+1)
C
C    SPACE(I)  I=1,...,LSPACE IS TEMPORARY COMPUTATION SPACE NEEDED
C                BY QXCOR1.
C
C OUTPUTS
C
C    CORR(I)   I=1,...,NLAGS CONTAINS THE CROSSCORRELATION
C                C(J) J=ILAG,...,ILAG+NLAGS-1  AS DEFINED IN THE
C                ABSTRACT.
C
C    IANS      =0  NORMALLY
C              =1  IF ILLEGAL LXX (LSTHN= 0)
C              =2  IF ILLEGAL LYY (LSTHN= 0)
C              =3  IF ILLEGAL MXACC (LSTHN=0, GRTHN 1000)
C              =4  IF ILLEGAL NLAGS (LSTHN= 0)
C              =5  IF ILLEGAL LSPACE (SEE ABOVE)
C              =24 IF ILLEGAL VALUE OF XX OR YY FOUND BY PROCOR
C                     (ABS(XX(I)) GRTHN MXACC).
C              =33 IF OVERFLOW OCCURS - SEE PROCOR WRITEUP.
C
C EXAMPLES
C
C 1. INPUTS  - LXX = 2  XX(1...2) = MLI 1,2
C              LYY = 3  YY(1...3) = MLI 5,4,3
C              NLAGS=5  CORR(1...5) = MLI 1,1,1,1,1
C              ILAG = 0  MXACC = 100  IAD = 0  LSPACE = 1050
C    OUTPUTS - IANS = 0 CORR(1...5) = MLI 13,10,0,0,0
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT ILAG = -4  IAD = 1
C    OUTPUTS - IANS = 0  CORR(1...5) = MLI 1,1,4,11,14
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION CM(2),CORR(2),SPACE(2)
      COMMON CM
C
C BRING IN SOME OF THE ARGUMENTS
C
      CALL SETKS  (LXX,LX, LYY,LY, ILAG,ILG, NLAGS,LZ)
      CALL IXCARG (XX,IX)
      CALL IXCARG (YY,IY)
C
C CHECK ERROR CONDITIONS
C
      CALL LIMITS (1,IAN, LX,1,32561, LY,1,32561, MXACC,1,1000,
     1 LZ,1,32561, LSPACE,1+XMINOF(LX,LY)+10*(MXACC+1),32561)
      IF (IAN) 5,5,900
 5    CONTINUE
C
C CLEAR THE OUTPUT AREA IF  IAD=0
C
      IF (IAD) 20,10,20
 10   CALL STZ    (LZ,CORR)
 20   CONTINUE
C
C CHOP OFF UNUSED PORTIONS FROM LEFT OF  XX  AND  YY.
C
      KMAX=ILG+LZ-1
      IF (ILG) 30,60,50
 30   CONTINUE
      IF (KMAX) 40,60,60
 40   CALL SETKS (LY+KMAX,LY, IY-KMAX,IY, ILG-KMAX,ILG, 0,KMAX)
      IF (LY) 900,900,60
 50   CALL SETKS (LX-ILG,LX, IX+ILG,IX, 0,ILG, LZ-1,KMAX)
      IF (LX) 900,900,60
C     SET UP PARAMETERS
```

```
   60   CONTINUE
        LXMLY=LX-LY
  C     SET SWITCHES WHICH DEPEND ON LX-LY
        IF (LXMLY) 230,220,220
  220   CALL SETKS (LY,LYT, IY,IYI, IX,IXI)
        GO TO 240
  230   CALL SETKS (LX,LYT, IX,IYI, IY,IXI, -KMAX,ILG, -ILG,KMAX)
        CALL REVERS (LZ,CORR)
  C     SET ARGUMENTS WHICH DEPEND ON ILG,LZ
  240   CALL SETKS (-1,IF3, -1,IF2, XMAXOF(ILG,-LYT+1),NF12, 0,NF13)
        IF (LXMLY) 320,310,320
  310   NF13=XMINOF(LYT-1,KMAX)
        GO TO 390
  320   IF (KMAX)390,390,330
  330   NF23=XMINOF(XABSF(LXMLY)-1,KMAX-1)
        KMAX1=KMAX-NF23-1
        IF (NF23) 350,350,340
  340   IF2=1
        IF (KMAX1) 390,350,350
  350   IF3=1
        NF33=XMINOF(LYT-1,KMAX1)
  390   CONTINUE
        CALL PROCOR (CM(IYI),LYT,MXACC,SPACE(LSPACE),SPACE,ERR)
        IF (ERR) 910,500,910
  500   CONTINUE
        IZ=-ILG+1
        CALL FASCR1 (CM(IXI),NF12,NF13,CORR(IZ),ERR)
        IF (ERR) 920,510,920
  510   IF (IF2) 530,530,520
  520   CONTINUE
        IZ=IZ+1
        CALL FASEP1 (CM(IXI+1),0,NF23-1,CORR(IZ),ERR)
        IF (ERR) 920,530,920
  530   IF (IF3) 550,550,540
  540   IXI=IXI+NF23+1
        IZ=-ILG+NF23+2
        CALL FASCR1 (CM(IXI),0,NF33,CORR(IZ),ERR)
        IF (ERR) 920,550,920
  550   CONTINUE
        IF (LXMLY) 600,610,610
  600   CALL REVERS (LZ,CORR)
  610   CONTINUE
  900   IANS=IAN
        RETURN
  910   IANS=ERR+20.
        GO TO 550
  920   IANS=ERR+30.
        GO TO 550
        END
```

```
*     RDATA (SUBROUTINE)                3/15/65   LAST CARD IN DECK IS NO. 0395
*     LABEL
CRDATA
      SUBROUTINE RDATA(ITAPE,ITPCPY,IANS,SPACE)
C
C
C                    ----ABSTRACT----
C
C TITLE - RDATA
C        READ DATA IN GENERALIZED FORMAT
C
C              SUBROUTINE RDATA PROVIDES A SIMPLIFIED TECHNIQUE FOR
C              INPUTING SMALL AMOUNTS OF DATA.  RDATA COMPARES A
C              HOLLERITH NAME FOUND ON THE DATA CARD WITH NAMES IN ITS
C              CALLING SEQUENCE.  WHEN IT FINDS MATCHING NAMES IT THEN
C              INTERPRETS THE DATA THAT FOLLOWS AS FIXED, FLOATING, OR
C              OCTAL NUMBERS OR AS HOLLERITH INFORMATION AND STORES IN
C              LOCATIONS CORRESPONDING TO THE HOLLERITH ARGUMENT.
C              THE POSITION OF STORAGE IN VECTORS IS CONTROLLED BY
C              GIVING INDEX VALUES ON THE CARD.  IF NONE IS GIVEN, RDATA
C              PUTS THE FIRST VALUE IN THE FIRST LOCATION OF THE VECTOR.
C              RDATA SCANS CARDS (80 COLUMNS PER CARD) UNTIL IT
C              ENCOUNTERS THE WORD 'RETURN', THEN IT RETURNS CONTROL
C              TO THE MAIN PROGRAM.  IF RDATA ENCOUNTERS UNINTERPRETABLE
C              INFORMATION ON THE CARDS, AN ERROR FLAG IS SET.
C
C              IF THE USER DESIRES, RDATA WILL COPY, VERBATIM, EACH
C              CARD THAT IT INTERPRETS ON OUTPUT TAPE 2 .
C
C              RDATA REQUIRES A SPECIAL (IOH) ROUTINE THAT CAN INTERPRET
C              INPUT 'G' FORMATS.  SUCH A ROUTINE IS DISTRIBUTED BY
C              SHARE AS  I9 SI GIOH  NUMBER  1402.
C
C LANGUAGE    - FORTRAN II SUBROUTINE
C EQUIPMENT   - 709 OR 7090 (MAIN FRAME AND TAPE UNIT)
C STORAGE     - 645 REGISTERS
C SPEED       -
C AUTHOR      - R.A. WIGGINS   4/64
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - ARG,CMPRA,HVTOIV,INTHOL,IVTOHV,
C                                     IXCARG,RETURN,SETUP,STORE
C       AND FORTRAN SYSTEM ROUTINES - (FIL),(RTN),(STH),(TSH)
C
C FORTRAN USAGE
C     CALL RDATA(ITAPE,ITPCPY,IANS,SPACE, X1NAME,X1, ..., XNNAME,XN)
C
C
C INPUTS
C
C    ITAPE     LOGICAL INPUT TAPE NUMBER.
C              IS NOT CHECKED FOR VALIDITY.
C
C    ITPCPY    LOGICAL TAPE NUMBER THAT RDATA WILL COPY EACH DATA CARD
C                    ONTO AND ON WHICH IT WILL INDICATE CARD COLUMNS
C                    IN WHICH ERRORS OCCUR.
C              =  0  INDICATES NOTHING WILL BE PRINTED.
C
C    SPACE(I)  I=1...110 IS TEMPORARY STORAGE SPACE NEEDED BY RDATA.
C
C    X1NAME    LEFT ADJUSTED HOLLERITH WORD GIVING THE NAME OF A
C                VARIABLE, OR VECTOR, THAT DATA MAY BE STORED IN.
C    .
C    .
C    XNNAME    N-TH HOLLERITH NAME.
C
C    DATA CARDS ON TAPE ITAPE.  RDATA SEARCHES FOR A NAME THAT
C      CORRESPONDS TO X1NAME...XNNAME, THEN IT STORES SUBSEQUENT
C      INFORMATION UNTIL IT FINDS ANOTHER NAME.  THE INFORMATION MAY
C      BE OF THE FOLLOWING TYPES (EACH FIELD MUST BE SEPARATED BY AT
C      LEAST ONE SPACE, COMMA, OR EQUALS SIGN).
C
C         INDEX OF VECTOR - INDICATED BY PLACING PARENTHESES AROUND A
```

```
C              FIXED POINT NUMBER.  MULTIPLE PERIODS INSIDE THE
C              PARENTHESES (SEPARATED FROM THE INDEX BY AT LEAST ONE
C              SPACE) AND SUCCEEDING EQUALS SIGNS ARE IGNORED.
C
C          FIXED POINT NUMBERS - INDICATED BY THE ABSENCE OF A DECIMAL
C              POINT IN THE NUMBER.
C
C          FLOATING POINT NUMBERS - INDICATED BY THE PRESENCE OF A
C              DECIMAL POINT, OR BY E FORMAT.
C
C          OCTAL NUMBERS - INDICATED BY A FINAL 'O' (OH) FOLLOWING
C              12 OCTAL DIGITS.
C
C          HOLLERITH CHARACTERS - INDICATED BY 'NH' WHERE N IS THE
C              COUNT OF THE NUMBER OF CHARACTERS FOLLOWING 'H' TO BE
C              INTERPRETED.  IF THIS FIELD EXTENDS BEYOND THE END OF
C              THE CARD, IT IS TRUNCATED TO THE END.
C
C          'RETURN' - CAUSES RDATA TO RETURN CONTROL TO THE CALLING
C          PROGRAM.
C
C
C OUTPUTS
C
C    X1..XN    HAVE DATA STORED IN THEM ACCORDING TO DATA CARDS
C
C    IANS      =  0 IF ALL OK
C              = -1 IF THE NUMBER OF ARGUMENTS IS LSTHN 6 OR ODD
C              =  A POSITIVE COUNT OF THE NUMBER OF FIELDS THAT RDATA
C                   FOUND UNINTERPRETABLE, IF THESE OCCUR.
C
C
C EXAMPLES (DATA INDICATES A CARD ON TAPE ITAPE)
C
C 1. INPUTS  - ITAPE = 2  ITPCPY = 0
C    USAGE   -       CALL RDATA(ITAPE,ITPCPY,IANS,SPACE,1HX,X,1HJ,J)
C    DATA    - X 1 1. 000010000000O  6HABCDEF  J   5  RETURN
C    OUTPUTS - IANS = 0  X(1...4) = 1,1.,8,6HABCDEF    J=5
C
C 2. INPUTS  - SAME AS EXAMPLE 1.
C    USAGE   - SAME AS EXAMPLE 1.
C    DATA    -  X 0 0 0 0 0 0 (2) 2 (4 ...)= 4 (6) = 6H******  RETURN
C    OUTPUTS - IANS = 0  X(1...6) = 0,2,0,4,0,6H******
C
C 3. INPUTS  - SAME AS EXAMPLE 1.
C    USAGE   - SAME AS EXAMPLE 1.
C    DATA    -  K=13 J=6   RETURN
C    OUTPUTS - IANS = 1   J = 6
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION CM(2),ICM(2)
      COMMON CM,ICM
      EQUIVALENCE(CM,ICM),(NUM,XNUM)
C
C SETUP LOCATE TO HANDLE VARIABLE ARGUMENT COUNT
C
      CALL SETUP  (LOCALL,NARGS,XR1,XR2)
C
C CHECK IF ARGUMENT COUNT IS LEGAL
C
      IF (NARGS-4) 20,20,10
 10   CONTINUE
      IF (XMODF(NARGS,2)) 40,40,20
 20   CONTINUE
      IANS=-1
 30   CONTINUE
      CALL RETURN (LOCALL,XR1,XR2)
 40   CONTINUE
C
C SET UP THE INDICES W.R.T. COMMON OF VARIOUS VECTORS.
C
      CALL IXCARG (SPACE,IHOL)
      IHOLE=IHOL+13
```

RDATA (PAGE 3)


```
      IV1B=IHOLE+1
      IV1E=IV1B+5
      IV2B=IV1E+3
      IV2E=IV2B+80
      IANS=0
C
C GET NEXT CARD
C
 100  CONTINUE
      READ INPUT TAPE ITAPE,110,(CM(I),I=IHOL,IHOLE)
 110  FORMAT(14A6)
      ICARD=ICARD+1
      HLN=0.
      IV=IV2B-1
C
C COPY CARD VERBATIM IF ITPCPY EXCEEDS ZERO
C
      IF (ITPCPY) 130,130,120
 120  CONTINUE
      WRITE OUTPUT TAPE ITPCPY, 125, ICARD,(CM(I),I=IHOL,IHOLE)
 125  FORMAT(3X6HCARD (I5,7H )  = '13A6,A2,1H')
 130  CONTINUE
C
C SCAN CARD. FIRST, SPREAD HOLLERITH - ONE LETTER PER WORD
C
      CALL HVTOIV (CM(IHOL),14,CM(IV2B))
C
C INITIALIZE SWITCHES - SCAN TO FIRST CHARACTER
C
 135  CONTINUE
      ASSIGN 550   TO KINDEX
 136  CONTINUE
      ASSIGN 190   TO KNND
      ASSIGN 160   TO KBLK
      ASSIGN 500   TO KNUM
      ASSIGN 400   TO KALPH
      ASSIGN 300   TO KPER
      ASSIGN 480   TO KLPRN
      ASSIGN 220   TO KE
      ASSIGN 220   TO KH
      ASSIGN 220   TO KO
      ASSIGN 135   TO KSTO
      ASSIGN 136   TO KALP1
C
C RESET COPY REGION TO BLKS
C
 140  CONTINUE
      IIV1=IV1B-1
      DO 150 I=IV1B,IV
 150  ICM(I)=48
C
C GET NEXT CHARACTER, CHECK IF CARD IS COMPLETED
C
 160  CONTINUE
      IV=IV+1
      IF (IV-IV2E) 180,180,100
 180  CONTINUE
      IVT=ICM(IV)
      ICM(IV)=48
      GO TO KNND ,(190,570)
C
C BRANCH ON CHARACTER TYPE
C
 190  CONTINUE
      IVI=IVT+1
C    0          0   1   2   3   4   5   6   7
      GO TO (210,210,210,210,210,210,210,210,
C    1          8   9 ILL   =   - ILL ILL ILL
     1         210,210,800,200,210,800,800,800,
C    2          +   A   B   C   D   E   F   G
     2         210,220,220,220,220,250,220,220,
C    3          H   I ILL   .   ) ILL ILL ILL
     3         260,220,800,230,200,800,800,800,
C    4          -   J   K   L   M   N   O   P
     4         210,220,220,220,220,220,270,220,
```

```
C     5          Q    R ILL    $    * ILL ILL ILL
     5          220,220,800,220,220,800,800,800,
C     6          BLK   /   S    T    U    V    W    X
     6          200,220,220,220,220,220,220,220,
C     7            Y    Z ILL    ,    ( ILL ILL ILL
     7          220,220,800,200,240,800,800,800), IVI
C BLANK CHARACTER ' ,)='
 200  GO TO KBLK, (160,320,420,520)
C NUMERICAL CHARACTER '+-0123456789'
 210  GO TO KNUM, (500,220,510)
C GENERAL ALPHABETIC 'ABCDFGIJKLMNPQRSTUVWXYZ$*/'
 220  GO TO KALPH,(400,410,800)
C PERIOD '.'
 230  GO TO KPER ,(300,160,200,210)
C LEFT PARENTHESIS '('
 240  GO TO KLPRN,(480,800)
C EEE 'E'
 250  GO TO KE    ,(220,210)
C AITCH 'H'
 260  GO TO KH    ,(220,750)
C OH 'O'
 270  GO TO KO    ,(220,700)
C
C SPECIAL ENTRY TO SKIP MULTIPLE PERIODS
C
 300  CONTINUE
      IF (ICM(IV+1)-27) 210,310,210
 310  CONTINUE
      ASSIGN 320 TO KBLK
      ASSIGN 160 TO KPER
      GO TO 160
 320  CONTINUE
      ASSIGN 160 TO KBLK
      ASSIGN 300 TO KPER
      GO TO 160
C
C FIRST CHARACTER IS ALPHABETIC, SCAN TO END
C
 400  CONTINUE
      ASSIGN 420   TO KBLK
      ASSIGN 410   TO KALPH
      ASSIGN 220   TO KNUM
      ASSIGN 200   TO KPER
      ASSIGN 420   TO KALP1
 410  CONTINUE
      IIV1=IIV1+1
      ICM(IIV1)=IVT
      GO TO 160
C
C END FOUND, CONVERT FIRST SIX CHARACTERS TO HOLLERITH
C
 420  CONTINUE
      CALL IVTOHV (CM(IV1B),1,HLN)
C
C IF HLN = 6HRETURN, LEAVE
C
      IF (CMPRAF(HLN,6HRETURN))430,30,430
C
C OTHERWISE, LOOK FOR HLN IN CALLING SEQUENCE
C
 430  CONTINUE
      DO 440 IARG=5,NARGS,2
      IF (CMPRAF(HLN,ARGF(LOCALL,IARG,1))) 440,450,440
 440  CONTINUE
C
C HLN CANNOT BE IDENTIFIED, GO TO ERROR PROCEDURE
C
      HLN=0.
      GO TO 800
C
C HLN IS NOW DEFINED BY IARG, GO SCAN FOR VALUES
C
 450  CONTINUE
      IARG=IARG+1
      IX=0
```

```
          GO TO 136
      C
      C FIRST CHARACTER IS LEFT PAREN
      C
       480  CONTINUE
            ASSIGN 540   TO KINDEX
            GO TO KALP1, (136,420)
      C
      C FIRST CHARACTER IS NUMERICAL, SCAN ACROSS
      C
       500  CONTINUE
            ASSIGN 520 TO KBLK
            ASSIGN 510 TO KNUM
            ASSIGN 800 TO KALPH
            ASSIGN 210 TO KPER
            ASSIGN 800 TO KLPRN
            ASSIGN 210 TO KE
            ASSIGN 750 TO KH
            ASSIGN 700 TO KO
       510  CONTINUE
            IIV1=IIV1+1
            ICM(IIV1)=IVT
            GO TO 160
      C
      C END OF NUMBER FOUND, CONVERT IT
      C
       520  CONTINUE
            FMT=3H(G)
       530  CONTINUE
            NHOL=(IIV1-IV1B+6)/6
            CALL IVTOHV (ICM(IV1B),NHOL,CM(IHOL))
            CALL INTHOL (NHOL,CM(IHOL),FMT,1,IDUM,NUM)
            GO TO KINDEX,(540,550,560)
      C
      C IF THIS IS AN INDEX, RESET IX
      C
       540  CONTINUE
            IX=NUM-1
            GO TO 135
      C
      C IF THIS IS OCTAL, OR NUMBER, STORE IT
      C
       550  CONTINUE
            IF (HLN) 555,556,555
       555  CONTINUE
            IX=IX+1
            CALL STORE (NUM,LOCALL,IARG,IX)
       556  CONTINUE
            GO TO KSTO, (135,140)
      C
      C IF THIS IS HOLLERITH, STORE NEXT NUM CHARACTERS
      C
       560  CONTINUE
            NUMH=NUM
            NUM1=XMINOF(NUM,IV2E-IV-1)
            FMT=4H(A6)
            IV2ET=IV+NUM1
            ASSIGN 570  TO KNND
            ASSIGN 550  TO KINDEX
            ASSIGN 140  TO KSTO
            GO TO 140
       570  CONTINUE
            IIV1=IIV1+1
            ICM(IIV1)=IVT
            IF (IV-IV2ET) 580,590,590
       580  IF (IIV1-IV1E) 160,530,530
       590  CONTINUE
            IF (IIV1-IV1B) 135,600,600
       600  CONTINUE
            ASSIGN 135 TO KSTO
            GO TO 530
      C
      C THIS IS AN OCTAL NUMBER
      C
       700  CONTINUE
```

```
      FMT=5H(012)
      GO TO 530
C
C THIS IS HOLLERITH
C
 750  CONTINUE
      ASSIGN 560 TO KINDEX
      GO TO 520
C
C ERROR FOUND, SCREAM, BUMP IANS, AND CONTINUE
C
 800  CONTINUE
      IANS=IANS+1
      IF (ITPCPY)  830,830,810
 810  CONTINUE
      INDEX=IV-IV2B+1
      WRITE OUTPUT TAPE ITPCPY, 820, INDEX
 820  FORMAT(42H  ILLEGAL CARD FORMAT BEGINNING IN COLUMN I4)
 830  CONTINUE
      GO TO 135
      END
```

```
************************
*    REFIT             *
************************
 REFER TO
     SPLIT
```



```
************************
*    REFIT             *
************************
 REFER TO
     SPLIT
```

```
*      REFLEC (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0107
*      FAP
*REFLEC
       COUNT   100
       LBL     REFLEC
       ENTRY   REFLEC ( X, LX,XMIROR,XIMAGE)
       ENTRY   XRFLEC (IX,LIX,IXMIRR,IXIMGE)
*
*                   ----ABSTRACT----
*
* TITLE - REFLEC  WITH SECONDARY ENTRY XRFLEC
*         REFLECT A FIXED OR FLOATING VECTOR THROUGH A CONSTANT
*
*              REFLEC SETS A FLOATING VECTOR EQUAL TO A CONSTANT MINUS
*              A GIVEN FLOATING VECTOR. OUTPUT CAN REPLACE INPUT.
*
*              XRFLEC DOES THE SAME THING FOR FIXED VECTORS.
*
* LANGUAGE    - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 28 REGISTERS
* SPEED       - REFLEC   34 +  12.4*L   MACHINES CYCLES,
*               XRFLEC   36 +   8.0*L      L= VECTOR LENGTH
* AUTHOR      - S.M. SIMPSON, SEPTEMBER 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL REFLEC( X, LX,XMIROR,XIMAGE)
*     CALL XRFLEC(IX,LIX,IXMIRR,IXIMGE)
*
* INPUTS
*
*    X(I)       I=1...LX   IS A FLOATING VECTOR
*    LX         SHOULD  EXCEED 0
*    XMIROR     IS A FLOATING CONSTANT
*
*    IX(I)      I=1...LIX  IS A FIXED VECTOR
*    LIX        SHOULD EXCEED 0
*    IXMIRR     IS A FIXED CONSTANT
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUTS IF LX OR LIX LSTHN 1
*
*    XIMAGE(I) I=1...LX  IS  XIMAGE(I) = XM - X(I)
*                   WHERE XM = INPUT VALUE OF XMIROR  (XMIROR WILL BE
*                   AN OUTPUT IF BOTH PERMISSIBLE EQUIVALENCES BELOW ARE
*                   USED)
*
*    IXIMGE(I) I=1...LIX  IS  IXIMGE(I) = IXM - IX(I)
*                   WHERE  IXM = INPUT VALUE OF IXMIRR
*
*              EQUIVALENCE(XMIROR,SOME X(I)),(XIMAGE,X) ARE
*              PERMITTED. SIMILARLY FOR THE FIXED POINT QUANTITIES.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)  = 1., 2., 3., 4.    IX(1...4) = 1,2,3,4
*              XR2 = -999.
*    USAGE   -       CALL REFLEC(  X, 4, 5.,  XR1)
*                    CALL XRFLEC( IX, 4,  0, IXR1)
*                    CALL REFLEC(  X, 4, 5.,    X)
*                    CALL XRFLEC( IX, 4, IX(3),IX)
*                    CALL REFLEC(  X, 0, 5.,  XR2)
*    OUTPUTS - XR1(1...4) =  4., 3., 2., 1.   IXR1(1...4) = -1,-2,-3,-4
*                X(1...4) =  4., 3., 2., 1.     IX(1...4) =  2, 1, 0,-1
*              XR2 = -999.  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
```

****
*   REFLEC   *
****
(PAGE 2)

```
       BCI     1,REFLEC
* PRINCIPAL ENTRY.  REFLEC (X,LX,XMIROR,XIMAGE)
REFLEC CLA     FSB
 SETUP STO     SUBTR
       SXD     REFLEC-2,4
 K1    CLA     1,4
       ADD     K1             A(X)+1
       STA     SUBTR
       CLA     4,4
       ADD     K1             A(XIMAGE)+1
       STA     STORE
       CLA*    3,4            XMIROR
       STO     MIROR
       CLA*    2,4            LX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* REFLECTING LOOP
 GET   CLA     MIROR
 SUBTR NOP                    FSB **,4 OR SUB **,4  **=A(X)+1
 STORE STO     **,4           **=A(XIMAGE)+1
       TIX     GET,4,1
* EXIT
 LEAVE LXD     REFLEC-2,4
       TRA     5,4
* SECOND ENTRY. XRFLEC(IX,LIX,IXMIRR,IXIMGE)
XRFLEC CLA     SUB
       TRA     SETUP
* CONSTANTS, TEMPORARIES
 FSB   FSB     **,4
 SUB   SUB     **,4
 MIROR PZE     **,**,**       =XMIROR
       END
```

```
************************
*    REIM              *
************************
 REFER TO
     AMPHZ
```


```
************************
*    REIM              *
************************
 REFER TO
     AMPHZ
```

```
*       REMAV  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0105
*       FAP
*REMAV
        COUNT   100
        LBL     REMAV
        ENTRY   REMAV (X,LX,XAVG,XNULD)
*
*                     ----ABSTRACT----
*
* TITLE - REMAV
*         REMOVE THE MEAN FROM A FLOATING VECTOR
*
*               REMAV COMPUTES THE AVERAGE VALUE OF A FLOATING VECTOR,
*               THEN SETS AN OUTPUT VECTOR WITH ELEMENTS EQUAL TO THOSE
*               OF THE INPUT VECTOR MINUS THE AVERAGE.  THE OUTPUT
*               VECTOR MAY REPLACE THE INPUT VECTOR.  THE AVERAGE IS
*               ALSO AN OUTPUT QUANTITY.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 36 REGISTERS
* SPEED      - 67.4 + 20.8*L  MACHINE CYCLES ON 7090,  L = VECTOR LENGTH
*              72.4 + 20.8*L  MACHINE CYCLES ON 709
* AUTHOR     - S.M. SIMPSON, SEPTEMBER 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL REMAV (X,LX,XAVG,XNULD)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLOATING VECTOR.
*
*    LX         SHOULD EXCEED ZERO.
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LX LSTHN 1.
*
*    XAVG       IS  (1/LX) * (SUM (FROM I=1 TO LX) OF X(I)).
*
*    XNULD(I)   I=1...LX   IS   XNULD(I) = X(I) - XAVG.
*
*               EQUIVALENCE(X,XNULD) IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5) = 1., 2., 3., 4., 5.      XAVG4 = Z = -999.
*    USAGE   -        CALL REMAV( X, 5, XAVG1, XNULD)
*                     CALL REMAV( X, 2, XAVG2, X)
*                     CALL REMAV( X, 1, XAVG3, Y)
*                     CALL REMAV( X, 0, XAVG4, Z)
*    OUTPUTS - XAVG1 = 3.   XNULD(1...5) = -2., -1., 0., 1., 2.
*              XAVG2 = 1.5      X(1...2) = -.5, .5
*              XAVG3 = -.5    Y = 0.
*              XAVG4 = Z = -999.  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,REMAV
* ONLY ENTRY.  REMAV (X,LX,XAVG,XNULD)
 REMAV  SXD     REMAV-2,4
 K1     CLA     1,4
        ADD     K1             A(X)+1
        STA     ADD1
        STA     GET
        CLA     4,4
        ADD     K1             A(XNULD)+1
        STA     STORE
        CLA     3,4            A(XAVG)
```

```
        STA    FSB
 * CHECK LS AND FLOAT IT.
        CLA*   2,4            LX
        TMI    LEAVE
        PDX    0,4
        TXL    LEAVE,4,0
        LRS    18
        ORA    OCTK
        FAD    OCTK           FLOATED LX
        STO    FLX
 * SUM X(1...LX), DIVIDE, STORE.
        PXD    0,0
  ADD1  FAD    **,4           **=A(X)+1
        TIX    ADD1,4,1
        FDP    FLX
        LXD    REMAV-2,4
        STQ*   3,4            XAVG
 * MEAN REMOVAL LOOP
        CLA*   2,4            LX
        PDX    0,4
  GET   CLA    **,4           **=A(X)+1
  FSB   FSB    **             **=A(XAVG)
  STORE STO    **,4           **=A(XNULD)+1
        TIX    GET,4,1
 * EXIT
  LEAVE LXD    REMAV-2,4
        TRA    5,4
 * CONSTANTS, VARIABLES
  OCTK  OCT    233000000000
  FLX   PZE    **             = LX FLOATED
        END
```

```
*       REREAD (SUBROUTINE)                 9/9/64   LAST CARD IN DECK IS NO. 0282
*       FAP
*REREAD
        COUNT   150
        LBL     REREAD
        ENTRY   REREAD
        ENTRY   EOFSET (ZIFTRN,EOF,ITAPE)
        ENTRY   ENDFIL (ITAPE)
        ENTRY   (TSH)     (TAPE TO STORAGE HOLLERITH)
        ENTRY   (TSHM)    (TAPE TO STORAGE HOLLERITH-MONITOR)
*
*
*                       ----ABSTRACT----
*
* TITLE - REREAD, WITH SECONDARY ENTRIES EOFSET, ENDFIL, (TSH), (TSHM)
*        REREAD DATA RECORD AND END FILE MONITOR
*
*               REREAD IS A MODIFICATION OF THE FORTRAN-II BCD TAPE
*               READING ROUTINE (TSH) THAT ALLOWS THE USER GREATER
*               FLEXIBILITY IN REINTERPRETATION OF CARDS AND IN THE
*               SELECTION OF PROGRAMMED REACTION TO READING END-OF-FILE
*               MARKS.
*
*               THE REREAD ENTRY ALLOWS THE REINTERPRETATION OF A CARD
*               AS MANY TIMES AS THE USER DESIRES WITHOUT ACTUALLY
*               REREADING THE INPUT TAPE.
*
*               THE EOFSET ENTRY ALLOWS THE SELECTION OF A REACTION TO
*               THE ENCOUNTER OF AN END-OF-FILE ON THE TAPE.  THE OPTIONS
*               AVAILABLE ARE 1) EXIT TO MONITOR CONTROL, 2) TRANSFER TO
*               A SPECIFIED POSITION IN THE MAIN PROGRAM, OR 3) SETTING
*               OF A FLAG WHICH THE USER MAY CHECK IF HE WISHES.
*
*               IF NEITHER OF THE SPECIAL OPTIONS ARE USED, THE PROGRAM
*               SIMPLY DUPLICATES THE FUNCTIONS OF (TSH) AND (TSHM).
*
* LANGUAGE    - FAP SUBROUTINES AND FUNCTION (FORTRAN II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME AND TAPE UNIT)
* STORAGE     - 114  REGISTERS
* SPEED       -
* AUTHOR      - R.A. WIGGINS    4/64
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (IOH),(RDS),(RDC),(RCH),(TCO),
*                                     (TEF),EXIT,(RER)
*
*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY REREAD
*
* FORTRAN USAGE OF REREAD
*     CALL REREAD
*
*               CAUSES THE NEXT 'READ INPUT TAPE' STATEMENT TO REINTER-
*               PRET THE LAST CARD READ.  WHEN USED, THESE READING STATE-
*               MENTS SHOULD READ ONLY ONE CARD.  REREAD MAY BE CALLED
*               AS MANY TIMES AS DESIRED FOR VARIED INTERPRETATIONS OF
*               THE CARD.
*
*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY EOFSET
*
* FORTRAN USAGE OF EOFSET
*     CALL EOFSET(ZIFTRN,EOF,ITAPE)
*
*               EOFSET INITIALIZES THE (TSH) SUBPROGRAM TO ESTABLISH THE
*               MODE OF REACTION TO AN END OF FILE ENCOUNTER.  MAY BE
*               RESET AS OFTEN AS DESIRED.
*
* INPUTS TO EOFSET
*
*    ZIFTRN     LSTHN 0 CAUSES (TSH) TO EXIT ON END-OF-FILE (STANDARD
*                 OPERATING MODE IF EOFSET IS NEVER CALLED).
*               = 0  CAUSES (TSH) TO TRANSFER TO THE FIRST STATEMENT
*                 FOLLOWING THIS 'CALL EOFSET' STATEMENT WITH AN ERROR
*                 FLAG (EOF=1.) WHEN AN END-OF-FILE IS ENCOUNTERED.
```

```
*              GRTHN 0 CAUSES (TSH) TO INTERPRET THE END-OF-FILE AS A
*                BLANK RECORD AND SETS AN END-OF-FILE FLAG WHICH USER
*                MAY CHECK AT WILL (USING ENDFIL).
*
* OUTPUTS FROM EOFSET
*
*    EOF       =0. AFTER EOFSET IS CALLED.
*              =1. IF ZIFTRN=0. IN THE LAST CALL OF EOFSET AND IF AN
*                END-OF-FILE IS ENCOUNTERED WHILE READING.
*
*    ITAPE     IS THE LOGICAL TAPE NUMBER THAT THE END-OF-FILE WAS
*                FOUND ON. (IS OUTPUTED ONLY WHEN EOF=1.).
*
*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ENTRY ENDFIL
*
* FORTRAN USAGE OF ENDFIL FUNCTION
*      EOF1 = ENDFIL(ITAPE)
*
*              ENDFIL IS USED TO CHECK IF AN END-OF-FILE WAS ENCOUNTERED
*              WHILE IN THE  ZIFTRN  GRTHN  0.  MODE (SEE EOFSET).
*
* OUTPUTS FROM ENDFIL
*
*    EOF1      =0. IF NO END-OF-FILE WAS ENCOUNTERED, OR IF EOFSET IS
*                NOT IN THE ZIFTRN=1. MODE.
*              =1. IF AN END-OF-FILE WAS ENCOUNTERED, AND IF EOFSET IS
*                IN THE ZIFTRN=1. MODE, AND IF THIS IS THE FIRST CALL OF
*                ENDFIL AFTER THE END-OF-FILE WAS ENCOUNTERED.
*
*    ITAPE     IS THE LAST LOGICAL TAPE NUMBER READ.
*
*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX COMPUTATIONAL EXAMPLES
*
* 1. INPUTS  - ITPEX = 5  KX(1...4) = 5,5,5,5   LX(1...4) = 5,5,5,5
*              RCHK = 0.  MX(1...4) = 5,5,5,5
*
*    USAGE   - C SET UP EXAMPLE TAPE
*                    REWIND ITPEX
*                    WRITE OUTPUT TAPE ITPEX, 10
*               10   FORMAT(8H 1 2 3 4)
*                    END FILE ITPEX
*                    END FILE ITPEX
*                    END FILE ITPEX
*                    REWIND ITPEX
*              C OPERATE ALL MODES
*              C 1. READ LINE IN I2 FORMAT
*                    READ INPUT TAPE ITPEX, 20, (IX(I),I=1,4)
*               20   FORMAT(4I2)
*              C 2. REINTERPRET IN I4 FORMAT
*                    CALL REREAD
*                    READ INPUT TAPE ITPEX, 30, (JX(I),I=1,2)
*               30   FORMAT(2I4)
*              C 3. REINTERPRET IN F FORMAT
*                    CALL REREAD
*                    READ INPUT TAPE ITPEX, 40, (X(I),I=1,4)
*               40   FORMAT(4F2.0)
*              C 4. SETUP EOFSET IN ZIFTRN=1. MODE AND ENCOUNTER END OF
*              C       FILE
*                    CALL EOFSET(1.,EOF1,ITAPE1)
*                    READ INPUT TAPE ITPEX, 20, (KX(I),I=1,4)
*                    EOF2 = ENDFIL (ITAPE2)
*                    EOF3 = ENDFIL (ITAPE3)
*              C 5. SET UP EOFSET IN ZIFTRN = 0. MODE AND ENCOUNTER NEXT
*              C       END OF FILE
*                    CALL EOFSET(0.,EOF4,ITAPE4)
*                    IF (EOF4) 50,50,60
*               50   READ INPUT TAPE ITPEX, 20, (LX(I),I=1,4)
*              C    PROGRAM NEVER REACHES HERE
*                    RCHK=1.
*              C    PROGRAM COMES HERE AFTER END OF FILE
*               60   CONTINUE
*              C 6. SET UP EOFSET IN ZIFTRN = -1. MODE AND ENCOUNTER
*              C       LAST END OF FILE.
*                    CALL EOFSET(-1.,EOF5,ITAPE5)
*                    READ INPUT TAPE ITPEX, 20, (MX(I),I=1,4)
```

```
*              C    PROGRAM EXITS
*
*    OUTPUTS - IX(1...4) = 1,2,3,4
*              JX(1...2) = 102,304
*               X(1...4) = 1.,2.,3.,4.
*              EOF1=0.  ITAPE1=0  KX(1...4)=-0,-0,-0,-0  EOF2=1.
*              EOF3=0.  ITAPE3=5  ITAPE2=5
*              EOF4=1.  ITAPE4=5  LX(1...4)=5,5,5,5  RCHK = 0.
*              EOF5=0.  ITAPE5=0  MX(1...4)=5,5,5,5
*
*
* PROGRAM FOLLOWS BELOW
*
 XR4   PZE     0
       BCI     1,REREAD
BUFSIZ EQU     22             BCD RECORD BUFFER SIZE
*
* READ INPUT TAPE ITAPE ENTRY IF NOT UNDER MONITOR CONTROL.
* LOGICAL TAPE NUMBER IS IN THE ACCUMULATOR.
*
 (TSH) STO     ITAPE          SAVE LOGICAL TAPE NUMBER.
       LDQ     NOP            PICKUP SWITCH SETTING, AND
       TRA*    $(IOH)        *GO INITIALIZE (IOH).
 NOP   NOP     TSH            INPUT / TAPE TO STORAGE HOLLERITH.
*
* READ INPUT TAPE ITAPE ENTRY IF UNDER MONITOR CONTROL.
*
(TSHM) STO     ITAPE          SAVE LOGICAL TAPE NUMBER.
       LDQ     NOP            PICKUP SWITCH SETTING, AND
       SLQ     TSHSW          SET MONITOR SWITCH, THEN
       TRA*    $(IOH)        *GO INITIALIZE (IOH).
*
* REENTRY FROM (IOH).
*
 TSH   SXA     TSHX,4         SAVE RETURN INDEX
 SWI1  NOP     RERE1          TRA RERE1 IF REREAD ENTRY
       XEC*    $(RDS)         SELECT CURRENT UNIT
       AXC     TSHC,4         INITIALIZE
       PXA     ,4             FOR
       STA*    $(RDC)         READ CHECKING.
       XEC*    $(RCH)         READ ONE TAPE RECORD.
 TSHSW TRA     RER           *EXIT TO CHECK READING, UNLESS
       LDQ*    $(TCO)         IN MONITOR MODE, THEN
       SLQ     TCO            GET TCO AND
       LDQ*    $(TEF)         TEF INSTRUCTIONS
       SLQ     TEF            FROM (IOS).
 TCO   TCOA    *              WAIT TO COMPLETE READING.
 TEF   TEFA    $EXIT         *EXIT (TO EXIT OR MAIN) IF END OF FILE.
 RER   TSX     $(RER),4      *GO CHECK READING ERROR.
       AXT     BUFSIZ,4       SAVE
       CLA     -1,4           INPUT
       STO     REC1+1,4       BUFFER
       TIX     *-2,4,1        IN REC1.
 TSHX  AXT     **,4           RESET IR4, AND
 TRA   TRA     1,4           *REENTER (IOH).
*
* THAT IS ALL OF THE READING OPERATION.
* REREAD ENTRY.
*
REREAD SXD     XR4,4          SAVE IR4.
       CLA     TRA            SET SWITCH
       STD     SWI1           SWI1, AND
       TRA     1,4           *RETURN TO MAIN.
*
* REENTRY FROM SWI1
*
 RERE1 AXT     BUFSIZ,4       RESTORE
       CLA     REC1+1,4       (IOH)
       STO     -1,4           BUFFER.
       TIX     *-2,4,1
       CLA     NOP            RESET SWITCH
       STD     SWI1           SWI1, AND
       TRA     TSHX          *RETURN TO (IOH).
*
```

```
* ENTRY TO DEFINE NEEDED ACTION IN CASE OF END-OF-FILE.
*
 XR4E  PZE     0              CONTAINS XR4 FOR RETURN ON EOF
EOFSET SXD     XR4E,4         SAVE IR4.
       STZ*    2,4            SET EOF TO ZERO, AND
       CLA*    1,4            CHECK DESIRED ACTION ON EOF ENCOUNTER.
       TZE     EOFZ
       TPL     EOFP
       CLA     EXAD           EOF=-X - EXIT ON EOF.
       TRA     STOE
 EOFZ  CLA     TRAD           EOF=0. - RETURN TO XR4E POSITION IN MAIN
       TRA     STOE             ON EOF.
 EOFP  CLA     COAD           EOF=+X - SET EOF FLAG.
 STOE  STA     TEF
       TRA     4,4           *RETURN TO MAIN.
*
* CONTROL COMES HERE ON END OF FILE IN NON-MONITOR MODES.
*
 TRAN  LXD     XR4E,4         PREPARE TO RETURN TO XR4E.
       CLA     =1.            SET
       STO*    2,4            EOF FLAG.
       CLA     ITAPE          SET TAPE NUMBER
       STO*    3,4            THAT EOF WAS ENCOUNTERED ON.
       TRA     4,4           *RETURN TO XR4E.
 CONTI AXT     BUFSIZ,4       SET
       CLA     =0606060606060 BUFFER
       STO     REC1+1,4       TO
       TIX     *-2,4,1        BLANKS.
       CLA     =1.            SET
       STO     EOF            EOF FLAG.
       TRA     RERE1         *RETURN TO (IOH) INDIRECTLY.
*
* END OF FILE FUNCTION ENTRY
*
ENDFIL CLA     ITAPE          GET
       STO*    1,4            TAPE NO.
       CLA     EOF
       STO     EOF1
       STZ     EOF
       CLA     EOF1
       TRA     2,4           *RETURN TO MAIN.
*
* CONSTANTS AND STORAGE NEEDED BY REREAD ET AL.
*
 EXAD  PZE     $EXIT
 TRAD  PZE     TRAN
 COAD  PZE     CONTI
 EOF1  PZE     0
 EOF   PZE     0
 ITAPE PZE     0
       BES     BUFSIZ-1
 REC1  PZE     0
*
* RECORD READING AND BUFFER DEFINITIONS
*
 TSHC  IORT    REC,,BUFSIZ    READ 1 20-WORD BCD RECORD.
       COMMON  -206+BUFSIZ
 REC   COMMON  1              INPUT BUFFER
       END
```

```
************************
*    RETURN            *
************************
 REFER TO
     LOCATE
```



```
************************
*    RETURN            *
************************
 REFER TO
     LOCATE
```

```
*       REVER  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0097
*       FAP
*REVER
        COUNT   100
        LBL     REVER
        ENTRY   REVER (X,LX,XREVD)
*
*                    ----ABSTRACT----
*
* TITLE - REVER
*        REVERSE A VECTOR ELSEWHERE OR IN PLACE
*
*              REVER REVERSES THE STORAGE ORDER OF A VECTOR.
*              OUTPUT MAY REPLACE INPUT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 30   REGISTERS
* SPEED      - 41 + 6*L   MACHINE CYCLES, IF L IS EVEN, L= VECTOR LENGTH
*              47 + 6*L   MACHINE CYCLES, IF L IS ODD
* AUTHOR     - S.M. SIMPSON, SEPT 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL REVER (X,LX,XREVD)
*
* INPUTS
*
*    X(I)      I=1...LX  IS A VECTOR IN ANY MODE
*
*    LX        SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUT IF  LX  LSTHN 1
*
*    XREVD(I)  I=1...LX IS  XREVD(1)= X(LX), XREVD(2)= X(LX-1), ETC
*
*              EQUIVALENCE (XREVD,X) IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...4) = 1,2,3,4
*    USAGE   -       CALL REVER( IX, 1, IXR1)
*                    CALL REVER( IX, 2, IXR2)
*                    CALL REVER( IX, 3, IXR3)
*                    CALL REVER( IX, 4, IXR4)
*    OUTPUTS - IXR1 = 1  IXR2(1...2)=  2,1   IXR3(1...3) = 3,2,1
*              IXR4(1...4) = 4,3,2,1
*
* 2. INPUTS  - X(1...3) = 1., 2., 3.     Y= -999.
*    USAGE   -       CALL REVER(X,0,Y)
*                    CALL REVER(X,3,X)
*    OUTPUTS - Y = -999. (NO OUTPUT CASE)    X(1...3) = 3.,2.,1.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0             XR1
        HTR     0             XR4
        BCI     1,REVER
* ONLY ENTRY.   REVER (X,LX,XREVD)
 REVER SXD      REVER-2,4
       SXD      REVER-3,1
 K1    CLA      1,4           A(X)
       STA      GETXH
       ADD      K1            A(X)+1
       STA      GETXL
       CLA      3,4           A(XREVD)
       STA      STOXL
       ADD      K1            A(XREVD)+1
       STA      STOXH
```

```
        CLA*    2,4             LX
        TMI     LEAVE
        ARS     1               LX/2
        PDX     0,1             TRUNCATED.
        ADD     KDHAF           (LX+1)/2
        PDX     0,4             TRUNCATED
        TXL     LEAVE,4,0
* EXCHANGE LOOP
*       XR1  STARTS AT LX/2 TRUNCATED, MOVES UP.
*       XR4  STARTS AT (LX+1)/2 TRUNCATED, MOVES DOWN.
 GETXH CLA      **,1            **=A(X)
 GETXL LDQ      **,4            **=A(X)+1
 STOXH STO      **,4            **=A(XREVD)+1
 STOXL STQ      **,1            **=A(XREVD)
       TXI      *+1,1,1
       TIX      GETXH,4,1
* EXIT
 LEAVE LXD      REVER-2,4
       LXD      REVER-3,1
       TRA      4,4
* CONSTANTS
 KDHAF OCT      000000400000
       END
```

```
*       REVERS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0076
*       FAP
*REVERS
        COUNT   60
        LBL     REVERS
        ENTRY   REVERS (LX,X)
*
*                    ----ABSTRACT----
*
* TITLE - REVERS
*         FAST REVERSE STORAGE ORDER OF A VECTOR
*
* LANGUAGE   - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 29 REGISTERS
* SPEED      - ABOUT  6*LX + 32  MACHINE CYCLES ON THE 7090
*                 WHERE LX IS THE LENGTH OF THE VECTOR.
* AUTHOR     - R.A. WIGGINS,    19/8/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -    NONE
*        AND FORTRAN SYSTEM ROUTINES -    NONE
*
* FORTRAN USAGE
*     CALL REVERS(LX,X)
*
* INPUTS
*
*    X(I)       I=1...LX IS A VECTOR OF NUMBERS (ANY MODE)
*
*    LX         IS FORTRAN II INTEGER
*               MUST BE GRTHN=1
*
* OUTPUTS
*
*    X(I)       I=1...LX SAME AS ABOVE ONLY REVERSED
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)=1,2,3,4  LX=4
*    OUTPUTS - X(1...4)=4,3,2,1
*
* 2. INPUTS  - X(1...5)=1.,2.,3.,4.,5.  LX=5
*    OUTPUTS - X(1...5)=5.,4.,3.,2.,1.
*
* 3. INPUTS  - X=1  LX=1
*    OUTPUTS - X=1
*
        PZE
        BCI     1,REVERS
REVERS  SXD     *-2,4
        SXA     ADR,1
        CLA     2,4
        ADD     =1B35
        STA     X
        STA     X+1
        STA     X+2
        STA     X+3
        CLA*    1,4
        PDX     ,4
        ARS     1
        PDX     ,1
        SUB     =1B17
        TMI     ADR-1
        STD     *+1
        TIX     *+1,4,**
X       CLA     **,1
        LDQ     **,4
        STO     **,4
        STQ     **,1
        TXI     *+1,4,1
        TIX     X,1,1
        LXD     REVERS-2,4
ADR     AXT     **,1
```

```
            TRA       3,4                                                           0075
            END                                                                     0076
```

```
*     RLSPR  (SUBROUTINE)              10/5/64   LAST CARD IN DECK IS NO. 0120
*     LABEL
CRLSPR
      SUBROUTINE RLSPR (LL,AA,RR,ALP)
C
C                     ----ABSTRACT----
C
C TITLE - RLSPR
C        REALIZABLE LEAST SQUARE PREDICTOR BY RECURSION, 1-DIMENSION
C
C              RLSPR INCREASES THE LENGTH OF A REALIZABLE, LEAST SQUARE
C              PREDICTION ERROR OPERATOR A(K,L) BY ONE.  THAT IS, GIVEN
C              THE VECTOR A(K,L) (K REFERS TO THE K-TH ELEMENT IN A
C              VECTOR OF L+1 ELEMENTS) THAT SATISFIES THE EQUATIONS
C
C                A(L,L)*R(0)   + ... + A(1,L)*R(L-1) + A(0,L)*R(L)  = 0
C
C                A(L,L)*R(1)   + ... + A(1,L)*R(L-2) + A(0,L)*R(L-1)= 0
C                .
C                .
C                .
C                A(L,L)*R(L-1) + ... + A(1,L)*R(0)   + A(0,L)*R(1)  = 0
C
C              WHERE A(0,L) IS CONSTRAINED TO BE 1, THEN RLSPR INCREASES
C              THE LENGTH OF A(K,L) SO THAT IT SATISFIES THE EQUATIONS
C
C                A(L+1,L+1)*R(0)+... + A(1,L+1)*R(L) + A(0,L+1)*R(L+1)=0
C                ETC.
C
C              IF R(K) REPRESENTS THE AUTOCORRELATION OF A TIME SERIES
C              X(T)
C
C                R(K) = EXPECTED VALUE (X(T+K),X(T))
C
C              THEN THE SET OF EQUATIONS ABOVE ARE THE NORMAL EQUATIONS
C              FOR THE PREDICTION ERROR OPERATOR
C
C                A(L,L)*X(T-L) +...+ A(1,L)*X(T-1)+A(0,L)*X(T)= EPS(T,L)
C
C              WHERE EPS(T,L) IS THE ERROR SERIES.
C              AS A MATTER OF TERMINOLOGY, WE DEFINE
C
C                A(L,L)*R(1) +...+ A(1,L)*R(L)+A(0,L)*R(L+1)= ALP(L+1,L)
C                A(1,L)*R(-L)+...+ A(1,L)*R(-1)+A(0,L)*R(0) = ALP(0,L)
C
C              WHERE ALP(0,L) IS THE COVARIANCE OF EPS(T,L).  THAT IS
C
C                ALP(0,L) = EXPECTED VALUE (EPS(T,L)*EPS(T,L)).
C
C              RLSPR RETURNS THE VALUE OF ALP(0,L+1).
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 142 REGISTERS
C SPEED     - ABOUT  .000071*L + .00040  SECONDS ON THE 7094 MOD 1 .
C AUTHOR    - R.A. WIGGINS     3/63
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FDOTR
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C       CALL RLSPR (LL,AA,RR,ALP)
C
C INPUTS
C
C    LL         IS THE LENGTH OF THE INPUT SERIES A. (EQUALS L+1)
C               MUST BE GRTHN=0
C
C    AA(I)      I=1,...,LL CONTAINS THE OPERATOR A(0,L) THROUGH A(L,L).
C
C    RR(I)      I=1,...,LL+1 CONTAINS THE AUTOCORRELATION VECTOR R(0)
C                 THROUGH R(L+1).
C
```

```
C     ALP       CONTAINS ALP(0,L) AS DEFINED IN THE ABSTRACT.
C
C OUTPUTS
C
C     LL        IS INCREASED ONE FROM THE INPUT VALUE.
C
C     AA(I)     I=1,...,LL (NEW LL) CONTAINS THE OPERATOR A(0,L+1)
C                 THROUGH A(L+1,L+1)
C
C     ALP       CONTAINS ALP(0,L+1),
C
C EXAMPLES
C
C 1. INPUTS  - LL=0  RR(1...5) = 1.25,.5,0.,0.,0.
C    OUTPUTS - AA(1)=1.   ALP=1.25
C
C 2. INPUTS  - SAME AS EXAMPLE 1.
C    USAGE   -         DO 10  I=1,5
C                      CALL RLSPR (LL,AA,RR,C)
C                   10 CONTINUE
C    OUTPUTS - AA(1...5) = 1.,-0.4985,0.2463,-0.1173,0.0469  ALP=1.0007
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),RR(10)
      L1=LL
      L2=L1+1
      IF(L1)  80,10,30
 10   AA(1)=1.
      ALP=RR(1)
      GO TO 70
 30   CALL FDOTR (L1,AA,RR(2),ALPL)
      AAL=-ALPL/ALP
      AA(L2)=0.
      J=L2
      LH=(L2+1)/2
 35   DO 40 I=1,LH
      AAT=AA(J)
      AA(J)=AA(J)+AAL*AA(I)
      IF (J-I) 60,60,38
 38   AA(I)=AA(I)+AAL*AAT
 40   J=J-1
 60   ALP=ALP+AAL*ALPL
 70   LL=L2
 80   RETURN
      END
```

```
*     RLSPR2  (SUBROUTINE)                9/9/64   LAST CARD IN DECK IS NO. 0280
*     LABEL
CRLSPR2
      SUBROUTINE RLSPR2 (NRA,NCAT,NCAN,AA,NRR,NCR,RR,C,IANS)
C
C                   ----ABSTRACT----
C
C TITLE - RLSPR2
C        REALIZABLE LEAST-SQUARE PREDICTOR BY RECURSION - 2 DIMENSIONS
C
C              RLSPR2 INCREASES THE LENGTH OF ONE DIMENSION OF NRA 2-
C              DIMENSIONAL LEAST SQUARE PREDICTION OPERATORS BY ONE,
C              WHERE NRA (NUMBER ROWS IN A) IS THE WIDTH OF THE OPERATOR
C              IN THE OTHER DIMENSION.  THAT IS, GIVEN THE PREDICTION
C              OPERATORS A(I,J,K) I=1,...,NRA, J=0,1,...,NCAN, K=1,...,
C              NRA  WHICH SOLVE THE EQUATIONS
C
C                NRA   NCAN
C                SUM ( SUM ( A(I,J,K)*R(I-M,J+N-1) ) ) = 0
C                I=1    J=0
C
C              FOR  K = 1,...,NRA
C                   M = 1,...,NRA
C                   N = 1,...,NCAN
C
C              WHERE                ( 1. IF I=K
C                      A(I,0,K) = (
C                                   ( 0. IF I NOT= K
C
C              THEN RLSPR2 INCREASES THE J/TH DIMENSION BY ONE SO THAT
C              THE EQUATIONS ARE SATISFIED FOR J=0,1,...,NCAN,NCAN+1.
C
C              IF R(I,J) I=-NRA,...,-1,0,1,...NRA, J=0,1,...NCAN
C              REPRESENTS ONE-HALF OF A TWO DIMENSIONAL AUTOCORRELATION
C
C                 R(K,L) = EXPECTED VALUE (X(I+K,J+L)*X(I,J))
C
C              THEN THE FIRST SET OF EQUATIONS ABOVE ARE THE NORMAL
C              EQUATIONS FOR THE PREDICTION ERROR OPERATOR
C
C                NRA   NCAN
C                SUM ( SUM ( A(I,J,K)*X(M-I,N-J) ) ) = EPS(M,N,K)
C                I=1    J=0
C
C              AS A SECONDARY OUTPUT, RLSPR2 RETURNS AN NRA X NRA MATRIX
C              C(L,K) THAT IS DEFINED BY
C
C                          NRA   NCAN
C                C(L,K) = SUM ( SUM ( A(I,J,K)*R(I-L,J) ) ).
C                          I=1    J=0
C
C              THE MATRIX CONTAINS THE COVARIANCE OF THE EXPECTED ERRORS
C              SQUARED
C
C                C(L,K) = EXPECTED VALUE ( EPS(M,N+L,K)**2 )
C
C              FOR L=1,...,NRA  K=1,...,NRA.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 700 REGISTERS
C SPEED      - ABOUT .00010*M*N**3 SECONDS ON THE 7094 MOD 1
C              FOR N GRTHN 7 AND M GRTHN 25 .
C AUTHOR     - R.A. WIGGINS      MAY, 1963   GEOSCIENCE, INC.
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - DOTJ,DOTP,IXCARG,MATML3,MOVREV,
C                                     SIMEQ,STZ
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C    CALL RLSPR2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,C,IANS)
C
```

```
      C INPUTS
      C
      C    NRA       NUMBER ROWS OF A.
      C              MUST BE GRTHN=1
      C
      C    NCAT      NUMBER COLUMNS OF A TOTAL, I.E. THIS IS THE UPPER LIMIT
      C                ON THE NUMBER OF COLUMNS OF A.
      C              MUST BE GRTHN=1
      C
      C    NCAN      NUMBER COLUMNS OF A NOW, I.E.  THIS IS THE PRESENT LENGTH
      C                OF THE A PREDICTORS.
      C              MUST BE GRTHN=0, LSTHN=NCAT
      C
      C    AA(L)     L=1,...,NRA*NCAT*NRA CONTAINS A(I,J,K) STORED AS FOLLOWS
      C                L=1...NRA                  CONTAINS  I=1...NRA,J=0,K=1
      C                 =NRA+1...2*NRA            CONTAINS  I=1...NRA,J=1,K=1
      C                  .
      C                  .                                              K=1
      C                 =(NCAN-1)*NRA+1...NCAN*NRA CONTAINS  I=1...NRA,J=NCAN
      C                 =NCAT*NRA+1...NCAT*NRA+NRA CONTAINS  I=1...NRA,J=0,K=2
      C                  .
      C                  .
      C                 =NCAT+(NCAN-1)*NRA+1...NCAN*NRA
      C                                        CONTAINS  I=1...NRA,J=NCAN,K=2
      C                ETC.
      C
      C    NRR       NUMBER ROWS OF R.
      C              MUST BE GRTHN=1 AND ODD.
      C
      C    NCR       NUMBER COLUMNS OF R.
      C              MUST BE GRTHN=1
      C
      C    RR(I)     I=1,...,NRR*NCR CONTAINS R(J,K) J=-NRR/2,...,-1,0,1,...,
      C                NRR/2  K=0,...,NCR-1 STORED CLOSELY SPACED.
      C
      C    C(I)      I=1,...,3*NRA*NRA+NRA IS COMPUTATION SPACE NEEDED BY
      C                RLSPR2.
      C
      C OUTPUTS
      C
      C    NCAN      IS INCREASED BY ONE
      C
      C    AA(L)     L=1,...,NRA*NCAT*NRA CONTAINS A(I,J,K), I=1,...,NRA,
      C                J=0,...,NCAN-1, K=1,...,NRA STORED WITH
      C                DIMENSION (NRA,NCAT,NRA).
      C
      C    C(I)      I=1,...,NRA*NRA CONTAINS THE ERROR MATRIX DEFINED IN THE
      C                ABSTRACT STORED CLOSELY SPACED BY COLUMNS.
      C
      C    IANS      =0 IF NO TROUBLE
      C              =1 IF NCAN GRTHN=NCAT ON ENTRANCE
      C              =2 IF NCAN LSTHN 0
      C              =3 IF OVERFLOW OCCURS DURING INVERSION OF MATRIX C.
      C              =4 IF MATRIX C IS SINGULAR (THEORETICALLY IMPOSSIBLE).
      C
      C
      C EXAMPLES
      C
      C 1. EXAMPLE OF A ONE-DIMENSIONAL AUTOCORRELATION
      C    INPUTS  - NRA = 1  NCAT = 5  NCAN = 0
      C              NRR = 1  NCR = 2  RR(1...2) = 1.25,.50
      C    USAGE   -      DO 10 I=1,NCAT
      C             10    CALL RLSPR2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,C,IANS)
      C    OUTPUTS - IANS = 0  AA(1...5) = 1.000,-0.499,0.246,-0.117,0.047
      C              C(1) = 1.001
      C
      C 2. EXAMPLE OF FIRST CALL OF RLSPR2
      C    INPUTS  - NRA = 1  NCAT = 5  NCAN = 0
      C              NRR = 3  NCR = 3  RR(1...9) = 0.302 0.105 0.010 (STORED
      C                                            1.340 0.621 0.202  BY
      C                                            0.302 0.105 0.010  COLUMNS)
      C    USAGE   -         CALL RLSPR2(NRA,NCAT,NCAN,AA,NRR,NCR,RR,C,IANS)
      C
      C    OUTPUTS - IANS = 0  AA(1...45) = 1.000  0.000  0.000  0.000  0.000
      C                (EACH  3 ROW BY      0.000  0.000  0.000  0.000  0.000
```

```
C                 5 COLUMN  ARRAY     0.000  0.000  0.000  0.000  0.000
C                 IS STORED CLOSELY
C                 SPACED BY COLUMNS.  0.000  0.000  0.000  0.000  0.000
C                 THE ARRAYS ARE      1.000  0.000  0.000  0.000  0.000
C                 ALSO CLOSELY        0.000  0.000  0.000  0.000  0.000
C                 SPACED.)
C                                     0.000  0.000  0.000  0.000  0.000
C                                     0.000  0.000  0.000  0.000  0.000
C                                     1.000  0.000  0.000  0.000  0.000
C
C              C(1...9) = 1.340  0.302  0.000
C                         0.302  1.340  0.302
C                         0.000  0.302  1.340
C
C 3. GENERAL EXAMPLE.
C    INPUTS  - SAME AS EXAMPLE 2.
C    USAGE   - SAME AS EXAMPLE 1.
C    OUTPUTS - IANS = 0  AA(1...45) = 1.000 -0.507  0.051  0.079 -0.046
C                                     0.000  0.032  0.015 -0.037  0.019
C                                     0.000 -0.008 -0.004  0.011 -0.006
C
C                                     0.000  0.032  0.015 -0.037  0.019
C                                     1.000 -0.515  0.048  0.090 -0.052
C                                     0.000  0.032  0.015 -0.037  0.019
C
C                                     0.000 -0.008 -0.004  0.011 -0.006
C                                     0.000  0.032  0.015 -0.037  0.019
C                                     1.000 -0.507  0.051  0.079 -0.046
C
C              C(1...9) = 1.039  0.271 -0.002
C                         0.271  1.037  0.271
C                        -0.002  0.271  1.039
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(2),RR(2),C(2),CM(2)
      COMMON CM
      L=NRA
      M=NCAN
      MT=NCAT
      LL=L*L
      LLMT=LL*MT
      LMT=L*MT
      LM=L*M
      CALL IXCARG (C,IC1)
      IC2=IC1+LL
      IC3=IC2+XMAXOF(LL,L+L)
      IC4=IC3+LL
      CALL IXCARG (AA,IA)
      LH=(L+1)/2
      LHL=LH*L
      L1=L+1
      M1=(NRR+1)/2
      IAN=0
      IF (MT-M)  10,10,20
 10   IAN=1
      GO TO 1000
 20   CONTINUE
      IF (M)  30,40,100
 30   IAN=2
      GO TO 1000
C SPECIAL CASE - M=0
 40   CONTINUE
      CALL STZ (LLMT,AA)
      CALL MOVREV(L,0,1.,LMT+1,AA,1)
      JC1=IC1
      JC2=IC1
      IR=M1
      CALL STZ (LL,C)
      DO 60  I1=1,L
      IF (IR)  70,70,50
 50   CONTINUE
      LMO=L-I1+1
      CALL MOVREV (LMO,0,RR(IR), L1,CM(JC1),1)
```

*   RLSPR2   *
****
(PAGE 4)

```
      CALL MOVREV (LMO,0,RR(IR), L1,CM(JC2),1)
      IR=IR-1
      JC1=JC1+1
   60 JC2=JC2+L
   70 CONTINUE
      GO TO 170
C GENERAL CASE
  100 CONTINUE
C FIND ZETA MATRIX
      JC1=IC3
      DO 120  I1=1,LLMT,LMT
      DO 110  I2=1,L
      CALL DOTP (L,M,AA(I1),NRR,NCR,RR,M1-I2,1,CM(JC1),-2.)
  110 JC1=JC1+1
  120 CONTINUE
C FIND K MATRIX
      CALL MATML3 (L,L,L,CM(IC2),CM(IC3),0,CM(IC4),0)
 1001 Z=1.
C FORM THE LENGTHENED PREDICTORS.
      LMH=(LM+L+1)/2
      KC1=IC2
      KC2=IC2+L
      IA1=1
      IA2=LM+L
      DO 160  I3=1,LMH
      CALL MOVREV (L,LMT,AA(IA1),1,CM(KC1),1)
      CALL MOVREV (L,LMT,AA(IA2),1,CM(KC2),1)
      JC1=KC1
      JC2=KC2
      KK1=IC4
      DO 150  I4=1,L
      CALL DOTJ (L,1,CM(KK1),LMT,AA(IA2),CM(JC1),1,1)
      CALL DOTJ (L,1,CM(KK1),LMT,AA(IA1),CM(JC2),1,1)
      KK1=KK1+L
      JC1=JC1+1
  150 JC2=JC2+1
      CALL MOVREV (L,1,CM(KC1),LMT,AA(IA1),1)
      CALL MOVREV (L,1,CM(KC2),LMT,AA(IA2),1)
      IA1=IA1+1
  160 IA2=IA2-1
C FORM NEW ALPHA MATRIX
      CALL MATML3 (L,L,L,CM(IC3),CM(IC4),0,CM(IC1),1)
C FIND INVERSE OF ALPHA
  170 CONTINUE
      CALL MOVREV (LL,1,CM(IC1),1,CM(IC2),-1)
      CALL STZ (LL,CM(IC3))
      CALL MOVREV (L,0,1., L1,CM(IC3),1)
      D=1.
      CALL SIMEQ (L,L,L,CM(IC2),CM(IC3),D,CM(IC4),ERR)
      IF (ERR)  190,190,180
  180 IAN=ERR+2.
      GO TO 1000
  190 CONTINUE
      NCAN=M+1
 1000 IANS=IAN
      RETURN
      END
```

```
*     RLSSR  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0114
*     LABEL
CRLSSR
      SUBROUTINE RLSSR (LL,AA,RR,GG,FF,ALP)
C
C                    ----ABSTRACT----
C
C
C TITLE - RLSSR
C        REALIZABLE LEAST SQUARE SHAPER BY RECURSION
C
C              RLSSR INCREASES THE LENGTH OF A REALIZABLE LEAST SQUARE
C              SHAPER FILTER F(K,L) BY ONE.  THAT IS, GIVEN THE VECTOR
C              F(K,L) (K REFERS TO THE K-TH ELEMENT IN A VECTOR OF
C              LENGTH L) THAT SATISFIES THE EQUATION
C
C                F(L,L)*R(0)    + ... + F(1,L)*R(L-1)  =  G(L-1)
C
C                F(L,L)*R(-1)   + ... + F(1,L)*R(L-2)  =  G(L-2)
C                .
C                .
C                F(L,L)*R(-L+1)+ ... + F(1,L)*R(0)     =  G(0)
C
C              AND A(K,L) AND ALP(0,L) AS GIVEN BY RLSPR
C              THEN RLSSR INCREASES THE LENGTH OF F(K,L) SO THAT
C              IT SATISFIES THE EQUATIONS
C
C                F(L+1,L+1)*R(0) + ... + F(1,L+1)*R(L) = G(L)
C                ETC.
C
C              IF R(K) REPRESENTS THE AUTOCORRELATION OF A WAVELET X(T)
C
C                R(K) = SUM (X(T+K)*X(T))
C
C              AND G(K) REPRESENTS THE CROSSCORRELATION OF A DESIRED
C              OUTPUT D(T) WITH THE WAVELET X(T)
C
C                G(K) = SUM (D(T)*X(T-K))
C
C              THEN THE FIRST SET OF EQUATIONS ABOVE ARE THE NORMAL
C              EQUATIONS FOR A SHAPER FILTER
C
C                D(T) - (F(L,L)*X(T-L) + ... + F(1,L)*X(T-1) = ZET(T,L)
C
C              WHERE ZET(T,L) IS THE ERROR SERIES.
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 82 REGISTERS
C SPEED     - ABOUT  .000162*L + .00011  SECONDS ON THE 7094 MOD 1 .
C AUTHOR    - R.A. WIGGINS     3/63
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FDOTR
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL RLSSR (LL,AA,RR,GG,FF,ALP)
C
C INPUTS
C
C    LL         =L+1 THE OUTPUT LENGTH OF THE SERIES F.
C               MUST BE GRTHN=1
C
C    AA(I)      I=1,...,LL CONTAINS THE PREDICTION ERROR OPERATOR A(0,L)
C                 THROUGH A(L,L).
C
C    RR(I)      I=1,...,LL CONTAINS THE AUTOCORRELATION VECTOR R(0)
C                 THROUGH R(L).
C
C    GG(I)      I=1,...,LL CONTAINS THE CROSSCORRELATION VECTOR G(0)
C                 THROUGH G(L).
C
C    FF(I)      I=1,...,LL-1 CONTAINS THE SHAPER FILTER F(1,L) THROUGH
```

```
C                F(L,L).
C
C    ALP       CONTAINS THE ERROR COVARIANCE ALP(0,L).
C
C OUTPUTS
C
C    FF(I)     I=1,...,LL CONTAINS THE NEW SHAPER FILTER F(1,L+1)
C                THROUGH F(L+1,L+1).
C
C EXAMPLES
C
C 1. INPUTS  - LL=0  RR(1...5)=1.25,.5,0.,0.,0.
C                    GG(1...5) = 1.,0.,0.,0.,0.
C    USAGE   -       CALL RLSPR (LL,AA,RR,ALP)
C                    CALL RLSSR (LL,AA,RR,GG,FF,ALP)
C    OUTPUTS - LL=1  AA(1)=1.  ALP=1.25  FF(1)=.8
C
C 2. INPUTS  - SAME AS EXAMPLE 1.
C    USAGE   -       DO 10  I=1,5
C                    CALL RLSPR (LL,AA,RR,ALP)
C                    CALL RLSSR (LL,AA,RR,GG,FF,ALP)
C                 10 CONTINUE
C    OUTPUTS - AA(1...5) = 1.000, -0.498, 0.246, -0.117, 0.047
C              FF(1...5) =  .999, -0.498, 0.246, -0.117, 0.047
C              LL=5  ALP=1.00073
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION AA(10),RR(10),GG(10),FF(10)
      L2=LL
      L1=L2-1
      CALL FDOTR (L1,FF,RR(2),C1)
      FL=(GG(L2)-C1)/ALP
      FF(L2)=0.
      J=L2
      DO 10  I=1,L2
      FF(I) = FF(I)+FL*AA(J)
   10 J=J-1
      RETURN
      END
```

```
*************************
*    RMSDAV             *
*************************
 REFER TO
     RMSDEV
```



```
*************************
*    RMSDAV             *
*************************
 REFER TO
     RMSDEV
```

```
*      RMSDEV (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0159
*      FAP
*RMSDEV
       COUNT   150
       LBL     RMSDEV
       ENTRY   RMSDEV (X,LX,XBASE,RMSXMB)
       ENTRY   RMSDAV (X,LX,XAVG,RMSXMA)
*
*
*                    ----ABSTRACT----
*
* TITLE - RMSDEV WITH SECONDARY ENTRY RMSDAV
*        R.M.S. DEVIATION FROM GIVEN BASE OR FROM TRUE AVERAGE
*
*              RMSDEV COMPUTES THE ROOT MEAN SQUARE VALUE OF THE
*              DEVIATIONS, FROM A GIVEN BASE, OF THE ELEMENTS OF
*              A VECTOR.
*
*              RMSDAV COMPUTES THE AVERAGE OF A VECTOR AND THEN
*              THE RMS VALUE OF THE DEVIATIONS OF ITS ELEMENTS
*              AROUND THE AVERAGE.  THE AVERAGE IS AN ADDITIONAL
*              OUTPUT.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 50 REGISTERS
* SPEED      - ON THE 7090
*                 RMSDEV TAKES ABOUT  70 + 33.8*L + K MACHINE CYCLES
*                 RMSDAV TAKES ABOUT 100 + 43.2*L + K MACHINE CYCLES
*                    WHERE L = VECTOR LENGTH
*                          K = CYCLES FOR ONE SQUARE ROOT
* AUTHOR     - S.M.SIMPSON, FEBRUARY 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - SQRT
*
* FORTRAN USAGE OF RMSDEV
*     CALL RMSDEV(X,LX,XBASE,RMSXMB)
*
* INPUTS TO RMSDEV
*
*    X(I)      I=1...LX IS A FLOATING POINT VECTOR
*
*    LX        SHOULD EXCEED ZERO
*
*    XBASE     IS A FLOATING POINT CONSTANT
*
* OUTPUTS FROM RMSDEV (STRAIGHT RETURN WITH NO OUTPUTS IF LX LSTHN 1)
*
*    RMSXMB    =SQUARE ROOT((SUM(FROM I=1..LX)OF(X(I)-XBASE)SQUARED)/LX)
*
*
* FORTRAN USAGE OF RMSDAV
*     CALL RMSDAV(X,LX,XAVG,RMSXMA)
*
* INPUTS TO RMSDAV
*
*    X(I)      I=1...LX IS A FLOATING POINT VECTOR
*
*    LX        SHOULD EXCEED ZERO
*
* OUTPUTS FROM RMSDAV (STRAIGHT RETURN WITH NO OUTPUTS IF LX LSTHN 1)
*
*    XAVG      = (1/LX)*(SUM(FROM I=1...LX) OF X(I))
*
*    RMSXMA    =SQUARE ROOT((SUM(FROM I=1..LX) OF(X(I)-XAVG)SQUARED)/LX)
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...9) = 1.,2.,3.,4.,5.,6.,7.,8.,9.   LX=9
*    USAGES  -        CALL RMSDEV(X,LX,0.,RMS1)
```

```
*                     CALL RMSDEV(X,LX,1.,RMS2)
*                     CALL RMSDAV(X,LX,XAVG,RMS3)
*    OUTPUTS - RMS1 = (1/3)*SQUARE ROOT(285) = 5.62731
*              RMS2 = (1/3)*SQUARE ROOT(204) = 4.76095
*              RMS3 = (1/3)*SQUARE ROOT(60)  = 2.58199
*
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS SQRT
*
       HTR     0              XR4
       BCI     1,RMSDEV
*
* PRINCIPAL ENTRY.  RMSDEV(X,LX,XBASE,RMSXMB)
*
RMSDEV STZ     ZFDEV          SET ENTRY INDICATOR
       TRA     SETUP
*
* SECONDARY ENTRY.  RMSDAV(X,LX,XAVG,RMSXMA)
*
RMSDAV SXA     ZFDEV,4        SET ENTRY INDICATOR
*
* SAVE XR4, SET ADDRESSES
*
 SETUP SXD     RMSDEV-2,4
 K1    CLA     1,4            A(X)
       ADD     K1             A(X)+1
       STA     GET
       CLA     3,4            A(XBASE OR XAVG)
       STA     SUBTR
*
* CHECK OUT LX, SET IT IN XR4, FLOAT IT, CLEAR SUM, BRANCH ON ENTRY
*
       CLA*    2,4            LX,
       TMI     LEAVE
       PDX     0,4            TO XR4,
       TXL     LEAVE,4,0
       LRS     18
       ORA     OCTK
       FAD     OCTK
       STO     FLX            AND FLOATED.
       STZ     SUM            SUM CLEARED.
       NZT     ZFDEV
       TRA     GET            ALL SET IF RMSDEV
*
* IF RMSDAV, COMPUTE AND STORE XAVG AND THEN RESTORE XR4 TO LX
*
       PXD     0,0
       FAD*    GET
       TIX     *-1,4,1
       FDP     FLX
       LXD     RMSDEV-2,4
       STQ*    3,4            STORE XAVG
       CLA*    2,4
       PDX     0,4            LX BACK TO XR4
*
* COMPUTE THE SUM OF SQUARE DEVIATIONS
*
 GET   CLA     **,4           **=A(X)+1
 SUBTR FSB     **             **=A(XBASE) OR A(XAVG)
       STO     TEMP
       XCA
       FMP     TEMP
       FAD     SUM
       STO     SUM
       TIX     GET,4,1
*
* COMPUTE RMS VALUE, STORE IT, EXIT.
*
       FDP     FLX
       XCA
       TSX     $SQRT,4
       LXD     RMSDEV-2,4
       STO*    4,4
```

```
 LEAVE TRA      5,4
*
* CONSTANTS, TEMPORARIES
*
 OCTK  OCT      233000000000
 SUM   PZE      **,**,**       SUM REGISTER
 TEMP  PZE      **,**,**       INDIVIDUAL DEVIATIONS
 FLX   PZE      **,**,**       LX FLOATING
 ZFDEV PZE      **,0,0         **=0 IF RMSDEV, NON-ZERO IF RMSDAV
       END
```

```
*      RND (FUNCTION)                    9/29/64   LAST CARD IN DECK IS NO. 0078
*      FAP
*RND
       COUNT   60
       LBL     RND
       ENTRY   RND    F(Y)
       ENTRY   RNDUP  F(Y)
       ENTRY   RNDDN  F(Y)
*
*                    ----ABSTRACT----
*
* TITLE - RND   , WITH SECONDARY ENTRY POINTS RNDUP, RNDDN
*        ROUNDS FLTG. PT. NO. UP, DOWN, OR TO NEAREST FLTG. PT. INTEGER
*
*              RND ROUNDS A FLOATING POINT NUMBER TO THE NEAREST FLOATING
*                POINT INTEGER.
*
*              RNDUP ROUNDS A POSITIVE (NEGATIVE) FLOATING POINT NUMBER
*                TO THE NEXT HIGHER (LOWER) FLOATING POINT INTEGER.
*
*              RNDDN ROUNDS A POSITIVE (NEGATIVE) FLOATING POINT NUMBER
*                TO THE NEXT LOWER (HIGHER) FLOATING POINT INTEGER.
*
* LANGUAGE  - FAP, FORTRAN II FUNCTION
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 15 REGISTERS
* SPEED     - 26 MACHINE CYCLES FOR RND
* AUTHOR    - R.A. WIGGINS,  15/9/62
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*        AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*    X1 =   RNDF(Y)
*    X2 = RNDUPF(Y)
*    X3 = RNDDNF(Y)
*
* INPUTS
*
*    Y          IS A FLOATING POINT NUMBER
*               MUST BE LSTHN= 10.**9
*
* OUTPUTS
*
*    X1         IS A FLOATING POINT INTEGER
*
*    X2         IS A FLOATING POINT INTEGER
*
*    X3         IS A FLOATING POINT INTEGER
*
* EXAMPLES
*
* 1. INPUT   - Y=104.2
*    OUTPUTS - X1=104.  X2=105.  X3=104.
*
* 2. INPUT   - Y=.5
*    OUTPUTS - X1=1.  X2=1.  X3=0.
*
* 3. INPUT   - Y=-49.7
*    OUTPUTS - X1=-50.  X2=-50.  X3=-49.
*
* 4. INPUT   - Y=1015.
*    OUTPUTS - X1=1015.  X2=1015.  X3=1015.
*
       BCI     1,RND
 RNDUP TMI     A
       FAD     =O177777777777
       FAD     =.5
 RNDDN UFA     =O233000000000
       FAD     =O233000000000
       TRA     1,4
 A     FSB     =O177777777777
       FSB     =.5
```

```
************************
*    RND               *
************************
              (PAGE  2)
```


```
            TRA     RNDDN
      RND   TMI     A+1
            TRA     RNDUP+2
            END
```

```
*************************
*     RNDDN             *
*************************
 REFER TO
     RND
```


```
*************************
*     RNDDN             *
*************************
 REFER TO
     RND
```


```
*************************
*     RNDUP             *
*************************
 REFER TO
     RND
```


```
*************************
*     RNDUP             *
*************************
 REFER TO
     RND
```

```
*       RNDV    (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0117
*       FAP
*RNDV
        COUNT   100
        LBL     RNDV
        ENTRY   RNDV   (X,LX,XR)
        ENTRY   RNDVUP (X,LX,XR)
        ENTRY   RNDVDN (X,LX,XR)
*
*                      ----ABSTRACT----
*
* TITLE - RNDV  WITH SECONDARY ENTRIES RNDVUP AND RNDVDN
*        ROUND, ROUND UP, OR ROUND DOWN  A FLOATING VECTOR
*
*               RNDV   ROUNDS A FLTG VECTOR TO NEAREST FLTG INTEGERS.
*               RNDVUP ROUNDS ELEMENTS OF A FLTG VECTOR TO LOWEST
*                  FLTG INTEGERS GRTHN= GIVEN ELEMENTS FOR POSITIVE
*                  ELEMENTS, OR TO GREATEST FLTG INTEGERS LSTHN= GIVEN
*                  ELEMENTS FOR NEGATIVE ELEMENTS.
*               RNDVDN ROUNDS ELEMENTS OF A FLTG VECTOR TO GREATEST
*                  FLTG INTEGERS LSTHN= GIVEN ELEMENTS FOR POSITIVE
*                  ELEMENTS, OR TO LOWEST FLTG INTEGERS GRTHN= GIVEN
*                  ELEMENTS FOR NEGATIVE ELEMENTS.
*
*               OUTPUTS MAY REPLACE INPUTS
*
* LANGUAGE   - FAP SUBROUTINES  (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34   REGISTERS
* SPEED      - ABOUT  36 + 32*LX  MACHINE CYCLES,  LX=VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - RND, RNDUP, RNDDN  (FUNCTIONS)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*    CALL RNDV  (X,LX,XR)
*    CALL RNDVUP(X,LX,XR)
*    CALL RNDVDN(X,LX,XR)
*
* INPUTS
*
*    X(I)       I=1...LX  IS ANY FLOATING VECTOR
*
*    LX         SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO ACTION IF LX LSTHN 1
*
*    XR(I)      I=1...LX  IS  XR(I) = ROUNDED FORM OF X(I)
*                  LET X= S*(XI+XF) WHERE S=+1. OR -1., XI IS POSITIVE
*                  WHOLE NUMBER, AND  0.  LSTHN= XF LSTHN 1.
*                  THEN XR(I) WILL ALWAYS = S*XI(I) IF XF=0.
*                  OTHERWISE
*                     XR(I) = S*(XI(I))        FOR RNDVDN
*                     XR(I) = S*(XI(I)+1.0)    FOR RNDVUP
*                     XR(I) = S*(XI(I))      IF XF(I) LSTHN  .5  FOR RNDV
*                     XR(I) = S*(XI(I)+1.0) IF XF(I) GRTHN= .5  FOR RNDV
*
*               EQUIVALENCE(XR,X) IS PERMITTED
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5)  =  1.1, 2.2, -3.5,  4.7, 5.0      XR4=0.0
*    USAGE   -        CALL RNDV  (X,5,XR1)
*                     CALL RNDVUP(X,5,XR2)
*                     CALL RNDVDN(X,5,XR3)
*                     CALL RNDV  (X,1, X )
*                     CALL RNDV  (X,0,XR4)
*    OUTPUTS - XR1(1...5)  = 1., 2., -4., 5., 5.
*              XR2(1...5)  = 2., 3., -4., 5., 5.
*              XR3(1...5)  = 1., 2., -3., 4., 5.
*              X(1) = 1.    XR4 = 0.0 (NO OUTPUT CASE)
*
```

****
*   RNDV   *
****
(PAGE 2)


```
     * PROGRAM FOLLOWS BELOW
     *
     *
     * TRANSFER VECTOR CONTAINS RND,RNDUP AND RNDDN FUNCTIONS
     *
            HTR     0              XR1
            HTR     0              XR4
            BCI     1,RNDV
     * PRINCIPAL ENTRY.  RNDV(X,LX,XR)
      RNDV  CLA     R
      SETUP STO     ROUND
            SXD     RNDV-2,4
            SXD     RNDV-3,1
      K1    CLA     1,4
            ADD     K1             A(X)+1
            STA     GET
            CLA     3,4
            ADD     K1             A(XR)+1
            STA     STORE
            CLA*    2,4            LX
            TMI     LEAVE
            PDX     0,1
            TXL     LEAVE,1,0
     * LOOP
      GET   CLA     **,1           **=A(X)+1
      ROUND TSX     **,4           **=$RND,$RNDUP, OR $RNDDN
      STORE STO     **,1           **=A(XR)+1
            TIX     GET,1,1
     * EXIT
      LEAVE LXD     RNDV-2,4
            LXD     RNDV-3,1
            TRA     4,4
     * SECOND ENTRY.  RNDVUP(X,LX,XR)
     RNDVUP CLA     RUP
            TRA     SETUP
     * THIRD ENTRY.  RNDVDN(X,LX,XR)
     RNDVDN CLA     RDN
            TRA     SETUP
     * CONSTANTS
      R     TSX     $RND,4
      RUP   TSX     $RNDUP,4
      RDN   TSX     $RNDDN,4
            END
```

```
************************
*    RNDVDN            *
************************
 REFER TO
     RNDV
```


```
************************
*    RNDVDN            *
************************
 REFER TO
     RNDV
```


PROGRAM LISTINGS

```
************************
*    RNDVUP            *
************************
 REFER TO
     RNDV
```


```
************************
*    RNDVUP            *
************************
 REFER TO
     RNDV
```

```
*     ROAR2  (SUBROUTINE)                 9/10/64   LAST CARD IN DECK IS NO. 0113
*     LABEL
CROAR2
      SUBROUTINE ROAR2 (JOB,XA,N,M,XRA)
C
C                         ----ABSTRACT----
C
C TITLE - ROAR2
C        ROTATE CENTRO-SYMMETRIC OR ANTISYMMETRIC 2-DIMENSIONAL ARRAY
C
C               ROAR2 ROTATES HALF OF A CENTRO-SYMMETRIC, OR ANTISYM-
C               METRIC 2-DIMENSIONAL ARRAY BY 90 DEGREES.  THUS, IF WE
C               ARE GIVEN HALF OF AN ARRAY X(I,J) I=-N,...,N  J=0,...,M
C               THAT IS STORED BY COLUMNS AS
C
C                     X(N,0)  X(N,1)  X(N,2)  .  .  X(N,M)
C                     .       .       .             .
C                     .       .       .             .
C                     X(0,0)  X(0,1)  X(0,2)  .  .  X(0,M)
C                     .       .       .             .
C                     .       .       .             .
C                     X(-N,0) X(-N,1) X(-N,2) .  .  X(-N,M)
C
C               THEN ROAR2 ROTATES THE TERMS SO THAT THEY ARE STORED BY
C               COLUMNS AS
C
C                     X(M,0)  X(M,1)  X(M,2)  .  .  X(M,N)
C                     .       .       .             .
C                     .       .       .             .
C                     X(0,0)  X(0,1)  X(0,2)  .  .  X(0,N)
C                     .       .       .             .
C                     .       .       .             .
C                     X(-M,0) X(-M,1) X(-M,2) .  .  X(-M,N)
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094  (MAIN FRAME ONLY)
C STORAGE    - 174 REGISTERS
C SPEED      - ABOUT  .000017*M*N**2 + .000012*N**2 + .00021*M*N
C                  + .00070*N + .00012*M + .00115 SECONDS
C                    ON THE 7094 MOD 1 .
C AUTHOR     - R.A. WIGGINS,  JUNE, 1963
C
C                         ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - MATRA,MOVREV,REVERS
C       AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL ROAR2 (JOB,XA,N,M,XRA)
C
C INPUTS
C
C    JOB        =1  INDICATES XA IS CENTRO-SYMMETRIC.
C               =-1 INDICATES XA IS CENTRO-ANTISYMMETRIC.
C
C    XA(I)      I=1,...,(N+N+1)*(M+1) CONTAINS X(J,K)  J=-N,...,N,
C                 K=0,...,M  AS DEFINED IN THE ABSTRACT.
C
C    N          MUST BE GRTHN  0
C
C    M          MUST BE GRTHN= 0
C
C OUTPUTS
C
C    XRA(I)     I=1,...,(M+M+1)*(N+1) CONTAINS X(K,J)  K=-M,...,M,
C                 J=0,...,N AS DEFINED IN THE ABSTRACT.
C               EQUIVALENCE WITH XA IS PERMITTED.
C
C EXAMPLES
C
C 1. INPUTS  - N=2  M=3  XA(1...20) = 2.0,1.0,5.0, 1.0,2.0,
C               JOB=1                   2.1,1.1,0.1,-1.1,-2.1,
C                                       2.2,1.2,0.2,-1.2,-2.2,
C                                       2.3,1.3,0.3,-1.3,-2.3
```

```
C     OUTPUTS -           XRA(1...21) = 0.3,0.2,0.1,5.0, 0.1, 0.2, 0.3,
C                                       1.3,1.2,1.1,1.0,-1.1,-1.2,-1.3,
C                                       2.3,2.2,2.1,2.0,-2.1,-2.2,-2.3
C
C  2. INPUTS   - N=2  M=3  XA(1...20) = 2.0,1.0,0.0,-1.0,-2.0,
C                JOB=-1                 2.1,1.1,0.1,-1.1,-2.1,
C                                       2.2,1.2,0.2,-1.2,-2.2,
C                                       2.3,1.3,0.3,-1.3,-2.3
C     OUTPUTS -           XRA(1...21) = 0.3,0.2,0.1,0.0,-0.1,-0.2,-0.3,
C                                       1.3,1.2,1.1,1.0, 1.1, 1.2, 1.3,
C                                       2.3,2.2,2.1,2.0, 2.1, 2.2, 2.3
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION XA(2),XRA(2)
      NN=N
      MM=M
      NRB=NN+NN+1
      NCB=MM+1
      NRA=NCB+MM
      NCA=NN+1
      LXB=NRB*NCB
      LXA=NRA*NCA
      CALL MATRA (XA,NRB,NCB,XRA)
      IF=1
      IL=LXB+1
      LRO=LXB
 1000 Z=0.
      DO 10 I=1,NCA
      IF1=IF+MM
      CALL MOVREV (LRO,1,XRA(IF),1,XRA(IF1),1)
      CALL MOVREV (MM,1,XRA(IL),-1,XRA(IF),JOB)
 1001 Z=1.
      IF=IF+NRA
      IL=IL-1
   10 LRO=LRO-NRA-1
      CALL REVERS(LXA,XRA)
      RETURN
      END
```

```
*      ROTAT1 (SUBROUTINE)               9/4/64   LAST CARD IN DECK IS NO. 0109
*      FAP
*ROTAT1
       COUNT    100
       LBL      ROTAT1
       ENTRY    ROTAT1 (X,NX,NUP,ROTX)
*
*                    ----ABSTRACT----
*
* TITLE - ROTAT1
*        ROTATE A VECTOR UPWARDS OR DOWNWARDS AN ARBITRARY AMOUNT
*
*              ROTAT1 ROTATES A VECTOR UPWARDS OR DOWNWARDS A PRESCRIBED
*              NUMBER OF UNITS SUCH THAT ELEMENTS SHIFTED OUT OF ONE END
*              ARE ROTATED INTO THE OPPOSITE END.  IT IS IMMATERIAL
*              WHETHER THE VECTOR IS FIXED POINT OR FLOATING POINT.  THE
*              ROTATION IS ACCOMPLISHED IN ONE PASS OF THE VECTOR RATHER
*              THAN BY SUCCESSIVE SHIFTING.  OUTPUT ON TOP OF INPUT IS
*              PERMITTED.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 46 REGISTERS
* SPEED      - TAKES  (12 TO 18) * VECTOR LENGTH  MACHINE CYCLES ON 7090
* AUTHOR     - R.A. WIGGINS AND J.CLARK,  JUNE 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL ROTAT1(X,NX,NUP,ROTX)
*
* INPUTS
*
*    X(I)       I=1,...,NX IS THE VECTOR TO BE ROTATED.
*                 (NOTE - X MAY BE EITHER FIXED OR FLOATING POINT WITHOUT
*                  MODIFICATION TO THE PROGRAM).
*
*    NX         IS THE LENGTH OF THE X VECTOR.
*               MUST HAVE VALUE 1 OR GREATER (STRAIGHT EXIT OTHERWISE)
*
*    NUP        IS THE NUMBER OF REGISTERS X IS TO BE ROTATED.
*                 (UPWARDS IF NUP POSITIVE)
*
* OUTPUTS
*
*    ROTX(I)    I=1,...,NX IS THE VECTOR ROTATED SUCH THAT
*                 ROTX(I)=X((I-NUP)MODULO NX).
*                 (NOTE - THE EQUIVALENCE OF ROTX(1) WITH X(1) IS
*                  PERMITTED).
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5) = 4.,6.,3.,9.,1.   NX=5   NUP=8
*    OUTPUTS - ROTX(1...5) = 3.,9.,1.,4.,6.
*
* 2. INPUTS  - X(1...5) = 4.,6.,3.,9.,1.   NX=5   NUP=-1
*    USAGE   -        CALL ROTAT1(X,NX,NUP,X)
*    OUTPUTS - X(1...5) = 6.,3.,9.,1.,4.
*
* PROGRAM FOLLOWS BELOW
*
 XR1   HTR      0
 XR4   HTR      0
       BCI      1,ROTAT1
ROTAT1 SXD      XR4,4
       SXD      XR1,1
       CAL      1,4             =ADR(X)
       ADD      =1B35
       STA      X
       CAL      4,4             =ADR(XR)
       ADD      =1B35
       STA      XR
```

****
*   ROTAT1   *
****
(PAGE 2)

```
        CLA*   2,4             =LX
        TZE    EXIT
        TMI    EXIT
        PDX    ,1
        TXI    *+1,1,1
        STO    LX
        STD    T2
        LDQ*   3,4             =NUP
        ZAC
        LLS    0
        DVP    LX
        TPL    *+2
        ADD    LX
        STD    T1
        AXT    1,4
        SXD    T3,4
        CLA*   X
        TRA    X
T1      TXI    *+1,4,**        **=MOD(MOD(NUP,LX)+LX,LX)
T2      TIX    T3,4,**         **=LX
        XCA
X       LDQ    **,4            **=ADR(X)+1
XR      STO    **,4            **=ADR(XR)+1
T4      TIX    T1,1,1
EXIT    LXD    XR1,1
        LXD    XR4,4
        TRA    5,4
T3      TXH    T2+1,4,**       **=FIRST VALUE OF LOOP
        STQ*   XR
        TXI    *+1,4,1
        SXD    T3,4
        LDQ*   X
        TRA    T4
LX      PZE
        END
```

```
*       RPLFMT (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0084
*       FAP
*RPLFMT
        COUNT   100
        LBL     RPLFMT
        ENTRY   RPLFMT (FMT,FMTNEW)
*
*                   ----ABSTRACT----
*
* TITLE - RPLFMT
*         REPLACE THE FORMAT OF A SUCCEEDING INPUT OR OUTPUT STATEMENT
*
*               RPLFMT HAS TWO ARGUMENTS FMT AND FMTNEW.  RPLFMT ASSUMES
*               THAT SHORTLY BELOW THE CALL RPLFMT STATEMENT THERE
*               APPEARS AN INPUT OR OUTPUT STATEMENT USING THE FORMAT
*               FMT.  THIS STATEMENT IS FOUND AND THE FORMAT FMTNEW IS
*               SUBSTITUTED FOR FMT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 17 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON JR.,   SEPTEMBER 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE (ILLUSTRATIVE)
*
*         CALL RPLFMT(FMT,FMTNEW)
*
*               (THEN ANY AMOUNT OF PROGRAM NOT INVOLVING
*               INPUT OR OUTPUT ACCORDING TO FORMAT FMT)
*
*         WRITE OUTPUT TAPE 2,FMT,LIST
*
*    LIST WILL BE PRINTED ACCORDING TO FORMAT FMTNEW RATHER THAN FMT.
*
*    CAUTION  -  THE CHANGE INDUCED BY RPLFMT (PZE FMT IS REPLACED BY
*                PZE FMTNEW) IS PERMANENT.  IF REPEATED USE OF THE SAME
*                SEQUENCE IS DESIRED USING DIFFERENT FMTNEW VALUES, THE
*                ORIGINAL FMT SHOULD BE RESTORED BY A SCHEME SUCH AS
*         CALL RPLFMT(FMT,FMTNEW)
*         GO TO 20
*      10 CALL RPLFMT(FMTNEW,FMT)
*         GO TO 30
*      20 WRITE OUTPUT TAPE 2,FMT,LIST
*         GO TO 10
*      30 CONTINUE
*
* EXAMPLES
*
* 1. INPUTS   - X = 3.14159  FMT(1)=4H(I7)
*                     FMTNEW(1...3) = 14H(5H X = ,F9.5)
*    USAGE    -          DIMENSION FMT(1),FMTNEW(3)
*                        CALL RPLFMT(FMT,FMTNEW)
*                        WRITE OUTPUT TAPE 2,FMT,X
*    OUTPUTS - X =    3.14159  IS PRINTED OFFLINE FROM LOGICAL TAPE 2.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR4
        BCI     1,RPLFMT
* ONLY ENTRY.   RPLFMT(FMT,FMTNEW)
RPLFMT  SXD     RPLFMT-2,4
        CLA     2,4            A(FMTNEW)
        STA     PZEA
        LDQ     PZEA           MQ HAS REPLACEMENT SETTING
        CLA     1,4            A(FMT)
        STA     PZEA           PZEA HAS OLD SETTING
* COMPARE LOOP
```

****
*   RPLFMT   *
****
(PAGE 2)


```
      CAL    CAL     4,4            (FIRST POSSIBILITY)
             LAS     PZEA
             TRA     *+2
             TRA     SET            GOT IT
             TXI     CAL,4,-1       TRY AGAIN
      SET    STQ     4,4
             LXD     RPLFMT-2,4
             TRA     3,4
      PZEA   PZE     **             ** = A(FMTNEW)  THEN A(FMT)
             END
```

```
*       RSKIP (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0089
*       FAP
*RSKIP
        COUNT   100
        LBL     RSKIP
        ENTRY   RSKIP (NTAPE,NRECS,EOF)
*
*                    ----ABSTRACT----
*
* TITLE - RSKIP
*         SKIP FORWARD OR BACKWARD OVER RECORDS ON TAPE
*
*               RSKIPS SKIPS AN ARBITRARY NUMBER OF RECORDS FORWARD OR
*               BACKWARD ON A TAPE.  END FILES ARE CHECKED FOR WHILE
*               SKIPPING FORWARD BUT NOT WHILE SKIPPING BACKWARDS.
*
* LANGUAGE   - FAP SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS 1 TAPE UNIT)
* STORAGE    - 37 REGISTERS
* SPEED      - .0085 SEC PER 80-CHARACTER, HIGH-DENSITY RECORD -
*                       FORWARD SKIPPING.
*            - .0378 SEC PER 80-CHARACTER, HIGH-DENSITY RECORD -
*                       BACK SKIPPING.
* AUTHOR     - R.A. WIGGINS   DEC, 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - (IOS),(TRC),(TCO),(TEF),(RDS),
*                                      (BSR)
*
* FORTRAN USAGE
*     CALL RSKIP (NTAPE,NRECS,EOF)
*
* INPUTS
*
*    NTAPE      IS LOGICAL TAPE NUMBER OF THE TAPE THAT IS TO BE SPACED.
*               IS FORTRAN II INTEGER
*
*    NRECS      IS THE NUMBER OF PHYSICAL RECORDS TO BE SKIPPED.
*               IF GRTHN 0 THE TAPE IS MOVED AWAY FROM THE BEGINNING
*                 POINT.
*               IF = 0 THE TAPE IS NOT MOVED.
*               IF LSTHN 0 THE TAPE IS MOVED TOWARD THE BEGINNING POINT.
*               IS FORTRAN II INTEGER.
*
* OUTPUTS  THE TAPE IS MOVED.
*
*    EOF        =1. IF AN END FILE WAS ENCOUNTERED DURING SKIPPING
*                    FORWARD.
*               =0. IF NO END FILE WAS ENCOUNTERED OR NRECS LSTHN 0.
*                 (END FILES ARE NOT DETECTED DURING BACKSPACING)
*
*               WHETHER SKIPPING FORWARD OR BACKWARD, EACH END-OF-FILE
*               ENCOUNTERED COUNTS AS ONE RECORD.
*
*
* PROGRAM FOLLOWS BELOW.
*
        HPR     0
        BCI     1,RSKIP
 RSKIP  SXD     *-2,4          SAVE INDEX.
        CLA*    1,4            GET LOGICAL TAPE NO.
        TSX     $(IOS),4       SET UP (IOS).
        LXD     RSKIP-2,4      RESET INDEX 4.
        STZ*    3,4            CLEAR EOF INDICATOR.
        CLA*    2,4            GET NO OF RECORDS.
        TZE     4,4            EXIT IF ZERO.
        PDX     ,4             SAVE.
        LDQ*    $(TRC)         SET UP
        SLQ     TRC            REDUNDANCY CHECK,
        LDQ*    $(TCO)
        SLQ     TCO            CHANNEL IN OPERATION,
        LDQ*    $(TEF)         AND
        SLQ     TEF            END FILE TRANSFERS.
```

RSKIP

(PAGE 2)


```
        TMI    MI
FORW    XEC*   $(RDS)
        TIX    *-1,4,1
        LXD    RSKIP-2,4      RESET IR 4.
TCO     TCOA   *              DELAY UNTIL TAPE STOPS.
TRC     TRCA   *+1            TURN OFF REDUNDANCY CHECK LIGHT.
TEF     TEFA   *+2            CHECK END FILE.
        TRA    4,4
        CLA    =1.
        STO*   3,4
        TRA    4,4
MI      XEC*   $(BSR)
        TIX    *-1,4,1
        TRA    TCO-1
        END
```

```
************************
*    RVPRTS            *
************************
 REFER TO
     CHPRTS
```


```
************************
*    RVPRTS            *
************************
 REFER TO
     CHPRTS
```

```
*       SAME (FUNCTION)                  9/29/64   LAST CARD IN DECK IS NO. 0039
*       FAP
*SAME
        COUNT   45
        LBL     SAME
        ENTRY   SAME    F(IX1)
        ENTRY   XSAME   F( X1)
*
*                    ----ABSTRACT----
*
* TITLE - SAME  , WITH SECONDARY ENTRY POINT XSAME
*        ENABLE MIXED EXPRESSIONS IN FORTRAN
*
*             SAME AND XSAME ARE FUNCTIONS WHICH DO NOTHING BUT RETURN
*             TO THE CALLING PROGRAM.  THIS ALLOWS THE USE OF MIXED
*             EXPRESSIONS IN FORTRAN.  FOR EXAMPLE, THE FIXED POINT
*             ADDITION OF TWO WORDS (CALLED X AND Y) WITH FLOATING
*             POINT NAMES IS ACCOMPLISHED BY
*
*                    ISUM = XSAMEF(X) + XSAMEF(Y)
*
* LANGUAGE  - FAP, FUNCTION  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 704, 709, OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 1 CELL
* SPEED     - 2 MACHINE CYCLES
* AUTHOR    - J.F. CLAERBOUT
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*      X1  =  SAMEF(IX1)
*      IX1 = XSAMEF(X1)
*
*
 XSAME BSS      0
 SAME  TRA      1,4
       END
```

```
*      SEARCH (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0094
*      FAP
*SEARCH
       COUNT   75
       LBL     SEARCH
       ENTRY   SEARCH  (LV,VECTOR,XNUM,INDEX)
*
*                   ----ABSTRACT----
*
* TITLE - SEARCH
*       SEARCH A VECTOR FOR A VALUE
*
*             SEARCH A VECTOR OF FIXED, FLOATING OR LOGICAL NUMBERS
*             FOR A PARTICULAR NUMBER.  IF THIS NUMBER IS FOUND, ITS
*             INDEX IN THE VECTOR IS RETURNED, IF IT IS NOT FOUND, A
*             ZERO IS RETURNED AS THE INDEX.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 25 REGISTERS
* SPEED     - LESS THAN (30 + 8*LENGTH OF LIST) MACHINE CYCLES
* AUTHOR    - R.A. WIGGINS,   16/7/62
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL SEARCH(LV,VECTOR,XNUM,INDEX)
*
* INPUTS
*
*    VECTOR(I) I=1...LV  IS A VECTOR OF FLOATING, FIXED, OR HOLLERITH
*                NUMBERS
*
*    LV         IS FORTRAN II INTEGER
*               MUST BE GRTHN=0
*
*    XNUM       IS A NUMBER OF SAME MODE AS VECTOR
*
* OUTPUTS
*
*    INDEX      INDEX OF XNUM IN VECTOR. I.E. VECTOR(INDEX) = XNUM
*               =0 IF XNUM IS NOT CONTAINED IN VECTOR
*               =0 IF LV=0
*
* EXAMPLES
*
* 1. INPUTS  - VECTOR(1...5)=1.,3.,2.5,4.,4.1  LV=5  XNUM=2.5
*    OUTPUTS - INDEX=3
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT XNUM=5.
*    OUTPUTS - INDEX=0
*
* 3. INPUTS  - VECTOR(1...5)=MLI1,4,7,11,9  LV=5  XNUM=MLI11
*    OUTPUTS - INDEX=4
*
* 4. INPUTS  - VECTOR(1...3)=6HA1    ,A2    ,B       LV=3  XNUM=6HA1
*    OUTPUTS - INDEX=1
*
* 5. INPUTS  - VECTOR(1...2)=1.,2.  LV=0  XNUM=1.
*    OUTPUTS - INDEX=0
*
* 6. INPUTS  - VECTOR(1...5)=MLI1,2,5,2,3  LV=5  XNUM=MLI 2
*    OUTPUTS - INDEX=2
*
* 7. INPUTS  - VECTOR(1)=MLI 1  LV=1  XNUM=MLI 1
*    OUTPUTS - INDEX=1
*
       HTR     0
       BCI     1,SEARCH
SEARCH SXD     *-2,4
       SXA     ADR,1
       CLA*  1,4
```

****
*   SEARCH   *
****
(PAGE 2)

```
         TZE     ADR-1
         STD      A
         CAL    2,4
         ADD      =1B35
         STA    DATA
         CAL    3,4
         STA    ITEM
         AXT      1,1
DATA     CLA    **,1
ITEM     SUB    **
         TZE    ADR+2
         TXI      A,1,1
A        TXL      DATA,1,**
         STZ*   4,4
ADR      AXT    **,1
         TRA    5,4
         PXD    ,1
         STO*   4,4
         TRA    ADR
         END
```

```
*      SEQSAC (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0277
*      FAP
*SEQSAC
       COUNT   300
       LBL     SEQSAC
       ENTRY   SEQSAC (ARGLO, ARGDEL)
       ENTRY   NEXCOS F(DUMMY)
       ENTRY   NEXSIN F(DUMMY)
*
*
*                    ----ABSTRACT----
*
* TITLE - SEQSAC,  WITH SECONDARY ENTRIES NEXCOS AND NEXSIN (FUNCTIONS)
*        FAST FUNCTIONS FOR SEQUENTIAL SINES AND COSINES
*
*              SEQSAC,  NEXCOS,  AND  NEXSIN  ARE A PROGRAM SET FOR
*              PROVIDING A SUCCESSION OF SINE AND/OR COSINE VALUES AT
*              HIGH SPEED, APPLICABLE IN CASES WHERE THE SUCCESSIVE
*              ARGUMENT VALUES DIFFER BY A CONSTANT.  SPEED IS ATTAINED
*              BY THE USE OF SUM ANGLE FORMULAS, AND ERROR GROWTH IS
*              LIMITED BY AUTOMATIC RESETTING EVERY HUNDREDTH ARGUMENT.
*
*              THE ENTRY  SEQSAC  IS USED ONCE TO INITIALIZE FOR THE
*              DESIRED BASE VALUE AND INCREMENTAL VALUE OF THE ARGUMENT.
*              THEREAFTER  NEXCOS  AND/OR  NEXSIN  ARE USED IN LOOP
*              FASHION AS FUNCTIONS (WITH DUMMY ARGUMENTS) TO PROVIDE
*              THE SUCCESSIVE VALUES.
*
* LANGUAGE   - FAP SUBROUTINE AND FUNCTIONS (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 94 REGISTERS
* SPEED      - THE INITIALIZING CALL OF  SEQSAC  TAKES ABOUT 80 MACHINE
*                 CYCLES (ON THE 7090) PLUS FOUR USES OF THE FORTRAN
*                 SYSTEM FUNCTIONS  COS  AND  SIN.
*              THEREAFTER EACH SINE OR COSINE VALUE TAKES ABOUT
*                 110 MACHINE CYCLES IF THE SUBSEQUENT LOOP USES ONLY
*                 NEXCOS  OR  NEXSIN,
*               OR
*                 60 MACHINE CYCLES IF THE SUBSEQUENT LOOP USES BOTH
*                 NEXCOS  AND NEXSIN.
* AUTHOR     - S.M. SIMPSON,  JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NOT ANY
*       AND FORTRAN SYSTEM ROUTINES - COS, SIN
*
* ILLUSTRATIVE FORTRAN USAGE
*
*     CALL SEQSAC(ARGLO, ARGDEL)
*     DO 10  I=1,N
*     C(I) = NEXCOSF(DUMMY)
* 10  S(I) = NEXSINF(DUMMY)
*
*           OR
*
*     CALL SEQSAC(ARGLO, ARGDEL)
*     DO 10  I=1,N
*     S(I) = NEXSINF(DUMMY)
* 10  C(I) = NEXCOSF(DUMMY)
*
*           OR
*
*     CALL SEQSAC(ARGLO, ARGDEL)
*     DO 10  I=1,N
* 10  C(I) = NEXCOSF(DUMMY)
*
*           OR
*
*     CALL SEQSAC(ARGLO, ARGDEL)
*     DO 10  I=1,N
* 10  S(I) = NEXSINF(DUMMY)
*
*
```

```
* INPUTS TO SEQSAC
*
*    ARGLO      IS THE ARGUMENT VALUE, IN RADIANS, OF THE FIRST COSINE
*                  AND/OR SINE TO BE COMPUTED BY THE SUBSEQUENT USE OF
*                  NEXCOS  AND/OR  NEXSIN.
*
*    ARGDEL     IS THE INCREMENT, IN RADIANS, BETWEEN SUCCESSIVE
*                  ARGUMENT VALUES.
*
*
* OUTPUTS FROM SEQSAC - NONE, ITS FUNCTION IS MERELY TO INITIALIZE
*                        NEXCOS  AND  NEXSIN.
*
*
* INPUTS TO NEXCOS AND NEXSIN
*
*    DUMMY      IS THE NAME OF A DUMMY VARIABLE NOT USED BY  NEXCOS  OR
*                  NEXSIN  .  ITS PURPOSE IS TO SATISFY THE FORTRAN
*                  REQUIREMENT THAT EVERY FUNCTION MUST HAVE AT LEAST ONE
*                  ARGUMENT.
*
*
* OUTPUTS FROM NEXSIN AND NEXCOS
*
*               IF  NEXSIN  AND/OR  NEXCOS  ARE USED WITHOUT A PRIOR CALL
*                  OF  SEQSAC,  THE VALUES GENERATED WILL BE ZERO.
*                  OTHERWISE, ON THE I-TH USE, SUBSEQUENT TO A CALL
*                  SEQSAC STATEMENT, OF  NEXCOS  AND/OR  NEXSIN  WE HAVE
*
*    NEXCOSF    HAS VALUE = COS(ARGLO+(I-1)*ARGDEL)
*
*    NEXSINF    HAS VALUE = SIN(ARGLO+(I-1)*ARGDEL)
*
*
* EXAMPLES
*
* 1. INPUTS   - ARGLO = 0.0    ARGDEL = 3.14159265/6.0  (30 DEGREES)
*    USAGE    -        CALL SEQSAC (ARGLO,ARGDEL)
*                      DO 10  I=1,202
*                      C(I) = NEXCOSF(DUMMY)
*                 10   S(I) = NEXSINF(DUMMY)
*    OUTPUTS - C(1...202) = 1.,.866,  .5,0., -.5,-.866,-1.,-.866,...,
*                           -.5,0.
*              S(1...202) = 0.,  .5,.866,1.,.866,  .5,0.,-.5,...,
*                           -.866,-1.
*
* 2. INPUTS   - SAME AS EXAMPLE 1.
*    USAGE    - SAME AS EXAMPLE 1. EXCEPT THE LAST TWO STATEMENTS ARE
*                  EXCHANGED.
*    OUTPUTS - SAME AS EXAMPLE 1.
*
* 3. INPUTS   - SAME AS EXAMPLE 1.
*    USAGE    - SAME AS EXAMPLE 1. BUT DELETING THE
*                  C(I) = NEXCOSF(DUMMY) STATEMENT.
*    OUTPUTS - S(1...202) = SAME AS EXAMPLE 1.
*
* 4. INPUTS   - SAME AS EXAMPLE 1.
*    USAGE    - SAME AS EXAMPLE 1. BUT DELETING THE
*                  S(I) = NEXSINF(DUMMY) STATEMENT AND RELABELLING THE
*                  PREVIOUS STATEMENT AS NUMBER 10 .
*    OUTPUTS - C(1...202) = SAME AS EXAMPLE 1.
*
* 5. INPUTS   - SAME AS EXAMPLE 1. EXCEPT ARGLO = 3.14159265/6.0
*    USAGE    - SAME AS EXAMPLE 1.
*    OUTPUTS - C(1...202) = .866,.5,0.,-.5,...,0.,.5
*              S(1...202) = .5,.866,1.,.866,...,-1.,-.866
*
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS  COS(F),  SIN(F).
*
        HTR      0                XR4
        BCI      1,SEQSAC
*
```

```
*      SCPSCL (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0110
*      FAP
*SCPSCL
       COUNT   100
       LBL     SCPSCL
       ENTRY   SCPSCL (SPACE, NOPTP, YTOP, YBOT, CONVK, CONVL)
*
*                    ----ABSTRACT----
*
* TITLE - SCPSCL
*       SCALE VECTOR TO INTEGERS FOR SCOPE, CLIPPING EXCESSIVE VALUES
*
*             SCPSCL SCALES DATA FOR SCOPE PLOT FOR GRAPH SUBROUTINE.
*             THE OUTPUT CAN BE EXPRESSED BY THE FORMULAE
*
*                X = MAX1F (MIN1F (SPACE(I),YTOP), YBOT)
*
*                SPACE(I) = XFIXF (CONVK + CONVL*X)
*
*             WHERE SPACE IS A FLOATING POINT VECTOR ON INPUT
*                         IS A FORTRAN II INTEGER VECTOR ON OUTPUT
*                    CONVK, CONVL, YTOP, AND YBOT ARE FLOATING POINT
*                          PARAMETERS SUPPLIED BY THE CALLING PROGRAM.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 33 REGISTERS
* SPEED     -
* AUTHOR    - S.M. SIMPSON
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL SCPSCL(SPACE,NOPTP,YTOP,YBOT,CONVK,CONVL)
*
* INPUTS
*
*    SPACE(I)  I=1...NOPTP IS A VECTOR OF FLOATING POINT NUMBERS
*
*    NOPTP     IS FORTRAN II INTEGER.
*              MUST BE GRTHN=1
*
*    YTOP      IS A FLOATING POINT NUMBER WHICH GIVES THE UPPER
*                LIMIT THAT THE NUMBERS IN SPACE MAY ATTAIN BEFORE
*                SCALING.  IF THE NUMBER EXCEEDS YTOP THEN IT IS
*                REPLACED BY YTOP.
*
*    YBOT      IS A FLOATING POINT NUMBER WHICH GIVES THE LOWER LIMIT
*                THAT THE NUMBERS IN SPACE MAY ATTAIN BEFORE SCALING.
*                IF THE NUMBER IS LSTHN YBOT THEN IT IS REPLACED BY
*                YBOT.
*              MUST BE LSTHN YTOP
*
*    CONVK     IS A FLOATING POINT NUMBER WHICH IS ADDED TO THE NUMBERS
*                IN SPACE AFTER THESE ARE SCALED.
*
*    CONVL     IS A FLOATING POINT NUMBER BY WHICH THE NUMBERS IN SPACE
*                ARE MULTIPLIED FOR SCALING.
*
* OUTPUTS
*
*    SPACE(I)  I=1...NOPTS IS A VECTOR OF FIXED POINT NUMBERS.
*
* EXAMPLES
*
* 1. INPUTS  - SPACE(1...5) = -10.,-5.,0.,5.,10.   NOPTP = 5
*              YTOP = 20.  YBOT = -20.  CONVK = 2.  CONVL = 10.
*    OUTPUTS - SPACE(1...5) = -98,-48,2,52,102
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  YTOP = 8.   YBOT = -2.
*    OUTPUTS - SPACE(1...5) = -18,-18,2,52,82
*
```

SCPSCL
(PAGE 2)

```
         BCI    1,SCPSCL
SCPSCL   SXA    LV,1
         CLA*   2,4         NOPTP
         PDX    0,1
         CLA    1,4         SPACE
         ADD    K1
         STA    SC3
*IS SPACE(I) GREATER OR = YTOP
    SC3  CLA    **,1        **=SPACE+1
         CAS*   3,4         YTOP
         NOP                YES
         TRA    SC5         YES
*IS IT LESS THAN OR = YBOT
         CAS*   4,4         YBOT
         TRA    SC10        NO
         NOP                YES
         CLA*   4,4         YES
         TRA    SC10
   SC5   CLA*   3,4
         TRA    SC10
  SC10   XCA
         FMP*   6,4         CONVL
         FAD*   5,4         CONVK
         UFA    ORF
         LRS    0
         ANA    AN
         LLS    0
         ALS    18
         STO*   SC3
         TIX    SC3,1,1
    LV   AXT    **,1
         TRA    7,4
    K1   PZE    1
   ORF   OCT    233000000000
    AN   OCT    377777
         END
```

```
* PRINCIPAL ENTRY.  SEQSAC (ARGLO,ARGDEL)
*       (THIS ENTRY TAKES 78 CYCLES PLUS 2 COSINES PLUS 2 SINES)
*
SEQSAC SXD     SEQSAC-2,4
       CLA*    1,4
       STO     LASBAS          LASBAS = ARGLO
       FSB*    2,4
       STO     TEMP
       TSX     $COS,4
       STO     LASCOS          LASCOS = COS(ARGLO-ARGDEL)
       CLA     TEMP
       TSX     $SIN,4
       STO     LASSIN          LASSIN = SIN(ARGLO-ARGDEL)
       LXD     SEQSAC-2,4
       CLA*    2,4
       TSX     $COS,4
       STO     COSDEL          COSDEL = COS(ARGDEL)
       LXD     SEQSAC-2,4
       CLA*    2,4
       TSX     $SIN,4
       STO     SINDEL          SINDEL = SIN(ARGDEL)
       LXD     SEQSAC-2,4
       LDQ*    2,4
       FMP     FNMAX
       STO     BASDEL          BASDEL = FNMAX*ARGDEL
       STZ     NSOFAR
       STZ     ZFBUSD
       TRA     3,4
*
* SECOND ENTRY.  NEXCOS F(DUMMY)
*       (AVERAGE - 9 CYCLES IF JUMP TO NEW, 6 OTHERWISE)
*
NEXCOS STZ     ZIFCOS
       NZT     ZFBUSD
       STZ     ZFLCOS
       NZT     ZFLCOS
       TRA     NEW
*
* (LAST ENTRY REQUESTED SINE AND COMPUTED BOTH SINE AND COSINE)
*
       CLA     LASCOS
       STZ     ZFBUSD
       TRA     1,4
*
* THIRD ENTRY.  NEXSIN F(DUMMY)
*
NEXSIN SXD     ZIFCOS,4
       NZT     ZFBUSD
       SXD     ZFLCOS,4
       ZET     ZFLCOS
       TRA     NEW
*
* (LAST ENTRY REQUESTED COSINE AND COMPUTED BOTH SINE AND COSINE)
*
       CLA     LASSIN
       STZ     ZFBUSD
       TRA     1,4
*
* RECOMPUTE THE 101-ST, THE 201-ST, ... VALUES
*       (AVERAGE TIME - 30 CYCLES PLUS 1 SINE PLUS 1 COSINE)
*
 RESET CLA     KD1
       STO     NSOFAR
       CLA     LASBAS
       FAD     BASDEL
       STO     LASBAS
       TSX     $SIN,4
       STO     LASSIN
       CLA     LASBAS
       TSX     $COS,4
       LXD     ZFBUSD,4
       TRA     STO
*
* IF NEW VALUES TO BE COMPUTED, CHECK FOR RESETTING FIRST
```

```
*       (AVERAGE TIME - 10 CYCLES)
*
 NEW    SXD     ZFBUSD,4
        CLA     NSOFAR
        ADD     KD1
        STO     NSOFAR
        CAS     NMAX
        TRA     RESET
        NOP
*
* IF NO RESETTING, COMPUTE
*
* NEWSIN = LASSIN*COSDEL+LASCOS*SINDEL
*
* NEWCOS = LASCOS*COSDEL-LASSIN*SINDEL
*
*       (AVERAGE TIME INCLUDING EXIT - 87 CYCLES)
*
        LDQ     LASSIN
        FMP     COSDEL
        STO     TEMP
        LDQ     LASCOS
        FMP     SINDEL
        FAD     TEMP
        LDQ     LASSIN
        STO     LASSIN          STORE NEWSIN
        FMP     SINDEL
        STO     TEMP
        LDQ     LASCOS
        FMP     COSDEL
        FSB     TEMP
 STO    STO     LASCOS          STORE NEWCOS
        ZET     ZIFCOS
        CLA     LASSIN
        TRA     1,4
*
* CONSTANTS, TEMPORARIES
*
 KD1    PZE     0,0,1
 NMAX   PZE     0,0,100
 FNMAX  DEC     100.0
LASBAS  PZE     **,**,**        ARGLO, ARGLO+BASDEL, ...
BASDEL  PZE     **,**,**        ARGDEL*FNMAX
NSOFAR  PZE     0,0,**          = 0,1,...,NMAX+1,1,2,...,NMAX+1,1,2,...
ZFBUSD  PZE     **,**,**        ZERO IF BOTH PREVIOUSLY COMPUTED SINE AND
*                                 COSINE VALUES WERE USED (0 INITIAL)
ZIFCOS  PZE     **,**,**        ZERO IF PRESENT REQUEST IS FOR COSINE
ZFLCOS  PZE     **,**,**        ZERO IF LAST REQUEST WAS FOR COSINE
COSDEL  PZE     **,**,**        COS(ARGDEL)
SINDEL  PZE     **,**,**        SIN(ARGDEL)
LASCOS  PZE     **,**,**        INITIAL = COS(ARGLO-ARGDEL)
LASSIN  PZE     **,**,**        INITIAL = SIN(ARGLO-ARGDEL)
 TEMP   PZE     **,**,**
        END
```

```
************************          PROGRAM LISTINGS          ************************
*    SETAPT             *                                   *    SETAPT             *
************************                                    ************************
 REFER TO                                                    REFER TO
     INDEX                                                       INDEX
```

```
************************          PROGRAM LISTINGS          ************************
*    SETEST             *                                   *    SETEST             *
************************                                    ************************
 REFER TO                                                    REFER TO
     INDEX                                                       INDEX
```

```
*     SETINO (SUBROUTINE)                 9/8/64   LAST CARD IN DECK IS NO. 0091
*     LABEL
CSETINO
      SUBROUTINE SETINO(ITAPE, ZIFNEW, NRECS, ERR)
C
C
C                    ----ABSTRACT----
C
C TITLE - SETINO
C        INITIALIZE FOR ADDING TO AN INDATA-OUDATA TAPE
C
C              SETINO  POSITIONS A TAPE FOR RECEIVING RECORDS WRITTEN
C              BY SUBROUTINE  OUDATA.  IF THE TAPE IS FRESH,  SETINO
C              MERELY REWINDS IT.  IF THE TAPE CONTAINS RECORDS
C              PREVIOUSLY WRITTEN BY SUBROUTINE  OUDATA  AND TERMINATED
C              BY A RECORD WITH ZERO RECORD NUMBER,  SETINO  REWINDS
C              THE TAPE AND SEARCHES FOR THE TERMINATING RECORD.
C              THE TAPE IS THEN LEFT POSITIONED SO THAT THE NEXT
C              RECORD WRITTEN BY  OUDATA  WILL REPLACE THE TERMINATING
C              RECORD.  A RECORD COUNT IS ALSO FURNISHED.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  MAIN FRAME PLUS 1 TAPE DRIVE
C STORAGE    - 84 REGISTERS
C SPEED      - DEPENDS ON INITIAL TAPE POSITION AND SEARCH TIME.
C AUTHOR     - S.M. SIMPSON, JUNE 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - XLIMIT, FSKIP
C       AND FORTRAN SYSTEM ROUTINES -  (RWT), (TSB), (RLR)
C
C FORTRAN USAGE
C     CALL SETINO(ITAPE, ZIFNEW, NRECS, ERR)
C
C
C INPUTS
C
C   ITAPE      IS THE LOGICAL TAPE NUMBER
C              MUST EXCEED ZERO AND BE  LSTHN=  20
C
C   ZIFNEW     =      0.0  IMPLIES NOTHING ON  ITAPE  IS TO BE SAVED.
C              NOT=  0.0  IMPLIES ITAPE CONTAINS INDATA-OUDATA FORMAT
C                 RECORDS, ALL OF WHICH ARE TO BE SAVED.
C
C
C OUTPUTS      THE TAPE IS POSITIONED AS DESCRIBED IN ABSTRACT.
C
C   NRECS      = 0  IF ZIFNEW = 0.  OTHERWISE,
C              = NUMBER OF RECORDS PASSED OVER IN SEARCH FOR
C                 TERMINATING RECORD, NOT COUNTING THE TERMINATING
C                 RECORD.
C
C   ERR        = 0.0  IF ALL OK
C              = 1.0  IF  ITAPE  IS ILLEGAL, IN WHICH CASE NO TAPE
C                 MOVEMENT IS ATTEMPTED AND  NRECS  IS NOT DISTURBED.
C
C
C EXAMPLES
C
C 1. INPUTS  - ASSUME A  6  RECORD INDATA-OUDATA TAPE HAS BEEN CREATED
C              ON LOGICAL  9  BY THE FOLLOWING SEQUENCE.
C                    REWIND 9
C                    DO 10  I=1,10
C                 10 X(I) = FLOATF(I)
C                    DO 20  I=1,5
C                    IRECNO = I
C                 20 CALL OUDATA(9,IRECNO,10,X,1)
C                    IRECNO = 0
C                    CALL OUDATA(9,IRECNO,1,DUMMY,1)
C
C   USAGE   -        CALL SETINO(9,1.0,NRECS,ERR)
C
C   OUTPUTS - NRECS=5,  ERR=0.0,  AND TAPE IS POSITIONED TO REWRITE
```

SETINO (PAGE 2)

```
      C                  THE 6-TH RECORD.
      C
      C PROGRAM FOLLOWS BELOW
      C
            ERR = XABSF(XLIMITF(ITAPE,1,20))
            IF (ERR)     9999,10,9999
       10   NRECS = 0
            REWIND ITAPE
            IF (ZIFNEW)  20,9999,20
       20   READ TAPE ITAPE, IRECNO
            CALL FSKIP(ITAPE, 1)
            IF (IRECNO)  30,60,30
       30   NRECS = NRECS+1
            GO TO 20
       60   CALL FSKIP(ITAPE, -1)
       9999 RETURN
            END
```

```
*      SETK  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0189
*      FAP
*SETK
       COUNT   200
       LBL     SETK
       ENTRY   SETK   ( C,X1,X2,...,XN)
       ENTRY   SETKS  (C1,X1,C2,X2,...,CN,XN)
       ENTRY   SETVEC ( X,C1,C2,...,CN)
*
*                    ----ABSTRACT----
*
* TITLE - SETK  WITH SECONDARY ENTRIES SETKS AND SETVEC
*        SET VARIABLES OR VECTORS TO GIVEN VALUES
*
*              SETK IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
*              WHICH SETS EACH OF ITS ARGUMENTS BEYOND THE FIRST EQUAL
*              TO THE FIRST ARGUMENT, THE MODE OF WHICH IS ARBITRARY.
*
*              SETKS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE,
*              REQUIRING AN EVEN NO. OF ARGUMENTS WHICH ARE TREATED IN
*              PAIRS.  THE SECOND ARGUMENT OF EACH PAIR IS SET EQUAL
*              TO THE FIRST ARGUMENT OF THE PAIR, THE MODE OF WHICH IS
*              ARBITRARY.
*
*              SETVEC IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
*              WHOSE FIRST ARGUMENT IS CONSIDERED A VECTOR OF LENGTH
*              EQUAL TO THE NO. OF ADDITIONAL ARGUMENTS PRESENT.  THE
*              ELEMENTS OF THIS VECTOR ARE SET SEQUENTIALLY EQUAL TO
*              THE REMAINING ARGUMENTS, WHOSE MODES ARE ARBITRARY.
*
*              THE NUMBER OF ARGUMENTS WHICH MAY BE USED IS NOT
*              RESTRICTED.
*
* LANGUAGE   - FAP SUBROUTINES, FORTRAN-II COMPATIBLE
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 37 REGISTERS
* SPEED      - SETK   TAKES  14 + 13*N1  MACHINE CYCLES,
*                     WHERE  N1+1 IS THE ARGUMENT COUNT.
*              SETKS  TAKES  11 + 16*N2  MACHINE CYCLES,
*                     WHERE  2*N2 IS THE ARGUMENT COUNT.
*              SETVEC TAKES  13 + 21*N3  MACHINE CYCLES,
*                     WHERE  N3+1 IS THE ARGUMENT COUNT.
* AUTHOR     - S.M. SIMPSON JR., SEPTEMBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE OF SETK
*     CALL SETK (C,X1,X2,X3,...,XN)
*                 WHERE N SHOULD EXCEED ZERO, AND THE MODES OF THE
*                 ARGUMENT NAMES ARE ARBITRARY.
*
* INPUTS
*
*    C         IS A QUANTITY IN ANY MODE
*
* OUTPUTS      PROGRAM RETURNS CONTROL WITH NO OUTPUT IF THE ARGUMENT
*              COUNT IS 1 (N=0).
*
*    X1        IS SET = C
*    X2        IS SET = C
*       ETC
*    XN        IS SET = C
*
*
* FORTRAN USAGE OF SETKS
*     CALL SETKS (C1,X1,C2,X2,C3,X3,...,CN,XN)
*                 WHERE N SHOULD EXCEED ZERO, AND WHERE THE MODES OF
*                 THE ARGUMENT NAMES ARE ARBITRARY.
*
* INPUTS
*
*    C1        IS A QUANTITY IN ANY MODE
```

```
*    C2         IS A QUANTITY IN ANY MODE
*       ETC
*    CN         IS A QUANTITY IN ANY MODE
*
* OUTPUTS       AN IMPROPER RETURN RESULTS IF THE ARGUMENT COUNT IS ZERO
*               OR NOT EVEN.
*
*    X1         IS SET = C1
*    X2         IS SET = C2
*       ETC
*    XN         IS SET = CN
*
*               EQUIVALENCE(CM,XL) IS PERMITTED.  BEHAVIOUR DEPENDS ON
*               THE FACT THAT THE SETTING SEQUENCE IS X1,X2,...,XN.
*
*
* FORTRAN USAGE OF SETVEC
*     CALL SETVEC(X,C1,C2,C3,...,CN)
*                  WHERE N SHOULD EXCEED ZERO, AND WHERE THE MODES OF
*                  THE ARGUMENT NAMES ARE ARBITRARY
*
* INPUTS
*
*    C1         IS VALUE FOR SETTING X(1)
*       ETC
*    CN         IS VALUE FOR SETTING X(N)
*
* OUTPUTS       PROGRAM RETURNS CONTROL WITH NO OUTPUTS IF THE ARGUMENT
*               COUNT IS 1.
*
*    X(1,2,...,N)  IS SET = C1,C2,...,CN
*
*               EQUIVALENCE (ANY TWO ARGUMENTS) IS PERMITTED, BEHAVIOUR
*               DEPENDING ON THE FACT THAT THE SETTING SEQUENCE IS
*                  X(1),X(2),...,X(N)
*
*
* EXAMPLES
*
* 1. EXAMPLES OF SETK
*    USAGE   -         CALL SETK(4.0,A,B,C,D,E,F,G,H)
*                      CALL SETK(3,I,J,K,L,M)
*                      CALL SETK(M,N)
*    OUTPUTS - A=B=C=D=E=F=G=H = 4.0
*              I=J=K=L=M      = 3
*              N              = 3
*
* 2. EXAMPLES OF SETKS
*    USAGE   -         CALL SETKS( 2.,A,3,I,4.,B,5,J,6,K)
*                      CALL SETKS( 3.1416,X,1963,L,X,Y,X,Z,L,M)
*                      CALL SETKS( 5.,C)
*    OUTPUTS - A=2.0, I=3, B=4.0, J=5, K=6
*              X=Y=Z = 3.1416,  L=M = 1963
*              C = 5.
*
* 3. EXAMPLES OF SETVEC
*    USAGE   -         CALL SETVEC( X,9.,7.,8.,14.)
*                      CALL SETVEC( I,19630,2,I,I,I(2),5)
*                      CALL SETVEC( J,5)
*    OUTPUTS - X(1...4) = 9., 7., 8., 14.
*              I(1...6) = 19630, 2, 19630, 19630, 2, 5
*              J = 5
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0               ORIGINAL XR4
       BCI     1,SETK
* PRINCIPAL ENTRY.  SETK(C,X1,X2,...,XN)
 SETK  LDQ*    1,4             C IN MQ (STAYS THERE)
       CLA     TRAK
 SXD4  SXD     SETK-2,4
       TXI     SETUP,4,-1      (SET TO START WITH X1)
* SECOND ENTRY.  SETKS(C1,X1,C2,X2,...,CN,XN)
```

****
*   SETK   *
****
(PAGE 3)

```
 SETKS SXD      SETK-2,4
       CLA      TRAKS
       TRA      SETUP
* THIRD ENTRY.  SETVEC(X,C1,C2,...,CN)
SETVEC CLA      1,4            A(X)
       STA      STO
       CLA      TRASV
       TRA      SXD4           (ADJUST XR4 TO START WITH C1)
* MERGE POINT
 SETUP STA      TRA
* LOOP.  CHECK IF 1,4 IS A TSX   X,0
 CAL   CAL      1,4
       ANA      MASK           KNOCK OUT ADDRESS
       LAS      TSXZ
       TRA      LEAVE
 TRA   TRA      **             **=MOREK  OR  MOREKS  OR  MORESV
* EXIT AT END OF ARGUMENT STRING
 LEAVE TRA      1,4
* SETK INSERT
 MOREK STQ*     1,4            C TO XJ
       TXI      CAL,4,-1       BACK FOR NEXT X
* SETKS INSERT
MOREKS CLA*     1,4            CJ
       STO*     2,4            TO XJ
       TXI      CAL,4,-2       BACK FOR NEXT PAIR
* SETVEC INSERT
MORESV CLA*     1,4                                  CJ
 STO   STO      **             **=A(X)               TO X(J)
       CLA      STO
       SUB      K1
       STO      STO
       TXI      CAL,4,-1
* CONSTANTS
 TRAK  TRA      MOREK
 TRAKS TRA      MOREKS
 TRASV TRA      MORESV
 MASK  OCT      777777700000
 TSXZ  TSX      0,0
 K1    PZE      1
       END
```

```
*     SETK  -II (SUBROUTINE)            9/29/64   LAST CARD IN DECK IS NO. 0072
*     LABEL
CSETK  -II
      SUBROUTINE SETK(C)
C
C                   ----ABSTRACT----
C
C TITLE - SETK  -II
C          SET ANY NO. OF VARIABLES EQUAL TO A SINGLE VALUE (FXD OR FLTG)
C
C              SETK IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
C              WHICH SETS EACH OF ITS ARGUMENTS BEYOND THE FIRST EQUAL
C              TO THE FIRST ARGUMENT, THE MODE OF WHICH IS ARBITRARY.
C
C              THIS VERSION OF SETK (SETK-II) IS THE FORTRAN EQUIVALENT
C              OF THE FAP SUBROUTINE OF THE SAME NAME.
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 63 REGISTERS
C SPEED      - ABOUT 700+165*N MACHINE CYCLES, WHERE N+1 * TOTAL
C              ARGUMENT COUNT.
C AUTHOR     - S.M. SIMPSON, AUGUST 1963
C
C                   ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - SETUP, STORE, RETURN
C        AND FORTRAN SYSTEM ROUTINES - (NONE)
C
C FORTRAN USAGE
C     CALL SETK (C,X1,X2,X3,...,XN)
C                 WHERE N SHOULD EXCEED ZERO, AND THE MODES OF THE
C                 ARGUMENT NAMES ARE ARBITRARY.
C
C INPUTS
C
C    C         IS A QUANTITY IN ANY MODE
C
C OUTPUTS      PROGRAM RETURNS CONTROL WITH NO OUTPUT IF THE ARGUMENT
C              COUNT IS LESS THAN 2.
C
C    X1        IS SET = C
C    X2        IS SET = C
C        ETC
C    XN        IS SET = C
C
C              EQUIVALENCE (ANY TWO ARGUMENTS) IS PERMITTED BUT SERVES
C              NO PARTICULAR FUNCTION.
C
C EXAMPLES
C
C 1. USAGE   -        CALL SETK(4.0,A,B,C,D,E,F,G,H)
C                     CALL SETK(3,I,J,K,L,M)
C                     CALL SETK(M,N)
C                     CALL SETK(1.)
C                     CALL SETK
C
C    OUTPUTS - A=B=C=D=E=F=G=H = 4.0
C              I=J=K=L=M       = 3
C              N               = 3
C                 AND NO OUTPUT RESULTS FROM LAST TWO CALLS
C
C PROGRAM FOLLOWS BELOW
C
C ACQUIRE ARGUMENT COUNT AND EXIT IF LESS THAN 2
      CALL SETUP(LOCALL,NARGS,XR1,XR2)
      IF (NARGS-1) 9999,9999,10
C SET X1...XN
 10   DO 20 NUMARG=2,NARGS
 20   CALL STORE(C,LOCALL,NUMARG,1)
C EXIT
 9999 CALL RETURN(LOCALL,XR1,XR2)
      END
```

```
*       SETKP  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0123
*       FAP
*SETKP
        COUNT   100
        LBL     SETKP
        ENTRY   SETKP  (C1,X11,X12,...,X1N1,STOP, C2,X21,X22,...,X2N2,
*                        STOP, ......, CM,XM1,XM2,...,XMNM)
        ENTRY   SETVCP (X1,C11,C12,...,C1N1,STOP, X2,C21,C22,...,C2N2,
*                        STOP, ......, XM,CM1,CM2,...,CMNM)
*
*                      ----ABSTRACT----
*
* TITLE - SETKP  WITH SECONDARY ENTRY SETVCP
*         PLURALIZED FORMS OF SUBROUTINES SETK AND SETVEC
*
*               SETKP IS THE PLURAL FORM OF SUBROUTINE SETK.  SETKP HAS
*               A VARIABLE NUMBER OF ARGUMENTS SEPARATED INTO AN
*               ARBITRARY NUMBER OF GROUPS BY A FENCE-TYPE ARGUMENT.
*               EACH SUCH GROUP REPRESENTS THE ARGUMENTS OF ONE CALL SETK
*               STATEMENT.  SUBROUTINE SETK IS CALLED SUCCESSIVELY, ONCE
*               FOR EACH GROUP.
*
*               SETVCP PERFORMS THE ANALOGOUS FUNCTION FOR SUBROUTINE
*                  SETVEC.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 40 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON JR.,   SEPTEMBER 1963
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - SETK,SETVEC
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE OF SETKP
*
*     CALL SETKP(C1,X11,X12,...,X1N1,STOP,C2,X12,X22,...,X2N2,STOP,
*    1             ......,CM,XM1,XM2,...,XMNM)
*
* WHERE STOP = OCT 777777712345, IS EQUIVALENT TO
*     CALL SETK(C1,X11,X12,...,X1N1)
*     CALL SETK(C2,X21,X22,...,X2N2)
*          ETC
*     CALL SETK(CM,XM1,XM2,...,XMNM)
*
* FORTRAN USAGE OF SETVCP
*
*     CALL SETVCP(X1,C11,C12,...,C1N1,STOP,X2,C21,C22,...,C2N2,STOP,
*    1              ......,XM,CM1,CM2,...,CMNM)
* IS EQUIVALENT TO
*     CALL SETVEC(X1,C11,C12,...,C1N1)
*     CALL SETVEC(X2,C21,C22,...,C2N2)
*          ETC
*     CALL SETVEC(XM,CM1,CM2,...,CMNM)
*
* SEE WRITEUPS OF SETK AND SETVEC FOR INPUT-OUTPUT DETAILS.
*
* EXAMPLES
*
* 1. USAGE    - B       STOP = 777777712345
*                       CALL SETKP(1.,X,Y,Z,STOP,2.,U,V,W,STOP,7,IX)
*                       CALL SETVCP(A,1.,2.,3.,STOP,B,7.)
*    OUTPUTS - X=Y=Z = 1.    U=V=W = 2.    IX = 7
*              A(1...3) = 1., 2., 3.   B(1) = 7.
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR CONTAINS SETK, SETVEC
        HTR     0               XR1
        HTR     0               XR4
        BCI     1,SETKP
```

```
     * PRINCIPAL ENTRY.  SETKP(C1,X11,X12,...,X1N,STOP,
     *                         C2,X21,X22,...,X2N,STOP,......
     *                         CM,XM1,XM2,...,XMN)
      SETKP CLA     TSXSK
            TRA     SETUP
     * SECOND ENTRY.  SETVCP(X1,C11,C12,...,C1N,STOP,
     *                       X2,C21,C22,...,C2N,STOP,......
     *                       XM,CM1,CM2,...,CMN)
     SETVCP CLA     TSXSV
      SETUP STA     TRASUB
            SXD     SETKP-2,4
            SXD     SETKP-3,1
     * USE XR1 TO SCAN FOR STOP OR END OF SEQUENCE,
     * HOLDING XR4 FOR FOOLING SETK OR SETVEC.
      NEXT  PXA     0,4
            PAX     0,1
      CAL   CAL     1,1            TSX ARG,0  OR SOMETHING ELSE
            ANA     AMASK
            LAS     TSXZ
            TRA     NOARG          NO
            TRA     ISARG          YES
     * IF NOT AN ARGUMENT, ENTER SUBROUTINE WITHOUT REPLACING 1,1
      NOARG TRA     GOOUT
     * IT IT IS AN ARGUMENT, CHECK FOR STOP
      ISARG CAL*    1,1
            LAS     STOP
            TRA     *+2            NO
            TRA     ISTOP          YES
            TXI     CAL,1,-1       MORE ARGUMENTS
     * IF STOP IS FOUND, REPLACE IT AND GO OPERATE
      ISTOP CLA     TRABAK
            STO     1,1
     * GO OPERATE SETK OR SETVEC.  (RETURNS ONLY IF 1,1 WAS A STOP)
      GOOUT SXA     BACK,1
            LXD     SETKP-3,1
     TRASUB TRA     **             ** = A(TTR SETK) OR A(TTR SETVEC)
     * IF IT COMES BACK, RESTORE XR4 TO OLD XR1, RESTORE STOP,
     * INDEX XR4, AND RETURN FOR NEXT CALL.
      BACK  AXT     **,4           ** = XR1 BEFORE SUBROUTINE
            CLA     TSXSTP
            STO     1,4
            TXI     NEXT,4,-1
     * CONSTANTS
      TSXSK TSX     $SETK,4
      TSXSV TSX     $SETVEC,4
     TRABAK TRA     BACK
     TSXSTP TSX     STOP,0
      STOP  CCT     777777712345
      TSXZ  TSX     0,0
      AMASK OCT     777777700000
            END
```

```
************************
*    SETKS             *
************************
 REFER TO
     SETK
```


```
************************
*    SETKS             *
************************
 REFER TO
     SETK
```

```
*     SETKS -II (SUBROUTINE)          9/29/64   LAST CARD IN DECK IS NO. 0085
*     LABEL
CSETKS -II
      SUBROUTINE SETKS
C
C                    ----ABSTRACT----
C
C TITLE - SETKS -II
C        SET ANY NO. OF VARIABLES EQUAL TO SEPARATE VALUES (FXD OR FLTG)
C
C              SETKS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE,
C              REQUIRING AN EVEN NO. OF ARGUMENTS WHICH ARE TREATED IN
C              PAIRS.  THE SECOND ARGUMENT OF EACH PAIR IS SET EQUAL
C              TO THE FIRST ARGUMENT OF THE PAIR,  THE MODE OF WHICH IS
C              ARBITRARY.
C
C              THIS VERSION OF SETKS (SETKS-II) IS THE FORTRAN
C              EQUIVALENT OF THE FAP SUBROUTINE CF THE SAME NAME.
C
C LANGUAGE    - FORTRAN II SUBROUTINE
C EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE     - 91 REGISTERS
C SPEED       - ABOUT 750+250*N MACHINE CYCLES, WHERE 2*N = TOTAL
C              ARGUMENT COUNT.
C AUTHOR      - S.M. SIMPSON, AUGUST 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - SETUP, ARG, STORE, RETURN
C       AND FORTRAN SYSTEM ROUTINES - (NONE)
C
C FORTRAN USAGE
C     CALL SETKS (C1,X1,C2,X2,C3,X3,....,CN,XN)
C                 WHERE N SHOULD EXCEED ZERO, AND WHERE THE MODES OF
C                 THE ARGUMENT NAMES ARE ARBITRARY.
C
C INPUTS
C
C    C1        IS A QUANTITY IN ANY MODE
C    C2        IS A QUANTITY IN ANY MODE
C       ETC
C    CN        IS A QUANTITY IN ANY MODE
C
C OUTPUTS      PROGRAM RETURNS CONTROL WITH NO OUTPUT IF THE ARGUMENT
C              COUNT IS ZERO OR IS NOT EVEN.
C
C    X1        IS SET = C1
C    X2        IS SET = C2
C       ETC
C    XN        IS SET = CN
C
C              EQUIVALENCE(CM,XL) IS PERMITTED. BEHAVIOUR DEPENDS ON
C              FACT THAT SETTING SEQUENCE IS X1,X2,....,XN.
C
C EXAMPLES
C
C 1. USAGE   -        CALL SETKS( 2.,A,3,I,4.,B,5,J,6,K)
C                     CALL SETKS( 3.1416,X,1963,L,X,Y,X,Z,L,M)
C                     CALL SETKS( 5.,C)
C                     D=0.
C                     CALL SETKS( 6.,D,7.)
C                     CALL SETKS( 8.)
C                     CALL SETKS
C    OUTPUTS - A=2.0, I=3, B=4.0, J=5, K=6
C              X=Y=Z = 3.1416,  L=M = 1963
C              C = 5.
C              D = 0.  (NO OUTPUTS SINCE ODD NO. ARGUMENTS)
C                 LIKEWISE LAST TWO CALLS PRODUCE NO OUTPUT
C
C PROGRAM FOLLOWS BELOW
C
C
C ACQUIRE ARGUMENT COUNT AND CHECK IT
      CALL SETUP(LOCALL,NARGS,XR1,XR2)
      NPAIRS=NARGS/2
```

****
* SETKS -II *
****
(PAGE 2)

```
      IF (NPAIRS) 9999,9999,10
   10 IF (2*NPAIRS-NARGS) 9999,20,9999
C SET X1,X2,...,XN
   20 DO 30 IXPAIR=1,NPAIRS
      NUMARG=2*IXPAIR-1
      C=ARGF(LOCALL,NUMARG,1)
      NUMARG=NUMARG+1
   30 CALL STORE(C,LOCALL,NUMARG,1)
C EXIT
 9999 CALL RETURN(LOCALL,XR1,XR2)
      END
```

```
*      SETKV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0074
*      FAP
*SETKV
       COUNT   100
       LBL     SETKV
       ENTRY   SETKV  (C,LX,X)
*
*                      ----ABSTRACT----
*
* TITLE - SETKV
*        SET ALL ELEMENTS OF VECTOR EQUAL TO A CONSTANT (ANY MODE)
*
*              SETKV  SETS X(1...LX) = C  WHERE C IS ANY MODE.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 15  REGISTERS
* SPEED      - 24 + 4*LX MACHINE CYCLES, WHERE LX = LENGTH OF VECTOR
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL SETKV(C,LX,X)
*
* INPUTS
*
*    C         IS A QUANTITY IN ANY MODE
*
*    LX        IS LENGTH OF VECTOR, GRTHN=1.
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF LX LSTHN 1.
*
*    X(I)      I=1...LX  EQUALS  C  (UNCHANGED MODE)
*              EQUIVALENCE(C, SOME X(I))  PERMITTED.
*
* EXAMPLES
*
* 1. USAGE   -       CALL SETKV (3.0,10,X)
*                    CALL SETKV (3,5,IX)
*                    CALL SETKV (3.0,1,Y)
*                    IY=0
*                    CALL SETKV (3,0,IY)
*    OUTPUTS - X(1...10)=3.0  IX(1...5)=3  Y=3.0
*              IY=0     (NO OUTPUT FROM LAST CALL)
*
* 2. INPUTS  - X(1...7)=1., 2.,..., 7.
*    USAGE           CALL SETKV (X(3),7,X)
*    OUTPUTS   X(1...7)=3.0
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR     0                  XR4
       BCI     1,SETKV
* ONLY ENTRY.  SETKV(C,LX,X)
 SETKV SXD     SETKV-2,4
 K1    LDQ*    1,4           C TO MQ
       CLA     3,4
       ADD     K1                 A(X)+1
       STA     STORE
       CLA*    2,4                LX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* STORE LOOP
 STORE STQ     **,4          **=A(X)+1
       TIX     STORE,4,1
* EXIT
 LEAVE LXD     SETKV-2,4
       TRA     4,4
       END
```

```
*      SETKVS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0105
*      FAP
*SETKVS
       COUNT   100
       LBL     SETKVS
       ENTRY   SETKVS (C1,L1,X1,C2,L2,X2,...,CN,LN,XN)
*
*                    ----ABSTRACT----
*
* TITLE - SETKVS
*       SET ANY NO. OF VECTORS EQUAL TO SEPARATE VALUES (FXD OR FLTG)
*
*              SETKVS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE
*              WHOSE ARGUMENTS ARE TREATED IN TRIPLETS.  THE THIRD
*              ARGUMENT OF EACH TRIPLET IS CONSIDERED A VECTOR OF
*              LENGTH GIVEN BY THE SECOND ARGUMENT.  ALL ELEMENTS IN
*              THIS VECTOR ARE SET EQUAL TO THE FIRST ARGUMENT OF THE
*              TRIPLET WHOSE MODE IS ARBITRARY.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 25 REGISTERS
* SPEED      - 9 + 28*N + 4*L  MACHINE CYCLES,  WHERE  N = NO. OF
*                 VECTORS TO BE SET, AND L = THEIR TOTAL COMBINED LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*      AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*    CALL SETKVS(C1,L1,X1,C2,L2,X2,...,CN,LN,XN)
*                 WHERE N SHOULD EXCEED ZERO, AND WHERE THE MODES
*                 OF THE ARGUMENT NAMES C1...CN AND X1...XN ARE
*                 ARBITRARY.
*
* INPUTS
*
*   C1         IS THE VALUE TO WHICH X1(1...L1) ARE TO BE SET.
*
*   L1         SHOULD EXCEED ZERO
*     ETC
*   CN         IS THE VALUE TO WHICH XN(1...LN) ARE TO BE SET
*
*   LN         SHOULD EXCEED ZERO
*
* OUTPUTS      AN IMPROPER RETURN RESULTS IF THE ARGUMENT COUNT IS NOT
*              A MULTIPLE OF 3.
*
*   X1(1...L1)  ARE ALL SET = C1  IF L1 EXCEEDS ZERO
*     ETC
*   XN(1...LN)  ARE ALL SET = CN  IF LN EXCEEDS ZERO
*
*              IF ANY  L VALUE IS ZERO OR NEGATIVE THE CORRESPONDING
*              X VECTOR IS NOT DISTURBED
*
*              EQUIVALENCE (ANY TWO ARGUMENTS) IS PERMITTED WITH
*              BEHAVIOUR DEPENDING ON THE FACT THAT THE SETTING SEQUENCE
*              IS X1,X2,...,XN.
*
* EXAMPLES
*
* 1. USAGE   -        CALL SETKVS( 2.,10,A, 4,3,I, 7.,I,B)
*                     K=0
*                     CALL SETKVS( 9.,15,C, 5,0,K, 6,-1,K, 11.,1,D)
*   OUTPUTS - A(1...10) = 2., I(1...3) = 4, B(1...4) = 7.
*              C(1...15) = 9., K = 0 (ILLEGAL LX VALUES), D = 11.0
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
      HTR      0              XR1
      HTR      0              ORIGINAL XR4
```

```
************************
*    SETKVS            *
************************
                (PAGE  2)
```


```
        BCI     1,SETKVS
      * ONLY ENTRY.  SETKVS(C1,L1,X1,C2,L2,X2,...,CN,LN,XN)
      SETKVS SXD     SETKVS-2,4
        SXD     SETKVS-3,1
      * CHECK THAT 1,4 IS A TSX   X,0
       CAL    CAL     1,4            CJ
              ANA     MASK
              LAS     TSXZ
              TRA     LEAVE
              TRA     MORE
      * IF NOT, EXIT
       LEAVE  LXD     SETKVS-3,1
              TRA     1,4
      * IF SO, ENTER STORE LOOP PROVIDED LENGTH IS LEGAL
       MORE   CLA*    2,4            LJ
              TMI     BACK
              PDX     0,1
              TXL     BACK,1,0
              CLA     3,4
              ADD     K1             A(X)+1
              STA     STORE
       K1     CLA*    1,4            CJ
      * STORE LOOP
       STORE  STO     **,1           **=A(X)+1
              TIX     STORE,1,1
      * BACK FOR NEXT TRIPLET
       BACK   TXI     CAL,4,-3
      * CONSTANTS
       MASK   OCT     777777700000
       TSXZ   TSX     0,0
              END
```

```
*      SETLIN (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0094
*      FAP
*SETLIN
       COUNT   100
       LBL     SETLIN
       ENTRY   SETLIN ( BASE, DELTA, LX, X)
       ENTRY   XSTLIN (IBASE,IDELTA,LIX,IX)
*
*                   ----ABSTRACT----
*
* TITLE - SETLIN WITH SECONDARY ENTRY XSTLIN
*        SET FXD OR FLTG VECTOR EQUAL TO A LINEAR SEGMENT
*
*              SETLIN SETS A FLOATING LINE SEGMENT.
*              XSTLIN SETS A FIXED LINE SEGMENT.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 27 REGISTERS
* SPEED      - SETLIN   35 + 12.4*LX  MACHINE CYCLES
*              XSTLIN   37 +  8.0*LX      LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL SETLIN( BASE, DELTA, LX, X)
*     CALL XSTLIN(IBASE,IDELTA,LIX,IX)
*
* INPUTS
*
*    BASE      IS VALUE FOR X(1)
*    DELTA     IS INCREMENT FOR SUCCESSIVE VALUES OF X(I)
*    LX        IS DESIRED OUTPUT LENGTH. SHOULD EXCEED 0.
*
*    IBASE     IS VALUE FOR IX(1)
*    IDELTA    IS INCREMENT FOR SUCCESSIVE VALUES OF IX(I)
*    LIX       IS DESIRED OUTPUT LENGTH.  SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF  LX OR LIX LSTHN 1
*
*    X(I)      I=1...LX  IS   X(I)= BASE + (I-1)*DELTA
*
*    IX(I)     I=1...LIX  IS  IX(I)= IBASE + (I-1)*IDELTA
*
* EXAMPLES
*
* 1. INPUTS  - X3 = 0.0
*    USAGE   -         CALL SETLIN(0.,2.,5, X1)
*                      CALL XSTLIN( 0, 2,5,IX1)
*                      CALL SETLIN(2.,2.,1, X2)
*                      CALL SETLIN(2.,2.,0, X3)
*    OUTPUTS - X1(1...5) = 0., 2., 4., 6., 8.  IX1(1...5)= 0,2,4,6,8
*              X2(1) = 2.     X3 = 0.  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
       BCI     1,SETLIN
* PRINCIPAL ENTRY.  SETLIN(BASE,DELTA,LX,X)
SETLIN CLA     FAD
 SETUP STO     NEXT
       SXD     SETLIN-2,4
       CLA     4,4
       ADD     K1             A(X)+1
       STA     STORE
       CLA*    2,4            DELTA
       STO     TEMP
       CLA*    3,4            LX
       TMI     LEAVE
```

****
*   SETLIN   *
****
(PAGE 2)

```
        TZE     LEAVE
        STD     TXL
 K1     CLA*    1,4             BASE
        AXT     1,4
* LOOP
 STORE  STO     **,4            **=A(X)+1
 NEXT   NOP                     =  FAD TEMP OR ADD TEMP
        TXI     *+1,4,1
 TXL    TXL     STORE,4,**      **=LX
* EXIT
 LEAVE  LXD     SETLIN-2,4
        TRA     5,4
* SECONDARY ENTRY. XSTLIN(IBASE,IDELTA,LIX,IX)
XSTLIN  CLA     ADD
        TRA     SETUP
* CONSTANTS, TEMPORARIES
 FAD    FAD     TEMP
 ADD    ADD     TEMP
 TEMP   PZE     **,**,**        = DELTA
        END
```

```
*       SETLNS (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0123
*       FAP
*SETLNS
        COUNT   150
        LBL     SETLNS
        ENTRY   SETLNS (BASE1,DELTA1,LX1,X1,BASE2,DELTA2,LX2,X2,...,
*                      BASEN,DELTAN,LXN,XN)
*
*                          ----ABSTRACT----
*
* TITLE - SETLNS
*         SET LINEAR VECTORS, FIXED AND/OR FLOATING
*
*               SETLNS IS A VARIABLE-LENGTH-CALLING-SEQUENCE SUBROUTINE,
*               ONE CALL OF WHICH IS EQUIVALENT TO A SUCCESSION OF CALLS
*               OF SUBROUTINES SETLIN (WHICH SETS A FLOATING VECTOR EQUAL
*               TO A LINEAR SEGMENT) AND/OR XSTLIN (FOR FIXED POINT
*               LINEAR SEGMENTS).  THE ARGUMENTS OF SETLNS ARE DIVIDED
*               INTO GROUPS OF LENGTH FOUR, EACH GROUP REPRESENTING THE
*               FOUR ARGUMENTS OF A DESIRED CALL SETLIN OR XSTLIN
*               STATEMENT.  SETLNS DECIDES TO USE XSTLIN (SETLIN) IF THE
*               CONSTANT INCREMENT DELTA, INTERPRETED AS FIXED, IS LESS
*               THAN OR EQUAL TO (EXCEEDS) 10000, BUT XSTLIN IS ALWAYS
*               USED IF BIT 9 OF DELTA IS ZERO.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 39 REGISTERS
* SPEED      - ABOUT  85 + 12.4*LX  MACHINE CYCLES FOR EACH FLTG. VECTOR
*               PLUS   85 +  8.0*LX  MACHINE CYCLES FOR EACH FIXED VECTOR
*                     WHERE  LX = VECTOR LENGTH.
* AUTHOR     - S.M. SIMPSON JR.,   SEPTEMBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - SETLIN,XSTLIN
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE OF SETLNS
*
*     CALL SETLNS(BASE1,DELTA1,LX1,X1, BASE2,DELTA2,LX2,X2,...,
*    1             BASEN,DELTAN,LXN,XN)
*
* IS EQUIVALENT TO
*     CALL SETLIN(BASE1,DELTA1,LX1,X1)       (OR XSTLIN IF DELTA1
*                                                      LSTHN 10001)
*     CALL SETLIN(BASE2,DELTA2,LX2,XI)       (OR XSTLIN IF DELTA2
*                                                      LSTHN 10001)
*          ETC
*     CALL SETLIN(BASEN,DELTAN,LXN,XN)       (OR XSTLIN IF DELTAN
*                                                      LSTHN 10001)
*
* SEE WRITEUPS OF SETLIN AND XSTLIN FOR INPUT-OUTPUT DETAILS
*
* EXAMPLES
*
* 1. ORDINARY CASES
*    USAGE   -        CALL SETLNS(1.,1.,5,X1, 2,1,3,IX2, 3,1,1,IX3)
*    OUTPUTS - X1(1...5)=1.,2.,3.,4.,5.   IX2(1...3)=2,3,4   IX3=3
*
* 2. MORE UNUSUAL CASES
*    INPUTS  - OCTK=OCT 100000000000 (EXCEEDS 10000, DECIMAL, BUT BIT 9
*                                    IS ZERO)
*    USAGE   -        CALL SETLNS(20000,10000,3,IX4, 0.,.0000000001,3,X5,
*                    1  0,OCTK,3,X6)
*    OUTPUTS - IX4(1...3)=20000,30000,40000    (CASE OF UPPER LIMIT ON
*                                              FIXED DELTA)
*               X5(1...3)=0.,.0000000001,.0000000002   (SMALL FLTG DELTA)
*               X6(1...3)=0, OCT100000000000, OCT200000000000   (FIXED
*                                                         OPERATIONS)
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS SETLIN,XSTLIN
```

```
       HTR     0             XR4
       BCI     1,SETLNS
* ONLY ENTRY.  SETLNS(BASE1,DELTA1,LX1,X1,BASE2,DELTA2,LX2,X2,...,
*                     BASEN,DELTAN,LXN,XN)
SETLNS SXD     SETLNS-2,4
* SET RETURN LINKAGE
 NEXT  CLA     5,4           1 PAST 4-GROUP
       STO     SAVNXT
       CLA     TRABAK
       STO     5,4
* DECIDE WHETHER TO USE SETLIN OR XSTLIN BY TESTING DELTA
       CLA*    2,4
* WE ASSUME FIXED POINT IF MAGNITUDE(AS FORTRAN-II INTEGER) LSTHN 10001
       SSP
       CAS     XBIGST
       NOP
       TRA     FLTG          FLOATING, MAYBE
* USE XSTLIN IF FIXED POINT
 FXD   CLA     XST
       TRA     GOOUT
* USE SETLIN IF MAGNITUDE GRTHN= 10001, UNLESS BIT9=0
 FLTG  ANA     B9MASK
       TZE     FXD
       CLA     SET
* GO SET THE LINE
 GOOUT STA     TRAOUT
       SXA     BACK,4
TRAOUT TRA     **            **=$SETLIN OR $XSTLIN
 BACK  AXT     **,4          **=XR4 BEFORE SETLIN
* RESTORE AND CHECK FOR END OF STRING (NON TSX X,0)
       CAL     SAVNXT
       SLW     5,4
       TXI     *+1,4,-4
       ANA     AMASK
       LAS     TSXZ
       TRA     *+2           NO MORE
       TRA     NEXT          MORE
* EXIT
       TRA     1,4           NO MORE
* CONSTANTS TEMPORARIES
TRABAK TRA     BACK
 AMASK OCT     777777700000
 TSXZ  TSX     0,0
XBIGST PZE     0,0,10001
 XST   TSX     $XSTLIN,4
 SET   TSX     $SETLIN,4
B9MASK OCT     000400000000  EXTRACTS BIT 9
SAVNXT PZE     **,**,**
       END
```

```
*      SEVRAL (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0948
*      FAP
*SEVRAL
       COUNT    300
       LBL      SEVRAL
       ENTRY    SEVRAL (SUBRUA,A1,...,ANA,SUBRUB,B1,...,BNB,......)
       ENTRY    PLURAL (SUBROU,A1,A2,...,AN,B1,B2,...,BN,......)
*
*                   ----ABSTRACT----
*
* TITLE - SEVRAL WITH SECONDARY ENTRY PLURAL, PSEUDO ENTRIES DO, IF
*         OPERATE SEVERAL SUBROUTINES OR ONE SUBROUTINE REPEATEDLY
*
*               SEVRAL IS A VARIABLE LENGTH CALLING SEQUENCE SUBROUTINE
*               WHOSE ARGUMENTS ARE DIVIDED INTO VARIBLE LENGTH GROUPS.
*               THE FIRST ARGUMENT WITHIN EACH GROUP IS THE PROXY NAME,
*               ESTABLISHED BY A PRIOR CALL LOCATE STATEMENT, OF A SUB-
*               ROUTINE TO BE OPERATED BY SEVRAL, AND THE REMAINING
*               ARGUMENTS IN THE GROUP ARE THOSE OF THAT SUBROUTINE.
*               SEVRAL THUS OPERATES A GROUP OF SUBROUTINES SEQUENTIALLY.
*
*               THE NUMBER OF ARGUMENTS ASSOCIATED WITH EACH PROXY
*               NAME IS ASSUMED BY SEVRAL TO BE THE SAME AS THE ARGUMENT
*               COUNT OF THE CALL SUBRU STATEMENT FOLLOWING ITS DEFINING
*               CALL LOCATE STATEMENT, EXCEPT THAT IF THIS COUNT IS
*               ZERO THEN SEVRAL COMPUTES THE NUMBER OF ARGUMENTS BY
*               SCANNING DOWN THE ARGUMENTS IN THE GROUP UNTIL THE NEXT
*               LEGITIMATE PROXY NAME APPEARS (OR TILL THE END OF
*               SEVRAL*S ARGUMENTS IS REACHED).  THIS SCHEME ALLOWS
*               VARIABLE LENGTH CALLING SEQUENCE SUBROUTINES TO BE
*               OPERATED BY SEVRAL.
*
*               PLURAL ALLOWS THE REPEATED OPERATION OF THE SAME
*               SUBROUTINE ON SUCCESSIVE ARGUMENT GROUPS.  THE FIRST
*               ARGUMENT OF SEVRAL IS THE SUBROUTINE PROXY NAME, AND
*               THE REMAINING ARGUMENTS ARE A SEQUENCE OF EQUAL-LENGTH
*               BLOCKS GIVING THE SUCCESSIVE ARGUMENT GROUPS.  PLURAL
*               ASSUMES THE NUMBER OF ARGUMENTS PER GROUP TO BE THE SAME
*               AS THAT OF THE CALL SUBROU STATEMENT FOLLOWING ITS
*               DEFINING CALL LOCATE STATEMENT.
*
*               THE ARGUMENT COUNTS FOR SEVRAL AND PLURAL MAY BE
*               ARBITRARILY LARGE.
*
*               DO IS A PSEUDO-SUBROUTINE WITH FUNCTIONS SIMILAR TO A
*               FORTRAN DO STATEMENT, BUT OPERATING WITHIN THE CONFINES
*               OF A CALL SEVRAL STATEMENT.  LOOPS WITHIN LOOPS ARE
*               NOT PERMITTED.
*
*               IF IS A PSEUDO-SUBROUTINE WITH FUNCTIONS SIMILAR TO A
*               FORTRAN IF STATEMENT, BUT OPERATING WITHIN THE CONFINES
*               OF A CALL SEVRAL STATEMENT.  PSEUDO-IF STATEMENTS ARE
*               NOT PERMITTED INSIDE PSEUDO-DO LOOPS.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 416 REGISTERS
* SPEED      - SEVRAL TAKES AT LEAST 1500 MACHINE CYCLES AND ALSO
*                  ADDS A MIMIMUM OF 500 MACHINE CYCLES TO THE TIME
*                  REQUIRED BY EACH SUBROUTINE OPERATED.
*               PLURAL ADDS A MINIMUM OF 2000 MACHINE CYCLES TO THE
*                  TIME REQUIRED FOR THE FIRST OPERATION OF THE
*                  SUBROUTINE AND A MINIMUM OF 100 FOR EACH
*                  ADDITIONAL OPERATION.
*               DO REQUIRES RELATIVELY NEGLIGIBLE TIME FOR THE LOOP
*                  CONTROL LOGIC, AND IS OTHERWISE THE SAME AS SEVRAL.
*               IF REQUIRES A MINIMUM OF 400*J MACHINE CYCLES WHERE
*                  J-1 IS THE NUMBER OF SUBROUTINES BYPASSED
* AUTHOR     - S.M. SIMPSON JR., SEPT 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - LOCATE, WHERE
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
```

```
* FORTRAN USAGE OF SEVRAL
*     CALL LOCATE(SUBRUA,SUBRUB,...,SUBRUZ)
*     CALL SUBRA (R1,R2,...,RNA)          (ARGUMENT LIST OPTIONAL)
*     CALL SUBRB (R1,R2,...,RNB)          (ARGUMENT LIST OPTIONAL)
*        .
*        .
*     CALL SUBRZ (R1,R2,...,RNZ)          (ARGUMENT LIST OPTIONAL)
*        .
*        .
*        .
*     CALL SEVRAL(SUBRUA,A1,A2,...,ANA, SUBRUB,B1,B2,...,BNB,
*    1            ......,SUBRUZ,Z1,Z2,...,ZNZ)
*
*     WHERE
*       1.  IF AN OPTIONAL ARGUMENT LIST IS PRESENT IT MUST BE THE
*           SAME LENGTH AS THE CORRESPONDING LIST IN THE CALL SEVRAL
*           STATEMENT.
*       2.  THE ORDERING OF ARGUMENTS IN THE CALL LOCATE STATEMENT
*           NEED NOT MATCH THE ORDERING OF SUBROUTINES IN THE CALL
*           SEVRAL STATEMENT.
*       3.  NONE OF THE SUBROUTINES TO BE OPERATED MAY USE INFORMATION
*           BEYOND THE END OF ITS CALLING SEQUENCE.
*       4.  IF THE PSEUDO-SUBROUTINES DO AND IF APPEAR AS SUBROUTINES
*           TO BE OPERATED BY SEVRAL, AS DESCRIBED BELOW, THEY NEED
*           NOT APPEAR IN THE CALL LOCATE STATEMENT.
*       5.  SEVRAL*S SECONDARY ENTRY PLURAL MAY NOT APPEAR
*           AS ONE OF THE SUBROUTINES TO BE OPERATED BY SEVRAL.
*       6.  THE SUBROUTINE ARGUMENTS MAY BE SUBSCRIPTED IN THE NORMAL
*           FASHION.
*
*     NOTE - A SLIGHT ELEMENT OF DANGER IS CONNECTED
*     WITH NOT WRITING DOWN THE OPTIONAL ARGUMENT
*     LIST.  IF THE LIST IS NOT WRITTEN DOWN FOR A SUBROUTINE AND IF
*     AT THE SAME TIME ONE OF THE ARGUMENTS OF THE SUBROUTINE INSIDE
*     THE CALL SEVRAL STATEMENT CAN BE INTERPRETED AS THE PROXY NAME
*     OF SOME SUBROUTINE WHICH HAS BEEN LOCATED BY A CALL
*     LOCATE STATEMENT, THEN SEVRAL WILL BE CONFUSED, THE
*     SUBROUTINE WILL BE OPERATED WITH ONE INCORRECT ARGUMENT,
*     AND, IF THE SUBROUTINE RETURNS, CONTROL WILL BE SENT TO AN
*     ILLEGAL LOCATION.  (THE LOCATION CONTAINING THE PROXY NAME OF THE
*     NEXT SUBROUTINE).  FOR VECTOR OR
*     OTHER ARRAY ARGUMENTS ONLY THE FIRST ELEMENT NEED BE
*     CONSIDERED IN THIS CONNECTION.
*
* FUNCTION
*    THE ABOVE SEQUENCE IS EQUIVALENT IN FUNCTION TO
*
*     CALL SUBRA(A1,A2,...,ANA)
*     CALL SUBRB(B1,B2,...,BNB)
*       .
*       .
*     CALL SUBRZ(Z1,Z2,...,ZNZ)
*
* FORTRAN USAGE OF PLURAL
*
*     CALL LOCATE(SUBROU)
*     CALL SUBRU (R1,R2,...,RN)       (PROPER ARGUMENT COUNT MANDATORY)
*         .
*         .
*         .
*     CALL PLURAL(SUBROU,A1,A2,...,AN,B1,B2,...,BN,......,Z1,Z2,...,ZN)
*
* FUNCTION
*     THE ABOVE SEQUENCE IS FUNCTIONALLY EQUIVALENT TO
*
*     CALL SUBRU(A1,A2,...,AN)
*     CALL SUBRU(B1,B2,...,BN)
*       .
*       .
*       .
*     CALL SUBRU(Z1,Z2,...,ZN)
*
* FORTRAN USAGE OF DO  (A DO LOOP INSIDE A CALL SEVRAL STATEMENT)
*
*
```

```
*      CALL SEVRAL(......,2HDO,NSUBS,I,ILO,IHI,SUBRUK,K1,K2,...,KNK,
*     1              SUBRUL,L1,L2,...,LNL,.......,SUBRUF,F1,F2,...,FNF,....)
*      WHERE
*        1. NSUBS MUST EXCEED ZERO AND ILO MUST BE LSTHN= IHI
*        2. NONE OF THE SUBROUTINES IN THE LOOP MAY BE DO OR IF.
*        3. WE HAVE OMITTED THE REQUIRED CALL LOCATE STATEMENT WHICH
*           IS SIMILAR TO THAT OF SEVRAL (NOTE THAT DO ITSELF NEED
*           NOT BE LOCATED)
*        4. DO SHOULD ONLY BE CALLED IN THE ABOVE FASHION, NEVER BY
*           A CALL DO STATEMENT.
*        5. CONFUSION MAY ARISE IF THE LOOP INDEX VARIABLE I
*           HAS BEEN LEFT IN AN UNDEFINED STATE PRIOR TO THE
*           CALL SEVRAL STATEMENT, BUT ONLY IN THOSE CASES WHERE I IS
*           ALSO USED AS A SUBSCRIPT FOR ONE OR MORE OF THE ARGUMENTS
*           INSIDE THE PSEUDO-DO LOOP.  (FOR EXAMPLE THE INDEX
*           VARIABLE OF A REAL DO LOOP IS UNDEFINED ONCE THE
*           LOOP IS COMPLETED).  IN SUCH CASES THE PSEUDO-DO
*           LOOP CONTROL IS MAINTAINED BUT THE ARGUMENTS ARE
*           IMPROPERLY SUBSCRIPTED.  HENCE IN SOME INSTANCES IT MAY
*           BE NECESSARY TO MAKE A DUMMY STATEMENT, SUCH AS
*           I = 0, JUST PRIOR TO THE CALL SEVRAL STATEMENT.  IF THE
*           LOOP VARIABLE IS DEFINED (UNDEFINED) PRIOR TO THE
*           CALL SEVRAL STATEMENT IT REMAINS DEFINED (UNDEFINED)
*           ON COMPLETION OF THE STATEMENT.
*
* FUNCTION
*      THE ABOVE CALL STATEMENT IS EQUIVALENT IN FUNCTION TO
*
*         .
*         .
*         .
*      DO 10    I=ILO,IHI
*      CALL SUBRK(K1,K2,...,KNK)
*      CALL SUBRL(L1,L2,...,LNL)
*         .
*         .
*         .
* 10   CALL SUBRF(F1,F2,...,FNF)
*         .
*         .
*         .
* WHERE
*     1. THE DO LOOP CONTAINS EXACTLY NSUBS CALL STATEMENTS.
*     2. THE LOOP VARIABLE IS AVAILABLE AS AN ARGUMENT TO THE
*        SUBROUTINES IN THE LOOP.
*     3. ILO MAY BE ZERO OR NEGATIVE.
*
* FORTRAN USAGE OF IF (CONDITIONAL BRANCHING IN A CALL SEVRAL STATEMENT)
*      CALL SEVRAL(......,SUBRUK,K1,K2,...,KNK,2HIF,X,NXNEG,NXZER,NXPOS,
*     1              SUBRUL,L1,L2,...,LNL,......)
*      WHERE
*        1. X IS THE BRANCHING DETERMINANT  (MAY BE FIXED POINT)
*        2. NXNEG, NXZER, AND NXPOS ARE NON ZERO
*        3. THE SEQUENCE SHOULD NOT OCCUR INSIDE A PSEUDO-DO LOOP.
*
*
* FUNCTION
*      THE ABOVE STATEMENT FUNCTIONS EQUIVALENTLY TO THE FOLLOWING
*      FORTRAN PROGRAM (WHERE WE USE NEGATIVE STATEMENT NUMBERS)
*       .
*       .
*       .
* -3   CALL ...
* -2   CALL ...
* -1   CALL SUBRK(K1,K2,...,KNK)
*      IF (X)    NXNEG,NXZER,NXPOS
* +1   CALL SUBRL(L1,L2,...,LNL)
* +2   CALL ...
* +3   CALL ...
*       .
*       .
*       .
* WHERE THE BRANCHING
*    1.  SHOULD NOT SEND CONTROL INSIDE A DO LOOP OR TO A PSEUDO
*        CALL STATEMENT PRIOR TO THE FIRST SUBROUTINE OF THE CALL
```

```
*        SEVRAL STATEMENT.
*    2.  MAY SEND CONTROL BEYOND THE LAST SUBROUTINE OF THE CALL
*        SEVRAL STATEMENT.  IN THIS CASE CONTROL RETURNS TO THE
*        STATEMENT IMMEDIATELY FOLLOWING THE CALL SEVRAL STATEMENT,
*        REGARDLESS OF THE AMOUNT OF THE APPARENT OVERSHOOT.
*
* EXAMPLES
*
*    FOR ILLUSTRATION WE SHALL USE THE FOLLOWING FOUR ELEMENTARY
*    SUBROUTINES
*
*                    SUBROUTINE ADD (X,Y,SUMXY)
*                    SUMXY=X+Y
*                    RETURN
*              AND
*                    SUBROUTINE MUL (X,Y,XTIMSY)
*                    XTIMSY=X*Y
*                    RETURN
*              AND
*                    SUBROUTINE SUB (X,Y,XMNUSY)
*                    XMNUSY=X-Y
*                    RETURN
*              AND
*                    SUBROUTINE FADD (I,J,K)
*                    K=I+J
*                    RETURN
*
* 1. EXAMPLES OF SEVRAL AND PLURAL WITHOUT DO OR IF
*
*    USAGE   -       CALL LOCATE (3HADD,3HMUL,3HSUB)
*                    CALL ADD (1,2,3)
*                    CALL MUL (1,2,3)
*                    CALL SUB
*                    CALL FADD(1,2,3)
*              C THEN ANY AMOUNT OF PROGRAM
*              C FOLLOWED BY
*                    CALL SEVRAL (3HADD,1.,1.,Z,  3HMUL,Z,2.,W,
*                   1             3HADD,W,Z,U)
*                    CALL PLURAL (3HADD,2.,2.,V, 3.,3.,X, 4.,4.,Y)
*                    CALL SEVRAL (3HADD,5.,5.,S, 3HSUB,S,3.,D1,
*                   1             3HSUB,D1,1.,D2)
*    OUTPUTS - Z = 2.   W = 4.   U = 6.
*              V = 4.   X = 6.   Y = 8.
*              S = 10.   D1 = 7.   D2 = 6.
*
* 2. EXAMPLES OF  DO  AND  IF
*
*    INPUTS  - X(1...5) = 0.,0.,0.,0.,0.   Y(1...5) = 1.,2.,3.,4.,5.
*    USAGE   - ASSUME THE SAME CALL LOCATE SEQUENCE AS IN EXAMPLE 1,
*                THEN
*                    I=7
*                    CALL SEVRAL (3HADD,1.,1.,Z, 2HDO,2,I,1,5, 3HADD,
*                   1             X(I),2.,X(I),  3HSUB,Y(I),1.,Y(I),
*                   2             3HMUL,2.,3.,U)
*                    W=-5.
*                    CALL SEVRAL (3HADD,1.,W,W, 2HIF,W,-1,1,3, 3HADD,1.,
*                   1             1.,S, 2HIF,W,-3,-3,-3, 3HADD,W,S,P)
*
*    OUTPUTS - Z=2.   X(1...5)=2.,2.,2.,2.,2.  Y(1...5)=0.,1.,2.,3.,4.
*              U = 6.   W = 1.   S = 2.   P = 3.
*
* 3. SHOWING USE OF COMPUTED SUBSCRIPTS
*
*    INPUTS  - I=J=K=L=M=N=1  A(1..2) = 1.,2.  B(1..2,1..2)=1.,2.,3.,4.
*              C(1...2,1..2,1..2) = 1.,2.,,3.,4.,,,5.,6.,,7.,8.
*
*    USAGE   -       DIMENSION A(2),B(2,2),C(2,2,2)
*                    CALL SEVRAL(4HFADD,I,1,I, 4HFADD,J,1,J,
*                   1   4HFADD,K,1,K, 4HFADD,L,1,L, 4HFADD,M,1,M,
*                   2   4HFADD,N,1,N, 3HADD,A(I),A(I),SA, 3HADD,B(J,K),
*                   3  B(J,K),SB, 3HADD,C(L,M,N),C(L,M,N),SC)
*
*    OUTPUTS - I=J=K=L=M=N = 2  SA = 4.   SB = 8.  SC = 16.
*
```

```
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS LOCATE, WHERE.
      HTR     0             XR1
      HTR     0             XR2
      HTR     0             XR4
      BCI     1,SEVRAL
* PRINCIPLE ENTRY.  SEVRAL(SUBRUA,A1,A2,...,ANA,SUBRUB,B1,B2,...,BNB,
*                       ......)
SEVRAL SXD    SEVRAL-4,1
      TSX     LOCDOF,1
* SEVRAL IS MERELY A LOOP TO OPSUB1
 TSXS  TSX    OPSUB1,1
      TZE     NOSUB         TROUBLE
      TPL     TSXS
      TRA     LEAVE
* SECONDARY ENTRY.  PLURAL(SUBROU,A1,...,AN,B1,...,BN,.....)
PLURAL SXD    SEVRAL-4,1
      TSX     LOCDOF,1
* PLURAL IS ONE JUMP TO OPSUB1 AND A LOOP TO OPSUB2
      TSX     OPSUB1,1
      TZE     NOSUB         TROUBLE
 TSXP  TSX    TSXZCK,1      IS THERE MORE
      TRA     *+2           YES
      TRA     LEAVE         NO
      TSX     OPSUB2,1
      TRA     TSXP          (AC MUST=1)
* EXIT
 LEAVE LXD    SEVRAL-4,1
      LXD     SEVRAL-3,2
      TRA     1,4
* STOP COMPUTER IF FAIL TO
* FIND SUBROUTINE, WITH AC=NAME OF
* SUBROUTINE.  EXIT ON RESTART
 NOSUB CLA*   1,4
      HTR     LEAVE
* INTERNAL SUBROUTINE TO LOCATE DO AND IF AND TO SET SUBSCRIPT PATCH
LOCDOF SXD    SEVRAL-2,4
      SXD     SEVRAL-3,2
      SXA     SSLEVE,1
      TSX     $LOCATE,4
      TSX     DONAME,0
      TSX     IFNAME,0
      TSX     GOTODO,4
      TSX     0,0
      TSX     0,0
      TSX     0,0
      TSX     0,0
      TSX     GOTOIF,4
      TSX     0,0
      TSX     0,0
      TSX     0,0
      TSX     0,0
* ROUTINE TO SET UP SUBSCRIPT ROUTINE.
*
* LOOP TO SET SCAN FENCE.  FENCE IS -(LOCATION 144 OCTAL) OR
* -(LOCATION OF AN SXD U,XR WHERE U PRECEEDS THE LOCATION)
*     (USES XR1 AND XR2)
      LXD     SEVRAL-2,1    ORIG XR4 TO XR1
CALSS1 CAL    -1,1          STARTS BEFORE TSX $SEVRAL,4
      PAC     0,2           SAVE -U
      ANA     ATMASK
      LAS     SXDZ          LOOKING FOR SXD
      TRA     *+2
      TRA     SCCK1
* CHECK FOR OCTAL 144 IF NOT SXD
      PXA     0,1
      CAS     MIN144
      NOP
      TRA     SETFNS
      TXI     CALSS1,1,1    LESS IF XR1 GREATHER THAN 144
* SET FENCE AND PROCEED
SETFNS SXD    TXLISC,1
      TRA     NARCNT
```

```
* CHECK ADDRESS OF THE SXD AGAINST XR1
SCCK1  SXD     *+1,2
       TXL     SETFNS,1,**    ** = -U
       TXI     CALSS1,1,1
* FIND NARGUS = FULL ARGUMENT COUNT OF SEVRAL
*      (USES XR4,XR2,XR1)
NARCNT LXD     SEVRAL-2,4
       AXT     0,2            XR2 COUNTS
TSXSS1 TSX     TSXZCK,1
       TXI     *+2,2,1        YES
       TRA     *+2            NO MORE
       TXI     TSXSS1,4,-1
       SXD     NARGUS,2
* SET STA Y SCAN LIMIT
       CLA     NARGUS
       ALS     2
       ADD     KD3            4*NARGUS+3
       STD     TXISS2
       LXD     SEVRAL-2,4
TXISS2 TXI     *+1,4,**       **=3+4*NARGUS
       SXD     TXLSC1,4
* RETURN TO SEVRAL OR PLURAL
       LXD     SEVRAL-2,4
       LXD     SEVRAL-3,2
SSLEVE AXT     **,1           **=XR1
       TRA     1,1
*
* INTERNAL SUBROUTINE FOR OPERATING SUBROUTINES - OPSUB1
*      LINKAGE WITH XR1, RETURNS TO 1,1
*
* ASSUMES XR4=-A WHERE
*A+1   TSX     SUBRU,0
*      TSX     ARG1,0
*         ETC
*A+N+1 TSX     ARGN,0
*A+N+2 VARY
*
* AND,
*   IF WHERE SETS NARGS GRTHN 0, THEN N=NARGS
*   OTHERWISE N IS COUNTED FROM A+1 TO VARY,
*   COUNT STOPPING WHEN
*        VARY=NON TSX X,0 (END OF ALL ARGUMENTS)
*   OR VARY = TSX SUBROU,0 (NEXT SUBROUTINE)
*
*   SETS XR4=-(A+N+1),AC=+1  IF OK
*        XR4 UNDISTURBED, AC=0  IF NO FIND SUBRU
*        XR4 UNDISTURBED AC=-1  IF A+1 NOT TSX X,0
*
OPSUB1 SXA     OLEVE,4
       SXA     OLEVE+1,1
       TRA     OFIND
* SECONDARY ENTRY OPSUB2 (ONLY USED BY PLURAL)
*
* SIMILAR BUT ASSUMES PREVIOUSLY FOUND SUBROUTINE AND NARGS
*   AND XR4 (=-A) OFF BY 1
*
* A+1  TSX     ARG1,0
*      ETC
* A+N  TSX     ARGN
*A+N+1 VARY
*
* LEAVES XR4 = - (A+N) (SAME RELATIVE POSITION)
*
OPSUB2 TXI     *+1,4,1        MAKE XR4 LIKE OPSUB1 CASE
       SXA     OLEVE,4
       SXA     OLEVE+1,1
       TRA     OSETUP
* GO FIND SUBRU FROM 1,4
 OFIND TSX     FIND,1
       TMI     OLEVE          NOT TSX X,0
       TZE     OLEVE          NOT TSX SUBRU,0
* IF FOUND LEAVE NARGS AS IS, PROVIDED IT IS NON ZERO
       ZET     NARGS
       TRA     OSETUP
* OTHERWISE COUNT DOWN TO VARY, FIRST PUTTING LOC ASIDE.
```

```
       CLA     LOC
       STO     LOCSAV
       AXT     -1,1
ONEMOR TXI     *+1,1,1         (XR1 STARTS AT ZERO)
       SXA     OCSV1,1
       TXI     *+1,4,-1        (XR4 STARTS AT TSX ARG1,0)
       TSX     FIND,1
 OCSV1 AXT     **,1
       TZE     ONEMOR          (ORDINARY TSX X,0)
* STORE THE COUNTED NARGS AND RESTORE LOC
       SXD     NARGS,1
       CLA     LOCSAV
       STO     LOC
* SETUP
OSETUP CLA     OLEVE           -A
       PAC     0,4             +A
       SXA     OCLA,4          (SET ASIDE)
       CLA     NARGS
       PDC     0,1             -NARGS
       SXD     OTXI,1
       ARS     18              +NARGS
       ADD     OCLA
       ADD     K2
       STA     OCLA            A+NARGS+2
* GO TO SUBSCRIPT ROUTINE ONLY IF NECESSARY
       ZET     NARGS           NOT IF NARGS=0
       TRA     SCRPTS
* SET UP RETURN FROM SUBROUTINE
 OCLA  CLA     **              **=A+NARGS+2
       STO     SAVNXT
       CLA     TRABAK
       STO*    OCLA
* NOW GO OPERATE SUBROUTINE
       LXA     OLEVE,4
       TXI     *+1,4,-1        XR4 = -(A+1)
       SXA     OLEVE,4
       CLA     LOC
       ARS     18
       STA     *+1
       TRA     **              **=LOC
* AFTER RETURNING, RESTORE NEXT INSTRUCTION,
* SET AC=1, ADJUST XR4 TO -(A+N+1), AND EXIT
 OBAK  CLA     SAVNXT
       STO*    OCLA
       CLA     K1
       LXA     OLEVE,4         GIVES  XR4 = -A-1
 OTXI  TXI     OLEVE+1,4,**    **=-(NARGS)
* EXIT
 OLEVE AXT     **,4            **=XR4  (-A THEN -A-1)
       AXT     **,1            **=XR1
       TRA     1,1
*
* INTERNAL SUBROUTINE FIND
*
*    LINKAGE WITH XR1 RETRUNS TO 1,1
*
* DETERMINES IF 1,4 IS
*      1.  TSX   SUBRU,0       - SETS AC=+1
* OR   2.  TSX   X,0           - SETS AC= 0
* OR   3.  ANYTHING ELSE       - SETS AC=-1
*
* FOR CONDITION 1. IT ALSO SETS LOC, AND NARGS
*
 FIND  SXA     FNDOUT,4
       SXA     FNDOUT+1,1
* IS IT TSX  X,0
       TSX     TSXZCK,1
       TRA     ASKW            YES
       CLS     K1              NO
       TRA     FNDOUT
* IF SO, ASK WHERE
 ASKW  CLA     1,4
       STA     FTSX
       TSX     $WHERE,4
 FTSX  TSX     **,0            **=SUBRU
```

```
       TSX     IANS,0
       TSX     LOC,0
       TSX     NARGS,0
* WHATS THE ANSWER
       PXD     0,0           AC=0
       NZT     IANS
       CLA     K1            FOUND
* RETURN
FNDOUT AXT     **,4          **=XR4
       AXT     **,1          **=XR1
       TRA     1,1
*
* INTERNAL SUBROUTINE TSXZCK
*      LINKAGE XR1
*    IF 1,4 IS TSX  X,0  RETURNS TO 1,1
* OTHERWISE              RETURNS TO 2,1
*      DESTROYS AC
TSXZCK CAL     1,4
       ANA     AMASK
       LAS     TSXZ
       TRA     *+2
       TRA     1,1           YES
       TRA     2,1           NO
*
* THIRD ENTRY.  DO(NSUBS,I,ILO,IHI)   (SUBENTRY OF SEVRAL)
*
*    AT TIME OF ENTRY HERE, WE HAVE XR4 IN OLEVE = -(A+1) = -B
*
*    B = A+1   TSX   $DO,4
*        A+2   TSX   NSUBS,0     =-3,2  RELATIVE TO C
*        A+3   TSX   I,0         =-2,2
*        A+4   TSX   ILO,0       =-1,2
*    C = A+5   TSX   IHI,0       = 0,2
*        A+6   TRA   OBAK        = 1,2
*
*
*    XR2 WILL BE SET AND HELD AT -C
*
*
* FIRST ADVANCE XR2 TO -C = -(B+4) = -(A+N+1)
* AND RESTORE C+1 FROM SAVNXT
 DO    LXA     OLEVE,2       -(A+1) = -B
       TXI     *+1,2,-4          -C
       CLA     SAVNXT
       STO     1,2           (SAME AS STO*   OCLA)
* SET ILO
       CLA*    -1,2          ILO
       STO*    -2,2          TO I
* INITIALIZE FOR NEXT LOOP
NXLOOP PXD     0,2
       PDX     0,4           XR4 STARTS AT -C, EACH LOOP
       CLA*    -3,2          NSUBS
       PDX     0,1           XR1 COUNTS SUBS
* INNER LOOP
 NXSUB SXA     DSV1,1
       TSX     OPSUB1,1
       TZE     NOSUB         TROUBLE
       TPL     DSV1          OK
       TRA     LEAVE         END OF STRING
 DSV1  AXT     **,1          **=XR1
       TIX     NXSUB,1,1
* INDEX THE LOOP VARIABLE
       CLA*    -2,2
       ADD     KD1
       STO*    -2,2          I=I+1
       CAS*    0,2
       TRA     TSXS          EXIT, BACK TO SEVRAL
       NOP
       TRA     NXLOOP
*
* FOURTH ENTRY.  IF(X,NXNEG,NXZER,NXPOS)
*
* ASSUME  XR4 = -A
* A    TSX     2HIF,0
* A+1  TSX     X,0
```

```
* A+2  TSX     NXNEG,0
* A+3  TSX     NXZER,0
* A+4  TSX     NXPOS,0
* A+5  TRA     OBAK
*
* LEAVES 1,4 POSITIONED FOR PROPER SUBROUTINE
*     AND RETURNS TO SEVRAL
* DESTROYS XR1 AND XR2
*
* PICK UP PROPER N ACCORDING TO X
 IF    CLA*    1,4             X
       LDQ*    4,4             NXPOS
       TZE     INZER
       TPL     IXCA
       LDQ*    2,4             NXNEG
 IXCA  XCA
       TRA     IGOTN
 INZER CLA*    3,4             NXZER
* SET FORWARD OR BACKWARD AND POSITION XR4
 IGOTN PDX     0,2             (XR2 WILL COUNT JUMPS)
       TXI     *+1,4,1         INITIALIZE XR4 TO -(A-1).
       LXA     K1,1            XR4 IS BUMPED BY +1
       TMI     ISXD            FOR N NEGATIVE.
       PXA     0,1             OTHERWISE,
       PAC     0,1             BY -1.
       TXI     *+1,4,-4        AND XR4 STARTS AT -(A+3)
 ISXD  SXD     ITXI,1
* RESTORE FROM SAVNXT
       CLA     SAVNXT
       STO*    OCLA
* LOOP
 ITXI  TXI     *+1,4,**        **=-1 OR +1
       TSX     FIND,1
       TZE     ITXI            ARGUMENT
       TMI     LEAVE           END OF STRING
* COUNT SUBROUTINES, FOR AC=1
       TIX     ITXI,2,1
* EXIT
       TRA     TSXS
*
* SUBSCRIPT SETTING ROUTINE
*
* WE HAVE      A+1 = TSX SUBRU,0
*              A+2 = TSX ARG1,0
*                   ETC
*        A+NARGS+1 = TSX ARGN,0
*
* THE SUBSCRIPT ROUTINE EXAMINES THE FORTRAN PROGRAM PRIOR TO
* THE TSX $SEVRAL,4 , LOCKING FOR STA Y OPERATIONS WITH Y IN THE RANGE
* A+2 TO A+NARGS+1.  (THE SCAN FOR STAY-S IS LIMITED TO 3+4*NARGUS
* REGISTERS.)  FOR EACH SUCH STA Y FOUND, IT TRACKS THE PERTINENT
* INSTRUCTIONS BACK TO THEIR SOURCE, AND THEN EXECUTES THESE
* INSTRUCTIONS.
*
SCRPTS SXA     SCLEVE,1
       SXA     SCLEVE+1,2
       SXA     SCLEVE+2,4
* FIRST SET LIMITS ON THE STA Y INSTRUCTION, IN STAALO, STAAHI
       LXA     OLEVE,4         -A
       TXI     *+1,4,-1        -(A+1)
       PXA     0,4
       PAC     0,4             XR4=A(TSX SUBRU,0)
       SXA     STAALO,4
       LXD     NARGS,2
       SXD     *+1,2
       TXI     *+1,4,**        **=NARGS
       SXA     STAAHI,4        A(TSX SUBRU,0)+NARGS
* INITIALIZE -(BETA+1) TO -A(TSX $SEVRAL,4)
       LXD     SEVRAL-2,4
       SXA     AXTSC1,4
* LOOP TO FIND NEXT STA Y, Y IN ADDRESS LIMITS, IF ANY
AXTSC1 AXT     **,4            **=-(BETA+1)
       TXI     *+1,4,1         -BETA
CALSC1 CAL     0,4
       ANA     AMASK           KNOCK OUT ADDRESS ONLY
```

```
        LAS     STAZ            IS IT STA
        TRA     NOTSTA          NO
        TRA     ISSTA           YES
 NOTSTA TXI     *+1,4,1         NO
 TXLSC1 TXL     CALSC1,4,**     **=SCAN LIMIT FOR STA Y
 * SUBSCRIPT SETTING COMPLETED
 SCLEVE AXT     **,1
        AXT     **,2
        AXT     **,4
        TRA     OCLA
 * IF IT IS STA Y, CHECK Y.
 ISSTA  CAL     0,4             GET STA Y
        LAS     STAALO          MUST EXCEED STAALO
        TRA     HICHEK
        NOP
        TRA     NOTSTA          IGNORE IF NOT IN RANGE
 HICHEK LAS     STAAHI          AND BE LSTHN= STAAHI
        TRA     NOTSTA          IGNORE IF NOT IN RANGE
        NOP                     GOT IT
 * GOT ONE.  SAVE BETA AND PROCEED.
        SXA     AXTSC1,4        GOT IT
 * SET THE STA Y, SUB *-1, PXA X,XRA
 * INSTRUCTIONS FOR LATER EXECUTION
        SLW     XEC1            STA Y
        CLA     -1,4            SUB *-1
        STO     XEC2
        CLA     -2,4            GIVES PXA X,XRA
        STO     XEC3
 * LOOK FOR PRECEEDING LXD A,XRA (MUST EXIST)
        TXI     *+1,4,3         (LOOK BEFORE PXA)
        TSX     LXDTSC,1        (AC HAS PXA X,XRA)
        HPR                     ILLEGAL
 * STORE IT FOR EXECUTION, THEN CHECK PRECEEDING TSX SCRSUB,4
        STO     XEC4
        TSX     CKTSXS,1
        TRA     CASE2           NO
 * CASE 1.  EXECUTE THE TSX SCRSUB,4 ROUTINE
 * AND THEN GO TO LXD,PXA,SUB,STA SEQUENCE
        TSX     XEC7,1
        TRA     XEC4
 * CASE 2.      XR4 HAS -GAMMA.  SAVE -(GAMMA-1)
 CASE2  TXI     *+1,4,1
        SXA     CAS2X4,4
 * LOOK FOR PRECEEDING STO A
        CLA     XEC4
        STA     STOADD
        STA     STQADD
        STA     SXDADD
        CLA     STOADD
        LDQ     NOMASK
        TSX     INSCAN,1
        LXD     NOMASK,4        NO FIND (PRETEND FOUND IN 1)
        SXD     LOCSTO,4
 * LOOK FOR PRECEEDING STQ A
        LXA     CAS2X4,4
        CLA     STQADD
        TSX     INSCAN,1
        LXD     NOMASK,4        NO FIND
        SXD     LOCSTQ,4
 * LOOK FOR PRECEDING TSX SCRSUB,4 LXD A,XRB COMBO
        LXA     CAS2X4,4
        CLA     XEC4
        TSX     LXDASC,1
        TRA     LXDSC1          NO
 * POSSIBLY
        TSX     CKTSXS,1
 LXDSC1 LXD     NOMASK,4        NO FIND
        SXD     LOCLXD,4
        STA     CAS2.1          AND SAVE SCRSUB
 * LOOK FOR PRECEDING COMBO OF FORM
 *      TSX SCRSUB,4 ,LXD B,XRB,...,SXD A,XRB
        LXA     CAS2X4,4
        CLA     SXDADD
        TSX     LDQTM,1
        TRA     LXDSC2          NO
```

```
     * POSSIBLY, CHECK FOR THE LXD B,XRB
            STO     XEC5
            TXI     *+1,4,1
            TSX     LXDTSC,1
            TRA     LXDSC2         NO
     * POSSIBLY, CHECK FOR THE TSX SCRSUB,4
            STO     XEC6
            TSX     CKTSXS,1
     LXDSC2 LXD     NOMASK,4       NO
            SXD     LOCSXD,4       (SCRSUB REMAINS IN XEC7)
     * NOW FIND WHICH CAME FIRST
     * CASE 2.3 IF STO OR STQ CAME FIRST
     * CASE 2.1 IF LXD CAME FIRST
     * CASE 2.2 IF SXD CAME FIRST
     *       (ADDRESS SIZE SENSE IS REVERSED, THEREFORE
     *        LOOKING FOR SMALLEST)
     * FIND FIRST OF STO AND STQ
            CLA     LOCSTO
            CAS     LOCSTQ
            CLA     LOCSTQ
            NOP
     * COMPARE IT AGAINST LXD AND SXD
            CAS     LOCLXD
            TRA     NOT2.3
            NOP
            CAS     LOCSXD
            TRA     NOT2.3
            HPR                    STOP ON UNDEFINED SUBSCRIPT
     * CASE 2.3.  GO DIRECTLY TO LXD,PXA,SUB,STA SEQUENCE
            TRA     XEC4
     * IS IT 2.1 OR 2.2
     NOT2.3 CLA     LOCSXD
            CAS     LOCLXD
            TRA     CAS2.1
            HPR                    SHOULDNT HAPPEN
     * CASE 2.2 OPERATE TSX SCRSUB,4 LXD B,XRB SXD A,XRB
     * AND GO TO LXD,PXA,SUB,STA SEQUENCE
            TSX     XEC7,1
     XEC6   NOP                    = LXD B,XRB
     XEC5   NOP                    = SXD A,XRB
            TRA     XEC4
     * CASE 2.1 OPERATE TSX SCRSUB,4, AND GO TO
     *      LXD,PXA,SUB,STA SEQUENCE
     CAS2.1 AXT     **,1           **=SCRSUB FOR 2.1
            SXA     XEC7,1
            TSX     XEC7,1
     * OPERATE THE LXD,PXA,SUB,STA SEQUENCE
     * AND RETURN TO SCAN FOR NEXT STA Y.
     XEC4   NOP                    = LXD A,XRA
     XEC3   NOP                    = PXA X,XRA
     XEC2   NOP                    = SUB *-1
     XEC1   NOP                    = STA Y
            TRA     AXTSC1
     *
     *
     * INTERNAL SUB TO CHECK IF -1,4 IS TSX SCRSUB,4
     *      IF NOT RETURNS TO 1,1
     *      IF SO RETURNS TO 2,1 SETTING XEC7 TO SCRSUB
     *                            AND AC ADDRESS TO SCRSUB
     *
     *      XEC7 IS SET IN ANY CASE
     *      USES XR2, LEAVES XR4 UNDISTURBED
     *
     CKTSXS CAL     -1,4           POTENTIAL TSX SCRSUB,4
            STA     XEC7           SET SCRSUB
            STA     CALSCK
            PAC     0,2            -(SCRSUB) TO XR2
            ANA     AMASK
            LAS     TSXZ4
            TRA     *+2
            TRA     CKADDS         GOT IT, MAYBE
     * FAILURE
     CTSXSX TRA     1,1
     * FURTHER CHECK ON ADDRESSES
     CKADDS SXD     TXLSC2,4
```

```
TXLSC2 TXH     CTSXSX,2,**  ** = -THETA
* ALMOST CERTAINLY HAVE IT.  BUT WE NEED TO VERIFY THAT THE SUBROUTINE
* CONTAINS AN STO A WHERE A = ADDRESS PORTION OF 0,4 (THIS VERIFICATION
* MAY BE UNNECESSARY BUT IS INSERTED TO GUARD AGAINST CONFUSION BETWEEN
* SCRIPT SETTING SUBROUTINES AND OTHER TYPES OF INTERNAL FORTRAN
* SUBROUTINES).
       CLA     0,4            (LXD A,XR)
       STA     STOADD
CALSCK CAL     **             **=SCRSUB  INIT
       LAS     STOADD
       TRA     *+2            NO
       TRA     ISSUB          FINAL VERIFICATION
* INDEX FOR NEXT CHECK BUT STOP AT FIRST TRA INSTRUCTION
       CAL     CALSCK
       ACL     K1
       SLW     CALSCK
       CAL*    CALSCK
       ANA     ATMASK
       LAS     TRAZ
       TRA     *+2
       TRA     CTSXSX         EXIT IF HIT A TRA
       TRA     CALSCK         BACK
* SUCCESS
 ISSUB CLA     XEC7
       TRA     2,1
*
* INTERNAL SUB, SCANNING BACK FROM 0,4
* LOOKING FOR LXD A,XR
* WHERE XR IS ARBITRARY, A IS IN AC ADDRESS
* RETURNS TO 1,1 IF NOT FOUND
*         TO 2,1 IF FOUND, WITH AC=FULL LXD A,XR
LXDASC STA     LXDADD
       CLA     LXDADD
LDQTM  LDQ     TMASK
       TRA     INSCAN
*
*
* INTERNAL SUB, SCANNING BACK FROM 0,4
* LOOKING FOR LXD A,XR
* WHERE A IS ARBITRARY, XR IS IN AC TAG
* RETURNS TO 1,1 IF NOT FOUND
*         TO 2,1 IF FOUND, WITH AC=FULL LXD A,XR
LXDTSC STT     LXDTAG
       CLA     LXDTAG
       LDQ     AMASK
*
*
*
* INTERNAL ROUTINE SCANNING BACKWARDS FROM 0,4
*
* LOOKS FOR AC MASKED BY MQ
* RETURNS TO 1,1 IF DONT FIND
*         TO 2,1 IF FIND, WITH FULL INSTRUC IN AC
*         MQ UNDISTURBED
INSCAN STO     SOMINS
       STQ     SOMASK
CALISC CAL     0,4
       ANA     SOMASK
       LAS     SOMINS
       TRA     *+2
       TRA     GOTINS
* CHECK LIMIT
       TXI     *+1,4,1
TXLISC TXL     CALISC,4,**  ** = SCFENS
* FAILURE EXIT
       TRA     1,1
* SUCCESS EXIT
GOTINS CLA     0,4
       TRA     2,1
* INTERNAL SUBROUTINE (RETURN TO 1,1)
* TO EXECUTE THE SUBROUTINE AT SCRSUB
XEC7   XEC     **             ** STARTS AT SCRSUB
       STO     XECTMP
       CAL     XEC7
       ACL     K1
```

```
        SLW    XEC7
        CAL*   XEC7          END CHECK
        LAS    TRA14
        TRA    *+2
        TRA    1,1
        CLA    XECTMP
        TRA    XEC7
* CONSTANTS, TEMPORARIES
 AMASK  OCT    777777700000
 TSXZ   TSX    0,0
IFNAME  BCI    1,IF
DONAME  BCI    1,DO
GOTODO  TTR    DO
GOTOIF  TTR    IF
TRABAK  TRA    OBAK
 K2     PZE    2
 K1     PZE    1
 KD1    PZE    0,0,1         (USED BY DO)
NOMASK  OCT    777777777777
ATMASK  OCT    777777000000
TMASK   OCT    777777077777
KD3     PZE    0,0,3
MIN144  PZE    -100,0,0
STAZ    STA    0
TSXZ4   TSX    0,4
TRA14   TRA    1,4
 TRAZ   TRA    0
SXDZ    SXD    0,0
 IANS   PZE    0,0,**        FROM WHERE
 LOC    PZE    0,0,**        FROM WHERE
 NARGS  PZE    0,0,**        FROM WHERE AND FROM COUNTING
SAVNXT  PZE    **,**,**      TEMPORARY
LOCSAV  PZE    0,0,**        TEMP FOR LOC
STAALO  STA    **            **=A(TSX SUBRU,0)
STAAHI  STA    **            **=A(TSX SUBRU,0)+NARGS
LXDTAG  LXD    0,**
LXDADD  LXD    **,0
STOADD  STO    **
STQADD  STQ    **
SXDADD  SXD    **,0
LOCSTO  PZE    0,0,**
LOCSTQ  PZE    0,0,**
LOCLXD  PZE    0,0,**
LOCSXD  PZE    0,0,**
CAS2X4  PZE    **            XR4 FOR CASE 2
SOMASK  PZE    **,**,**
SOMINS  PZE    **,**,**
NARGUS  PZE    0,0,**
XECTMP  PZE    **,**,**
        END
```

```
***********************
*    SETSBV           *
***********************
 REFER TO
     LOCATE
```

```
***********************
*    SETUP            *
***********************
 REFER TO
     LOCATE
```

```
***********************
*    SETVCP           *
***********************
 REFER TO
     SETKP
```

```
***********************
*    SETVEC           *
***********************
 REFER TO
     SETK
```


```
***********************
*    SETSBV           *
***********************
 REFER TO
     LOCATE
```

```
***********************
*    SETUP            *
***********************
 REFER TO
     LOCATE
```

```
***********************
*    SETVCP           *
***********************
 REFER TO
     SETKP
```

```
***********************
*    SETVEC           *
***********************
 REFER TO
     SETK
```

```
*       SHFTR1 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0157
*       FAP
*SHFTR1
        COUNT   140
        LBL     SHFTR1
        ENTRY   SHFTR1 (NSHFT,IV,LIV,IVSH,IANS)
*
*                  ----ABSTRACT----
*
* TITLE - SHFTR1
*         SHIFT VECTOR ELEMENTS ARITHMETICALLY LEFT OR RIGHT
*
*                SHFTR1 SHIFTS A FORTRAN VECTOR ARITHMETICALLY TO THE
*                RIGHT OR LEFT A SPECIFIED NUMBER OF PLACES.
*
* LANGUAGE  - FAP, SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 704, 709, OR 7090  (MAIN FRAME ONLY)
* STORAGE   - 70 REGISTERS
* SPEED     - TIME IS LENGTH OF VECTOR TIMES 8 MACHINE CYCLES OR MORE
*             DEPENDING ON NO. OF SHIFTS REQUIRED
* AUTHOR    - S.M. SIMPSON,  JUNE, 1962
*
*                  ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*         AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*      CALL SHFTR1(NSHFT,IV,LIV,IVSH,IANS)
*
* INPUTS
*
*     NSHFT      IS NO. OF PLACES TO SHIFT (TREATED MODULO 36), IN DECR.
*                IF NSHFT GRTHAN 0 SHIFT IS TO RIGHT.
*                IF NSHFT LSTHAN 0 SHIFT IS TO LEFT.
*                IF NSHFT = 0 NO SHIFT IS MADE BEFORE IV IS STORED IN IVSH
*
*     IV(I)      I=1...LIV IS THE FORTRAN VECTOR.
*
*     LIV        IS IN DECREMENT.
*                LIV MUST EXCEED 0
*
* OUTPUTS
*
*     IVSH(I)    I=1...LIV = IV(I)*2**(-(NSHFT)MOD 36)
*                IVSH(1) AND IV(1) MAY BE EQUIVALENT.
*
*     IANS       =  0 NORMAL.
*                = +1 OVERFLOW OCCURRED BUT SHIFTING COMPLETED.
*                = -3 ILLEGAL LIV.
*
* EXAMPLES
*
* 1. INPUTS  - NSHFT=6  IV(1...2) = OCT 450000000000, 527210000012
*                LIV=2
*    OUTPUTS - IVSH(1...2) = OCT 400500000000, 401272100000  IANS=0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NSHFT=0
*    OUTPUTS - IVSH(1...2) = OCT 450000000000, 527210000012  IANS=0
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT NSHFT=-3
*    OUTPUTS - IVSH(1...2) = OCT 500000000000, 672100000120  IANS=1
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  LIV=0
*    OUTPUTS - IVSH IS UNCHANGED   IANS=-3
*
*
        HTR     0
        HTR     0
        HTR     0
        BCI     1,SHFTR1
SHFTR1  SXD     SHFTR1-4,1
        SXD     SHFTR1-3,2
        SXD     SHFTR1-2,4
*
```

SHFTR1
(PAGE 2)

```
     * ADDRESS SETTINGS.
            CLA      1,4             A(A(NSHFT))
            STA      GET1
            CLA      2,4             A(A(IV))
            ADD      K1
            STA      CLA
            CLA      3,4             A(A(LIV))
            STA      GET3
            CLA      4,4             A(A(IVSH))
            ADD      K1
            STA      STO
            CLA      5,4             A(A(IANS))
            STA      PUT5
     *
     * GET INPUTS NSHFT, LIV, CHECK LIV.
       GET1 CLA      **              A(NSHFT)
            ARS      18
            STO      NSHFT
            CLS      K3
            STO      IANS
       GET3 CLA      **              A(LIV)
            ARS      18
            STO      LIV
            TMI      LEAVE
            TZE      LEAVE
            STZ      IANS
     *
     * SET SHIFT INSTRUCTION.
            CLA      NSHFT
            TMI      LEFT
      RIGHT CLA      KARS
            STO      ASHFT
            TRA      MOD
       LEFT CLA      KALS
            STO      ASHFT
     *
     * SET MAGNITUDE OF SHIFT
        MOD CLA      NSHFT
            SSP
      TZE   TZE      SETSH+2
            SUB      K36
            TMI      SETSH
            TRA      TZE
      SETSH ADD      K36
            STA      ASHFT
     *
     * TURN OFF OVERFLOW BEFORE LOOP.
            LXA      LIV,1
            TOV      CLA
     *
     * LOOP.
        CLA CLA      **,1            A(IV)+1
      ASHFT NOP      **              ARS ** , OR ALS **
        STO STO      **,1            A(IVSH)+1
            TIX      CLA,1,1
     *
     * CHECK FOR OVERFLOW.
            TOV      OVSET
            TRA      LEAVE
      OVSET CLA      K1
            STO      IANS
     *
     * LEAVE, STORING IANS.
      LEAVE CLA      IANS
            ALS      18
       PUT5 STO      **              A(IANS)
            LXD      SHFTR1-4,1
            LXD      SHFTR1-3,2
            LXD      SHFTR1-2,4
            TRA      6,4
     *
     * CONSTANTS
         K1 PZE      1
         K3 PZE      3
        K36 PZE      36
```

****
*   SHFTR1   *
****
(PAGE 3)


```
       KARS ARS      0
       KALS ALS      0
      *
      * VARIABLES
      NSHFT PZE      **
       IANS PZE      **          -3, 0, +1
        LIV PZE      **
            END
```

```
*       SHFTR2 (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0162
*       FAP
*SHFTR2
        COUNT   150
        LBL     SHFTR2
        ENTRY   SHFTR2 (NSHFT,IV,LIV,IVSH,IANS)
*
*                  ----ABSTRACT----
*
* TITLE - SHFTR2
*         SHIFT VECTOR ELEMENTS LOGICALLY LEFT OR RIGHT
*
*               SHFTR2 SHIFTS A FORTRAN VECTOR LOGICALLY TO THE RIGHT A
*               SPECIFIED NUMBER OF PLACES (OR LEFT IF THE NUMBER OF
*               PLACES IS NEGATIVE).
*
* LANGUAGE   - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 704, 709, OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 72 REGISTERS
* SPEED      - TAKES 8*LENGTH OF VECTOR MACHINE CYCLES OR MORE DEPENDING
*               ON NO. OF SHIFTS REQUIRED
* AUTHOR     - S.M. SIMPSON AND R.A. WIGGINS     9/28/62
*
*                  ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -    NONE
*        AND FORTRAN SYSTEM ROUTINES -    NONE
*
* FORTRAN USAGE
*     CALL SHFTR2(NSHFT,IV,LIV,IVSH,IANS)
*
* INPUTS
*
*    NSHFT      IS NO. OF PLACES TO SHIFT (TREATED MODULO 36), IN DECR.
*               IF NSHFT GRTHAN 0 SHIFT IS TO RIGHT.
*               IF NSHFT LSTHAN 0 SHIFT IS TO LEFT.
*               IF NSHFT = 0 NO SHIFT IS MADE BEFORE IV IS STORED IN IVSH
*
*    IV(I)      I=1...LIV IS THE FORTRAN VECTOR.
*                 (NAME NEED NOT BE FIXED POINT)
*
*    LIV        IS IN DECREMENT.
*               LIV MUST EXCEED 0
*
* OUTPUTS
*
*    IVSH(I)    I=1...LIV = IV(I)*2**(-(NSHFT)MOD 36)
*               IVSH(1) AND IV(1) MAY BE EQUIVALENT.
*                 (NAME NEED NOT BE FIXED POINT)
*
*    IANS        =  0 NORMAL.
*                = +1 OVERFLOW OCCURRED BUT SHIFTING COMPLETED.
*                = -3 ILLEGAL LIV.
*
* EXAMPLES
*
* 1. INPUTS  - NSHFT=6  IV(1...2)=OCT450000000000,527210000012  LIV=2
*    OUTPUTS - IVSH(1...2)=OCT004500000000,005272100000  IANS=0
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  NSHFT=0
*    OUTPUTS - IVSH(1...2)=OCT 450000000000, 527210000012  IANS=0
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  NSHFT=-3
*    OUTPUTS - IVSH(1...2)=OCT 500000000000,272100000120  IANS=1
*
* 4. INPUTS  - NSHFT=-3  IV(1...2)=OCT000714221216,002142606060  LIV=2
*    OUTPUTS - IVSH(1...2)=OCT007142212160,021426060600  IANS=0
*
* 5. INPUTS  - SAME AS EXAMPLE 4. EXCEPT LIV=0
*    OUTPUTS - IVSH(1...2)=0,0  IANS=-3
*
        HTR     0
        HTR     0
        HTR     0
        BCI     1,SHFTR2
```

****
*   SHFTR2   *
****
(PAGE 2)

```
SHFTR2 SXD     SHFTR2-4,1
       SXD     SHFTR2-3,2
       SXD     SHFTR2-2,4
*
* ADDRESS SETTINGS.
       CLA     1,4            A(A(NSHFT))
       STA     GET1
       CLA     2,4            A(A(IV))
       ADD     K1
       STA     CLA
       CLA     3,4            A(A(LIV))
       STA     GET3
       CLA     4,4            A(A(IVSH))
       ADD     K1
       STA     STO
       CLA     5,4            A(A(IANS))
       STA     PUT5
*
* GET INPUTS NSHFT, LIV, CHECK LIV.
  GET1 CLA     **             A(NSHFT)
       ARS     18
       STO     NSHFT
       CLS     K3
       STO     IANS
  GET3 CLA     **             A(LIV)
       ARS     18
       STO     LIV
       TMI     LEAVE
       TZE     LEAVE
       STZ     IANS
*
* SET SHIFT INSTRUCTION.
       CLA     NSHFT
       TMI     LEFT
 RIGHT CLA     KARS
       STO     ASHFT
       TRA     MOD
  LEFT CLA     KALS
       STO     ASHFT
*
* SET MAGNITUDE OF SHIFT (EXIT IF ZERO).
   MOD CLA     NSHFT
       SSP
 TZE   TZE     SETSH+2
       SUB     K36
       TMI     SETSH
       TRA     TZE
 SETSH ADD     K36
       STA     ASHFT
*
* TURN OFF OVERFLOW BEFORE LOOP.
       LXA     LIV,1
       LDQ     =0
       TOV     CLA
*
* LOOP.
   CLA CAL     **,1                A(IV)+1
 ASHFT NOP     **             ARS ** , OR ALS **
   STO SLW     **,1                A(IVSH)+1
       TIX     CLA,1,1
*
* CHECK FOR OVERFLOW.
       TOV     OVSET
       TRA     LEAVE
 OVSET CLA     K1
       STO     IANS
*
* LEAVE, STORING IANS.
 LEAVE CLA     IANS
       ALS     18
  PUT5 STO     **             A(IANS)
       LXD     SHFTR2-4,1
       LXD     SHFTR2-3,2
       LXD     SHFTR2-2,4
```

```
         TRA     6,4
     *
     * CONSTANTS
        K1 PZE    1
        K3 PZE    3
       K36 PZE    36
      KARS ARS     0
      KALS ALS     0
     *
     * VARIABLES
     NSHFT PZE    **
      IANS PZE    **           -3, 0, +1
       LIV PZE    **
           END
```

```
*     SHUFFL (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0124
*     LABEL
CSHUFFL
      SUBROUTINE SHUFFL(ITPRD,NITEMS,ISPACE,IXSHUF)
C
C
C                    ----ABSTRACT----
C
C TITLE - SHUFFL
C        SHUFFL A LIST OF INTEGERS FROM 1 TO N
C
C              SHUFFL IS GIVEN A NUMBER N, FROM WHICH IT INFERS THE
C              SET OF INTEGERS 1,2,...,N.  IT THEN PRODUCES AN OUTPUT
C              VECTOR OF LENGTH N WHOSE ELEMENTS ARE THE INTEGERS FROM
C              THIS SET BUT RANDOMLY SCRAMBLED.  REPEATED CALLS YIELD
C              INDEPENDENT SHUFFLINGS.
C
C              THE TECHNIQUE UTILIZES THE RAND RANDOM DIGITS TAPE
C              (ACCESS THRU SUBROUTINE GETRD1) AS FOLLOWS.  EACH
C              ORIGINAL INTEGER IS ASSIGNED A UNIQUE EQUALLY LIKELY
C              RANDOM NUMBER IN THE RANGE 0 TO 99,999.  AN INDEX BY SIZE
C              OF THESE NUMBERS IS THE DESIRED LIST OF SHUFFLED NUMBERS.
C
C              A SPACE VECTOR OF LENGTH N IS REQUIRED FOR SCRATCH.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  (MAIN FRAME PLUS ONE TAPE UNIT)
C STORAGE    - 101 REGISTERS
C SPEED      - TAKES ON THE ORDER OF .004*N SECONDS ON THE 7094.
C AUTHOR     - S.M.SIMPSON, FEBRUARY,1964
C
C
C                         USAGE
C
C TRANSFER VECTOR CONTAINS ROUTINES - GETRD1,SEARCH,SIZEUP
C       AND FORTRAM SYSTEM ROUTINES - (NOT ANY)
C
C
C FORTRAN USAGE
C     CALL SHUFFL(ITPRD,NITEMS,ISPACE,IXSHUF)
C
C
C INPUTS
C
C    ITPRD     IS THE LOGICAL TAPE NO. OF THE RAND RANDOM DIGITS TAPE.
C                 SHUFFL DOES NOT POSITION ITPRD BEFORE OR AFTER CALLING
C                 SUBROUTINE GETRD1.
C
C    NITEMS    IS THE GIVEN NO. OF ITEMS (CALLED N IN ABSTRACT).
C
C    ISPACE(I) I=1...NITEMS  MUST BE AVAILABLE FOR SCRATCH.
C
C
C OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF NITEMS=0 OR LESS.
C
C    IXSHUF(I) I=1...NITEMS  IS A SHUFFLED LIST OF THE INTEGERS
C                 1...NITEMS.
C
C
C EXAMPLES
C
C 1. INPUTS  - ASSUME THE FIRST TWO RANDOM DIGITS CARDS CONTAIN DIGITS
C                 AS FOLLOW
C                    10097325337652013586346735487680959091173929274945
C                    37542048056489474296248052403720636104020082291665
C                 AND THAT THESE ARE ON LOGICAL 9, WHICH IS REWOUND.
C
C    USAGES  -       CALL SHUFFL(9,7,ISPACE,IXSHF1)
C                    CALL SHUFFL(9,10,ISPACE,IXSHF2)
C
C    OUTPUTS - IXSHF1(1...7)  = 1,4,2,5,6,3,7
C              IXSHF2(1...10) = 5,1,10,9,8,4,2,6,7,3
C
C
```

```
C PROGRAM FOLLOWS BELOW
C
C
C DUMMY DIMENSIONS
C
      DIMENSION  ISPACE(2),IXSHUF(2)
C
C TRUE DIMENSIONS
C
      DIMENSION  IRD(5)
C
C (NDIGS COULD BE CUT BACK TO 3 OR 4 TO SAVE DIGITS)
C
      NDIGS=5
C
C CHECK OUT
C
      IF (NITEMS)  9999,9999,10
 10   CONTINUE
C
C FIRST SET UP THE ISPACE VECTOR WITH RANDOM NUMBERS
C
      DO 100  IXSP=1,NITEMS
C
C ACQUIRE THE NEXT GROUP OF DIGITS (IGNORE IANS) INTO IRD(1..NDIGS)
C
 40   CALL GETRD1(ITPRD,NDIGS,IRD,IANS)
C
C CONVERT TO INTEGER IN RANGE 0 TO 10EXP(NDIGS)-1
C
      NUMB=0
      DO 50  IXD=1,NDIGS
 50   NUMB=10*NUMB+IRD(IXD)
C
C RETURN TO GETRD1 STATEMENT IF THIS NUMBER HAS OCCURRED ALREADY
C      (SEARCH WORKS FOR LNOW=0).
C
      LNOW=IXSP-1
      CALL SEARCH(LNOW,ISPACE,NUMB,INDEX)
      IF (INDEX)  70,70,40
C
C STORE THE NEW NUMBER
C
 70   ISPACE(IXSP)=NUMB
 100  CONTINUE
C
C NOW MAKE A SIZE INDEX OF ISPACE INTO IXSHUF AND EXIT.
C
      CALL SIZEUP(ISPACE,NITEMS,IXSHUF)
 9999 RETURN
      END
```

```
*       SIFT (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0117
*       FAP
*SIFT
        COUNT   150
        LBL     SIFT
        ENTRY   SIFT   (X, MESH, LXSFTD, XSFTD)
*
*
*                   ----ABSTRACT----
*
* TITLE - SIFT
*       FORM A VECTOR BY SIFTING ANOTHER AT EVEN INCREMENTS
*
*             SIFT  FORMS A VECTOR
*
*                   XSFTD(I) = X(1+(I-1)*MESH)  I=1...LXSFTD
*
*             GIVEN THE INPUT VECTOR  X(1...)  AND THE VALUES  MESH
*             AND  LXSFTD.  OUTPUT VECTOR MAY REPLACE INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 30 REGISTERS
* SPEED      - 43 + 10*LXSFTD  MACHINE CYCLES ON THE 7090
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NOT ANY
*      AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     CALL SIFT(X, MESH, LXSFTD, XSFTD)
*
*
* INPUTS
*
*    X(I)       I=1,2,...  IS A FIXED OR FLOATING VECTOR.
*
*    MESH       SHOULD NOT BE LESS THAN ZERO.
*
*    LXSFTD     SHOULD EXCEED ZERO.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT FOR ILLEGAL  MESH  OR
*               LXSFTD.
*
*    XSFTD(I)  I=1...LXSFTD  IS DESCRIBED IN ABSTRACT.
*                  EQUIVALENCE  (X,XSFTD)  IS OK.
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...10) = 1.,2.,3.,...,10.  XS5(1) = XS6(1) = -9.
*    USAGE   -       CALL SIFT(X,0,3,XS1)
*                    CALL SIFT(X,1,3,XS2)
*                    CALL SIFT(X,3,3,XS3)
*                    CALL SIFT(X,3,1,XS4)
*                    CALL SIFT(X,-1,3,XS5)
*                    CALL SIFT(X,1,0,XS6)
*                    CALL SIFT(X,5,2,X)
*
*    OUTPUTS - XS1(1...3) = 1.,1.,1.  XS2(1...3) = 1.,2.,3.
*              XS3(1...3) = 1.,4.,7.  XS4(1) = 1.  XS5(1) = XS6(1) = -9.
*              X(1...2) = 1.,6.
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
        HTR                     XR1
        HTR                     XR4
```

```
       BCI     1,SIFT
*
* ONLY ENTRY.  SIFT(X, MESH, LXSFTD, XSFTD)
*
SIFT   SXD     SIFT-3,1
       SXD     SIFT-2,4
*
* SET ADDRESSES, CHECK  MESH  GRTHN=  ZERO,  LXSFTD  GRTHN=  1 .
*
       CLA     1,4              A(X)
       ADD     K1               A(X)+1
       STA     CLA
       CLA     4,4              A(XSFTD)
       ADD     K1               A(XSFTD)+1
       STA     STO
       CLA*    2,4              MESH
       TZE     STD
       TMI     LEAVE
STD    STD     TXI
       CLA*    3,4              LXSFTD
       TMI     LEAVE
       PDX     0,1
       TXL     LEAVE,1,0
       STD     TXL
*
* LOOP WITH XR1, XR4 STARTING AT 1
*
       AXT     1,5
CLA    CLA     **,4             ** = A(X)+1
STO    STO     **,1             ** = A(XSFTD)+1
TXI    TXI     *+1,4,**         ** = MESH
       TXI     *+1,1,1
TXL    TXL     CLA,1,**         ** = LXSFTD
*
* EXIT
*
LEAVE  LXD     SIFT-3,1
       LXD     SIFT-2,4
       TRA     5,4
*
* CONSTANT
*
K1     PZE     1
       END
```

```
*       SIMEQ (SUBROUTINE)                 9/9/64   LAST CARD IN DECK IS NO. 0641
*       FAP
*SIMEQ
        COUNT   550
        LBL     SIMEQ
        ENTRY   SIMEQ (N,LN,LM,A,B,D,E,ERR)
        ENTRY   DETRM (N,LN,A,D,ERR)
*
*                    ----ABSTRACT----
*
* TITLE - SIMEQ  WITH SECONDARY ENTRY POINT DETRM
*       SOLUTION OF SIMULTANEOUS EQUATIONS AND DETERMINANT EVALUATION
*
*             SIMEQ SOLVES THE MATRIX EQUATION
*
*                         AX=B
*
*             WHERE  A HAS LN ROWS AND LN COLUMNS
*                    B HAS LN ROWS AND LM COLUMNS
*                    X HAS LN ROWS AND LM COLUMNS
*
*             THE SOLUTION MATRIX, X, IS STORED IN A.
*             THE SOLUTION OF THE MATRIX EQUATION IS ACOMPLISHED BY
*             UPPER TRIANGULARIZATION OF THE A MATRIX USING A MAXIMUM
*             PIVOT FOR EACH REDUCTION STEP. A SCALED VERSION OF THE
*             DETERMINANT IS COMPUTED AT THE SAME TIME.
*
*             DETRM COMPUTES THE DETERMINANT OF A. THE DETRM ENTRY
*             POINT CAUSES ONLY THE TRIANGULARIZATION PROCESS OF SIMEQ
*             TO BE OPERATIVE. THE COMPUTATION IS PERFORMED BY FORMING
*             PRODUCTS OF SUCCESSIVE PIVOTS WITH PROPER SIGN ADJUSTMENT
*             TO COMPENSATE FOR THE ROW AND COLUMN INTERCHANGES. D, THE
*             DETERMINANT VALUE IS THUS
*
*               D=(...S(((D)A(1,1))A(2,2))...)A(LN,LN)
*
*             WHERE THE A(I,I) ARE THE PIVOTS. S IS SET INITIALLY BY THE
*             CALLING PROGRAM SO THAT A SCALED VERSION OF THE
*             DETERMINANT MAY BE OBTAINED. S SHOULD BE SET TO 1. IF NO
*             SCALING IS DESIRED.
*
*             IF THE MATRIX IS SINGULAR THE VALUE OF THE DETERMINANT
*             WHICH IS RETURNED IS ZERO.
*
*             NOTE- SIMEQ DESTROYS BOTH THE A AND B MATRICES.
*                   DETRM DESTROYS THE A MATRIX.
*
*
* LANGUAGE  - FAP
* EQUIPMENT - 709/7090/7094 (MAIN FRAME ONLY)
* STORAGE   - 441 REGISTERS
* SPEED     - SIMEQ -
*                    13*LN**3 + 20*LM*LN**2 + 49*LN**2 + 51*LM*LN
*                             + 158*LN + 100 MACHINE CYCLES ON THE 7090.
*             DETRM -
*                    11*LN**3 + 39*LN**2 + 126*LN + 28
*                               MACHINE CYCLES ON THE 7090.
* AUTHOR    - XSIMEQ AND XDETRM WERE ORIGINALLY WRITTEN IN FORTRAN FOR
*             THE 704 BY J.T. OLSZTYN (SHARE DISTRIBUTIONS 355 AND 364).
*             THEY HAVE BEEN REWRITTEN IN FAP WITH SOME CORRECTIONS AND
*             SPEED IMPROVEMENTS BY ARCADIO M. NIELL OF THE COMPUTATION
*             CENTER AT M.I.T.  ADDITIONAL CORRECTIONS TO TAKE INTO
*             ACCOUNT CHANGES IN FORTRAN HAVE BEEN ADDED BY THE
*             COMPUTATION CENTER STAFF.  SIMEQ, AND DETRM ARE THE
*             RESULT OF A CHANGE IN DEFINITION OF XSIMEQ AND XDETRM
*             MADE BY R. A. WIGGINS.
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES- NONE
*       AND FORTRAN SYSTEM ROUTINES- NONE
*
* FORTRAN USAGE
*     CALL SIMEQ (N,LN,LM,A,B,D,E,ERR)
```

```
*     CALL DETRM (N,LN,A,D,ERR)
*
*    A(I,J)     I=1...LN, J=1...LN IS NORMALIZED FLOATING POINT VECTOR
*                 CONTAINING THE ELEMENTS OF THE A MATRIX.
*               J REFERS TO THE COLUMN INDEX, I TO THE ROW INDEX.
*               IS DESTROYED BY THE SUBROUTINE.
*
*    LN         1 LSTHN= LN LSTHN= N
*               IS FORTRAN II INTEGER
*
*    N          IS THE LARGEST VALUE WHICH I (OF A(I,J)) MAY TAKE ON.
*                 IF A(I,J) IS DIMENSIONED AS A ONE-DIMENSIONED VECTOR
*                   WITH THE ROWS STACKED TOGETHER, THEN N=LN.
*                 IF A(I,J) IS DIMENSIONED AS A TWO DIMENSIONAL VECTOR,
*                   THEN N IS THE VALUE DIMENSIONED FOR I (SEE ALSO B)
*                   I.E.   DIMENSION A(N,N)
*               IS FORTRAN II INTEGER
*
*    B(I,J)     I=1...LN, J=1...LM IS NORMALIZED FLOATING POINT VECTOR
*                 CONTAINING THE ELEMENTS OF THE B MATRIX.
*               J REFERS TO THE COLUMN INDEX, I TO THE ROW INDEX.
*               IS DESTROYED BY THE SUBROUTINE
*               IF B IS DIMENSIONED AS A TWO-DIMENSIONED VECTOR, THEN
*                 THE FOLLOWING LIMITS HOLD (SEE ALSO N)
*                     DIMENSION A(N,N), B(N,N1)
*                       1 LSTHN= LN LSTHN= N
*                      LM LSTHN= N1 LSTHN= N
*                      LM MAY BE GRTHN= LN
*
*    LM         IS FORTRAN II INTEGER
*
*    D          IS A FLOATING POINT VARIABLE WHICH SERVES AS A SCALE
*                 BY WHICH THE VALUE OF THE DETERMINANT OF A IS
*                 MULTIPLIED.
*               CAUTION - THIS IS ALSO AN OUTPUT VARIABLE.
*
*    E(I)       I=1...LN IS ERASABLE COMPUTATION SPACE.
*               NEED NOT HAVE FLOATING POINT NAME.
*
* OUTPUTS
*
*    A(I,J)     I=1...LN, J=1...LM IS THE FLOATING POINT VECTOR
*                 CONTAINING THE ELEMENTS OF THE X MATRIX.
*               J REFERS TO THE COLUMN INDEX, I TO THE ROW INDEX.
*
*    D          IS THE SCALED VERSION OF THE DETERMINANT OF A.
*
*    ERR        =0. IF SOLUTION WAS SUCCESSFUL
*               =1. IF UNDERFLOW OR OVERFLOW OCCURRED.
*               =2. IF MATRIX A IS SINGULAR.
*
* EXAMPLES
*
*    SIMEQ EXAMPLES
*
* 1. INPUTS  - A(1,1...2) = 2., 3.   B(1...2) = 1., 0.
*               A(2,1...2) =  1., 2.
*               LN=2  N=2  LM=1  D=1.
*    USAGE   -        DIMENSION A(2,2), B(2), E(2)
*                     CALL SIMEQ (N,LN,LM,A,B,D,E,ERR)
*    OUTPUTS - A(1...2,1) =  2.000, -1.000  D=1.  ERR=0.
*
* 2. INPUTS  - A(1,1...2) =  2., 1.   B(1,1...2) = 1., 0.
*               A(2,1...2) =  1., 2.   B(2,1...2) = 0., 1.
*               LN=2  N=2  LM=2  D=1.
*    USAGE   -        DIMENSION A(2,2), B(2,2), E(2)
*                     CALL SIMEQ (N,LN,LM,A,B,D,E,ERR)
*    OUTPUTS - A(1,1...2) = .667, -.333   D=3.  ERR=0.
*               A(2,1...2) =-.333,  .667
*
* 3. INPUTS  - A(1,1...2) = 2., 1.   B(1,1...3) = 1., 0., 1.
*               A(2,1...2) = 1., 2.   B(2,1...3) = 0., 1., 1.
*               LN=2  N=3  LM=3  D=1.
*    USAGE   -        DIMENSION A(3,3), B(3,3), E(2)
*    OUTPUTS - A(1,1...3) = .667, -.333,  .333   D=3.  ERR=0.
```

```
*              A(2,1...3) =-.333,  .667,  .333
*
* 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT D=2.
*    OUTPUTS - A(1...2,1) = 2.000, -1.000   D=2.   ERR=0.
*
* 5. INPUTS  - A(1...4) = 2., 1., 1., 2.   B(1...2) = 1., 0.
*              LN=2  N=2  LM=1  D=1.
*    USAGE   -       DIMENSION A(4), B(2), E(2)
*    OUTPUTS - A(1...2) = .667, -.333   D=3.  ERR=0.
*
* 6. INPUTS  - A(1...4) = 1., 0., 0., 0.   B(1...2) = 1., 0.
*              LN=2  N=2  LM=1  D=1.
*    OUTPUTS - D=0.   ERR=2.
*
*
*    DETRM EXAMPLES
*
* 1. INPUTS  - A(1..2,1)=2.,1.   LN=2   N=5   D=1.
*              A(1..2,2)=1.,2.
*    OUTPUTS - ERR=0.   D=3.
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT D=2.
*    OUTPUTS - ERR=0.   D=6.
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LN=3
*              A(1..3,3) =0., A(3,1..3)=0.
*    OUTPUTS - ERR=2.   D=0.
*
*
* PROGRAM FOLLOWS BELOW
*
       PZE     0
       BCI     1,SIMEQ
       REM
       REM     LOCATE PIVOT AND RECORD I AND J
 T1    LXD     AKK,1            INITIALIZE ELEMENT LOCATION INDEX
       SXD     AKQ,1
       LXD     K,2              INITIALIZE ROW INDEX
       LXD     K,4              INITIALIZE COLUMN INDEX
       SXD     I,2              INITIALIZE MAXIMUM PIVOT ROW
       SXD     J,4              INITIALIZE MAXIMUM PIVOT COLUMN
 T7    PXD     0,0
 T8    ADM     0,1              AC CONTAINS MAGNITUDE CURRENT MAXIMUM
       TXI     T10,1,1          NEXT ELEMENT
 T10   TXI     T11,2,1          NEXT ROW
 T11   TXH     T17,2,LN         TRANSFER IF LAST ROW TESTED
 T12   SBM     0,1              TEST CURRENT ELEMENT
       TPL     T8               CURRENT MAXIMUM PIVOT HOLDS
       SXD     I,2              CHANGE MAXIMUM PIVOT
       SXD     J,4
       TRA     T7
 T17   LXD     AKQ,1            KTH ELEMENT, CURRENT COLUMN
 T18   TXI     T19,1,N
 T19   SXD     AKQ,1            KTH ELEMENT, NEXT COLUMN
       LXD     K,2              KTH ROW
       TXI     T20,4,1          NEXT COLUMN
 T20   TXL     T12,4,LN         EXIT IF LAST COLUMN TESTED
       REM
       REM     INTERCHANGE ROWS IF NECESSARY
       REM
 T21   CLA     I
       SUB     K
       TZE     T55              NO ROW INTERCHANGE
       ADD     AKK
       PDX     0,2              INITIALIZE ITH ROW INDEX
       LXD     AKK,1            INITIALIZE KTH ROW INDEX
       LXD     K,4              INITIALIZE COLUMN INDEX
 T28   CLS   , D                CHANGE SIGN OF
 T29   STO   , D                     DETERMINANT
 T30   LDQ     0,1              INTERCHANGE
       CLA     0,2                   KTH AND ITH
       STO     0,1                   ROWS OF
       STQ     0,2                   MATRIX A
 T34   TXI     T35,1,N          NEXT ELEMENT, KTH ROW
 T35   TXI     T36,2,N          NEXT ELEMENT, ITH ROW
```

```
T36    TXI      T37,4,1          NEXT COLUMN
T37    TXL      T30,4,LN
T38    NOP                       TRANSFER TO T55 FOR XDETRM
       CLA      KM1
       ADD      B
       PDX      0,1              INITIALIZE KTH ROW INDEX
       SUB      K
       ADD      I
       PDX      0,2              INITIALIZE ITH ROW INDEX
       LXD      =01000000,4      INITIALIZE COLUMN INDEX
T46    LDQ      0,1              INTERCHANGE
       CLA      0,2                   KTH AND ITH
       STO      0,1                   ROWS OF
       STQ      0,2                   MATRIX B
T50    TXI      T51,1,N          NEXT ELEMENT, KTH ROW
T51    TXI      T52,2,N          NEXT ELEMENT, ITH ROW
T52    TXI      T53,4,1          NEXT COLUMN
T53    TXL      T46,4,LM         EXIT IF LAST COLUMN PROCESSED
       REM
       REM      INTERCHANGE COLUMNS IF NECESSARY
       REM
T55    CLA      J
       SUB      K
       TZE      T85              NO COLUMN INTERCHANGE
       ADD      KM1
       LRS      35
       MPY      N
       ALS      17
       ADD      A
       PDX      0,1              INITIALIZE JTH COLUMN INDEX
       CLA      KM1N
       ADD      A
       PDX      0,2              INITIALIZE KTH COLUMN INDEX
       LXD      LN,4             INITIALIZE COMPLEMENTARY ROW INDEX
T68    CLS   ,  D                CHANGE SIGN OF
T69    STO   ,  D                     DETERMINANT
T70    LDQ      0,1              INTERCHANGE
       CLA      0,2                   KTH AND JTH
       STO      0,1                   COLUMNS OF
       STQ      0,2                   MATRIX A
       TXI      T75,1,1          NEXT ELEMENT, JTH COLUMN
T75    TXI      T76,2,1          NEXT ELEMENT, KTH COLUMN
T76    TIX      T70,4,1
T77    NOP                       TRANSFER TO T85 FOR XDETRM
       LXD      J,1
       LXD      K,2
T80    CLA     ,1 E+1,1          INTERCHANGE
T81    LDQ     ,2 E+1,2               JTH AND KTH
T82    STO     ,2 E+1,2               ELEMENTS OF
T83    STQ     ,1 E+1,1               ARRAY E
       REM
       REM      COMPUTE DETERMINANT
       REM
T85    LXD      AKK,1
       CLA      0,1              PIVOT ELEMENT
       TZE      T251             MATRIX A SINGULAR
       LRS      35
T89    FMP   ,  D
T90    STO   ,  D
       REM
       REM      ROW REDUCTION
       REM
       LXD      KP1,1
       SXD      E1,1             INITIALIZE ROW TO BE REDUCED
       LXD      AKK,1
       SXD      E2,1
       CLA      KM1
       ADD      B
       STD      E3
T99    LXD      E3,1
       TXI      T101,1,1
T101   SXD      E3,1             FIRST ELEMENT, CURRENT ROW, MATRIX B
       LXD      E2,1
       TXI      T104,1,1
T104   SXD      E2,1             LEADING ELEMENT, CURRENT ROW, MATRIX A
```

```
         LXD     AKK,2
         LXD     KP1,4           INITIALIZE COLUMN INDEX
         CLA     0,1
         TZE     T136            ROW NEEDS NO REDUCTION
         FDP     0,2
         STQ     G
  T111   TXI     T112,1,N
  T112   TXI     T113,2,N
  T113   LDQ     G
         FMP     0,2
         CHS
         FAD     0,1
         STO     0,1             ELEMENT REDUCED
  T118   TXI     T119,1,N        NEXT ELEMENT, CURRENT ROW
  T119   TXI     T120,2,N        NEXT ELEMENT, KTH ROW
  T120   TXI     T121,4,1        NEXT COLUMN
  T121   TXL     T113,4,LN
  T122   NOP                     TRANSFER TO T136 FOR XDETRM
         LXD     E3,1            BEGIN REDUCTION OF MATRIX B
         CLA     KM1
         ADD     B
         PDX     0,2
         LXD     LM,4
  T128   LDQ     0,2
         FMP     G
         CHS
         FAD     0,1
         STO     0,1             ELEMENT REDUCED
  T133   TXI     T134,1,N        NEXT ELEMENT, CURRENT ROW
  T134   TXI     T135,2,N        NEXT ELEMENT, KTH ROW
  T135   TIX     T128,4,1
  T136   LXD     E1,1
         TXI     T138,1,1
  T138   SXD     E1,1            NEXT ROW TO BE REDUCED
  T139   TXL     T99,1,LN
         LXD     KP1,1
         TXI     T142,1,1
  T142   TXH     T156,1,LN       REDUCTION COMPLETE
         SXD     KP1,1           K+1
         TIX     T145,1,1
  T145   SXD     K,1             K
         TIX     T147,1,1
  T147   SXD     KM1,1           K-1
         CLA     KM1N
         ADD     N
         STO     KM1N            (K-1)N
         CLA     AKK
         ADD     N
         ADD     =01000000
         STO     AKK
         TRA     T1              BEGIN NEW STAGE
  T156   CLA     AKK
         ADD     N
         ADD     =01000000
         PDX     0,1
         CLA     0,1             LAST PIVOT
         TZE     T251            MATRIX A SINGULAR
         LRS     35
  T163   FMP  ,  D               FINAL VALUE OF
  T164   STO  ,  D                     DETERMINANT
  T165   NOP                     THRU FOR XDETRM
         REM
         REM     BACK SUBSTITUTION
         REM
         SXD     AKK,1
         CLA     LN
         SUB     =01000000
         ADD     B
         STD     E3
         LXD     LM,1
         SXD     E1,1
  T174   LXD     LN,1
         SXD     E4,1
         LXD     AKK,1
         SXD     E2,1
```

```
        LXD    E3,2
        CLA    0,2
        FDP    0,1
        STQ    0,2
T182    LXD    E2,1
        TXI    T184,1,-1
T184    SXD    E2,1             LAST ELEMENT, CURRENT ROW, MATRIX A
        STZ    G
        LXD    E4,4
        TNX    T204,4,1
        SXD    E4,4             ROW TO BE PROCESSED
        LXD    E3,2
T191    LDQ    0,1
        FMP    0,2
        FAD    G
        STO    G
T195    TXI    T196,1, -N
T196    TXI    T197,2,-1
T197    TXI    T198,4,1
T198    TXL    T191,4,LN-1
        CLA    0,2
        FSB    G
        FDP    0,1
        STQ    0,2              VALUE OF UNKNOWN
        TRA    T182
T204    LXD    E3,2
T205    TXI    T206,2,N
T206    SXD    E3,2             LAST ROW, NEXT COLUMN, MATRIX B
        LXD    E1,2
        TNX    T212,2,1
T209    SXD    E1,2             NUMBER OF REMAINING COLUMNS
        TRA    T174             USE (LM-E1+1)TH COLUMN OF B
        REM
        REM    REARRANGEMENT AND PERMANENT STORAGE ASSIGNMENT
        REM
T212    CLA    A
        STD    E1
        CLA    =01000000
        STD    E2
T216    LXD    =0,1
T217    CLA    E2
T218    SUB  ,1 E,1
        TZE    T221
        TXI    T217,1,1
T221    PXD    0,1
        ADD    B
        PDX    0,1
        LXD    E1,2
        LXD    LM,4
T226    CLA    0,1
        STO    0,2
T228    TXI    T229,1,N
T229    TXI    T230,2,N
T230    TIX    T226,4,1
        LXD    LN,4
        TNX    T242,4,1         THRU WITH XSIMEQ
        SXD    LN,4
        CLA    E1
        ADD    =01000000
        STO    E1               FIRST ELEMENT, NEXT ROW, MATRIX A
        CLA    E2
        ADD    =01000000
        STO    E2               NEXT ROW
        TRA    T216
        REM
        REM    FINAL RESULTS
        REM
T242    CLA    =0               SOLUTION SUCCESSFUL
T243    LXD    REG12,1          RESTORE INDEX REGISTERS
        LXA    REG12,2
        LXD    T1-2,4
        LDQ    SAVE             RESTORE LOCATION 8
        STQ    8
T244    STO*   **,4
        TRA    **,4
```

```
T249  CLA    =1.            SPILL
      TRA    T243
T251  LXD    T1-2,4         MATRIX A SINGULAR
      STZ*   **,4
      CLA    =2.
      TRA    T243
      REM
      REM    ENTRY POINTS
      REM
T254  SXD    T1-2,4         ENTRY FOR SIMEQ
      CLA    =9
      STA    T244+1
      SUB    =1
      STA    T244
      SUB    =2
      STA    T251+1
      CLA    =O76100000000  OCTAL CODE FOR NOP
      STO    T38
      STO    T77
      STO    T122
      STO    T165
      STO    T298
      CLA*   3,4            THIRD ARGUMENT (LM)
      STO    LM
      CLA    4,4            FOURTH ARGUMENT (A)
      STA    T282
      STA    T285
      STA    T286
      ALS    18
      STD    A
      CLA    5,4            FIFTH ARGUMENT (B)
      STA    T284
      ALS    18
      STD    B
      CLA    7,4            SEVENTH ARGUMENT (E)
      STA    T218
      STA    T301
      ADD    =1
      STA    T80
      STA    T81
      STA    T82
      STA    T83
      CLA    6,4            SIXTH ARGUMENT (D)
T280  STA    T28
      STA    T29
      STA    T68
      STA    T69
      STA    T89
      STA    T90
      STA    T163
      STA    T164
      STA    T281
      STA    T283
      CLA*   1,4            FIRST ARGUMENT (N)
      STO    N
      CLA*   2,4            SECOND ARGUMENT (LN)
      CAS    =1B17
      TRA    T287
T281  LDQ    **        D
T282  FMP    **        A(1)
T283  STO    **        D
T284  CLA    **        B(1) OR =1.
T285  FDP    **        A(1) OR =1.
T286  STQ    **        A(1) OR A (INTERNAL)
      PXD    ,0
      TRA    T244
T287  STO    LN
      STD    T11
      STD    T20
      STD    T37
      STD    T121
      STD    T139
      STD    T142
      SUB    =1B17
      STD    T198
```

```
        STD    T304
 T298   NOP                    TRANSFER TO T305 FOR XDETRM
        LXD    =0,4
        CLA    =01000000
 T301   STO ,4 E,4             FILL ARRAY E
        ADD    =01000000
        TXI    T304,4,1
 T304   TXL    T301,4,LN-1
 T305   SXD    REG12,1
        SXA    REG12,2
        LDC    A,4
        SXD    A,4
        SXD    AKK,4
        LDC    B,4
        SXD    B,4
        CLA    =01000000
        STO    K
        ADD    =01000000
        STO    KP1
        STZ    KM1
        STZ    KM1N
        CLA    LM
        STD    T53
        CLA    N
        STD    T18
        STD    T34
        STD    T35
        STD    T50
        STD    T51
        STD    T111
        STD    T112
        STD    T118
        STD    T119
        STD    T133
        STD    T134
        STD    T205
        STD    T228
        STD    T229
        LDC    N,4
        SXD    T195,4
        CLA    8
        STO    SAVE
        CLA    SPILL
        STO    8
        TRA    T1
 T343   SXD    T1-2,4          ENTRY FOR DETRM
        CLA    =6
        STA    T244+1
        SUB    =1
        STA    T244
        SUB    =1
        STA    T251+1
        CLA    TRA1
        STO    T38
        CLA    TRA2
        STO    T77
        CLA    TRA3
        STO    T122
        CLA    TRA4
        STO    T165
        CLA    TRA5
        STO    T298
        CAL    T249
        STA    T284
        STA    T285
        CAL    T212
        STA    T286
        CLA    3,4             THIRD ARGUMENT (A)
        STA    T282
        ALS    18
        STD    A
        CLA    4,4             FOURTH ARGUMENT (D)
        TRA    T280
        REM
 TRA1   TRA    T55
```

```
TRA2  TRA    T85
TRA3  TRA    T136
TRA4  TRA    T242
TRA5  TRA    T305
A     PZE                -A
AKK   PZE                -A+(K-1)(N+1)
AKQ   PZE
B     PZE                -B
E1    PZE
E2    PZE
E3    PZE
E4    PZE
G     PZE
I     PZE
J     PZE
K     PZE                STAGE OF REDUCTION
KM1   PZE                K-1
KM1N  PZE                (K-1)N
KP1   PZE                K+1
LM    PZE
LN    PZE
N     PZE
REG12 PZE
SAVE  PZE                CONTENTS OF LOCATION 8
SPILL TRA    TEST        MODIFIED TREATMENT OF UNDERFLOWS
TMP   PZE                CONTENTS OF INDICATORS
TEST  STI    TMP
      LDI    0
      LFT    4
      TRA    OVER        SKIPPED IF UNDERFLOW
      LFT    2
      CLM                SKIPPED IF ONLY MQ UNDERFLOW
      XCA
      LFT    1
      CLM                SKIPPED IF ONLY AC UNDERFLOW
      XCA
      LDI    TMP
      TRA*   0
OVER  LDI    TMP
      TRA    T249
SIMEQ SYN    T254
DETRM SYN    T343
      END
```

PROGRAM LISTINGS

```
***********************
*    SINTBL           *
***********************
 REFER TO
     COSTBL
```


```
***********************
*    SINTBL           *
***********************
 REFER TO
     COSTBL
```


PROGRAM LISTINGS

```
***********************
*    SINTBX           *
***********************
 REFER TO
     COSTBL
```


```
***********************
*    SINTBX           *
***********************
 REFER TO
     COSTBL
```


PROGRAM LISTINGS

```
***********************
*    SISP             *
***********************
 REFER TO
     COSP
```


```
***********************
*    SISP             *
***********************
 REFER TO
     COSP
```

```
*     SIZEUP (SUBROUTINE)              3/15/65   LAST CARD IN DECK IS NO. 0246
*     FAP
*SIZEUP
      COUNT   200
      LBL     SIZEUP
      ENTRY   SIZEUP (X,LX,INDEX)
      ENTRY   SIZUPL (X,LX,INDEX)
*
*
*                   ----ABSTRACT----
*
* TITLE - SIZEUP WITH SECONDARY ENTRY POINT SIZUPL
*        FAST MAKE INDEX (BY INCREASING SIZE) OF ELEMENTS IN A VECTOR.
*
*              SIZEUP  MAKES A VECTOR,  INDEX(I)  I=1...LX,  WHICH GIVES
*              THE ORDERING, WITH RESPECT TO INCREASING SIZE, OF ANOTHER
*              VECTOR,  X(I)  I=1...LX,  SUCH THAT  X( INDEX(I) )  IS
*              ALGEBRAICALLY GREATER THAN OR EQUAL TO  X( INDEX(I-1) )
*              FOR  I=2...LX.  EQUAL VALUES OF  X(I)  WILL NOT
*              NECESSARILY OCCUR IN THE ORDER OF THEIR ORIGINAL
*              APPEARANCE IN THE  X  VECTOR.  +0  IS CONSIDERED GREATER
*              THAN  -0 .  THE INPUT VECTOR  X(I)  MAY BE ANY MODE.
*
*              SIZUPL  PERFORMS THE SAME FUNCTION AS  SIZEUP  EXCEPT
*              THAT THE SORTING IS LOGICAL RATHER THAN ALGEBRAIC.  THAT
*              IS, THE SIGN BIT IS CONSIDERED AS THE HIGHEST NUMERICAL
*              BIT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709, 7090, OR 7094 (MAIN FRAME ONLY)
* STORAGE    - 136 REGISTERS
* SPEED      - AVERAGES ABOUT  .0007*LX  SECONDS ON 7094 MOD 1 FOR
*                 RANDOM NUMBERS BUT WITH DEVIATIONS UP TO 50 PERCENT
*                 FROM THIS FORMULA.
* AUTHORS    - R.A.WIGGINS AND S.M.SIMPSON     AUGUST,1964
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
*
* FORTRAN USAGE OF SIZEUP
*     CALL SIZEUP(X, LX, INDEX)
*
*
* INPUTS
*
*    X(I)       I=1...LX  IS A VECTOR IN ANY MODE.
*
*    LX         LENGTH OF  X  VECTOR.
*               ROUTINE RETURNS WITH NO COMPUTATIONS IF  LSTHN  1 .
*
*
* OUTPUTS
*
*    INDEX(I)  I=1...LX  IS THE VECTOR OF INDICES AS DESCRIBED IN THE
*                 ABSTRACT.
*
*
* FORTRAN USAGE OF SIZUPL  -  SAME AS SIZEUP.
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5) = 3.,-10.,-1.,2.,0.      LX = 5
*    USAGE   -        CALL SIZEUP(X, LX, INDEX1)
*                     CALL SIZUPL(X, LX, INDEX2)
*    OUTPUTS - INDEX1(1...5) = 2,3,5,4,1
*              INDEX2(1...5) = 5,4,1,3,2
*
* 2. INPUTS  - X(1...5) = 1HX,1HA,1HC,1HN,1HA      LX = 5
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - INDEX1(1...5) = 1,4,5,2,3
```

```
*              INDEX2(1...5) = 5,2,3,4,1
*
*
* PROGRAM FOLLOWS BELOW.
XR4    HTR     0
       BCI     1,SIZEUP
SIZEUP STZ     ZIFALG         ALGEBRAIC SORTING ENTRY
       TRA     SIZUPL+1
SIZUPL SXD     ZIFALG,4       LOGICAL SORTING ENTRY
       SXD     XR4,4          SAVE
       SXA     XR2,2          INDEX
       SXA     XR1,1          REGISTERS, AND
       STI     INDIC          INDICATORS.
       CLA     =1B17
       STO*    3,4
       CLA*    2,4            GET LX.
       PDX     ,1
       SXA     LX,1
       CAS     =0             CHECK IF LX IS LEGAL.
       TIX     *+3,1,1        LEGAL
       TRA     XR1            ILLEGAL OR LX=1
       TRA     XR1            ILLEGAL
       SXD     LX1,1
       CAL     3,4            GET
       ADD     =1             INDEX
       STA     IX1            ADDRESS
       STA     OFT1           AND
       STA     OFT2           SPREAD
       STA     IX2            IT
       STA     IX3            AROUND.
       STA     IX4
       STA     IX5
       STA     IX6
       STA     IX7
       STA     IX8
       STA     IX9
       STA     IX10
       CAL     1,4            GET
       PAX     ,2
       TXI     *+1,2,1
       SXA     X1,2
       SXA     X2,2
       SXD     X3,2
       PXA     ,2
       SUB     LX             SET UP INDEX VECTOR.
       TXI     *+1,1,1
IX1    STO     **,1           **=ADR(INDEX)+1
       ADD     =1
       TIX     IX1,1,1
       CLA     TMIPL
       SSM
       STO     TMIPL
CHSIGN ZET     ZIFALG         IF THIS IS ALGEBRAIC SORT.
       TRA     CONT
       LXA     LX,1
X1     CLS     **,1           **=ADR(X)+1
TMIPL  TMI     *+2            =TMI FIRST PASS, TPL SECOND PASS
       COM
X2     STO     **,1           **=ADR(X)+1
       TIX     X1,1,1
CONT   CLS     TMIPL
       TMI     XIT1
       STO     TMIPL          CONTINUE.
       AXT     1,1            SET UP
       SXA     IFTB,1         INDEX
       LXA     LX,2           REGISTERS
       SXA     ILTB,2         FOR
       AXT     0,4            BEGINNING.
       LDQ     =-1B17         FLAG LAST
       SLQ*    IX3            INDEX.
*
* THIS IS BEGINNING OF MAIN PROCESSING LOGIC.
*
* SET UP THE INDICATORS FROM IBIT=XR4 TO SCAN ON A PARTICULAR BIT.
*
```

```
SETIND CAL      =-0
       SXA      *+1,4
       ARS      **
       PAI
*
* SAVE THE CURRENT SCANNING LIMITS.
*
       SXD      ALSMC,1         XR1 IS INDEX FOR FORWARD SCANNING.
       SXD      SCNDN,2         XR2 IS INDEX FOR BACK SCANNING.
*
* SCAN DOWN THE VECTOR FROM IFTB LOOKING FOR 1'S.
*
SCNDN  TXH      ALSM,1,**       **=ILTB TRANSFER IF ALL BITS ARE SAME.
OFT1   OFT*     **,1            **=ADR(INDEX)+1
       TRA      SETLM           ONE BIT, GO SCAN UP.
       TXI      SCNDN,1,1       ZERO BIT, CONTINUE SCAN.
*
* SCAN UP THE VECTOR FROM ILTB TO IBF=IX1 LOOKING FOR 0'S.
*
SETLM  SXD      SCNUP,1
SCNUP  TXL      ALSMC,2,**      **=IBF TRANSFER IF (IX2=IBL)=IBF
OFT2   OFT*     **,2            **=ADR(INDEX)+1
       TIX      SCNUP,2,1       ONE BIT, CONTINUE SCAN.
IX2    CLA      **,1            ZERO BIT,
IX3    LDQ      **,2            EXCHANGE
IX4    STA      **,2            THE
       XCA                      INDEX
IX5    STA      **,1            ADDRESSES AND
       TXI      OFT1,1,1        CONTINUE SCANNING.
*
* THE EXIT FROM SCNUP DOESN'T INDICATE WHETHER ALL BITS WERE SAME.
* CHECK THIS.
*
ALSMC  TXH      BMPR,2,**       **=IFTB TRANSFER IF MIXED BITS
*
* ALL THE BITS WERE ONES OR ZEROS, SCAN ON NEXT BIT.
*
ALSM   LXA      ILTB,2          RESET IBL,
ALSM1  LXA      IFTB,1          AND RESET IBF.
       PXD      ,4              RECORD THE
       SSM
       XCA
IX6    SLQ      **,2            NEW BIT INDICATOR
       TXI      *+1,4,1         BUMP IBIT
       TXL      NWILTD,4,35     AND GO TO SCAN.
*
* IF THAT WAS ALL THE BITS, SEEK A NEW RANGE.
*
RECON  CLA      ILTB            NEXT RANGE STARTS
       ADD      =1              ONE REGISTER AFTER LAST ILTB.
       PAX      ,1              SET NEW IBF.
LX1    TXH      EXIT,1,**       **=LX-1 TRANSFER IF LAST WORD IN LIST.
       SXA      IFTB,1          SET NEW IFTB.
       LXA      IFTB,2          AND SCAN
       CLA      =0              INDEX VECTOR
IX7    CAS      **,2            **=ADR(INDEX)+1
       TRA      IX8             FOR NEXT
       NOP
       TXI      IX7,2,1         NEGATIVE VALUE
IX8    CLA      **,2            THAT DEFINES
       PDX      ,4              NEW IBIT, AND
NWILTD SXA      ILTB,2          THAT DEFINES NEW ILTB.
       SXD      *+1,1           IF NEGATIVE IS IN IFTB REGISTER
       TXL      RECON,2,**      GO BUMP ILTB AGAIN.
       TRA      SETIND          GO BACK FOR MORE SCANS.
*
* CONTROL COMES HERE WHEN IBF MEETS IBL IN MID VECTOR.
*
BMPR   TIX      ALSM1,2,1       BACK OFF ONE ON IBL.
*
* END PROCEDURE.  INTERPRET INDEX.
*
EXIT   LXD      XR4,4
       LXA      LX,1
IX9    CAL      **,1
```

SIZEUP
(PAGE 4)

```
       PAC     ,2
X3     TXI     *+1,2,**          **=ADR(X)+1
       PXD     ,2
IX10   STO     **,1
       TIX     IX9,1,1
*
* GO RESTORE X
*
       TRA     CHSIGN
XIT1   LDI     INDIC
XR1    AXT     **,1
XR2    AXT     **,2
       TRA     4,4
*
* DATA
*
INDIC  PZE
LX     PZE
ZIFALG PZE
IFTB   PZE
ILTB   PZE
       END
```

```
************************
*    SIZUPL            *
************************
 REFER TO
     SIZEUP
```


```
************************
*    SIZUPL            *
************************
 REFER TO
     SIZEUP
```


```
************************
*    SMPRDV            *
************************
 REFER TO
     POWER
```


```
************************
*    SMPRDV            *
************************
 REFER TO
     POWER
```

```
*     SMPSON (SUBROUTINE)                  9/4/64   LAST CARD IN DECK IS NO. 0196
*     LABEL
CSMPSON
      SUBROUTINE SMPSON(JOB,X,LX,DELX,XINT,IANS)
C
C
C                    ----ABSTRACT----
C
C TITLE - SMPSON
C        UNSCALE OR SCALE VECTOR FOR SIMPSON INTEGRAL AND/OR INTEGRATE.
C
C               SMPSON WILL SCALE AN INPUT VECTOR ACCORDING TO THE
C               SIMPSON'S RULE AND RETURN THE SCALED VECTOR AND THE
C               INTEGRAL, OR WILL RETURN THE INTEGRAL AND THE ORIGINAL
C               VECTOR, OR WILL UNSCALE A VECTOR WHICH HAS BEEN SCALED
C               FOR SIMPSON'S RULE.  IF THE DATA LENGTH IS EVEN THE LAST
C               POINT IS INTEGRATED BY THE TRAPEZOIDAL RULE.  THE
C               SIMPSON'S RULE SCALES FOR ODD DATA LENGTH ARE
C
C                    DELX*(1/3,4/3,2/3,4/3,...,4/3,1/3)
C
C               AND FOR EVEN DATA LENGTH  SMPSON  USES
C
C                    DELX*(1/3,4/3,2/3,4/3,...,4/3,5/6,1/2).
C
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  (MAIN FRAME ONLY)
C STORAGE    - 317 REGISTERS
C SPEED      - TAKES ABOUT  25*LX  MACHINE CYCLES TO OBTAIN THE
C                 INTEGRAL, TO SCALE OR TO UNSCALE.
C AUTHOR     - J.N.GALBRAITH,JR.,   FEBRUARY 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C      CALL SMPSON(JOB,X,LX,DELX,XINT,IANS)
C
C
C INPUTS
C
C    JOB       FORTRAN II INTEGER INDICATES WHICH JOB IS TO BE DONE.
C              =      0  INTEGRATE BUT LEAVE DATA UNSCALED.  (DATA=X(I).)
C              GRTHN 0  SCALE DATA AND INTEGRATE.
C              LSTHN 0  UNSCALE DATA.
C
C    X(I)      I=1,LX  INPUT FLOATING POINT VECTOR FOR SMPSON OPERATIONS
C
C    LX        FORTRAN II INTEGER. LENGTH OF X(I) VECTOR. GRTHN 3.
C
C    DELX      FLOATING POINT.  SPACING BETWEEN  X  VALUES.
C              SHOULD BE NON-ZERO.
C
C
C OUTPUTS
C
C    X(I)      I=1...LX  IS UNCHANGED FOR  JOB = 0 .
C                        IS SCALED, AS DEFINED IN ABSTRACT,
C                                                 FOR JOB GRTHN 0 .
C                        IS UNSCALED (SCALED BY RECIPROCALS)
C                                                 FOR JOB LSTHN 0 .
C
C    XINT      SIMPSON*S RULE INTEGRAL. (NOT CHANGED IF JOB LSTHN 0.)
C
C    IANS      FORTRAN II INTEGER ERROR INDICATOR.
C              =  0  NORMAL
C              = -3  ILLEGAL  LX
C
C
C EXAMPLES
C
```

```
C 1. INPUTS  - JOB = 0   X(1...10) = 1.,2.,3.,4.,3.,3.,4.,2.,6.,7.
C              LX = 3    DELX = 0.2
C    OUTPUTS - IANS = -3
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX = 4.
C    OUTPUTS - IANS = 0  XINT = 1.5  X(1...4) = 1.,2.,3.,4.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX = 9.
C    OUTPUTS - IANS = 0  XINT = 4.7333333
C              X(1...9) = 1.,2.,3.,4.,3.,3.,4.,2.,6.
C
C 4. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX = 10.
C    OUTPUTS - IANS = 0  XINT = 6.0333333
C              X(1...10) = 1.,2.,3.,4.,3.,3.,4.,2.,6.,7.
C
C 5. INPUTS  - JOB = 1   X(1...10) = 1.,2.,3.,4.,3.,3.,4.,2.,6.,7.
C              LX = 10   DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 6.0333332
C              X(1...10) = 0.0666667, 0.5333333, 0.4, 1.0666667, 0.4,
C                          0.8, 0.5333333, 0.5333333, 1.0, 0.7
C
C 6. INPUTS  - JOB = -1  X(1...10) = SAME AS OUTPUTS FROM EXAMPLE 5.
C              LX = 10   DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 0.
C              X(1...10) = 1.,2.,3.,4.,3.,3.,4.,2.,6.,7.
C
C 7. INPUTS  - JOB = 2   X(1...9) = 1.,2.,3.,4.,3.,3.,4.,2.,6.
C              LX = 9    DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 4.7333332
C              X(1...9) = 0.0666667, 0.5333333, 0.4, 1.0666666, 0.4,
C                         0.8, 0.5333333, 0.5333333, 0.4
C
C 8. INPUTS  - JOB = -2  X(1...9) = SAME AS OUTPUTS FROM EXAMPLE 7.
C              LX = 9    DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 0.
C              X(1...9) = 1.,2.,3.,4.,3.,3.,4.,2.,6.
C
C 9. INPUTS  - JOB = 0   X(1...9) = -1.,2.,3.,4.,-3.,3.,4.,2.,6.
C              LX = 9    DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 3.7999998
C              X(1...9) = SAME AS INPUTS
C
C10. INPUTS  - SAME AS EXAMPLE 9. EXCEPT JOB = 1.
C    OUTPUTS - IANS = 0  XINT = 3.7999998
C              X(1...9) = -0.0666667, 0.5333333, 0.4, 1.0666666, -0.4,
C                         0.8, 0.5333333, 0.5333333, 0.4
C
C11. INPUTS  - JOB = -1  X(1...9) = SAME AS OUTPUTS FROM EXAMPLE 10.
C              LX = 9    DELX = 0.2
C    OUTPUTS - IANS = 0  XINT = 0.
C              X(1...9) = -1.,2.,3.,4.,-3.,3.,4.,2.,6.
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION X(100)
      IANS=-3
      IF(LX-3) 99,99,2
 2    IANS=0
      XINT=0.
      SCALE=DELX/3.
      IF((LX/2)*2-LX) 10,5,10
C
C     LX EVEN
C
 5    JSWTCH=1
      GO TO 15
C
C     LX ODD
C
 10   JSWTCH=0
 15   NN=LX-2
      IF(JOB) 60,20,40
C
```

****
* SMPSON *
****
(PAGE 3)

```
C     INTEGRATE BUT DO NOT SCALE.
C
 20   DO 25 I=2,NN,2
 25   XINT=XINT+4.*X(I)+2.*X(I+1)
      IF(JSWTCH) 35,30,35
 30   XINT=(XINT+X(1)+X(LX)+4.*X(LX-1))*SCALE
      GO TO 99
C
C     DO LAST POINT BY TRAPEZOIDAL RULE
C
 35   XINT=(XINT+X(1)+.5*X(LX-1)+1.5*X(LX))*SCALE
      GO TO 99
C
C     SCALE VECTOR AND INTEGRATE.
C
 40   FACT1=SCALE*4.
      FACT2=SCALE*2.
      DO 45 I=2,NN,2
      X(I)=X(I)*FACT1
      X(I+1)=X(I+1)*FACT2
 45   XINT=XINT+X(I)+X(I+1)
      X(1)=X(1)*SCALE
      IF(JSWTCH) 55,50,55
 50   X(LX)=X(LX)*SCALE
      X(LX-1)=X(LX-1)*FACT1
      XINT=XINT+X(1)+X(LX)+X(LX-1)
      GO TO 99
 55   X(LX-1)=X(LX-1)*1.25
      X(LX)=X(LX)*1.5*SCALE
      XINT=XINT+X(1)+X(LX)+X(LX-1)*.2
      GO TO 99
C
C     UNSCALE VECTOR
C
 60   FACT1=.25/SCALE
      FACT2=.5/SCALE
      DO 65 I=2,NN,2
      X(I)=X(I)*FACT1
 65   X(I+1)=X(I+1)*FACT2
      X(1)=X(1)/SCALE
      IF(JSWTCH) 75,70,75
 70   X(LX)=X(LX)/SCALE
      X(LX-1)=X(LX-1)*FACT1
      GO TO 99
 75   X(LX-1)=.8*X(LX-1)
      X(LX)=X(LX)/(1.5*SCALE)
 99   RETURN
      END
```

```
*     SPCOR2  (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0180
*     LABEL
CSPCOR2
      SUBROUTINE SPCOR2 (NRX,NCX,XX,NRY,NCY,YY,MXACC,ILGR,NRZ,
     1                  ILGC,INC,NCZ,ZZ,SPACE,IANS)
C
C                    ----ABSTRACT----
C
C TITLE - SPCOR2
C        SPATIAL CROSSCORRELATION OF 2-DIMENSIONAL SPATIAL ARRAYS
C
C              SPCOR2 EVALUATES THE SPATIAL CROSSCORRELATION OF AN
C              ARRAY X(I,J) I=1,...,NRX  J=1,...,NCX  WITH ANOTHER
C              ARRAY Y(I,J) I=1,...,NRY  J=1,...,NCY
C
C                          NCX   NRX
C                Z(I,J) =  SUM ( SUM ( X(K+I-1,L+J)*Y(K,L) ) )
C                          K=1   L=1
C
C                FOR I = ILGR,...,ILGR+NRZ-1
C                    J = ILGC,...,ILGC+NCZ-1
C                    J1= ILGC,ILGC+INC,...,ILGC+(NCZ-1)*INC
C              WHERE
C                    NRX, NCX, NRY, NCY, ILGR, NRZ, ILGC, INC, AND NCZ
C                      ARE INPUT PARAMETERS.
C                    X(I,J) AND Y(I,J) ARE TREATED AS ZERO WHEN I AND J
C                      ARE OUTSIDE THE RANGE OF DEFINITION.
C
C              SPEED IS OBTAINED BY FIXING X(I,J) AND Y(I,J) BETWEEN
C              THE LIMITS OF -MXACC AND MXACC AND THEN USING THE HIGH-
C              SPEED LOGIC OF PROCOR, FASCR1, AND FASEP1 VIA QXCOR1.
C
C              X(I,J) AND Y(I,J) ARE SLIGHTLY MODIFIED BY THE FIXING
C              AND REFLOATING PROCESS.
C
C              USER MUST PROVIDE SPCOR1 WITH A BLOCK OF TEMPORARY
C              REGISTERS OF LENGTH  MIN(NRX,NRY) + 10*(MXACC+1) + 1
C
C LANGUAGE   - FORTRAN II SUBROUTINE
C EQUIPMENT  - 709, 7090, 7094  (MAIN FRAME ONLY)
C STORAGE    - 291  REGISTERS
C SPEED      - HALF OF THE COMPLETE AUTOCORRELATION OF AN ARRAY
C              TAKES ABOUT
C                 (.0070 + .0000040*(NRX SQUARED))*(NCX SQUARED)
C              SECONDS ON THE 7094 MOD 1.
C AUTHOR     - R.A. WIGGINS  JULY, 1963
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FLDATA,FXDATA,QXCOR1,STZ
C       AND FORTRAN SYSTEM ROUTINES - XLOC
C
C FORTRAN USAGE
C     CALL SPCOR2(NRX,NCX,XX,NRY,NCY,YY,MXACC,ILGR,NRZ,
C    1             ILGC,INC,NCZ,ZZ,SPACE,IANS)
C
C INPUTS
C
C    NRX       NUMBER ROWS OF X.
C              MUST BE GRTHN= 1
C
C    NCX       NUMBER COLUMNS OF X.
C              MUST BE GRTHN= 1
C
C    XX(K)     K=1,...,NRX*NCX CONTAINS THE ARRAY X(I,J) I=1,...,NRX
C                J=1,...,NCX STORED CLOSELY PACKED BY COLUMNS.
C
C    NRY       NUMBER ROWS OF Y.
C              MUST BE GRTHN= 1
C
C    NCY       NUMBER COLUMNS OF Y.
C              MUST BE GRTHN= 1
C
C    YY(K)     K=1,...,NRY*NCY CONTAINS THE ARRAY Y(I,J) I=1,...,NRY
```

```
C                J=1,...,NCY STORED CLOSELY PACKED BY COLUMNS.
C
C    MXACC     DEFINES THE ACCURACY OF THE TWO ARRAYS.  XX(I)  AND YY(I)
C                WILL BE FIXED SO AS TO HAVE VALUES LYING BETWEEN -MXACC
C                AND +MXACC INCLUSIVE.
C
C    ILGR      INITIAL LAG ALONG ROWS FOR CROSSCORRELATION AS DEFINED
C                IN THE ABSTRACT.
C
C    NRZ       NUMBER ROWS IN Z  I.E. THE NUMBER OF LAGS ALONG THE ROWS
C                IN THE CROSSCORRELATION.
C              MUST BE GRTHN= 1
C
C    ILGC      INITIAL LAG ALONG COLUMNS FOR CROSSCORRELATION AS DEFINED
C                IN THE ABSTRACT.
C
C    INC       INCREMENT IN THE LAG ALONG THE COLUMNS AS DEFINED IN THE
C                ABSTRACT.
C              MUST BE GRTHN= 1
C
C    NCZ       NUMBER COLUMNS IN Z  I.E. THE NUMBER OF LAGS ALONG THE
C                COLUMNS FOR WHICH THE CROSSCORRELATION IS EVALUATED.
C
C    SPACE(I)  I=1,...,LSPACE IS TEMPORARY COMPUTATION SPACE NEEDED BY
C                SPCOR1, WHERE
C                    LSPACE = MIN(NRX,NRY) + 10*(MXACC+1) + 1
C
C OUTPUTS
C
C    ZZ(K)     K=1,...,NRZ*NCZ CONTAINS Z(I,J) I=ILGR,...,ILGR+NRZ-1
C                J=ILGC,...,ILGC+NCZ-1 AS DEFINED IN THE ABSTRACT.
C
C    IANS      =0 IF NO TROUBLE
C              =1 IF ILLEGAL NRX
C              =3 IF ILLEGAL NRY
C              =5 IF ILLEGAL MXACC
C              =7 IF ILLEGAL NRZ
C
C
C EXAMPLES
C
C 1. INPUTS  - NRX = 3  NCX = 3  XX(1...9) = 0.1  0.4  0.7    (STORED BY
C                                            0.2  0.5  0.8     COLUMNS)
C                                            0.3  0.6  0.9
C              NRY = 2  NCY = 3  YY(1...6) = 4.9  4.7  4.5    (STORED BY
C                                            4.8  4.6  4.4     COLUMNS)
C              MXACC = 100  ILGR = -2  NRZ = 4  ILGC = -2  INC = 1
C              NCZ = 4
C    OUTPUTS - IANS = 0  ZZ(1...16) = 0.00  0.00  0.00  0.00  (STORED BY
C                                     0.44  2.20  5.40  5.14   COLUMNS)
C                                     1.32  5.39 12.32 11.35
C                                     2.20  7.23 15.12 13.28
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT  INC = 2
C    OUTPUTS - IANS = 0  ZZ(1...8)  = 0.00  0.00     (STORED BY COLUMNS)
C                                     0.44  5.40
C                                     1.32 12.32
C                                     2.20 15.12
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION XX(2),YY(2),ZZ(2),SPACE(2)
C CALCULATE FUNDAMENTAL CONSTANTS
      LX=NRX*NCX
      LY=NRY*NCY
      LZ=NRZ*NCZ
      LSPACE=XMINOF(NRX,NRY)+1+10*(MXACC+1)
      IAUTO = XLOCF(XX)-XLOCF(YY)
C CLEAR THE OUTPUT AREA
      CALL STZ (LZ,ZZ)
C FIX THE INPUTS
      CALL FXDATA (LX,XX,MXACC,SCLX)
      SCLY=SCLX
      IF (IAUTO)  10,15,10
```

```
 10   CALL FXDATA (LY,YY,MXACC,SCLY)
 15   CONTINUE
C DO CORRELATIONS
      IDX=NRX*INC
      IX1=XMAXOF(0,ILGR)+ILGC*NRX+1
      NRX1=NRX-XMAXOF(0,ILGR)
      ILGR1=XMINOF(0,ILGR)
CXXXXXXXXXX
      DO 50  I2=1,LZ,NRZ
      IX2=IX1
CXXXXXXXXXXXXXXXXXXXX
      DO 40  I3=1,LY,NRY
      IF (IX2) 30,30,19
 19   IF (LX-IX2) 30,30,20
 20   CALL QXCOR1 (NRX1,XX(IX2),NRY,YY(I3),MXACC,ILGR1,NRZ,ZZ(I2),1,
     1      LSPACE,SPACE,IANS)
      IF (IANS)  30,30,60
 30   CONTINUE
 40   IX2=IX2+NRX
CXXXXXXXXXXXXXXXXXXXX
 50   IX1=IX1+IDX
CXXXXXXXXXX
C REFLOAT EVERYTHING
 60   CONTINUE
      CALL FLDATA (LX,XX,SCLX)
      CALL FLDATA (LZ,ZZ,SCLX*SCLY)
      IF (IAUTO)  70,80,70
 70   CALL FLDATA (LY,YY,SCLY)
 80   CONTINUE
C THAT'S ALL THERE IS TO IT.
      RETURN
      END
```

```
*       SPLIT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0394
*       FAP
*SPLIT
        COUNT   400
        LBL     SPLIT
        ENTRY   SPLIT (X,LX,TYPE,SYM,ANT)
        ENTRY   REFIT (X,LX,TYPE,SYM,ANT)
*
*                       ----ABSTRACT----
*
* TITLE - SPLIT  WITH SECONDARY ENTRY POINT REFIT
*         SPLIT A VECTOR INTO ITS EVEN AND ODD PARTS (OR INVERSE)
*
*               SPLIT FINDS THE SYMMETRIC AND ANTISYMMETRIC PARTS OF A
*               FIXED OR FLOATING POINT VECTOR.  THE ORIGIN IS ASSUMED
*               TO BE AT THE MIDPOINT OF THE VECTOR.  THE VECTOR MAY BE
*               OF EVEN OR ODD LENGTH.  STORAGE OF THE PARTS ON TOP OF
*               THE VECTOR IS PERMITTED.
*
*               REFIT PUTS THE PARTS BACK TOGETHER TO REFORM THE VECTOR.
*
* LANGUAGE  - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709, OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 224 REGISTERS
* SPEED     - SPLIT (FIXED)- ABOUT 180  + 23*LX  MACHINE CYCLES
*                   (FLTG) - ABOUT 180  + 34*LX  MACHINE CYCLES
*             REFIT (FIXED)- ABOUT 180  + 23*LX  MACHINE CYCLES
*                   (FLTG) - ABOUT 180  + 68*LX  MACHINE CYCLES
*                WHERE LX = LENGTH OF SERIES
* AUTHOR    - S.M. SIMPSON
*
*                       ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -    NONE
*       AND FORTRAN SYSTEM ROUTINES -    NONE
*
* FORTRAN USAGE OF SPLIT
*     CALL  SPLIT(X,LX,TYPE,SYM,ANT)    FOR FLTG. PT. DATA
*
*     (FOR FIXED PT. DATA   X,SYM AND ANT WOULD BE FIXED POINT NAMES)
*
* INPUTS TO SPLIT
*
*    X(I)       I=1...LX IS A FIXED OR FLOATING POINT VECTOR
*
*    LX         IS FORTRAN II INTEGER = LENGTH OF X SERIES
*               SHOULD EXCEED ZERO
*                 (IF LX IS LSTHN= 0 PROGRAM EXITS WITH NO OUTPUT)
*
*    TYPE       = 0.0 SIGNIFIES X(I) IS FIXED POINT
*               = 1.0 SIGNIFIES X(I) IS FLOATING POINT
*
* OUTPUTS
*
*    SYM(I)     I=1...LS  HOLDS SYMMETRIC PART, WHERE
*                 FOR LX ODD, LS = (LX+1)/2   AND
*                        SYM(1) = X(LS)
*                        SYM(I) = X(LS-1+I) + X(LS+1-I)   I=2,3,...,LS
*                 FOR LX EVEN, LS = LX/2  AND
*                        SYM(I) = X(LS+I) + X(LS+1-I)   I=1,2,...,LS
*
*    ANT(I)     I=1...LA  HOLDS ANTISYMMETRIC PART, WHERE
*                 FOR LX ODD, LA = (LX-1)/2  AND
*                       ANT(I) = X(LS+I) - X(LS-I)     I= 1,2,...,LA
*                 FOR LX EVEN, LA = LX/2  AND
*                       ANT(I) = X(LA+I) - X(LA+1-I)  I= 1,2,...,LA
*               (ANT(I) IS AN OUTPUT ONLY IF LA IS GRTHN=1)
*
*               (SYM AND ANT WILL BE FIXED OR FLOATING ACCORDING TO TYPE)
*               STORAGE OF SYM AND ANT ON TOP OF X SERIES IS PERMITTED
*               ONLY IF SYM(1) IS EQUIVALENT TO X(1)
*                   AND ANT(1) IS EQUIVALENT TO X(LS+1)
*
* FORTRAN USAGE OF REFIT
```

```
      *     CALL REFIT (X,LX,TYPE,SYM,ANT)   FOR FLTG. PT. DATA
      *
      *      (FOR FIXED PT. DATA X, SYM AND ANT WOULD BE FIXED PT. NAMES)
      *
      * INPUTS TO REFIT
      *
      *    LX         SAME MEANING AS FOR SPLIT
      *
      *    TYPE       SAME MEANING AS FOR SPLIT
      *
      *    SYM(I)     I=1...LS  IS SYMMETRIC PART
      *
      *    ANT(I)     I=1...LA  IS ANTISYMMETRIC PART (NOT USED IF LA=0)
      *
      * OUTPUTS FROM REFIT
      *
      *    X(I)       I=1...LX IS REFITTED SERIES FROM SYM AND ANT, WHERE
      *                 FOR LX ODD
      *                   X(I) = (SYM(LS+1-I) - ANT(LS-I))/2.    I=1...LS-1
      *                   X(LS) = SYM(1)
      *                   X(I) = (SYM(I-LA) + ANT(I-LA-1)/2.    I=LS+1...LX
      *                 FOR LX EVEN
      *                   X(I) = (SYM(LS+1-I) - ANT(LS+1-I))/2. I=1...LS
      *                   X(I) = (SYM(I-LS) + ANT(I-LS))/2.     I=LS+1...LX
      *
      *               (NOTE- FOR FIXED DATA, DIVISION BY 2 INCLUDES ROUNDING
      *                INTO BIT 35)
      *
      * EXAMPLES
      *
      * 1. PARTS AWAY, REFIT AWAY, LX ODD, FIXED AND FLOATING
      *    INPUTS  - X(1...7) = 80.,60.,50.,40.,30.,20.,10.
      *              IX(1...7)= 80,60,50,40,30,20,10   LX=7
      *    USAGE   -         CALL SPLIT (X,LX,1.0,SYM,ANT)
      *                      CALL REFIT (Y,LX,1.0,SYM,ANT)
      *                      CALL SPLIT (IX,LX,0.0,ISYM,IANT)
      *                      CALL REFIT (IY,LX,0.,ISYM,IANT)
      *    OUTPUTS - SYM(1...4) = 40.,80.,80.,90.  ANT(1...3) = -20.,-40.,-70.
      *                Y(1...7) = X(1...7)
      *              ISYM(1...4) = 40,80,80,90  IANT(1...3) = -20,-40,-70
      *                IY(1...7) = IX(1...7)
      *
      * 2. PARTS AWAY, REFIT AWAY, LX EVEN, FIXED AND FLOATING
      *    INPUTS  - SAME AS EXAMPLE 1. EXCEPT LX=6
      *    USAGE   - SAME AS EXAMPLE 1.
      *    OUTPUTS - SYM(1...3) = 90.,90.,100.  ANT(1...3) = -10.,-30.,-60.
      *                Y(1...6) = X(1...6)
      *              ISYM(1...3) = 90,90,100  IANT(1...3) = -10,-30,-60
      *                IY(1...6) = IX(1...6)
      *
      * 3. PARTS ON TOP, REFIT ON TOP, LX ODD, FIXED AND FLOATING
      *    INPUTS  - SAME AS EXAMPLE 1.
      *    USAGE   -         CALL SPLIT (X,LX,1.0,X,X(5))
      *                      CALL REFIT (X,LX,1.0,X,X(5))
      *                      CALL SPLIT (IX,LX,0.0,IX,IX(5))
      *                      CALL REFIT (IX,LX,0.0,IX,IX(5))
      *    OUTPUTS - X(1...7) = 80.,60.,50.,40.,30.,20.,10.
      *              IX(1...7) = 80,60,50,40,30,20,10
      *              (NOTE- FOLLOWING FIRST CALL OF SPLIT X(1...7) =
      *                  40.,80.,80.,90.,-20.,-40.,-70.)
      *
      * 4. PARTS ON TOP, REFIT ON TOP, LX EVEN, FIXED AND FLOATING
      *    INPUTS  - SAME AS EXAMPLE 2.
      *    USAGE   -         CALL SPLIT (X,LX,1.0,X,X(4))
      *                      CALL REFIT (X,LX,1.0,X,X(4))
      *                      CALL SPLIT (IX,LX,0.0,IX,IX(4))
      *                      CALL REFIT (IX,LX,0.0,IX,IX(4))
      *    OUTPUTS - X(1...6) = 80.,60.,50.,40.,30.,20.
      *              IX(1...6) = 80,60,50,40,30,20
      *
      * 5. CHECK ON SPECIAL CASES LX=1, LX=2
      *    INPUTS  - SAME AS EXAMPLE 1.
      *    USAGE   -         CALL SPLIT (X,1,0.0,SYM,ANT)
      *                      CALL REFIT (Y,1,0.0,SYM,ANT)
      *                      CALL SPLIT (IX,2,0.0,ISYM,IANT)
```

```
     *                     CALL REFIT (IY,2,0.0,ISYM,IANT)
     *    OUTPUTS - SYM(1) = 80.  ANT(1) = UNDEFINED  Y(1) = 80.
     *              ISYM(1) = 140  IANT(1) = -20  IY(1...2) = 80,60
     *
     * PROGRAM FOLLOWS BELOW
     *    NOTATION EQUIVALENCES USED IN PROGRAM NOTES
     *              M = LX
     *              N = LS
     *              P = LA
     *
             HTR      0
             BCI      1,SPLIT
       SPLIT STZ      S65            SET PROGRAM INDICATOR = 0
             TRA      S2
       REFIT CLA      S55            SET PROG INDIC = 1
             STA      S65
          S2 SXD      SPLIT-2,4      SAVE IR4 FOR STD ERROR PROC
             SXA      S49+1,1
             SXA      S49+2,2
             CLA*     3,4           SET FIXED-FLOATING
             STO      S64           INDICATOR
             CLA*     2,4            GET M
             ARS      18             IN ADDRESS
             STO      S62            STORE IT
             CAS      S67            CHECK M
             TRA      *+3            M GRTR 1
             TRA      S200           SPECIAL CASE  M=1
             TRA      S49            ILLEGAL M,  GO EXIT
             LRS      1              FORM P=M/2 OR (M-1)/2
             STO      S63            STORE P
             STO      S68            STORE TRIAL N=P
             LLS      1              CHECK IF M
             LBT                     ODD OR EVEN
             TRA      S3             EVEN
             TRA      S4             ODD
          S3 STZ      S61            SET EVEN-ODD INDIC=0 (EVEN)
             TRA      S18           (TRIAL N OK)
          S4 CLA      S55            SET EVEN-ODD INDIC =1 (ODD)
             STA      S61
             CLA      S68            INDEX TRIAL N BY 1
             ADD      S67
             STO      S68
         S18 CLA      S63            SET P IN DECREMENT OF
             ALS      18             ONE LOOP COUNTER
             STD      S39
             CLA      S68            SET
             ALS      17             N/2 TRUNCATED
             STD      S80            IN DECREMENT
             CLA      4,4            SET
             ADD      S67            XS+1 ADDR
             STA      S81
             STA      S83            AND
             SUB      S67            XS-N ADDR.
             SUB      S68
             STA      S82            IN REVERSE SYM LOOP
             STA      S84
             NZT      S65            IS IT SPLIT OR REFIT
             TRA      S6             (SPLIT IF ZERO)
    *SET UP EXCHANGE AND MIDPOINT ROUTINES FOR REFIT
             CLA      S81            SET XS+1 ADDRESS IN
             STA      S45            EXCHANGE
             STA      S85            AND MIDPOINT
             CLA      1,4            SET X+1 ADDRESS IN
             ADD      S67            EXCH
             STA      S38            AND MDPT
             STA      S86
             SUB      S67            SET X-M ADDRESS
             SUB      S62            IN
             STA      S36            EXCH LOOP
             CLA      5,4            SET XA-P ADDRESS
             SUB      S63            IN
             STA      S32            EXCH LOOP
             CLA      S53            SET S59
             STA      S34+1          STO ADDRESS
             CLA      S36+1          SET S60
```

SPLIT
(PAGE 4)

```
          STA    S31+1       STO ADDRESS
          ZET    S64         FXD OR FLTG
          TRA    S100        FLTG
          TRA    S101        FXD
    S100  CLA    S50         SET FDP INSTR
          STO    S30
          STO    S33
          CLA    S52         SET XCA INSTR
   S102   STO    S31
          STO    S34
          TRA    S7
    S101  CLA    S55         SET LRS INSTR
          STO    S30
          STO    S33
          CLA    S56         SET RND INSTR
          TRA    S102
*SET UP EXCHANGE AND MIDPOINT ROUTINES FOR SPLIT
      S6  CLA    S36+1       SET S60
          STA    S34+1       STO ADDRESS
          CLA    S53         SET S59
          STA    S31+1       STO ADDRESS
          CLA    S81         SET XS+1 ADDRESS
          STA    S86         IN MDPT
          STA    S36         IN EXCH
          CLA    1,4         SET X+1 ADDRESS
          ADD    S67
          STA    S85         IN MDPT
          STA    S45         IN EXCH
          SUB    S67         SET X-M ADDRESS
          SUB    S62
          STA    S32         IN EXCH
          CLA    5,4         SET XA-P ADDRESS
          SUB    S63
          STA    S38         IN EXCH
*SET EXCH LOOP FOR EITHER FXD OR FLTG POINT SPLIT
   S9     CLA    S66         SET 4 NOP*S
          STO    S30
          STO    S31
          STO    S33
          STO    S34
          TRA    S7
*FINISH SETTING FOR EITHER SPLIT OR REFIT
      S7  ZET    S64         FIXED OR FLTG
          TRA    S12         FLTG
          TRA    S13         FIXED
     S13  CLA    S57         SET ADD AND SUB INSTRUCTIONS
          STO    S35
          CLA    S58
          STO    S37
          TRA    S14
     S12  CLA    S53         SET FAD AND FSB INSTRUCTIONS
          STO    S35
          CLA    S54
          STO    S37
    S14   ZET    S65         SPLIT OR REFIT
          TRA    S90         REFIT
          TRA    S91         SPLIT
    S90   CLA    S36+1       REFIT - PATCH UP S60 ADDRESS
          STA    S35         IN LOOP
          CAL    S36         GET INSTR WITH WRONG TAG
          ANA    S99         WIPE OUT TAG
          ORA    S97         PUT IN TAG OF 2
          SLW    S36
          CAL    S38         REPEAT
          ANA    S99
          ORA    S98         PUT IN TAG OF 1
          SLW    S38
          TRA    S15         ON TO REFIT SEQUENCE
   S91    CLA    S53         SPLIT - PATCH UP S59 ADDRESS
          STA    S35
          CAL    S36         GET INSTR WITH WRONG TAG
          ANA    S99         WIPE OUT TAG
          ORA    S98         PUT IN TAG OF ONE
          SLW    S36
          CAL    S38         DITTO
```

```
         ANA     S99
         ORA     S97             PUT IN TAG OF 2
         SLW     S38
         TRA     S16             ON TO SPLIT SEQUENCE
     S99 OCT     -377777077777
     S98 PZE     0,1,0
     S97 PZE     0,2,0
     S15 TSX     S48,4           REFIT SEQUENCE
         TSX     S47,4
         TSX     S44,4
         CLA     S83       XS+1        AVOID
         STA     S61                   REREVERSE
         LXA     S38,4     X+1         IF
         PXA     0,4                   X=XS
         SUB     S61
         TZE     *+2
         TSX     S48,4
         TRA     S49
     S16 TSX     S44,4           SPLIT SEQUENCE
         TSX     S47,4
         TSX     S48,4
         TRA     S49
*EXCHANGE LOOP
     S44 AXT     1,1       FOR         FOR         FOR         FOR
         AXC     1,2       SPLIT       SPLIT       REFIT       REFIT
*                          FIXED       FLTG        FIXED       FLTG
     S45 CLA     **,1    CLA X+1,1  CLA X+1,1  CLA XS+1,1  CLA XS+1,1
    S30  NOP     **        NOP         NOP         LRS 1       FDP S51
    S31  NOP     **        NOP         NOP         RND         XCA
         STO     **          STO S59         OR      STO S60
     S32 CLA     **,2    CLA X-M,2  CLA X-M,2  CLA XA-P,2  CLA XA-P,2
    S33  NOP     **        NOP         NOP         LRS 1       FDP S51
    S34  NOP     **        NOP         NOP         RND         XCA
         STO     **          STO S60         OR      STO S59
    S35  NOP     **        ADD S59    FAD S59    ADD S60    FAD S60
     S36 STO     **,1    STO XS+1,1  STO XS+1,1  STO X-M,2  STO X-M,2
         CLA     S60
     S37 NOP     **        SUB S59    FSB S59    SUB S59    FSB S59
     S38 STO     **,2    STO XA-P,2  STO XA-P,2  STO X+1,1  STO X+1,1
         TXI     *+1,1,1         INCREASE I BY 1
         TXI     *+1,2,-1        DECREASE -I BY 1
     S39 TXL     S45,1,**        **=P
         TRA     1,4
*MIDPOINT MOVE ROUTINE (IF M IS ODD) ASSUMES IR1 = P+1
     S47 NZT     S61
         TRA     1,4       SPLIT             REFIT
         LXA     S63,1           PUT P+1 IN IR1
         TXI     *+1,1,1
     S85 CLA     **,1      CLA X+1,1         CLA XS+1,1
    S86  STO     **,1      STO XS+1,1        STO X+1,1
         TRA     1,4
*REVERSE SYMMETRIC PART ROUTINE
     S48 AXT     1,1
         AXC     1,2
     S81 CLA     **,1            CLA XS+1,1
     S82 LDQ     **,2            LDQ XS-N,2
     S83 STQ     **,1            STQ XS+1,1
     S84 STO     **,2            STO XS-N,2
         TXI     *+1,1,1
         TXI     *+1,2,-1
     S80 TXL     S81,1,**        **=N/2 TRUNCATED
         TRA     1,4
* TREAT SPECIAL CASE M=1
    S200 CLA     S65             IS IT SPLIT OR REFIT
         TZE     S205
         CLA*    4,4             REFIT
         STO*    1,4
         TRA     S49
    S205 CLA*    1,4             SPLIT
         STO*    4,4
*EXIT ROUTINE
     S49 LXD     SPLIT-2,4
         AXT     **,1
         AXT     **,2
         TRA     6,4
```

SPLIT (PAGE 6)

```
*CONSTANTS ETC FOR SPLIT AND REFIT
    S50 FDP      S51
    S51 DEC      2.0
    S52 XCA
    S53 FAD      S59
    S54 FSB      S59
    S55 LRS      1
    S56 RND
    S57 ADD      S59
    S58 SUB      S59
    S59 PZE      **             STORAGE FOR X(I)/2 OR XS(I)
    S60 PZE      **             STORAGE FOR X(M-(I-1))/2 OR XA(P-(I-1))
    S61 PZE      **             **=0 IF M EVEN, =1 IF M ODD
    S62 PZE      **             **=M
    S63 PZE      **             **=P
    S64 PZE      **             **=0 IF FIXED PT, =1 IF FLTG
    S65 PZE      **             **=0 IF SPLIT, **=1 IF REFIT
    S66 NOP
    S67 PZE      1
    S68 PZE      **             **=N
        END
```

```
************************
*    SQRDEV            *
************************
 REFER TO
     SQRDFR
```


```
************************
*    SQRDEV            *
************************
 REFER TO
     SQRDFR
```

```
*      SQRDFR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0110
*      FAP
*SQRDFR
       COUNT    100
       LBL      SQRDFR
       ENTRY    SQRDFR (X,Y,LXY,SSQXMY)
       ENTRY    SQRDEV (X,XBASE,LX,SSQXMB)
*
*                    ----ABSTRACT----
*
* TITLE - SQRDFR WITH SECONDARY ENTRY SQRDEV
*        SUM SQUARE DIF. OF FLTG VECTOR FROM ANOTHER OR FROM A CONSTANT
*
*              SQRDFR SUMS THE SQUARES OF THE DIFFERENCES BETWEEN THE
*              ELEMENTS OF TWO FLOATING VECTORS.
*
*              SQRDEV SUMS THE SQUARES OF THE DIFFERENCES BETWEEN THE
*              ELEMENTS OF A FLOATING VECTOR AND A CONSTANT.
*
* LANGUAGE    - FAP SUBROUTINES  (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 36   REGISTERS
* SPEED       -        7090      709
*              38 + (33.8 OR 37.0)*LX MACHINE CYCLES, LX= VECTOR LENGTH
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*    CALL SQRDFR(X,Y,LXY,SSQXMY)
*    CALL SQRDEV(X,XBASE,LX,SSQXMB)
*
* INPUTS
*
*    X(I)      I=1...LXY  IS  INPUT  TO  SQRDFR
*    Y(I)      I=1...LXY  IS  INPUT  TO  SQRDFR
*    LXY       SHOULD EXCEED 0
*
*    X(I)      I=1...LX   IS  INPUT  TO  SQRDEV
*    XBASE                IS  INPUT  TO  SQRDEV
*    LX        SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO ACTION IF LXY OR LX LSTHN 1
*
*    SSQXMY    IS  SUM (FROM I=1 TO LXY) OF (X(I)-Y(I))*(X(I)-Y(I))
*
*    SSQXMB    IS SUM (FROM I=1 TO LX) OF (X(I)-XBASE)*(X(I)-XBASE)
*
*              EQUIVALENCE(SSQXMY,ANY INPUT),(SSQXMB,ANY INPUT)  IS
*                PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 1., 2., 3.    Y(1...3)= 3.,4.,5.    SDIF2=0.0
*    USAGE   -        CALL SQRDFR(X,Y,3,SDIF1)
*                     CALL SQRDEV(X,3.0,3,SDEV1)
*                     CALL SQRDFR(X,Y,1,X)
*                     CALL SQRDFR(X,Y,0,SDIF2)
*    OUTPUTS - SDIF1 = 12.0    SDEV1= 5.0      X(1)= 4.0
*              SDIF2 = 0.0 (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR      0              XR4
       BCI      1,SQRDFR
* PRINCIPAL ENTRY.  SQRDFR(X,Y,LXY,SSQXMY)
SQRDFR CLA      2,4
       ADD      K1             A(Y)+1
       STA      FSB
       CLA      FSB
```

```
 SETUP STO     SUBTR
       SXD     SQRDFR-2,4
 K1    CLA     1,4
       ADD     K1             A(X)+1
       STA     GET
       CLA*    3,4            LXY
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
       STZ     TEMP1
* LOOP
 GET   CLA     **,4           **=A(X)+1
 SUBTR FSB     **,**          = FSB A(Y)+1,4, OR FSB A(XBASE)
       STO     TEMP2
       XCA
       FMP     TEMP2
       FAD     TEMP1
       STO     TEMP1
       TIX     GET,4,1
* STORE RESULT
       LXD     SQRDFR-2,4
       STO*    4,4
* EXIT
 LEAVE LXD     SQRDFR-2,4
       TRA     5,4
* SECOND ENTRY. SQRDEV (X,XBASE,LX,SSQXMB)
SQRDEV CLA     2,4            A(XBASE)
       STA     FSBXB
       CLA     FSBXB
       TRA     SETUP
* CONSTANTS, TEMPORARIES
 FSB   FSB     **,4           **=A(Y)+1
 FSBXB FSB     **             **=A(XBASE)
 TEMP1 PZE     **,**,**       SUM
 TEMP2 PZE     **,**,**       TEMP FOR DIFFERENCE
       END
```

```
*       SQRMLI (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0127
*       FAP
*SQRMLI
        COUNT   100
        LBL     SQRMLI
        ENTRY   SQRMLI (MLIVEC,ILO,IHI,MLISQR,IANS)
*
*                    ----ABSTRACT----
*
* TITLE - SQRMLI
*        FAST SQUARE ELEMENTS OF A MACHINE LANGUAGE INTEGER VECTOR
*
*              SQRMLI TREATS A SPECIFIED RANGE OF A FORTRAN-TYPE VECTOR
*              AS MACHINE INTEGERS, FORMING A SECOND VECTOR WHOSE
*              ELEMENTS ARE THE MLI SQUARES OF THOSE OF THE FIRST
*              VECTOR, CHECKING FOR OVERFLOW.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 55 REGISTERS
* SPEED     - LENGTH OF RANGE TIMES 20 MACHINE CYCLES (AVG INTEGERS)
* AUTHOR    - S.M. SIMPSON JR, JUNE 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*        AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*       CALL SQRMLI(MLIVEC,ILO,IHI,MLISQR,IANS)
*
* INPUTS
*
*    MLIVEC(I) I=ILO,....,IHI IS THE INPUT VECTOR RANGE.
*
*    ILO       MUST EXCEED 0.
*
*    IHI       MUST EQUAL OR EXCEED ILO.
*
* OUTPUTS
*
*    MLISQR(I) I=1,2,....,(IHI-ILO+1) CONTAINS
*              SQUARE(MLIVEC(ILO,....,IHI)).
*
*    IANS      = 0 MEANS JOB DONE OK.
*              =-1 MEANS ILLEGAL SPECIFICATION OF ILO, IHI.
*              =-2 MEANS THE SQUARE OF ONE OF THE MLIVEC ELEMENTS
*                  EXCEEDED 35 BITS IN LENGTH (PROGRAM DOES NOT FINISH
*                  SQUARING REST OF ELEMENTS WHEN THIS CONDITION OCCURS)
*
* EXAMPLES
*
* 1. INPUTS  - MLIVEC(1...5)=OCT 2,4,6,10,12  ILO=2  IHI=5
*    OUTPUTS - IANS=0, MLISQR(1...4)=OCT 20,44,100,144
*
* 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT MLIVEC(3)=OCT 700000
*    OUTPUTS - IANS =-2  MLISQR(1...2)= OCT 20,210000000000
*              I.E., MLISQR(2) = LEAST SIGNIFICANT 35 BITS OF
*              OCT 700000 SQUARED = 610000000000
*
* 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT IHI=1
*    OUTPUTS - IANS=-1
*
        HTR     0
        BCI     1,SQRMLI
SQRMLI  SXA     EXIT,1
        SXD     SQRMLI-2,4
        CLA     2,4            A(A(ILO))
        STA     GET2
        CLA     3,4            A(A(IHI))
        STA     GET3
        CLA     5,4            A(A(IANS))
        STA     PUT5
* SET UP CONSTANTS ILO, IHI, LVECT AND CHECK THEM.
* SET IANS FOR ILLEGAL INPUT.
```

```
           CLS     K1
           STO     IANS
      GET2 CLA     **            A(ILO)
           ARS     18
           STO     ILO
           TMI     LEAVE
           TZE     LEAVE
      GET3 CLA     **            A(IHI)
           ARS     18
           STO     IHI
           TMI     LEAVE
           TZE     LEAVE
           SUB     ILO
           ADD     K1
           STO     LVECT
           TMI     LEAVE
           TZE     LEAVE
    * SET LOOP UP
           CLA     1,4           A(A(MLIVEC))
           SUB     ILO
           ADD     K2
           STA     LDQ
           STA     MPY
           CLA     4,4           A(A(MLISQR))
           ADD     K1
           STA     STQ
    * SET IANS FOR POSSIBLE OVERFLOW INDICATION DURING LOOP.
           CLS     K2
           STO     IANS
    * LOOP
           LXA     LVECT,1
       LDQ LDQ     **,1          A(MLIVEC)-ILO+2
       MPY MPY     **,1          A(MLIVEC)-ILO+2
       STQ STQ     **,1          A(MLISQR)+1
           TNZ     LEAVE
           TIX     LDQ,1,1
    * ALL OK IF FALLS THRU LOOP.
           STZ     IANS
    * STORE IANS AND LEAVE.
     LEAVE CLA     IANS
           ALS     18
      PUT5 STO     **            A(IANS)
      EXIT AXT     **,1
           TRA     6,4
    * CONSTANTS
        K1 PZE     1
        K2 PZE     2
    * VARIABLES
       ILO PZE     **
       IHI PZE     **
      IANS PZE     **            0 OR -1 OR -2
     LVECT PZE     **            IHI-ILO+1
           END
```

```
*      SQROOT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0082
*      FAP
*SQROOT
       COUNT   100
       LBL     SQROOT
       ENTRY   SQROOT (X,LX,XSQRTD)
*
*                    ----ABSTRACT----
*
* TITLE - SQROOT
*         SQUARE ROOT OF A FLOATING VECTOR
*
*              SQROOT FORMS A VECTOR WITH ELEMENTS EQUAL TO THE SQUARE
*              ROOTS OF THE ELEMENTS OF ANOTHER (FLOATING) VECTOR.
*              OUTPUT MAY REPLACE INPUT,
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 24   REGISTERS
* SPEED       - ABOUT  31 + 220*LX  MACHINE CYCLES, LX = VECTOR LENGTH
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - SQRT  (FUNCTION)
*
* FORTRAN USAGE
*     CALL SQROOT(X,LX,XSQRTD)
*
* INPUTS
*
*    X(I)      I=1...LX  IS A NON-NEGATIVE VECTOR
*
*    LX        SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF LX LSTHN 1
*
*    XSQRTD(I) I=1...LX  IS  XSQRTD(I) = SQRTF(X(I)).  NEGATIVE VALUES
*                 OF X(I) ARE TREATED AS THOUGH THEY WERE POSITIVE.
*
*              EQUIVALENCE (XSQRTD,X)  IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4) = 100., 200., -300., 400.   XSQRT2=0.0
*    USAGE   -        CALL SQROOT(X,4,XSQRT1)
*                     CALL SQROOT(X,0,XSQRT2)
*                     CALL SQROOT(X,4,X)
*    OUTPUTS - XSQRT1(1...4)= 10.0, 14.1, 17.3, 20.0
*              XSQRT2 = 0.0  (NO OUTPUT CASE)  X(1...4)= XSQRT1(1...4)
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS SQRT FUNCTION
       HTR     0             XR1
       HTR     0             XR4
       BCI     1,SQROOT
* ONLY ENTRY.  SQROOT(X,LX,XSQRTD)
SQROOT SXD     SQROOT-2,4
       SXD     SQROOT-3,1
 K1    CLA     1,4
       ADD     K1            A(X)+1
       STA     GET
       CLA     3,4
       ADD     K1            A(XSQRTD)+1
       STA     STORE
       CLA*    2,4           LX
       TMI     LEAVE
       PDX     0,1
       TXL     LEAVE,1,0
* LOOP
 GET   CLA     **,1          **=A(X)+1
       SSP
```

```
          TSX     $SQRT,4
    STORE STO     **,1            **=A(XSQRTD)+1
          TIX     GET,1,1
   * EXIT
    LEAVE LXD     SQROOT-2,4
          LXD     SQROOT-3,1
          TRA     4,4
          END
```

```
*       SQRSUM (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0106
*       FAP
*SQRSUM
        COUNT    150
        LBL      SQRSUM
        ENTRY    SQRSUM ( X, LX,SUMSQX)
        ENTRY    XSQSUM (IX,LIX,IXMSQX)
*
*                     ----ABSTRACT----
*
* TITLE - SQRSUM  WITH SECONDARY ENTRY XSQSUM
*            SUM THE SQUARED ELEMENTS OF A FLTG OR FXD VECTOR
*
*                SQRSUM ADDS UP THE SQUARED ELEMENTS OF A FLTG PT VECTOR
*                XSQSUM ADDS UP THE SQUARED ELEMENTS OF A FXD PT VECTOR
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 36 REGISTERS
* SPEED      -             7090       709
*                SQRSUM  42 + (19.6  OR 23.8)*LX  MACHINE CYCLES,
*                XSQSUM  39 + (23.4  OR 26.6)*LX      LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*          AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*      CALL SQRSUM( X, LX,SUMSQX)
*      CALL XSQSUM(IX,LIX,IXMSQX)
*
* INPUTS
*
*    X(I)        I=1...LX  IS A FLTG VECTOR
*
*    LX          SHOULD EXCEED ZERO
*
*    IX(I)       I=1...LIX IS A FXD VECTOR
*
*    LIX         SHOULD EXCEED ZERO
*
* OUTPUTS        STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1 .
*
*    SUMSQX      IS SUM OF  X(I)*X(I)
*
*    IXMSQX      IS SUM OF IX(I)*IX(I) .  OVERFLOW MAY OCCUR AND
*                   IS NOT TESTED FOR BY XSQSUM.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...4)=1.,2.,3.,4.  IX(1...4)=1,2,3,4  U=0.0
*    USAGE   -         CALL SQRSUM(X,4,SUMSQX)
*                      CALL XSQSUM(IX,4,ISMSQX)
*                      CALL SQRSUM(X,1,Y)
*                      CALL SQRSUM(X,0,U)
*    OUTPUTS - SUMSQX=30.  ISMSQX=30  Y=1.  U=0.0 (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR      0                  XR4
        BCI      1,SQRSUM
* PRINCIPAL ENTRY.  SQRSUM(X,LX,SUMSQX)
SQRSUM SXD       TEMP,4
        TRA      SETUP
* SECOND ENTRY.  XSQSUM(IX,LIX,ISMSQX)
XSQSUM STZ       TEMP
 SETUP SXD       SQRSUM-2,4
 K1     CLA      1,4
        ADD      K1                 A(X)+1
        STA      LGET
        STA      LMUL
        STA      XGET
```

****
*   SQRSUM   *
****
(PAGE 2)


```
       STA     XMUL
* CHECK FOR ILLEGAL LX
       CLA*    2,4              LX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* BRANCH TO PROPER LOOP
       CLA     TEMP
       TZE     XGET
       STZ     TEMP
* FLOATING LOOP
 LGET  LDQ     **,4        **=A(X)+1
 LMUL  FMP     **,4        **=A(X)+1
       FAD     TEMP
       STO     TEMP
       TIX     LGET,4,1
       TRA     STORE
* FIXED LOOP
 XGET  LDQ     **,4        **=A(X)+1
 XMUL  MPY     **,4        **=A(X)+1
       ADD     TEMP
       STO     TEMP
       TIX     XGET,4,1
       ALS     17
* STORE SUM OF SQUARES
 STORE LXD     SQRSUM-2,4
       STO*    3,4
* EXIT
 LEAVE LXD     SQRSUM-2,4
       TRA     4,4
 TEMP  PZE     **               **=0 IF FXD, LATER SUMMATION
       END
```

```
*      SQUARE (SUBROUTINE)                 9/29/64  LAST CARD IN DECK IS NO. 0110
*      FAP
*SQUARE
       COUNT   150
       LBL     SQUARE
       ENTRY   SQUARE ( X, LX, XSQRD)
       ENTRY   XSQUAR (IX,LIX,IXSQRD)
*
*                    ----ABSTRACT----
*
* TITLE - SQUARE WITH SECONDARY ENTRY XSQUAR
*         SQUARE ELEMENTS OF FXD OR FLTG VECTOR
*
*               SQUARE FORMS A VECTOR EQUAL TO THE SQUARE OF A GIVEN
*               FLTG PT VECTOR.  OUTPUT MAY REPLACE INPUT.
*
*               XSQUAR FORMS A VECTOR EQUAL TO THE SQUARE OF A GIVEN
*               FXD PT VECTOR.  OUTPUT MAY REPLACE INPUT.  NO TEST
*               MADE FOR OVERFLOW.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 32 REGISTERS
* SPEED      -                7090      709
*               SQUARE    37 + (19    OR 22.2)*LX    MACHINE CYCLES,
*               XSQUAR    39 + (20.6 OR 24.8)*LX      XL = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL SQUARE( X, LX, XSQRD)
*     CALL XSQUAR(IX,LIX,IXSQRD)
*
* INPUTS
*
*    X(I)       I=1...LX  IS A FLTG VECTOR
*
*    LX         SHOULD EXCEED 0
*
*    IX(I)      I=1...LIX IS A FIXED VECTOR
*
*    LIX        SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1.
*
*    XSQRD(I)   I=1...LX  HAS VALUES XSQRD(I)=X(I)*X(I)
*               EQUIVALENCE (X,XSQRD) IS PERMITTED.
*
*    IXSQRD(I)  I=1...LIX  HAS VALUES IXSQRD(I)=IX(I)*IX(I)
*               EQUIVALENCE (IX,IXSQRD) IS PERMITTED.  OVERFLOW OCCURS
*               IF ANY IX HAS MAGNITUDE GRTHN SQRT(2**17-1).
*
* EXAMPLES
*
* 1. INPUTS   - X(1...5)=1.,2.,...,5.  IX(1...5)=-1,-2,...,-5
*               YY=0.
*    USAGE    -       CALL SQUARE(X,5,Y)
*                     CALL XSQUAR(IX,5,IY)
*                     CALL SQUARE(X,1,Z)
*                     CALL SQUARE(X,0,YY)
*                     CALL XSQUAR(IX,5,IX)
*
*    OUTPUTS - Y(1...5)=1,4.,9.,16.,25.  IY(1...5)=1,4,9,16,25
*              Z=1.  YY=0.  (NO OUTPUT CASE)  IX(1...5)=1,4,9,16,25
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR     0                    XR4
       BCI     1,SQUARE
* PRINCIPAL ENTRY.  SQUARE(X,LX,XSQRD)
```

```
     SQUARE CLA     FMP
            LDQ     NOP
      SETUP STO     SQR
            STQ     VARY
            SXD     SQUARE-2,4
      K1    CLA     1,4
            ADD     K1              A(X)+1
            STA     GET
            STA     SQR
            CLA     3,4
            ADD     K1              A(XSQRD)+1
            STA     STORE
     * CHECK LX
            CLA*    2,4             LX
            TMI     LEAVE
            PDX     0,4
            TXL     LEAVE,4,0
     * SQUARING LOOP
      GET   LDQ     **,4          **=A(X)+1
      SQR   NOP                   = FMP  **,4  OR  MPY  **,4    **=A(X)+1
      VARY  NOP                         = NOP OR ALS  17
      STORE STO     **,4          **=A(XSQRD)+1
            TIX     GET,4,1
     * EXIT
      LEAVE LXD     SQUARE-2,4
            TRA     4,4
     * SECOND ENTRY.  XSQUAR(IX,LIX,IXSQRD)
     XSQUAR CLA     MPY
            LDQ     ALS
            TRA     SETUP
     * CONSTANTS
      FMP   FMP     **,4
      NOP   NOP
      MPY   MPY     **,4
      ALS   ALS     17
            END
```

```
*     SRCH1  (SUBROUTINE)                9/8/64   LAST CARD IN DECK IS NO. 0092
*     LABEL
CSRCH1
      SUBROUTINE SRCH1 (JOB,LV,V,VN,INDEX)
C
C
C                        ----ABSTRACT----
C
C TITLE - SRCH1
C          SEARCH VECTOR FOR NUMBER, STARTING FROM FIRST OR LAST TERM
C
C                SRCH1 SEARCHES A VECTOR OF FIXED OR FLOATING NUMBERS
C                FOR A PARTICULAR NUMBER.  THE INDEX OF THE FIRST NUMBER
C                FOUND IS RETURNED.  IF NO SUCH NUMBER IS FOUND THE INDEX
C                IS SET = 0.  THE DIRECTION OF SEARCHING (FROM BEGINNING
C                TO END, OR FROM END TO BEGINNING) IS CONTROLLED BY A
C                PARAMETER.  PLUS ZERO IS CONSIDERED NOT EQUAL TO MINUS
C                ZERO DURING THE SEARCH.
C
C LANGUAGE    - FORTRAN II SUBROUTINE
C EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE     - 93 REGISTERS
C SPEED       - TAKES ABOUT  100 + 38*N  MACHINE CYCLES ON THE 7090,
C                  WHERE  N  IS THE NO. OF VECTOR ELEMENTS EXAMINED
C                  (EXAMINATION STOPS IF VALUE FOUND).
C AUTHOR      - R.A. WIGGINS,  AUGUST, 1963
C
C                        ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - XACTEQ
C       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
C
C FORTRAN USAGE
C     CALL SRCH1 (JOB,LV,V,VN,INDEX)
C
C INPUTS
C
C    JOB        =1 FOR FORWARD SEARCHING (BEGINNING WITH LOWEST INDEX)
C               =2 FOR REVERSE SEARCHING (BEGINNING WITH HIGHEST INDEX)
C
C    LV         IS LENGTH OF INPUT VECTOR V.
C               IF LSTHN 1 SRCH1 ONE RETURNS WITH INDEX=0
C
C    V(I)       I=1...LV IS THE VECTOR TO BE SEARCHED.
C               NEED NOT BE FLOATING POINT MODE.
C
C    VN         IS THE VALUE TO BE SEARCHED FOR.
C               SHOULD BE SAME MODE AS V.
C
C OUTPUTS
C
C    INDEX      THE INDEX OF THE FIRST VALUE FOUND FOR WHICH
C                  V(INDEX) = VN .  WILL = 0  IF NONE FOUND.
C
C EXAMPLES
C
C 1. INPUTS   - JOB = 1 IVN = -1   VN = 2.  LV = 20
C               V(1...20) = -2.,-1.,0.,1.,2.,-3.,...,0.,1.,2.,-3.,-2.,-1.
C               IV(1...20)= -2,-1,0,1,2,-3,-2,-1,0,...,0,1,2,-3,-2,-1
C    USAGE    -        CALL SRCH1 (JOB,LV,V,VN,INDEX1)
C                      CALL SRCH1 (JOB,LV,IV,IVN,INDEX2)
C    OUTPUTS - INDEX1 = 5  INDEX2 = 2
C
C 2. INPUTS   - SAME AS EXAMPLE 1. EXCEPT JOB = 2
C    USAGE    - SAME AS EXAMPLE 1.
C    OUTPUTS - INDEX1 = 17  INDEX2 = 20
C
C 3. INPUTS   - SAME AS EXAMPLE 1. EXCEPT JOB = 2  VN = -0.  IVN = -0
C               V(13) = -0.   IV(9) = -0
C    USAGE    - SAME AS EXAMPLE 1.
C    OUTPUTS - INDEX1 = 13   INDEX2 = 9
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION  V(2)
```

****
*   SRCH1   *
****
(PAGE 2)

```
      INDEX=0
      IF (LV)  10,10,20
10    RETURN
20    CONTINUE
      GO TO (30,40),JOB
30    J=1
      IJ=1
      GO TO 50
40    J=LV
      IJ=-1
50    CONTINUE
      DO 70  I=1,LV
      IF (XACTEQF(V(J),VN)) 70,60,70
70    J=J+IJ
      GO TO 10
60    INDEX=J
      GO TO 10
      END
```

PROGRAM LISTINGS

```
************************
*    STEPC             *
************************
 REFER TO
     DELTA

************************
*    STEPL             *
************************
 REFER TO
     DELTA

************************
*    STEPR             *
************************
 REFER TO
     DELTA

************************
*    (STH)             *
************************
 REFER TO
     CNLINE

************************
*    (STHD)            *
************************
 REFER TO
     CNLINE

************************
*    (STHM)            *
************************
 REFER TC
     CNLINE

************************
*    STORE             *
************************
 REFER TO
     LOCATE
```


```
************************
*    STEPC             *
************************
 REFER TO
     DELTA

************************
*    STEPL             *
************************
 REFER TO
     DELTA

************************
*    STEPR             *
************************
 REFER TO
     DELTA

************************
*    (STH)             *
************************
 REFER TO
     ONLINE

************************
*    (STHD)            *
************************
 REFER TO
     ONLINE

************************
*    (STHM)            *
************************
 REFER TO
     ONLINE

************************
*    STORE             *
************************
 REFER TO
     LOCATE
```

```
*      STZ (SUBROUTINE)                  9/29/64   LAST CARD IN DECK IS NO. 0059
*      FAP
*STZ
       COUNT   50
       LBL     STZ
       ENTRY   STZ (LX,X)
*
*                    ----ABSTRACT----
*
* TITLE - STZ
*       FAST SET VECTOR TO ZERO
*
*             STZ SETS A VECTOR TO ZERO.
*
* LANGUAGE   - FAP SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT  - IBM 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 14 REGISTERS
* SPEED      - ABOUT  4*N + 18  MACHINE CYCLES WHERE N IS THE LENGTH
*                OF THE VECTOR.
* AUTHOR     - J.F. CLAERBOUT
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - NONE
*
* FORTRAN USAGE
*     CALL STZ(LX,X)
*
* INPUTS
*
*    X(I)      I=1...LX IS THE VECTOR TO BE SET TO ZERO.
*              NEED NOT HAVE A FLOATING POINT NAME.
*
*    LX        MUST BE GRTHN=1.
*              IS FORTRAN II INTEGER.
*
* OUTPUTS
*
*    X(I)      I=1...LX IS SET TO ZERO.
*
* EXAMPLES
*
* 1. INPUTS  - LX = 5   X(1...5) = 1.,1.,1.,1.,1.
*    OUTPUTS - X(1...5) = 0.,0.,0.,0.,0.
*
       HTR     0
       BCI     1,STZ
 STZ   SXD     *-2,4
       CLA     2,4
       ADD     =1
       STA     Z
       CLA*    1,4
       TZE     3,4
       PDX     ,4
     Z STZ     **,4
       TIX     *-1,4,1
 SV    LXD     STZ-2,4
       TRA     3,4
       END
```

```
*       STZS   (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0096
*       FAP
*STZS
        COUNT   100
        LBL     STZS
        ENTRY   STZS (LX1,X1,LX2,X2,...,LXN,XN)
*
*                     ----ABSTRACT----
*
* TITLE - STZS
*         SET A LIST OF VECTORS TO ZERO
*
*               STZS IS A VARIABLE LENGTH CALLING SEQUENCE SUBROUTINE
*               WHOSE ARGUMENTS ARE CONSIDERED IN PAIRS.  THE SECOND
*               ARGUMENT OF EACH PAIR IS CONSIDERED TO BE A VECTOR
*               WHOSE LENGTH IS GIVEN BY THE FIRST ARGUMENT.  ON OUTPUT
*               ALL SUCH VECTORS WILL BE CLEARED EXCEPT THAT NO ACTION
*               IS TAKEN ON VECTORS OF NEGATIVE OR ZERO LENGTH.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 24 REGISTERS
* SPEED      - 9 + 25*N + 4*L  MACHINE CYCLES,
*                WHERE  N = THE NUMBER OF VECTORS TO BE CLEARED
*                       L = THE SUM OF THEIR LENGTHS
* AUTHOR     - S.M. SIMPSON,  SEPTEMBER 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL STZS  (LX1,X1, LX2,X2, ..., LXN,XN)
*                  WHERE THE NO. OF PAIRS, N, SHOULD EXCEED 0
*
* INPUTS
*
*    LX1        IS LENGTH OF FIRST VECTOR.  SHOULD EXCEED 0
*    LX2        IS LENGTH OF SECOND VECTOR. SHOULD EXCEED 0
*          ETC
*    LXN        IS LENGTH OF LAST VECTOR.   SHOULD EXCEED 0
*
* OUTPUTS
*
*    X1(I)      I=1...LX1  IS X1(I) = 0,   PROVIDED LX1 GRTHN= 1
*    X2(I)      I=1...LX2  IS X2(I) = 0,   PROVIDED LX2 GRTHN= 1
*          ETC
*    XN(I)      I=1...LXN  IS XN(I) = 0,   PROVIDED LXN GRTHN= 1
*
*               IF ANY LX IS 0 OR NEGATIVE, THE CORRESPONDING VECTOR
*               IS NOT DISTURBED.  THE MODES OF THE VECTORS ARE
*               ARBITRARY.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...10) = 999.    Y = Z = W = U = 999.
*              IX(1...3) = 999    IY(1...4) = 999
*    USAGE   -       CALL STZS(10,X,  3,IX,  1,Y,  0,Z,  -2,W,  4,IY)
*                    CALL STZS(1,U)
*    OUTPUTS - X(1...10) = 0.    Y = 0.    IX(1...3) = 0
*              IY(1...4) = 0    Z = W = 999.    U = 0.
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0              XR1
        HTR     0              XR4
        BCI     1,STZS
* ONLY ENTRY.  STZS(LX1,X1,LX2,X2,...,LXN,XN)
 STZS   SXD     STZS-2,4
        SXD     STZS-3,1
* EXAMINE FOR NEXT L
 CAL    CAL     1,4
```

```
        ANA    AMASK
        LAS    TSXZ
        TRA    LEAVE
        TRA    CLEAR
* EXIT
 LEAVE  LXD    STZS-3,1
        TRA    1,4
* CLEARING LOOP
 CLEAR  CLA    2,4
        ADD    K1             A(X)+1
        STA    STZ
 K1     CLA*   1,4            L
        TMI    BACK
        PDX    0,1
        TXL    BACK,1,0
 STZ    STZ    **,1           **=A(X)+1
        TIX    STZ,1,1
 BACK   TXI    CAL,4,-2
* CONSTANTS
 AMASK  OCT    777777700000
 TSXZ   TSX    0,0
        END
```

```
***********************
*    SUBK             *
***********************
REFER TO
     ADDK
```


```
***********************
*    SUBK             *
***********************
REFER TO
     ADDK
```


```
***********************
*    SUBKS            *
***********************
REFER TO
     ADDK
```


```
***********************
*    SUBKS            *
***********************
REFER TO
     ADDK
```

```
*       SUM      (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0091
*       FAP
*SUM
        COUNT    150
        LBL      SUM
        ENTRY    SUM  ( X, LX, SUMX)
        ENTRY    XSUM (IX,LIX,ISUMIX)
*
*                    ----ABSTRACT----
*
* TITLE - SUM WITH SECONDARY ENTRY XSUM
*         SUM ELEMENTS OF FLTG OR FIXED VECTOR
*
*                SUM ADDS UP ELEMENTS OF A FLTG PT VECTOR.
*                XSUM ADDS UP ELEMENTS OF A FXD PT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE, (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 23 REGISTERS
* SPEED      - SUM    32 + 8.4*LX   MACHINE CYCLES,
*              XSUM   30 + 4*LX       LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL SUM ( X, LX,  SUMX)
*     CALL XSUM(IX,LIX,ISUMIX)
*
* INPUTS
*
*    X(I)        I=1...LX  IS A FLTG VECTOR
*
*    LX          SHOULD EXCEED ZERO
*
*    IX(I)       I=1...LIX  IS A FXD VECTOR
*
*    LIX         SHOULD EXCEED ZERO
*
* OUTPUTS        STRAIGHT RETURN WITH NO OUTPUT IF LX OR LIX LSTHN 1 .
*
*    SUMX        IS SUM OF X(1...LX)
*
*    ISUMIX      IS SUM OF IX(1...LIX) .  OVERFLOW MAY OCCUR AND IS
*                   NOT CHECKED FOR BY XSUM.
*
* EXAMPLES
*
* 1. INPUTS   - X(1...4)=1.,2.,3.,4.  IX(1...4)=1,2,3,4
*               U=0.0
*    USAGE    -       CALL SUM (X,4,SUMX)
*                     CALL XSUM(IX,4,ISUMIX)
*                     CALL SUM  (X,1,Y)
*                     CALL SUM  (X,0,U)
*    OUTPUTS - SUMX = 10.  ISUMIX = 10  Y = 1.  U = 0. (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
        HTR      0                 XR4
        BCI      1,SUM
* PRINCIPAL ENTRY.  SUM(X,LX,SUMX)
 SUM    CLA      FAD
        TRA      SETUP
* SECOND ENTRY.  XSUM(IX,LIX,ISUMIX)
 XSUM   CLA      ADD
 SETUP  STO      ADD1
        SXD      SUM-2,4
 K1     CLA      1,4
        ADD      K1                A(X)+1
        STA      ADD1
* CHECK FOR ILLEGAL LX
```

```
        CLA*    2,4             LX
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
 * FORM AND STORE THE SUM
        PXD     0,0
 ADD1   NOP                 =  FAD  **,4  OR  ADD  **,4    **=A(X)+1
        TIX     ADD1,4,1
        LXD     SUM-2,4
        STO*    3,4
 * EXIT
 LEAVE  LXD     SUM-2,4
        TRA     4,4
 * CONSTANTS
 FAD    FAD     **,4
 ADD    ADD     **,4
        END
```

```
************************
*    SUMDEV            *
************************
 REFER TO
     SUMDFR
```


```
************************
*    SUMDEV            *
************************
 REFER TO
     SUMDFR
```

```
*      SUMDFR (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0155
*      FAP
*SUMDFR
       COUNT    150
       LBL      SUMDFR
       ENTRY    SUMDFR ( X, Y,LXY,SUMXMY)
       ENTRY    XSMDFR (IX,IY,LXY,ISMXMY)
       ENTRY    SUMDEV ( X, XBASE, LX,SUMXMB)
       ENTRY    XSMDEV (IX,IXBASE,LIX,ISMXMB)
*
*                    ----ABSTRACT----
*
* TITLE - SUMDFR WITH SECONDARY ENTRIES XSMDFR, SUMDEV AND XSMDEV
*         SUM DIFFERENCE OF VECTOR FROM ANOTHER OR FROM A CONSTANT
*
*               SUMDFR SUMS THE DIFFERENCES OF TWO FLOATING VECTORS.
*               XSMDFR SUMS THE DIFFERENCES OF TWO FIXED VECTORS.
*               SUMDEV SUMS THE DEVIATIONS OF A FLOATING VECTOR FROM
*                  A CONSTANT.
*               XSMDEV SUMS THE DEVIATIONS OF A FIXED VECTOR FROM
*                  A CONSTANT.
*
*               FOR THE FIXED ENTRIES THE BINARY POINT IS ARBITRARY.
*
* LANGUAGE    - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 44   REGISTERS
* SPEED       - SUMDFR   47 + 14.8*LX   MACHINE CYCLES,
*               XSMDFR   49 +  6.0*LX      LX = VECTOR LENGTH
*               SUMDEV   45 + 14.8*LX
*               XSMDEV   43 +  6.0*LX
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL SUMDFR( X, Y,LXY,SUMXMY)
*     CALL XSMDFR(IX,IY,LXY,ISMXMY)
*     CALL SUMDEV( X, XBASE, LX,SUMXMB)
*     CALL XSMDEV(IX,IXBASE,LIX,ISMXMB)
*
* INPUTS
*
*    X(I)       I=1...LXY  IS A FLOATING INPUT TO SUMDFR
*    Y(I)       I=1...LXY  IS A FLOATING INPUT TO SUMDFR
*    LXY        SHOULD EXCEED 0  (FORTRAN-II INTEGER)
*    IX(I)      I=1...LXY  IS A FIXED INPUT TO XSMDFR
*    IY(I)      I=1...LXY  IS A FIXED INPUT TO XSMDFR WITH THE SAME
*                  BINARY POINT AS IX(I)
*
*    X(I)       I=1...LX   IS A FLOATING INPUT TO SUMDEV
*    XBASE      IS A FLOATING CONSTANT
*    LX         SHOULD EXCEED 0  (FORTRAN-II INTEGER)
*    IX(I)      I=1...LIX   IS A FIXED INPUT TO XSMDEV
*    IXBASE     IS A FIXED CONSTANT WITH THE SAME BINARY POINT AS IX(I)
*    LIX        SHOULD EXCEED ZERO
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUTS IF LXY OR LX LSTHN 1
*
*    SUMXMY     IS  SUM(FROM I=1 TO LXY) OF  (X(I) - Y(I))
*
*    ISMXMY     IS  SUM(FROM I=1 TO LXY) OF (IX(I) - IY(I))
*
*    SUMXMB     IS  SUM(FROM I=1 TO LX)  OF  (X(I) - XBASE)
*
*    ISMXMB     IS  SUM(FROM I=1 TO LIX) OF (IX(I) - IXBASE)
*
*               BINARY POINT OF FIXED OUTPUTS IS SAME AS THAT OF INPUTS.
*               DANGER OF FIXED POINT OVERFLOW IS NOT TESTED FOR.
*
*               EQUIVALENCE(SUMXMY,ANY INPUT),(ISMXMY,ANY INPUT),
```

```
*                  (SUMXMB,ANY INPUT),(ISMXMB, ANY INPUT) IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 1., 2., 3.   Y(1...3) = 2.,4.,6.  XBASE = 3.
*              IX(1...3) = 1, 2, 3     IY(1...3) = 2, 4, 6   IXBASE = 3
*    USAGE   -       CALL SUMDFR( X, Y, 3, DIF1)
*                    CALL XSMDFR(IX,IY, 3,IDIF1)
*                    CALL SUMDEV( X, XBASE, 3, DEV1)
*                    CALL XSMDEV(IX,IXBASE, 3,IDEV1)
*    OUTPUTS - DIF1 = -6.0    IDIF1 = -6   DEV1 = -3.0   IDEV1 = -3
*
* 2. INPUTS  - IX(1...2) = OCT 000000000001, 000000000002
*              IY(1...2) = OCT 000000000002, 000000000004
*              IXBASE    = OCT 000000000003        IDIF4 = 0
*    USAGE   -       CALL XSMDFR(IX,IY,2,IDIF2)
*                    CALL XSMDEV(IX,IXBASE,2,IDEV2)
*                    CALL XSMDFR(IX,IY,1,IDIF3)
*                    CALL XSMDFR(IX,IY,-1,IDIF4)
*                    CALL XSMDFR(IX,IY,2,IY)
*    OUTPUTS - IDIF2 = OCT 400000000003     IDEV2 = OCT 400000000003
*              IDIF3 = OCT 400000000001     IDIF4 = 0 (NO OUTPUT CASE)
*              IY(1) = OCT 400000000003
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
       BCI     1,SUMDFR
* PRINCIPAL ENTRY. SUMDFR(X,Y,LXY,SUMXMY)
SUMDFR LDQ     FAD
       CLA     FSB
SETUP1 STO     SUB1
       CLA     2,4
       ADD     K1             A(Y)+1
       STA     SUB1
SETUP2 STQ     ADD1
       SXD     SUMDFR-2,4
       CLA*    2,4            XBASE OR Y(DUMMY)
       STO     TEMP
 K1    CLA     1,4
       ADD     K1             A(X)+1
       STA     ADD1
       CLA*    3,4            LXY OR LX
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
       PXD     0,0            CLEAR AC
* LOOP
 ADD1  NOP                    FAD **,4  ADD **,4    **=A(X)+1
 SUB1  NOP                    FSB **,4  SUB **,4  FSB TEMP, SUB TEMP,
*                                                 **=A(Y)+1
       TIX     ADD1,4,1
* STORE RESULT
       LXD     SUMDFR-2,4
       STO*    4,4            SUMXMY ETC
* EXIT
 LEAVE LXD     SUMDFR-2,4
       TRA     5,4
* SECOND ENTRY.  XSMDFR(IX,IY,LXY,ISMXMY)
XSMDFR LDQ     ADD
       CLA     SUB
       TRA     SETUP1
* THIRD ENTRY.  SUMDEV(X,XBASE,LX,SUMXMB)
SUMDEV LDQ     FAD
       CLA     FSBT
       TRA     SETUP3
* FOURTH ENTRY.  XSMDEV(IX,IXBASE,LX,ISMXMB)
XSMDEV LDQ     ADD
       CLA     SUBT
SETUP3 STO     SUB1
       TRA     SETUP2
* CONSTANTS, TEMPORARIES
 FAD   FAD     **,4
```

****
*   SUMDFR   *
****
(PAGE 3)

```
      FSB   FSB    **,4
      ADD   ADD    **,4
      SUB   SUB    **,4
      FSBT  FSB    TEMP
      SUBT  SUB    TEMP
      TEMP  PZE    **,**,**       =XBASE  OR IXBASE
            END
```

```
*      SWITCH (FUNCTION)                   9/4/64   LAST CARD IN DECK IS NO. 0083
*      FAP
*SWITCH
       COUNT   100
       LBL     SWITCH
       ENTRY   SWITCH F(ISENSE)
*
*
*                    ----ABSTRACT----
*
* TITLE - SWITCH
*       TEST THE CONDITION OF ANY SENSE SWITCH
*
*              SWITCH IS A FUNCTION WHICH TESTS THE STATUS OF THE
*              SENSE SWITCH WHOSE NUMBER IS THE ARGUMENT OF THE FUNCTION
*
* LANGUAGE   - FAP FUNCTION (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS CONSOLE)
* STORAGE    - 15 REGISTERS
* SPEED      - ABOUT 22 MACHINE CYCLES
* AUTHOR     - S. M. SIMPSON, MARCH 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     ZIFOFF = SWITCHF(ISENSE)
*
*
* INPUTS
*
*    ISENSE    IS ANY INTEGER
*
*
* OUTPUTS
*
*    ZIFOFF    = 0.0  IF ISENSE IS LSTHN= 0 OR GRTHN= 7
*              = 0.0  IF ISENSE = 1,2,3,4,5, OR 6 AND THE
*                     CORRESPONDING SENSE SWITCH IS OFF.
*              = 1.0  IF ISENSE = 1,2,3,4,5, OR 6 AND THE
*                     CORRESPONDING SENSE SWITCH IS DEPRESSED (ON).
*
*
* EXAMPLES
*
* 1. THIS EXAMPLE ASSUMES ALL THE SENSE SWITCHES ARE OFF
*    INPUTS  - ZFOFFV(1...10) = -99.,-99.,...,-99.
*    USAGE   -        DO 10  I=1,10
*                     ISENSE = I-2
*              10     ZFOFFV(I) = SWITCHF(ISENSE)
*    OUTPUTS - ZFOFFV(1...10) = 0.0,0.0,...,0.0
*
* 2. THIS EXAMPLE ASSUMES ALL THE SENSE SWITCHES ARE ON (DEPRESSED)
*    INPUTS  - SAME AS EXAMPLE 1.
*    USAGE   - SAME AS EXAMPLE 1.
*    OUTPUTS - ZFOFFV(1...10) = 0.0,0.0,1.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       BCI     1,SWITCH
*
* ONLY ENTRY.  SWITCH F (ISENSE)
*
SWITCH TMI     PXD
       TZE     PXD
       ARS     18
       SUB     K7
       TPL     PXD
       ADD     K119            GIVES 112 + ISENSE
```

```
************************
*    SWITCH            *
************************
(PAGE  2)
```


```
             STA    PSE
             CLA    K1L
       PSE   PSE    **     ** = 113,114,...    (161,162,... OCTAL)
       PXD   PXD    0,0
             TRA    1,4
       K7    PZE    7
       K119  PZE    119           119 = 112+7
       K1L   DEC    1.0
             END
```

```
*     TAMVL (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0188
*     FAP
*TAMVL
      COUNT   200
      LBL     TAMVL
      ENTRY   TAMVL (X, LX, LAVG, AVGL)
      ENTRY   TAMVR (X, LX, LAVG, AVGR)
*
*
*                   ----ABSTRACT----
*
* TITLE - TAMVL WITH SECONDARY ENTRY TAMVR
*        TRIANGULAR AVERAGING, MOVING LEFT OR RIGHT END
*
*             TAMVL  COMPUTES
*
*                             1     LX
*                AVGL(I)  =  ------  SUM  X(J)    FOR I=I...LAVG
*                            LX-I+1  J=I
*
*             GIVEN  X(1...LX),  LX  AND  LAVG.
*
*             TAMVR  HAS THE SAME INPUTS BUT COMPUTES
*
*                             1    LX-I+1
*                AVGR(I)  =  ------  SUM   X(J)   FOR I=1...LAVG
*                            LX-I+1  J=1
*
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 63 REGISTERS
* SPEED      - EITHER ENTRY TAKES ABOUT
*                80 + 8.4*LX + 41.8*LAVG  MACHINE CYCLES ON THE 7090
* AUTHOR     - S.M. SIMPSON,   JULY 1964
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL TAMVL(X, LX, LAVG, AVGL)
*     CALL TAMVR(X, LX, LAVG, AVGR)
*
*
* INPUTS TO BOTH TAMVL AND TAMVR
*
*    X(I)      I=1...LX  IS A FLOATING VECTOR.
*
*    LX        MUST EXCEED ZERO.
*
*    LAVG      IS DESIRED NUMBER OF OUTPUT AVERAGES.
*              MUST EXCEED ZERO AND BE LSTHN=  LX.
*
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF  LX  OR  LAVG  ILLEGAL
*
*    AVGL(I)   I=1...LAVG  IS OUTPUT FROM  TAMVL  AS DEFINED IN ABSTRACT
*
*    AVGR(I)   I=1...LAVG  IS OUTPUT FROM  TAMVR  AS DEFINED IN ABSTRACT
*
*
* EXAMPLES
*
* 1. TESTING EXTREMAL RANGES OF  LX  AND  LAVG
*    INPUTS  - X(1...3) = 1.,2.,3.
*              AVGL(1...3,1...3,1...3) = -9.,-9.,...
*              AVGR(1...3,1...3,1...3) = -9.,-9.,...
*    USAGE   -       DO 10  LX=1,3
*                    DO 10  LAVG=1,LX
*                    CALL TAMVL(X, LX, LAVG, AVGL(1, LAVG, LX))
*               10   CALL TAMVR(X, LX, LAVG, AVGR(1, LAVG, LX))
```

* TAMVL *
****
(PAGE 2)

```
*     OUTPUTS - AVGL(1...3,1...3,1) = 1.0,-9.,-9.,,-9.,-9.,-9.,,
*                                     -9.,-9.,-9.
*               AVGL(1...3,1...3,2) = 1.5,-9.,-9.,,1.5,2.0,-9.,,
*                                     -9.,-9.,-9.
*               AVGL(1...3,1...3,3) = 2.0,-9.,-9.,,2.0,2.5,-9.,,
*                                     2.0,2.5,3.0
*               AVGR(1...3,1...3,1) = 1.0,-9.,-9.,,-9.,-9.,-9.,,
*                                     -9.,-9.,-9.
*               AVGR(1...3,1...3,2) = 1.5,-9.,-9.,,1.5,1.0,-9.,,
*                                     -9.,-9.,-9.
*               AVGR(1...3,1...3,3) = 2.0,-9.,-9.,,2.0,1.5,-9.,,
*                                     2.0,1.5,1.0
*
* 2. ILLEGAL USAGES
*     INPUTS  - SAME AS EXAMPLE 1.
*     USAGE   -       CALL TAMVL(X,-1, 2,AVGL)
*                     CALL TAMVR(X, 0, 2,AVGR)
*                     CALL TAMVL(X, 3,-1,AVGL)
*                     CALL TAMVR(X, 3, 0,AVGR)
*                     CALL TAMVL(X, 3, 4,AVGL)
*     OUTPUTS - AVGL = AVGR = -9.
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       HTR     0               XR2
       HTR     0               XR4
       BCI     1,TAMVL
*
* PRINCIPAL ENTRY.  TAMVL(X, LX, LAVG, AVGL)
*
 TAMVL STZ     ZFTAML          SET SWITCH = 0
       TRA     MERGE
*
* SECONDARY ENTRY.  TAMVR(X, LX, LAVG, AVGR)
*
 TAMVR CLA*    2,4             SET
       ARS     18              SWITCH
       ADD     K1              EQUAL
       STO     ZFTAML          LX+1
 MERGE SXD     TAMVL-3,2
       SXD     TAMVL-2,4
*
* SET ADDRESSES,  LAVG,  LX,  AND LOOP FOR  TAMVL  OR  TAMVR
*
       CLA     1,4             A(X)
       ADD     K1              A(X)+1
       STA     FAD
       SUB     ZFTAML          MINUS 0 OR MINUS LX+1
       STA     FSB
       CLA     4,4             A(AVG)  AVG = AVGL OR AVGR
       ADD     K1              A(AVG)+1
       STA     STQ
       CLA*    3,4             LAVG
       STD     TXL
       TMI     LEAVE
       TZE     LEAVE
       CAS*    2,4             AGAINST LX
       TRA     LEAVE           NG
       NOP                     OK
       CLA*    2,4             OK, WORK ON LX
       TMI     LEAVE
       PDX     0,2             LX TO XR2 FOR LOOP AT FAD
       TXL     LEAVE,2,0
       LRS     18
       ORA     OCTK
       FAD     OCTK
       STO     LENGTH          LX FLOATED TO LENGTH
       AXT     1,4             ANTICIPATE TAMVL
       ZET     ZFTAML
       AXT     -1,4            CHANGE FOR TAMVR
       SXD     TXI,4
*
```

****
*   TAMVL   *
****
(PAGE 3)

```
* NOW FORM  SUM (FROM I=1 TO LX)  OF  X(I)  IN  AC
* THE MAJOR LOOP AT  CLA  WILL SUBTRACT OFF ONE END
* VALUE OF  X  FOR EACH ADDITIONAL OUTPUT
*
       CLM
 FAD   FAD      **,2             ** = A(X)+1
       TIX      FAD,2,1          (LEAVES XR2 = 1)
       AXT      0,4              XR4 = 0
       TRA      STO
*
* LOOP.  (XR2,XR4) = (1,0),(2,1),(3,2),...  OR  (1,0),(2,-1),(3,-2),...
*
 CLA   CLA      TEMP
 FSB   FSB      **,4             ** = A(X)+1 (TAMVL)  OR  A(X)-LX (TAMVR)
 STO   STO      TEMP
       FDP      LENGTH
 STQ   STQ      **,2             ** = A(AVG)+1
       CLA      LENGTH
       FSB      K1L
       STO      LENGTH
 TXI   TXI      *+1,4,**         ** = 1 (TAMVL)  OR  -1 (TAMVR)
       TXI      *+1,2,1
 TXL   TXL      CLA,2,**         ** = LAVG
*
* EXIT
*
 LEAVE LXD      TAMVL-3,2
       LXD      TAMVL-2,4
       TRA      5,4
*
* CONSTANTS, TEMPORARIES
*
 K1    PZE      1
 K1L   DEC      1.0
 OCTK  OCT      233000000000
ZFTAML PZE      **,0,0           ** = 0  IF TAMVL,  = LX+1  IF TAMVR
 TEMP  PZE      **,**,**         SUMS OF X(I)
LENGTH PZE      **,**,**         = LX FLOATED, THEN REDUCED SUCCESSIVELY
*                                  BY 1.0
       END
```

```
************************
*     TAMVR            *
************************
 REFER TO
     TAMVL
```


```
************************
*     TAMVR            *
************************
 REFER TO
     TAMVL
```

```
*      TIMA2B (7094) (SUBROUTINE)        9/9/64   LAST CARD IN DECK IS NO. 0257
*      FAP
*TIMA2B (7094)
       COUNT    200
       LBL      TIMA2B
       ENTRY    TIMA2B (LOCA,LOCB,MINACC,SECS)
*
*                           ----ABSTRACT----
*
* TITLE - TIMA2B (7094)
*     REAL TIME, TO SPECIFIED ACCURACY, OF GIVEN PROGRAM RANGE
*
*               TIMA2B ASSUMES THAT A PROGRAM EXISTS AT MACHINE ADDRESS
*               LOCA WHICH WILL EVENTUALLY SEND CONTROL TO LOCB, AND
*               WHICH MAY BE OPERATED REPETITIVELY.  TIMA2B DETERMINES
*               THE TIME IN SECONDS, TO A SPECIFIED ACCURACY, THAT ONE
*               OPERATION OF THE PROGRAM REQUIRES.  THE TIME INCLUDES THE
*               TIME OF THE OPERATION AT LOCA BUT NOT THAT OF THE
*               OPERATION AT LOCB.  THE AC AND MQ ARE LEFT AS PRODUCED BY
*               THE PROGRAM.
*
*               CONSTANTS USED IN THE PRESENT VERSION PERTAIN TO THE
*               7094.  THE NECESSARY MODIFICATIONS FOR THE 7090 ARE
*               INDICATED IN THE DECK.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 7090 OR 7094 (MAIN FRAME PLUS INTERVAL TIMER)
* STORAGE    - 124 REGISTERS
* SPEED      - TAKES SOMEWHAT LESS THAN
*                 MAX(2*MINACC/60., SECS,
*                     MINACC*(SECS+.000048)/(SECS*60.) )
*               SECONDS ON THE 7094 MOD 1, WHERE  SECS  IS THE
*               MEASURED TIME BETWEEN  LOCA  AND  LOCB  IN SECONDS,
*               AND WHERE THE USER SPECIFIES THAT THE TIMING ERROR
*               SHALL NOT EXCEED ONE PART IN  MINACC  PARTS.
* AUTHOR     - S.M. SIMPSON JR. AND R.A. WIGGINS
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL TIMA2B(LOCA,LOCB,MINACC,SECS)
*
* INPUTS
*
*     LOCA      IS MACHINE ADDRESS (AS FORTRAN-II INTEGER) OF FIRST
*                  INSTRUCTION IN PROGRAM TO BE TIMED.
*
*     LOCB      IS MACHINE ADDRESS TO WHICH CONTROL IS SENT AFTER
*                  PROGRAM.  IT EQUALS 1 + MACHINE ADDRESS OF LAST
*                  INSTRUCTION IF LAST INSTRUCTION IS NOT A TRANSFER.
*
*     MINACC    SPECIFIES THAT THE TIMING ERROR SHALL NOT EXCEED ONE PART
*                  IN MINACC PARTS.
*               MUST EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF MINACC IS ILLEGAL.
*
*     SECS      IS THE REQUIRED TIME IN FLOATING POINT SECONDS.
*
*     THE ACCUMULATOR AND MULTIPLIER QUOTIENT REGISTERS WILL HAVE VALUES
*               AS LEFT BY THE PROGRAM WHEN CONTROL ARRIVES AT LOCB.
*
*               WARNING  -  IF THE PROGRAM CONTAINS OUTPUT OR INPUT
*               INSTRUCTIONS THEY WILL BE OPERATED REPETITIVELY IF
*               NECESSARY TO ACHIEVE THE REQUIRED ACCURACY.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...1001) = OCT 053400000000. THIS IS THE MACHINE
*               INSTRUCTION LXA , WHICH ALWAYS TAKES 2 CYCLES ON THE
*               7090,7094.
*               LOCB=XLOCF(X),  LOCA1=LOCB-1000,  LOCA2=LOCB-100,
```

```
*              LOCA3=LOCB-10, LOCA4=LOCB-2.  THESE WILL DEFINE 4
*              PROGRAMS OF LENGTH 1000, 100, 10, AND 2 INSTRUCTIONS
*              RESPECTIVELY.  MINACC = 100
*
*    USAGE   -        CALL TIMA2B(LOCA1,LOCB,MINACC,SECS1)
*                     CALL TIMA2B(LOCA2,LOCB,MINACC,SECS2)
*                     CALL TIMA2B(LOCA3,LOCB,MINACC,SECS3)
*                     CALL TIMA2B(LOCA4,LOCB,MINACC,SECS4)
*
*    OUTPUTS - SECS1...SECS4= .00400 .000400 .0000400 .00000800(ON 7094)
*                           = .00436 .000436 .0000436 .00000872(ON 7090)
*                  THE ACTUAL RESULTS SHOULD DEVIATE FROM THESE ANSWERS
*                  BY NO MORE THAN 1 PERCENT.
*
* PROGRAM FOLLOWS BELOW
*
* THE CONSTANTS KC2MC AND KLUP BELOW PERTAIN TO THE 7094
* FOR THE 7090 THEY SHOULD READ
*
*      KC2MC  DEC    7645.259            NO. MACH. CYCLES PER COUNT (7090)
*      KLUP   DEC    27.0                NO. MACH. CYCLES IN LOOP CONTROL
*                                                     (7090)
*
* NO TRANSFER VECTOR
       HTR     0             XR1
       HTR     0             XR2
       HTR     0             XR4
       BCI     1,TIMA2B
* ONLY ENTRY.  TIMA2B(LOCA,LOCB,MINACC,SECS)
TIMA2B SXD     TIMA2B-2,4
       SXD     TIMA2B-3,2
       SXD     TIMA2B-4,1
       STO     AC
       STQ     MQ
* CHECK MINACC  (SHOULDNT EXCEED 1000)
       CLA*    3,4           MINACC
       TMI     LEAVE
       TZE     LEAVE
* OK, FLOAT IT
       ARS     18
       STO     MINACX
       ORA     OCTK
       FAD     OCTK
       STO     MINACC
       FAD     ONE
       STO     MINAC1
* THEN SET UP LINKAGE TO LOCA AND BACK FROM LOCB
       CLA*    1,4           LOCA
       ARS     18
       STA     TRAOUT
       CLA*    2,4           LOCB
       ARS     18
       STA     STOXEC
       CLA*    STOXEC        (ORIGINAL CONTENTS OF LOCB)
       STO     SAVNXT
       CLA     TRABAK        (XEC TRABAK)
STOXEC STO     **            ** = LOCB
* SET KLUP = 23. IF ADDRESS(LOOP) IS ODD
* SET KLUP = 24. IF ADDRESS(LOOP) IS EVEN
       CLA     TRALUP
       ANA     K1
       SSM
       ADD     K24
       ORA     OCTK
       FAD     OCTK
       STO     KLUP
* CLEAR THE LOOPS COUNTER
 STZ   STZ     NLOOPS
* SET INITIAL TIME
       CLA     5             ADDS 0 CYCLES TO KEDGE
       STO     BEGIN         ADDS 2 CYCLES TO KEDGE
* LOOP BEGINS
* RESTORE XRS, AC, MQ, BEFORE ENTERING
 LOOP  LXD     TIMA2B-2,4    ADDS 2 CYCLES TO KLUP
       LXD     TIMA2B-3,2    ADDS 2 CYCLES TO KLUP
```

```
       LXD     TIMA2B-4,1     ADDS 2 CYCLES TO KLUP
       CLA     AC             ADDS 2 CYCLES TO KLUP
       LDQ     MQ             ADDS 2 CYCLES TO KLUP
TRAOUT TRA     ** **=LOCA     ADDS 1 CYCLE  TO KLUP
*
*TRABAKTRA     BACK           ADDS 1 CYCLE  TO KLUP
*
 BACK  STO     ACAFTR         ADDS 2 CYCLES TO KLUP
       CLA     NLOOPS         ADDS 2 CYCLES TO KLUP
       ADD     K1             ADDS 2 CYCLES TO KLUP
       STO     NLOOPS         ADDS 2 CYCLES TO KLUP
       CLA     5              ADDS 2 CYCLES TO KLUP
       SUB     BEGIN          ADDS 2 CYCLES TO KLUP
       CAS     MINACX         ADDS 2 CYCLES TO KLUP
       NOP
       TRA     LUPOVR
TRALUP TRA     LOOP           ADDS 1 CYCLES TO KLUP
LUPOVR STQ     MQAFTR
*
* THEN FORM DIFFERENCE OF COUNTS AND CHECK IF ADEQUATE
* WE HAVE
*   KC2MC*COUNTD = KEDGE + NLOOPS*(X+KLUP)
*   WHERE KC2MC = NO. MACHINE CYCLES/COUNT
*        COUNTD = TERMINAL COUNT MINUS INITIAL COUNT
*         KEDGE = NO. MACHINE CYCLES IN LOOP EDGE EFFECT
*          KLUP = NO. MACHINE CYCLES INSIDE LOOP EXCLUSIVE OF PROGRAM
*             X = NO. MACHINE CYCLES IN PROGRAM
*
* THUS THE NUMBER OF COUNTS SPENT INSIDE THE PROGRAM IS
*   NCIP = (NLOOPS*X)/KC2MC
*        = COUNTD - (KEDGE + NLOOPS*KLUP)/KC2MC
* NCIP MUST EXCEED MINACC
*
* FIND NCIP
 CNTD2 ORA     OCTK
       FAD     OCTK           FLOATING COUNTD
       STO     COUNTD
       CLA     NLOOPS
       ORA     OCTK
       FAD     OCTK
       STO     FNLUPS
       XCA
       FMP     KLUP           TIMES KLUP
       FAD     KEDGE          PLUS KEDGE
       FDP     KC2MC          OVER KC2MC)
       XCA
       CHS
       FAD     COUNTD  NCIP=COUNTD MINUS DITTO
       STO     NCIP
       CAS     MINACC
       NOP
       TRA     ENUF           ENOUGH
* FOR AN INADEQUATE COUNT WE COMPUTE A REASONABLE MINACC AND TRY AGAIN
       FSB     ONE
       STO     NCIP
       LDQ     KC05
       TLQ     *+2
       STQ     NCIP
       CLA     MINAC1
       FDP     NCIP
       FMP     COUNTD
       UFA     OCTK
       ANA     OCTK1
       ADD     K1
       STO     MINACX
       TRA     LOOP
* FOR ADEQUATE COUNT, RESTORE LOCB, CONVERT NCIP TO SECONDS, EXIT
 ENUF  LXD     TIMA2B-2,4
       CLA     SAVNXT
       STO*    STOXEC         RESTORE LOCB
       LDQ     NCIP
       FMP     KC2S           KC2S = NO. SECONDS PER COUNT
       FDP     FNLUPS         SECS = (COUNTD*KC2S)/NLOOPS
       STQ*    4,4
 LEAVE LXD     TIMA2B-2,4
```

****
*   TIMA2B (7094)   *
****
(PAGE 4)

```
        LXD     TIMA2B-3,2
        LXD     TIMA2B-4,1
        CLA     ACAFTR
        LDQ     MQAFTR
        TRA     5,4
* CONSTANTS
 ONE    DEC     1.
 K1     PZE     1
K24     PZE     24
 OCTK   OCT     233000000000
 OCTK1  OCT     000777777777
TRABAK  TRA     BACK
 KC2MC  DEC     8333.3333     = NO. MACHINE CYCLES PER CLOCK COUNT (7094)
 KC2S   DEC     .016666667    = NO. SECONDS PER CLOCK COUNT (1/60)
KC05    DEC     .05
KEDGE   DEC     86.0          = APPROXIMATE NUMBER CYCLES IN LOOP EDGES
*                               40. LSTHN  KEDGE  LSTHN 134.
 KLUP   DEC     0.             = NO. CYCLES IN LOOP CONTROL (EMPIRICAL)
* VARIABLES
SAVNXT PZE     **,**,**       INITIAL CONTENTS OF LOCB
NLOOPS PZE     **             INITIAL SET = 0
FNLUPS PZE     **,**,**       FLTG. NLOOPS
COUNTD PZE     **,**,**       FLTG. COUNT DIFFERENCE
 NCIP   PZE     **,**,**       FLTG. COUNT INSIDE PROGRAM
MINACC PZE     **,**,**       FLTG. MINACC
MINACX PZE     **,**,**       FXD. MINACC
MINAC1 PZE     **,**,**       FLTG. MINACC+1
 BEGIN PZE     **,**,**       INITIAL CLOCK COUNT (FIXED)
 AC     PZE     **,**,**       ORIGINAL AC
 MQ     PZE     **,**,**       ORIGINAL MQ
ACAFTR PZE     **,**,**       AC AFTER PROGRAM
MQAFTR PZE     **,**,**       MQ AFTER PROGRAM
        END
```

```
*      TIMSUB (SUBROUTINE)              9/8/64   LAST CARD IN DECK IS NO. 0449
*      FAP
*TIMSUB
       COUNT    250
       LBL      TIMSUB
       ENTRY    TIMSUB (MINACC,SECS)
       ENTRY    INTMSB
*
*                          ----ABSTRACT----
*
* TITLE - TIMSUB
*         FIND OPERATION TIME OF NEXT SUBROUTINE TO GIVEN ACCURACY
*
*               TIMSUB IS CALLED JUST PRIOR TO A CALL SUBROUTINE
*               STATEMENT, OF FORM CALL SUBRU(A,B,...Z), OR TO A FUNCTION
*               STATEMENT, OF FORM X = SOMEF(A,B,...,Z).  THE SUBROUTINE
*               OR FUNCTION SHOULD BE CAPABLE OF BEING CALLED IN A
*               REPETITIVE LOOP.  TIMSUB THEN OPERATES THE SUBROUTINE OR
*               FUNCTION ENOUGH TIMES TO MEASURE THE REAL TIME IT TAKES
*               (IN SECONDS) TO A SPECIFIED ACCURACY.  THE TIME RESULTING
*               IS MEASURED FROM THE FIRST INSTRUCTION IN THE SUBROUTINE
*               THROUGH ITS FINAL RETURN TRANSFER.  TIMSUB LEAVES THE AC
*               AND MQ WITH VALUES AS INDUCED BY THE SUBROUTINE OR
*               FUNCTION, AND RETURNS CONTROL JUST BEYOND THE CALL SUBRU
*               (OR X=SOMEF) STATEMENT.
*
*               INTMSB IS USED FOR CERTAIN SUBROUTINES (SAY WHERE
*               OUTPUTS REPLACE INPUTS) WHICH REQUIRE AN INPUT SETUP
*               SEQUENCE BEFORE EVERY USAGE.  IN SUCH CASES TIMING IS
*               PERFORMED AS ABOVE EXCEPT 1)IMMEDIATELY PRECEEDING THE
*               CALL TIMSUB STATEMENT THE INPUT SETUP SEQUENCE MUST
*               OCCUR, AND 2)IMMEDIATELY PRECEEDING THE INPUT SETUP
*               SEQUENCE THERE EXISTS A CALL INTMSB STATEMENT.  THE
*               COMBINED SEQUENCE OF INPUT SETUP PLUS CALL SUBRU (OR
*               X=SOMEF) SHOULD BE CAPABLE OF REPETITIVE OPERATION.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090, OR 7094 (MAIN FRAME PLUS SOME FORM OF REAL TIME
*               CLOCK)
*               THE TIMING IS PERFORMED BY SUBROUTINE TIMA2B.  A
*               DIFFERENT VERSION OF TIMA2B MUST BE USED FOR EACH
*               MACHINE.  AS OF SEPTEMBER, 1963 NO VERSION OF TIMA2B
*               EXISTS FOR THE 709.
* STORAGE    - 229 REGISTERS
* SPEED      - IF WE LET  TMSUB  BE THE TIME IN SECONDS TAKEN BY THE
*               SUBROUTINE AND  TMSET  BE THE TIME IN SECONDS TAKEN BY
*               THE SETUP SEQUENCE, THEN  TTIM,  THE TIME IN SECONDS
*               TAKEN BY  TIMSUB  TO OBTAIN THE ESTIMATE IS APPROXIMATED,
*               IN THE CASE OF THE 7094 MOD 1, BY
*
*                     TTIM = MINACC*(TIME1+TIME2)
*
*               WHERE  TIME1 = MAX(.0375, TMSUB,
*                                 (TMSUB+.000048)/(53.33*TMSUB) )
*                 AND  TIME2 = MAX(.00469, TMSET,
*                                 (TMSET+.000048)/(426.67*TMSET) )
*
*               IF  TMSUB*60.  GRTHN= MINACC,  OR
*               IF  TMSUB/(8.*TMSET)  GRTHN=  1.
*
*               IF NEITHER OF THESE CONDITIONS HOLD, THEN  TTIM  IS
*               APPROXIMATED BY
*
*                     TTIM LSTHN= MINACC(TIME1*(1.+(TMSUB+TMSET)/TMSUB) +
*                                         TIME2*(1.+TMSET/TMSUB)
*
*               WHEN  INTMSB IS NOT USED,  THE MINIMUM  TMSET  IS
*               .000028 SECONDS.  IT MAY BE GREATER THAN THIS IF THE
*               SUBROUTINF CALLED HAS SUBSCRIPTED ARGUMENTS.
*
* AUTHOR     - S.M. SIMPSON JR. AND R.A. WIGGINS
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - TIMA2B
```

```
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE FOR ROUTINES NOT REQUIRING INPUT SETUP FOR EACH CALL
*
*     CALL TIMSUB(MINACC,SECS)
*     CALL SUBRU(A,B,...,Z)
* OR
*     CALL TIMSUB(MINACC,SECS)
*     X = SOMEF(A,B,...,Z)
*
* FORTRAN USAGE FOR ROUTINES REQUIRING INPUT SETUP BEFORE EACH CALL
*
*     CALL INTMSB
*   (INSERT HERE PROGRAM TO SET UP INPUTS FOR SUBRU OR SOMEF)
*     CALL TIMSUB(MINACC,SECS)
*     CALL SUBRU(A,B,...,Z)      OR   X = SOMEF(A,B,...,Z)
*
* INPUTS
*
*    MINACC     SPECIFIES THAT THE TIMING ERROR SHOULD NOT EXCEED ONE
*                  PART IN MINACC PARTS.
*               MUST EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF MINACC IS ILLEGAL.
*
*    SECS       IS THE DESIRED TIME.
*
*               IF THE SUBROUTINE OR FUNCTION HAS ANY OUTPUTS THEY WILL
*               BE THE SAME AS IF THE CALL TIMSUB STATEMENT WERE OMITTED.
*
* EXAMPLES
*
*    SUPPOSE SUBROUTINE LXAZ IS THE FOLLOWING PROGRAM REQUIRING 100
*    MACHINE CYCLES
*
*               *     FAP
*                     COUNT   50
*                     LBL     LXAZ
*                     ENTRY   LXAZ
*               LXAZ  LXA     0,0
*                     LXA     0,0
*                         .
*                         .
*                         .
*                        ETC FOR A TOTAL OF 48 LXA 0,0 INSTRUCTIONS
*                         .
*                         .
*                         .
*                     LXA     0,0
*                     XEC     *+1
*                     XEC     *+1
*                     TRA     1,4
*
*
* 1. USAGE   -        CALL TIMSUB(100,SECS)
*                     CALL LXAZ
*    OUTPUTS - SECS = .000218 (7090) OR .000200 (7094) WITH ERROR LESS
*                      THAN .000002
*
* 2. USAGE   -        CALL INTMSB
*                     X=4
*                     Y=COSF(3.)
*                     CALL TIMSUB(100,SECS)
*                     CALL LXAZ
*    OUTPUTS - SAME AS EXAMPLE 1.
*
* 3. USAGE   -        CALL TIMSUB(100,SECS)
*                     X = SQRTF(9.8696044)
*
*    OUTPUTS - X = 3.1415627 AND SECS = .000168 (APPROXIMATELY)
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS TIMA2B(LOCA,LOCB,MINACC,SECS)
```

```
       HTR     0              XR1      RELATIVE
       HTR     0              XR2         TO
       HTR     0              XR4       TIMSUB
       BCI     1,TIMSUB
* PRINCIPLE ENTRY.  TIMSUB(MINACC,SECS)
* (STRAIGHT RETURN IF UNDER CONTROL OF INTMSB)
TIMSUB SXD     TIMSUB-2,4
       SXD     TIMSUB-3,2
       SXD     TIMSUB-4,1
       NZT     ZIFINT
       TRA     3,4
*
* PRELIMINARY NOTATION FOR TIMSUB AND INTMSB
*
* TTSUB  TTR   SUB
* TTINT  TTR   INTMSB         (MAY BE MISSING)
* TTTIM  TTR   TIMSUB
*              .
*              .              (ARBITRARY AREA)
*              .
* TSXINT TSX   TTINT,4        (MAY BE MISSING)
*              .
*              .              (INPUT SETUP AREA)
*              .
* TSXTIM TSX   TTTIM,4
*        TSX   A(MINACC),0
*        TSX   A(SECS),0
*              .
*              .              (FORTRAN INPUT SETUP AREA)
*              .
* TSXSUB TSX   TTSUB,4
*        TSX   A(ARG1),0
*        TSX   A(ARG2),0
*              .
*              .
*              .
* TSXRGN TSX   A(ARGN),0
* FINISH       .
*              .              (ARBITRARY AREA)
*              .
*
*
*
* CONTINUE WITH TIMSUB ENTRY
       SXA     BEGIN,1
       SXA     BEGIN+1,2
       PXA     0,4
       PAC     0,1
       TXI     *+1,1,3
       SXA     START,1
       SXD     DSTART,1
       TRA     SETUP
* SECOND ENTRY.  INTMSB
       HTR     0              XR1 RELATIVE
       HTR     0              XR2   TO
       HTR     0              XR4  INTMSB
INTMSB SXD     INTMSB-1,4
       SXD     INTMSB-2,2
       SXD     INTMSB-3,1
       SXA     BEGIN,1
       SXA     BEGIN+1,2
* TURN ON INTMSB SWITCH
       STZ     ZIFINT
       PXA     0,4
       PAC     0,1
       TXI     *+1,1,1
       SXA     START,1
       SXD     DSTART,1
* SCAN DOWN THE TSX X,4*S UNTIL HIT TSX $TIMSUB,4
LUKTSB TSX     TSX4SC,1
       TXI     *+1,4,-1
       CLA     0,4
       STA     *+1
       CAL     **             PICK UP THE TTR
       LAS     TTRTIM
```

```
       TRA     *+2
       TRA     SETUP
       TRA     LUKTSB
* MERGE POINT.  0,4 IS NOW = TSX $TIMSUB,4
* SET MNCWI,MNCWO,MNACC, AND FMNACC
 SETUP CLA*    1,4            MINACC FXD
       TZE     3,4
       TMI     3,4
       STO     MNACC
       ARS     3
       ADD     MNACC
       STO     MNCWI
       STO     MNACC
       ZET     ZIFINT
       ARS     3
       STO     MNCWO
       LDQ     KD1
       TLQ     *+2
       STQ     MNCWO
       CLA*    1,4
       ARS     18
       ORA     OCTK
       FAD     OCTK
       STO     FMNACC
* SCAN DOWN TO TSX $SUB,4    (IN 1,4)
SUBSCN TSX     TSX4SC,1
       TXI     *+1,4,-1
       CLA     0,4            TSX $SUB,4
* WE HAVE TO IGNORE INTERNAL FORTRAN SUBROUTINES, FOR WHICH THE
* TSX X,4 HAS X GRTHN THE LOCATION OF THAT INSTRUCTION, IJE.  -X LSTHN
* PRESENT VALUE OF XR4.
       PAC     0,1            -X TO XR1
       SXD     *+1,4
       TXL     SUBSCN,1,**    ** = XR4
* OK, LEGITIMATE SUBROUTINE FOUND
       STO     SAVTSX
       PXA     0,4
       PAC     0,1            A(TSX $SUB,4)
       SXA     TSXSUB,1
       CLA     0,4            TSX A(TTR SUB),4
       STA     TTSUB
       CLA*    TTSUB
       STO     SAVTTR
       CLA     TSXSXA
       STO     0,4
* SCAN DOWN TILL 1,4 = NON TSX X,0
ARGSCN TSX     TSXZCK,1
       TXI     PXAARG,4,-1    NO MORE
       TXI     ARGSCN,4,-1    KEEP GOING
PXAARG PXA     0,4     0,4 IS NOW EQUIV  A(TSX ARGN,0)+1
       PAC     0,1
       SXA     FINISH,1
       SXD     LOCB,1
       SXA     TTRLOB,1
*
* TIMING LOOP. STEPS ARE
*
* STEP 1. SET BYPASS IN TTSUB AND GO FIND TMWOUT
TIMLUP CLA     TTRLOB
       STO*    TTSUB
       CLA     MNCWO
       TSX     OPTMAB,1
       CLA     SECSL
       STO     TMWOUT
* STEP 2.  REPLACE THE TTR SUB AND GO FIND TMWITH
TMLUP1 CLA     SAVTTR
       STO*    TTSUB
       CLA     MNCWI
       TSX     OPTMAB,1
       CLA     SECSL
       STO     TMWITH
* STEP 3. FIND TMSUB
       FSB     TMWOUT         FORM TMSUB,
       STO     TMSUB
* STEP 4.  IF TMSUB*60. GRTHN= MNACC, LEAVE
```

****
* TIMSUB *
****
(PAGE 5)

```
       XCA
       FMP     F60
       CAS     FMNACC
       NOP
       TRA     LEAVE
* STEP 5. IF MNCWO*TMSUB/TMWOUT  GRTHN= MNACC,  LEAVE
       CLA     MNCWO          FORM FLOATING MNCWO
       ARS     18
       ORA     OCTK
       FAD     OCTK
       STO     FMNCWO         DONE.
       CLA     TMSUB
       FDP     TMWOUT
       FMP     FMNCWO
       CAS     FMNACC
       NOP
       TRA     LEAVE
* STEP 6. PREDICT NEW MNCWO AND MNCWI
       CAS     F1
       TRA     FMCWO
* TMSUB HAS  LSTHN= 1 SIGNIFICANT FIGURES
* SET NEW MNCWO=MNCWI = MNACC*MNCWI
       NOP
       LDQ     MNCWI
       MPY     MNACC
       ALS     17
       STO     MNCWI
       STO     MNCWO
       TRA     TIMLUP
* SET MNCWI = MNACC*TMWITH/TMSUB
* SET MNCWO = MAX (MNCWO, MNACC*TMWOUT/TMSUB)
FMCWO  CLA     FMNACC
       FDP     TMSUB
       FMP     TMWITH
       UFA     OCTK
       ALS     15
       STO     MNCWI
       ALS     3
       ADD     MNCWI
       ADD     K1
       STO     MNCWI
       CLA     FMNACC
       FDP     TMSUB
       FMP     TMWOUT
       UFA     OCTK
       ALS     15
       LDQ     MNCWO
       ADD     K1
       STO     MNCWO
       ALS     3
       ADD     MNCWO
       STO     MNCWO
       TLQ     TIMLUP
       STQ     MNCWO
       TRA     TMLUP1
* EXIT SEQUENCE
 LEAVE LXD     TIMSUB-2,4
       CLA     TMSUB
       STO*    2,4            RELATIVE TO TIMSUB
       CLA     SAVTSX
       STO*    TSXSUB
       CLA     K1
       STO     ZIFINT
 AXTLV AXT     **,1           ** = XR1 ENTERING SUB
       AXT     **,2           ** = XR2 ENTERING SUB
       AXT     **,4           ** = XR4 ENTERING SUB
       CLA     SAVAC          AC AFTER SUBROUTINE
       LDQ     SAVMQ          MQ AFTER SUBROUTINE
       TOV     *+1
       TRA*    FINISH
* INTERNAL SUBROUTINE TO OPERATE TIMA2B
*      LINKAGE XR1, RETURNS TO 1,1
OPTMAB SXA     OTABLV,1
       SXA     OTABLV+1,2
       STO     MNAC
```

```
       TSX    $TIMA2B,4
       TSX    LOCA,0
       TSX    LOCB,0
       TSX    MNAC,0
       TSX    SECSL,0
       STO    SAVAC
       STQ    SAVMQ
OTABLV AXT    **,1             ** = XR1
       AXT    **,2             ** = XR2
       TRA    1,1
* THE FOLLOWING IS LOCA FOR TIMA2B
 BEGIN AXT    **,1             ** = APPROPRIATE XR1
       AXT    **,2             ** = APPROPRIATE XR2
       TRA*   START
* (THE FOLLOWING IS A PATCH IN OPERATING LOOP LOCA TO LOCB)
LUPSXA SXA    AXTLV,1
       SXA    AXTLV+1,2
       SXA    AXTLV+2,4
       TRA*   TTSUB
* INTERNAL SUBROUTINE TO BUMP XR4 UNTIL 1,4 = TSX X,4
*      LINKAGE XR1,  RETURN TO 1,1
TSX4SC SXA    T4SCLV,1
CLATS4 CLA    TSXZ4
       TSX    INSTCK,1
       TXI    CLATS4,4,-1   NO
T4SCLV AXT    **,1
       TRA    1,1
* INTERNAL SUBROUTINE TO CHECK IF 1,4 = TSX X,0
*      RETURNS TO 1,1 IF NOT,  TO 2,1 IF SO
TSXZCK CLA    TSXZ
* INTERNAL SUBROUTINE TO CHECK IF 1,4 , LESS ADDRESS, EQUALS AC
*      RETURNS TO 1,1 IF NOT,  TO 2,1 IF SO
INSTCK STO    TEMP
       CAL    1,4
       ANA    AMASK
       LAS    TEMP
       TRA    *+2
       TRA    2,1             YES
       TRA    1,1             NO
* CONSTANTS, TEMPORARIES
 AMASK OCT    777777700000
 TSXZ  TSX    0,0
 TSXZ4 TSX    0,4
TSXSXA TSX    LUPSXA,4
 K1    PZE    1
KD1    PZE    ,,1
F1     DEC    1.
F60    DEC    60.
TTRTIM TTR    TIMSUB
 OCTK  OCT    233000000000
TTRLOB TTR    **               ** = LOCB
 START PZE    **               ** = TSXINT+1 OR TSXTIM+3
DSTART PZE    0,0,**           ** = DITTO
FINISH PZE    **               ** = A(TSX ARGN,0)+1
TSXSUB PZE    **               ** = A(TSX $SUBZ4)
 TTSUB PZE    **               ** = A(TTR SUB)
SAVTSX TSX    **,4             ** = $SUB
SAVTTR TTR    **               ** = SUB
TMWOUT PZE    **,**,**
TMWITH PZE    **,**,**
 TMSUB PZE    **,**,**
FMNACC PZE    **,**,**         FLTG. MNACC
 MNACC PZE    0,0,**           FOR TIMA2B
 MNAC  PZE
FMNCWO PZE
 MNCWO PZE
 MNCWI PZE
 SECSL PZF    **,**,**         FROM TIMA2B
 LOCB  PZE    0,0,**           FOR TIMA2B (SAME AS FINISH)
  LOCA PZE    0,0,BEGIN        FOR TIMA2B
 SAVAC PZE    **,**,**         AC AFTER SUBROUTINE
 SAVMQ PZE    **,**,**         MQ AFTER SUBROUTINE
 TEMP  PZE    **,**,**
ZIFINT PZE    1     ZEROED BY INTSSB, RESET TO 1 ON EXIT
       END
```

```
*      TINGL (SUBROUTINE)                 9/8/64   LAST CARD IN DECK IS NO. 0146
*      FAP
*TINGL
       COUNT    150
       LBL      TINGL
       ENTRY    TINGL  (YOFX, LY, DELX, TING)
       ENTRY    TINGLA (YOFX, LY, DELX, TINGA)
*
*
*                    ----ABSTRACT----
*
* TITLE - TINGL,  WITH SECONDARY ENTRY TINGLA
*        DEFINITE TRAPEZOIDAL INTEGRAL OF FUNCTION OR ITS MAGNITUDE
*
*              TINGL  COMPUTES TRAPEZOIDAL INTEGRALS OF FORM
*
*                                  LY-1
*                    DELX * (Y(1)/2 +   SUM Y(I)   + Y(LY)/2)
*                                    I=2
*
*              WHERE Y(1...LY),  LY,  AND  DELX  ARE INPUTS, AND
*              WHERE THE SUMMATION IS SUPPRESSED FOR  LY = 2 .
*
*              TINGLA  COMPUTES THE SAME EXPRESSION BUT USES ABSOLUTE
*              VALUES OF  Y  RATHER THAN  Y  ITSELF.
*
* LANGUAGE    - FAP SUBROUTINES ( FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709,7090,7094  (MAIN FRAME ONLY)
* STORAGE     - 43 REGISTERS
* SPEED       - TAKES ABOUT  70 + 8.4*LY  MACHINE CYCLES, ON THE 7090
* AUTHOR      - S.M. SIMPSON, JUNE 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NOT ANY
*      AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     CALL TINGL (YOFX, LY, DELX, TING)
*     CALL TINGLA(YOFX, LY, DELX, TINGA)
*
*
* INPUTS
*
*    YOFX(I)    I=1...LY  IS THE FLOATING VECTOR  Y(I)  OF THE ABSTRACT.
*
*    LY         MUST EXCEED  1 .
*
*    DELX       IS THE INCREMENT.  MAY BE POSITIVE OR NEGATIVE.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LY  LSTHN  2 .
*
*    TING       IS THE OUTPUT OF  TINGL,  GIVEN BY THE EXPRESSION IN THE
*                  ABSTRACT.
*
*    TINGA      IS THE OUTPUT OF  TINGLA  AS DESCRIBED IN ABSTRACT.
*                  NOTE THAT THE SIGN OF  TINGA  IS THAT OF  DELX.
*
*
* EXAMPLES
*
* 1. INPUTS   - YOFX(1...4) = -2.,-1.,1.,3.  DELX = 2.0
*               TING(1...4) = TINGA(1...4) = -99.,-99.,-99.,-99.
*    USAGE    -       DO 10  LY=1,4
*                     CALL TINGL (YOFX, LY, DELX, TING(LY))
*                  10 CALL TINGLA(YOFX, LY, DELX, TINGA(LY))
*    OUTPUTS - TING (1...4) = -99.,-3.0,-3.0,1.0
*              TINGA(1...4) = -99., 3.0, 5.0, 9.0
*
*
* PROGRAM FOLLOWS BELOW
*
```

```
* NO TRANSFER VECTOR
*
       HTR     0               XR1
       BCI     1,TINGL
*
* PRINCIPAL ENTRY.  TINGL(YOFX, LY, DELX, TING)
*
 TINGL STZ     ZFTNGL
       TRA     SETUP
*
* SECONDARY ENTRY.  TINGLA(YOFX, LY, DELX, TINGA)
*
TINGLA SXD     ZFTNGL,4
*
* CHECK  LY  AND SET ADDRESS OF  YOFX
*
 SETUP SXD     TINGL-2,1
       CLA*    2,4             LY
       TMI     LEAVE
       PDX     0,1
       TXL     LEAVE,1,1       MUST EXCEED  1
       CLA     1,4             A(YOFX)
       STA     FAD
       STA     FAM
*
* SET VARIABLE INSTRUCTIONS ACCORDING TO ENTRY
*
       CLA     NOP             ANTICIPATE FOR TINGL
       LDQ     FAD
       NZT     ZFTNGL
       TRA     NOPSET
       CLA     SSP             NO, IT WAS TINGLA
       LDQ     FAM
NOPSET STO     NOP1
       STQ     NOP2
       STQ     NOP3
*
* FIRST AND LAST TERMS OVER 2
*
       TXI     *+1,1,-1        (XR1 = LY-1)
       CLA*    1,4             YOFX(1)
 NOP1  NOP                     =  NOP  OR  SSP
 NOP2  NOP                     =  FAD **,1  OR  FAM **,1    ** = A(YOFX)
       XCA
       FMP     KP5
       TXI     *+1,1,-1        (XR1 = LY-2)
       TXL     XCA,1,0         (BYPASS FOR LY = 2)
*
* CENTRAL TERMS LOOP.  XR1 = LY-2,LY-3,...,1
*
 NOP3  NOP                     =  FAD **,1  OR  FAM **,1    ** = A(YOFX)
       TIX     NOP3,1,1
*
* TIMES DELX AND STORED
*
 XCA   XCA
       FMP*    3,4             DELX
       STO*    4,4             TING
*
* EXIT
*
 LEAVE LXD     TINGL-2,1
       TRA     5,4
*
* CONSTANTS, VARIABLES
*
 KP5   DEC     .5
 NOP   NOP
 SSP   SSP
 FAD   FAD     **,1            ** = A(YOFX)
 FAM   FAM     **,1            ** = A(YOFX)
ZFTNGL PZE     0,0,**          ** = 0 IF TINGL,  = XR4 IF TINGLA
       END
```

```
************************
*    TINGLA            *
************************
 REFER TO
     TINGL
```

```
************************
*    TINGLA            *
************************
 REFER TO
     TINGL
```

```
*     TRMINO (SUBROUTINE)                 9/4/64   LAST CARD IN DECK IS NO. 0076
*     LABEL
CTRMINO
      SUBROUTINE TRMINO(ITAPE, NBAKUP)
C
C
C                    ----ABSTRACT----
C
C TITLE - TRMINO
C        TERMINATE AN INDATA-OUDATA TAPE
C
C              TRMINO  USES SUBROUTINE  OUDATA  TO WRITE A ZERO-RECORD
C              NUMBER DUMMY RECORD ON A SPECIFIED UNIT AND THEN BACKS
C              THE TAPE UP A SPECIFIED NUMBER OF FILES OR REWINDS IT.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709,7090,7094  (MAIN FRAME PLUS ONE TAPE DRIVE)
C STORAGE    - 67 REGISTERS
C SPEED      - CONTROLLED BY SUBROUTINES OUDATA AND FSKIP
C AUTHOR     - S.M. SIMPSON, JUNE 1964
C
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - XLIMIT, OUDATA, FSKIP
C       AND FORTRAN SYSTEM ROUTINES - (RWT)
C
C FORTRAN USAGE
C     CALL TRMINO(ITAPE, NBAKUP)
C
C
C INPUTS
C
C    ITAPE     MUST EXCEED ZERO AND BE  LSTHN=  20
C
C    NBAKUP    GRTHN=  0  REQUESTS  TRMINO  TO LEAVE TAPE POSITIONED
C                 NBAKUP  FILES CLOSER TO TAPE START THAN ITS INPUT
C                 POSITION.  NBAKUP=0  LEAVES TAPE READY TO READ
C                 THE DUMMY RECORD CREATED.
C              LSTHN   0  REQUESTS  TRMINO  TO LEAVE TAPE REWOUND.
C
C
C OUTPUTS      NO ACTION IF  ITAPE  ILLEGAL.  OTHERWISE SEE ABSTRACT
C                 AND  NBAKUP.
C
C
C EXAMPLES
C
C 1. INPUTS  - SUPPOSE A 5 RECORD SAMPLE INDATA-OUDATA TAPE HAS BEEN
C              CREATED ON LOGICAL 9 BY THE FOLLOWING SEQUENCE.
C                    DO 10  I=1,10
C                 10 X(I) = FLOATF(I)
C                    REWIND 9
C                    DO 20  I=1,5
C                    IRECNO = I
C                 20 CALL OUDATA(9, IRECNO, 10, X, 1)
C
C    USAGE   -       CALL TRMINO(9, 2)
C                    DO 30  I=1,3
C                    IRECNO(I) = 0
C                    NOPTS = -1
C                 30 CALL INDATA(9, IRECNO(I), NOPTS, DUMMY, ERR)
C
C    OUTPUTS - IRECNO(1...3) = 4,5,0
C
C
C PROGRAM FOLLOWS BELOW
C
      IF (XLIMITF(ITAPE,1,20))  9999,10,9999
 10   IRECNO = 0
      CALL OUDATA(ITAPE, IRECNO, 1, DUMMY, 1)
      IF (NBAKUP)  30,20,20
 20   CALL FSKIP(ITAPE, -NBAKUP-1)
      GO TO 9999
 30   REWIND ITAPE
```

```
************************
*    TRMINO            *
************************
(PAGE  2)

      9999 RETURN
           END
```

****
*   TRMINO   *
****
(PAGE 2)

```
************************
*     (TSH)            *
************************
 REFER TO
     REREAD


************************
*     (TSHM)           *
************************
 REFER TO
     REREAD
```


```
************************
*     (TSH)            *
************************
 REFER TO
     REREAD


************************
*     (TSHM)           *
************************
 REFER TO
     REREAD
```

```
*       UNPAKN (SUBROUTINE)               9/9/64   LAST CARD IN DECK IS NO. 0149
*       FAP
*UNPAKN
       COUNT   140
       LBL     UNPAKN
       ENTRY   UNPAKN  (N,LD,D,SCALE)
*
*                    ----ABSTRACT----
*
* TITLE - UNPAKN
*       UNPACK AND RESCALE A PACKED DATA VECTOR
*
*             UNPAKN UNPACKS A VECTOR (SUCH AS IS PACKED BY PAKN) AND
*             FLOATS AND SCALES THE VALUES.
*
* LANGUAGE   - FAP, SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090  (MAIN FRAME ONLY)
* STORAGE    - 78 REGISTERS
* SPEED      - TIME IS LENGTH OF UNPACKED VECTOR TIMES 52 MACHINE CYCLES.
* AUTHOR     - J.F. CLAERBOUT,    JULY, 1962
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL UNPAKN(N,LD,D,SCALE)
*
* INPUTS
*
*    N          IS THE NUMBER OF POINTS IN A PACKED REGISTER.
*               MUST BE GRTHN=1  LSTHN=18
*               IF =1 THE DATA IS UNCHANGED.
*               IS FORTRAN II INTEGER
*
*    D(I)       I=1...(LD+N-1)/N)  IS THE PACKED DATA.
*
*    LD         IS THE NUMBER OF UNPACKED DATA POINTS.
*               IS FORTRAN II INTEGER.
*
*    SCALE      IS A FLOATING POINT SCALING BY WHICH THE UNPACKED
*                 DATA IS DIVIDED.
*
* OUTPUTS
*
*    D(I)       I=1...LD    IS THE FLOATING POINT UNPACKED DATA.
*
* EXAMPLES
*
* 1. INPUTS  - D(1...6) = 1.,4.,8.,-7.,5.,2.  LD=6  N=1  SCALE = 1.
*    OUTPUTS - D(1...6) = 1.,4.,8.,-7.,5.,2.
*
* 2. INPUTS  - D(1...3) = OCT 200000040000,  737777377777,  100000237777
*               LD=6  N=2  SCALE=16383.875
*    OUTPUTS - D(1...6) = 1.,4.,8.,-7.,5.,2.
*
* 3. INPUTS  - D(1...2) = OCT 237567720020,  000000000040
*               LD = 6   N = 5   SCALE = 7.875
*    OUTPUTS - D(1...6) = 1.02, 4.06, 8.00,-6.98, 4.95, 2.03
*
* 4. INPUTS  - D(1) = OCT 002117275004  LD=6  N=7  SCALE=1.875
*    OUTPUTS - D(1...6) = 1.07, 4.27, 8.00,-6.93, 4.80, 2.13
*
* 5. INPUTS  - D(1) = OCT 000000000724  LD=6  N=18  SCALE=0.125
*    OUTPUTS - D(1...6) = 0., 8., 8.,-8., 8., 0.
*
*
* PROGRAM FOLLOWS BELOW
*
       HTR     0
       BCI     1,UNPAKN
UNPAKN SXA     SV1,1
       SXA     SV2,2
       SXD     UNPAKN-2,4
```

```
         CLA*     1,4
         ARS      18
         STO      N
         SUB      =1
         TZE      5,4
         CLA*     2,4
         ARS      18
         STO      L
         CLA      3,4
         ADD      =1
         STA      D10
         STA      D11
         STA      D12
         STA      D13
*        SET UP TO JUMP IN LOOP IN PROPER SPOT
*        FOR EASIEST PROOFREADING, READ UNPACKING LOOP FIRST.
         LXA      L,2          C(L)=NO. DATA PTS.,XR2 IS SET FOR LOOP
         CLA      =0
         LDQ      =36
         DVP      N
         STQ      NB           C(NB)=BITS PER PACKED WORD
         XCA
         SUB      =1     NUMBER OF BITS PER PACKED WORD EXCLUDING SIGNBIT
         STA      LLS
         CLA      L            COMPUTE NUMBER OF PACKED REGISTERS
         ADD      N
         SUB      =1
         XCA
         CLA      =0
         DVP      N
         STQ      M            C(M)=NUMBER OF PACKED REGISTERS
         LXA      M,4          SETS XR4 PROPERLY FOR LOOP
         CLA      =0    WHAT IS REMAINDER
         LDQ      L
         DVH      N
         TZE      NEXT   IF ZERO, LOCP CAN BE ENTERED NOW AT THE BEGINNING.
         PAX      ,1     SET REMAINDER TO XR1 = WORDS IN LAST REG.
         SSM
         ADD      N
         XCA             (N-WORDS LEFT IN LAST PACKED REGISTER)
         CLA      =0
         MPY      NB     (N-WORDS LEFT IN REGISTER)*(BITS PER WORD)
         XCA
         STA      RQL
         XEC      D10          DO LDQ INSTR.
 RQL     RQL      **           SHIFT OUT MEANINGLESS INFO.
         TRA      RESET
*        BEGIN    UNPACKING LOOP
 NEXT    LXA      N,1          N=NUMBER PACKED PER REGISTER
 D10     LDQ      **,4         **=DATA+1, GET NEW PACKED REGISTER
 RESET   CLA      =0
  LLS    LLS      **           **=NO. BITS PER WORD LESS SIGNBIT
         RQL      1            GET RID OF OLD SIGN BIT
 D11     STO      **,2         **=DATA+1, XR2 HAS UNPACKED WORD INDEX
         TXI      *+1,2,-1     INDEX WORD STORAGE COUNT
         TIX      RESET,1,1    CONTINUE UNPACKING THIS WORD
         TIX      NEXT,4,1     GET NEXT PACKED WORD
*        END  UNPACKING  LOOP
*        FLOAT AND SCALE
 SV4     LXD      UNPAKN-2,4
         LXA      L,1          C(L)=TOTAL NUMBER OF UNPACKED WORDS
 D12     CLA      **,1         ** = DATA+1
         ORA      =0233000000000
         FAD      =0233000000000
         FDP*     4,4
 D13     STQ      **,1         ** = DATA+1
         TIX      D12,1,1
 SV1     AXT      **,1
 SV2     AXT      **,2
         TRA      5,4
 N       PZE                   NUMBER OF NUMBERS PACKED IN ONE REGISTER
 L       PZE                   TOTAL NUMBER OF UNPACKED REGISTERS
 M       PZE                   TOTAL NUMBER OF PACKED REGISTERS
 NB      PZE                   NUMBER OF BITS PER PACKED WORD
         END
```

```
*      VARARG (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0131
*      FAP
*VARARG
       COUNT    130
       LBL      VARARG
       ENTRY    VARARG (LOCS)
*
*                    ----ABSTRACT----
*
* TITLE - VARARG
*       ENABLE FORTRAN VARIABLE LENGTH CALLING SEQUENCES
*
*             VARARG IS USED IN CONJUNCTION WITH FORTRAN II SUBROUTINES
*             TO ENABLE THEM TO HAVE VARIABLE LENGTH CALLING SEQUENCES.
*
* LANGUAGE  - FAP SUBROUTINE  (FORTRAN II COMPATIBLE)
* EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE   - 44 REGISTERS
* SPEED     -
* AUTHOR    - J.F. CLAERBOUT
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     CALL VARARG(LOCS)
*                 THIS MUST BE THE FIRST STATEMENT IN THE FORTRAN
*                 SUBROUTINE AND SHOULD BE FOLLOWED AS CLOSELY AS
*                 POSSIBLE BY THE RETURN STATEMENT.
*
* OUTPUTS
*
*    LOCS(I)   I=1...N+1  N=NUMBER OF ARGUMENTS IN CALLING STATEMENT
*              CONTAIN THE ADDRESSES OF THE CONTENTS OF THE ARGUMENTS.
*                I.E.
*                    LOCS(1) = XLOCF(ARG1)
*                    LOCS(2) = XLOCF(ARG2)
*                    .
*                    .
*                    .
*                    LOCS(N) = XLOCF(ARGN)
*                    LOCS(N+1) = 0
*
*              ARE FORTRAN II INTEGERS.
*
*              THE CODING FOR THE RETURN TO THE CALLING PROGRAM IS
*                ALTERED SO THAT THE SUBROUTINE RETURNS TO THE PROPER
*                POSITION.  THIS ALTERATION OCCURS ONLY FOR THE RETURN
*                STATEMENT IMMEDIATELY FOLLOWING THE CALL VARARG
*                STATEMENT.
*
* EXAMPLES
*
*    USAGE   -  ASSUME A SUBROUTINE WITH THE FOLLOWING FORM
*                    SUBROUTINE XXXXXX (ARG1,...,ARGK)
*                    DIMENSION LOCS(N+1)
*                    CALL VARARG (LOCS)
*                    GO TO 20
*               10   RETURN
*               20   CONTINUE
*              C
*              C     THE REST OF THE SUBROUTINE IS INSERTED HERE
*              C
*                    GO TO 10
*                    END
*
* 1. INPUTS  - XLOCF(ARG1) = 32561
*              XLOCF(ARG2) = 32560
*              .
*              .
*              .
*              XLOCF(ARG10) = 32552
*    USAGE   -       CALL XXXXXX (ARG1,ARG2,...,ARG10)
```

```
     *     OUTPUTS - INSIDE THE SUBROUTINE XXXXXX LOCS(1)=32561,
     *                                             .
     *                                             .
     *                                             LOCS(10)=32552
     *              NORMAL RETURN
     *
            HTR     0
            BCI     1,VARARG
     VARARG SXD     *-2,4
            SXA     SV1,1
            CLA     1,4
            ADD     =1
            STA     LOCS1
            AXT     1,1           XR1 WILL REFER TO THE VECTOR  LOCS
     *      NEED TO FIND PROG. WHICH CALLED XXXXXX
     *      LOOK FOR SXD INSTRUCTION IN XXXXXX
      A     CAL     -1,4
            ANA     =0777777700000          CHECKS XR4 ALSO
            LAS     SXD4
            TXI     A,4,1
            TRA     *+2
            TXI     A,4,1
     *      HAVE GOT SXD, NOW RESET XR4 TO LOOK AT XXXXXX S CALLING PROG
            CLA     -1,4
            STA     *+1
            LXD     **,4
     *      NOW     LOOK FOR TAGLESS TSX INSTRUCTIONS
      B     CAL     1,4
            ANA     =0777777700000
            SUB     TSX
            TNZ     C
     *      SET ADDRESSES IN LOCS VECTOR
            CLA     1,4
            ALS     18
      LOCS1 STO     **,1
            TXI     *+1,1,1
            TXI     B,4,-1        CONTINUE SCAN
     *      END OF CALLING SEQUENCE, LOCS ALMOST ALL SET UP
      C     TOV     *+1
            STZ*    LOCS1         LOCS ALL SET UP
     *      SET UP RETURN STATEMENT IN XXXXXX
            LXD     VARARG-2,4    LOOK FOR
      D     CAL     0,4           TRA   X,4
            ANA     =0777777700000
            SUB     TRA4
            TZE     *+2
            TXI     D,4,-1
            PXA     ,1
            STA     0,4
     *      RETURN TO XXXXXX
      SV4   LXD     VARARG-2,4
      SV1   AXT     **,1
            TRA     2,4
      SXD4  SXD     0,4
      TSX   TSX
      TRA4  TRA     0,4
            END
```

```
*       VDOTV (SUBROUTINE)                9/4/64   LAST CARD IN DECK IS NO. 0120
*       FAP
*VDOTV
        COUNT   150
        LBL     VDOTV
        ENTRY   VDOTV (X, Y, LXY, DVSR, XDYODV)
*
*
*                      ----ABSTRACT----
*
* TITLE - VDOTV
*         DOT PRODUCT OF TWO VECTORS WITH DIVISION BY CONSTANT
*
*               VDOTV  COMPUTES
*
*                                1     LXY
*                  XDYODV  =  ------   SUM  X(I)*Y(I)
*                              DVSR    I=1
*
*               WHERE  LXY,  DVSR,  X(1...LXY)  AND  Y(1...LXY)  ARE
*               INPUTS, EXCEPT THAT IF THE MAGNITUDE OF  DVSR  IS ZERO,
*               IT IS SET EQUAL TO THE SUM, AND  XDYODV  IS SET = 1.0 .
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 25 REGISTERS
* SPEED      - TAKES  47 + 23.4*LXY  MACHINE CYCLES ON THE 7090
* AUTHOR     - S.M. SIMPSON,   JULY 1964
*
*
*                      ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*        AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*      CALL VDOTV(X, Y, LXY, DVSR, XDYODV)
*
*
* INPUTS
*
*    X(I)       I=1...LXY  IS FLOATING POINT.
*
*    Y(I)       I=1...LXY  IS FLOATING POINT.  EQUIVALENCE  (X,Y)  IS
*                  PERMITTED AS IS ANY OTHER TYPE OF OVERLAP.
*
*    LXY        MUST EXCEED ZERO.
*
*    DVSR       IS FLOATING POINT, OR ZERO.
*
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF  LXY  IS ILLEGAL.
*
*    DVSR       EQUALS ITS INPUT VALUE UNLESS THAT WERE ZERO IN WHICH
*                  CASE IT EQUALS  SUM  (FROM I=1 TO LXY)  OF  X(I)*Y(I).
*
*    XDYODV     HAS VALUE GIVEN BY THE EXPRESSION IN THE ABSTRACT, EXCEPT
*                  THAT IT HAS VALUE  1.0  IF THE INPUT VALUE OF  DVSR
*                  WERE ZERO.
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 1.,2.,3.  Y(1...3) = -1.,2.,-3.
*              XDY(1...5) = -99.,-99.,...,-99.  DVSR = 2.0
*    USAGE   -       DO 10  I=1,5
*                    LXY = I-2
*              10    CALL VDOTV(X, Y, LXY, DVSR, XDY(I))
*    OUTPUTS - XDY(1...5) = -99.,-99.,-.5,1.5,-3.0
*
* 2. INPUTS  - X(1...4) = 1.,2.,3.,4.     SQRSUM = 0.0
*    USAGE   -       DO 10  I=1,4
*              10    CALL VDOTV(X, X(I), 5-I, SQRSUM, ACOR(I))
```

```
*    OUTPUTS - SQRSUM = 30.0  ACOR(1...4) = 1.0,0.66667,0.36667,0.13333
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       BCI     1,VDOTV
*
* ONLY ENTRY.  VDOTV(X, Y, LXY, DVSR, XDYODV)
*
 VDOTV SXA     LEAVE,1
 K1    CLA     1,4              A(X)
       ADD     K1               A(X)+1
       STA     LDQ
       CLA     2,4              A(Y)
       ADD     K1               A(Y)+1
       STA     FMP
*
* CHECK OUT LENGTH
*
       CLA*    3,4              LXY
       TMI     LEAVE
       PDX     0,1
       TXL     LEAVE,1,0
       STZ     TEMP
*
* LOOP
*
 LDQ   LDQ     **,1             ** = A(X)+1
 FMP   FMP     **,1             ** = A(Y)+1
       FAD     TEMP
       STO     TEMP
       TIX     LDQ,1,1
*
* NORMALIZE, IF  DVSR  NON-ZERO, AND STORE
*
       NZT*    4,4              DVSR
       STO*    4,4
       FDP*    4,4
       STQ*    5,4              XDYODV
 LEAVE AXT     **,1             ** = XR1 INITIAL
       TRA     6,4
*
* TEMPORARY
*
 TEMP  PZE     **,**,**
       END
```

```
*       VDVBYV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0089
*       FAP
*VDVBYV
        COUNT   100
        LBL     VDVBYV
        ENTRY   VDVBYV (X,Y,LXY,XDVBYY)
*
*                    ----ABSTRACT----
*
* TITLE - VDVBYV
*          DIVIDE ELEMENTS OF ONE VECTOR BY THOSE OF ANOTHER
*
*               VDVBYV DIVIDES EACH ELEMENT OF A FLOATING VECTOR BY THOSE
*               OF ANOTHER, MAKING NO TEST FOR ZERO DIVISORS.  OUTPUT
*               MAY REPLACE EITHER INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 22 REGISTERS
* SPEED      -       7090   709
*               33 + (19  OR  24)*LXY MACHINE CYCLES, LXY = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL VDVBYV(X,Y,LXY,XDVBYY)
*
* INPUTS
*
*    X(I)       I=1...LXY  IS A FLOATING VECTOR
*
*    Y(I)       I=1...LXY  IS A FLOATING VECTOR, NONE OF WHOSE ELEMENTS
*                            = 0.0
*
*    LXY        SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LXY LSTHN 1
*
*    XDVBYY(I) I=1...LXY  IS  XDVBYY(I) = X(I)/Y(I) (USING FDP
*                  INSTRUCTION)
*
*               EQUIVALENCE(XDVBYY,X  OR  Y)  IS PERMITTED.
*
*               DIVISION BY ZERO TURNS ON THE DIVIDE CHECK INDICATOR BUT
*               VDVBYV  DOESN'T TEST THIS OR STOP THE PROGRAM.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 2.,4.,6.  Y(1...3) = 1.,2.,3.  Z=0.0
*    USAGE   -        CALL VDVBYV (X,Y,3,W)
*                     CALL VDVBYV (X,Y,1,U)
*                     CALL VDVBYV (X,Y,3,X)
*                     CALL VDVBYV (X,Y,0,Z)
*    OUTPUTS - W(1...3) = 2.,2.,2.  U(1) = 2.
*               X(1...3) = 2.,2.,2.  Z=0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
        HTR     0             XR4
        BCI     1,VDVBYV
* ONLY ENTRY.  VDVBYV(X,Y,LXY,XDVBYY)
VDVBYV SXD      VDVBYV-2,4
 K1     CLA     1,4
        ADD     K1            A(X)+1
        STA     GET
        CLA     2,4
        ADD     K1            A(Y)+1
        STA     DIV
        CLA     4,4
```

```
        ADD     K1          A(XDVBYY)+1
        STA     STORE
        CLA*    3,4         LXY
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
* DIVISION LOOP
 GET    CLA     **,4        **=A(X)+1
 DIV    FDP     **,4        **=A(Y)+1
 STORE  STQ     **,4        **=A(XDVBYY)+1
        TIX     GET,4,1
* EXIT
 LEAVE  LXD     VDVBYV-2,4
        TRA     5,4
        END
```

```
*     VECOUT  (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0090
*     LABEL
CVECOUT
      SUBROUTINE VECOUT(ITAPE,FMT,X,ILO,IHI)
C
C                        ----ABSTRACT----
C
C TITLE - VECOUT
C        OFFLINE VECTOR OUTPUT WITH NORMAL OR LITERAL FORMAT
C
C              VECOUT OUTPUTS A VECTOR RANGE ON A GIVEN TAPE UNIT
C              ACCORDING TO A GIVEN FORMAT VECTOR.  THE FORMAT VECTOR IS
C              EITHER OF THE ORDINARY FORM OR MAY APPEAR AS LITERAL
C              HOLLERITH, STRIPPED OF THE INITIAL AND TERMINAL
C              PARENTHESES, IN THE CALLING SEQUENCE.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS 1 TAPE UNIT)
C STORAGE    - 66 REGISTERS
C SPEED      -
C AUTHOR     - S.M. SIMPSON, SEPTEMBER 1963
C
C                        ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - FNDFMT, RPLFMT
C       AND FORTRAN SYSTEM ROUTINES - (STH), (FIL)
C
C FORTRAN USAGE
C     CALL VECOUT(ITAPE,FMT,X,ILO,IHI)
C
C INPUTS
C
C    ITAPE     IS LOGICAL TAPE NUMBER.  (NOT CHECKED FOR LEGALITY)
C
C    FMT(I)    I=1,2,...  IS AN ORDINARY FORMAT (FIRST CHARACTER IS ()
C                 OR
C              I=1,0,-1,...  IS A FORMAT STRIPPED OF ITS ENCLOSING
C                            PARENTHESES AND TERMINATED BY AN ALL-ONES
C                            FENCE.  ITS FIRST CHARACTER MAY NOT BE A (,
C                            AND ITS SECOND CHARACTER MUST NOT BE A ).
C                            IF THIS IS NEEDED, INSERT LEADING SPACES.
C
C    X(I)      I=ILO...IHI  IS THE VECTOR TO BE PRINTED
C
C    ILO       SHOULD EXCEED 0 (NOT CHECKED)
C
C    IHI       SHOULD BE GRTHN= ILO  (NOT CHECKED)
C
C OUTPUTS      FUNCTION IS EQUIVALENT TO THE FORTRAN PROGRAM BELOW
C
C                    WRITE OUTPUT TAPE ITAPE,10,(X(I),I=ILO,IHI)
C               10   FORMAT(FMT)
C
C EXAMPLES
C
C 1. WITH LITERAL HOLLERITH (WITH REPETITION TO CHECK REVERSAL
C                            SCHEME USED)
C    INPUTS  - X(1...5)=1.,2.,3.,4.,5.
C
C    USAGE   -       CALL VECOUT(2,21H12H X(2...5) = ,4F5.1,X,2,5)
C                    DO 10  I=1,3
C               10   CALL VECOUT(2,15H8H X(1) = ,F5.1,X,1,1)
C
C    OUTPUTS - (PRINTED OFF-LINE FROM LOGICAL 2)
C              X(2...5) =    2.0  3.0  4.0  5.0
C              X(1) =   1.0
C              X(1) =   1.0
C              X(1) =   1.0
C
C 2. WITH ORDINARY FORMAT
C    INPUTS  - FMT(1...3) = 17H(8H X(1) = ,F5.1)       X(1)=1.
C
C    USAGE   -       CALL VECOUT(2,FMT,X,1,1)
C
C    OUTPUTS - (PRINTED OFF-LINE FROM LOGICAL 2)
```

```
C              X(1) =    1.0
C
C
C PROGRAM FOLLOWS BELOW
C
      DIMENSION COM(2)
      COMMON COM
      CALL FNDFMT(FMT,IXCFMT)
      CALL RPLFMT(COM,COM(IXCFMT))
      GO TO 20
 10   CALL RPLFMT(COM(IXCFMT),COM)
      GO TO 9999
 20   WRITE OUTPUT TAPE ITAPE ,COM,(X(I),I=ILO,IHI)
      GO TO 10
 9999 RETURN
      END
```

```
************************
*    VINDEX            *
************************
 REFER TO
     INDEX
```



```
************************
*    VINDEX            *
************************
 REFER TO
     INDEX
```



```
************************
*    VMNUSV            *
************************
 REFER TO
     VPLUSV
```



```
************************
*    VMNUSV            *
************************
 REFER TO
     VPLUSV
```

```
*     VOUT  (SUBROUTINE)                 9/29/64   LAST CARD IN DECK IS NO. 0110
*     LABEL
CVOUT
      SUBROUTINE VOUT(ITAPE,NSPACE,X,XNAME,XFMT,ILO,IHI)
C
C                         ----ABSTRACT----
C
C TITLE - VOUT
C        OUTPUT NAMED VECTOR BY NORMAL OR LITERAL FORMAT WITH SPACING
C
C              VOUT WRITES OUT A VECTOR RANGE, X(ILO...IHI), ON A
C              SPECIFIED TAPE UNIT ACCORDING TO A SPECIFIED FORMAT,
C              WITH LABELING AND INITIAL SPACING (OR PAGE RESTORE).
C              THE FORMAT IS SPECIFIED EITHER AS A NORMAL FORMAT VECTOR
C              OR AS LITERAL HOLLERITH IN THE CALLING SEQUENCE.
C
C LANGUAGE   - FORTRAN-II SUBROUTINE
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
C STORAGE    - 104 REGISTERS
C SPEED      -
C AUTHOR     - S.M. SIMPSON JR,   OCTOBER 1963
C
C                     ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - CARIGE, HRADJ, VECOUT
C       AND FORTRAN SYSTEM ROUTINES - (STH), (FIL)
C
C FORTRAN USAGE
C     CALL VOUT(ITAPE,NSPACE,X,XNAME,XFMT,ILO,IHI)
C
C INPUTS       DEFINE A NORMLIT FORMAT VECTOR AS EITHER
C                 A) A NORMAL FORMAT VECTOR,
C              OR B) LITERAL HOLLERITH IN A CALLING SEQUENCE WHOSE
C                 CHARACTERS (READING CONTINUOUSLY FROM LEFT TO RIGHT)
C                 ARE THE DESIRED FORMAT STRIPPED OF THE ENCLOSING
C                 PARENTHESES.  THE FIRST AND SECOND CHARACTERS MUST
C                 NOT BE QUOTE ( UNQUOTE OR QUOTE ) UNQUOTE
C                 RESPECTIVELY.  (TWO BLANKS FOLLOWED BY ( WOULD BE OK.)
C
C    ITAPE     IS DESIRED LOGICAL TAPE NO.
C
C    NSPACE    IS DESIRED NO. OF SPACES (MAY BE ZERO) BEFORE ANY OUTPUT.
C                 IF NEGATIVE, AN INITIAL PAGE RESTORE OCCURS.
C
C    X(I)      I=ILO...IHI IS THE VECTOR RANGE TO BE PRINTED.  MAY BE
C                 ANY MODE.
C
C    XNAME     IS THE NAME OF VECTOR X, IN FORMAT(A6) OR (A5) OR...(A1).
C
C    XFMT(I)   IS A NORMLIT FORMAT VECTOR CONTROLLING THE OUTPUT
C                 OF X(I).
C
C    ILO       SHOULD EXCEED ZERO (NOT CHECKED).
C
C    IHI       SHOULD BE GRTHN = ILO (NOT CHECKED).
C
C OUTPUTS      1. NSPACE SPACES OR A PAGE RESTORE OCCURS
C              2. A HEADING LINE OF THE GENERAL FORM
C                    XNAME (    ILO ,ILO+1 , . . . ,  IHI )  =
C                 APPEARS, IF IHI EXCEEDS ILO.  IF IHI=ILO THE HEADING
C                 IS
C                    XNAME (    ILO )  =
C              3. THE VALUES X(ILO...IHI) ARE THEN PRINTED ACCORDING TO
C                 XFMT.
C
C EXAMPLES
C
C 1. WITH NORMAL FORMATS AND NAMES
C    INPUTS  - X(1...14) = 1.,2.,...,14.   XNAME = 1HX,
C              XFMT(1...2) = 11H(10X,5F8.1)   Y = 7.   YNAME = 1HY
C              YFMT(1...2) = 10H(20X,F4.1)
C    USAGE   -        CALL VOUT(2,3,X,XNAME,XFMT,3,14)
C                     CALL VOUT(2,3,Y,YNAME,YFMT,1,1)
```

```
C     OUTPUTS - THE FOLLOWING 12 LINES
C
C
C
C                       X (      3 ,     4 , . . . ,    14 )  =
C                                  3.0       4.0       5.0       6.0       7.0
C                                  8.0       9.0      10.0      11.0      12.0
C                                 13.0      14.0
C
C
C
C                       Y (      1 )  =
C                                           7.0
C                  WILL BE PRINTED OFF LINE FROM LOGICAL TAPE 2 (UNDER
C                  PROGRAM CONTROL)
C
C 2. SAME DATA BUT WITH LITERAL ARGUMENTS
C     INPUTS  - X(I...I4) AND Y SAME AS EXAMPLE 1
C     USAGE   -       CALL VOUT(2,3,X,1HX,9H10X,5F8.1,3,14)
C                     CALL VOUT(2,3,Y,1HY,8H20X,F4.1,1,1)
C     OUTPUTS - IDENTICAL TO THOSE OF EXAMPLE 1.
C
C PROGRAM FOLLOWS BELOW
C
      CALL CARIGE(ITAPE,NSPACE)
      XNMADJ=HRADJF(XNAME)
      IF (IHI-ILO)   9999,10,20
C SINGLE ELEMENT OUTPUT
 10   WRITE OUTPUT TAPE ITAPE,15,XNMADJ,ILO
 15   FORMAT(1H ,A6,2H (,I5,5H )  =)
      GO TO 30
C MULTIPLE ELEMENT OUTPUT
 20   ILOP1=ILO+1
      WRITE OUTPUT TAPE ITAPE,25,XNMADJ,ILO,ILOP1,IHI
 25   FORMAT(1H ,A6,2H (,I5,2H ,,I5,10H , . . . ,,I5,5H )  =)
 30   CALL VECOUT(ITAPE,XFMT,X,ILO,IHI)
 9999 RETURN
      END
```

```
*      VPLUSV (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0126
*      FAP
*VPLUSV
       COUNT   150
       LBL     VPLUSV
       ENTRY   VPLUSV ( X, Y,LXY,XPLUSY)
       ENTRY   XVPLSV (IX,IY,LXY,IXPLSY)
       ENTRY   VMNUSV ( X, Y,LXY,XMNUSY)
       ENTRY   XVMNSV (IX,IY,LXY,IXMNSY)
*
*                   ----ABSTRACT----
*
* TITLE - VPLUSV  WITH SECONDARY ENTRIES XVPLSV, VMNUSV AND XVMNSV
*        ADD OR SUBTRACT TWO FLOATING OR FIXED VECTORS
*
*              VPLUSV ADDS TWO FLOATING VECTOR
*              XVPLSV ADDS TWO FIXED VECTORS
*              VMNUSV SUBTRACTS TWO FLOATING VECTORS
*              XVMNSV SUBTRACTS TWO FIXED VECTORS
*
*              OUTPUT MAY REPLACE EITHER INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34 REGISTERS
* SPEED      - VPLUSV   37 + 12.4*LXY  MACHINE CYCLES,
*              XVPLSV   39 +  8.0*LXY       LXY = VECTOR LENGTH
*              VMNUSV   39 + 12.4*LXY
*              XVMNSV   39 +  8.0*LXY
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL VPLUSV( X, Y,LXY,XPLUSY)
*          XVPLSV(IX,IY,LXY,IXPLSY)
*          VMNUSV( X, Y,LXY,XMNUSY)
*          XVMNSV(IX,IY,LXY,IXMNSY)
*
* INPUTS
*
*    X(I)       I=1...LXY  IS A FLOATING VECTOR
*
*    Y(I)       I=1...LXY  IS A FLOATING VECTOR
*
*    LXY        SHOULD EXCEED ZERO
*
*    IX(I)      I=1...LXY  IS A FIXED VECTOR
*
*    IY(I)      I=1...LXY  IS A FIXED VECTOR
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LXY LSTHN 1
*
*    XPLUSY(I) I=1...LXY  IS  XPLUSY(I) =   X(I) +  Y(I)
*    IXPLSY(I) I=1...LXY  IS  IXPLSY(I) =  IX(I) + IY(I)
*    XMNUSY(I) I=1...LXY  IS  XMNUSY(I) =   X(I) -  Y(I)
*    IXMNSY(I) I=1...LXY  IS  IXMNSY(I) =  IX(I) - IY(I)
*
*              EQUIVALENCE (XPLUSY OR XMNUSY, X OR Y),(IXPLSY OR IXMNSY,
*                           IX OR IY) IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 2.,4.,6.  Y(1...3) = 1.,2.,3.
*              IX(1...3) = 2,4,6  IY(1...3) = 1,2,3  Z=0.0
*    USAGE   -       CALL VPLUSV ( X, Y,3, X1)
*                    CALL XVPLSV (IX,IY,3,IX1)
*                    CALL VMNUSV ( X, Y,3, X2)
*                    CALL XVMNSV (IX,IY,3,IX2)
*                    CALL VPLUSV ( X, Y,3, X)
*                    CALL XVMNSV (IX,IY,1,IY)
*                    CALL VPLUSV ( X, Y,0, Z)
```

```
*    OUTPUTS - X1(1...3) = 3.,6.,9.  IX1(1...3) = 3,6,9
*              X2(1...3) = 1.,2.,3.  IX2(1...3) = 1,2,3
*              X (1...3) = 3.,6.,9.  IY(1) = 1
*              Z = 0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
       BCI     1,VPLUSV
* PRINCIPAL ENTRY.  VPLUSV(X,Y,LXY,XPLUSY)
VPLUSV CLA     FAD
 SETUP STO MODIFY
       SXD     VPLUSV-2,4
 K1    CLA     1,4
       ADD     K1             A(X)+1
       STA     GET
       CLA     2,4
       ADD     K1             A(Y)+1
       STA     MODIFY
       CLA     4,4
       ADD     K1             A(XPLUSY)+1
       STA     STORE
       CLA*    3,4            LXY
       TMI     LEAVE
       PDX     0,4
       TXL     LEAVE,4,0
* LOOP
 GET   CLA     **,4           **=A(X)+1
MODIFY NOP                    =FAD  **,4  ADD  **,4  FSB  **,4  SUB  **,4
*                               **=A(Y)+1
 STORE STO     **,4           **=A(XPLUSY)+1
       TIX     GET,4,1
* EXIT
 LEAVE LXD     VPLUSV-2,4
       TRA     5,4
* SECOND ENTRY.  XVPLSV(IX,IY,LXY,IXPLSY)
XVPLSV CLA     ADD
       TRA     SETUP
* THIRD ENTRY.  VMNUSV(X,Y,LXY,XMNUSY)
VMNUSV CLA     FSB
       TRA     SETUP
* FOURTH ENTRY.  XVMNSV(IX,IY,LXY,IXMNSY)
XVMNSV CLA     SUB
       TRA     SETUP
* CONSTANTS
 FAD   FAD     **,4
 ADD   ADD     **,4
 FSB   FSB     **,4
 SUB   SUB     **,4
       END
```

```
*      VRSOUT (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0137
*      FAP
*VRSOUT
       COUNT   150
       LBL     VRSOUT
       ENTRY   VRSOUT (ITAPE,NSPACE,FMT,SPACE,X1,X2,...,XN)
*
*                    ----ABSTRACT----
*
* TITLE - VRSOUT
*        OUTPUT VARIABLES BY NORMAL OR LITERAL FORMAT
*
*              VRSOUT IS A VARIABLE-LENGTH-CALLING-SEQUENCE PROGRAM
*              WHICH WRITES OUT, ON A SPECIFIED TAPE, A LIST OF VARI-
*              ABLES ACCORDING TO A GIVEN FORMAT, WITH INITIAL SPACING
*              OR PAGE RESTORE.  THE FORMAT MAY BE EITHER  A NORMAL
*              FORMAT VECTOR, OR APPEAR AS LITERAL HOLLERITH IN THE
*              CALLING SEQUENCE.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
* STORAGE    - 47 REGISTERS
* SPEED      -
* AUTHOR     - S.M.SIMPSON JR.,   OCTOBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - CARIGE, VECOUT
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL VRSOUT(ITAPE,NSPACE,FMT,SPACE,X1,X2,...,XN)
*              WHERE THE NO. OF VARIABLES, N, MUST EXCEED 0.
*
* INPUTS       DEFINE A NORMLIT FORMAT VECTOR AS EITHER
*                 A) A NORMAL FORMAT VECTOR,
*              OR B) LITERAL HOLLERITH IN A CALLING SEQUENCE WHOSE
*                 CHARACTERS (READING CONTINUOUSLY FROM LEFT TO RIGHT)
*                 ARE THE DESIRED FORMAT STRIPPED OF THE ENCLOSING
*                 PARENTHESES.  THE FIRST AND SECOND CHARACTERS MUST
*                 NOT BE QUOTE ( UNQUOTE OR QUOTE ) UNQUOTE
*                 RESPECTIVELY.  (TWO BLANKS FOLLOWED BY $ WOULD BE OK.)
*
*    ITAPE     IS LOGICAL TAPE NO. OF DESIRED OUTPUT TAPE.
*
*    NSPACE    IS DESIRED NO. (MAY BE ZERO) OF SPACES BEFORE ANY OUTPUT.
*                 IF NEGATIVE AN INITIAL PAGE RESTORE OCCURS.
*
*    FMT(I)    IS A NORMLIT FORMAT VECTOR CONTROLLING THE OUTPUT OF
*                 X1...XN .
*
*    SPACE(I)  I=1...N MUST BE AVAILABLE AS SCRATCH.
*                 IF N=1, EQUIVALENCE (SPACE,X1) IS PERMITTED.
*
*    X1,X2,...,XN ARE THE N VARIABLES (MODES ARBITRARY) TO BE PRINTED.
*
* OUTPUTS      1. NSPACE SPACES OR A PAGE RESTORE OCCURS.
*              2. X1,X2,...,XN ARE PRINTED ACCORDING TO FORMAT FMT.
*
* EXAMPLES
*
* 1. USING NORMAL FORMAT
*    INPUTS  - X1 = 1.,  IX2 = 2,  IX3 = 3
*              FMT(1...6) = 33H(5H X1 =,F4.1,11H, IX2,IX3 =,2I4)
*    USAGE   -       DIMENSION SPACE(3)
*                    CALL VRSOUT(2,3,FMT,SPACE,X1,IX2,IX3)
*    OUTPUTS - THE FOLLOWING 4 LINES
*
*
*
*              X1 = 1.0, IX2,IX3 =   2   4
*              WILL PRINT OFF-LINE FROM LOGICAL TAPE 2 UNDER PROGRAM
*                 CONTROL
*
* 2. USING LITERAL FORMAT
```

```
*    INPUTS  - X1,IX2,IX3  SAME AS FOR EXAMPLE 1.
*    USAGE   -        CALL VRSOUT(2,3,31H5H X1 =,F4.1,11H, IX2,IX3 =,2I4,
*                 1               SPACE,X1,IX2,IX3)
*    OUTPUTS - IDENTICAL TO THOSE OF EXAMPLE 1.
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS CARIGE, VECOUT
       HTR     0               XR1
       HTR     0               XR4
       BCI     1,VRSOUT
* ONLY ENTRY.  VRSOUT(ITAPE,NSPACE,FMT,SPACE,X1,X2,....,XN)
VRSOUT SXD     VRSOUT-2,4
       SXD     VRSOUT-3,1
       CLA     1,4             A(ITAPE)
       STA     CA1
       STA     VA1
       CLA     2,4             A(NSPACE)
       STA     CA2
       CLA     3,4             A(FMT)
       STA     VA2
       CLA     4,4             A(SPACE)
       STA     STO
       STA     VA3
* LOOP TO PACK UP THE VARIABLES INTO SPACE (1...N)
       AXT     0,1             XR1 COUNTS N
       TXI     *+1,4,-4        (INITIALIZE TO 1,4=X1)
 CAL   CAL     1,4             POSSIBLE TSX X,0
       ANA     AMASK
       LAS     TSXZ
       TRA     *+2
       TRA     GETX
* WHEN NO MORE, SET NO. VARIABLES AND GO OPERATE CARIGE.
       SXD     N,1
       TRA     OPCAR
* STORE EACH X IN THE SPACE VECTOR, COUNT AND CYCLE
 GETX  CLA*    1,4
 STO   STO     **,1            **=A(SPACE)
       TXI     *+1,1,1
       TXI     CAL,4,-1
* GO OPERATE CARIGE
 OPCAR SXA     LEAVE,4
       TSX     $CARIGE,4
 CA1   TSX     **,0            **=A(ITAPE)
 CA2   TSX     **,0            **=A(NSPACE)
* AND THEN VECOUT
       TSX     $VECOUT,4
 VA1   TSX     **,0            **=A(ITAPE)
 VA2   TSX     **,0            **=A(FMT)
 VA3   TSX     **,0            **=A(SPACE)
       TSX     KD1,0           ILO=1
       TSX     N,0             IHI=N
* AND THEN EXIT
 LEAVE AXT     **,4            **=XR4 TEMP
       LXD     VRSOUT-3,1
       TRA     1,4
* CONSTANTS, TEMPORARIES
 KD1   PZE     0,0,1
 AMASK OCT     777777700000
 TSXZ  TSX     0,0
 N     PZE     0,0,**          **=NO. OF VARIABLES = N
       END
```

```
*      VSOUT  (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0124
*      FAP
*VSOUT
       COUNT   150
       LBL     VSOUT
       ENTRY   VSOUT (ITAPE,NSPACE,X1,X1NAME,X1FMT,ILO1,IHI1,  X2,
*                     X2NAME,X2FMT,ILO2,IHI2,......,XN,XNNAME,XNFMT,
*                     ILON,IHIN)
*
*                             ----ABSTRACT----
*
* TITLE  -  VSOUT
*      OUTPUT NAMED VECTORS BY NORMAL OR LITERAL FORMATS WITH SPACING
*
*               VSOUT IS A VARIABLE-LENGTH-CALLING-SEQUENCE PROGRAM WHICH
*               WRITES OUT, ON A SPECIFIED TAPE UNIT, A LIST OF VECTOR
*               RANGES, EACH RANGE ACCORDING TO A GIVEN FORMAT, WITH
*               LABELING AND INITIAL SPACING OR PAGE RESTORING BEFORE
*               EACH VECTOR.  THE FORMATS ARE EITHER NORMAL FORMAT
*               VECTORS OR MAY APPEAR AS LITERAL HOLLERITH IN THE CALLING
*               SEQUENCE.  ONE CALL OF VSOUT IS EQUIVALENT TO A
*               SUCCESSION OF CALLS OF SUBROUTINE VOUT.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME PLUS ONE TAPE UNIT)
* STORAGE    - 37 REGISTERS
* SPEED      -
* AUTHOR     - S.M. SIMPSON JR., OCTOBER 1963
*
*                        ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - VOUT
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL VSOUT(ITAPE,NSPACE,X1,X1NAME,X1FMT,ILO1,IHI1,  X2,X2NAME,
*    1             X2FMT,ILO2,IHI2, ..., XN,XNNAME,XNFMT,ILON,IHIN)
*
*          WHERE THE NO. OF VECTORS, N, MUST EXCEED ZERO.
*
* THE ABOVE IS EQUIVALENT TO A SEQUENCE OF N CALLS OF VOUT.
*     CALL VOUT(ITAPE,NSPACE,X1,X1NAME,X1FMT,ILO1,IHI1)
*     CALL VOUT(ITAPE,NSPACE,X2,X2NAME,X2FMT,ILO2,IHI2)
*          ETC
*     CALL VOUT(ITAPE,NSPACE,XN,XNNAME,XNFMT,ILON,IHIN)
*
* SEE WRITEUP OF SUBROUTINE VOUT FOR INPUT-OUTPUT DETAILS.
*
* EXAMPLES
*
* 1. WITH NORMAL FORMATS AND NAMES
*    INPUTS  - X(1...14) = 1.,2.,...,14.   XNAME = 1HX,
*              XFMT(1...2) = 11H(10X,5F8.1)   Y = 7.   YNAME = 1HY
*              YFMT(1...2) = 10H(20X,F4.1)
*    USAGE   -        CALL VSOUT(2,3,X,XNAME,XFMT,3,14, Y,YNAME,YFMT,1,1)
*    OUTPUTS - THE FOLLOWING 12 LINES
*
*
*
*                  X (    3 ,    4 , . . . ,   14 )  =
*                            3.0      4.0      5.0      6.0      7.0
*                            8.0      9.0     10.0     11.0     12.0
*                           13.0     14.0
*
*
*
*                  Y (    1 )  =
*                                     7.0
*               WILL BE PRINTED OFF LINE FROM LOGICAL TAPE 2 (UNDER
*               PROGRAM CONTROL)
*
* 2. SAME DATA BUT WITH LITERAL ARGUMENTS
*    INPUTS  - X(1...14) AND Y SAME AS EXAMPLE 1.
*    USAGE   -        CALL VSOUT(2,3,X,1HX,9H10X,5F8.1,3,14, Y,1HY,
*                              8H20X,F4.1,1,1)
```

```
      *    OUTPUTS - IDENTICAL TO THOSE OF EXAMPLE 1
      *
      * PROGRAM FOLLOWS BELOW
      *
      *
      * TRANSFER VECTOR CONTAINS VOUT
             HTR     0              XR4
             BCI     1,VSOUT
      * ONLY ENTRY.  VSOUT(ITAPE,NSPACE,X1,XNAM1,XFMT1,ILO1,IHI1,X2,XNAM2,
      *                     XFMT2,ILO2,IHI2,...,XN,XNAMN,XFMTN,ILON,IHIN)
       VSOUT SXD     VSOUT-2,4
             CLA     1,4            A(ITAPE)
             STA     TSX1
             CLA     2,4            A(NSPACE)
             STA     TSX2
             TXI     *+1,4,-2
      * START LOOP
       CAL   CAL     1,4            A(XK)  K=1,2,...
             STA     TSX3
             ANA     AMASK
             LAS     TSXZ
             TRA     *+2
             TRA     CLA
      * EXIT AT END OF GROUPS OF 5
             TRA     1,4
      * COMPLETE THE CALLING SEQUENCE
       CLA   CLA     2,4            A(XNAMK)
             STA     TSX4
             CLA     3,4            A(XFMTK)
             STA     TSX5
             CLA     4,4            A(ILOK)
             STA     TSX6
             CLA     5,4            A(IHIK)
             STA     TSX7
      * GO OPERATE VOUT
             SXA     SVXR4,4
             TSX     $VOUT,4
       TSX1  TSX     **,0           **=A(ITAPE)
       TSX2  TSX     **,0           **=A(NSPACE)
       TSX3  TSX     **,0           **=A(XK)  K=1,2,...
       TSX4  TSX     **,0           **=A(XNAMK)
       TSX5  TSX     **,0           **=A(XFMTK)
       TSX6  TSX     **,0           **=A(ILOK)
       TSX7  TSX     **,0           **=A(IHIK)
       SVXR4 AXT     **,4
             TXI     CAL,4,-5
      * CONSTANTS
       AMASK OCT     777777700000
       TSXZ  TSX     0,0
             END
```

```
*      VTIMSV  (SUBROUTINE)             9/29/64   LAST CARD IN DECK IS NO. 0111
*      FAP
*VTIMSV
       COUNT    100
       LBL      VTIMSV
       ENTRY    VTIMSV ( X, Y,LXY,XTIMSY)
       ENTRY    XVTMSV (IX,IY,LXY,IXTMSY)
*
*                    ----ABSTRACT----
*
* TITLE - VTIMSV  WITH SECONDARY ENTRY XVTMSV
*         MULTIPLY ELEMENTS OF TWO VECTORS FIXED OR FLOATING
*
*               VTIMSV MULTIPLIES ELEMENTS OF TWO FLOATING VECTORS
*               XVTMSV MULTIPLIES ELEMENTS OF TWO FIXED VECTORS
*
*               OUTPUT MAY REPLACE EITHER INPUT VECTOR
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34 REGISTERS
* SPEED      -               7090     709
*               VTIMSV   41 + (19.0 OR 22.2)*LXY   MACHINE CYCLES,
*               XVTMSV   43 + (20.6 OR 24.8)*LXY      LXY = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL VTIMSV( X, Y,LXY,XTIMSY)
*     CALL XVTMSV(IX,IY,LXY,IXTMSY)
*
* INPUTS
*
*    X(I)       I=1...LXY  IS A FLOATING PT VECTOR
*    Y(I)       I=1...LXY  IS A FLOATING PT VECTOR
*
*    LXY        SHOULD EXCEED ZERO
*
*    IX(I)      I=1...LXY  IS A FIXED VECTOR, FORTRAN-II INTEGERS
*    IY(I)      I=1...LXY  IS A FIXED VECTOR, FORTRAN-II INTEGERS
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LXY LSTHN 1
*
*    XTIMSY     I=1...LXY  IS  XTIMSY(I) =  X(I) *  Y(I)
*    IXTMSY     I=1...LXY  IS  IXTMSY(I) = IX(I) * IY(I)
*
*               DANGER OF FIXED POINT OVERFLOW IS NOT TESTED FOR BY
*               XVTMSV.
*
*               EQUIVALENCE (XTIMSY, X OR Y), (IXTMSY, IX OR IY) IS
*               PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - X(1...3) = 1.,2.,3.  Y(1...3) = 2.,4.,6.
*              IX(1...3) = 1,2,3  IY(1...3) = 2,4,6  Z=0.0
*    USAGE   -       CALL VTIMSV ( X, Y,3,X1)
*                    CALL XVTMSV (IX,IY,3,IX1)
*                    CALL VTIMSV ( X, Y,3, Y)
*                    CALL XVTMSV (IX,IY,1,IX)
*                    CALL VTIMSV ( X, Y,0, Z)
*    OUTPUTS - X1 (1...3) = 2.,8.,18.  IX1(1...3) = 2,8,18
*              Y  (1...3) = 2.,8.,18.  IX(1) = 2
*              Z = 0.0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
       HTR      0             XR4
       BCI      1,VTIMSV
* PRINCIPAL ENTRY.  VTIMSV(X,Y,LXY,XTIMSY)
```

VTIMSV
(PAGE 2)

```
VTIMSV CLA    FMP
       LDQ    NOP
 SETUP STO    MLPLY
       STQ    VARY
       SXD    VTIMSV-2,4
 K1    CLA    1,4
       ADD    K1             A(X)+1
       STA    GET
       CLA    2,4
       ADD    K1             A(Y)+1
       STA    MLPLY
       CLA    4,4
       ADD    K1             A(XTIMSY)+1
       STA    STORE
       CLA*   3,4            LXY
       TMI    LEAVE
       PDX    0,4
       TXL    LEAVE,4,0
* MULTIPLICATION LOOP
 GET   LDQ    **,4           **=A(X)+1
 MLPLY NOP                   =FMP  **,4  OR  MPY  **,4    **=A(Y)+1
 VARY  NOP                   =NOP  OR  ALS  17
 STORE STO    **,4           **=A(XTIMSY)+1
       TIX    GET,4,1
* EXIT
 LEAVE LXD    VTIMSV-2,4
       TRA    5,4
* SECOND ENTRY  XVTMSV(IX,IY,LXY,IXTMSY)
XVTMSV CLA    MPY
       LDQ    ALS
       TRA    SETUP
* CONSTANTS
 FMP   FMP    **,4
 NOP   NOP
 MPY   MPY    **,4
 ALS   ALS    17
       END
```

```
*     WAC (SUBROUTINE)                    9/29/64   LAST CARD IN DECK IS NO. 0082
*     LABEL
CWAC
      SUBROUTINE WAC (LY,Y,LA,A)
C
C                       ----ABSTRACT----
C
C TITLE - WAC
C          WIENER AUTOCORRELATION
C
C              WAC FINDS THE AUTOCORRELATION, A, OF AN ARBITRARY NUMBER
C              OF LAGS, LA, FOR A SERIES, Y, OF LENGTH, LY
C
C                            LY
C                 A(I)  =  SUM Y(J)*Y(J+I-1)    I=1,2,....,LA
C                           J=1
C
C              WHERE WE ASSUME Y TO BE ZERO OUTSIDE THE RANGE FOR
C              WHICH IT IS SPECIFIED.
C
C LANGUAGE  - FORTRAN SUBROUTINE
C EQUIPMENT - 709 OR 7090
C STORAGE   - 107 REGISTERS
C SPEED     -
C AUTHOR    - J.F. CLAERBOUT
C
C                       ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES - NONE
C        AND FORTRAN SYSTEM ROUTINES - NONE
C
C FORTRAN USAGE
C     CALL WAC    (LY,Y,LA,A)
C
C INPUTS
C    Y(I)       I=1...LY IS THE SERIES TO BE AUTOCORRELATED.
C               MUST BE FLOATING POINT.
C
C    LY         IS FORTRAN INTEGER
C               MUST BE GRTHN=1.
C
C    LA         IS LENGTH OF THE DESIRED AUTOCORRELATION.
C               IS FORTRAN INTEGER
C               MUST BE GRTHN=1.
C               MAY BE GRTHN LY.
C
C OUTPUTS
C    A(I)       I=1...LA IS THE AUTOCORRELATION.
C                   A(1) = AUTOCORRELATION AT ZERO LAG.
C                   A(I) = 0. FOR I GRTHN LY.
C
C EXAMPLES
C
C 1. INPUTS  - LY = 3   Y(1...3) = 1.,2.,3.   LA = 1
C
C    OUTPUTS - A(1) = 14.
C
C 2. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LA = 3
C
C    OUTPUTS - A(1...3) = 14., 8., 3.
C
C 3. INPUTS  - SAME AS EXAMPLE 1. EXCEPT LA = 5
C
C    OUTPUTS - A(1...5) = 14., 8., 3., 0., 0.
C
      DIMENSION Y(100),A(100)
      MM=XMINOF (LY,LA)
      DO 20  I=1,MM
      A(I)=0.
      L=LY-I+1
      DO 10  J=1,L
      K=J+I
      A(I)=A(I)+Y(J)*Y(K-1)
   10 CONTINUE
   20 CONTINUE
```

WAC

(PAGE 2)

```
C     IF (LA GRTHN LY), WE MUST FILL SOME ZEROS IN A
      IF (LA-LY) 50,50,30
   30 NP = LY+1
      DO 40 I=NP,LA
      A(I)=0.
   40 CONTINUE
   50 RETURN
      END
```

```
************************
*    WHERE             *
************************
 REFER TO
     LOCATE
```



```
************************
*    WHERE             *
************************
 REFER TO
     LOCATE
```

```
*      WHICH (FUNCTIONS)                  9/4/64   LAST CARD IN DECK IS NO. 0076
*      FAP
*WHICH
       COUNT   100
       LBL     WHICH
       ENTRY   WHICH  F( X1, X2,ZIFX1)
       ENTRY   XWHICH F(IX1,IX2,ZIFIX1)
*
*
*                   ----ABSTRACT----
*
* TITLE - WHICH, WITH SECONDARY ENTRY XWHICH
*        CHOOSE BETWEEN TWO VARIABLES BY A THIRD ONE BEING ZERO
*
*              WHICH IS A FUNCTION WITH VALUE EQUAL TO ITS FIRST
*              ARGUMENT IF ITS THIRD ARGUMENT IS ZERO IN MAGNITUDE.
*              OTHERWISE WHICH HAS VALUE EQUAL TO ITS SECOND ARGUMENT.
*
*              XWHICH IS THE FIXED POINT NAME FOR WHICH.
*
* LANGUAGE   - FAP FUNCTIONS (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 4 REGISTERS
* SPEED      - 5 TO 7 MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, APRIL 1964
*
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     X = WHICHF( X1, X2,ZIFX1)
*     IX=XWHICHF(IX1,IX2,ZIFIX1)
*
*
* INPUTS
*
*    X1,X2      ARE FLOATING POINT.
*
*    IX1,IX2,   ARE FIXED POINT.
*
*    ZIFX1      =0. IF X1 IS TO BE CHOSEN.
*               NOT=0. IF X2 IS TO BE CHOSEN.
*
*    ZIFIX1     =0 IF IX1 IS TO BE CHOSEN.
*               NOT=0 IF IX2 IS TO BE CHOSEN.
*
*
* OUTPUTS
*
*    X OR IX    IS SET AS DESCRIBED IN ABSTRACT.
*
*
* EXAMPLES
*
* 1. USAGES  -         XA = WHICHF(1.,2.,0.)
*                      XB = WHICHF(1.,2.,1.)
*                      XC = WHICHF(1.,2.,-137)
*                      IXA = XWHICHF(1,2,-0)
*                      IXB = XWHICHF(1,2,-.0001)
*                      IXC = XWHICHF(1,2,36)
*    OUTPUTS - XA = 1.0  XB = 2.0  XC = 2.0
*              IXA = 1   IXB = 2   IXC = 2
*
*
* PROGRAM FOLLOWS BELOW
*
* NO TRANSFER VECTOR
*
       BCI     1,WHICH
 WHICH BSS     0
XWHICH ZET     32765          = 77775 OCTAL
       XCA
```

****
*   WHICH   *
****
(PAGE 2)

```
         TRA      1,4                                      0075
         END                                               0076
```

```
*     WLLSFP (SUBROUTINE)              10/6/64   LAST CARD IN DECK IS NO. 0263
*     LABEL
CWLLSFP
      SUBROUTINE WLLSFP (LR,R,G,LA,A,C)
C
C                    ----ABSTRACT----
C
C TITLE - WLLSFP
C        WIENER-LEVINSON LEAST SQUARE ERROR FILTER OR PREDICTOR
C
C             WLLSFP FINDS SOLUTIONS FOR A CLASS OF SIMULTANEOUS
C             EQUATIONS WHICH ARISE IN MANY LEAST SQUARE ERROR FILTERING
C             AND PREDICTION PROBLEMS.  SPECIFICALLY IT SOLVES THE
C             FOLLOWING EQUATIONS FOR AA(I), I=0,1,...,M
C
C                    M
C                   SUM (AA(N)*RR(K-N)) = GG(K)  K=0,1,...,M
C                   N=0
C
C             GIVEN ANY RIGHT HAND SIDE GG(I), I=0,1,...,M
C             AND GIVEN ANY VECTOR RR(I), I=0,1,...,M,M+1
C             WHICH IS SYMMETRIC  (RR(I) = RR(-I))
C             AND FOR WHICH THE (M+1)*(M+1) TOEPLITZ MATRIX
C                   RR(K-N)   K=0,1,...,M
C                             N=0,1,...,M
C             IS POSITIVE DEFINITE.
C
C             IN TIME SERIES PROBLEMS AA(I) IS A SET OF OPTIMUM FILTER
C             COEFFICIENTS, RR(I) IS AN AUTOCORRELATION FUNCTION OR A
C             SUM OF AUTOCORRELATION FUNCTIONS, AND GG(I) IS A CROSS-
C             CORRELATION (OF INPUT WITH DESIRED OUTPUT).
C
C             A SOLUTION IS ACCOMPLISHED BY A RECURSIVE PROCESS GIVEN
C             BY N. LEVINSON IN APPENDIX B OF THE BOOK
C             WIENER, N., 1949, EXTRAPOLATION, INTERPOLATION, AND
C                  SMOOTHING OF STATIONARY TIME SERIES,  JOHN WILEY AND
C                  SONS, INC., NEW YORK, PP 129-139.
C
C             AN ADDITIONAL OUTPUT OF WLLSFP IS AN AUXILIARY SEQUENCE
C             CC(I), I=0,1,...,M (DEFINED BY LEVINSON PAGE 137,
C             EQUATIONS 17, 18).  IT IS IN THE COMPUTATION OF CC THAT
C             THE EXTRA VALUE OF RR (I.E. RR(M+1)) IS REQUIRED.
C             THE SOLUTION FOR AA IS UNAFFECTED BY THE CHOICE OF R(M+1).
C             HOWEVER IF THE (M+2)*(M+2) MATRIX R(K-N) (K,N=0,1,...,M+1)
C             IS ALSO POSITIVE DEFINITE, THEN CC HAS AN IMPORTANT
C             INTERPRETATION. IN THIS CASE CC IS THE TIME REVERSE OF
C             THE M+1 TERM LEAST SQUARES UNIT-PREDICTION-DISTANCE
C             OPERATOR FOR ANY TIME SERIES WHOSE AUTOCORRELATION
C             FUNCTION (THRU LAG M+1) EQUALS RR(0,1,...,M+1).
C
C             WLLSFP PROVIDES A REENTRY OPTION ALLOWING EFFICIENT
C             RECURSION TO LARGER EQUATION SETS.
C
C LANGUAGE   - FORTRAN II
C EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE    - 216 REGISTERS
C SPEED      - ABOUT  60(M**2) MACHINE CYCLES
C                WHERE M IS THE LENGTH OF THE A-VECTOR.
C AUTHOR     - R.A. WIGGINS,  9/28/62
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   FDOTR, FDOT, MOVE
C       AND FORTRAN SYSTEM ROUTINES -   NONE
C
C FORTRAN USAGE
C      CALL WLLSFP (LR,R,G,LA,A,C)
C
C INPUTS
C
C    R(I)       I=1...LR,LR+1 CONTAINS THE VECTOR RR(0,1,...,LR)
C
C    G(I)       I=1...LR CONTAINS GG(0,1,...,LR-1)
C
```

```
      C    LR         IS FORTRAN II INTEGER, GRTHN=2.
      C
      C    LA         MUST LIE IN THE RANGE 2,...,LR OR -2,...,-LR+1
      C               IF LA GRTHN=2, LA STANDS FOR THE DESIRED LENGTH OF THE
      C                 FILTER, I.E. LA=M+1
      C               IF LA LSTHN=-2, WLLSFP ASSUMES THAT THIS IS A REENTRY
      C                 CALL AND THAT THE USER WISHES TO EXTEND THE PREVIOUS
      C                 FILTER WHOSE LENGTH WAS = MAGNITUDE (LA) TO A NEW
      C                 FILTER OF GREATER LENGTH = LR, I.E. THE NEW M=LR-1.
      C                 IN THIS CASE A(1...LLA) AND C(1...LLA) WHERE LLA =
      C                 MAGNITUDE (LA) MUST NOT HAVE BEEN DISTURBED FOLLOWING
      C                 THE PREVIOUS CALL.
      C               IS FORTRAN II INTEGER
      C
      C    C(I)       I=1...2*LR IS ERASABLE COMPUTATION SPACE NEEDED BY WLLSFP
      C
      C OUTPUTS
      C
      C    LA         IS SET EQUAL TO LENGTH OF A, SEE INPUT.
      C               IS FORTRAN INTEGER
      C
      C    A(I)       I=1...LA CONTAINS THE SOLUTION VECTOR AA(0,1,...,M=LA-1)
      C
      C    C(I)      I=1...LA WILL CONTAIN THE LEVINSON AUXILIARY SEQUENCE
      C                 CC(0,1,...,LA-1=M)
      C
      C EXAMPLES
      C
      C 1. INPUTS  - LR = 3   LA = 2
      C               R(1...3) = 1.25,0.5,0.   G(1...3) = 1.,0.,0.
      C    OUTPUTS - A(1...2) = 0.95238, -0.38095  LA=2
      C
      C 2. INPUTS  - SAME AS EXAMPLE 1. (AFTER EXAMPLE 1. IS COMPUTED)
      C                 EXCEPT LA = -2
      C    OUTPUTS   - A(1...3) = 0.98824, -0.47059, 0.18824  LA = 3
      C
      C 3. EXAMPLE OF USE OF WLLSFP TO CONSTRUCT A LEAST SQUARE REALIZABLE
      C      FILTER WHICH, WHEN CONVOLVED WITH A SPECIFIED SIGNAL, WILL RESULT
      C      IN A DESIRED OUTPUT SIGNAL (USING SUBROUTINES QACORR AND QXCORR).
      C
      C    INPUTS  - LET S(I), I=1,LS BE THE INPUT SIGNAL
      C                   D(I), I=1,LS BE THE DESIRED OUTPUT
      C
      C    USAGE   - C      FORM THE AUTOCORRELATION OF THE SIGNAL
      C              C
      C                     CALL QACORR (S,LS,MXACC,LA,SPACE,R,IANS1)
      C              C
      C              C      FORM THE CROSSCORRELATION OF THE DESIRED OUTPUT
      C              C        WITH THE SIGNAL. NOTE THAT ONLY HALF OF THE
      C              C        CROSSCORRELATION FORMED BY QXCORR IS NEEDED BY
      C              C        WLLSFP.
      C              C
      C                     CALL QXCORR (S,D,LS,MXACC,LA,SPACE,G,IANS2)
      C              C
      C              C      FORM THE DESIRED FILTER
      C              C
      C                     CALL WLLSFP (LA,R,G(LA+1),LA,A,SPACE)
      C
      C    OUTPUTS - A(I), I=1,LA IS THE DESIRED FILTER.
      C
      C
      C 4. EXAMPLE OF USE OF WLLSFP TO CONSTRUCT A LEAST SQUARE REALIZABLE
      C      FILTER WHICH, WHEN CONVOLVED WITH A SIGNAL PLUS NOISE, WILL
      C      RESULT IN THE NOISE BEING SUPRESSED AND THE SIGNAL SHAPE BEING
      C      CHANGED TO A DESIRED OUTPUT.
      C
      C    INPUTS  - LET  S(I), I=1,LS BE THE INPUT SIGNAL
      C                  XN(I), I=1,LN BE A SAMPLE OF THE NOISE
      C                   D(I), I=1,LS BE THE DESIRED OUTPUT
      C
      C    USAGE   - C      FORM THE AUTOCORRELATION VECTOR R.
      C              C
      C                     CALL QACORR (S,LS,MXACC,LA,SPACE,AUTOS,IANS1)
      C                     CALL QACORR (XN,LN,MXACC,LA,SPACE,AUTON,IANS2)
      C                     DO 10 I=1,LA
```

```
C                   10  R(I) = AUTOS(I)+AUTON(I)
C                 C
C                 C      FORM THE CROSSCORRELATION VECTOR G. (SEE COMMENT
C                 C        IN EXAMPLE 3.)
C                 C
C                        CALL QXCORR (S,D,LS,MXACC,LA,SPACE,G,IANS3)
C                 C
C                 C      FORM THE DESIRED FILTER
C                 C
C                        CALL WLLSFP (LA,R,G(LA+1),LA,A,SPACE)
C
C     OUTPUTS - A(I), I=1,LA IS THE DESIRED FILTER.
C
C  5. EXAMPLE OF USE OF WLLSFP TO FORM A LEAST-SQUARE PREDICTION FILTER.
C
C     INPUTS   - LET S(I), I=1,LS BE A SIGNAL WAVELET
C                      NP  BE THE PREDICTION DISTANCE
C
C     USAGE    - C       FORM THE AUTOCORRELATION VECTOR R.
C                C
C                       MXLAG=LA+NP-1
C                       CALL QACORR (S,LS,MXACC,MXLAG,SPACE,R,IANS1)
C                C
C                C       FORM THE PREDICTION ERROR FILTER WITH PREDICTION
C                C         DISTANCE NP=1.
C                C
C                       IG=NP+1
C                       CALL WLLSFP (LA,R,R(IG),LA,A,SPACE)
C                C
C
C     OUTPUTS - A(I), I=1,LA IS THE DESIRED FILTER.
C
C  6. EXAMPLE OF USE OF WLLSFP TO FACTOR A TIME SERIES,  THAT IS, TO
C       FINE THE LEAST SQUARE MINIMUM PHASE WAVELET ASSOCIATED WITH
C       A SERIES.
C
C     INPUTS   - LET  X(I), I=1,LX BE THE SERIES TO BE FACTORED.
C
C     USAGE    - C       FORM THE AUTOCORRELATION VECTOR R.
C                C
C                        CALL QACORR (X,LX,MXACC,LA,SPACE,R,IANS1)
C                C
C                C       FORM THE CROSSCORRELATION VECTOR G.
C                C
C                        G(1) = 1.
C                        DO 10 I=2,LA
C                 10     G(I) = 0.
C                C
C                C       FORM THE PREDICTION ERROR FILTER LA.
C                C
C                        CALL WLLSFP (LA,R,G,LA,A,SPACE)
C                C
C                C       FORM THE MINIMUM PHASE WAVELET XMW.
C                C
C                        CALL POLYDV (LA,A,1,1.,LXMW,XMW)
C
C     OUTPUTS - XMW(I), I=1,LXMW IS THE MINIMUM PHASE WAVELET.
C               A(I), I=1,LA IS THE PREDICTION ERROR FILTER WITH
C                              PREDICTION DISTANCE 1.
C
C
C
C      PROGRAM FOLLOWS BELOW.
C
       DIMENSION R(10),G(10),A(10),C(20)
C
C      REDEFINE INPUT CONSTANTS WHICH ARE USED A LOT
       LR1=LR
       N=2
       LA1=LA
       R1=R(1)
C
C      SET UP THE MODE OF OPERATION DEFINED BY LA
       IF(LA1)  10,220,30
  10   N=XABSF(LA1)+1
```

```
      GO TO 75
   30 LR1=LA1
C
C     SET UP INITIAL VALUES OF C, A, AND E.
   40 A(1)=G(1)/R1
      C(1)=R(2)/R1
   75 LR2=LR1+1
C
C     DO THE RECURSIONS
C*****
      DO 200 M=N,LR1
      M1 = M-1
C
C     FORM THE NEXT A(K) VECTOR
      CALL FDOTR (M1,A,R(2),C2)
      CALL FDOTR (M1,C,R(2),C3)
    1 A(M)=(G(M)-C2)/(R1-C3)
C    *
      DO 100  K=1,M1
  100 A(K)=A(K)-C(K)*A(M)
C    *
C
C     FORM THE NEXT C VECTOR
      CALL FDOT (M1,C,R(2),C2)
    2 C(LR2) = (R(M+1)-C2)/(R1-C3)
C    *
      DO 150  K=2,M
      K1=M-K
      K2=K+LR1
  150 C(K2)=C(K-1)-C(LR2)*C(K1+1)
C    *
      CALL MOVE (M,C(LR2),C)
C
  200 CONTINUE
C*****
C
C     SET OUTPUT PARAMETERS
  210 LA=M1+1
  220 RETURN
      END
```

```
*      WRTDAT (SUBROUTINE)             9/8/64   LAST CARD IN DECK IS NO. 0125
*      FAP
*WRTDAT
       COUNT   100
       LBL     WRTDAT
       ENTRY   WRTDAT (ITAPE,DATA,LDATA,IANS)
*
*                    ----ABSTRACT----
*
* TITLE - WRTDAT
*          WRITE BINARY DATA ON TAPE
*
*              WRTDAT WRITES A BINARY RECORD FROM A FORTRAN VECTOR ON A
*              SPECIFIED OUTPUT TAPE.  ERROR RETURNS INDICATE IF A
*              REDUNDANCY OR END-TAPE CONDITION WAS FOUND.  NO SUM-CHECK
*              IS PROVIDED.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709, 7090, 7094 (MAIN FRAME AND TAPE UNIT)
* STORAGE    - 77 REGISTERS
* SPEED      - (PRIMARILY CONTROLLED BY DATA-CHANNEL TIMING)
* AUTHOR     - R.A. WIGGINS    JULY, 1964
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (IOS), (TCO), (WRS), (RCH), (TRC),
*                                     (ETT)
*
* FORTRAN USAGE
*     CALL WRTDAT (ITAPE,DATA,LDATA,IANS)
*
* INPUTS
*    ITAPE     LOGICAL OUTPUT TAPE NUMBER.
*              MUST BE GRTHN= 1, LSTHN= 20
*
*    DATA(I)   I=1,...,LDATA IS A DATA VECTOR (IN ANY MODE) TO BE
*                WRITTEN.
*
*    LDATA     LENGTH OF DATA VECTOR.
*              MUST BE GRTHN= 1
*
* OUTPUTS
*
*    IANS      =0  IF ALL OK.
*              =2  IF A REDUNDANCY IS ENCOUNTERED.
*              =3  IF AN END-TAPE MARK IS ENCOUNTERED.
*              =-1 IF ITAPE LSTHN 1 OR GRTHN 20
*              =-2 IF LDATA LSTHN 1
*
* EXAMPLE
*
* 1. CONSTRUCTION OF A FORTRAN STYLE BINARY RECORD
*    INPUTS  - ITAPE = 5  DATA(1...3) = OCT 000002000001, 123456123456,
*              654321654321  LDATA = 3
*    USAGE   -       CALL WRTDAT (ITAPE,DATA,LDATA,IANS)
*                    BACKSPACE ITAPE
*                    READ TAPE ITAPE, (DATIN(I),I=1,2)
*    OUTPUTS - IANS = 0  DATIN(1...2) = OCT 123456123456, 654321654321
*
* PROGRAM FOLLOWS BELOW
*
XR4    HTR     0
       BCI     1,WRTDAT
WRTDAT SXD     XR4,4
       SXA     XR1,1
       SXA     XR2,2
       CLA*    3,4
       TZE     IANM2
       TMI     IANM2
       PDX     ,1
       CAL     2,4
       STA     UT
       LDQ     =20B17
       CLA*    1,4
```

```
       TZE    IANM1
       TMI    IANM1
       TLQ    IANM1
       ADD    =O20
       TSX    $(IOS),4
       LXD    XR4,4
       LDQ*   $(TCO)
       SLQ    TCOA
       SLQ    TCOA1
       LDQ*   $(WRS)
       STQ    WRSA
       LDQ*   $(RCH)
       SLQ    RCHA
       XCL
       ADD    =O000400000000
       XCL
       SLQ    LCHA
       LDQ*   $(TRC)
       SLQ    TRCA
       LDQ*   $(ETT)
       SLQ    ETTA
TCOA   TCOA   *
WRSA   WTDA   **
RCHA   RCHA   UT
       TRA    INC
LCHA   LCHA   UT
INC    CAL    UT
       SUB    =1B35
       SLW    UT
       TIX    LCHA,1,1
TCOA1  TCOA   *
TRCA   TRCA   IAN2
ETTA   ETTA
       TRA    IAN3
       IOT
       NOP
       CLA    =0
RETURN STO*   4,4
XR1    AXT    **,1
XR2    AXT    **,2
       TRA    5,4
IANM1  CLS    =1B17
       TRA    RETURN
IANM2  CLS    =2B17
       TRA    RETURN
IAN2   CLA    =2B17
       TRA    RETURN
IAN3   CLA    =3B17
       TRA    RETURN
UT     IOCT   **,,1
       END
```

```
*       XACTEQ (FUNCTION)                     9/4/64   LAST CARD IN DECK IS NO. 0075
*       FAP
*XACTEQ
        COUNT   50
        LBL     XACTEQ
        ENTRY   XACTEQ F(X,Y)
*
*
*                    ----ABSTRACT----
*
* TITLE - XACTEQ
*         SIGN OF DIFFERENCE OF 2 VARIABLES OR 0 IF SAME INCLUDING SIGN
*
*               XACTEQ TAKES THE DIFFERENCE BETWEEN TWO VARIABLES OF ANY
*               MODE AND RETURNS -1 OR +1 AS THE SIGN OF THE DIFFERENCE,
*               OR +0 IF THE TWO VARIABLES ARE EXACTLY EQUAL, INCLUDING
*               SIGNS.  (+0 IS CONSIDERED GREATER THAN -0)
*
* LANGUAGE   - FAP FUNCTION (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 11 REGISTERS
* SPEED      - 6 OR 10 MACHINE CYCLES
* AUTHOR     - S.M.SIMPSON,JR.  APRIL 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*      AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     IXMY=XACTEQF(X,Y)
*
*
* INPUTS
*
*    X          IS ANY MODE.
*
*    Y          IS ANY MODE.
*
*
* OUTPUTS
*
*    IXMY       =  0  IF BITS S,1...35 OF X ARE EXACTLY IDENTICAL TO
*                          BITS S,1...35 OF Y.
*               = +1  IF X IS GREATER THAN Y.
*               = -1  IF X IS LESS THAN Y.
*
*
* EXAMPLES
*
* 1. INPUTS  - X(1...5) = 1.,0.,-0.,0.,0.   IX(1...5) = 1,0,-0,0,0
*              Y(1...5) = 0.,-0.,0.,0.,1.   IY(1...5) = 0,-0,0,0,1
*    USAGE   -       DO  10  I=1,5
*                    IXMY1(I)=XACTEQF(X(I),Y(I))
*                 10 IXMY2(I)=XACTEQF(IX(I),IY(I))
*    OUTPUTS - IXMY1(1...5) = +1, +1, -1, 0, -1
*              IXMY2(1...5) = +1, +1, -1, 0, -1
*
*
*
* PROGRAM FOLLOWS BELOW.
*
*
        BCI     1,XACTEQ
XACTEQ  TLQ     ABIGRQ
        XCA
        TLQ     QBIGRA
        PXD     0,0
        TRA     1,4
ABIGRQ  CLA     KD1
        TRA     1,4
QBIGRA  CLS     KD1
        TRA     1,4
 KD1    PZE     0,0,1
        END
```

```
************************
*    XADDK             *
************************
 REFER TO
     ADDK

************************
*    XADDKS            *
************************
 REFER TO
     ADDK

************************
*    XARG              *
************************
 REFER TO
     LOCATE
```


```
************************
*    XADDK             *
************************
 REFER TO
     ADDK

************************
*    XADDKS            *
************************
 REFER TO
     ADDK

************************
*    XARG              *
************************
 REFER TO
     LOCATE
```

```
*      XAVRGE (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0103
*      FAP
*XAVRGE
       COUNT   150
       LBL     XAVRGE
       ENTRY   XAVRGE (IX,LIX,IXAVG)
       ENTRY   XAVRGR (IX,LIX,IXAVG)
*
*                    ----ABSTRACT----
*
* TITLE - XAVRGE WITH SECONDARY ENTRY XAVRGR
*         FIND AVERAGE OF FIXED PT VECTOR
*
*               XAVRGE FINDS THE MEAN, TRUNCATED TO A FORTRAN-II INTEGER,
*               OF A GIVEN FXD VECTOR.  OVERFLOW CAN NOT OCCUR.
*               XAVRGR FINDS THE ROUNDED MEAN.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34 REGISTERS
* SPEED      -        7090      709
*               XAVRGE  80  OR  87 + 11*LX  MACHINE CYCLES,
*               XAVRGR  84  OR  91 + 11*LX      LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - XDIV,XDIVR
*        AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL XAVRGE(IX,LIX,IXAVG)
*     CALL XAVRGR(IX,LIX,IXAVG)
*
* INPUTS
*
*    IX(I)      I=1...LIX IS A FXD VECTOR
*
*    LIX        SHOULD EXCEED ZERO
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LIX LSTHN 1
*
*    IXAVG      WILL = (1/LIX) * SUM (FROM I=1 TO LIX)  OF  IX(I),
*                      TRUNCATED TO INTEGER (XAVRGE) OR ROUNDED (XAVRGR)
*
*               THE SUMMATION IS CARRIED OUT IN A MANNER WHICH AVOIDS
*                  OVERFLOW
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...4)=1,2,3,4
*               IY(1...4)=90000,91000,92000,93000
*               IU=0
*    USAGE   -        CALL XAVRGE(IX,4,IXAV1)
*                     CALL XAVRGR(IX,4,IXAV2)
*                     CALL XAVRGR(IX(2),3,IXAV3)
*                     CALL XAVRGE(IY,4,IXAV4)
*                     CALL XAVRGE(IX,1,IXAV5)
*                     CALL XAVRGE(IX,0,IU)
*    OUTPUTS - IXAV1=2  IXAV2=3  IXAV3=3  IXAV4=91500
*               IXAV5=1  IU=0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR HAS XDIV,XDIVR (FUNCTIONS)
       HTR     0                   XR1
       HTR     0                   XR4
       BCI     1,XAVRGE
* PRINCIPAL ENTRY.  XAVRGE(IX,LIX,IXAVG)
XAVRGE CLA     DIV
 SETUP STO     VARY
       SXD     XAVRGE-3,1
       SXD     XAVRGE-2,4
 K1    CLA     1,4
       ADD     K1                  A(IX)+1
```

****
*   XAVRGE   *
****
(PAGE 2)

```
        STA     GET
        CLA*    2,4                 LIX
        ARS     18
        TMI     LEAVE
        PAX     0,1
        TXL     LEAVE,1,0
        XCA                         SAVE LIX IN MQ
* SUM   IX(1...LIX)  IN ADDRESS OF AC
        STZ     SUM
 GET    CLA     **,1                **=A(IX)+1
        ARS     18
        ADD     SUM
        STO     SUM
        TIX     GET,1,1
* FIND THE AVERAGE (LIX STILL IN MQ ADDRESS)
 VARY   TSX     **,4                **=$XDIV OR $XDIVR
* STORE AND LEAVE
        LXD     XAVRGE-2,4
        STO*    3,4
 LEAVE  LXD     XAVRGE-3,1
        TRA     4,4                 IXAVG
* SECOND ENTRY.  XAVRGR(IX,LIX,IXAVG)
XAVRGR  CLA     DIVR
        TRA     SETUP
* CONSTANTS, VARIABLES
 DIV    TSX     $XDIV,4
 DIVR   TSX     $XDIVR,4
 SUM    PZE     **                  SUMMATION
        END
```

```
***********************
*    XAVRGR           *
***********************
 REFER TO
     XAVRGE

***********************
*    XBOOST           *
***********************
 REFER TO
     BOOST

***********************
*    XCMPRA           *
***********************
 REFER TO
     CMPRA

***********************
*    XDANL            *
***********************
 REFER TO
     ADANL

***********************
*    XDANX            *
***********************
 REFER TO
     ADANL

***********************
*    XDELTA           *
***********************
 REFER TO
     DELTA

***********************
*    XDFPRS           *
***********************
 REFER TO
     DIFPRS
```


```
***********************
*    XAVRGR           *
***********************
 REFER TO
     XAVRGE

***********************
*    XBOOST           *
***********************
 REFER TO
     BOOST

***********************
*    XCMPRA           *
***********************
 REFER TO
     CMPRA

***********************
*    XDANL            *
***********************
 REFER TO
     ADANL

***********************
*    XDANX            *
***********************
 REFER TO
     ADANL

***********************
*    XDELTA           *
***********************
 REFER TO
     DELTA

***********************
*    XDFPRS           *
***********************
 REFER TO
     DIFPRS
```

```
*      XDIV   (FUNCTION)                9/29/64   LAST CARD IN DECK IS NO. 0108
*      FAP
*XDIV
       COUNT   100
       LBL     XDIV
       ENTRY   XDIV   F(NUMERA,IDENOM)
       ENTRY   XDIVR  F(NUMERA,IDENOM)
*
*                    ----ABSTRACT----
*
* TITLE - XDIV  WITH SECONDARY ENTRY XDIVR
*        FXD PT DIVIDE WITH TRUNCATION OR ROUNDING TO FORTRAN-II INTEGER
*
*              XDIV IS A FUNCTION WHOSE VALUE IS THE RATIO OF ITS TWO
*              FIXED POINT ARGUMENTS, TRUNCATED AS A FORTRAN-II INTEGER.
*
*              XDIVR IS IDENTICAL EXCEPT THAT THE RESULT IS ROUNDED.
*
* LANGUAGE   - FAP FUNCTIONS (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 27 REGISTERS
* SPEED      - XDIV  TAKES 35(7090) OR 42(709) MACHINE CYCLES
*              XDIVR TAKES 52(7090) OR 59(709) MACHINE CYCLES
*
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     IQUOT = XDIVF (NUMERA,IDENOM)
*     IQUOT = XDIVRF(NUMERA,IDENOM)
*
* INPUTS
*
*    NUMERA    IS ANY FXD.PT. NO.
*
*    IDENOM    IS ANY NON-ZERO FXD.PT. NO. WITH THE SAME BINARY POINT
*                 AS NUMERA.
*
* OUTPUTS    - IF IDENOM = 0  PROGRAMS RETURN WITH NO ACTION (AC AND
*              MQ ARE LEFT AS IS) SO THAT THE EFFECTIVE VALUE OF THE
*              FUNCTION WILL BE = NUMERA.  THE MQ IS ALWAYS RESTORED
*              TO = IDENOM.
*
*    IQUOT     WILL EQUAL NUMERA/IDENOM, TRUNCATED (XDIV) OR ROUNDED
*                (XDIVR) TO A FORTRAN II INTEGER.
*              WILL = NUMERA  IF IDENOM = 0
*
*              OVERFLOW CAN NOT OCCUR IF THE INPUTS ARE FORTRAN-II
*              INTEGERS.  NO OVERFLOW TEST IS MADE.
*
* EXAMPLES
*
* 1. USAGE   -        IQ1 = XDIVF   ( 1, 4)
*                     IQ2 = XDIVF   ( 2, 4)
*                     IQ3 = XDIVF   ( 4, 4)
*                     IQ4 = XDIVF   ( 9, 4)
*                     IQ5 = XDIVRF  ( 1, 4)
*                     IQ6 = XDIVRF  ( 2, 4)
*                     IQ7 = XDIVF   ( 1,-1)
*                     IQ8 = XDIVRF  (-1, 1)
*                     IQ9 = XDIVF   ( 3, 0)
* OUTPUTS      IQ1=0  IQ2=0  IQ3=1  IQ4=2
*              IQ5=0  IQ6=1  IQ7=-1  IQ8=-1
*              IQ9=3  (DIVISION BY ZERO)
*
* 2. INPUTS  - X = OCT 000000000011  Y = OCT 000000000004
*    USAGE   -        IQ10 = XDIVF (X,Y)
*    OUTPUTS - IQ10 = OCT 000002000000
*
* PROGRAM FOLLOWS BELOW
*
```

****
*   XDIV   *
****
(PAGE 2)

```
* NO TRANSFER VECTOR
*
* PRINCIPAL ENTRY,  XDIVF(NUMERA,IDENOM)
 XDIV  STQ     TEMP                SAVE IDENOM
       LDQ     XCA            SET FOR TRUNCATION
 SETUP STQ     VARY
       NZT     TEMP                ZERO DENOMINATOR CHECK
       TRA     LEAVE
       LRS     35                  0 IN AC, NUMERA IN MQ
       DVP     TEMP
 VARY  NOP                    =  XCA  OR  TRA ROUND
 ALS   ALS     18
 LEAVE LDQ     TEMP                RESTORE IDENOM
       TRA     1,4
* ROUNDING INSERT, COMPARES TWICE THE REMAINDER AGAINST IDENOM.
 ROUND SSP
       ALS     1              (OVERFLOW IMPOSSIBLE)
       SBM     TEMP
       CLM                    PREPARE FOR ROUNDING DOWN
       TMI     RXCA           (SIGN BIT UNDISTURBED)
       CLA     KRND           PREPARE FOR ROUNDING UP
 RXCA  XCA
       RND
       TRA     ALS
* SECOND ENTRY,  XDIVRF(NUMERA,IDENOM)
 XDIVR STQ     TEMP                SAVE IDENOM
       LDQ     RND                 SET FOR ROUNDING
       TRA     SETUP
* CONSTANTS, VARIABLES
 XCA   XCA
 RND   TRA     ROUND
 KRND  OCT     200000000000
 TEMP  PZE     **,**,**            =IDENOM
       END
```

```
************************
*    XDIVK             *
************************
 REFER TO
     ADDK

************************
*    XDIVKS            *
************************
 REFER TO
     ADDK

************************
*    XDIVR             *
************************
 REFER TO
     XDIV

************************
*    XDPRSS            *
************************
 REFER TO
     BOOST
```


```
************************
*    XDIVK             *
************************
 REFER TO
     ADDK

************************
*    XDIVKS            *
************************
 REFER TO
     ADDK

************************
*    XDIVR             *
************************
 REFER TO
     XDIV

************************
*    XDPRSS            *
************************
 REFER TO
     BOOST
```

```
*       XDVIDE (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0104
*       FAP
*XDVIDE
        COUNT   150
        LBL     XDVIDE
        ENTRY   XDVIDE (IX,LIX,IXDVSR,IXDVDD)
        ENTRY   XDVIDR (IX,LIX,IXDVSR,IXDVDD)
*
*                   ----ABSTRACT----
*
* TITLE - XDVIDE
*        DIVIDE A FXD VECTOR BY A CONSTANT
*
*               XDVIDE FORMS A VECTOR EQUAL TO A GIVEN VECTOR DIVIDED
*               BY A FXD CONSTANT, TRUNCATING THE DIVISIONS.  OUTPUT
*               MAY REPLACE INPUT.
*
*               XDVSR  IS IDENTICAL EXCEPT IT ROUNDS THE DIVISIONS.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 33 REGISTERS
* SPEED      -                7090   709
*               XDVIDE  42 + (47  OR  54)*LX  MACHINES CYCLES,
*               XDVIDR  44 + (49  OR  56)*LX    LX = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - XDIV, XDIVR
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL XDVIDE(IX,LIX,IXDVSR,IXDVDD)
*     CALL XDVIDR(IX,LIX,IXDVSR,IXDVDD)
*
* INPUTS
*
*    IX(I)      I=1...LIX  IS A FXD VECTOR
*
*    LIX        SHOULD EXCEED ZERO
*
*    IXDVSR     IS A NON-ZERO FXD QUANTITY.  EQUIVALENCE(IXDVSR,
*                  SOME IX(I))  IS PERMITTED.
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LIX LSTHN 1 OR IXDVSR=0
*
*    IXDVDD(I) I=1...LIX  HAS VALUES IXDVDD(I)=IX(I)/IXDVSR  TRUNCATED
*                  TO INTEGERS (XDVIDE) OR ROUNDED TO INTEGERS (XDVIDR).
*               EQUIVALENCE (IXDVCD,IX) IS PERMITTED.
*
*               THE INITIAL VALUE OF IXDVSR IS ALWAYS USED AS THE DIVISOR
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...4)=1,2,3,4  IU=0  IV=0
*    USAGE   -       CALL XDVIDE(IX,4,2,IY)
*                    CALL XDVIDR(IX,4,2,IZ)
*                    CALL XDVIDR(IX,1,2,IW)
*                    CALL XDVIDR(IX,0,2,IU)
*                    CALL XDVIDR(IX,1,0,IV)
*                    CALL XDVIDR(IX,4,IX(3),IX)
*    OUTPUTS - IY(1...4)=0,1,1,2  IZ(1...4)=1,1,2,2
*               IW=1  IU=IV=0  (NO OUTPUT CASES)  IX(1...4)=0,1,1,1
*
* PROGRAM FOLLOWS BELOW
*
* TRANSFER VECTOR HAS XDIV,XDIVR FUNCTIONS
        HTR     0                  XR1
        HTR     0                  XR4
        BCI     1,XDVIDE
* PRINCIPAL ENTRY, XDVIDE(IX,LIX,IXDVSR,IXDVDD)
XDVIDE CLA      DIV
 SETUP STO      VARY
        SXD     XDVIDE-3,1
```

```
        SXD     XDVIDE-2,4
  K1    CLA     1,4
        ADD     K1              A(IX)+1
        STA     GET
        CLA     4,4
        ADD     K1              A(IXDVDD)+1
        STA     STORE
        CLA*    2,4             LIX
        TMI     LEAVE
        PDX     0,1
        TXL     LEAVE,1,0
        CLA*    3,4             ZERO TEST FOR IXDVSR
        TZE     LEAVE
 * DIVISION LOOP
        XCA                     IXDVSR (REMAINS IN MQ)
  GET   CLA     **,1            **=A(IX)+1
  VARY  NOP                     =TSX  $XDIV,4  OR  TSX  $XDIVR,4
  STORE STO     **,1            **=A(IXDVDD)+1
        TIX     GET,1,1
 * EXIT
  LEAVE LXD     XDVIDE-3,1
        LXD     XDVIDE-2,4
        TRA     5,4
 * SECONDARY ENTRY, XDVIDR(IX,LIX,IXDVSR,IXDVDD)
 XDVIDR CLA     DIVR
        TRA     SETUP
 * CONSTANTS
  DIV   TSX     $XDIV,4
  DIVR  TSX     $XDIVR,4
        END
```

```
************************
*    XDVIDR            *
************************
 REFER TO
     XDVIDE
```

```
************************
*    XDVRK             *
************************
 REFER TO
     ADDK
```

```
************************
*    XDVRKS            *
************************
 REFER TO
     ADDK
```


```
************************
*    XDVIDR            *
************************
 REFER TO
     XDVIDE
```

```
************************
*    XDVRK             *
************************
 REFER TO
     ADDK
```

```
************************
*    XDVRKS            *
************************
 REFER TO
     ADDK
```

```
*       XFIXM (FUNCTION)                9/29/64   LAST CARD IN DECK IS NO. 0097
*       FAP
*XFIXM
        COUNT    100
        LBL      XFIXM
        ENTRY    XFIXM  F(JOB,FLTG)
*
*                   ----ABSTRACT----
*
* TITLE - XFIXM
*        TRUNCATE OR ROUND FLOATING PT. NUMBER TO MACHINE INTEGER.
*
*              XFIXM TRUNCATES OR ROUNDS A FLOATING POINT NUMBER TO A
*              FIXED POINT INTEGER WHOSE BINARY POINT IS TO THE RIGHT OF
*              BIT 35.  FLOATING POINT NUMBERS WITH MAGNITUDES
*              EXCEEDING 2.**27-1. ARE TREATED AS THOUGH THEIR
*              MAGNITUDES EQUALLED 2.**27-1.
*
* LANGUAGE  - FAP SUBROUTINE (FORTRAN II FUNCTION)
* EQUIPMENT - 709, 7090  (MAIN FRAME ONLY)
* STORAGE   - 31 REGISTERS
* SPEED     - ABOUT 35 MACHINE CYCLES
* AUTHOR    - S.M. SIMPSON JR. , NOV/1962
*
*                   ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES -   NONE
*       AND FORTRAN SYSTEM ROUTINES -   NONE
*
* FORTRAN USAGE
*     INTEGR = XFIXMF (JOB,FLTG)
*
* INPUTS
*
*    JOB           =0 MEANS TRUNCATE
*               NOT=0 MEANS ROUND TO NEAREST INTEGER
*
*    FLTG       IS A FLOATING POINT NUMBER.
*
* OUTPUTS
*
*    INTEGR     IS THE MACHINE LANGUAGE INTEGER EQUIVALENT TO FLTG.
*               = PLUS OR MINUS OCT 000777777777 IF MAGNITUDE OF
*                 FLTG EXCEEDS 2.**27-1.
*
* EXAMPLES
*
* 1. INPUTS  - JOB = 0   FLTG = 3.52
*    OUTPUTS - INTEGR = OCT 000000000003
*
* 2. INPUTS  - JOB = 1   FLTG = 3.52
*    OUTPUTS - INTEGR = OCT 000000000004
*
* 3. INPUTS  - JOB = 1   FLTG = -3.52
*    OUTPUTS   INTEGR = OCT 400000000004
*
* 4. INPUTS  - JOB = 0   FLTG =  -1234567890. (EXCEEDS  2.**27-1.)
*    OUTPUTS - INTEGR = OCT 400777777777
*
* 5. INPUTS  - JOB = 1   FLTG = 1234567890.
*    OUTPUTS - INTEGR = OCT 000777777777
*
        HTR      0
        BCI      1,XFIXM
 XFIXM SXD       XFIXM-2,4
* CHECK MAGNITUDE OF FLTG
        STO      JOB
        STQ      FLTG
        XCA
        SSP
        CAS      LIMIT
        NOP                    TOO BIG
        TRA      TOOBIG        TOO BIG
* OK, FIX IT
        CLA      FLTG
```

****
*   XFIXM   *
****
(PAGE 2)

```
        UFA     K1
        LRS     0
        ANA     K2
        LLS     0
* CHECK FOR ROUNDING
        NZT     JOB
        TRA     LEAVE
        RQL     8
        RND
 LEAVE  TRA     1,4
* CLIP BIG NUMBERS
TOOBIG  CLA     FLTG
        TMI     *+3
        CLA     K2
        TRA     LEAVE
        CLS     K2
        TRA     LEAVE
 LIMIT  OCT     234400000000   (=2.**27)
 K1     OCT     233000000000
 K2     OCT     000777777777
 JOB    PZE     **
 FLTG   PZE     **
        END
```

```
************************
*    XINDEX            *
************************
 REFER TO
     LOCATE
```


```
************************
*    XINDEX            *
************************
 REFER TO
     LOCATE
```

```
*      XLCOMN (FUNCTION)                  9/4/64   LAST CARD IN DECK IS NO. 0075
*      FAP
*XLCOMN
       COUNT   100
       LBL     XLCOMN
       ENTRY   XLCOMN F(ZIFACT)
*
*
*                     ----ABSTRACT----
*
* TITLE - XLCOMN
*        FIND LENGTH OF COMMON STORAGE
*
*              XLCOMN EXAMINES OCTAL LOCATION 143 TO FIND EITHER THE
*              LENGTH OF COMMON SPACE BEYOND THE LAST STORED ROUTINE, OR
*              THE TOTAL LENGTH OF COMMON DIMENSIONED IN THE ROUTINES.
*
*              UNDER THE FORTRAN MONITOR SYSTEM, OCTAL LOCATION 143
*              CONTAINS THE ADDRESS OF THE FIRST UNUSED SPACE IN THE
*              DECREMENT, AND THE ADDRESS OF THE LAST COMMON SPACE USED
*              IN THE ADDRESS.
*
* LANGUAGE   - FAP FUNCTION (FORTRAN II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 14 REGISTERS
* SPEED      - ABOUT 16 MACHINE CYCLES.
* AUTHOR     - R.A. WIGGINS    4/64
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     LCOMON = XLCOMNF(ZIFACT)
*
*
* INPUTS
*
*    ZIFACT    =0. IF ACTUAL LENGTH OF AVAILABLE COMMON FROM THE END OF
*                  THE STORED PROGRAMS THROUGH LOCATION 32561 IS
*                  DESIRED.
*              NOT= 0. IF DIMENSIONED LENGTH OF COMMON IS DESIRED.
*
*
* OUTPUTS
*
*    LCOMON    IS LENGTH OF COMMON ACCORDING TO ZIFACT
*
*
* EXAMPLES
*
* 1. INPUTS  - SUPPOSE A MAIN PROGRAM AND A SET OF SUBROUTINES ARE
*              LOADED INTO LOCATIONS 144 THROUGH 4114 (OCTAL)
*              (2124 DECIMAL), AND THAT THE MAIN IS DIMENSIONED WITH
*              COMMON STORAGE OF LENGTH 2000 (DECIMAL).
*    USAGE   -       LCOMN1 = XLCOMN (0.)
*                    LCOMN2 = XLCOMN (1.)
*    OUTPUTS - LCOMN1 = 30436      LCOMN2 = 2000
*
*
* PROGRAM FOLLOWS BELOW
*
       BCI     1,XLCOMN
XLCOMN TZE     A1
       CLA     99
       ANA     =0000000777777
       ALS     18
       TRA     A1+1
 A1    CLA     99
       ANA     =0777777000000
       SSM
       ADD     =32561B17
       TRA     1,4
       END
```

```
*     XLIMIT (FUNCTION)                9/4/64   LAST CARD IN DECK IS NO.
*     FAP
*XLIMIT
      COUNT   100
      LBL     XLIMIT
      ENTRY   XLIMIT F(X, XA, XB)
*
*
*                 ----ABSTRACT----
*
* TITLE - XLIMIT
*        FIND IF ARGUMENT FALLS INSIDE TWO LIMITING VALUES
*
*              XLIMIT  HAS VALUE  +0  IF ITS FIRST ARGUMENT LIES IN THE
*              INCLUSIVE RANGE DEFINED BY ITS SECOND AND THIRD
*              ARGUMENTS, HAS VALUE  -1  IF ITS FIRST ARGUMENT IS LESS
*              THAN THE SMALLER OF ITS OTHER TWO ARGUMENTS, OR  +1  IF
*              GREATER THAN THE LARGER OF THE OTHER TWO.  THE MODE OF
*              THE ARGUMENTS IS IMMATERIAL, AND  +0  IS CONSIDERED EQUAL
*              TO  -0  IN THE COMPARISONS.
*
* LANGUAGE   - FAP FUNCTION (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 25 REGISTERS
* SPEED      - 21 TO 33 MACHINE CYCLES
* AUTHOR     - S.M. SIMPSON, JUNE 1964
*
*
*                 ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NOT ANY
*       AND FORTRAN SYSTEM ROUTINES - NOT ANY
*
* FORTRAN USAGE
*     IZIFIN = XLIMITF(X,XA,XB)
*
*
* INPUTS
*
*    X         IS ANY MODE.
*
*    XA        IS SAME MODE AS X.
*
*    XB        IS SAME MODE AS X.
*
*
* OUTPUTS
*
*    IZIFIN    HAS VALUE  0,-1,+1  AS DESCRIBED IN ABSTRACT.
*
*
* EXAMPLES
*
* 1. USAGE   -         IZFIN1 = XLIMITF(3,2,4)
*                      IZFIN2 = XLIMITF(2,2,4)
*                      IZFIN3 = XLIMITF(4,2,4)
*                      IZFIN4 = XLIMITF(-0,+0,4)
*                      IZFIN5 = XLIMITF(+0,-0,4)
*                      IZFIN6 = XLIMITF(+0,-2,-0)
*                      IZFIN7 = XLIMITF(-0,-2,+0)
*                      IZFIN8 = XLIMITF(1,2,4)
*                      IZFIN9 = XLIMITF(5,2,4)
*    OUTPUTS - IZFIN1...IZFIN9 = 0,0,0,0,0,0,0,-1,+1
*
* 2. USAGE   - SAME AS EXAMPLE 1. BUT WITH THE THREE FUNCTION ARGUMENTS
*                 FLOATING POINT.
*    OUTPUTS - SAME AS EXAMPLE 1.
*
* 3. USAGE   - SAME AS EXAMPLE 1. BUT WITH REVERSED ORDER OF THE SECOND
*                 AND THIRD ARGUMENTS.
*    OUTPUTS - SAME AS EXAMPLE 1.
*
*
* PROGRAM FOLLOWS BELOW.
*
```

****
* XLIMIT *
****
(PAGE 2)

```
        BCI    1,XLIMIT
XLIMIT  STO    TEMP         FORCE XLO
        CLA    32765        INTO MQ,
        TLQ    *+2          XHI INTO XHI
        XCA
        STO    XHI
        CLA    TEMP         AND RESTORE X
        TNZ    XCA
        SSP                 (ADJUST FOR ZERO AMBIGUITY)
XCA     XCA
        TLQ    CLS1         BAD IF X LSTHN XLO  (+0 IS GRTHN= + OR -0)
        XCA
        TNZ    LDQ
        SSM                 (READJUST)
LDQ     LDQ    XHI
        TLQ    CLA1         BAD IF XHI LSTHN X  (+ OR -0 IS GRTHN= -0)
        PXD    0,0
        TRA    1,4
CLS1    CLS    KD1
        TRA    1,4
CLA1    CLA    KD1
        TRA    1,4
KD1     PZE    0,0,1
XHI     PZE    **,**,**
TEMP    PZE    **,**,**
        END
```

```
*       XLOCV   (SUBROUTINE)              9/4/64   LAST CARD IN DECK IS NO. 0099
*       FAP
*XLOCV
        COUNT   100
        LBL     XLOCV
        ENTRY   XLOCV (LOCV,X1,X2,...,XN)
*
*
*                    ----ABSTRACT----
*
* TITLE - XLOCV
*        CREATE VECTOR OF MACHINE ADDRESSES OF VARIABLES IN A LIST
*
*               XLOCV  IS A VARIABLE LENGTH-CALLING-SEQUENCE SUBROUTINE
*               WHOSE FIRST ARGUMENT IS ITS OUTPUT AND IS A VECTOR OF
*               FORTRAN-II INTEGERS GIVING THE MACHINE ADDRESSES OF ITS
*               REMAINING ARGUMENTS.
*
* LANGUAGE   - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709, 7090, 7094 (MAIN FRAME ONLY)
* STORAGE    - 24 REGISTERS
* SPEED      - ABOUT  15 + 20*N  MACHINE CYCLES ON THE  7090
*                 WHERE  N+1  IS THE ARGUMENT COUNT.
* AUTHOR     - S.M.SIMPSON, FEBRUARY 1964
*
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*       AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*     CALL XLOCV(LOCV,X1,X2,...,XN)
*
*
* INPUTS
*
*    X1         ARGUMENT WHOSE MACHINE ADDRESS IS TO BECOME LOCV(1)
*    X2         ARGUMENT WHOSE MACHINE ADDRESS IS TO BECOME LOCV(2)
*       ETC
*    XN         ARGUMENT WHOSE MACHINE ADDRESS IS TO BECOME LOCV(N)
*                   N SHOULD EXCEED ZERO
*
*
* OUTPUTS
*
*    LOCV(I)    I=1...N  CONTAINS THE MACHINE ADDRESSES OF  X1...XN
*
*
* EXAMPLES
*
* 1. INPUTS  - SUPPOSE  X  IS A VECTOR EQUIVALENT TO THE COMMON BLOCK
*                 (AT 32561 BASE 10)
*    USAGE   -       CALL XLOCV(LOCV1,X(1),X(10),X(7))
*                    CALL XLOCV(LOCV2,X)
*    OUTPUTS - LOCV1(1...3) = 32561, 32552, 32555     LOCV2(1)=32561
*
*
* PROGRAM FOLLOWS BELOW
*
        HTR     0                  XR1
        HTR     0                  XR4
        BCI     1,XLOCV
*
* ONLY ENTRY,  XLOCV(LOCV,X1,X2,...,XN)
*
 XLOCV  SXD     XLOCV-3,1
        SXD     XLOCV-2,4
        CLA     1,4                A(LOCV)
        STA     STO
        AXT     0,1
*
* GET NEXT ARGUMENT AND TEST FOR A TSX  X,0
*
 CAL    CAL     2,4                PICKS UP TSX  X1,0  FIRST
```

****
*   XLOCV   *
****
(PAGE 2)

```
       STA    TEMP
       ANA    AMASK
       LAS    TSXZ
       TRA    *+2              NO
       TRA    STORE            YES
*
* EXIT
*
       LXD    XLOCV-3,1        NO
       TRA    2,4
*
* STORE AN ADDRESS AND GO BACK FOR ANOTHER
*
 STORE CLA    TEMP
       ALS    18
 STO   STO    **,1             ** = A(LOCV)
       TXI    *+1,1,1
       TXI    CAL,4,-1
*
* CONSTANTS, TEMPORARIES
*
 AMASK OCT    777777700000
 TSXZ  TSX    0,0
 TEMP  PZE    **               ** = POSSIBLE ADDRESS
       END
```

```
************************
*     XLSHFT           *
************************
 REFER TO
     LSHFT
```

```
************************
*     XMLPLY           *
************************
 REFER TO
     MULPLY
```

```
************************
*     XMULK            *
************************
 REFER TO
     ADDK
```

```
************************
*     XMULKS           *
************************
 REFER TO
     ADDK
```

```
************************
*     XNAME            *
************************
 REFER TO
     LOCATE
```

```
************************
*     XNARGS           *
************************
 REFER TO
     LOCATE
```

```
************************
*     XNTHA            *
************************
 REFER TO
     NTHA
```

```
************************
*     XNTSUM           *
************************
 REFER TO
     INTSUM
```


```
************************
*     XLSHFT           *
************************
 REFER TO
     LSHFT
```

```
************************
*     XMLPLY           *
************************
 REFER TO
     MULPLY
```

```
************************
*     XMULK            *
************************
 REFER TO
     ADDK
```

```
************************
*     XMULKS           *
************************
 REFER TO
     ADDK
```

```
************************
*     XNAME            *
************************
 REFER TO
     LOCATE
```

```
************************
*     XNARGS           *
************************
 REFER TO
     LOCATE
```

```
************************
*     XNTHA            *
************************
 REFER TO
     NTHA
```

```
************************
*     XNTSUM           *
************************
 REFER TO
     INTSUM
```

```
*       XOOZE  (FUNCTION)                9/4/64   LAST CARD IN DECK IS NO. 0060
*       FAP
*XOOZE
        COUNT   50
        LBL     XOOZE
        ENTRY   XOOZE  F(INT)
*
*
*                     ----ABSTRACT----
*
* TITLE - XOOZE
*         DETERMINE WHETHER FORTRAN-II INTEGER IS EVEN OR ODD
*
*               XOOZE FUNCTION RETURNS 1 IF ITS ARGUMENT IS ODD, ZERO IF
*               EVEN.
*
*
* LANGUAGE   - FAP FUNCTION (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709,7090,7094 (MAIN FRAME ONLY)
* STORAGE    - 4 REGISTERS
* SPEED      - 4 MACHINE CYCLES
* AUTHOR     - S.M.SIMPSON,JR.  APRIL 1964
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NOT ANY)
*        AND FORTRAN SYSTEM ROUTINES - (NOT ANY)
*
* FORTRAN USAGE
*      IZIFEV=XOOZEF(INT)
*
*
* INPUT
*
*    INT        IS A FORTRAN-II INTEGER
*
*
* OUTPUT
*
*    IZIFEV     = 0 IF INT IS EVEN
*               = 1 IF INT IS ODD
*
*
* EXAMPLES
*
* 1. USAGE             DIMENSION  IZIFEV(9)
*                      DO 10  I=1,9
*                   10 IZIFEV(I)=XOOZEF(I-5)
*    OUTPUTS - IZIFEV(1...9) = 0,1,0,1,0,1,0,1,0
*
*
*
* PROGRAM FOLLOWS BELOW
*
*
        BCI     1,XOOZE
 XOOZE  ANA     MASK
        TRA     1,4
 MASK   OCT     000001000000
        END
```

```
*       XREMAV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0111
*       FAP
*XREMAV
        COUNT    100
        LBL      XREMAV
        ENTRY    XREMAV (IX,LIX,IXAVG,IXNULD)
*
*                    ----ABSTRACT----
*
* TITLE - XREMAV
*         REMOVE THE MEAN FROM A FIXED VECTOR
*
*              XREMAV COMPUTES THE AVERAGE VALUE OF A FIXED POINT
*              VECTOR (ROUNDING THE AVERAGE TO NEAREST INTEGER), AND
*              THEN SETS AN OUTPUT VECTOR WITH ELEMENTS EQUAL TO THOSE
*              OF THE INPUT VECTOR MINUS THE AVERAGE.  THE OUTPUT
*              VECTOR MAY REPLACE THE INPUT VECTOR.  THE AVERAGE IS
*              ALSO AN OUTPUT QUANTITY.
*
*              THERE IS NO DANGER OF FIXED POINT OVERFLOW.
*
* LANGUAGE    - FAP SUBROUTINE (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 31 REGISTERS
* SPEED       - 125 + 19*L  MACHINE CYCLES ON 7090,    L = VECTOR LENGTH
*               132 + 19*L  MACHINE CYCLES ON 709
* AUTHOR      - S.M. SIMPSON, SEPTEMBER 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - XAVRGR
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL XREMAV(IX,LIX,IXAVG,IXNULD)
*
* INPUTS
*
*    IX(I)     I=1...LIX IS A FIXED VECTOR (MUST BE FORTRAN-II INTEGERS)
*
*    LIX       SHOULD EXCEED ZERO
*
* OUTPUTS      STRAIGHT RETURN WITH NO OUTPUTS IF LIX LSTHN 1.
*
*    IXAVG     IS = (1/LIX) * ( SUM(FROM I=1 TO LIX) OF X(I) )
*                   ROUNDED TO NEAREST INTEGER.  IT IS COMPUTED IN
*                   A MANNER WHICH ELIMINATES THE POSSIBILITY OF
*                   FIXED POINT OVERFLOW.
*
*    IXNULD(I) I=1...LIX  IS IXNULD(I) = IX(I) - IXAVG
*
*              EQUIVALENCE(IXNULD,IX) IS PERMITTED.
*
* EXAMPLES
* 1. INPUTS   - IX1(1...5) = 90000, 91000, 92000, 93000, 94000
*               IX2(1...5) =  1, 2, 3, 4, 5
*               IXAVG6 = IXNLD6 = -999
*
*    USAGE    -       CALL XREMAV(IX1, 5, IXAVG1, IXNLD1)
*                     CALL XREMAV(IX2, 5, IXAVG2, IXNLD2)
*                     CALL XREMAV(IX2, 4, IXAVG3, IXNLD3)
*                     CALL XREMAV(IX2, 5, IXAVG4, IX2)
*                     CALL XREMAV(IX1, 1, IXAVG5, IXNLD5)
*                     CALL XREMAV(IX1, 0, IXAVG6, IXNLD6)
*
*    OUTPUTS - IXAVG1 = 92000  IXNLD1(1...5) = -2000,-1000,0,1000,2000
*              IXAVG2 =  3     IXNLD2(1...5) = -2, -1, 0, 1, 2
*              IXAVG3 =  3     IXNLD3(1...4) = -2, -1, 0, 1
*              IXAVG4 =  3        IX2(1...5) = -2, -1, 0, 1, 2
*              IXAVG5 = 90000     IXNLD5(1) =  0
*              IXAVG6 = IXNLD6 = -999   (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
```

```
* TRANSFER VECTOR CONTAINS XAVRGR(IX,LIX,IXAVG)
       HTR     0             XR4
       BCI     1,XREMAV
* ONLY ENTRY.  XREMAV(IX,LIX,IXAVG,IXNULD)
XREMAV SXD     XREMAV-2,4
 K1    CLA     1,4           A(IX)
       STA     TSX1
       ADD     K1            A(IX)+1
       STA     GET
       CLA     3,4           A(IXAVG)
       STA     TSX3
       STA     SUB
       CLA     4,4
       ADD     K1            A(IXNULD)+1
       STA     STORE
       CLA*    2,4           LIX
       TMI     LEAVE
       STD     LIX
       NZT     LIX
       TRA     LEAVE
* COMPUTE IXAVG
       TSX     $XAVRGR,4
 TSX1  TSX     **,0          **=A(IX)
       TSX     LIX,0
 TSX3  TSX     **,0          **=A(IXAVG)
* MEAN REMOVAL LOOP
       LXD     LIX,4
 GET   CLA     **,4          **=A(IX)+1
 SUB   SUB     **             **=A(IXAVG)
 STORE STO     **,4           **=A(IXNULD)+1
       TIX     GET,4,1
* EXIT
 LEAVE LXD     XREMAV-2,4
       TRA     5,4
* TEMPORARY
 LIX   PZE     0,0,**        **=LIX
       END
```

```
************************
*    XRFLEC            *
************************
 REFER TO
     REFLEC


************************
*    XSAME             *
************************
 REFER TO
     SAME


************************
*    XSMDEV            *
************************
 REFER TO
     SUMDFR


************************
*    XSMDFR            *
************************
 REFER TO
     SUMDFR
```


```
************************
*    XRFLEC            *
************************
 REFER TO
     REFLEC


************************
*    XSAME             *
************************
 REFER TO
     SAME


************************
*    XSMDEV            *
************************
 REFER TO
     SUMDFR


************************
*    XSMDFR            *
************************
 REFER TO
     SUMDFR
```

```
*     XSPECT (SUBROUTINE)                9/29/64   LAST CARD IN DECK IS NO. 0238
*     LABEL
CXSPECT
      SUBROUTINE XSPECT(XCOR,N,COSTAB,SINTAB,M,JMIN,JMAX,CSP,SSP,
     1                 SPACE,ERR)
C
C                    ----ABSTRACT----
C
C TITLE - XSPECT
C       FAST COSINE, SINE TRANSFORMS OF CROSS-CORRELATION FUNCTIONS
C
C             XSPECT PRODUCES A HI-SPEED CROSS-POWER (OR ENERGY) DENSITY
C             SPECTRUM (OR PORTION THEREOF) FROM AN N-LAG CROSS-
C             CORRELATION FUNCTION, XC(I) I=-N,-N+1,...,N , IN TERMS
C             OF THE REAL AND IMAGINARY PARTS
C
C                           N
C               CS(J)  =   SUM  ( XC(I)*COS(I*J*PI/M) )
C                          I=-N
C
C                           N
C               SS(J)  =   SUM  ( XC(I)*SIN(I*J*PI/M) )
C                          I=-N
C
C                     FOR J = JMIN,JMIN+1,...,JMAX
C               WHERE
C                     PI = 3.14159265
C                     N,M,JMIN AND JMAX ARE INPUT PARAMETERS
C                     COS(J*PI/M)  AND  SIN(J*PI/M) J=0,1,...,M  ARE
C                         REQUIRED AS INPUT TABLES
C                     0 LSTHN= JMIN LSTHN JMAX LSTHN= M
C
C             SPEED IS ATTAINED BY
C                  1. (FOR M LSTHN= N)
C                     -COLLAPSING XC(I) INTO THE RANGE -M TO +M
C                     -SPLITTING THE COLLAPSED CORRELATION INTO ODD AND
C                        EVEN PARTS AND RESPLITTING THESE INTO THEIR
C                        ODD AND EVEN SUBPARTS.
C
C                  2. USING THE HI-SPEED LOOPING LOGIC OF SUBROUTINE
C                     COSISP TO PERFORM THE TRANSFORMS OF THE SHORTENED
C                     SUBPARTS (LENGTH = M/2)
C
C             2*M+4 TEMPORARY REGISTERS ARE NEEDED UNLESS USER IS
C             WILLING TO SACRIFICE THE CROSS-CORRELATION FOR SCRATCH
C             (TEMPORARIES NOT REQUIRED IF M GRTHN N)
C
C LANGUAGE  - FORTRAN II SUBROUTINE
C EQUIPMENT - 709 OR 7090 (MAIN FRAME ONLY)
C STORAGE   - 523 REGISTERS
C SPEED     - FOR M LSTHN= N   - 36*(JMAX-JMIN+1)*M   MACHINE CYCLES
C             FOR M GRTHN  N   - 72*(JMAX-JMIN+1)*N   MACHINE CYCLES
C AUTHOR    - S.M. SIMPSON JR., NOV 1961
C
C                    ----USAGE----
C
C TRANSFER VECTOR CONTAINS ROUTINES -   SPLIT, COSISP, REFIT, KOLAPS,
C                                       CHPRTS
C       AND FORTRAN SYSTEM ROUTINES -   XLOC
C
C FORTRAN USAGE
C      CALL XSPECT(XCOR,N,COSTAB,SINTAB,M,JMIN,JMAX,CSP,SSP,SPACE,ERR)
C
C INPUTS
C
C    XCOR(I)   I=-N+1,...,N+1 CONTAINS  XC(J) J= -N,...,N  RESPECTIVELY
C                 (THIS FORMAT PLACES THE ZERO LAG CORRELATION,
C                  XC(0), IN XCOR(1)
C
C    N         MUST EXCEED ZERO
C
C    COSTAB(I) I=1...M+1  CONTAINS COS(J*PI/M) J=0,1,...,M
C
C    SINTAB(I) I=1...M+1  CONTAINS SIN(J*PI/M) J=0,1,...,M
C
```

```
      C    M          MUST EXCEED ZERO
      C
      C    JMIN       MUST BE NON-NEGATIVE
      C
      C    JMAX       MUST EXCEED JMIN AND BE LSTHN= M
      C
      C    SPACE(I)   IS NOT USED IF M EXCEEDS N
      C               IS A BLOCK OF 2*M+4 TEMPORARIES IF  M LSTHN= N
      C                 ORDINARILY  (SPACE NOT EQUIVALENT TO XCOR) THIS BLOCK
      C                    CONSISTS OF SPACE(I) I=1,2,....,2*M+4
      C                 HOWEVER, IF SPACE AND XCOR ARE EQUIVALENT, THIS BLOCK
      C                    CONSISTS OF SPACE(I) = XCOR(I)  I=-M+1,....,M+4
      C                    (NOTE THAT IF M=N, 3 REGISTERS BEYOND XCOR(N+1)
      C                    ARE USED)
      C
      C OUTPUTS
      C
      C    CSP(I)     I=1,2,...JMAX-JMIN+1  CONTAINS CS(J) J=JMIN...JMAX AS
      C                  DEFINED IN ABSTRACT.
      C
      C    SSP(I)     I=1,2,...JMAX-JMIN+1  CONTAINS SS(J) J=JMIN...JMAX AS
      C                  DEFINED IN ABSTRACT
      C
      C    ERR        = 0.0 NORMALLY
      C               = 1.0 IF N,M,JMIN OR JMAX IS ILLEGAL (PROGRAM EXITS
      C                     WITHOUT COMPUTING SPECTRUM IN THIS CASE)
      C
      C EXAMPLES
      C
      C 1. COMPLETE SPECTRUM, NOT TRYING TO SAVE SPACE, M LSTHN N
      C    INPUTS  - XCR(1...7) = -36.,-27.,-18.,2.,22.,33.,44.          N=3
      C              COSTAB(1...3) = 1.,0.,-1.  SINTAB(1...3) = 0.,1.,0. M=2
      C              JMIN=0  JMAX=M
      C    USAGE   -       CALL XSPECT(XCR(4),N,COSTAB,SINTAB,M,JMIN,JMAX,
      C                                CSP,SSP,SPACE,ERR)
      C    OUTPUTS - ERR=0.
      C              CSP(1...3) = 20.,-4.,-4.  SSP(1...3) = 0.,-40.,0.
      C
      C 2. COMPLETE SPECTRUM SAVING SPACE
      C    INPUTS  - SAME AS EXAMPLE 1.
      C    USAGE   -       CALL XSPECT(XCR(4),N,COSTAB,SINTAB,M,JMIN,JMAX,CSP,
      C                                SSP,XCR(4),ERR)
      C    OUTPUTS - SAME AS EXAMPLE 1.  (BUT XCR(2...9) WILL HAVE BEEN
      C                DESTROYED)
      C
      C 3. PARTIAL SPECTRUM
      C    INPUTS  - SAME AS EXAMPLE 1.  EXCEPT JMIN=1
      C    USAGE   - SAME AS EXAMPLE 1.
      C    OUTPUTS - ERR=0.
      C              CSP(1...2) = -4.,-4.  SSP(1...2) = -40.,0.
      C
      C 4. FINER GRAINED SPECTRUM, M GRTHN N
      C    INPUTS  - SAME AS EXAMPLE 1. EXCEPT  M=JMAX=4         AND
      C              COSTAB(1...5) = 1.,.70711,0.,-.70711,-1.
      C              SINTAB(1...5) = 0.,.70711,1.,.70711,0.
      C    USAGE   - SAME AS EXAMPLE 1.
      C    OUTPUTS - ERR=0.   AND
      C              CSP(1...5) = 20.,-.82844,-4.,4.82844,-4.
      C              SSP(1...5) = 0.,144.85320,-40.,24.85320,0.
      C
      C 5. ERROR EXITS WITH NO COMPUTATION
      C    USAGE   -       CALL XSPECT(XCOR,-1,COSTAB,SINTAB,3,0,3,
      C                                CSP,SSP,SPACE,ERR1)
      C                    CALL XSPECT(XCOR,2,COSTAB,SINTAB,0,0,3,
      C                                CSP,SSP,SPACE,ERR2)
      C                    CALL XSPECT(XCOR,2,COSTAB,SINTAB,3,-1,3,
      C                                CSP,SSP,SPACE,ERR3)
      C                    CALL XSPECT(XCOR,2,COSTAB,SINTAB,3,0,4,
      C                                CSP,SSP,SPACE,ERR4)
      C                    CALL XSPECT(XCOR,2,COSTAB,SINTAB,3,2,2,
      C                                CSP,SSP,SPACE,ERR5)
      C    OUTPUTS - ERR1=1. (ILLEGAL N)      ERR2=1. (ILLEGAL M)
      C              ERR3=1. (ILLEGAL JMIN)  ERR4=1. (ILLEGAL JMAX)
      C              ERR5=1. (ILLEGAL JMAX)
      C
```

```
C PROGRAM FOLLOWS BELOW
C
C
C ARGUMENT DIMENSIONS
      DIMENSION XCOR(15000),COSTAB(15000),SINTAB(15000)
      DIMENSION CSP(15000),SSP(15000),SPACE(15000)
C  CHECK CONDITIONS ON N,JMIN,JMAX,M
      MM=M
      IF (N)  15,10,10
   10 IF (JMIN)  15,12,12
   12 IF (JMAX-JMIN)  15,15,14
   14 IF (JMAX-MM)  17,17,15
C BAD
   15 ERR=1.0
      GO TO 99
C OK
   17 ERR=0.0
C CHECK IF FOLDING ETC IS VALID (NOT IF M GREATER THAN N)
      IF (MM-N)  30,30,20
C NOTE,  IN WHAT FOLLOWS WE OBTAIN EFFECTIVE NEGATIVE INDICES SINCE
C     X(2)  = X(2)
C     X(1)  = X(1)
C     X(0)  = X(32768)
C     X(-1) = X(32768-1)
C           ETC
C     X(-J) = X(32768-J)
C FOLDING IS NOT POSSIBLE, COMPUTE SPECTRUM DIRECTLY AND EXIT
C
C FIRST SPLIT THE CORRELATION ON TOP OF ITSELF
   20 JJ=32768-(N-1)
      NN=2*N+1
      CALL SPLIT(XCOR(JJ),NN,1.0,XCOR(JJ),XCOR(2))
C THEN FEED THE PARTS TO COSISP. (NOTE SHIFT OF ORIGIN FOR ANTISYM PART
C WHICH MAKES ITS FIRST ELEMENT NON-ZERO - BUT IT IS MULTIPLIED
C    BY SIN(0).)
      CALL COSISP(XCOR(JJ),XCOR(JJ),XCOR,XCOR,N,COSTAB,SINTAB,MM,
     1JMIN,JMAX,1.0,CSP,SSP)
C THEN PUT THE CORRELATION BACK TOGETHER AND EXIT.
      CALL REFIT(XCOR(JJ),NN,1.0,XCOR(JJ),XCOR(2))
      GO TO 99
C FOLDING IS POSSIBLE.  SETUP
   30 LCOL=2*MM+1
      LSYM=MM+1
      LSMSM=(MM+2)/2
C IS FOLDING TO TAKE PLACE ON TOP OF CORRELATION BLOCK
      IF (XLOCF(SPACE)-XLOCF(XCOR))  32,34,32
C NO, SET UP FOR THIS CASE
   32 ISS=1
      IAS=1+LSMSM
      IMID=MM+1
      ISA=MM+3
      IAA=MM+3+LSMSM
      IZER3=2*MM+4
      GO TO 40
C YES, SET UP
   34 ISS=32768-(MM-1)
      IAS=32768-(MM-1)+LSMSM
      IMID=1
      ISA=3
      IAA=3+LSMSM
      IZER3=MM+4
      GO TO 40
C THEN COLLAPSE THE CORRELATION INTO THE RANGE -M TO +M
C   (IGNORE ERROR RETURN)
   40 CALL KOLAPS(XCOR,N,1.0,M,SPACE(IMID),DUMMY)
C THEN SPLIT THE COLLAPSED CORRELATION ON TOP OF ITSELF
      CALL SPLIT(SPACE(ISS),LCOL,1.0,SPACE(ISS),SPACE(IMID+1))
C THEN SHIFT THE ANTISYMMETRIC PART UP TWO NOTCHES
C           AND FILL IN THREE ZEROES
      DO 45  I=1,MM
      J=IMID+1+MM-I
   45 SPACE(J+2)=SPACE(J)
      SPACE(IMID+1)=0.0
      SPACE(IMID+2)=0.0
      SPACE(IZER3)=0.0
```

```
C NOW SPLIT THE SYMMETRIC PART ON TOP OF ITSELF
      CALL SPLIT(SPACE(ISS),LSYM,1.0,SPACE(ISS),SPACE(IAS))
C THEN REVERSE SYMSYM,REVERSE AND CHANGE SIGNS OF ANTSYM
      CALL CHPRTS(SPACE(ISS),SPACE(IAS),LSYM)
C NOW SPLIT THE ANTISYMMETRIC PART (EXTENDED) ON TOP OF ITSELF
      CALL SPLIT(SPACE(ISA),LSYM,1.0,SPACE(ISA),SPACE(IAA))
C THEN REVERSE SYMANT, REVERSE AND CHANGE SIGNS OF ANTANT
      CALL CHPRTS(SPACE(ISA),SPACE(IAA),LSYM)
C FINALLY FEED THE FOUR PARTS TO COSISP AND THEN EXIT
      LMONE=LSMSM-1
      CALL COSISP(SPACE(ISS),SPACE(IAS),SPACE(ISA),SPACE(IAA),
     1LMONE,COSTAB,SINTAB,MM,JMIN,JMAX,1.0,CSP,SSP)
   99 RETURN
      END
```

```
************************
*    XSQDEV            *
************************
 REFER TO
     XSQDFR
```



```
************************
*    XSQDEV            *
************************
 REFER TO
     XSQDFR
```

```
*      XSQDFR (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0112
*      FAP
*XSQDFR
       COUNT   100
       LBL     XSQDFR
       ENTRY   XSQDFR (IX,IY,LXY,ISSXMY)
       ENTRY   XSQDEV (IX,IXBASE,LIX,ISSXMB)
*
*                    ----ABSTRACT----
*
* TITLE - XSQDFR WITH SECONDARY ENTRY XSQDEV
*        SUM SQUARE DIF. OF FXD. VECTOR FROM ANOTHER OR FROM A CONSTANT
*
*              XSQDFR SUMS THE SQUARES OF THE DIFFERENCES BETWEEN THE
*              ELEMENTS OF TWO FIXED (FORTRAN-II) VECTORS
*
*              XSQDEV SUMS THE SQUARES OF THE DIFFERENCES BETWEEN THE
*              ELEMENTS CF A FIXED VECTOR AND A CONSTANT.
*
* LANGUAGE   - FAP SUBROUTINES  (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 37   REGISTERS
* SPEED      -        7090      709
*              38 + (28.6 OR 32.8)*LX  MACHINE CYCLES,LX=VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - (NONE)
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL XSQDFR(IX,IY,LXY,ISSXMY)
*     CALL XSQDEV(IX,IXBASE,LIX,ISSXMB)
*
* INPUTS
*
*    IX(I)     I=1...LXY  ARE FORTRAN-II INTEGERS, INPUT TO XSQDFR
*    IY(I)     I=1...LXY  ARE FORTRAN-II INTEGERS, INPUT TO XSQDFR
*    LXY       SHOULD EXCEED 0
*
*    IX(I)     I=1...LIX  ARE FORTRAN-II INTEGERS INPUT TO XSQDEV
*    IXBASE           IS A FORTRAN-II INTEGER INPUT TO XSQDEV
*    LIX       SHOULD EXCEED 0
*
* OUTPUTS      STRAIGHT RETURN WITH NO ACTION IF LXY OR LIX LSTHN 1
*
*    ISSXMY    IS SUM(FROM I=1 TO LXY) OF (IX(I)-IY(I))*(IX(I)-IY(I))
*
*    ISSXMB    IS SUM(FROM I=1 TO LIX) OF (IX(I)-IXBASE)*(IX(I)-IXBASE)
*
*              DANGER OF OVERFLOW, NOT TESTED FOR BY EITHER ENTRY.
*
*              EQUIVALENCE(ISSXMY,ANY INPUT),(ISSXMB, ANY INPUT)
*                 IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...3) =  1, 2, 3  IY(1...3)= 3, 4, 5     ISDIF2=0
*    USAGE   -        CALL XSQDFR(IX,IY,3,ISDIF1)
*                     CALL XSQDEV(IX, 3,3,ISDEV1)
*                     CALL XSQDFR(IX,IY,1,IX)
*                     CALL XSQDFR(IX,IY,0,ISDIF2)
*    OUTPUTS - ISDIF1 = 12, ISDEV1 = 5, IX(1)= 4,ISDIF2 = 0 (NO OUTPUT)
*
* PROGRAM FOLLOWS BELOW
*
*
* NO TRANSFER VECTOR
       HTR     0              XR4
       BCI     1,XSQDFR
* PRINCIPAL ENTRY.  XSQDFR(IX,IY,LXY,ISSXMY)
XSQDFR CLA     2,4
       ADD     K1             A(IY)+1
       STA     SUB
```

****
*   XSQDFR   *
****
(PAGE 2)

```
        CLA     SUB
 SETUP  STO     SUBTR
        SXD     XSQDFR-2,4
 K1     CLA     1,4
        ADD     K1            A(IX)+1
        STA     GET
        CLA*    3,4           LXY
        TMI     LEAVE
        PDX     0,4
        TXL     LEAVE,4,0
        STZ     TEMP1
* LOOP
 GET    CLA     **,4          **=A(IX)+1
 SUBTR  SUB     **,**         =SUB A(IY)+1,4          OR SUB A(IXBASE)
        STO     TEMP2
        XCA
        MPY     TEMP2
        ALS     17
        ADD     TEMP1
        STO     TEMP1
        TIX     GET,4,1
* STORE RESULT
        LXD     XSQDFR-2,4
        STO*    4,4
* EXIT
 LEAVE  LXD     XSQDFR-2,4
        TRA     5,4
* SECOND ENTRY.  XSQDEV(IX,IXBASE,LIX,ISSXMB)
XSQDEV  CLA     2,4           A(IXBASE)
        STA     SUBXB
        CLA     SUBXB
        TRA     SETUP
* CONSTANTS, TEMPORARIES
 SUB    SUB     **,4          **=A(IY)+1
 SUBXB  SUB     **            **=A(IXBASE)
 TEMP1  PZE     **,**,**      SUM
 TEMP2  PZE     **,**,**      DIFFERENCES
        END
```

```
*       XSQRUT (SUBROUTINE)               9/29/64   LAST CARD IN DECK IS NO. 0102
*       FAP
*XSQRUT
        COUNT   100
        LBL     XSQRUT
        ENTRY   XSQRUT (IX,LIX,IXSQRT)
*
*                     ----ABSTRACT----
*
* TITLE - XSQRUT
*          SQUARE ROOT OF A FIXED VECTOR WITH ROUNDING
*
*               XSQRUT FORMS A FIXED VECTOR WHOSE ELEMENTS ARE THE
*               SQUARE ROOTS (ROUNDED TO FORTRAN-II INTEGERS) OF ANOTHER
*               FIXED VECTOR (ALSO FORTRAN-II INTEGERS).  OUTPUT MAY
*               REPLACE INPUT.
*
* LANGUAGE    - FAP SUBROUTINE  (FORTRAN-II COMPATIBLE)
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE     - 37 REGISTERS
* SPEED       - ABOUT  78 + 250*LX MACHINE CYCLES,  LX = VECTOR LENGTH
* AUTHOR      - S.M. SIMPSON, AUGUST 1963
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - FIXVR
*       AND FORTRAN SYSTEM ROUTINES - SQRT (FUNCTION)
*
* FORTRAN USAGE
*     CALL XSQRUT(IX,LIX,IXSQRT)
*
* INPUTS
*
*    IX(I)      I=1...LIX  IS A NON-NEGATIVE FORTRAN-II INTEGER VECTOR
*
*    LIX        SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUT IF LIX LSTHN 1
*
*    IXSQRT(I) I=1...LIX  IS IXSQRT(I) =XFIXRF(SQRTF(FLOATF(IX(I))))
*                  WHERE XFIXRF IMPLIES FIXING WITH ROUNDING. NEGATIVE
*                  VALUES OF IX(I) ARE TREATED AS THOUGH THEY WERE
*                  POSITIVE.
*
*               EQUIVALENCE (IXSQRT,IX)  IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...5) = 1,2,3,4,5  IY(1...5) = 100,-200,300,-400,500
*               ISQRT3 = 0
*    USAGE   -        CALL XSQRUT(IX,5,ISQRT1)
*                     CALL XSQRUT(IY,5,ISQRT2)
*                     CALL XSQRUT(IY,1,IY)
*                     CALL XSQRUT(IX,0,ISQRT3)
*    OUTPUTS - ISQRT1(1...5) = 1,1,2,2,2
*               ISQRT2(1...5) = 10,14,17,20,22
*               IY(1) = 10      ISQRT3 = 0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS SQRT, FIXVR
        HTR     0              XR1
        HTR     0              XR4
        BCI     1,XSQRUT
* ONLY ENTRY.   XSQRUT (IX,LIX,IXSQRT)
XSQRUT  SXD     XSQRUT-3,1
        SXD     XSQRUT-2,4
 K1     CLA     1,4
        ADD     K1             A(IX)+1
        STA     GET
        CLA     3,4            A(IXSQRT)
        STA     TSX1
        STA     TSX3
        ADD     K1             A(IXSQRT)+1
```

```
***********************
*    XSQRUT          *
***********************
(PAGE  2)
```


```
        STA     STORE
        CLA     2,4          A(LIX)
        STA     TSX2
        CLA*    2,4          LIX
        TMI     LEAVE
        PDX     0,1
        TXL     LEAVE,1,0
* LOOP
 GET    CLA     **,1         **=A(IX)+1
        LRS     18
        ORA     OCTK
        FAD     OCTK
        SSP
        TSX     $SQRT,4
 STORE  STO     **,1         **=A(IXSQRT)+1
        TIX     GET,1,1
* THEN FIX WITH ROUNDING
        TSX     $FIXVR,4
 TSX1   TSX     **,0         **=A(IXSQRT)
 TSX2   TSX     **,0         **=A(LIX)
 TSX3   TSX     **,0         **=A(IXSQRT)
* EXIT
 LEAVE  LXD     XSQRUT-2,4
        LXD     XSQRUT-3,1
        TRA     4,4
* CONSTANTS
 OCTK   OCT     233000000000
        END
```

```
************************
*    XSQSUM            *
************************
 REFER TO
     SQRSUM

************************
*    XSQUAR            *
************************
 REFER TO
     SQUARE

************************
*    XSTEPC            *
************************
 REFER TO
     DELTA

************************
*    XSTEPL            *
************************
 REFER TO
     DELTA

************************
*    XSTEPR            *
************************
 REFER TO
     DELTA
```


```
************************
*    XSQSUM            *
************************
 REFER TO
     SQRSUM

************************
*    XSQUAR            *
************************
 REFER TO
     SQUARE

************************
*    XSTEPC            *
************************
 REFER TO
     DELTA

************************
*    XSTEPL            *
************************
 REFER TO
     DELTA

************************
*    XSTEPR            *
************************
 REFER TO
     DELTA
```

```
***********************
*    XSTLIN           *
***********************
 REFER TO
     SETLIN

***********************
*    XSUBK            *
***********************
 REFER TO
     ADDK

***********************
*    XSUBKS           *
***********************
 REFER TO
     ADDK

***********************
*    XSUM             *
***********************
 REFER TO
     SUM

***********************
*    XVDRBV           *
***********************
 REFER TO
     XVDVBV
```


```
***********************
*    XSTLIN           *
***********************
 REFER TO
     SETLIN

***********************
*    XSUBK            *
***********************
 REFER TO
     ADDK

***********************
*    XSUBKS           *
***********************
 REFER TO
     ADDK

***********************
*    XSUM             *
***********************
 REFER TO
     SUM

***********************
*    XVDRBV           *
***********************
 REFER TO
     XVDVBV
```

```
*      XVDVBV (SUBROUTINE)              9/29/64   LAST CARD IN DECK IS NO. 0108
*      FAP
*XVDVBV
       COUNT    100
       LBL      XVDVBV
       ENTRY    XVDVBV (IX,IY,LXY,IXDVBY)
       ENTRY    XVDRBV (IX,IY,LXY,IXDVBY)
*
*                    ----ABSTRACT----
*
* TITLE - XVDVBV WITH SECONDARY ENTRY XVDRBV
*         DIVIDE ELEMENTS OF TWO FIXED VECTORS WITH OR WITHOUT ROUNDING
*
*               XVDVBV DIVIDES THE ELEMENTS OF ONE FIXED VECTOR BY THOSE
*               OF ANOTHER, TRUNCATING FRACTIONAL PARTS.
*               XVDRBV ROUNDS FRACTIONAL PARTS.
*
*               OUTPUT MAY REPLACE EITHER INPUT VECTOR.
*
* LANGUAGE   - FAP SUBROUTINES (FORTRAN-II COMPATIBLE)
* EQUIPMENT  - 709 OR 7090 (MAIN FRAME ONLY)
* STORAGE    - 34 REGISTERS
* SPEED      -              7090  709
*               XVDVBV  41 + (49  OR 56)*LXY   MACHINE CYCLES,
*               XVDRBV  43 + (51  OR 58)*LXY      LXY = VECTOR LENGTH
* AUTHOR     - S.M. SIMPSON, AUGUST 1963
*
*                    ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - XDIV, XDIVR
*       AND FORTRAN SYSTEM ROUTINES - (NONE)
*
* FORTRAN USAGE
*     CALL XVDVBV(IX,IY,LXY,IXDVBY)
*     CALL XVDRBV(IX,IY,LXY,IXDVBY)
*
* INPUTS
*
*    IX(I)      I=1...LXY  IS A FORTRAN-II INTEGER VECTOR
*
*    IY(I)      I=1...LXY  IS A FORTRAN-II INTEGER VECTOR, NONE OF WHICH
*                              = 0
*
*    LXY        SHOULD EXCEED 0
*
* OUTPUTS       STRAIGHT RETURN WITH NO OUTPUTS IF LXY LSTHN 1
*
*    IXDVBY(I) I=1...LXY  IS  IXDVBY(I) = IX(I)/IY(I) ,
*                     TRUNCATED IF  XVDVBV IS USED,
*                     ROUNDED IF  XVDRBV IS USED.
*
*               DIVISIONS ARE PERFORMED BY XDIV AND XDIVR FUNCTIONS IN
*               WHICH DIVISION BY ZERO GIVES RESULT EQUAL TO NUMERATOR
*               AND THE DIVIDE CHECK INDICATOR IS NOT TURNED ON.
*
*               EQUIVALENCE (IXDVBY, IX OR IY) IS PERMITTED.
*
* EXAMPLES
*
* 1. INPUTS  - IX(1...5) = 1,2,3,4,5  IY(1...5) = 4,4,4,4,4  IZ=0
*    USAGE   -        CALL XVDVBV(IX,IY,5,IQ1)
*                     CALL XVDRBV(IX,IY,5,IQ2)
*                     CALL XVDVBV(IX,IY,1,IY)
*                     CALL XVDVBV(IX,IY,-1,IZ)
*    OUTPUTS - IQ1(1...5) = 0,0,0,1,1  IQ2(1...5) = 0,1,1,1,1
*              IY(1) = 0  IZ = 0  (NO OUTPUT CASE)
*
* PROGRAM FOLLOWS BELOW
*
*
* TRANSFER VECTOR CONTAINS XDIV AND XDIVR FUNCTIONS
       HTR      0             XR1
       HTR      0             XR4
       BCI      1,XVDVBV
* PRINCIPAL ENTRY.  XVDVBV(IX,IY,LXY,IXDVBY)
```

```
XVDVBV CLA     DIV
 SETUP STO     VARY
       SXD     XVDVBV-3,1
       SXD     XVDVBV-2,4
 K1    CLA     1,4
       ADD     K1             A(IX)+1
       STA     GETN
       CLA     2,4
       ADD     K1             A(IY)+1
       STA     GETD
       CLA     4,4
       ADD     K1             **=A(IXDVBY)+1
       STA     STORE
       CLA*    3,4            LXY
       TMI     LEAVE
       PDX     0,1
       TXL     LEAVE,1,0
* DIVISION LOOP
 GETN  CLA     **,1           **=A(IX)+1
 GETD  LDQ     **,1           **=A(IY)+1
 VARY  TSX     **,4           **=$XDIV OR $XDIVR
 STORE STO     **,1           **=A(IXDVBY)+1
       TIX     GETN,1,1
* EIXT
 LEAVE LXD     XVDVBV-3,1
       LXD     XVDVBV-2,4
       TRA     5,4
* SECOND ENTRY.  XVDRBV(IX,IY,LXY,IXDVBY)
XVDRBV CLA     DIVR
       TRA     SETUP
* CONSTANTS
 DIV   TSX     $XDIV,4
 DIVR  TSX     $XDIVR,4
       END
```

```
************************
*     XVMNSV           *
************************
 REFER TO
     VPLUSV
```

```
************************
*     XVPLSV           *
************************
 REFER TO
     VPLUSV
```

```
************************
*     XVTMSV           *
************************
 REFER TO
     VTIMSV
```

```
************************
*     XWHICH           *
************************
 REFER TO
     WHICH
```


```
************************
*     XVMNSV           *
************************
 REFER TO
     VPLUSV
```

```
************************
*     XVPLSV           *
************************
 REFER TO
     VPLUSV
```

```
************************
*     XVTMSV           *
************************
 REFER TO
     VTIMSV
```

```
************************
*     XWHICH           *
************************
 REFER TO
     WHICH
```

```
*       ZEFBCD (FUNCTION)                  9/8/64   LAST CARD IN DECK IS NO. 0128
*       FAP
*ZEFBCD
        COUNT    100
        LBL      ZEFBCD
        ENTRY    ZEFBCD F(ITAPE)
        ENTRY    ZEFBIN F(ITAPE)
*
*
*                     ----ABSTRACT----
*
* TITLE - ZEFBCD WITH SECONDARY ENTRY ZEFBIN
*        TEST IF NEXT TAPE RECORD IS END OF FILE AND REPOSITION TAPE
*
*              ZEFBCD READS ONE BCD RECORD AND CHECKS TO SEE IF THAT
*              RECORD WAS AN END OF FILE.  IT BACKSPACES OVER THE RECORD
*              BEFORE RETURNING.  A REDUNDANCY INDICATION IS PROVIDED.
*
*              ZEFBIN DOES THE SAME THING FOR A BINARY TAPE.
*
*
* LANGUAGE    - FAP SUBROUTINE
* EQUIPMENT   - 709 OR 7090 (MAIN FRAME AND TAPE DRIVE)
* STORAGE     - 54 REGISTERS
* SPEED       -
* AUTHOR      - J.N. GALBRAITH, JR.
*
*
*                     ----USAGE----
*
* TRANSFER VECTOR CONTAINS ROUTINES - NONE
*       AND FORTRAN SYSTEM ROUTINES - (IOS),(RDS),(RCH),(TCO),(TRC),
*                                     (TEF),(BSR)
*
* FORTRAN USAGE
*     ENDFIL=ZEFBCDF(ITAPE)
*     ENDFIL=ZEFBINF(ITAPE)
*
*
* INPUTS
*
*    ITAPE     LOGICAL TAPE NUMBER TO BE CHECKED.
*
*
* OUTPUTS
*
*    ENDFIL    FLOATING POINT INDICATOR.
*              = 0. IF END OF FILE
*              = 1. IF NO END OF FILE
*              =-1. IF REDUNDANCY FOUND (READ TEN TIMES).
*                   A REDUNDANCY WILL NOT BE SIGNALLED IF BOTH THE
*                   REDUNDANCY INDICATOR AND END OF FILE INDICATOR ARE
*                   TURNED ON, BUT THE END OF FILE WILL BE SIGNALLED.
*
*
* EXAMPLE
*
* 1. USAGE    -        ITP = 9
*                      REWIND ITP
*                      A = 6HCARD 1
*                      WRITE OUTPUT TAPE ITP, 10, A
*                   10 FORMAT(A6)
*                      END FILE ITP
*                      REWIND ITP
*                      ENDFL1 = ZEFBCDF(ITP)
*                      ENDFL2 = ZEFBINF(ITP)
*                      READ INPUT TAPE ITP, 10, A
*                      ENDFL3 = ZEFBCDF(ITP)
*                      ENDFL4 = ZEFBINF(ITP)
*                      REWIND ITP
*                      WRITE TAPE ITP, A
*                      END FILE ITP
*                      REWIND ITP
*                      ENDFL5 = ZEFBINF(ITP)
*                      ENDFL6 = ZEFBCDF(ITP)
```

****
*   ZEFBCD   *
****
(PAGE 2)

```
* OUTPUTS
*    OUTPUTS - ENDFL1...6 = 1., -1., 0., 0., 1., -1.
*
*
* PROGRAM FOLLOWS BELOW
*
       PZE
       BCI     1,ZEFBCD
ZEFBIN SSP
       ADD     BINARY         SET FOR BINARY MODE
ZEFBCD SSP
       STO     TAPE
       SXA     RETURN,1
       SXA     RETURN+1,2
       SXD     ZEFBIN-2,4
       CAL     TAPE
       TSX     $(IOS),4
       AXT     1,1
 READ  XEC*    $(RDS)
       LDQ*    $(RCH)
       SLQ     *+1
       RCHA    IO
       LDQ*    $(TCO)
       SLQ     TCO
       LDQ*    $(TEF)
       SLQ     TEF
       LDQ*    $(TRC)
       SLQ     TRC
       SLQ     SETOFF
TCO    TCOA    *
TEF    TEFA    ENDFIL
TRC    TRCA    REDUND
       XEC*    $(BSR)
       CLA     ONE
RETURN AXT     **,1
       AXT     **,2
       LXD     ZEFBIN-2,4
SETOFF TRCA    *+1
       TRA     1,4
ENDFIL XEC*    $(BSR)
       CLA     ZERO
       TRA     RETURN
REDUND XEC*    $(BSR)
       TXI     *+1,1,1
       TXL     READ,1,10
       CLS     ONE
       TRA     RETURN
ZERO   PZE     0
ONE    DEC     1.
TAPE   PZE
BINARY PZE     16
IO     IOCD    DUMMY,0,1
DUMMY  PZE
       END
```

```
************************
*    ZEFBIN            *
************************
 REFER TO
     ZEFBCD
```



```
************************
*    ZEFBIN            *
************************
 REFER TO
     ZEFBCD
```