211
8/21/72
LA-4785

14.2241

RAYMATH, a Computer Language
for the Reduction of Data from
Neutron Cross-Section Experiments

los alamos
scientific laboratory

of the University of California

LOS ALAMOS, NEW MEXICO 87544

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

REPRODUCTION OF THIS DOCUMENT IS UNLIMITED

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Printed in the United States of America. Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.95

LA-4785

UC-37

ISSUED: July 1972

RAYMATH, a Computer Language for the Reduction of Data from Neutron Cross-Section Experiments

by

Marvin V. Harlow, Jr.

Kenneth D. Ferrell*

Anne N. Phillips

Rita M. Wigglesworth

*Permanent address: EGG, Albuquerque, New Mexico

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

UNCLASSIFIED
24

CONTENTS

ABSTRACT

I.	INTRODUCTION	1
II.	ARGUMENT VOCABULARY, SYNTAX, AND PROGRAM STRUCTURE	3
III.	TRANSLATOR AND RAYMATH INSTRUCTION REFERTOIRE	6
IV.	STRUCTURE OF SUBROUTINE PACKAGES AND EXECUTION OF PROGRAM FORPROG	10
V.	EXAMPLES OF RAYMATH PROGRAMS	14

ACKNOWLEDGMENTS

APPENDIX A:	RAYMATH SYNTAX	17
APPENDIX B:	CONTROL DECK CHARTS	19
APPENDIX C:	CHARTS FOR MODULAR INSTRUCTIONS	30
APPENDIX D:	SOURCE PROGRAM LISTING	56
REFERENCES AND NOTES		147

RAYMATH, A COMPUTER LANGUAGE FOR THE REDUCTION OF DATA FROM NEUTRON CROSS-SECTION EXPERIMENTS

by

Marvin V. Harlow, Jr.

Kenneth D. Ferrell

Anne N. Phillips

Rita M. Wigglesworth

ABSTRACT

RAYMATH is a problem-oriented computer language for reduction of data residing in one-dimensional arrays. It consists of 52 modular instructions designed to manipulate 6 arrays and 32 constants. Array mathematics, special-purpose operations, and extensive plotting may be achieved readily. Free-input format permits problems to be coded and submitted through a remote-keyboard terminal. The subroutine packages for the modular instructions, as well as the language translator, are written in FORTRAN IV for a CDC-6600 computer. FORTRAN IV is a proper subset of RAYMATH.

I. INTRODUCTION

RAYMATH is a problem-oriented computer language for reduction of data from neutron-capture experiments^{1,2} involving bomb neutron sources.³ It is sufficiently versatile to have applications in other areas. The name RAYMATH is a composite of the last three characters of the word array and the first four characters of the word mathematics, i.e., ARRAY MATHEMATICS. Development of the language began in the fall of 1967. A user's manual⁴ was published early in 1969. FORTRAN coding of the current version was started in the fall of 1969 and finished in spring of 1971. The present report is a comprehensive description of this version.

RAYMATH is principally an extension of FORTRAN syntax. To a much lesser extent, it may also be considered a semantic extension or an extension of the meaning of FORTRAN. We wrote this extension to obtain a new user-oriented language with the following characteristics:

- a. A well-defined treatment of boundary conditions, special data values, and potential user errors
- b. Solution of ad hoc data-reduction problems in a

manner almost independent of the length and character of input parameters and data sets

- c. Freedom from FORTRAN recoding
- d. Isolation of fundamental data manipulations and operations into modular units to permit modular programming
- e. Data input and processing through a remote-keyboard terminal.

That scientists and engineers benefit substantially from well-designed, specialized languages is a well-documented fact.⁵⁻²⁰ Instructions that appear simple but are powerful in operation permit straightforward data reduction and analysis. Language processors can handle intricate logic, messy details, and special cases. Instructions reflect the terminology of the applications. Input procedures are flexible; they may use free format, keywords, and even conversational mode. Thus, a specialized language provides a highly user-oriented interface with the computer. The scientist or engineer is much more inclined to use the computer as a tool for exploration and sophisticated data analysis if such an interface is available.^{19,20}

RAYMATH uses the translator or preprocessor technique that Bobrow and Weizenbaum discuss in their paper on list processing and language extension.²¹ RAYMATH's main program, TELERAY, and subroutines, FREEB, A321, A212, and A216, form the translator program. This program converts RAYMATH instructions to a sequence of FORTRAN subroutine calls and allows embedded FORTRAN statements to pass unaltered. The translator creates a FORTRAN equivalent to the RAYMATH program. When executed, this FORTRAN program produces the solution to the problem specified in RAYMATH.

Modular program design is a feature shared by many current problem-oriented languages, including RAYMATH. The high degree of task separation in modular instructions results in less efficient programs. However, introducing new operations by rewriting and checking individual modules is easier and more efficient than changing and debugging a large, intricate program. Selection of an effective set of modular instructions is a difficult task, best performed by those familiar with their application area and experienced in computer programming.²² Once found, however, a good instruction set permits programming of many unique but similar problems with minimal effort.²²

Modular systems commonly use a central data area. Restricted access regulations may protect parts of this area. In RAYMATH, the data area consists of 6 large arrays residing in central memory. The user may modify the array's contents as long as he does not transgress its functions. A more sophisticated approach to data handling occurs in the JOSHUA system developed at the Savannah River Laboratory.¹³ JOSHUA's data base consists of standard, user, and job data sets. Standard data sets are available to all users on a read-only basis. A user's data sets are available only to that user's programs. Job data sets in JOSHUA consist of output resulting from program execution; they are usually destroyed after job termination but can be saved at the user's request.

As C. L. Miller has pointed out, a modular system should be able to handle problems of indefinite size.²⁰ Ability to shuffle modules and data in and out of central memory enables the system to manage long problems involving large data sets. The present version of RAYMATH has a similar but lesser

capability. Only those modules and arrays required by the program reside in central memory.

Neutron-capture experiments done by time of flight with bomb neutron sources are complex and expensive and require many months of preparation. Each experiment generates large amounts of data recorded on film. One to two man-years are required to reduce it. Because so much expense and effort goes into each experiment, making as many special calculations and corrections to the data as are required to extract meaningful results is worthwhile. If the computer code system for data reduction and analysis is to accommodate these special calculations, the system must be more versatile than one written in classical²⁰ FORTRAN. A combination of classical codes with the RAYMATH code provides this versatility.

Figure 1 shows the computer code system. Phase-1 and Phase-3 codes, DATA²³ and SIGM7,²⁴ respectively, are the two classical codes. Each is relatively small and fast. Routine use of these codes necessitates constraints on the problems they solve. In Phase 2, RAYMATH compensates for these constraints.

Fig. 1. The computer code system.

Inherent to the DATA-RAYMATH-SIGM7 system is the permanency of the FORTRAN coding; i.e., once the FORTRAN codes are thoroughly checked out, they seldom require alteration to accommodate new problems that arise with each new experiment or data set.

In Phase 1, DATA processes as many as 100 film traces per problem; each trace contains a few hundred point pairs. DATA generates a card deck containing many thousands of point pairs. This deck is edited by hand, mistakes are corrected, and film data are reprocessed. The final data sets become input to Phase-2 calculations. Phase-2 operations are very simple in principle but difficult in practice, because each problem deals with many data sets. Step D of Phase 2 is the hand extraction and tabulation of neutron-capture resonance areas. Reduction of non-resonant neutron-capture data requires neither this step nor Phase 3.

The code in Phase 2 consists of a RAYMATH language translator and a set of subroutine packages for the 48 modular instructions. A subroutine package is defined as the set of related subroutines for a modular instruction. A principal routine is defined as the first routine called in a subroutine package. Most RAYMATH subroutine packages, as well as the language translator, are written in FORTRAN IV for a CDC-6600 computer (65000₁₀-size central memory). Some of the CALCOMP plotting subroutines are written in COMPASS, the CDC-6600 assembly language. The total length of RAYMATH's source code is about 8200 statements; it includes a total of 116 routines. Execution of most problems coded in RAYMATH requires < 1 min of central processor time, ~ 4 min of peripheral processor time, and a memory allocation of between 60000 and 156000 octal words. Short problems may run in fewer than 60000₈ words.

RAYMATH syntax and vocabulary are explicit and easily remembered. Each coding line specifies one statement that may be written in either free or fixed format. A line consists of five fields, each having a definite content and maximum length. Fields 1 and 3 hold arguments only. Field 2 contains a RAYMATH instruction. All numeric constants, read as decimal numbers, appear in field 4. Only array data can enter from cards. Field 5 holds Hollerith constants such as comments, data-block titles, or graph titles. The vocabulary of arguments consists

of the 6 array names (YX-CY), 32 constants (C1-TITLEC), and 2 special symbols (Blank and **) listed in Fig. 2. To be recognized in a RAYMATH program, arguments and instructions must be correctly spelled.

Table I shows the instruction repertoire. Some of the names resemble FORTRAN instruction names, but the RAYMATH instructions are much more sophisticated than their FORTRAN counterparts. Embedding FORTRAN statements in a RAYMATH program is legal. This allows RAYMATH to process special problems that require access to individual array elements. Normally, RAYMATH treats each array as a unit, processing all elements in the same way.

The six data arrays are organized in pairs called data sets. All array elements are real numbers, and the independent variables are > 0. The interval between neighboring independent variables is small enough to allow linear interpolation. A RAYMATH program may handle up to four 6000-point data sets at once, provided the independent variables of two of these sets are inferred by interpolation.

The structure of a RAYMATH program is very simple. Each program has a main program that may be followed by subprograms. Subprograms begin with a HEREIS instruction and end with a RETURN instruction. GOTO instructions call subprograms. Successive levels of subprogram calls are legal.

II. ARGUMENT VOCABULARY, SYNTAX, AND PROGRAM STRUCTURE

The set of problems that RAYMATH programs were intended to solve dictated the argument vocabulary, syntax, and structure of the language. Figure 2 lists the argument (operand) set. The program assigns unique type and identification tags to each argument. Columns 4, 5, and 6 in the figure specify the types of operations in which arguments may appear as operands. The operations TO and EXCHANGE require arguments of the same type. Subroutines A215 and A217 initialize arguments at the start of RAYMATH program execution. The argument list cannot be easily expanded; to do so would involve reprogramming the translator and many subroutine packages. Arguments must be correctly spelled.

RAYMATH's six data arrays are organized in pairs so that each pair may represent a data set. This report calls the array pairs YX and YY the "Y-arrays," and the pairs AX and AY the "A-arrays," etc.

ARGUMENT	TYPE	ID NUMBER	MATHEMATICAL OPERATIONS ⁺	TO, EXCHANGE OPERATIONS	READ, PRINT OPERATIONS	INITIAL VALUE SET BY SUB- ROUTINE A215	REMARKS
YX YY AX AY BY CY	ARRAY ↓	1 2 3 4 5 6	YES	YES	YES		All 6000 elements are set to zero by A217 ↓
C1 C2 C3 C4	CONSTANT ↓	7 8 9 10	↓	↓	↓	0. 0. 0. 0.	
YI YF AI AF BI BF CI CF NI NF	DOLOOPMT ↓	15 16 17 18 19 20 21 22 23 24	NO	↓	↓	1 1 1 1 1 1 1 1 1 6000	
XLIMUP XLIMLR YLIMUP YLIMLR EMIN EMAX	PARAMETER ↓	25 26 27 28 29 30	↓	NO	↓	0. 0. 0. 0. 0. 0.	
NSTEP KEY100 KEY101 LINE100 LINE101 DUMPLT PRINO	NUMBER ↓	31 32 33 34 35 37 38	↓	↓	↓	1 0 0 0 0 1 0	
TITLE COMMENT TITLEA TITLEB TITLEC	AFORMAT ↓	39 40 41 42 43	↓	↓	↓	BLANK ↓	
**	BLANK STAR	45 46	↓	↓	NO		

⁺Mathematical operations are the RAYMATH instructions +, -, *, /, **, INVERT, SQRT, LOGE, and EXP.

Fig. 2. The argument vocabulary.

Indices corresponding to each pair delimit data-set boundaries as shown below.

Data-Set Name	Independent Variable	Dependent Variable	Boundary Pointers	
			Upper	Lower
Y-arrays	YX	YY	YI	YF
A-arrays	AX	AY	AI	AF
B-arrays	(BX)	BY	BI	BF
C-arrays	(CX)	CY	CI	CF

Arrays CX and BX do not actually exist, but by interpolation may be considered identical to AX or YX. Thus, four data sets can be represented at once.

Each array receives 6000 consecutive words of central memory. Array indices NI and NF denote the length of the data sets. All data sets have the same length to facilitate programming in RAYMATH. The user seldom needs to alter the initial values of NI=1 and NF=6000; if he does, he should restore

TABLE I. INSTRUCTION REPERTOIRE

INPUT/OUTPUT	GRAPHICS	MATHEMATICAL	SPECIAL
READ	SPLIT	+	AVOVERTIME
PRINT	PLOT	-	RUNSUM
PRALLARRAY	SETGRIDY	*	RESINTEGRAL
READTAPE1	SETGRIDXY	/	
READTAPE2	MPLLOT2	INVERT	
WRITETAPE1	CALCOM2	**	
WRITETAPE2	MPLLOT	SQRT	
PUNCH	CALCOMP	LOGE	
DUMP	PRINTPLOT	EXP	
INTEGRATE			
ARRAY MANIPULATION	UTILITY	TRANSLATION CONTROL	
EMINMAX	LASTTITLE1	END	
POINTER	LASTTITLE2	GOTO	
YROLLEDTOA	LOADYLIMUP	HEREIS	
AROLLEDTOT	LOADYLIMLR	RETURN	
MELDYANDA	NOEQUALX		
EXCHANGE	DELETEx=		
TO	CLOCK		
LEGALORDER	INTERPLATE		
	(BLANK)		

them as soon as possible. The EXCHANGE instruction illustrates the use of these indices.

The translator allocates space in central memory to arrays other than the Y-arrays required by the RAYMATH program. The Y-arrays are called "working arrays" because array input/output usually passes through them, and because YY receives the results of most arithmetic operations involving arrays. Thus, to load the A-arrays from data on cards requires the instruction sequence

```
,READ
, YROLLEDTOA
```

Normally, the Y-arrays are implied arguments of instructions that do not have operands. An example is INTEGRATE which integrates the Y-arrays between YI and YF.

Note that in Fig. 2 all arguments from YX through NF are two characters long and may appear as left- or right-hand operands in statements. The same holds for the null argument, blank. Arguments XLIMUP through ** may only be right-hand operands. Arguments from XLIMUP through TITLEC specify data-set boundary conditions, output formats, and graphics-routine parameters. The argument ** denotes

a "new" tape unit for a WRITETAPE1(2) instruction. One should distinguish this argument from the raise-to-a-power instruction ** that may appear in field 2 of a RAYMATH statement card.

Constants C1, C2, C3, and C4 appear as arguments in mathematical instructions or as parameters specifying some of the desired features of a graphics instruction plot. Experience has shown that four floating-point constants are sufficient for solution of the Phase-2 problems described in Sec. I. The instructions

```
,READ,C1
,PRINT,C1
```

where C1 is an example of any of the four constants, effect input/output for constants. C2 is called the "working" constant because it holds results of mathematical operations involving constants and the results of the RESINTEGRAL and INTEGRATE instructions.

Rules governing RAYMATH program statement structure, or RAYMATH syntax, are simple and few. The RAYMATH language allows only one statement per coding line or record. Each coding line is essentially independent of all other coding lines; that is, RAYMATH is contextually independent. A coding line or statement may have at most five fields. Columns and commas delineate fields in fixed- and free-format statements, respectively. See Appendix A for specific details and examples. The user may not mix fixed- and free-format statements in the same program. The following chart defines the maximum length and content of each field.

Field	1	2	3	4	5
Maximum Character Length	2	10	10	15	40
Required Contents	OPERAND	OPERATOR	OPERAND	DECIMAL NUMBER	HOLLERITH CONSTANTS

Blank fields and blank coding lines or statements are allowed. Field 4 contains integer and floating-point constants written as decimal numbers. Fixed-format data cards or data-library tapes (specially prepared by another RAYMATH program) are the only sources of array data. The Hollerith constants in field 5 may be either statement comments, data-block titles, graph titles, graph comments, embedded FORTRAN statements, or RAYMATH subroutine names. Experience has shown that the 40-character length of this field is sufficient for its intended contents.

A subtle property of free-format statements used for programs that are run in local batch mode is that no more than 72 columns are ever needed for any statement. Thus, existing computer system routines designed for the creation and maintenance of 72-column textual or FORTRAN-source program cards, such as UPDATE,²⁵ can create and modify libraries of RAYMATH programs.

Because RAYMATH is user-oriented, it must be simple and require little repetition. Dividing the program into a main program and subprograms and not having loops, logical branches, or statement numbers provides structural simplicity. Placing groups of statements used more than once in subprograms and calling these subprograms as often as required achieves minimal repetition of program steps.

The main program consists of a sequence of statements and subprogram calls which are executed serially. It begins with a statement-format declaration and ends with an END instruction. Subprograms must follow the END instruction.

Subprograms begin with a HEREIS instruction and end with a RETURN. The name of the routine may be a number, a character string, or both; it appears in field 4 and/or 5 of the HEREIS. The name must be different from all other subprogram names in the program. Subroutines are called by name using a GOTO instruction that contains the routine name in fields 4 and 5. Up to 10 successive levels of sub-routine calls are legal. During program assembly, the translator inserts the subprogram serially into the assembled program, omitting the GOTO, HEREIS, and RETURN. Therefore, one usually puts a comment in the first statement after the HEREIS to identify the subroutine.

Module Chart 41 of Appendix C discusses and illustrates the GOTO, HEREIS, END, and RETURN instructions. Also included in the chart is a table of error diagnostics and default actions that may occur when these statements are used incorrectly. If the structure of the main program or any subprogram is illegal, the job terminates with diagnostic messages.

III. TRANSLATOR AND RAYMATH INSTRUCTION REPERTOIRE

An interpretive processor translated and executed the 1969 version of RAYMATH.⁴ It first assembled the RAYMATH program by inserting subprograms

into the main program stream in place of subroutine calls. Then, it deciphered and immediately executed the instructions as ordered in the assembled RAYMATH program. The processor was simple, easily implemented, and efficient; however, it did not produce a complete translation of the original RAYMATH program. A serious disadvantage was the need to load all subroutine packages and arrays into central memory regardless of source-program requirements. A minor disadvantage was the inability to insert FORTRAN statements into source programs.

During the summer of 1970, a translator without these disadvantages replaced the interpretive processor. This translator needs only two additional minutes of peripheral processor time for execution. It converts a RAYMATH source program to an equivalent sequence of FORTRAN statements which are then compiled and executed in the customary manner. The translator is a two-pass system, but it appears as a single-pass system to the RAYMATH coder. The fact that only those subroutine packages and arrays needed for the problem receive central memory permits use of an open-ended list of modular instructions. Execution of short RAYMATH programs previously requiring up to 156000₈ memory words now requires, at most, 60000₈. The advantage of FORTRAN-RAYMATH language compatibility was also realized.

Program translation entails searching a table of subroutine names to find the principal subroutine corresponding to the RAYMATH instruction, allocating required array storage, determining the type and identification number of each argument, writing a program-information file (TAPE18), and constructing the appropriate FORTRAN routines. RAYMATH subprogram calls are translated in the same manner as in the earlier interpretive version. The translator normally constructs four FORTRAN routines: a main program called FORPROG, and three subroutines, A200, A201, and A217. These subroutines load, unload, and initialize arrays at FORPROG execution time. FORTRAN statements and comments embedded in a RAYMATH program pass through the translator unmodified.

Embedding FORTRAN statements alters the structure of FORPROG. This procedure allows a RAYMATH program to process special problems that require access to individual array elements. Embedding FORTRAN statements is hazardous because these statements are not processed in a user-oriented manner. Therefore,

they can cause errors that are extremely difficult to trace and correct. The syntax for these FORTRAN statements must conform to the rules set forth in Appendix A.

Figure 3 presents a conceptual view of the translator, showing the translator structure and program flow. Box numbers indicate the order in which the translator executes the functions described in each box. The right-hand segment of each box contains the name of the FORTRAN subprogram that executes the translator function. TELERAY, a modified version of the 1969 interpretive processor, is the main translator program.

Box 1 gives the initialization procedures for translator input. If the first statement in the main RAYMATH program is the declaration OPEN or OPENTELE, program TELERAY expects statement cards written in free format. Otherwise, it assumes fixed format. OPENTELE differs from OPEN because it calls for an output file for a remote-keyboard terminal. The FORTRAN variable NTELE is a key that contains 1 when the user requests a remote-terminal-output file; it is 0 otherwise.

The operation of the second box is to read RAYMATH source cards, to count them, and to check cards having either a WRITETAPE1(2) or END instruction. Subroutine FREEB of box 3 decodes free-format RAYMATH source program cards. See Appendix A for details. The card number of the END instruction defines the boundary between the main program and the subprograms. WRITETAPE1(2) sets switches that cause the table of contents of the library files to be listed when program execution is complete. In the fourth box of the figure, the addition of the STOPHERE instruction provides a program terminator that can be encountered only if there is an error in the program. Boxes 5 and 6 of the figure are self-explanatory.

Subroutine A212 of the translator determines the identification number (ID) and type (TYPE) of an argument by searching a RAYMATH vocabulary list. See box 7. Arguments not in the list, or illegal arguments, receive an ID of 1000 and a TYPE of ERROR, which causes an error diagnostic to be printed in the execution listing. As indicated in box 8 of the figure, the translator subroutine A321 searches for the principal subroutine of an instruction.

Fig. 3. Conceptual view of the RAYMATH translator.

Subroutine A321 contains the list of modular instruction names given in Table I and the array requirements for each. The dump package, A308, replaces instructions not in the list; for such instructions, an error message appears in the listing of the assembled program. The actions of boxes 9 and 10 are self-evident.

The translator calculates the total amount of array storage required by the RAYMATH program and passes this number to subroutine A216. A216 incorporates it in the constructed FORTRAN subroutines A200, A201, and A217. See box 11. Box 12 shows the conclusion of the translation procedure in which files are rewound and positioned for FORPROG execution.

Table II shows the structure of the input/output files dedicated to the translator and to FORPROG. Each file contains only one type of information. This property significantly simplifies the translator design and facilitates remote-terminal file manipulation and modification. An example of a dedicated input file is TAPE7 which contains records of RAYMATH source program cards. TAPE11 is a typical output file. It contains line printer records generated by the translator and FORPROG. Monitor control statements copy this file to the line-printer file, OUTPUT, following program execution.

Table I shows the RAYMATH instruction set. Only three of the 52 instructions are special, i.e., usable only in neutron cross-section data reduction problems. The other 49 are applicable to more general problems. Modular design has stereotyped each instruction to the extent that it performs only one type of function. This function serves as a basis for the instruction classification. Each Module Chart in Appendix C describes an instruction by stating its purpose, illustrating its syntax, and giving its sequential program action. The charts list quantities that must be preset, arguments whose values may change, boundary conditions, error diagnostics, and default actions.

Routines classified as graphics plot the Y-arrays, and for MPLOT2 and CALCOM2, plot the A-arrays simultaneously. Plots may be made on either microfilm or paper. When the arrays involved contain only a few hundred points, the user may employ SPLOT, PLOT, SETGRIDY, or SETGRIDX to obtain a graph plotted on microfilm. For large arrays, MPLOT and MPLOT2 can plot successive frames of microfilm which the user can print and join into a continuous scroll graph. The CALCOMP plotter produces a similar scroll graph from CALCOMP or CALCOM2. The PRINTPLOT instruction can plot on either the line printer or the keyboard terminal.

TABLE II. INPUT/OUTPUT FILES

File Name	Dedication				Contents
	Program		Mode		
	Translator	FORPROG	Input	Output	
TAPE1		X	X		Special-array libraries
TAPE2		X	X		Special-array libraries
TAPE3		X	X		File-name dictionary of TAPE1
TAPE4		X	X		File-name dictionary of TAPE2
TAPE7	X		X		RAYMATH program
		X		X	Off-line CALCOMP plotter input
TAPE10		X	X		All array-data card decks
TAPE11	X	X		X	Line printer
TAPE12	X	X		X	Remote-keyboard terminal or Punch card output
TAPE18	X			X	Assembled RAYMATH program and the identification number and type of each argument
		X	X		As above
TAPE19	X			X	FORTTRAN source statements of FORPROG, A200, A201, and A217
OUTPUT	X	X		X	FORTTRAN error messages
FILM		X		X	Off-line SC4020 microfilm plotter input

All of the input/output instructions except DUMP use the Y-arrays as implied arguments whenever no argument, other than ** or blank, is specified. DUMP prints the nonarray common block. READ and PRINT read and print Hollerith and numeric constants located in their right-hand arguments. PRALLARRAY prints all six arrays. WRITETAPE1(2) or READTAPE1(2) write or read the Y-arrays to or from a disk file or a magnetic tape. These instructions do bookkeeping that permits sequential files to be written and retrieved from the same unit. PUNCH punches the Y-arrays on cards.

In the special-instruction classification, AVOVERTIME averages neutron time-of-flight data stored in the Y-arrays over a differential time interval. RUNSUM uses the trapezoidal rule to determine the cumulative integral or running sum of YY versus YX. The point-by-point listing of this integral helps determine integrals over selected ranges of YX. RESINTEGR1 calculates a partial resonance integral of the neutron-capture cross section over neutron energy (E), weighted by a $1/E$ neutron flux.

Utility instructions are those which do not belong to any other classification. LASTTITLE1(2) identifies the last "good" data-block title on the library units TAPE1(2). LOADYLIMUP and LOADYLIMLR determine and load the upper and lower graphics-scaling boundaries into variables YLIMUP and YLIMLR, respectively. NOEQUALX makes the Y-arrays single valued with respect to YX. DELETTEX= deletes from the Y-arrays a point that appears in field 4 of the instruction. CLOCK determines elapsed central-processor time during program execution and prints it. The INTERPLATE instruction linearly interpolates the Y-arrays at AX values.

The mathematics classification in Table I lists the available operations; they perform either arithmetic or mathematical functions. The six arrays and four constants are the only legal arguments for mathematical instructions. The instructions contain protection against illegal arguments and illegal argument values. The Module Charts list the default actions and diagnostics. Mathematical operations treat each array as a unit, and operations between or upon arrays perform floating-point arithmetic. Arithmetic on or between the four constants proceeds like array arithmetic. The need for integer mathematics was insufficient to justify its inclusion

in RAYMATH.

Except for INVERT, mathematical instructions having constant arguments store their result in C2, and those having array arguments store their results in array YY between indices YI and YF. INVERT stores results in its argument. The domain of mathematical operations, irrespective of the array arguments of the operation, goes from index YI through index YF. Because interpolations usually precede a mathematical operation, causing all arrays to have the same domain, no difficulties result from this fixed-domain property. By adding several more instructions to the RAYMATH program, the user can alter the standard range of operation.

The array-manipulation instructions of Table I alter array boundaries, contents, and locations. Instructions EMINMAX and POINTER have no arguments; they operate on the Y-arrays to change the boundary pointers YI and YF. Both POINTER and EMINMAX find the indices of the Y-arrays corresponding to variables EMIN and EMAX, thereby selecting a discrete group of Y-array data. POINTER differs from EMINMAX in that the total number of points between Y-array boundaries is constrained to be less than, or equal to, the constant KEY100. The Y-arrays store the resulting data set repositioned so that YI is 1 and YF equals the number of points.

MELDYANDA merges the contents of the Y-arrays between YI and YF with the contents of the A-arrays between AI and AF. The resultant arrays reside in the Y-arrays and may be out of order. LEGALORDER reorders YX by checking the Y-array data against a set of specifications, identifying and discarding any that do not agree, and ordering, rechecking, and counting the rest. The resulting Y-arrays are in descending order, and YX is greater than zero. Elements of the Y-arrays are real numbers.

EXCHANGE, TO, AROLLEDTOY, and YROLLEDTOA manipulate arrays. These instructions move AX, AY, BY, and CY from or to the Y-arrays for input/output. They operate on the range 1 to 6000, causing the contents of an entire array to be transferred from one set of storage locations to another. In the TO instruction, the memory locations of the right-hand operand receive the contents of the left-hand operand. EXCHANGE interchanges the storage locations of the left- and right-hand operands. Neither TO nor EXCHANGE alters the value of array-boundary pointers

YI, YF, etc. AROLLETOY transfers the A-arrays and boundary indices AI and AF to the Y-arrays and indices YI and YF, respectively. The inverse of AROLLETOY is YROLLETOA.

IV. STRUCTURE OF SUBROUTINE PACKAGES AND EXECUTION OF PROGRAM FORPROG

During the translation phase, the processor inserts in FORPROG calls to the principal subroutines of the subroutine packages required by the RAYMATH instructions. Some instructions have as many as five subroutines in their package, whereas others have only one. Packages and subroutines within a package almost always communicate through variables stored in blank common.

The internal structure of subroutines in a package is highly coherent; that is, the logical environment of each FORTRAN statement consists of numerous other statements scattered throughout the program. As Cocke and Swartz²² pointed out, the difficulty of completing, correcting, and testing such a program may be proportional to the square, or higher, power of its length. Checking the performance of each instruction with test programs and data sets modeling illegal situations and special cases that the instruction must treat required a lot of time and effort. One result was the development of a set of subroutine packages that require little maintenance or correction. Because the need to modify is rare, the high coherence is not unduly objectionable.

A fundamental property of RAYMATH is the user-oriented solution of ad hoc problems that have different RAYMATH codes. Subroutine-package design incorporates several features giving RAYMATH this ability. These features are specialized allocation and use of blank common, algorithms that check input data for the package and monitor its calculations, and error exits that skip the instruction, if necessary, without stopping overall program execution. RAYMATH programs execute to the end, even if some instructions detect errors and print diagnostic messages. Therefore, each program execution output must be hand-checked to ensure that it is error-free. Interspersing PRINT instructions in the code often enough to display program execution clearly aids hand checking.

Automatic checking of all parameter inputs to instructions that have many preset parameters, such as MPLOT, requires excessive additional FORTRAN coding. To avoid this excess, these instructions perform only a reasonable amount of automatic checking, leaving the rest to the user. Each Module Chart states or implies which input parameters undergo automatic checking and which do not.

The charts also list parameter boundaries. If the user feels unsure about the effect of preceding instructions on a parameter, he should load the parameter immediately ahead of the instruction in question. If an input parameter subject to automatic checking is outside its specified range, a diagnostic message usually results. In a few cases, e.g., EMINMAX, the program alters incorrect input parameters to try to change them to values within their specified ranges. The execution output reports the new values assigned in such cases. If the type and/or identification number of an argument are incorrect, the instruction is skipped and a diagnostic is printed; for an example, see EXCHANGE.

Most of the instructions operating on arrays assume that the arrays contain only legal points arranged in descending order. To guarantee this condition, such instructions call subroutine CHECKER at the outset of their operation. This subroutine prints a summary of what it did to the arrays and array-boundary pointers. In fact, most subroutine packages print execution information to tell the user how they processed the instruction.

As mentioned above, subroutine packages and subroutines within a package pass variables through blank common. Transfer through blank common is faster than argument transfer and it simplifies subroutine calls. A few subroutines in RAYMATH's source code, for example, subroutine PRTPLOT, do transfer variables by subroutine arguments.

That the arrangement of RAYMATH instructions changes from code to code leads to restrictions on the allocation and use of common-storage locations to prevent one set of instructions from overwriting variables intended for another set. These restrictions are peculiar to RAYMATH and are not usually encountered in classical FORTRAN codes. Organization of the common block into a nonarray block and an array block aids in understanding these restraints.

Detailed knowledge of common-block storage has facilitated the addition of new instruction modules to RAYMATH and the embedding of FORTRAN statements in RAYMATH programs. A discussion of Table III, as well as a description of how the restricted-use concept is applied to the nonarray common block, may help provide this knowledge.

Column one of Table III names all of the common-block variables. The nonarray common consists of the locations from NTELE through LOCK. Some of these variables are actually small arrays (2 to 8 words long). However, for this discussion we may consider them constants. Arrays YX through CY form the array common. The length and allocation of the nonarray common is fixed. The length of array common is a function of the RAYMATH problem being solved. Only that part of array common required by the problem and consistent with the array-allocation hierarchy described below receives space in central memory.

In the hierarchy of arrays, YX is highest and CY is lowest. All arrays from YX through the lowest array needed in the problem receive space in common. As an example, consider a problem that needs arrays YX, YY, and BY. Arrays YX, YY, AX, AY, and BY will receive space in central memory, occupying a total of 30000 memory words. A subroutine usually contains the nonarray common block plus whatever array storage it needs consistent with the hierarchical rule above. The 6000 dimension of each array is fixed, and can be changed only by recompiling all subroutine packages containing array common. However, any subroutine may alter the value of array elements, provided the change is implied by the instruction or explicitly reported in the execution output. The instructions READ and DELETEx are examples of the first and second class, respectively.

Variables NY to NC, FAULTLT, and DUMMY are not vocabulary words, but are accessible through a READ instruction. Their purpose is local transfer of internal variables among specified subroutines or groups of subroutines. Column 4 of the table gives the FORTRAN type of each common variable. WORD has a mixed type; WORD(4) is real, and the other elements of the array are Hollerith type. The Hollerith type is implicit in the subroutines because no alphanumeric declaration exists in CDC-6600 FORTRAN. The Remarks and Application section

of the table is self-explanatory.

The last two columns of the table serve as a guide to show which nonarray common variables may be redefined and which may be overwritten or altered in embedded FORTRAN or in new subroutines. We exclude variables NY, NA, NB, NC, FAULTLT, and DUMMY from further discussion in order to comment generally on the rest of the nonarray variables. Internal type, allocation of memory space, and length of the other nonarray variables are fixed. For those variables whose usage or application may change, "yes" appears in the next-to-last column; if the value of the variable also may be changed, "yes" appears in the last column. Many variables in the table have fixed applications, but may be revalued or overwritten; examples are C1, C2, YI, and YF. As is noted in the table, for these variables, change in value should be reported in the execution output. PRINTNO, WORD, ELEMENT, and LIMIT belong to the same class of variables as these, but do not require that change of value be noted in the execution output.

Read-only variables are NTELE, LENGTHY, LENGTHA, TYPE, and ID. Subroutine A217 initializes the first three at the beginning of program execution, and subroutine A319 initializes the last two for each statement-card input. No other routine should alter these variables.

Variables that have a partial read-only character are called restricted, nonarray common. In this class are IREADLK, NI, NF, NSTEP, DUMPLT, and II through LOCK. Either a READ instruction or particular subroutines may alter each restricted variable. For example, READ may alter NI and NF; subroutines A326, A346, and A327 alter DISKA and DISKB. The user should restore each of NI, NF, NSTEP, and DUMPLT to its original value immediately following the application in which it is altered. To do otherwise is to complicate the understanding of the RAYMATH program needlessly.

Instructions in Table I which have no explicit arguments, e.g., EMINMAX, have subroutine packages coded in conventional FORTRAN. DC loops and expressions written in terms of indexed variables handle manipulation and mathematics. The FORTRAN subroutine packages for instructions that operate on arguments use special, nonstandard techniques which are explained below. These instructions are as follows:

TABLE III. COMMON BLOCK ALLOCATION

Variable	Vocabulary Word	Accessible Through Read Instruction	Internal Type	Remarks And Application	Application May Be Redefined In Subroutines	Value May Be Changed In Subroutines
NTELE	No	No	Integer	1 = Remote-keyboard-terminal output. 0 = Local batch output.	No	No
LENGTHY	↓	↓	↓	Maximum length of Y-arrays	↓	↓
LENGTHA	↓	↓	↓	Maximum length of A-arrays	↓	↓
IREADLK	↓	↓	↓	Switch for lockout of READ or READTAPE1(2)	↓	↓
C1	Yes	Yes	Real	Floating-point constant	↓	Yes ^a
C2	↓	↓	↓	Floating-point constant	↓	↓
C3	↓	↓	↓	Floating-point constant	↓	↓
C4	↓	↓	↓	Floating-point constant	↓	↓
NY	No	↓	Integer	Integer constants used within subroutines and for passing temporary constants.	Yes	Yes
NA	↓	↓	↓		↓	↓
NB	↓	↓	↓		↓	↓
NC	↓	↓	↓		↓	↓
YI	Yes	↓	↓	Initial index of Y-arrays	No	Yes ^a
YF	↓	↓	↓	Final index of Y-arrays	↓	↓
AI	↓	↓	↓	Initial index of A-arrays	↓	↓
AF	↓	↓	↓	Final index of A-arrays	↓	↓
BI	↓	↓	↓	Initial index of BY-array	↓	↓
BF	↓	↓	↓	Final index of BY-array	↓	↓
CI	↓	↓	↓	Initial index of CY-array	↓	↓
CF	↓	↓	↓	Final index of CY-array	↓	↓
NI	↓	↓	↓	NI = 1	↓	No
NF	↓	↓	↓	NF = LENGTHA	↓	Yes ^a
XLIMUP	↓	↓	Real	Selected maximum abscissa for plotting	↓	↓
XLIMLR	↓	↓	↓	Selected minimum abscissa for plotting	↓	↓
YLIMUP	↓	↓	↓	Selected maximum ordinate for plotting	↓	↓
YLIMLR	↓	↓	↓	Selected minimum ordinate for plotting	↓	↓
EMIN	↓	↓	↓	Selected minimum energy value	↓	↓
EMAX	↓	↓	↓	Selected maximum energy value	↓	↓
NSTEP	↓	↓	Integer	Array transfer interval	Yes	No
KEY100	↓	↓	↓	Parameter input variable	↓	Yes ^a
KEY101	↓	↓	↓	↓	↓	↓
LINE100	↓	↓	↓	↓	↓	↓
LINE101	↓	↓	↓	↓	↓	↓
FAULTLT	No	↓	↓	Available for future use	↓	↓
DUMPLT	Yes	Yes	Integer	Dump switch: if ≤ 0 ., program execution is terminated after dump; if = 1, execution is continued.	No	No
PRNO	↓	↓	↓	Print switch or printing interval	↓	Yes
TITLE	↓	↓	Hollerith	<u>Hollerith constants</u>	↓	Yes ^a
COMMENT	↓	↓	↓	↓	↓	↓
TITLEA	↓	↓	↓	↓	↓	↓
TITLEB	↓	↓	↓	↓	↓	↓
TITLEC	↓	↓	↓	↓	↓	↓
DUMMY	No	↓	↓	↓	Yes	Yes
WORD	↓	↓	↓	RAYMATH statement	No	No
ELEMENT	↓	No	Real	Temporary storage for real constants	↓	↓
LIMIT	↓	↓	Integer	Temporary storage for integer constants	↓	↓
TYPE	↓	↓	Hollerith	Vocabulary type	↓	↓
ID	↓	↓	Integer	Identification number	↓	↓
II	↓	↓	↓	Subscripts or indices of DO loops	↓	↓
JJ	↓	↓	↓		↓	↓
LL	↓	↓	↓		↓	↓
NTITLE	↓	↓	↓	Number of tape titles in table of contents	↓	↓
NPTSBLK	↓	↓	↓	Number of points in a data set	↓	↓
TAPENO	↓	↓	↓	Tape number	↓	↓

TABLE III. COMMON BLOCK ALLOCATION (cont.)

Variable	Vocabulary Word	Accessible Through Read Instruction	Internal Type	Remarks and Application	Application May Be Redefined In Subroutines	Value May Be Changed In Subroutines
DISKA	No	No	Integer	Data-disk identifier	No	No
DISKB				Table of contents disk identifier		
NFILE			Hollerith	Number of files on the tape		
LASTTTL				Title of last active data block on the tape		
LOCK			Integer	Write switch for table of contents		
YX	Yes	Yes	Real	Single-dimensioned data arrays		Yes
YY						
AX		No				
AY						
BY						
CY						

^aChange in value (except by a READ instruction) should be reported in the execution output.

READ	INVERT
PRINT	**
+	SQRT
-	LOGE
*	EXP
/	TO
	EXCHANGE

The 1969 version of the algorithms for these subroutines, improved by additional diagnostic procedures and messages, appears in the present version. Considering that the present translator writes an intermediate code (FORPROG) in FORTRAN, other ways of coding instructions that have arguments may be more straightforward. However, these 1969 algorithms are well-proven and reasonably efficient.

Algorithms that are variations of a basic algorithm implement the instructions listed above. The algorithm replaces arguments with temporary constants, performs the desired operation, and loads the results into the appropriate constant or array elements. Addition of two arrays illustrates most features of the algorithm. If AX is one array and YX is the other, then the RAYMATH statement is

AX,+,YX

The subroutine for +, A302, first confirms that each argument is one of the allowed types, either array or constant. Because both of the arguments in the example are arrays, control passes to the array section of the subroutine. This section consists of a DO loop over the range of indices YI to YF. Within the loop is a call to subroutine A200 which loads an element of AX and one of YX into temporary

constants ELEMENT(1) and ELEMENT(3), respectively.

The FORTRAN statement

YY(I)=ELEMENT(1)+ELEMENT(3)

adds these, and array YY stores the result as indicated. When the loop is exhausted, A302 terminates. The subroutine also contains provisions for adding two constants or a constant and an array.

Two other important features of the basic algorithm require discussion. The first concerns the rules governing the allowed magnitude of arguments. These rules are coded into several of the above-mentioned instructions. Each argument must pass tests for the particular instruction before execution of the operation. Detection of an illegal argument causes execution to be omitted, a default answer to be loaded, and a diagnostic to be printed. An example that illustrates this feature is the occurrence of a zero-array, argument YY(I) for the reciprocation operation

,INVERT,YY

When the INVERT subroutine encounters this element, it skips the invert operation, loads a -0. into the result YY(I), and prints a remark stating that a zero argument has occurred. The second feature is the technique utilized in instructions such as PRINT and EXCHANGE for transferring results in constants ELEMENT(I) to central memory using an unloader subroutine. The FORTRAN code of the EXCHANGE instruction illustrates this technique.

The subroutine package for a new RAYMATH instruction must be thoroughly checked out using

test-data sets before it is added permanently to the RAYMATH code. It must contain enough diagnostic checks to be user-oriented. The new instruction's name, the call to its principal subroutine, and the statement

NOARRAY=N

must be added to subroutine A321 of the translator. N is the largest of the identification numbers of the arrays used by the package.

The binary object codes of subroutine packages are stored in a library file. A new version of an object code replaces an old version having the same name through the COPYL routine of the computer system. COPYL cannot add object codes with new names to the library; so, to facilitate writing new RAYMATH instructions, the library contains eight spare subroutines (A344, A870, A872, A874, A876, A888, A900, and A902).

Another library stores the FORTRAN source codes of the subroutine packages. The user can update and expand this library by using the computer system's UPDATE²⁵ routine. Control Deck 2 of Appendix B updates both source-code and object-code libraries and runs a RAYMATH program to test library modifications. The user can add and check out new subroutine packages with this control deck.

The execution of FORPROG begins with the initialization of the common block. Then follow calls to subroutine A319 alternating with calls to the subroutine packages. Subroutine A319 reads one record from TAPE18 and stores it in the common block of memory. As the reader may recall, a record on TAPE18 contains the original RAYMATH coding line or statement card plus the type and identification number of each argument of the instruction. Printed execution output consists of each coding line and its execution. This output usually serves as a permanent record of the execution of the problem. At the end of the program, output goes to the line printer file, OUTPUT, the card punch file, PUNCH, and the CALCOMP plotter file, TAPE7. Remote-keyboard-terminal access is through the PUNCH file; it is not normally possible to punch data cards from the terminal.

V. EXAMPLES OF RAYMATH PROGRAMS

Figure 4 shows a sample program that typifies many RAYMATH codes. The code plots the yield of gamma rays from neutron capture in ⁹³Nb as a function

of neutron energy between 4 and 5 kiloelectron volts (keV). These data were acquired in a self-indication measurement done in August 1969.² Time of flight over a 250-m flight path determined the neutron energy. Glass et al.¹ have described the recording techniques and instrumentation used for this type of measurement.

The main program begins with a series of comments that define the problem. LASTTITLE1 declares the name of the last active block on the library tape. Subprogram calls refer to subprograms designed to prepare and plot data and to calculate and plot estimated errors. The comments for each statement usually explain the action of the statement. Exceptions are tape and graph titles. The abbreviated fixed format of subprogram GENERATE ERROR BARS emphasizes statement syntax. In this subprogram, YROLLEDTOA moves the Y data set (arrays YX and YY) to locations occupied by the A data set (arrays AX and AY). The operation C1,*,YY multiplies every element of YY by the constant C1. The instruction LEGALORDER acts on the Y-arrays to remove illegal points and to order the remaining points in descending order on YX. The statement AX,TO,AY transfers the AX array to the storage locations of the AY array. MELDYANDA combines the Y- and A-arrays and stores the results in the Y-arrays.

In the PLOT RESULTS subprogram, MPlot2 produces a scroll graph on microfilm, and CALCOM2 writes a tape for off-line plotting on the CALCOMP plotter. Figure 5 shows the CALCOMP plot. The ordinate goes from -1.6 to 6.4 barns. The circular symbols on every 20th point indicate the estimated $\pm 15\%$ error. Program translation and execution takes a total of 28 sec of CDC-6600 central processor time.

A RAYMATH program to correct nuclear-particle detector output for a small background signal illustrates data-set processing that requires interpolation. Storing the detector output data in the A-arrays is the first step in the problem. Array AY contains the output current in amperes, and the array AX contains the corresponding neutron energy in electron volts. The background current versus neutron energy resides in the Y-arrays. A plot of part of these data may appear as shown below. The Δ -shaped points are the results of the background subtraction.

Preceding the subtraction, the Y-data set is interpolated at AX values. Assuming that the Y- and

OPEN FORMAT

```

...PROBLEM DESCRIPTION.
...READ TAPE FOR NB-1 GAMMA YIELD(BARNS) VS
...ENERGY DATA IN THE RANGE 4100-5100 EV.
...MAKE NECESSARY CORRECTIONS ON ENERGY
...AND GAMMA YIELD(BARNS).
...PLOT 15 PERCENT ERROR BARS ON GRAPHS OF
...GAMMA YIELD(BARNS) VS ENERGY(EV)
...USING MPLO2 AND CALCOM2.

```

```

, LASTTITLE1,,, *L14, SH8, C, NB1, CT, 220-37EV, 8/14/70

```

```

, GOTO,,, PROBLEM SETUP

```

```

, GOTO,,, CORRECT DATA

```

```

, GOTO,,, GENERATE ERROR BARS

```

```

, GOTO,,, PLOT RESULTS

```

```

, END,,, END OF MAIN PROGRAM.

```

```

... SUBROUTINE LIBRARY.

```

```

, HEREIS , , , GENERATE ERROR BARS
, , , , GENERATE ERROR BARS SUBROUTINE.
, , , , 15PERCENT ERROR BARS GENERATED.
, , , , ERROR BAR ARRAY IS IN A-ARRAYS.
, , , , DATA IN Y-ARRAYS.
, YROLLEDOA , , , *SAFE STORAGE FOR Y-ARRAYS.
, WRITETAPEZ,, , ,
, READ , , , C1 , 0.
, CL, , , Y , , , ZERO THE YY-ARRAY.
, Y, TO , , , YX , , , ZERO THE YX-ARRAY.
, READ , , , NSTEP , 20.
, AROLLEDOY , , , MOVE EVERY 20TH POINT FROM A TO Y-ARRAY.
, READ , , , NSTEP , 1.
, LEGALORDER , , , RESET NSTEP.
, YROLLEDOA , , , REMOVE ZERO POINTS FROM Y-ARRAY.
, READ , , , C1 , 1.15
, CL, , , Y , , , UPPER LIMITS ON ERROR BARS.
, Y, TO , , , AX , , , *SAFE STORAGE.
, READ , , , C1 , 0.85
, CL, , , Y , , , LOWER LIMITS ON ERROR BARS.
, , , , LOWER LIMITS STORED IN YY.
, AX, TO , , , AY , , , UPPER LIMITS ON ERROR BARS STORED IN AY.
, YX, TO , , , AX , , ,
, READ , , , PRTN0 , 1.
, MELOYAND0 , , , COMBINE UPPER AND LOWER LIMITS.
, PRINT
, YROLLEDOA , , , ERROR BAR ARRAY STORED IN A-ARRAY.
, WRITETAPEZ , , , *SAFE STORAGE FOR Y-ARRAYS.
, PRINT
, RETURN

```

```

... (CONTINUATION OF SUBROUTINE LIBRARY)

```

```

, HEREIS,,, PROBLEM SETUP
...PROBLEM SETUP SUBROUTINE.
, READ, PRTN0, 1., OBTAIN CHECKER DIAGNOSTICS FOR READTAPE1.
, READTAPE1,,, *NB1, CAPXSEC, SH8, L14CTC, H1 ENERGY6/03/70
, PRINT
...READ MAX AND MIN ENERGY VALUES.
, READ, EMIN, 4100.
, READ, EMAX, 5100.
, EMINMAX,,, SELECT GROUP OF Y-ARRAY DATA.
, READ, PRTN0, 1.
, PRINT
, RETURN

```

```

, HEREIS,,, CORRECT DATA
...DATA CORRECTION SUBROUTINE.
...MULTIPLY ENERGY BY 0.9769
, YROLLEDOA,,, Y-ARRAYS PLACED IN A-ARRAYS.
, READ, C1,,, 9769
, AX, *C1,,, PRODUCT IS STORED IN YY.
, Y, TO, YX,,, REPLACE YX WITH CORRECTED VALUE.
, READ, PRTN0, 1.
, PRINT,,, CHECK RESULTS.
...MULTIPLY GAMMA YIELD(BARNS) BY 1.10.
, READ, C1, 1.1
, AY, *C1,,, PRODUCT STORED IN YY.
, READ, PRTN0, 1.
, PRINT,,, PRINT CORRECTED Y-ARRAYS.
, RETURN

```

```

, HEREIS,,, PLOT RESULTS
...PLOTTING SUBROUTINE
...READ CONSTANTS FOR MPLO2.
, READ, KEVID1, 1., LINEAR-LINEAR GRAPH.
, READ, C4, 0.00, NUMBER OF EV PER PLOT.
, READ, YLIMLR, -1., MIN ORDINATE.
, READ, YLIMUP, 5.0, MAX ORDINATE.
, READ, LINE100, 0.
, READ, LINE101, 0., POINTS NOT CONNECTED BY LINE.
, READ, TITLE, GAMMA YIELD(BARNS) VS ENERGY(EV), 3950-
, READ, COMMENT, 5000 EV FOR NB-1. 15 PERCENT ERRORS.
, PLOT2,,, 2., PLOT A AND Y-ARRAYS ON SAME GRAPH
...WITH GRID.
...READ CONSTANTS FOR CALCOM2.
...VALUES OF KEY101, YLIMUP, YLIMLR, LINE100,
...LINE101, TITLE, AND COMMENT, SAME AS
...FOR MPLO2.
, READ, C1, 0.0, NO GRID.
, READ, C4, 0.07, INCHES PER 5 EV.
, CALCOM2,,, 2., PLOT A- AND Y-ARRAYS ON SAME GRAPH
...WITHOUT GRID.
, RETURN

```

Fig. 4. A RAYMATH code for plotting.

A-arrays are already loaded, the RAYMATH code is as follows:

```

, INTERPLATE,,, INTERPOLATION
, AROLLEDOY,,, RELOAD YX, YI, AND YF

```

```

BY, TO, YY , , , TRANSFER INTERPOLATED
, , , , VALUES TO YY
AY, -, YY , , , PERFORM SUBTRACTION
, PRINT , , , PRINT A SAMPLE OF THE RESULTS

```

A comment describes the function of each statement. As can be seen on the graph, the automatic boundary condition controls of INTERPLATE have caused the two points on the left-hand side of the A data set to be disregarded because they were outside the boundaries of the Y-data set.

Evaluation of an algebraic formula over a large parameter range is easy in RAYMATH. A representative example is the calculation of neutrons per second incident on the ^{93}Nb sample used in the above measurement. Figure 6 shows a program that performs this calculation. As before, the comment in field 5 describes the action of each statement.

Fig. 5. CALCOMP plotter output.

```

OPEN
,,,CALCULATE NEUTRONS/SEC FROM LI BEAM
,,,MONITOR DATA.
,READ,,,READ LI CURRENT VS ENERGY CARDS.
,PRINT
,,,READ CONSTANTS NEEDED FOR CALCULATIONS
,,,IN NEUT/SEC CALCULATOR SUBROUTINE.
,READ,C1,.21428E-17,
,READ,C3,.34390E-21,
,GOTO,,,NEUT/SEC CALCULATOR
,END

,HEREIS,,,NEUT/SEC CALCULATOR
,,,CONVERT I VS EV TO NEUT/SEC VS EV.
YX,TO,CY,,SAFE STORAGE FOR ENERGY VALUES.
YX,EXCHANGE,YY,,STORE CURRENT VALUES IN YX
,,,AND ENERGY VALUES IN YY.
,SQRT,YY,, SQRT(ENERGY IN EV)
,INVERT,YY,, 1./SQRT(E)
,PRINT,,,PRINT Y-ARRAYS.
C1,*,YY,, C1*(1./SQRT(E)).
YY,-,C3,, (C1*(1./SQRT(E))-C3.
,PRINT
YX,/,YY,,BEAM INTENSITY=I/((C1*(1./SQRT(E))-C3).
,PRINT,,,PRINT NEUTS/SEC VS I.
CY,TO,YX,,ENERGY VALUES RETURNED TO YX-ARRAY.
,PRINT,,,PRINT NEUTS/SEC VS ENERGY.
,,,END NEUT/SEC CALCULATOR.
,RETURN

```

Fig. 6 Evaluation of a formula.

The program evaluates the formula

$$\phi = \frac{I}{\frac{0.214248 \times 10^{-17}}{\sqrt{E}} - 0.3429 \times 10^{-21}}$$

ϕ is neutrons/sec, I is beam monitor output in amperes, and E is neutron energy in electron volts.

ACKNOWLEDGMENTS

It is a pleasure to acknowledge the support and interest of our colleagues in Group J-16. We acknowledge the work of G. L. Carter (Group C-2) in writing the RAYMATH free-format decoder.

APPENDIX A. RAYMATH SYNTAX

A. GENERAL

RAYMATH instruction cards have five fields. The first and third fields are operands or arguments. The second field is reserved for operators (modular instructions). Only one operator per card is allowed. The fourth field is for entry of numeric constants, and the fifth field is reserved for a maximum of 40 characters of Hollerith information expressed in FORTRAN alphanumeric and/or special characters. FORTRAN statements may be embedded in RAYMATH programs, provided that RAYMATH syntax is obeyed.

B. STATEMENT CARD FORMAT DECLARATIONS

The following declarations must begin in column one on the first card of the RAYMATH program deck. The program may contain either free-field or fixed-field formatted program cards, but not a mixture of both. Free-field formatted cards are prechecked during decoding. Illegal field entries are taken into account, and an attempt to correct them is made; see Sec. C. The fixed-field formatted program cards are not prechecked, so FORTRAN execution errors and diagnostics occur if nonnumeric data are placed in FIELD 4 of these cards.

Declaration	Explanation
OPEN	This declares that the program is written in free-field format. Only a line printer output file is generated. Card output from the PUNCH module is written on TAPE12 and subsequently punched.
OPENTELE	Like OPEN except that TAPE12 becomes the output file for the remote terminal. Cards from the PUNCH module are listed on TAPE12 but are not punched.
Blank card or first statement card	Fixed-field formatted statements are declared. Only a line printer output is generated. Card output from the PUNCH module is written on TAPE12 and subsequently punched. Program cards must be formatted as follows: 3XA2, 2A10, E15.6, 4A10 That is, FIELD 1=A2, FIELD 4=E15.6, and FIELD 2=A10, FIELD 5=4A10. FIELD 3=A10,

C. DESCRIPTION OF THE FREE-FIELD FORMAT

- Columns 1-72 may be used. Columns 73-80 are ignored.
- Five fields or less are permitted

	FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
Maximum Column Length	5	10	10	15	40

3. FIELD 1 begins with the first nonblank character of the statement. Only the two left-most nonblank characters of the field are used.
4. All other fields begin with the first nonblank character following a comma. The field-length count begins with this nonblank character.
5. A field ends upon encountering a comma (except for FIELD 5; see below), a trailing string of blanks through column 72, or column 73, whichever occurs first. Blank cards are allowed.
6. FIELD 4 is read by an E or F format. All other fields are read by an A format.
7. A comma in column 1 represents a blank FIELD 1. Elsewhere, commas in adjacent columns represent blank fields.
8. All commas following the fourth will be treated as part of FIELD 5.
9. Example: FIELDS 1, 3, and 4 are blank. FIELD 5 is terminated by a trailing string of blanks through column 72.

,PRINT,,,FIELD 2 IS NOT BLANK

10. Sequential program action
 - a. Subroutine FREEB is called from routine TELERAY following an OPEN or OPENTELE declaration.
 - b. Fields are identified and checked for valid length and format.
 - c. Card is decoded.
11. Default action and error diagnostics
 - a. In a given field, if the number of characters exceeds the maximum length of that field, truncation on the right occurs and a warning is given except for FIELD 5. If FIELD 5 exceeds 40 characters, it is truncated on the right but no warning is given. Card is decoded.
 - b. The numeric (4th field) is scanned for a valid E or F format. If the format is incorrect, the card will not be decoded and an error diagnostic is given. A blank card image is returned to routine TELERAY.
 - c. In FIELD 4, if a number in an E format does not have a delimiting comma on the right, trailing blanks will be counted up

to a field length of 15 columns. A FORTRAN execution diagnostic will probably occur, thus terminating program execution.

D. RULES FOR EMBEDDING FORTRAN STATEMENTS IN A RAYMATH PROGRAM

FORTRAN statements may appear anywhere in a RAYMATH code. These statements are treated exactly like RAYMATH statements through the RAYMATH program assembly phase and then are passed unaltered to the constructed FORTRAN program FORPROG. Embedded FORTRAN must adhere to the syntax of RAYMATH as follows:

Field	Contents	Explanation
1	* or \$	An * indicates a FORTRAN comment, and the \$ a FORTRAN statement. The symbol must be left-adjusted in the field.
2	Statement Number	First four columns of the field contain the statement number.
3	Continuation	Continuation must be in the first column of this field.
4		Not used
5		A FORTRAN statement or comment that does not exceed 40 characters in length.

As an example, consider a program for printing constant C1. The program begins with a comment and contains a continued FORMAT statement.

```

*,*,*, PRINT C1 USING EMBEDDED FORTRAN CODING
$,*,*, WRITE (11, 20) C1
$,*,*, IF (NTELE. EQ. 1) WRITE (12, 20) C1
$,20,,, FORMAT (* PRINT A CONSTANT *
$,X,, *C1 = * E12.5)

```

By keeping in mind the structure of routine FORPROG one can add FORTRAN statements at desirable locations in FORPROG and even add additional subroutines called by FORPROG. Suitable RAYMATH comments should be included to indicate the presence of embedded FORTRAN statements in the execution output.

APPENDIX B. CONTROL DECK CHARTS

A. GENERAL COMMENTS

The following control deck charts give enough information to create and maintain the FORTRAN source decks for the RAYMATH preprocessor and instruction modules and to run RAYMATH in either the local or remote batch modes. Input and output file structures are described in detail in Sections 2 and 3 of each chart. Individual control cards are described in Vol. 4 of the LASL Programmers' Information Manual. A few words here about several of the control cards may be useful. The control card UPDATE²⁵ is a systems code designed to create and maintain a library of FORTRAN source programs. COPYL is a similar systems routine, but it is used for updating an existing library of binary-object program decks. The control card USERLIB permits a designated file to be searched before the systems library, for external routines required by object codes at load time.

The general BJOB card referred to in each of the control deck charts is as follows:

FIELD	COLUMN(S)	CONTENT
	1	B
	2-4	
CATEGORY	5-6	03
TIME ESTIMATE	7-9	002 minutes
JOB,	10-13	JOB,
NAME	14-26	13 columns (1st Character Alphabetic)
PROBLEM NAME	27-33	TELERAY
GROUP	34-36	Alphabetic + 2 numerics
	37-39	
PRIORITY	40-41	06
	42	
FIELD LENGTH	43-48	60000, or greater depending on the particular job to be done or the particular RAYMATH program
	49-53	
CLASSIFICATION	54	U
	55-63	
PAGE LIMIT	64-66	≥10

Tape assignments for RAYMATH data-library tapes, TAPE1 and TAPE2, and for the CALCOMP output tape must be made before the RAYMATH-generated FORTRAN program is loaded and executed. These assignments may be placed immediately after the BJOB card.

When a job is run from a remote-keyboard terminal, it is not possible to obtain properly formatted punched cards from the PUNCH module of RAYMATH. If punched cards are desired, the job must be rerun in the local batch mode. This provision was made to conserve file names and buffer space.

File name INPUT is reserved for card-reader input. File name OUTPUT is reserved for a file that is copied to the line printer at the conclusion of the job. Transfer of files to the output printer is duplicated after the EXIT control card in order to have existing output copied to the printer in case of a fatal FORTRAN or control-card error.

B. Control Deck Number: 1

1. Purpose: To create an UPDATE library from FORTRAN source decks.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	
COL 1-42		See general BJOB card.
COL 43-48		Field length = 60000.
COL 49-80		See general BJOB card.
ASSIGN MT,NEWLIB(NLB,LJ249L00,SHB)	2	Assign a magnetic tape for the new library.
COMMENT.	3	
UPDATE(N=NEWPL)	4	Create new UPDATE library on file NEWPL.
COMMENT.	5	
RUN(S,I=COMPILE,B=BLIB)	6	Compile source decks.
COMMENT.	7	
REWIND(BLIB)	8	
CATALOG(BLIB,,1)	9	List the length of each object deck.
COMMENT.	10	
REWIND(NEWLIB)	11	
REWIND(BLIB)	12	
REWIND(NEWPL)	13	
COPYBR(NEWPL,NEWLIB)	14	Copy UPDATE program library to tape NEWLIB.
COPY(BLIB,NEWLIB)	15	Copy object decks to tape NEWLIB.
COMMENT.	16	
REWIND(COMPILE)	17	
REWIND(NEWLIB)	18	
REWIND(BLIB)	19	
COPYBR(NEWLIB,COMPILE)	20	Position file NEWLIB.
COPY(NEWLIB,BLIB)	21	Copy object decks to file BLIB.
COMMENT.	22	
REWIND(BLIB)	23	
CATALOG(BLIB,,1)	24	Generate lists to check against those of card 9.
7 ₈ 9	25	Binary end of record

2. Input File Structure: The input file is INPUT and consists of two binary records.

- a. The first record contains the control cards.
- b. The second binary record contains UPDATE control cards and the FORTRAN source decks.

3. Output File Structure: The output file is OUTPUT and it contains the UPDATE report, the compilation of FORTRAN programs, and the catalog listing of the binary object decks.

C. Control Deck Number: 2

1. Purpose: To update the FORTRAN source program library and binary object program decks and run a test problem.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	
COL 1-42		See general BJOB card.
COL 43-48		Field length = 60000, for a small test problem. Card 44 has same field length.
COL 49-80		See general BJOB card.
ASSIGN MT,NLIB(PLB,LJ249L00,SHB)	2	Assign old program tape to file NLIB.
ASSIGN MT,NEWLIB(NLB,LD123I.00,SHB)	3	Assign new program tape to file NEWLIB.
COMMENT.	4	
COPYBR(NLIB,OLDPLIB)	5	Copy UPDATE library from NLIB to OLDPLIB.
COPY(NLIB,OLDBLIB)	6	Copy binary-object program decks from NLIB to OLDBLIB.
RELTAPE(NLIB)	7	Release file NLIB.
COMMENT.	8	
REWIND(OLDBLIB)	9	
CATALOG(OLDBLIB,,2)	10	Catalog the length of each binary-object program deck.
RFL(60000)	11	Restore field length.
COMMENT.	12	
UPDATE(P=OLDPLIB,N=NEWLIB)	13	Update old program library to NEWLIB.
RELTAPE(OLDPLIB)	14	Release file OLDPLIB.
COMMENT.	15	
RUN(S,I=COMPILE,B=LGO)	16	Compile updated FORTRAN decks.
RELTAPE(COMPILE)	17	Release file COMPILE.
COMMENT.	18	
REWIND(NEWLIB)	19	
REWIND(OLDBLIB)	20	
REWIND(LGO)	21	
COPYL(OLDBLIB,LGO,NEWLIB)	22	Replace old binary-object program decks with updated ones.
RELTAPE(LGO)	23	Release file LGO.
COMMENT.	24	
REWIND(NEWLIB)	25	
COPY(NEWLIB,NEWLIB)	26	Copy new binary-object program library to file NEWLIB.
RELTAPE(NEWLIB)	27	Release file NEWLIB.
COMMENT.	28	

CONTROL CARD	NO.	EXPLANATION
REWIND(NEWBLIB)	29	
CATALOG(NEWBLIB,,2)	30	Catalog the length of each binary-object program deck.
COMMENT.	31	
RFL(60000)	32	Restore field length.
COPYBF(INPUT,TAPE7)	33	Copy RAYMATH program cards to file TAPE7.
COMMENT.	34	
COPYBF(INPUT,TAPE10)	35	Copy RAYMATH array-data cards to file TAPE10.
COMMENT.	36	
REWIND(NEWBLIB)	37	
REWIND(OLDBLIB)	38	
COPYBR(NEWBLIB,OLDBLIB,5)	39	Copy first 5 object programs to file OLDBLIB.
OLDBLIB.	40	Load and execute file OLDBLIB.
RELTAPE(OLDBLIB)	41	Release file OLDBLIB.
COMMENT.	42	
RUN(3,I=TAPE19,B=DUM)	43	Compile RAYMATH-generated FORTRAN programs.
RFL(60000)	44	Field length must be same as on card 1.
REWIND(NEWBLIB)	45	
USERLIB(NEWBLIB)	46	Create a user library out of NEWBLIB.
DUM.	47	Load and execute object program file created by card 43.
COMMENT.	48	
REWIND(TAPE11)	49	
COPY(TAPE11,OUTPUT)	50	Copy line-printer output to line printer.
REWIND(TAPE12)	51	
COPY(TAPE12,OUTPUT)	52	Copy terminal-keyboard output to printer.
REWIND(TAPE18)	53	
COPYSBF(TAPE18,OUTPUT)	54	Copy type, ID, and statement card file to printer.
REWIND(TAPE19)	55	
COPYSBF(TAPE19,OUTPUT)	56	Copy RAYMATH-generated FORTRAN source decks to line printer.
REWIND(TAPE7)	57	
COPY(TAPE7,CALCOMP)	58	Copy CALCOMP plotter data to tape CALCOMP.
COMMENT.	59	
EXIT.	60	Normal program termination
REWIND(TAPE11)	61	} Same as cards 49-54 above
COPY(TAPE11,OUTPUT)	62	
REWIND(TAPE12)	63	
COPY(TAPE12,OUTPUT)	64	
REWIND(TAPE18)	65	
COPYSBF(TAPE18,OUTPUT)	66	

CONTROL CARD	NO.	EXPLANATION
REWIND(TAPE19)	67	Same as cards 55-58 above
COPYSBF(TAPE19,OUTPUT)	68	
REWIND(TAPE7)	69	
COPY(TAPE7,CALCOMP)	70	
7 ₈ 9	71	Binary end of record

2. Input File Structure: Input file is INPUT and consists of two binary records followed by two or more binary files.

- First binary record contains control cards.
- Second binary record contains UPDATE control cards and correction cards.
- First binary file contains RAYMATH program cards. It is advisable to use the OPENTELE declaration in this program to test the terminal output.
- Next binary file to end of information contains one or more files of RAYMATH array-data cards.

3. Output File Structure: Parallel file output is used with the following outputs contained in the first file of each of the units below:

- TAPE11: line printer output
- TAPE12: remote terminal output
- TAPE7: CALCOMP plotter output
- TAPE18: type, ID, and statement card file written by program TELERAY
- TAPE19: RAYMATH-generated FORTRAN source decks

D. Control Deck Number: 3

1. Purpose: To run in local batch mode from binary cards.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	See general BJOB card.
COL 1-42		Field length = 156000, for a large
COL 43-48		RAYMATH program. Card 8 has same
		field length.
COL 49-80		See general BJOB card.
COPYBF(INPUT,TAPE7)	2	Copy RAYMATH program cards to file TAPE7.
COPYBF(INPUT,TAPE10)	3	Copy array-data cards to file TAPE10.
COPYBR(INPUT,FIRST,5)	4	Copy first 5 object programs to file
		FIRST.
COPY(INPUT,MYLIB)	5	Copy remaining binary-object program
		decks to file MYLIB.
FIRST.	6	Load and execute file FIRST.

CONTROL CARD	NO.	EXPLANATION
RUN(S,I=TAPE19,B=DUM)	7	Compile RAYMATH-generated FORTRAN programs.
RFL(156000)	8	Field length must be same as on card 1.
REWIND(MYLIB)	9	
USERLIB(MYLIB)	10	Create a user library out of file MYLIB.
DUM.	11	Load and execute object program file created by card 7.
REWIND(TAPE11)	12	
COPY(TAPE11,OUTPUT)	13	Copy line-printer output to line printer.
REWIND(TAPE12)	14	
COPY(TAPE12,PUNCH)	15	Copy punch output to punch file.
REWIND(TAPE7)	16	
COPY(TAPE7,CALCOMP)	17	Copy CALCOMP plotter data to tape CALCOMP.
EXIT.	18	Normal program termination
REWIND(TAPE18)	19	
COPYSBF(TAPE18,OUTPUT)	20	Copy type, ID, and statement card file to printer.
REWIND(TAPE19)	21	
COPYSBF(TAPE19,OUTPUT)	22	Copy RAYMATH-generated FORTRAN source decks to line printer.
REWIND(TAPE11)	23	} Same as cards 12-17 above.
COPY(TAPE11,OUTPUT)	24	
REWIND(TAPE12)	25	
COPY(TAPE12,OUTPUT)	26	
REWIND(TAPE7)	27	}
COPY(TAPE7,CALCOMP)	28	
7 ₆ 9	29	Binary end of record.

2. Input File Structure: The input file is INPUT and consists of three binary files.

- a. The first file contains the RAYMATH program cards. Do not use the OPENTELE declaration in this program.
- b. The second file contains RAYMATH array-data cards.
- c. The third file contains the binary-object decks for the RAYMATH pre-processor and modules.

3. Output File Structure: Parallel file output is used with the following outputs contained in the first file of each of the units below:

- a. TAPE11: line-printer output
- b. TAPE12: punched card output
- c. TAPE7: Calcomp plotter output
- d. TAPE18: type, ID, and statement card file written by program TELERAY
- e. TAPE19: RAYMATH-generated source decks.

E. Control Deck Number: 4

1. Purpose: To run in local-batch mode from a binary-object program library residing on tape.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	See general BJOB card.
COL 1-42		Field length = 60000, for a small
COL 43-48		RAYMATH program. Card 13 has same
		field length.
COL 49-80		See general BJOB card.
RFL(60000)	2	
ASSIGN MT,NLIB(PLB,LB630L00,SHB)	3	Library tape containing UPDATE and binary
		object program libraries.
COPYBR(NLIB,FIRST)	4	Position file NLIB.
REWIND(FIRST)	5	
COPYBR(NLIB,FIRST,5)	6	Copy first 5 binary-object program decks
		to file FIRST.
COPY(NLIB,MYLIB)	7	Copy remaining binary-object program decks
		to file MYLIB.
RELTAPE(NLIB)	8	Release file NLIB.
COPYBP(INPUT,TAPE7)	9	Copy RAYMATH program cards to file TAPE7.
COPYBP(INPUT,TAPE10)	10	Copy RAYMATH array-data cards to file
		TAPE10.
FIRST.	11	Load and execute file FIRST.
RUN(S,I=TAPE19,B=DUM)	12	Compile RAYMATH-generated FORTRAN programs.
RFL(60000)	13	Field length must be the same as on card 1.
REWIND(MYLIB)	14	
USERLIB(MYLIB)	15	Create a user library out of MYLIB.
DUM.	16	Load and execute object program file
		created by card 12.
REWIND(TAPE11)	17	
COPY(TAPE11,OUTPUT)	18	Copy line-printer output to line printer.
REWIND(TAPE12)	19	
COPY(TAPE12,PUNCH)	20	Output punched cards.
REWIND(TAPE7)	21	
COPY(TAPE7,CALCOMP)	22	Copy CALCOMP plotter data to tape CALCOMP.
EXIT.	23	Normal program termination
REWIND(TAPE18)	24	
COPYSBF(TAPE18,OUTPUT)	25	Copy type, ID, and statement card file
		to printer.
REWIND(TAPE19)	26	
COPYSBF(TAPE19,OUTPUT)	27	Copy RAYMATH-generated FORTRAN source
		decks to line printer.
REWIND(TAPE11)	28	
COPY(TAPE11,OUTPUT)	29	Same as cards 17 and 18 above.

CONTROL CARD	NO.	EXPLANATION
REWIND(TAPE12)	30	Same as cards 19-22 above.
COPY(TAPE12,OUTPUT)	31	
REWIND(TAPE7)	32	
COPY(TAPE7,CALCOMP)	33	
7 ₈ 9	34	Binary end of record.

2. Input File Structure: The input file consists of two binary files.
 - a. The first file contains the RAYMATH program cards. Do not use the OPENTELE declaration in this program.
 - b. The second file contains RAYMATH array-data cards.
3. Output File Structure: Parallel file output is used with the following outputs contained in the first file of each of the units below:
 - a. TAPE11: line-printer output
 - b. TAPE12: punched card output
 - c. TAPE7: CALCOMP plotter output
 - d. TAPE18: type, ID, and statement card file written by program TELERAY
 - e. TAPE19: RAYMATH-generated source decks.

F. Control Deck Number: 5

1. Purpose: To run in local-batch mode from binary decks residing on machine 2 permanent files.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	See general BJOB card.
COL 1-42		Field length = 60000 ₉ for a small
COL 43-48		RAYMATH program. Maximum is 130000 ₉ .
		Card 22 has same field length.
COL 49-80		See general BJOB card.
RPL(60000)	2	Set field length to 60000 ₉ .
ASSIGN AB,J16MVH2(,MARXXXXXX)	3	Assign permanent file holding binary-object program decks.
COMMENT.	4	
COPYBR(J16MVH2,FIRST,5)	5	Copy first 5 binary-object program decks to file FIRST.
COPY(J16MVH2,MYLIB)	6	Copy remaining binary-object program decks to file MYLIB.
RETURN(J16MVH2)	7	Return permanent file J16MVH2.
COPYBF(INPUT,TAPE7)	8	Copy RAYMATH program cards to file TAPE7.
COPYBF(INPUT,TAPE10)	9	Copy RAYMATH array-data cards to file TAPE10.

CONTROL CARD	NO.	EXPLANATION
COMMENT. SAVE PROGRAM AND DATA DECKS	10	
ASSIGN AB,J16MVH(,MARVHAXXXX)	11	Assign permanent file J16MVH.
ASSIGN AB,J16MVH3(,MARVHAXXXX)	12	Assign permanent file J16MVH3.
REWIND(TAPE7)	13	
COPY(TAPE7,J16MVH)	14	Copy RAYMATH program cards to file J16MVH.
REWIND(TAPE10)	15	
COPY(TAPE10,J16MVH3)	16	Copy RAYMATH array-data cards to file J16MVH3.
RETURN(J16MVH)	17	Return permanent file J16MVH.
RETURN(J16MVH3)	18	Return permanent file J16MVH3.
COMMENT.	19	
FIRST.	20	Load and execute file FIRST.
RUN(S,I=TAPE19,B=DUM)	21	Compile RAYMATH-generated FORTRAN programs.
RFL(60000)	22	Field length must be the same as on card 1.
REWIND(MYLIB)	23	
USERLIB(MYLIB)	24	Create a user library out of MYLIB.
DUM.	25	Load and execute object program file created by card 21.
REWIND(TAPE11)	26	
COPY(TAPE11,OUTPUT)	27	Copy line printer-output to line printer.
REWIND(TAPE12)	28	
COPY(TAPE12,PUNCH)	29	Output punched cards.
REWIND(TAPE7)	30	
COPY(TAPE7,CALCOMP)	31	Copy CALCOMP data to tape CALCOMP.
EXIT.	32	Normal program termination
REWIND(TAPE1)	33	
COPYSEF(TAPE16,OUTPUT)	34	Copy tape, ID, and statement card file to printer.
REWIND(TAPE19)	35	
COPYSEF(TAPE19,OUTPUT)	36	Copy RAYMATH-generated FORTRAN source decks to line printer.
REWIND(TAPE11)	37	Same as cards 26-31 above
COPY(TAPE11,OUTPUT)	38	
REWIND(TAPE12)	39	
COPY(TAPE12,OUTPUT)	40	
REWIND(TAPE7)	41	
COPY(TAPE7,CALCOMP)	42	
789	43	Binary end of record

2. Input File Structure: The input file is INPUT and consists of a binary record followed by two or more binary files.

- a. Binary record contains the control cards.
- b. First binary file contains the RAYMATH program cards. Do not use the OPENTELE declaration in this program.
- c. Next binary file to end of information contains one or more files of RAYMATH array-data cards.

3. Output File Structure: Parallel file output is used with the following outputs contained in the first file of each of the units below:

- a. TAPE11: line-printer output
- b. TAPE12: punched card output
- c. TAPE7: CALCOMP plotter output
- d. TAPE18: type, ID, and statement card file written by program TELERAY
- e. TAPE19: RAYMATH-generated source decks
- f. J16MVH: RAYMATH program cards
- g. J16MVH3: RAYMATH array-data cards

G. Control Deck Number: 6

1. Purpose: To run in a remote-batch mode from a MUX keyboard terminal.

CONTROL CARD	NO.	EXPLANATION
BJOB	1	
COL 1-42		See general BJOB card.
COL 43-48		Field length = 125000, for a medium-size RAYMATH program. Maximum is 130000. Card 19 has same field length.
COL 49-80		See general BJOB card.
RFL(60000)	2	
ASSIGN AB,J16MVH2(,MARXXXXXX)	3	Assign permanent file holding binary object program decks.
ASSIGN AB,J16MVH(,MARXXXXXX)	4	Assign permanent file holding RAYMATH program cards.
ASSIGN AB,J16MVH3(,MARXXXXXX)	5	Assign permanent file holding RAYMATH data cards.
COMMENT.	6	
COPYBR(J16MVH2,FIRST,5)	7	Copy first 5 binary-object program decks to file FIRST.
COPY(J16MVH2,MYLIB)	8	Copy remaining binary-object program decks to file MYLIB.
RETURN(J16MVH2)	9	Return permanent file J16MVH2.
COMMENT.	10	
COPY(J16MVH,TAPE7)	11	Copy RAYMATH program cards to file TAPE7.
COPY(J16MVH3,TAPE10)	12	Copy RAYMATH array-data cards to TAPE10.
RETURN(J16MVH)	13	Return permanent file to J16MVH.
RETURN(J16MVH3)	14	Return permanent file J16MVH3.

CONTROL CARD	NO.	EXPLANATION
COMMENT.	15	
FIRST.	16	Load and execute file FIRST.
RUN(S,I=TAPE19,B=DUM)	17	Compile RAYMATH-generated FORTRAN programs.
COMMENT.	18	
RFL(125000)	19	Field length must be the same as card 1.
REWIND(MYLIB)	20	
USERLIB(MYLIB)	21	Create a user library out of MYLIB.
DUM.	22	Load and execute object program file created by card 17.
REWIND(TAPE11)	23	
COPY(TAPE11,OUTPUT)	24	Copy line-printer output to line printer.
REWIND(TAPE12)	25	
COPY(TAPE12,PUNCH)	26	Copy terminal output to file accessible to terminal.
REWIND(TAPE7)	27	
COPY(TAPE7,CALCOMP)	28	Copy CALCOMP plotter data to tape CALCOMP.
EXIT.	29	Normal program termination.
REWIND(TAPE18)	30	
COPYSBF(TAPE18,OUTPUT)	31	Copy type, ID, and statement card file to printer.
REWIND(TAPE19)	32	
COPYSBF(TAPE19,OUTPUT)	33	Copy RAYMATH-generated FORTRAN source decks to line printer.
REWIND(TAPE11)	34	} Same as cards 23-28 above.
COPY(TAPE11,OUTPUT)	35	
REWIND(TAPE12)	36	
COPY(TAPE12,PUNCH)	37	
REWIND(TAPE7)	38	
COPY(TAPE7,CALCOMP)	39	
78 9	40	Binary end of record

2. Input File Structure: Parallel file input is used.

- a. INPUT: The control deck given above is the first binary record.
- b. J16MVH: RAYMATH program cards are written in the first file.
Use OPENFELE declaration in the RAYMATH program.
- c. J16MVH3: RAYMATH array-data card decks are written in successive files beginning at the front of J16MVH3.

3. Output File Structure: Parallel file output is used with the following outputs contained in the first file of each of the units below:

- a. TAPE11: line-printer output
- b. TAPE12: remote-terminal output

- c. TAPE7: CALCOMP plotter output
- d. TAPE18: type, ID, and statement card file written by program TELERAY
- e. TAPE19: RAYMATH-generated source decks

APPENDIX C. CHARTS FOR MODULAR INSTRUCTIONS

Each of the RAYMATH instructions shown in Table I is explained in detail in a Module Chart. Table C-I is the glossary of Module Charts. It serves as a telegraphic view of each instruction, giving the Module Chart Number, the principal routine name, possible array arguments, and the location of results for the array instructions. Table C-II is a listing and explanation of the statement-card format declarations.

1. READ

A. Purpose: To read array numbers from cards.

(1) Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	READ			May be used for a comment

(2) Quantities that must be set beforehand

- a. Punch data cards with three YX,YY pairs per card in 6E12.5 format.
- b. Terminate each deck with a card containing a -1. in the first field of 12 columns or with an end-of-file (7-8-9 level 17) card.
- c. Set PRTNO equal to 1. to obtain diagnostics from subroutine CHECKER. See section 7.

(3) Example: Read data cards.

,READ

(4) Sequential program action

- a. Number of data cards is counted by subroutine TANDCRD. -1., and EOF terminating cards are not included in the count. YF is set to three times the number of cards read.

- b. Points are checked for legality and order by subroutine CHECKER. YF is reset if any illegal points are encountered.

(5) Boundary conditions

- a. No more than LENGTHY/3 cards can be read in a data block. Usually LENGTHY = 6000, so maximum number of cards, exclusive of -1. or EOF card, is 2000.
- b. Each data block must have a terminator.
- c. Data must satisfy restrictions of subroutine CHECKER. Data block should contain at least two legal points.
- d. Data block begins at YI = 1.

(6) Default action and error diagnostics

- a. If number of cards exceeds LENGTHY/3, the number of points is truncated to LENGTHY and the READ module is locked out; i.e., no more data may be read from cards, disk files, or tape files.
- b. If the block contains only a -1. or EOF card with no data cards, a diagnostic is given, YF is not reset, and execution is continued.
- c. If the data block contains only one legal point, then BAD TABLE is indicated, YF is not reset, and execution is continued.

(7) Arguments whose values may be changed by the operation

- a. YX,YY, and YF all may be changed by the CHECKER subroutine if data contain negative or zero YX, or if the data are improperly ordered.
- b. If a list of illegal and misordered points is desired, set PRTNO=1.0. It is set to zero at the completion of the CHECKER subroutine.
- c. Module is not executed if the second argument (FIELD 3) is illegal. An error diagnostic is given, and the program execution continues.
- d. Data block begins at YI = 1.

TABLE C-I. MODULE CHART GLOSSARY

<u>Module Chart Number</u>	<u>FORTTRAN Entry or Subroutine Name</u>	<u>Module Name</u>	<u>Operates on Arrays</u>	<u>Range of Operation</u>	<u>Location of Results</u>
1	A315	READ	YX,YY	YI-YF	YX,YY,YI,YF
2	A316	PRINT	Y _A ,YY	YI-YF	
3	A320	PRALLARRAY	All	NI-NF	
4	A302	+	All	YI-YF	YY
4	A303	-	All	YI-YF	YY
4	A304	*	All	YI-YF	YY
4	A305	/	All	YI-YF	YY
5	A317	(Null or Blank)			
6	A312	INVERT	All	YI-YF	Operand
4	A306	**	All	YI-YF	YY
7	A307	SQRT	All	YI-YF	YY
8	A309	LOGE	All	YI-YF	YY
9	A311	EXP	All	YI-YF	YY
10	A322	SPLIT	YX,YY	YI-YF	
11	A323	PLOT	YX,YY	YI-YF	
12	A324	SETGRIDY			
13	A325	SETGRIDXY			
14	A314	EMINMAX	YX,YY	YI-YF	YI,YF
15	A348	POINTER	YX,YY	YI-YF	YX,YY,YI,YF
16	A326	READTAPE1	YX,YY	YI-YF	YX,YY,YI,YF
16	A346	READTAPE2	YX,YY	YI-YF	YX,YY,YI,YF
17	A327	WRITETAPE1	YX,YY	YI-YF	
17	A347	WRITETAPE2	YX,YY	YI-YF	
18	A328	MPLT2	YX,YY,AX,AY	YI-YF,AI-AF	
19	A349	CALCOM2	YX,YY,AX,AY	YI-YF,AI-AF	
18	MPLTCAL	MPLT	YX,YY	YI-YF	
19	CALCALL	CALCOMP	YX,YY	YI-YF	
20	A329	DUMP			
21	A330	PUNCH	YX,YY	YI-YF	
22	A332	INTEGRATE	YX,YY	YI-YF	C2
23	A333	LASTTITLE1			
23	A350	LASTTITLE2			
24	A334	LOADYLIMUP	YX,YY	YI-YF	YLIMUP
25	A351	LOADYLIMLR	YX,YY	YI-YF	YLIMLR
26	A336	AVOVERTIME	YX,YY	YI-YF	AX,AY,AI,AF
27	A335	YROLLEDTOA	YX,YY	NI-NF	AX,AY,AI,AF
28	A352	AROLLEDTOY	AX,AY	NI-NF	YX,YY,YI,YF
29	A337	RUNSUM	YX,YY	YI-YF	
30	A353	RESINTEGRAL	YX,YY	YI-YF	C2
31	A338	NOEQUALX	YX,YY	YI-YF	YX,YY,YI,YF
32	A340	PRINTPLOT	YX,YY	YI-YF	
33	A341	MELDYANDA	YX,YY,AX,AY	YI-YF,AI-AF	YX,YY,YI,YF
34	A342	DELETEx=	YX,YY	YI-YF	YX,YY

TABLE C-I (Continued)

Module Chart Number	FORTTRAN Entry or Subroutine Name	Module Name	Operates on Arrays	Range of Operation	Location of Results
35	A343	CLOCK			
36	A344	A344			
36	A870	A870			
36	A872	A872			
36	A874	A874			
36	A876	A876			
36	A888	A888			
36	A900	A900			
36	A902	A902			
37	A300	EXCHANGE	All	NI-NP	Operands
38	A301	TO	All	NI-NP	FIELD 3 Operand
39	CHECKER	LEGALORDER	YX,YY	YI-YF	YX,YY,YI,YF
40	SEARCH	INTERPLATE	YX,YY,AX,AY	YI-YF	BY,BI,BF
41	TELERAY	GOTO			
41	TELERAY	HEREIS			
41	TELERAY	END			
41	TELERAY	RETURN			

TABLE C-II. STATEMENT-CARD FORMAT DECLARATIONS

Declarations	Description
OPEN	Free-field-formatted modular instructions
OPENTELE	Same as OPEN, but a terminal output file is written.
(Blank or First Statement)	Fixed-field-formatted modular instructions

B. Purpose: To read a CONSTANT, DOLOOPLMT, PARAMETER, NUMBER, or an AFORMAT type argument.

(1) Instruction and permissible arguments

a.

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	READ	CONSTANT, DOLOOPLMT, PARAMETER, or NUMBER type argument	Value of argument in E or F format	May be used for a comment

b.

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	READ	AFORMAT type argument		Title or comment

(2) Example:

a. Load constant C4 with the value 3.2E-4.

```
,READ,C4,3.2E-4,
```

Note: The comma on the right is mandatory.

b. Load TITLE, an AFORMAT type operand.

```
,READ,TITLE,,CURRENT IN AMPS VS ENERGY IN EV
```

(3) Default action and error diagnostic: See 1.A.(7)c above.

2. PRINT

A. Purpose: To print Y-arrays.

(1) Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PRINT			May be used for a comment

(2) Quantities that may be set beforehand:

PRTNO

(3) Example:

.PRINT

(4) Sequential program action

a. The Y-arrays are printed over an index range of YI,YF by a printing interval PRTNO.
b. PRTNO may be set to any desired interval before the PRINT instruction.

c. If PRTNO is not set before the PRINT instruction, subroutine SELECTR sets the printing interval so that only one page of point pairs is printed.

d. PRTNO is reset to 0 at the conclusion of PRINT.

e. Five YX-YY pairs per line are printed on the line printer, and three pairs are printed on the remote-terminal output. An E12.5 format is used.

(5) Default action and error diagnostic: If FIELD 3 contains an illegal argument, the Y-arrays are printed and no diagnostic is given in the execution output.

B. Purpose: To print CONSTANT, DOLOOPINT, PARAMETER, NUMBER, or an AFORMAT type argument.

(1) Instruction and permissible arguments

a.

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PRINT	CONSTANT, DOLOOPINT, PARAMETER, or NUMBER type argument		May be used for a comment

b.

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PRINT	AFORMAT type argument		May be used for a comment

(2) Example: Print previously loaded TITLE.

.PRINT,TITLE

3. PRALLARRAY

A. Purpose: To print all six arrays.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PRALLARRAY			May be used for a comment

C. Quantities that may be set beforehand:

PRTNO

D. Example:

.PRALLARRAY

E. Sequential program action

(1) The arrays are printed over an index range NI,NF by an interval PRTNO.

(2) PRTNO can be set to any desired interval before the PRALLARRAY instruction.

(3) If PRTNO is not set before PRALLARRAY, subroutine SELECTR sets the printing interval. Usually NI = 1, NF = 6000, and PRTNO = 50.

(4) PRTNO is reset to 0 at the completion of PRALLARRAY.

4. ARITHMETIC OPERATIONS +, -, *, /, **

A. Purpose: To perform arithmetic operations on constants and arrays.

B. Instruction and permissible arguments

Field	Value	Explanation
1	Constant or Array	Argument 1
2	%, -, *, /, or **	The operations are possible with both arguments constants, both arguments arrays, or with one argument a constant and the other an array.
3	Constant or Array	Argument 2
4		
5		May be used for a comment

C. Example: Multiply an array by a constant.

YY, %, C3

D. Sequential program action

- (1) Arguments 1 and 2 are checked for type.
- (2) If both arguments are constants, the result of the arithmetic operation is stored in C2. Otherwise, the results are stored in YY.
- (3) For mixed-mode or array-array arithmetic, the operation is performed over an index range of Y1,YF by a step of 1.

E. Boundary conditions

- (1) All operations: Arguments must be constants or arrays.
- (2) Operation /: Second argument should not be zero.
- (3) Operation **:
 - a. First argument should be non-negative.
 - b. If 1st argument is zero, 2nd argument should not be negative or zero.
 - c. If 1st argument is zero and 2nd argument is positive, the result is defined to be 0.

F. Default action and error diagnostics

- (1) All operations: If argument 1 or argument 2 is illegal, module is skipped and a diagnostic is given. Program execution continues.

(2) Operation /: If argument 2 is zero, result is -0.0. An error diagnostic is given.

(3) Operation **: If argument 1 is negative, result is -0.0. If argument 1 is zero and 2nd argument is nonpositive, result is -0.0. Diagnostic is given.

G. Arguments whose values may be changed by the operation

All operations: If both arguments are constants, C2 is loaded with the result of the operation. YY is loaded with the result of any mixed-mode or array-array operation.

5. BLANK

A. Purpose: To allow insertion of blank cards or comments in program.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
				May be used for program comment

A blank card may be used for spacing program printout. One, two, three, or four commas may also be used for an open-field-formatted instruction.

C. Example: Write a program comment.

...,THIS IS A CHECKOUT OF TELERAY

6. INVERT

A. Purpose: To find the reciprocal of a constant or an array.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	INVERT	Constant or Array		May be used for a comment

C. Example: Invert array AY.

,INVERT,AY

D. Sequential program action

- (1) The operand is checked for type.
- (2) If the operand is an array, the operation is performed over an index range of YI,YF by a step of 1.
- (3) The result of the operation is stored in the constant or array, replacing the original argument.

E. Boundary conditions

- (1) The operand must be a constant or an array.
- (2) The operand should not have a value of zero.

F. Default action and error diagnostics

- (1) If the operand is not a constant or an array, the operation is skipped, a diagnostic is given, and execution is continued.
- (2) If the operand is zero, the operation is skipped and -0.0 is stored as the result.

G. Arguments whose values may be changed by the operation

The operation replaces the original argument with its reciprocal.

- (2) If the operand is a constant, the result of the operation is stored in C2.

- (3) If the operand is an array, the operation is performed over an index range of YI,YF by a step 1, and the results are stored in YY.

E. Boundary conditions

- (1) The operand must be a constant or an array.
- (2) The operand should not have a negative value.

F. Default action and error diagnostics

- (1) If the operand is not a constant or an array, the operation is skipped, a diagnostic is given, and execution is continued.
- (2) If the operand is negative, the operation is skipped and -0.0 is stored as the result.

G. Arguments whose values may be changed by the operation

The result of the operation is stored in C2 or YY, replacing their previous values.

8. LOGE

A. Purpose: To take the natural logarithm.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	LOGE	Constant or Array		May be used for a comment

C. Same as parts 7.C through 7.G of SQRT, replacing SQRT statements with LOGE, and noting the following additions to parts 7.E and 7.F: the operand should not contain a zero value. If it does, -0.0 is stored as the result.

9. EXP

A. Purpose: To raise e to a constant or an array power.

B. Instruction and permissible arguments

7. SQRT

A. Purpose: To take the square root of a constant or an array.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	SQRT	Constant or Array		May be used for a comment

C. Example: To take square root of C3.

,SQRT,C3

D. Sequential program action

- (1) The operand is checked for type.

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	EXP	Constant or Array		May be used for a comment

C. Example: Raise e to the array power YX.

,EXP,YX

D. Sequential program action

Same as part 7.D of SQRT.

E. Boundary conditions

Same as part 7.E.(1).

F. Default action and error diagnostics

Same as part 7.F.(1).

G. Arguments whose values may be changed by the operation

Same as part 7.G,

10. SPLOT

A. Purpose: To plot Y-arrays on a single frame of 35-mm film.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	SPLOT			May be used for a comment

C. Arguments that must be set beforehand

Name	Value	Explanation
KEY101	1., 2., 3., or 4.	1. = linear abscissa, linear ordinate. 2. = linear abscissa, log ordinate. 3. = log abscissa, linear ordinate. 4. = log abscissa, log ordinate.
LINE101	0., 1.	0. = no line 1. = points connected by line
TITLE		Graph title
COMMENT		Problem identification

D. Example:

,SPLOT

E. Sequential program action

(1) Subroutine GRAPH is called.

(2) If KEY101 = 2., points containing $YY \leq 0$. are deleted. If KEY101 = 3., points containing $YX \leq 0$. are deleted, and if KEY101 = 4., points containing $YX \leq 0$. and $YY \leq 0$. are deleted. In each case, the value of YF may be altered.

(3) The grid is internally scaled.

(4) The points in the Y-arrays are plotted over an index of YI,YF with abscissa YX and ordinate YY.

(5) TITLE is printed as the heading for the plot, and COMMENT is spaced one line below TITLE.

F. Boundary conditions

Y-arrays must contain at least two points after deletions; see 10.E.(2).

G. Default action and error diagnostics

(1) If Y-arrays contain only one legal point, plot is omitted. A diagnostic is given.

(2) If KEY101 is less than 1. or more than 4., the results may be unreliable.

H. Arguments whose values may be changed by the operation

YX, YY, and YF may be altered to remove zero and negative values. For logarithmic plots, the log of an array is loaded back into the array, thereby replacing the original contents.

11. PLOT

A. Purpose: To plot Y-arrays on an existing grid on 35-mm film.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PLOT			May be used for a comment

C. Action that should precede instruction

Set up a grid with SETGRIDY, SETGRIDXY or SPLOT.

D. Arguments that must be set beforehand
Same as SPLOT, part 10.C, except that TITLE and COMMENT should be blank if grid is set up with SPLOT.

E. Example:

.PLOT

F. Sequential program action, boundary conditions, and error diagnostics and actions

- (1) Same as SPLOT, part 10.E.(1), 10.E.(2), 10.E.(4), and 10.E.(5).
- (2) See SPLOT, parts 10.F through 10.H for additional information.

12. SETGRIDY

- A. Purpose: To set up a grid for plotting on 35-mm film. Internally scale the abscissa and externally scale the ordinate.
- B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	SETGRIDY			May be used for a comment

C. Arguments that must be set beforehand

Name	Value And/Or Explanation	
YLIMUP	Upper or maximum ordinate	
YLIMLR	Lower or minimum ordinate	
KEY101	1., 2., 3., or 4.	1. = linear abscissa, linear ordinate. 2. = linear abscissa, log ordinate. 3. = log abscissa, linear ordinate. 4. = log abscissa, log ordinate.
TITLE	Blank	
COMMENT	Blank	

D. Example:

.SETGRIDY

E. Sequential program action

- (1) Grid is plotted, type determined by KEY101.
- (2) Maximum and minimum ordinates are determined by YLIMUP and YLIMLR. Abscissa is internally scaled on YX(YI) and YX(YF).
- (3) Arguments listed in part 12.C are not altered by SETGRIDY.

F. Boundary conditions

YLIMUP and YLIMLR cannot be ≤ 0 . if KEY101 = 2. or 4.

G. Default action and error diagnostics

- (1) If KEY101 = 2. or 4. and YLIMUP or YLIMLR is ≤ 0 ., no grid is plotted and a diagnostic is given.
- (2) If KEY101 is less than 1. or more than 4., results may be unreliable.

13. SETGRIDXY

- A. Purpose: To set up a grid for plotting on 35-mm film. Externally scale both the abscissa and the ordinate.
- B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	SETGRIDXY			May be used for a comment

C. Arguments that must be set beforehand

Name	Value And/Or Explanation	
XLIMUP	Maximum abscissa	
XLIMLR	Minimum abscissa	
YLIMUP	Maximum ordinate	
YLIMLR	Minimum ordinate	
KEY101	1., 2., 3., or 4.	1. = linear abscissa, linear ordinate. 2. = linear abscissa, log ordinate. 3. = log abscissa, linear ordinate. 4. = log abscissa, log ordinate.
TITLE	Blank	
COMMENT	Blank	

D. Example:

,SETGRIDXY

E. Sequential program action

- (1) Grid is plotted, type determined by KEY101.
- (2) Maximum and minimum abscissas are determined from XLIMUP and XLIMLR. Maximum and minimum ordinates are determined from YLIMUP and YLIMLR.
- (3) Arguments listed in 13.C are not altered by SETGRIDXY.

F. Boundary conditions

- (1) YLIMUP and YLIMLR cannot be $\leq 0.$, if KEY101 = 2. or 4.
- (2) XLIMUP and XLIMLR cannot be $\leq 0.$, if KEY101 = 3. or 4.

G. Default action and error diagnostics

- (1) If KEY101 = 2. or 4. and YLIMUP or YLIMLR is $\leq 0.$, no grid is plotted and a diagnostic is given.
- (2) If KEY101 = 3. or 4. and XLIMUP or XLIMLR is $\leq 0.$, no grid is plotted and a diagnostic is given.
- (3) If KEY101 is less than 1. or more than 4., the result may be unreliable.

Appropriate diagnostics are printed.

- (3) Set PRTNO = 1. to obtain subroutine CHECKER diagnostics. PRTNO is reset to 0 after Y-arrays are checked.

D. Example:

,EMINMAX

E. Sequential program action

- (1) Y-arrays are checked by subroutine CHECKER. The arrays are reordered if necessary.
- (2) EMIN and EMAX are checked against the bounds of the YX array. EMIN or EMAX is reset if necessary, and a diagnostic is printed.
- (3) The YX array is searched over the index range YI-YF to locate the indices of the first occurrence of $YX \geq EMIN$ and $YX \leq EMAX$. The indices YI and YF are reset, and a report is printed.

F. Error diagnostics and default actions

- (1) If the Y-arrays do not satisfy subroutine CHECKER rules, EMINMAX is not executed.
- (2) In the table below, the two defaults, no execution and error return, cause diagnostic messages to be printed by the calling subroutine.

14. EMINMAX

- A. Purpose: To find indices of Y-arrays corresponding to EMIN and EMAX so that a discrete group of Y-array data may be chosen.
3. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	EMINMAX			May be used for a com- ment

C. Arguments that must be set beforehand

- (1) Load EMIN and EMAX with decimal numbers that are the desired upper and lower bounds, respectively, of the Y-array data.
- (2) If $EMIN < YX(YF)$, EMIN is reset to $YX(YF)$; and if $EMAX > YX(YI)$, EMAX is reset to $YX(YI)$.

Case	EMIN or EMAX Reset	Diag- nostic Message Printed	Default Action	YI and YF reset by EMINMAX
EMIN=EMAX=0.		Yes	No Exe- cution	
EMIN $\geq$ EMAX		Yes	No Exe- cution	
EMIN $>$ YX(YI)		Yes	No Exe- cution	
EMAX $<$ YX(YF)		Yes	No Exe- cution	
EMIN $<$ YX(YF)	Yes	Yes		Upon Execution
EMAX $>$ YX(YI)	Yes	Yes		Upon Execution
Only One YX(I) Found		Yes	Error Return	

- G. Arguments whose values may be changed
- (1) Subroutine CHECKER may alter Y-arrays and YF.
 - (2) EMIN, EMAX, YI, and YF may be changed during execution; see above.

15. POINTER

A. Purpose: To select a discrete set of Y-array data which have a YX range within EMIN-EMAX and a length $\leq$ KEY100. This set is moved to the top of the Y-arrays so that the resulting YI = 1.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	POINTER			May be used for a comment

C. Arguments that may be set beforehand

Name	Value	Explanation
EMIN	Decimal number	Lower boundary
EMAX	Decimal number	Upper boundary
KEY100	Decimal number	Desired number of points in selected interval. Required range: $1 < \text{KEY100} \leq 6000$.
PRTNO	1.	Subroutine CHECKER diagnostics are printed.

D. Example:

```

      .POINTER
  
```

E. Sequential program action

- (1) The Y-arrays are checked by subroutine CHECKER. The arrays are reordered if necessary.
- (2) Subroutine MINMAX is called to reset YI and YF for EMIN-EMAX range of YX-array. See 14.E.(2) and 14.E.(3) of EMINMAX module.
- (3) Entry is made to routine A345 where the length of the selected data set is made $\leq$ KEY100 by discarding points at either boundary alternately. The resulting data set is shifted to

the top of the Y-arrays so that YI becomes 1. and YF becomes the length of the set. Diagnostic messages are printed.

F. Error diagnostics and default actions

- (1) If the Y-arrays do not satisfy the rules of subroutine CHECKER, POINTER execution is terminated.
 - (2) If no execution or error return occurs in subroutine MINMAX, POINTER execution is terminated; see 14.F.(2) of the EMINMAX module.
 - (3) If $\text{KEY100} < 2$ or > 6000 , ENTRY A345 is skipped.
- G. Arguments whose values may be changed
- (1) Subroutine CHECKER may alter Y-arrays and YF.
 - (2) EMIN, EMAX, YI, and YF may be changed during execution of the module; see above.
 - (3) PRTNO is reset to 0.

16. READTAPE1 or READTAPE2

A. Purpose: To read data from tape or disk into the Y-arrays. The following discussion applies to either READTAPE1 or READTAPE2.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	READTAPE1			Data-block title. Title must begin with an * and contain no more than 39 additional FORTRAN alphanumeric or special characters. It must be identical to the data-block title written on the library data tape or disk.

C. Quantities that must be set beforehand

- (1) READTAPE1 means read logical unit TAPE1 which may be either a disk unit or tape unit. If a unit is not assigned to a magnetic tape by an ASSIGN control card, it is assumed to be a disk file. The unit must have been written in high-density (356 bits/in.) binary mode by a WRITETAPE1 instruction.

(2) Preceding the first READTAPE1 instruction, there must be a WRITETAPE1 instruction or the declaration of the last active data block title on the unit by a LASTTITLE1 instruction.

(3) Set PRTNO=1. If subroutine CHECKER diagnostics are desired.

D. Example: Read data block entitled *TEST DATA SET ONE from library tape TAPE1.

```
,READTAPE1,,, *TEST DATA SET ONE
```

E. Sequential program action

(1) The first character of the data block title is checked to determine if it is an *.

(2) For the first reference to the tape (by either a *WRITETAPE1 or READTAPE1 instruction), a table of contents of data-block titles is constructed by reading the titles from the tape and writing them on a separate disk unit. This operation continues until the title given with the LASTTITLE1 instruction has been recorded.

(3) The tape is then positioned to the requested data block, and the title of the data block on the tape is compared to the requested title.

(4) A check is made to ensure that the number of points in the data block is less than 6000.

(5) The data are read into YX and YY from YI = 1 through YF = number of points in the block.

(6) Subroutine CHECKER is called to check the Y-arrays.

F. Boundary conditions

Because the data written on the tape were checked beforehand by subroutine CHECKER, there should normally be no problems with the data read from the tape.

G. For error diagnostics and default actions other than those given or implied above, see Table C-III.

H. Arguments whose values may be changed

(1) YX, YY, and YF may be altered by subroutine CHECKER.

(2) If READTAPE1 aborts before subroutine CHECKER is called, PRTNO is not reset; otherwise, it is reset to 0 at the completion of subroutine

CHECKER.

17. WRITETAPE1 or WRITETAPE2

A. Purpose: To write Y-arrays on a disk file or a magnetic tape. The following discussion applies to either WRITETAPE1 or WRITETAPE2.

B. Instruction and permissible arguments

Field	Value	Explanation
1		
2	WRITETAPE1	
3	** or Blank	** indicates a unit that has never been written. A blank field indicates that the unit contains data and that the last data-block title has been declared.
4	1. or Blank	If field is 1. the table of contents of active titles is listed following the writing of the new data block.
5	Data Block Title	Title must begin with an * and may contain no more than 39 additional FORTRAN alphanumeric or special characters.

C. Quantities that must be set beforehand

(1) WRITETAPE1 means to write logical unit TAPE1. If a unit is not assigned to a magnetic tape by an ASSIGN control card, it is assumed to be a disk file. This disk is released at the completion of the job. A magnetic tape is written in odd parity with a density of 556 bits/in. (characters/in.) and is assumed to be 2200 ft long.

(2) Preceding the first WRITETAPE1, for an existing library tape, the title of the last good data block must be declared by a LASTTITLE1 instruction. Failure to do so will result in an error diagnostic.

(3) Set PRTNO=1. if subroutine CHECKER diagnostics are desired.

D. Example:

(1) Write a data block with the title *TEST DATA SET ONE on a new tape.

(2) List the table of contents of the tape following the write operation.

TABLE C-III
READTAPE1 OR READTAPE2 ERROR DIAGNOSTICS AND DEFAULT ACTIONS

NO.	Situation	Subroutine Where Situation Detected	Diagnostic	Default Actions			Remarks
				Module Execution Terminated	YX, YY, or YF Altered	Tape Read	
1	First character of title is not an *.	A326	REQUESTING TITLE INCORRECT	Yes	No	No	
2	TANDCRD locked out owing to previous attempt to read data block longer than LENGHTY from cards or tape.	TANDCRD	READCARDS AND READTAPE MODULE LOCKED OUT	Yes	No	No	TANDCRD locked out
3	Subroutine PLACE locked out owing to previous tape-positioning error.	PLACE	PLACE LOCKED OUT TO TAPENO=_____	Yes	No	No	PLACE locked out
4	End of file encountered (This occurs if LAST-TITLE1 has not been declared.)	PLACE	EOF READ ON TAPENO=_____	Yes	No	No	PLACE locked out during construction of table-of-contents file
5	Data-block title on tape does not have an * as first character.	PLACE	NO STAR ON TITLE OF TAPENO=_____	Yes	No	No	Same as 4
6	Data block on tape has no points.	PLACE	BLOCK LENGTH=0 ON TAPENO=_____	Yes	No	No	Same as 4
7	Requested title not found in table of contents.	PLACE	ERROR ON DATA TAPE=_____	Yes	No	No	
8	Title read from positioned tape is not the requested title.	PLACE	ERROR ON DATA TAPE=_____	Yes	No	No	
9	Any of 2-8 occurs.	A326	DATA READ FROM TAPE EITHER NOT THERE OR ILLEGAL	Yes	No	No	
10	Number of points in data block is greater than LENGHTY.	TANDCRD	NUMBER OF POINTS EXCEED LENGHTY-FOR A READ INSTRUCTION ONLY DATA TRUNCATED READCARDS AND READTAPE MODULE LOCKED OUT	Yes	No	No	TANDCRD locked out
11	Y-array data fail tests made in subroutine CHECKER	A326	DATA READ FROM TAPE EITHER NOT THERE OR ILLEGAL	Yes	Possibly	Yes	
12	Tape is correctly positioned and read.	A326	DATA UNDER TITLE READ AND CHECKED TO BE LEGAL	Yes	Possibly	Yes	

,WRITETAPE1,,1.,*TEST DATA SET ONE**

E. Sequential program action

- (1) The Y-arrays are checked by subroutine CHECKER.
- (2) The first character of the new data-block title is checked to determine if it is an *.
- (3) For the first WRITETAPE1 encountered in the program, unit TAPE1 is rewound in subroutine PLACE. If the tape has been declared to be a new one by ** in FIELD 3, the unit is considered to be in the correct position to be written. For the first reference to an old tape (by either a WRITETAPE1 or READTAPE1 instruction), a table of active data-block titles is constructed. Titles on the data tape are read and written on another disk until the title declared by the LASTTITLE1 instruction has been read and recorded on the table-of-contents disk. The tape is then positioned past the end of file of this data block.
- (4) For subsequent WRITETAPE1 instructions, the tape is positioned past the end of file for the last active data block before the new data block is written on the tape.
- (5) In all cases, the title of the new data block is checked against those in the table of contents to make sure that the new title is unique. If the new title is not unique, the tape is not written.
- (6) A calculation is made to determine if there is enough unused tape to hold the new data block.
- (7) The data-block title and the data block are written as one file.
 - a. The first logical record of the file contains the title of the data block and the number of YX, YY pairs (YF-YI+1) in the next logical record.
 - b. The second logical record contains the YX,YY data block.
 - c. The last logical record is an end-of-file mark.
- (8) The title for the data block is written on the table-of-contents file.
- (9) At the completion of the RAYMATH program, the disk containing the table of contents is rewound and printed.

F. Boundary conditions

- (1) For a normal write operation, the data must satisfy subroutine CHECKER.
- (2) The data blocks contain no more than 6000 YX,YY pairs.
- (3) In subroutine TWRITE a 2200-ft-long reel of tape is assumed, to prevent writing past the end-of-tape mark.
- (4) The title of the new data block must be different from all the other titles on the table-of-contents file.
- (5) When the unit is not assigned to a physical magnetic tape, it is still treated as if it were a 2200-ft-long tape.

G. Error diagnostics and default actions other than those given or implied above

- (1) See Table C-IV.
- (2) When one intends to use a magnetic tape but mistakenly leaves out the ASSIGN control card for that unit, excessive central processor time will probably be used in trying to set up the disk. Experience has shown that the computer job will probably exceed the time limit in this instance.

H. Arguments whose values may be changed

- (1) YX,YY, and YF may be altered by subroutine CHECKER.
- (2) If the report from the CHECKER subroutine is desired, set PRTNO = 1. At the completion of CHECKER, PRTNO is set to 0.

18. MPlot and MPlot2

A. Purpose: MPlot and MPlot2 plot the Y-arrays on frames of microfilm scaled so that they may be joined together to make a continuous scroll graph. MPlot2 has an option that the A-arrays may be included on these graphs. However, 12000 more central memory words are required for MPlot2 than for MPlot. When the ordinate scale is logarithmic, the first decade (values 0. to 10.) may be scaled and plotted on a linear grid that is part of the logarithmic grid. The following discussion applies to both MPlot and MPlot2 except that information pertinent to plotting the A-arrays must be disregarded when considering MPlot.

TABLE C-IV
WRITETAPE1 OR WRITETAPE2 ERROR DIAGNOSTICS AND DEFAULT ACTIONS

No.	Situation	Subroutine Where Situation Detected	Diagnostic	Default Actions			Remarks
				Module Execution Terminated	YX,YY or YF Altered	Tape Written	
1.	FIELD 3 not blank or **	A326	TAPE NOT WRITTEN BECAUSE NEW OR OLD NOT SET	Yes	No	No	
2.	First character of title is not an *.	A326	REQUESTING TITLE INCORRECT	Yes	No	No	
3.	Number of points exceeds LENGTHY.	A326	DATA ILLEGAL - NOT WRITTEN ON TAPE	Yes	No	No	
4.	Y-array data fail tests made in subroutine CHECKER	A326	DATA ILLEGAL - NOT WRITTEN ON TAPE	Yes	Possibly	No	
5.	No space for new data remaining on tape.	TWRITE	TAPE NO=_____ IS FULL	Yes	Possibly	No	
6.	Subroutine PLACE locked out owing to previous tape positioning error.	PLACE	PLACE LOCKED OUT TO TAPENO=_____	Yes	Possibly	No	PLACE locked out
7.	End of file encountered. (This occurs for an old tape if LASTTITLE1 has not been declared.)	PLACE	EOF READ ON TAPENO=_____	Yes	Possibly	No	PLACE locked out during construction of table-of-contents file.
8.	Data-block title on tape does not have an * as first character.	PLACE	NO STAR ON TITLE OF TAPENO=_____	Yes	Possibly	No	Same as 7
9.	Data block on tape has zero points.	PLACE	BLOCK LENGTH=0 ON TAPENO=_____	Yes	Possibly	No	Same as 7
10.	Attempt to write new data-block title which is the same as an existing active title.	PLACE	DATA BLOCK ALREADY ON TAPE	Yes	Possibly	No	
11.	Last title in table of contents does not agree with last active title on tape.	PLACE	LAST TITLE IN TOC AND ON DATA TAPE DONT AGREE	Yes	Possibly	No	
12.	For an old tape the number of titles in table of contents is zero.	TWRITE	FOR TAPE_____, NTITLE=0 OR DISK INCORRECTLY WRITTEN	Yes	Possibly	No	
13.	Report of amount of tape remaining before new block written	TWRITE	USEFUL TAPE LEFT ON TAPENO=_____ BEFORE NEW BLOCK IS WRITTEN IS=____FT.	No	Possibly	No	

TABLE C-IV (Continued)
WRITETAPE1 OR WRITETAPE2 ERROR DIAGNOSTICS AND DEFAULT ACTIONS

No.	Situation	Subroutine Where Situation Detected	Diagnostic	Default Actions			Remarks
				Module Execution Terminated	YX,YY or YF Altered	Tape Written	
14.	New data block is too long for remaining tape	TWRITE	NEW BLOCK WONT FIT ON TAPE_____	Yes	Possibly	No	
15.	Errors in called routine	A326	TAPE NOT WRITTEN	Yes	Possibly	No	
16.	Tape positioned and written successfully	A326	DATA PUT ON TAPE IS IN ACCORDANCE WITH CHECKER RULES	Yes	Possibly	Yes	

B. Instruction and permissible arguments

Field	Value	Explanation
1	MPLLOT2	
2		
3		
4	Blank or 2.	If equal 2., the A-arrays will be included on the plot, where applicable.
5		May be used for a comment.

C. Arguments that must be set beforehand

Name	Value	Explanation
KEY101	1.,2.,3., or 4.	1. = linear abscissa, linear ordinate. 2. = linear abscissa, log ordinate. 3. = log abscissa, linear ordinate. 4. = log abscissa, log ordinate.
C4	Decimal Number	Number of units (multiple of 5.) for linear abscissa. Number of decades per plot for log-arithmetic abscissa. If zero, only one plot is generated.
YLIMUP	Decimal Number	Used only if C4 is non-zero.
YLIMLR	Decimal Number	These arguments are the maximum (YLIMUP) and minimum (YLIMLR) ordinates for scaling. If both are zero, MPLLOT will utilize min and max of ordinate data for scaling. For a

Name	Value	Explanation
		log ordinate, MPLLOT replaces negative YLIMUP and/or YLIMLR with the max and/or min of the ordinate data, respectively.
LINE100	Decimal Number	If 1., first decade (0. to 10.) of log ordinate may be scaled linearly.
LINE101	Decimal Number	If nonzero, sequential points will be connected by lines; if 0., no lines.
PRINO	Decimal Number	If 1., subroutine CHECKER diagnostics are printed for both Y- and A-arrays. At the completion of the module, PRINO is set to zero.
TITLE	Alphanumeric and Special Characters	Graph title.
COMMENT	Alphanumeric and Special Characters	Problem identification

D. Examples: Plot Y- and A-arrays

,MPLLOT2,,2.

E. Sequential program action

- (1) The Y-arrays (A-arrays) are checked by subroutine CHECKER.
- (2) Subroutine REMOV is called to remove points with ordinates ≤ 0 when logarithmic plots are specified and to determine scale boundaries for the plots. For MPLLOT2 where both Y- and A-arrays

are to be plotted, subroutine REMOV will adjust graph boundaries to reflect the overall ranges of the Y- and A-arrays.

(3) When the two-part ordinate is desired for a logarithmic ordinate scale, two special cases may occur.

- a. If all points to be plotted are in the first decade (0 to 10 units), the ordinate will be linearly scaled only.
- b. If the number of linearly scaled points is zero, the ordinate will be plotted logarithmically only.

(4) Plots with no points are generated to cover gaps in the data.

F. Boundary conditions, error diagnostics, and default actions

(1) See Section E.(1).

(2) If subroutine REMOV determines that no points are available for plotting an array, a message is printed and plotting is deleted for that array.

(3) If more than 25 graphs are generated, a return is made and plotting is terminated.

(4) Little, if any, checking is done by the module on the arguments listed in Section C. Thus, zero them or load them with intended values preceding each use of the module. Otherwise, uncertain results may occur.

G. Arguments whose values may be changed

(1) For a plot of both Y- and A-arrays, YX,YY,YF,AX,AY, or AF may be altered.

(2) For a plot of only the Y-arrays, YX,YY, or YF may be altered.

(3) YLIMUP and/or YLIMLR will be altered for a log-ordinate plot if either one is <0.

Use of CALCOMP may save 12000₁₀ central memory words. The following discussion applies to both modules, except that information pertinent to plotting the A-arrays must be disregarded when considering CALCOMP.

B. Instruction and permissible arguments

Field	Value	Explanation
1	CALCOM2	
2		
3		
4	Blank or 2.	If equal 2., the A-arrays will be included on the plot, where applicable.
5		May be used for a comment.

C. Arguments that must be set beforehand

Name	Value	Explanation
KEY101	1.,2.,3., or 4.	1. = linear abscissa, linear ordinate. 2. = linear abscissa, log ordinate. 3. = log abscissa, linear ordinate. 4. = log abscissa, log ordinate.
C4	Decimal Number	Number of inches per five units for linear abscissa. Number of inches per decade for logarithmic abscissa. If zero, 10-x 17-in. standard-size plot is produced.
YLIMUP	Decimal Number	Used only if C4 is nonzero.
YLIMLR	Decimal Number	Maximum (YLIMUP) and minimum (YLIMLR) ordinates for scaling. If either or both is less than or equal to zero, CALCOM2 will utilize max and min of ordinate data for scaling.
C1	Decimal Number	If -1., grid overlay is drawn.
LINE100	Decimal Number	If 1., first decade (0 to 10 units) of log ordinate may be scaled linearly. If -1., first decade (0 to 10 units) of log ordinate may be scaled linearly and negative data are plotted on a 0 to -10 unit linear scale. Note: The above two options are reflected on the plot only if there are data in the range specified.

19. CALCOMP and CALCOM2

A. Purpose

(1) CALCOMP and CALCOM2 cause the Y-arrays to be plotted on a single paper graph by a CALCOMP plotting device. If the ordinate scale is logarithmic, the first decade (0 to 10 units) may be scaled and plotted on a linear grid.

(2) CALCOM2 has two plotting options. When FIELD 4 = 0. or blank only the Y-arrays are plotted, YY versus YX. Setting FIELD 4 = 2. causes both the Y- and A-arrays to be plotted. CALCOMP has only one option: plot Y-arrays.

Name	Value	Explanation
LINE101	Decimal Number	If -2., sequential points will be connected by lines; if 0., no lines.
PRINO	Decimal Number	If 1., subroutine CHECKER diagnostics are printed for both Y- and A-arrays. At the completion of CALCOM2, PRINO is set to zero.
TITLE	Alpha-numeric and Special Characters	Graph title.
COMMENT	Alpha-numeric and Special Characters	Problem identification.

Copy TAPE7 to a nonprotected, unlabeled, high-density (556 BPI) tape written in a binary mode. This copy operation should be done at the end of the monitor control card deck. The tape is used off-line to produce the desired plots.

D. Example: Plot Y- and A-arrays.

,CALCOM2,,2.

E. Sequential program action

(1) The Y-arrays (A-arrays) are checked by subroutine CHECKER. For plots with conventional logarithmic ordinates, subroutine REMOOV alters Y-arrays (A-arrays) by removing points where $YY(I) \leq 0$ ($AY(I) \leq 0$). These graphs may lead to replacement of the Y-arrays (A-arrays) with values other than the array contents; thus, the original contents are lost.

(2) If FIELD 4 is 2., CALCOM2 will include the A-arrays on all plots, where applicable.

(3) For logarithmic ordinates, if LINE100 = ± 1 ., CALCOM2 plots the first decade (0 to 10 units) of the data on a linear grid. If all data are within this range, the ordinate is linear only; if the data are beyond this range, the ordinate is logarithmic only.

(4) Messages about array sizes, number of plotting points, etc., are printed.

F. Boundary conditions

(1) See 19.E(1).

(2) If subroutine REMOOV determines that no points are available for the plotting of an

array, a message is printed, and plotting is deleted for that array.

(3) If FIELD 4 = 2., subroutine REMOOV adjusts the abscissa and ordinate scaling to reflect the overall ranges of the Y- and A-arrays, if applicable.

(4) The variable C4 is used to control the physical length of the paper plot. Therefore, considerable care must be taken to utilize this feature properly. A poorly chosen value could yield an unnecessarily long or short plot. An upper limit of 13 ft (156 in.) is automatically selected if a longer plot has inadvertently been designated. If $C4 = 0$., a standard length of 17 in. is used. If $C4 \neq 0$., the plotting limits for the ordinates (YLIMUP, YLIMLR) may be specified.

G. Error diagnostics and default actions other than those given above

(1) Y-array (A-array) data must pass subroutine CHECKER specifications for CALCOM2 to be executed.

(2) CALCOM2 does little, if any, checking on the arguments listed in Section C. Therefore, zero them or load them with intended values before each use of CALCOM2. Otherwise, uncertain results may occur.

H. Arguments whose values may be changed

(1) For a plot of both Y- and A-arrays, YX, YY, YF, AX, AY, or AF may be altered.

(2) For a plot of only the Y-arrays, YX, YY, or YF may be altered.

20. DUMP

A. Purpose: To dump nonarray common storage.

B. Instruction and permissible arguments

FIELD 1	FIELD	FIELD 3	FIELD 4	FIELD 5
	AF			May be used for a comment

C. Example

,DUMP

D. Sequential program action

- (1) DUMPLT is set to 1, and subroutine A308 is called.
- (2) All nonarray common is printed.
- (3) DUMPLT is reset to its original value.
- (4) Program execution continues.

21. PUNCH

- A. Purpose: To punch Y-arrays on cards.
- B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PUNCH			May be used for a comment

C. Quantities that must be set beforehand

- (1) The punch output is written on disk TAPE12, so this unit must be copied to the PUNCH file at the conclusion of the job by the control cards REWIND(TAPE12) and COPY(TAPE12,PUNCH).
- (2) To inhibit output other than the PUNCH output from being written on TAPE12, the RAYMATH program card input format declaration must not be OPENTELE.
- (3) To identify output from successive PUNCH statements, load COMMENT with the appropriate identification for each. However, if only numeric output is desired, load COMMENT with blanks (equivalent to -0.'s).
- (4) Set PRTNO equal to 1. if diagnostics from subroutine CHECKER are desired.

D. Example

,PUNCH

E. Sequential program action

- (1) Y-array points are checked for legality and order by subroutine CHECKER and reordered if necessary.
- (2) Identification COMMENT is punched on first card.
- (3) Y-array points are punched over an index range of YI,YF by a step of 1 in a format 6E12.5.

F. Boundary conditions: Y-array must contain at least two legal points.

G. Default action and error diagnostics

- (1) If OPENTELE is declared on first program card and TAPE12 is copied to the OUTPUT file, no cards are punched but a listing is obtained on TAPE12. The Y-array points are printed over an index range YI,YF by an interval PRTNO.
- (2) If Y-array does not contain at least two legal points, the execution of PUNCH is not completed and a diagnostic is given. Program execution continues.

H. Arguments whose values may be changed

- (1) YX,YY, and YF all may be changed by the CHECKER subroutine.
- (2) PRTNO is set to zero at the completion of the CHECKER subroutine.

22. INTEGRATE

A. Purpose: To integrate YY versus YX from an upper boundary of YX(YI) to a lower boundary of YX(YF).

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	INTEGRATE			May be used for a comment

C. Arguments that may be set beforehand: Set PRTNO to 1. to obtain a report of Y-array modifications made by subroutine CHECKER and the NOEQUALX module. PRTNO is reset to 0. at end of module.

D. Example

,INTEGRATE

E. Sequential program action

- (1) Subroutine CHECKER and A338 (NOEQUALX module) are called to legalize data and to combine points having the same YX-values.
- (2) The integration is carried out over the range from YI to YF for all values of YY between these boundaries. The final result is stored in C2.

F. Boundary conditions and error diagnostics

- (1) Data must satisfy restrictions of sub-routines CHECKER and A338 (NOEQUALX module).
- (2) If A338 combines points so that there is only one point remaining, INTEGRATE is not executed and diagnostic messages are written on the output files.

G. Arguments whose values may be changed:
YX,YY,YF, and C2.

23. LASTTITLE1 or LASTTITLE2

A. Purpose: To identify last title on library TAPE1 or TAPE2.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	LASTTITLE1			* TAPE TITLE

The tape title must begin with * and contain no more than 39 characters exclusive of *.

C. Example

```
,LASTTITLE1,,, *DATA FOR RAYMATH PROGRAM
```

D. Remarks: For a library tape containing data blocks, it is necessary to declare the last "good" title on the tape with a LASTTITLE1 instruction before first reading or writing this tape. If this is not done, switches are set in the positioning subroutine PLACE which prevent any data transfer to or from the tape during the course of the program.

24. LOADYLIMUP

A. Purpose: To load largest element of YY-array into YLIMUP.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	LOADYLIMUP			May be used for a comment

C. Example

```
,LOADYLIMUP
```

D. Sequential program action

- (1) The largest element of the YY-array is loaded into YLIMUP.
- (2) The operation does not alter the YY-array.

25. LOADYLIMLR

A. Purpose: To load smallest element of YY-array into YLIMLR.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	LOADYLIMLR			May be used for a comment

C. Example

```
,LOADYLIMLR
```

D. Sequential program action

- (1) The smallest element of the YY-array is loaded into YLIMLR.
- (2) The operation does not alter the YY-array.

26. AVOVERTIME

A. Purpose: To average neutron time-of-flight data over a differential time interval.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	AVOVERTIME			May be used for a comment

C. Arguments that must be set beforehand

Name	Explanation
YI,YF	

Name	Explanation
YX	Neutron energy in electron volts
YY	Dependent variable
C1	Neutron flight path in meters
C3	Differential time interval in micro-seconds
PRTNO	May be set to 1. to obtain diagnostics from subroutine CHECKER

D. Example

,AVOVERTIME

E. Sequential program action

- (1) Y-arrays are checked by subroutine CHECKER. At conclusion of check, PRTNO is reset to 0.
- (2) YX is converted to time, t, in microseconds by the formula

$$t = \frac{72.3 \times C1}{\sqrt{YX}}$$

- (3) Upper and lower integral time boundaries, separated by an interval C3, are generated.
- (4) The number of points having a time within the boundaries is determined, and the arithmetic average of the corresponding YY's is calculated. The average time for points within the interval is determined.
- (5) The energy corresponding to this average time is calculated and then loaded into AX. The average ordinate is loaded into AY.
- (6) The time averaging is done for all Y-array points between indices YI and YF.
- (7) As indicated above, averaged results are stored in the A-arrays between the indices AI=1 and AF. Initial and final AI and AF are printed at the completion of the module.

F. Boundary conditions

- (1) Y-array data must satisfy subroutine CHECKER rules.
- (2) C1 and C3 must be applicable to the particular set of Y-array data. Otherwise, excessive central processor time may be consumed.

G. Arguments whose values may be changed

- (1) Subroutine CHECKER may alter YX, YY, and YF.
- (2) The A-arrays, AI, and AF may be changed.

27. YROLLEDTOA

A. Purpose: To load A-arrays with Y-arrays.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	YROLLEDTOA			May be used for a comment

C. Quantities that may be set beforehand:

NSTEP may be set to the desired transfer interval.

D. Example

,YROLLEDTOA

E. Sequential program action

- (1) YX is loaded into AX, YY is loaded into AY, and indices YI and YF are loaded into AI and AF, respectively.
- (2) Array elements are transferred over an index range NI,NF by a step of NSTEP. Usually NI=1, NF=6000, and NSTEP=1.
- (3) The instruction does not alter YI,YF, or the Y-arrays.

F. Arguments whose values may be altered

- (1) A-arrays, AI, and AF.
- (2) If NSTEP was set beforehand, it should be reset to 1 immediately following YROLLEDTOA.

28. AROLLEDTOT

A. Purpose: To load Y-arrays with A-arrays.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	AROLLEDTOT			May be used for a comment

C. Quantities that may be set beforehand:

NSTEP may be set to the desired transfer interval.

D. Example

,AROLLEDTOT

E. Sequential program action

(1) AX is loaded into YX, AY is loaded into YY, and indices AI and AF are loaded into YI and YF, respectively.

(2) Array elements are transferred over an index range NI,NF by a step of NSTEP. Usually NI=1, NF=6000, and NSTEP=1.

(3) The instruction does not alter AI, AF, or the A-arrays.

F. Arguments whose values may be changed

(1) Y-arrays, YI, and YF.

(2) If NSTEP was set beforehand, it should be reset to 1 immediately following AROLLEDTOT.

YX to the smallest. The running sum at YX(YI) is defined to be zero. If the running sum exceeds C2, it is decreased by subtracting the minimum integral multiple of C2 required to keep the running sum $\leq C2$. The cumulative decrement is written on the output file to indicate the change in the running sum.

(3) The prescription for finding an integral using the running sums at the boundaries of the integral is

$$\text{INTEGRAL} = \text{RS}(\text{YX}_{\text{small}}) - \text{RS}(\text{YX}_{\text{large}})$$

$$+ \text{RSD}(\text{YX}_{\text{small}}) - \text{RSD}(\text{YX}_{\text{large}}),$$

where RS is the running sum at YX, and RSD is either the running-sum decrement at YX or the nearest one preceding YX. Only in exceptional cases where $C2 \ll \text{RS}$ does $\text{RSD}(\text{YX}_{\text{small}}) - \text{RSD}(\text{YX}_{\text{large}}) \neq 0$. Thus, one may ordinarily neglect RSD.

F. Boundary conditions and error diagnostics

(1) A minimum of two points having different YX > 0 are required. If such is not the case, RUNSUM terminates with no results.

(2) Data must satisfy restrictions of subroutines CHECKER and A338 (NOEQUALX module).

(3) If the difference between the running sum and the product $C2 \times 999$ is greater than C2, the module terminates with the diagnostic:

INFINITE LOOP ENCOUNTERED IN RUNSUM MODULE.
EXECUTION TERMINATED.

(4) integration extends over negative, as well as positive, YY.

C. Arguments whose values may be changed:

YX, YY, YF, and C2. If PRIMO is set to 1., it will be returned to 0. at exit from the module.

29. RUNSUM

A. Purpose: To determine by the trapezoidal rule the cumulative integral or running sum of YY versus YX. The point-by-point listing of this integral may then be used in conjunction with a desk calculator to find integrals over selected ranges of YX.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	RUNSUM			May be used for a comment

C. Arguments that may be set beforehand

(1) Set C2 to maximum desired value of the running sum. If $C2=0$ or $C2 > 10^6$, C2 is set to 10^6 to prevent overflow of the F12.5 output format of the running sum.

(2) Set PRIMO to 1. to obtain a report of Y-array modification made by subroutine CHECKER and A338.

D. Example

,RUNSUM

E. Sequential program action.

(1) Subroutines CHECKER and A338 (NOEQUALX module) are called to legalize data and to combine points having the same YX-values.

(2) The running sums are calculated and written along with the YX,YY for that point on the output file. The integration is from the largest

30. RESINTEGR1

A. Purpose: To calculate a partial resonance integral

$$\int_{E_1}^{E_2} \frac{\sigma_{n,\gamma} dE}{E},$$

where $\sigma_{n,\gamma}$ is the neutron capture cross section at energy F, and E_1 and E_2 are the bounds of the partial resonance integral.

8. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	RESINTEGRL			May be used for a comment

C. Arguments that must be set beforehand

- (1) Indices of the Y-arrays, YI and YF.
- (2) Cross section in units of barns stored in YY, and neutron energy in eV stored in YX.
- (3) PRINO=1. for diagnostics from subroutine CHECKER and NOEQUALX module.

D. Example

,RESINTEGRL

E. Boundary conditions and sequential program action

- (1) Y-arrays checked for legality and order by subroutine CHECKER. This is followed by a pass through the NOEQUALX module (subroutine A338) to combine points having same YX value.
- (2) Integration is by a modified trapezoidal rule with negative YY treated as YY=0. At least two points are required. Integration extends from point with index YI to point having index YF. Integration is from large YX to small YX; i.e., large to small energy.
- (3) The final result is loaded into C2 and has units of barn-eV. (Note: 1 barn = 10^{-24} cm².)

F. Default action and error diagnostics

- (1) If data fail to pass rules of subroutine CHECKER, RESINTEGRL is not executed.

G. Arguments whose values may be changed by the operation

- (1) CHECKER subroutine may alter YX, YY, and YF.
- (2) NOEQUALX module may alter YX, YY, and YF.
- (3) Results are stored in C2.

31. NOEQUALX

A. Purpose: To make Y-arrays single-valued with respect to YX.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	NOEQUALX			May be used for a comment

C. Arguments that may be set beforehand

Set PRINO=1. to obtain a report of Y-array modifications made by subroutine CHECKER and NOEQUALX module.

D. Example

,NOEQUALX

E. Sequential program action and diagnostics

- (1) Legality of Y-array data is checked by subroutine CHECKER. If data cannot be legalized, the module is skipped.
- (2) Y-arrays are searched for points having equal YX. The range of action of the module is from YI through YF.
- (3) When all points with the same YX are found, the YY values are averaged, and this average YY is stored in the first YY value. The remaining points with equal YX are eliminated by sliding the Y-arrays over them. The search for the next set of points having equal YX is resumed.
- (4) A report of combined points is printed if PRINO=1.
- (5) At the end of the module, initial and final YI and YF are printed. If NOEQUALX has combined all points into one, that is, if YI=YF, a diagnostic is printed.

F. Data-boundary conditions and arguments whose values may be changed: Y-array data must satisfy restrictions of subroutine CHECKER. YX, YY, and YF may be changed by the module.

32. PRINTPLOT

A. Purpose: To plot a one-page linear-linear graph of Y-array data on the line printed and/or remote-keyboard terminal.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	PRINTPLOT			May be used for a comment

C. Arguments that must be set beforehand

Name	Value	Explanation
LINE101	0. or 1.	0. = points not connected by line. 1. = points connected by line.
C4	0., 1., 2.	0. = no grid. 1. = + printed at grid-line intersections. 2. = full grid is printed.

Note: The POINTER module may be employed preceding PRINTPLOT to restrict the Y-arrays to less than or equal to 1000 points.

D. Example

.PRINTPLOT

E. Sequential program action

(1) A linear-linear plot of YY vs YX from YI through YF is printed, if $(YF - YI + 1) \leq 1000$. The plot printed on the remote-keyboard terminal is less than 80 columns wide.

(2) Execution of the module is skipped if the number of points exceeds 1000.

F. Boundary conditions: Uncertain results may be obtained if LINE101 or C4 is given values other than those above.

33. MELDYANDA

A. Purpose: To merge the contents of the A-arrays between AI and AF with the contents of the Y-arrays between YI and YF.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	MELDYANDA			May be used for a comment

C. Example

.MELDYANDA

D. Sequential program action

(1) The length of the combined arrays is calculated.

(2) If this length is less than or equal to 6000, the contents of the Y-arrays between YI and YF are shifted to the top of the Y-arrays.

(3) The A-array contents between AI and AF are loaded into the Y-arrays beginning at an index address of YF+1. AX is loaded into YX, and AY is loaded into YY.

(4) At the completion of the merging operation, a message is printed giving the final YI and YF.

E. Error diagnostic and default action: If the length of the combined arrays exceeds 6000, the shift and merge operations are skipped and a message is printed.

F. Arguments whose values may be changed by module execution

(1) YX, YY, YI, and YF.

(2) After module execution, the Y-arrays may not be in legal order.

34. DELETEx=

A. Purpose: To delete a specified point in the Y-arrays.

B. Instruction and permissible arguments

Field	Value	Explanation
1	DELETEx= Decimal Number	YX of the point to be deleted. Enter number in an E12.5 format. One may enter the number as it appears in a PRINT module listing because roundoff in the last decimal place is taken into account by the delete routine.
2		
3		
4		May be used for a comment.
5		

C. Module that may be executed beforehand: To remove all points with the same YX, precede module by NOEQUALX. Otherwise, only the first occurrence of the point will be deleted.

D. Example: Delete point $Y\bar{X} = .31983E+04$,
 $Y\bar{Y} = -.16004E-01$.

`,DELETEx=,,0.31983E+04,`

E. Sequential program action

(1) The Y-arrays are searched from $Y\bar{I}$ to the first occurrence of the point to be deleted. The point is set to 0, $Y\bar{X}=Y\bar{Y}=0$. A message is printed indicating that the point has been deleted. If desired point is not found in the Y-arrays, the search terminates at $Y\bar{F}$ and a diagnostic message is printed.

(2) Use LEGALORDER instruction after the last DELETEx= operation to purge the deleted points from the Y-arrays.

35. CLOCK

A. Purpose: To obtain the elapsed central processor time during program execution.

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	CLOCK			May be used for a comment

C. Example

`,CLOCK`

D. Sequential program action

Elapsed central-processor time is printed out in seconds to the nearest thousandth of a second.

36. SPARE MODULE NAMES

A. Purpose: To facilitate addition of new modules to the system.

B. Available names for new modules: A344, A870, A872, A874, A876, A888, A900, and A902.

C. Method: Insert, by using appropriate UPDATE techniques,²⁵ the FORTRAN program for the new module into one of the existing sub-routines that has the same name as the module.

To change the module name, modify the logical IF statement in subroutine A321 and include array-storage requirements. Note that Y-arrays are always declared in FORPROG.

37. EXCHANGE

A. Purpose: To interchange arguments 1 and 2.

B. Instruction and permissible arguments

Field	Value	Explanation
1	Array, Constant, or DOLOOPMT	Argument 1
2	EXCHANGE	Argument 2
3	Array, Constant, or DOLOOPMT	
4		
5		May be used for a comment

C. Quantities that may be set beforehand

(1) DUMPLT may be set ≤ 0 . to terminate program if an error is encountered during execution.

(2) NSTEP may be set to the desired exchange interval.

D. Example: Interchange arrays $Y\bar{X}$ and $Y\bar{Y}$.

`YX,EXCHANGE,YY`

E. Sequential program action

(1) Arguments 1 and 2 are checked for type.

(2) If they are the same type, the arguments are exchanged. Array indices $Y\bar{I}$, $Y\bar{F}$, $A\bar{I}$, etc. are not altered.

(3) For arrays the operation is performed over an index range $N\bar{I}$, $N\bar{F}$ by an interval of $N\bar{S}\bar{T}\bar{E}\bar{P}$. Usually $N\bar{I}=1$, $N\bar{F}=6000$, and $N\bar{S}\bar{T}\bar{E}\bar{P}=1$.

F. Default action and error diagnostics

(1) If arguments 1 and 2 are not the same type, or if either argument is illegal, EXCHANGE is not executed and a dump is obtained.

(2) If $DUMPLT \leq 0$. the program is terminated after the dump; otherwise, program execution continues.

- G. Arguments whose values may be changed
- (1) Argument 1 is loaded with the original value of argument 2, and argument 2 is loaded with the original value of argument 1.
 - (2) If NSTEP was set beforehand, reset it to 1. immediately after EXCHANGE.

38. TO

- A. Purpose: To load argument 2 with contents of argument 1.
- B. Instruction and permissible arguments

Field	Value	Explanation
1	Array, Constant, or DOLOOPMT	Argument 1
2	TO	
3	Array, Constant, or DOLOOPMT	Argument 2
4		
5		May be used for a comment

- C. Quantities that may be set beforehand
- (1) DUMPLT may be set ≤ 0 . to end program if an error is encountered during execution.
 - (2) NSTEP may be set to the desired loading interval. If reset, it should be returned to 1. following TO.
- D. Example: Load array YX into array AX.

YX,TO,AX

- E. Sequential program action
- (1) Arguments 1 and 2 are checked for type.
 - (2) If they are the same type, argument 1 contents are loaded into argument 2. This has no effect on argument 1.
 - (3) For arrays, the operation is performed over an index range NI,NF by an interval NSTEP.
- F. Default action and error diagnostics
- (1) If arguments 1 and 2 are not the same type, or if either argument is illegal, the module is not executed and a dump is obtained.
 - (2) If DUMPLT ≤ 0 . the program is terminated after an error; otherwise, program execution continues.

- G. Arguments whose values may be changed by the operation:
- Argument 2 is loaded with the value of argument 1. Argument 1 remains unchanged.

39. LEGALORDER

- A. Purpose: To check Y-array points against a set of specifications, identify and discard any that do not agree, and order, recheck, and count those remaining.
- B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	LEGALORDER			May be used for a comment

- C. Arguments that may be set beforehand: If a detailed report of misordered points and points that have $YX \leq 0$. is desired, set PRTNO to 1..
- D. Example

,LEGALORDER

- E. Sequential program action
- (1) Each point between YI and YF is checked in subroutine CHECKER for the following:
 - a. YX must be in descending order.
 - b. YX must be greater than zero except for points at the end of the array.
 - c. There must be at least two points with YX greater than zero. Otherwise YF is not reset.
 - (2) Points having $YX > 0$. are counted.
 - (3) If conditions (1) a and b are not met, $YX \leq 0$ points are discarded, and the Y-arrays are reordered and then rechecked and counted.
 - (4) Appropriate diagnostic messages are printed, including the number of good points, YI, and the initial and final values of YF.
- F. Error diagnostics and default actions
- (1) See above.
 - (2) If more than two reorderings are required, a message is printed, and a return is made without resetting YF.
- G. Arguments whose values may be changed
- (1) YX, YY, and YF.

(2) PRTNO is reset to 0. before any return is made from the subroutine.

40. INTERPLATE

A. Purpose: To linearly interpolate Y-arrays at AX values that are within a range YX(YI) through YX(YF).

B. Instruction and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	INTERPLATE			May be used for a comment

C. Arguments that must be set beforehand

- (1) Load A-arrays with standard table.
- (2) Load Y-arrays with table to be interpolated.
- (3) PRTNO may be set to 1. to obtain subroutine CHECKER diagnostics for both Y- and A-arrays.

D. Example

,INTERPLATE

E. Sequential program action

- (1) Legality of data in Y- and A-arrays is checked by subroutine CHECKER.
- (2) The boundaries of the AX-array are checked. Indices AI and AF are altered, if necessary, so that the conditions $AX(AI) \leq YX(YI)$ and $AX(AF) \geq YX(YF)$ are satisfied.
- (3) Linear interpolation is done over the range YX(YI) through YX(YF), and the results of the interpolation are loaded into the BY-array.
- (4) BI and BF are reset to AI and AF, respectively.
- (5) Except for changes in the Y- and A-arrays made by subroutine CHECKER and possible modification of A-array indices as described above, the interpolation operation does not affect either the Y- or A-arrays.

F. Default actions and error diagnostics

- (1) If Y- or A-array data fail to pass subroutine CHECKER requirements, execution is terminated before A-array boundaries are checked. A diagnostic is printed.

(2) If AI becomes greater than AF, or if AF becomes less than AI during alteration of A-array indices, execution is terminated without the interpolation being done. A diagnostic message is printed.

(3) If only one point was obtained by the interpolation, BI=BF, a diagnostic is printed.

G. Arguments whose values may be changed

- (1) The CHECKER subroutine may alter the Y-arrays or the A-arrays.
- (2) AI and AF may be altered by the boundary check given above.
- (3) BY, BI, and BF are altered by the results of the interpolation.
- (4) If PRTNO has been set to 1., it will be returned to 0 at the exit from the module.

41. END, HEREIS, RETURN, GOTO

A. Purpose: These are program control instructions that control program assembly and translation.

(1) The main program consists of those modular instructions between the format declaration (on first line) and END. All subprograms follow END.

(2) Subprograms begin with a HEREIS and end with a RETURN. The label or name of the subprogram is written in FIELDS 4 and 5. No two subprograms may have the same name. The label may consist of a number in FIELD 4 and/or a 40-character name in FIELD 5.

(3) Subprograms are called with a GOTO. GOTO may appear in the main program or in any subprogram. Ten successive calling levels are allowed; i.e., one program may call another which calls another, which, in turn, calls another, and so on. Circular calling schemes are not allowed.

(4) The name of the desired subprogram must appear in FIELDS 4 and 5 of the GOTO exactly as it appears in FIELDS 4 and 5 of the HEREIS of the subroutine.

B. Instructions and permissible arguments

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	END			May be used for a comment

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	HEREIS		Blank or Decimal Number	Subroutine Name

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	RETURN			May be used for a comment

FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5
	GOTO		Blank or Decimal Number	Subroutine Name

C. Example program: Print a constant C2=5. in a subprogram, return to the main program, read C3=6., and print C3. Print program and its execution on the line printer and on the remote-keyboard terminal.

```
OPENTELE
,READ,C4,5.
,GOTO,,5.,TEST ROUTINE
,READ,C3,6.
,PRINT,C3
,END
,HEREIS,,5.,TEST ROUTINE
,PRINT,C4
,RETURN
```

D. Sequential program action

When the RAYMATH program is assembled, a search is made for subroutines called with a GOTO. These subroutines are inserted in place of the respective GOTO in the program, and the GOTO, HEREIS, and RETURN instruction lines are omitted. END is also omitted from the assembled program. The assembled program is printed and then executed.

E. Error diagnostics and default actions

Error	Type of Error	Default Action In Addition to a Printed Diagnostic
Missing END Statement	Fatal	Run terminated
Missing subroutine called by a GOTO	Fatal	Run terminated
Missing RETURN Statement	See Default Action	If RETURN is missing from the last subroutine in the program deck, the run is terminated. Otherwise, the RETURN in the following subroutine is used thereby adding further, and probably undesirable, statements to the program.
More than 10 sequential subroutine calls	Fatal	Run terminated
Total length of object program deck, including subroutines and declarations, exceeds 2009 statements	Fatal	Run terminated

APPENDIX D. SOURCE PROGRAM LISTING

This appendix contains the RAYMATH source program listing. The listing augments the Module Chart descriptions for the modular instructions and the Sec. III description of the translator. When adding a new instruction to the RAYMATH repertoire, careful study of the listings of selected routines is usually necessary. The printer plotting routine, PRTPLT, obtained from J. C. Solem (LASL Group TD-1), and the assembly language routines for the CALCOMP plotter, obtained from the LASL program library, are included in the listing to make it complete.

Table D-I is a glossary characterizing each routine of the RAYMATH code. If a routine is the principal routine of a subroutine package, the module or instruction name is listed. Tabulation of array dependence helps the user determine the amount of array storage needed by a RAYMATH program. The PURPOSE column serves to remind the reader of the functions of most routines. Routines are tabulated in the order of their appearance in the source program listing. The coding language of all routines,

except a few written in assembly language for the CALCOMP plotter, is FORTRAN IV. Names of routines constructed by the translator are included for completeness.

The line-editing code, UPDATE,²⁵ maintains the source program. The length of the source deck (approximately 8200 cards) makes machine editing more efficient than hand editing. Fig. D-1 shows the UPDATE common decks; the source program listing in-

cludes these decks inserted in the code in the appropriate places. Changing the number 6000 in the common decks to the desired value, updating the old library, and recompiling the new source deck alters the array dimensions in the entire code. The SPARE common deck is for future additions to the COMMON declarations. Table D-II is the common deck use index which gives the common decks appear- in each routine.

TABLE D-I
GLOSSARY OF ROUTINES

Routine	Module Name	Sub-routine	Entry	Entry In Subroutine	Array Dependence Of Dimension Declarations In The Routine	Purpose
TELERAY					None	Translator
FREEB		X			None	Interprets free-field format
A321		X			None	Checks table of permitted instructions
A212		X			None	Argument classification and numbering
A216		X			None	Writes array loader, unloader, and initializer
A319		X			None	Loads parts of COMMON
A318			X	A319	None	Lists tables of contents of tapes
A215		X			None	COMMON initialization
A202		X			None	CONSTANT loader
A203			X	A202	None	CONSTANT unloader
A204			X	A202	None	DOLOOPMT loader
A205			X	A202	None	DOLOOPMT unloader
A206			X	A202	None	PARAMETER loader
A207			X	A202	None	PARAMETER unloader
A208			X	A202	None	NUMBER loader
A209			X	A202	None	NUMBER unloader
A210			X	A202	None	AFORMAT loader
A211			X	A202	None	AFORMAT unloader
A214			X	A202	None	BLANK loader, unloader
A213			X	A202	None	STAR loader, unloader
A221		X			YX	YX array loader
A222		X			YX YY	YY array loader
A223		X			YX YY AX	AX array loader
A224		X			YX YY AX AY	AY array loader
A225		X			YX YY AX AY BY	BY array loader
A226		X			YX YY AX AY BY CY	CY array loader
A231		X			YX	YX array unloader
A232		X			YX YY	YY array unloader
A233		X			YX YY AX	AX array unloader
A234		X			YX YY AX AY	AY array unloader
A235		X			YX YY AX AY BY	BY array unloader
A236		X			YX YY AX AY BY CY	CY array unloader
A315	READ	X			(YX YY)	Reads cards, CONSTANTS, and AFORMATS
A316	PRINT	X			YX YY	Prints Y-arrays, CONSTANTS, and AFORMATS
A320	PRALLARRAY	X			YX YY AX AY BY CY	Prints all arrays
A302	+	X			YX YY	Adds
A303	-		X	A302	YX YY	Subtracts
A304	*		X	A302	YX YY	Multiplies

TABLE D-I (CONTINUED)

GLOSSARY OF ROUTINES

Routine	Module Name	Sub-routine	Entry	Entry In Subroutine	Array Dependence Of Dimension Declarations In The Routine	Purpose
A305	/		X	A302	YX YY	Divides
A317			X	A302	YX YY	Blank
A312	INVERT	X			(YX YY)	Inverts
A306	**		X	A312	YX YY	Raises to a power
A307	SQRT		X	A312	YX YY	Square root
A309	LOGE		X	A312	YX YY	Natural logarithm
A311	EXP		X	A312	YX YY	Raises to a power
A322	SPLIT	X			YX YY	Plots graph
A323	PLOT		X	A322	YX YY	Plots graph
A324	SETGRIDY		X	A322	YX YY	Sets grid
A325	SETGRIDXY		X	A322	YX YY	Sets grid
A314	EMINMAX	X			(YX YY)	Finds max/min indices
A348	POINTER		X	A314	(YX YY)	Shifts block of data
A326	READTAPE1	X			(YX YY)	Reads tape
A346	READTAPE2		X	A326	(YX YY)	Reads tape
A327	WRITETAPE1		X	A326	(YX YY)	Writes tape
A347	WRITETAPE2		X	A326	(YX YY)	Writes tape
A328	MPLT2	X			YX YY AX AY	
A349	CALCOM2	X			YX YY AX AY	
MPLTAL	MPLT	X			(YX YY)	
CALCALL	CALCOMP	X			(YX YY)	
A329	DUMP	X			YX YY	Dumps nonarray common block
A330	PUNCH		X	A329	YX YY	Punches cards
A332	INTEGRATE	X			YX YY	Integrates numerically
SELECTR		X			None	Sets printing interval
A333	LASTTITLE1	X			(YX YY)	Last useful title on tape 1
A350	LASTTITLE2		X	A333	(YX YY)	Last useful title on tape 2
A334	LOADYLIMUP		X	A333	YX YY	Loads YLIMUP
A351	LOADYLIMLR		X	A333	YX YY	Loads YLIMLR
A336	AVOVERTIME	X			YX YY AX AY	Averages over time
A335	YROLLEDTOA		X	A336	YX YY AX AY	Transfers Y-arrays to A-arrays
A352	AROLLEDTOY		X	A336	YX YY AX AY	Transfers A-arrays to Y-arrays
A337	RUNSUM	X			YX YY	Running sum of YY-array
A353	RESINTEGRAL		X	A337	YX YY	Resonance integral of cross section vs energy
A338	NOEQUALX	X			YX YY	Combines points with same YX values
SEARCH	INTERPLATE	X			YX YY AX AY BY	
GRAPH		X			YX YY	
MINMAX		X			YX	
A340	PRINTPLOT	X			YX YY	Prints 11- x 17-in. graph on line printer
A341	MELDYANDA	X			YX YY AX AY	Combines Y- and A-arrays
A342	DELETEx=		X	A341	YX YY AX AY	Deletes specified point
A345			X	A341	YX YY AX AY	
A343	CLOCK	X			None	Prints out central-processor clock time
A344	A344	X			None	Spare
A870	A870	X			None	Spare
A872	A872	X			None	Spare
A874	A874	X			None	Spare
A876	A876	X			None	Spare
A888	A888	X			None	Spare
A900	A900	X			None	Spare
A902	A902	X			None	Spare
A300	EXCHANGE	X			None	Exchanges arguments
A301	TO		X	A300	None	Transfers arguments
A308		X			None	Error exit or dump

TABLE D-I (CONTINUED)
GLOSSARY OF ROUTINES

Routine	Module Name	Sub-routine	Entry	Entry In Subroutine	Array Dependence Of Dimension Declarations In The Routine	Purpose
CHECKER	LEGALORDER	X			YX YY	
MPL0T		X			YX YY	Plots graph on microfilm
MPL0T2		X			YX YY AX AY	Plots graph on microfilm
REMOV		X			None	
PRTPLT		X			None	
PLACE		X			None	
A399			X	PLACE	None	Lists table of contents
TANDCRD		X			YX YY	
TWRITE			X	TANDCRD	YX YY	Tape-writing routine
CALCOMP		X			YX YY	Plots graph on paper
CALCOM2		X			YX YY AX AY	Plots graph on paper
REMOOV		X			None	
BLOCK		X			None	
TAPWRI		X			None	
STORE		X			None	
NUMBER		X			None	
SCALE		X			None	
PLTZ		X			None	
PLOTS		X			None	

Note: Parentheses indicate implicit array dependence through called subroutines.

SUBROUTINES WRITTEN IN ASSEMBLY LANGUAGE

ROUTINES CONSTRUCTED BY THE TRANSLATOR

ENCODE
SYMBOL
SINCOS
GETEQN
UNLODE

FORPROG
A200
A201
A217

```

*COMDECK,COM1
  116 FORMAT(*      COMMON *A2* (6000) *)
  113 FORMAT(*      COMMON *A2* (6000) *)
*COMDECK,COM2
  COMMON X(6000),Y(6000)
*COMDECK,COM3
  COMMON AX(6000),AY(6000)
*COMDECK,COM4
  LENGTHY=6000
  LENGTHA=6000
*COMDECK,BLOCK
  COMMON NTELE,LENGTHY,LENGTHA,IREADLK
  COMMON C1,C2,C3,C4
  COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,
  INI,NF
  COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
  COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,
  1FAULTLT,DUMPLT,PRINO
  COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),
  1TITL(4),DUMMY(4)
  COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),
  1ID(3),II,JJ,LL
  COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,
  1DISKB,NFILE(2)
  COMMON LASTTTL(2,4)
  COMMON LOCK(2)
*COMDECK,SPARE

*COMDECK,COMYX
  COMMON YX(6000)
*COMDECK,COMYY
  COMMON YY(6000)
*COMDECK,COMAX
  COMMON AX(6000)
*COMDECK,COMAY
  COMMON AY(6000)
*COMDECK,COMBY
  COMMON BY(6000)
*COMDECK,COMCY
  COMMON CY(6000)
*COMDECK,REALDEC
  REAL C1,C2,C3,C4
  REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
  1ELEMENT
  REAL LASTTTL
*COMDECK,INTGDEC
  INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,
  INI,NF
  INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,
  1FAULTLT,DUMPLT,PRINO
  INTEGER LIMIT,ID,II,JJ,LL
  INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,
  1NFILE
*COMDECK,A5
  2020 FORMAT(6X,*COMMON YX(6000)*/6X,*COMMON *
  1*YY(6000) *)

```

Fig. D-1. UPDATE common decks.

TABLE D-II
UPDATE COMMON DECK USE INDEX

Subroutine	COMDEC														A5
	COM1	COM2	COM3	COM4	BLOCK	SPARE	COMYX	COMYY	COMAX	COMAY	COMBY	COMCY	REALDEC	INTDEC	
TELERAY	X	X													X
FREEB															
A321															
A212															
A216															
A319															
A215															
A202															
A221															
A222															
A223															
A224															
A225															
A226															
A231															
A232															
A233															
A234															
A235															
A236															
A315															
A316															
A320															
A302															
A312															
A322															
A314															
A326															
A328															
A349															
MPLTCAL															
CALLCALL															
A329															
A332															
SELECTR															
A333															
A336															
A337															
A338															
SEARCH															
GRAPH															
MINMAX															

TABLE D-II (CONTINUED)
UPDATE COMMON DECK USE INDEX

Subroutine	COMDEC														A5
	COM1	COM2	COM3	COM4	BLOCK	SPARF	COMYX	COMYY	COMAX	COMAY	COMBY	COMCY	REALDEC	INTDEC	
A340					X	X	X	X					X	X	
A341					X	X	X	X	X	X			X	X	
A343					X	X							X	X	
A344					X	X							X	X	
A870					X	X							X	X	
A872					X	X							X	X	
A874					X	X							X	X	
A876					X	X							X	X	
A888					X	X							X	X	
A900					X	X							X	X	
A902					X	X							X	X	
A300					X	X							X	X	
A308					X	X							X	X	
CHECKER					X	X	X	X					X	X	
MPL0T						X	X	X							
MPL0T2						X	X	X	X	X					
REMOV															
PRTPLT															
PLACE					X	X							X	X	
TANDCRD					X	X	X	X					X	X	
CALCOMP		X													
CALCOM2		X	X												
REMOOV															
BLOCK															
TAPEWRI															
STORE															
NUMBER															
SCALE															
PLTZ															
PLOTS															

SOURCE PROGRAM LISTING

```

      PROGRAM TELERAY (TAPE11=101,TAPE12=101,TAPE7=1001,OUTPUT=101,TAPE1
      18=1001,TAPE19=1001)
C     TAPE OR DISK UNITS. IF THE UNIT IS NOT ASSIGNED TO A TAPE IT IS
C     AUTOMATICALLY ASSUMED TO BE A DISC.
C     TAPE11 -LOCAL BATCH MODE OUTPUT TO THE LINE PRINTER.
C     TAPE12 -PRINTER OUTPUT FOR REMOTE BATCH MODE FROM A TERMINAL.
C             AT THE END OF THE PROGRAM, TRANSFER THIS FILE TO ONE
C             AVAILABLE TO THE REMOTE TERMINAL. FOR LOCAL BATCH
C             OPERATION, TAPE12 SHOULD BE ASSIGNED TO THE PUNCH FILE.
C     TAPE7  -INPUT FILE FOR RAYMATH PROGRAM DECK. AFTER THE MODULAR
C             PROGRAM IS ASSEMBLED, THIS UNIT BECOMES THE CALCOMP PLOTTER
C             OUTPUT. COPY THIS UNIT TO AN ASSIGNED TAPE AT THE END OF THE
C             PROGRAM IF THE CALCOMP MODULE HAS BEEN CALLED.
C     OUTPUT -LINE PRINTER OUTPUT. USED FOR FORTRAN AND SYSTEMS
C             DIAGNOSTIC MESSAGES.
C     TAPE18 -ASSEMBLED RAYMATH PROGRAM PLUS ID
C             AND TYPE OF EACH VARIABLE.
C     TAPE19 -FORTRAN SOURCE CARDS FOR PROGRAM FORPROG.
C     PROGRAM FORPROG FILES
C     TAPE10 -DATA CARD INPUT. SEPARATE EACH DATA BLOCK BY A CARD HAVING
C             A -1. IN THE FIRST FIELD OF 12 COLUMNS OR BY A BINARY
C             END-OF-FILE CARD.
C     TAPE11 -AS DESCRIBED ABOVE.
C     TAPE12 -AS DESCRIBED ABOVE.
C     FILM   -OUTPUT FOR THE SC4020 MICROFILM PLOTTER. A TAPE UNIT.
C     TAPE1  -A DATA TAPE OR DISK UNIT WRITTEN IN A 556 BIT/INCH FORMAT
C             READABLE BY THE READTAPE1 MODULE.
C     TAPE2  -A DATA TAPE OR DISK UNIT WRITTEN IN A 556 BIT/INCH FORMAT
C             READABLE BY THE READTAPE2 MODULE.
C     TAPE3  -HOLDS TABLE-OF-CONTENTS OF TAPE1.
C     TAPE4  -HOLDS TABLE-OF-CONTENTS OF TAPE2.
C     TAPE7  -OUTPUT DISK FOR THE CALCOMP PLOTTER.
C     TAPE18 -DESCRIBED ABOVE.
C     OUTPUT -LINE PRINTER OUTPUT. USED FOR FORTRAN AND SYSTEMS
C             DIAGNOSTIC MESSAGES.

      DIMENSION JUMP(11)
      COMMON WORD(16080),KK,OP,IDI(3),TYPE(3),MAX
      COMMON FIELD(9),NTELE,NEOF,NOARRAY
      REAL WORD,OP,TYPE,FIELD
      INTEGER KK,IDI,NTELE,NEOF,WORDDIM
      DATA WORDDIM/16080/
      REWIND 7
      REWIND 11
      REWIND 12
      REWIND 18
      REWIND 19
C     OBTAIN AND PRINT CALENDAR DATE
      CALL DATE (ZZ)
      WRITE (11,320)ZZ
      320 FORMAT(*1 SOURCE PROGRAM*10X*DATE *,A10,2X,*VERSION Y=A=6CCC FEB*
      1*0171*,//)
C     INTEGER VARIABLE INITIALIZATION
      NSTMCRD=WORDDIM/8
      NSTMCRD=NSTMCRD-1
      NOARRAY=0
      NOSTMT=0
      JKL=0
      KEY=0
      NSWITCH=0
C     COMMON INITIALIZATION
      MAX=0
      NTELE=0
      NEOF=0
      OP=10H
      DO 20 I=1,WORDDIM
      IF ((MOD(I,8)).NE.4) GO TO 10
      WORD(I)=0.
      GO TO 20
      10 WORD(I)=10H
      20 CONTINUE
      DO 30 I=1,3
      TYPE(I)=10H
      30 IC(I)=0
C     CHECK FOR OPEN OR OPENTELE FIRST STATEMENT DECLARATION
      GO TO 230
C     COUNT STATEMENTS AND CHECK FOR PRESENCE OF SPECIAL ONES
      JJ=1
      IF (JJ.GT.NSTMCRD) GO TO 200
C     ADDRESS OF INSTRUCTION CARD
      KK=8*(JJ-1)
C     NSWITCH=1 FOR FREE FIELD FORMAT
      IF (NSWITCH.EC.1) 300,280
      60 IF (WORD(2+KK).EQ.10HWRITETAPE1) KEY=1
      IF (WORD(2+KK).EQ.10HWRITETAPE2) KEY=1
      IF (WORD(2+KK).EQ.3HEND) NOSTMT=JJ
      JJ=JJ+1
      GO TO 50
C     ADVANCE PAGE
      70 WRITE (11,330)
      IF (NTELE.EQ.1) WRITE (12,330)
      330 FORMAT (I11)
      IF (NSWITCH.EC.1) 80,90
      80 KK=8*(JJ-1)
      90 WRITE (11,340)

```

```

340 IF (NTELE.EQ.1) WRITE (12,340)
    FORMAT (*1*13X,*ASSEMBLED PROGRAM*)
    KK=KK+8
    WORD(KK+2)=8HSTOPHERE
C END INSTRUCTION MISSING
    IF (NOSTMT.EQ.0) GO TO 190
C WRITE PROGRAM CARD AND COMMON DECLARATIONS
    WRITE (19,350)
350 FORMAT (* PROGRAM FORPRUG(TAPE10=101,TAPE11=101,FILM=600,TAPE
    112=101,TAPE1=
    21,TAPE7=101,OUTPUT=101,TAPE18=1
    11001,TAPE2=1001,TAPE3=101,TAPE4=10
    101)
    *)
    WRITE(19,1000)
1000 FORMAT(6X,*COMMON NTELE,LENGTHY,LENGTHA,IREADLK*/
    16X,*COMMON C1,C2,C3,C4*/
    26X,*COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF*/
    36X,*COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX*/
    46X,*COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,*
    5*PRIND*/
    66X,*COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),*
    7*DUMMY(4)*/
    86X,*COMMON WCRD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL*/
    96X,*COMMON NTITLE(2),NPTSBLK(2),TAPEND,DISKA,DISKB,NFILE(2)*/
    16X,*COMMON LASTTTL(2,4)*/
    26X,*COMMON LOCK(2)*/
    36X,*REAL C1,C2,C3,C4*/
    46X,*REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT*/
    56X,*REAL LASTTTL*/
    66X,*INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF*/
    76X,*INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,*
    8*PRIND*/
    96X,*INTEGER LIMIT,IO,II,JJ,LL*/
    16X,*INTEGER NTITLE,NPTSBLK,TAPEND,DISKA,DISKB,NFILE*///)
    WRITE(19,2020)
2020 FORMAT(6X,*COMMON YX(6000)*/6X,*COMMON YY(6000)*)
    WRITE (19,360)
360 FORMAT (80HC
    1
    WRITE (19,370)
370 FORMAT(15H CALL A215/15H CALL A217)
C HERE BEGINS THE GENERATION OF ASSEMBLED INSTRUCTIONS
    MM=1
    JUMP(1)=0
    JJ=1
100 CONTINUE
    KK=8*(JJ-1)
    IF (WORD(2+KK).EQ.3HEND) GO TO 160
    IF (WORD(2+KK).EQ.4HGOTO) GO TO 120
    IF (WORD(2+KK).EQ.6HRETURN) GO TO 110
    IF (WORD(2+KK).EQ.8HSTOPHERE)GOTO600
C CHECK FOR FORTRAN STATEMENT
    IF (WORD(1+KK).EQ.1H*).OR.(WORD(1+KK).EQ.1H*)GOTO900
C DETERMINE TYPE AND ID FOR EACH ARGUMENT
    CALL A212
C DETERMINE AND RECORD MAXIMUM ARRAY ID ENTERED SO FAR.
    DO 102 I=1,3,2
102 IF (TYPE(I).EQ.5HARRAY)MAX=MAXO(MAX,ID(I))
C WRITE ASSEMBLED INSTRUCTION ON OUTPUT AND TELETYPE
    WRITE (11,380)(WORD(I+KK),I=1,8)
380 FORMAT (* *12X,A2,2A10,E15.6,4A10)
    IF (NTELE.EQ.1) WRITE (12,390)(WORD(I+KK),I=1,8)
390 FORMAT (1X,*$*,A2,2A10,E14.6,4A10,*$*)
C SEARCH MODULE LIST
    CALL A321
    IF (OP.EQ.4HA308)WORD(KK+2)=4HA308
C RECORD COMPLETE INSTRUCTION AND ID AND TYPE ON TAPE18
    WRITE (18,400)WORD(KK+1),TYPE(1),ID(1),WORD(KK+2),WORD(KK+3),TYPE(
    13),ID(3),WORD(KK+1),I=4,8)
400 FORMAT (2A10,I10,2A10,I10,E15.6,4A10)
C WRITE CALLS FOR FORTRAN PROGRAM
    MAX=MAXO(MAX,NOARRAY)
    WRITE (19,410)OP
410 FORMAT(15H CALL A319/11H CALL ,A10)
    GO TO 180
110 IF (JUMP(1).EQ.0) GO TO 220
    MM=MM-1
    JJ=JUMP(MM)
    GO TO 180
120 JUMP(MM)=JJ
    ITEMP=JJ
    MM=MM+1
    IF (MM.GT.11) GO TO 210
    JJ=NOSTMT+1
C SEARCH FOR CALLED GROUP
    DO 150 K=JJ,NSTMCRD
    KK=8*(K-1)
    IF ((WORD(2+KK).EQ.6HHEREIS).AND.(WORD(4+KK).EQ.WORD(4+8*(ITEMP-1)
    1))) 130,150
130 DO 140 MJK=5,8
    INDEX3=MJK+8*(ITEMP-1)
    IF (WORD(MJK+KK).EQ.WORD(INDEX3)) 140,150
140 CONTINUE
    GO TO 170
150 CONTINUE
    WRITE (11,420)

```

```

      IF (INTELE.EQ.1) WRITE (12,420)
420  FORMAT (* CALLED GROUP NOT IN PROGRAM*)
      GOTO700
160  CONTINUE
C   CHECK WHETHER TAPE TABLES OF CONTENTS SHOULD BE WRITTEN
      IF (KEY.EQ.1) WRITE (19,430)
430  FORMAT (15H      CALL A318)
C   WRITE END CARD FOR GENERATED FORTRAN PROGRAM FORPROG
      WRITE (19,440)
440  FORMAT(9H      END)
C   CONSTRUCT SUBROUTINES A200, A201, AND A217
      CALL A216
C   REWIND FORPROG FILES
      REWIND 7
      REWIND 18
      REWIND 19
      CALL EXIT
170  JJ=K
180  JJ=JJ+1
C   END OF LOOP 100
      GO TO 100
C   DIAGNOSTICS FOR FAULTS DETECTED DURING PROGRAM ASSEMBLY
190  WRITE (11,450)
      IF (INTELE.EQ.1) WRITE (12,450)
450  FORMAT (* END INSTRUCTION MISSING*)
      GOTO700
200  WRITE (11,460)NSTMCRD
      IF (INTELE.EQ.1) WRITE (12,460)NSTMCRD
460  FORMAT (* OBJECT PROGRAM EXCEEDS*,15,* STEPS*)
      GOTO700
210  WRITE (11,470)
      IF (INTELE.EQ.1) WRITE (12,470)
470  FORMAT (* SEQUENTIAL GROUP CALL EXCEEDS 10*)
      GOTO700
220  WRITE (11,480)
      IF (INTELE.EQ.1) WRITE (12,480)
480  FORMAT (* RETURN STATEMENT OUT OF PLACE*)
      GOTO700
600  WRITE(11,610)
      IF(INTELE.EQ.1)WRITE(12,610)
610  FORMAT(* FINAL RETURN STATEMENT MISSING*)
      GOTO700
C   PASS FORTRAN STATEMENTS TO FORPROG
900  IF(WORD(1+KK).NE.1H$)GO TO 913
      WRITE (19,912) (WORD(JKL+KK),JKL=2,3), (WORD(JKL+KK),JKL=5,8)
912  FORMAT(1H ,A4,A1,4A10)
      GO TO 914
913  WRITE (19,910) (WORD(JKL+KK),JKL=1,3), (WORD(JKL+KK),JKL=5,8)
910  FORMAT(A1,A4,A1,4A10)
C   WRITE FORTRAN STATEMENT INTO ASSEMBLED RAYMATH PROGRAM
914  STAR=WORD(1+KK)
      IF(STAR.EQ.1H$)STAR=10H
      WRITE(11,920)STAR,(WORD(JKL+KK),JKL=2,3), (WORD(JKL+KK),JKL=5,8)
      IF(INTELE.EQ.1)WRITE(12,930)STAR,(WORD(JKL+KK),JKL=2,3),
1    (WORD(JKL+KK),JKL=5,8)
920  FORMAT(25X,19H**FORTRAN INST.** ,A1,A4,A1,4A10)
930  FORMAT(* $ ,11X,19H**FORTRAN INST.** ,A1,A4,A1,4A10,**$)
      GO TO 180
C   DETERMINATION OF INPUT CARD FORMAT OF INSTRUCTION CARDS
230  READ (7,490)WORD(1)
490  FORMAT (A10)
      IF (EOF,7) 240,250
240  WRITE (11,500)
      WRITE(12,500)
500  FORMAT (*1NO PROGRAM CARDS ON TAPE 7*)
700  IF(INTELE.EQ.1)WRITE(12,702)
      WRITE(11,702)
702  FORMAT(* FATAL ERROR IN TELERAY, EXPECT FORMAT ERROR IN OUTPUT */*
1    FILE AND IN DAY FILE*)
      PRINT 702,JJ
      CALL EXIT
C   FREE FIELD FORMAT
250  IF (1WORD(1).EQ.4HOPEN'.OR.(WORD(1).EQ.8HOPEN TELE)) 260,270
260  NSWITCH=1
      WRITE (11,510)
510  FORMAT (* OPEN FORMAT*)
      IF (WORD(1).EQ.8HOPEN TELE) WRITE (12,320)ZZ
      IF (WORD(1).EQ.8HOPEN TELE) WRITE (12,520)
520  FORMAT (* TELETYPE OUTPUT*)
      IF (WORD(1).EQ.8HOPEN TELE) NTELE=1
      WRITE (18,530)NTELE
530  FORMAT (12)
      GO TO 40
270  REWIND 7
      NTELE=0
      WRITE(18,530)NTELE
      GO TO 40
C   FREE FIELD FORMAT
280  READ (7,540)(WORD(I+KK),I=1,8)
540  FORMAT (3XA2,2A10,E15.6,4A10)
      IF (EOF,7) 70,290
290  WRITE (11,550)JJ,(WORD(I+KK),I=1,8)
550  FORMAT (* *10X13,2XA2,2A10,E15.6,4A10)
      GO TO 60
C   DECODE FREE FIELD FORMAT

```

```

TELE 191
TELE 192
TELE 193
TELE 194
TELE 195
TELE 196
TELE 197
TELE 198
TELE 199
TELE 200
TELE 201
TELE 202
TELE 203
TELE 204
TELE 205
TELE 206
TELE 207
TELE 208
TELE 209
TELE 210
TELE 211
TELE 212
TELE 213
TELE 214
TELE 215
TELE 216
TELE 217
TELE 218
TELE 219
TELE 220
TELE 221
TELE 222
TELE 223
TELE 224
TELE 225
TELE 226
TELE 227
TELE 228
TELE 229
TELE 230
TELE 231
TELE 232
TELE 233
TELE 234
TELE 235
TELE 236
TELE 237
TELE 238
TELE 239
TELE 240
TELE 241
TELE 242
TELE 243
TELE 244
TELE 245
TELE 246
TELE 247
TELE 248
TELE 249
TELE 250
TELE 251
TELE 252
TELE 253
TELE 254
TELE 255
TELE 256
TELE 257
TELE 258
TELE 259
TELE 260
TELE 261
TELE 262
TELE 263
TELE 264
TELE 265
TELE 266
TELE 267
TELE 268
TELE 269
TELE 270
TELE 271
TELE 272
TELE 273
TELE 274
TELE 275
TELE 276
TELE 277
TELE 278
TELE 279
TELE 280
TELE 281
TELE 282
TELE 283
TELE 284
TELE 285

```

TELE 286
TELE 287
TELE 288
TELE 289
TELE 290
TELE 291
TELE 292

FREE	1
FREE	2
FREE	3
FREE	4
FREE	5
FREE	6
FREE	7
FREE	8
FREE	9
FREE	10
FREE	11
FREE	12
FREE	13
FREE	14
FREE	15
FREE	16
FREE	17
FREE	18
FREE	19
FREE	20
FREE	21
FREE	22
FREE	23
FREE	24
FREE	25
FREE	26
FREE	27
FREE	28
FREE	29
FREE	30
FREE	31
FREE	32
FREE	33
FREE	34
FREE	35
FREE	36
FREE	37
FREE	38
FREE	39
FREE	40
FREE	41
FREE	42
FREE	43
FREE	44
FREE	45
FREE	46
FREE	47
FREE	48
FREE	49
FREE	50
FREE	51
FREE	52
FREE	53
FREE	54
FREE	55
FREE	56
FREE	57
FREE	58
FREE	59
FREE	60
FREE	61
FREE	62
FREE	63
FREE	64
FREE	65
FREE	66
FREE	67
FREE	68
FREE	69
FREE	70
FREE	71
FREE	72
FREE	73
FREE	74
FREE	75
FREE	76
FREE	77
FREE	78
FREE	79
FREE	80
FREE	81
FREE	82
FREE	83

```

65 KF(J)=KI(J)+40
C PHASE 2 - INITIAL AND FINAL VALUES FOUND FOR ALL FIELDS
C CALCULATE FIELD LENGTHS - RECORD AS ERROR IF TOO LONG.
100 DO 110 I=1,5
    KE(I)=0
    KL(I)=KF(I)-KI(I)
    IF(KL(I).LE.KLM(I)) GO TO 110
    KF(I)=KI(I)+KLM(I)
    KL(I)=KLM(I)
    KE(I)=1
110 CONTINUE
    VERIFY THAT FIELD 4 CAN BE READ BY F FORMAT
C DO 120 I=6,10
    KE(I)=0
120 IF(KL(4).EQ.0) GO TO 152
    IF(KI(4).GE.73) GO TO 152
    KLE=0
    KNM=0
    KPM=0
    KDC=0
    I1=KI(4)
    I2=KF(4)-1
    DO 150 I=I1,I2
        IF(KB(I).EQ.1R) GO TO 150
        IF(KB(I).GE.1R0.AND.KB(I).LE.1R9) GO TO 145
        IF(KB(I).EQ.1R) GO TO 130
        IF(KB(I).EQ.1R+.OR.KB(I).EQ.1R-) GO TO 135
        IF(KB(I).EQ.1RE) GO TO 140
C ILLEGAL CHARACTER IN FIELD
        KE(8)=I
        GO TO 150
130 IF(KDC.EQ.1) GO TO 132
    KDC=1
    GO TO 150
C MORE THAN ONE DECIMAL POINT IN FIELD
132 KE(6)=I
    GO TO 150
135 IF(KPM.EQ.1) GO TO 137
    KPM=1
    GO TO 150
C TOO MANY PLUS AND/OR MINUS SIGNS IN FIELD
137 KE(7)=I
    GO TO 150
140 IF(KLE.EQ.1) GO TO 142
    IF(KPM.EQ.1) GO TO 147
    KLE=1
    KPM=0
    GO TO 150
C OCCURRENCE OF MORE THAN ONE E IN FIELD
142 KE(9)=I
    GO TO 150
145 IF(KNM.NE.0) GO TO 150
    KPM=0
    KNM=1
    GO TO 150
C PLUS SIGN PRECEDES E
147 KE(10)=I
150 CONTINUE
C PRINT OUT ALL DIAGNOSTICS FOUND, WITH CORRECTIVE ACTION
152 KLE=0
    DO 160 I=1,5
        IF(KE(I).EQ.0) GO TO 160
        IF(KLE.NE.0) GO TO 155
        WRITE(11,902)
        IF(INTELE.EQ.1)WRITE(12,902)
        KLE=1
155 WRITE(11,903)KE(I)
        IF(INTELE.EQ.1)WRITE(12,903)KE(I)
160 CONTINUE
        IF(KLE.NE.1) GO TO 162
        WRITE(11,904)
        IF(INTELE.EQ.1)WRITE(12,904)
        FIELD(9)=1
162 DO 170 I=6,10
        IF(KE(I).EQ.0) GO TO 170
        IF(KLE.NE.0) GO TO 165
        WRITE(11,902)
        IF(INTELE.EQ.1)WRITE(12,902)
165 KLE=-1
        J=I-5
        WRITE(11,905)KER(J)
        IF(INTELE.EQ.1)WRITE(12,905)KER(J)
170 CONTINUE
        IF(KLE.GE.0) GO TO 200
        WRITE(11,906)
        IF(INTELE.EQ.1)WRITE(12,906)
        FIELD(9)=-1
    RETURN
C PHASE 3 - CARD LOOKS OK TO READ - SET UP FORMAT
C BLANK OUT COL 73-80 IN KD
C 200 KD(8)=(KD(8).AND..NOT.77777777777777778).OR.8R
C BLANK OUT COMMAS TO ALLOW CONVENIENT DECODE OF NULL FIELDS
    IF(KCMX.EQ.0) GO TO 212
    DO 210 I=1,KCMX
        J=KC(I)

```

```

FREE 84
FREE 85
FREE 86
FREE 87
FREE 88
FREE 89
FREE 90
FREE 91
FREE 92
FREE 93
FREE 94
FREE 95
FREE 96
FREE 97
FREE 98
FREE 99
FREE 100
FREE 101
FREE 102
FREE 103
FREE 104
FREE 105
FREE 106
FREE 107
FREE 108
FREE 109
FREE 110
FREE 111
FREE 112
FREE 113
FREE 114
FREE 115
FREE 116
FREE 117
FREE 118
FREE 119
FREE 120
FREE 121
FREE 122
FREE 123
FREE 124
FREE 125
FREE 126
FREE 127
FREE 128
FREE 129
FREE 130
FREE 131
FREE 132
FREE 133
FREE 134
FREE 135
FREE 136
FREE 137
FREE 138
FREE 139
FREE 140
FREE 141
FREE 142
FREE 143
FREE 144
FREE 145
FREE 146
FREE 147
FREE 148
FREE 149
FREE 150
FREE 151
FREE 152
FREE 153
FREE 154
FREE 155
FREE 156
FREE 157
FREE 158
FREE 159
FREE 160
FREE 161
FREE 162
FREE 163
FREE 164
FREE 165
FREE 166
FREE 167
FREE 168
FREE 169
FREE 170
FREE 171
FREE 172
FREE 173
FREE 174
FREE 175
FREE 176
FREE 177
FREE 178

```

	N=((J-1)/10)+1	FREE 179
	M=J-10*(N-1)	FREE 180
210	KD(N)=(KD(N),AND.,NOT,KM(M)).OR.(KBL.AND.,KM(M))	FREE 181
C	INITIALIZE FORMAT	FREE 182
212	DO 215 I=1,5	FREE 183
215	KFF(I+1)=KFI(I)	FREE 184
C	SPECIAL TREATMENT TO SKIP TO BEGINNING OF FIRST FIELD	FREE 185
	IF(KI(I).NE.1) GO TO 217	FREE 186
	KFF(2)=(KBL.AND.7777770000000000B).OR.KFF(2)	FREE 187
	GO TO 220	FREE 188
C	CONVERT KI(1)-1 TO DISPLAY	FREE 189
217	L=(KI(1)-1)/10	FREE 190
	M=KI(1)-10*L-1	FREE 191
	KV=(L*64+M)+3333B	FREE 192
C	AFFIX X(DISPLAY) AND OR INTO FORMAT	FREE 193
	KFF(2)=KFF(2).OR.((KV*64+30B)*1073741824)	FREE 194
C	HANDLE REMAINDER OF FORMAT	FREE 195
220	DO 250 I=1,4	FREE 196
	IF(KL(I).EQ.0) GO TO 252	FREE 197
	L=KL(I)/10	FREE 198
	M=KL(I)-10*L	FREE 199
	KV=(L*64+M)+3333C	FREE 200
222	KFF(I+1)=KFF(I+1).OR.(KV*64)	FREE 201
	KX=KI(I+1)-KF(I)	FREE 202
	IF(KX.EQ.0) GO TO 254	FREE 203
	L=KX/10	FREE 204
	M=KX-10*L	FREE 205
	KV=(L*64+M)+3333B	FREE 206
	KFF(I+2)=KFF(I+2).OR.((KV*64+30B)*1073741824)	FREE 207
	GO TO 250	FREE 208
252	KF(I)=KF(I)+1	FREE 209
	KL(I)=1	FREE 210
	KV=3334B	FREE 211
	GO TO 222	FREE 212
254	KFF(I+2)=(KBL.AND.7777770000000000B).OR.KFF(I+2)	FREE 213
250	CONTINUE	FREE 214
C	FORMAT FINISHED, DECODE AND RETURN	FREE 215
300	DECODE(120,KFF,KD) (FIELD(I),I=1,8)	FREE 216
	RETURN	FREE 217
950	FORMAT(1X,7A10,A9)	FREE 218
901	FORMAT(72R1)	FREE 219
903	FORMAT(1X,* FIELD *,13,* IS TOO LONG*)	FREE 220
902	FORMAT(1X,* THE ABOVE CARD IS IN ERROR AS FOLLOWS...*)	FREE 221
904	FORMAT(1X* OVERLONG FIELDS ARE RIGHT TRUNCATED-CARD WILL BE DECODED*)	FREE 222
905	FORMAT(1X* ERROR IN NUMERIC FIELD-*,A10)	FREE 223
906	FORMAT(1X* CARD ABANDONED - ERROR RETURN INDICATED*/)	FREE 224
	END	FREE 225

SUBROUTINE A321		A321 1
C	OPERATIONS ROUTINE	A321 2
C	SEARCH FOR ROUTINES WHICH EXECUTE MODULAR INSTRUCTIONS	A321 3
	COMMON WORD(16080),KK,OP,EDUM(18),NOARRAY	A321 4
	REAL WORD,OP	A321 5
	INTEGER KK	A321 6
	NOARRAY=0	A321 7
	OP=10H	A321 8
10	IF (WORD(KK+2).NE.4HREAD) GO TO 20	A321 9
	OP=4HA315	A321 10
	NOARRAY=2	A321 11
	RETURN	A321 12
20	IF (WORD(KK+2).NE.5HPRINT) GO TO 30	A321 13
	OP=4HA316	A321 14
	NOARRAY=2	A321 15
	RETURN	A321 16
30	IF (WORD(KK+2).NE.9HREADTAPE1)GOTO35	A321 17
	OP=4HA326	A321 18
	NOARRAY=2	A321 19
	RETURN	A321 20
35	IF (WORD(KK+2).NE.9HREADTAPE2)GOTO40	A321 21
	OP=4HA346	A321 22
	NOARRAY=2	A321 23
	RETURN	A321 24
40	IF (WORD(KK+2).NE.10HWRITETAPE1)GOTO45	A321 25
	OP=4HA327	A321 26
	NOARRAY=2	A321 27
	RETURN	A321 28
45	IF (WORD(KK+2).NE.10HWRITETAPE2)GOTO50	A321 29
	OP=4HA347	A321 30
	NOARRAY=2	A321 31
	RETURN	A321 32
50	IF (WORD(KK+2).NE.8HEXCHANGE) GO TO 60	A321 33
	OP=4HA300	A321 34
	RETURN	A321 35
60	IF (WORD(KK+2).NE.2HTO) GO TO 70	A321 36
	OP=4HA301	A321 37
	RETURN	A321 38
70	IF (WORD(KK+2).NE.10HINTERPLATE) GO TO 80	A321 39
		A321 40
		A321 41
		A321 42

	OP=6HSEARCH	A321	43
	NOARRAY=5	A321	44
	RETURN	A321	45
80	IF (WORD(KK+2).NE.6HMPLOT2)GOTO82	A321	46
	OP=4HA328	A321	47
	NOARRAY=4	A321	48
	RETURN	A321	49
82	IF (WORD(KK+2).NE.7HCALCOM2)GOTO85	A321	50
	OP=4HA349	A321	51
	NOARRAY=4	A321	52
	RETURN	A321	53
85	IF (WORD(KK+2).NE.5HMPLOT)GOTO87	A321	54
	OP=7HMPLTCL	A321	55
	NOARRAY=2	A321	56
	RETURN	A321	57
87	IF (WORD(KK+2).NE.7HCALCOMP)GOTO90	A321	58
	OP=7HCALCALL	A321	59
	NOARRAY=2	A321	60
	RETURN	A321	61
90	IF (WORD(KK+2).NE.5HPUNCH) GO TO 100	A321	62
	OP=4HA330	A321	63
	NOARRAY=2	A321	64
	RETURN	A321	65
100	IF (WORD(KK+2).NE.10HLEGALORDER) GO TO 110	A321	66
	OP=7HCHECKER	A321	67
	NOARRAY=2	A321	68
	RETURN	A321	69
110	IF (WORD(KK+2).NE.9HINTEGRATE) GO TO 120	A321	70
	OP=4HA332	A321	71
	NOARRAY=2	A321	72
	RETURN	A321	73
120	IF (WORD(KK+2).NE.1H+) GO TO 130	A321	74
	OP=4HA302	A321	75
	NOARRAY=2	A321	76
	RETURN	A321	77
130	IF (WORD(KK+2).NE.1H-) GO TO 140	A321	78
	OP=4HA303	A321	79
	NOARRAY=2	A321	80
	RETURN	A321	81
140	IF (WORD(KK+2).NE.1H*) GO TO 150	A321	82
	OP=4HA304	A321	83
	NOARRAY=2	A321	84
	RETURN	A321	85
150	IF (WORD(KK+2).NE.1H/) GO TO 160	A321	86
	OP=4HA305	A321	87
	NOARRAY=2	A321	88
	RETURN	A321	89
160	IF (WORD(KK+2).NE.10H) GO TO 170	A321	90
	OP=4HA317	A321	91
	NOARRAY=2	A321	92
	RETURN	A321	93
170	IF (WORD(KK+2).NE.3HEXP) GO TO 180	A321	94
	OP=4HA311	A321	95
	NOARRAY=2	A321	96
	RETURN	A321	97
180	IF (WORD(KK+2).NE.6HINVERT) GO TO 190	A321	98
	OP=4HA312	A321	99
	NOARRAY=2	A321	100
	RETURN	A321	101
190	IF (WORD(KK+2).NE.2H**) GO TO 200	A321	102
	OP=4HA306	A321	103
	NOARRAY=2	A321	104
	RETURN	A321	105
200	IF (WORD(KK+2).NE.4HSQRT) GO TO 210	A321	106
	OP=4HA307	A321	107
	NOARRAY=2	A321	108
	RETURN	A321	109
210	IF (WORD(KK+2).NE.5HSPLLOT) GO TO 220	A321	110
	OP=4HA322	A321	111
	NOARRAY=2	A321	112
	RETURN	A321	113
220	IF (WORD(KK+2).NE.7HEMINMAX)GOTO225	A321	114
	OP=4HA314	A321	115
	NOARRAY=2	A321	116
	RETURN	A321	117
225	IF (WORD(KK+2).NE.7HPOINTER)GOTO230	A321	118
	OP=4HA348	A321	119
	NOARRAY=2	A321	120
	RETURN	A321	121
230	IF (WORD(KK+2).NE.4HPLOT) GO TO 240	A321	122
	OP=4HA323	A321	123
	NOARRAY=2	A321	124
	RETURN	A321	125
240	IF (WORD(KK+2).NE.8HSETGRIDY) GO TO 250	A321	126
	OP=4HA324	A321	127
	NOARRAY=2	A321	128
	RETURN	A321	129
250	IF (WORD(KK+2).NE.9HSETGRIDXY) GO TO 260	A321	130
	OP=4HA325	A321	131
	NOARRAY=2	A321	132
	RETURN	A321	133
260	IF (WORD(KK+2).NE.4HDUMP) GO TO 270	A321	134
	OP=4HA329	A321	135
	NOARRAY=2	A321	136
	RETURN	A321	137

270	IF (WORD(KK+2).NE.4HLOGE) GO TO 280	A321 138
	OP=4HA309	A321 139
	NOARRAY=2	A321 140
	RETURN	A321 141
280	IF (WORD(KK+2).NE.10HPRALLARRAY) GOTO290	A321 142
	OP=4HA320	A321 143
	NOARRAY=6	A321 144
	RETURN	A321 145
290	IF (WORD(KK+2).NE.10HLASTTITLE1) GOTO295	A321 146
	OP=4HA333	A321 147
	NOARRAY=2	A321 148
	RETURN	A321 149
295	IF (WORD(KK+2).NE.10HLASTTITLE2) GOTO300	A321 150
	OP=4HA350	A321 151
	NOARRAY=2	A321 152
	RETURN	A321 153
300	IF (WORD(KK+2).NE.10HLOADYLIMUP) GOTO305	A321 154
	OP=4HA334	A321 155
	NOARRAY=2	A321 156
	RETURN	A321 157
305	IF (WORD(KK+2).NE.10HLOADYLIMLR) GOTO310	A321 158
	OP=4HA351	A321 159
	NOARRAY=2	A321 160
	RETURN	A321 161
310	IF (WORD(KK+2).NE.10HYROLLEDTOA) GOTO315	A321 162
	OP=4HA335	A321 163
	NOARRAY=4	A321 164
	RETURN	A321 165
315	IF (WORD(KK+2).NE.10HAROLLEDTOY) GOTO320	A321 166
	OP=4HA352	A321 167
	NOARRAY=4	A321 168
	RETURN	A321 169
320	IF (WORD(KK+2).NE.10HAVOVERTIME) GO TO 330	A321 170
	OP=4HA336	A321 171
	NOARRAY=4	A321 172
	RETURN	A321 173
330	IF (WORD(KK+2).NE.6HRUNSUM) GOTO335	A321 174
	OP=4HA337	A321 175
	NOARRAY=2	A321 176
	RETURN	A321 177
335	IF (WORD(KK+2).NE.10HRESINTEGR1) GOTO340	A321 178
	OP=4HA353	A321 179
	NOARRAY=2	A321 180
	RETURN	A321 181
340	IF (WORD(KK+2).NE.8HNOEQUALX) GO TO 350	A321 182
	OP=4HA338	A321 183
	NOARRAY=2	A321 184
	RETURN	A321 185
350	IF (WORD(KK+2).NE.9HPRINTPLOT) GO TO 360	A321 186
	OP=4HA340	A321 187
	NOARRAY=2	A321 188
	RETURN	A321 189
360	IF (WORD(KK+2).NE.9HMELDYANDA) GO TO 370	A321 190
	OP=4HA341	A321 191
	NOARRAY=4	A321 192
	RETURN	A321 193
370	IF (WORD(KK+2).NE.8HDELETEx) GO TO 380	A321 194
	OP=4HA342	A321 195
	NOARRAY=4	A321 196
	RETURN	A321 197
C	NOARRAY=0	A321 198
380	IF (WORD(KK+2).NE.4HA344) GOTO381	A321 199
	OP=4HA344	A321 200
	RETURN	A321 201
381	IF (WORD(KK+2).NE.4HA870) GOTO382	A321 202
	OP=4HA870	A321 203
	RETURN	A321 204
382	IF (WORD(KK+2).NE.4HA872) GOTO383	A321 205
	OP=4HA872	A321 206
	RETURN	A321 207
383	IF (WORD(KK+2).NE.4HA874) GOTO384	A321 208
	OP=4HA874	A321 209
	RETURN	A321 210
384	IF (WORD(KK+2).NE.4HA876) GOTO385	A321 211
	OP=4HA876	A321 212
	RETURN	A321 213
385	IF (WORD(KK+2).NE.4HA888) GOTO386	A321 214
	OP=4HA888	A321 215
	RETURN	A321 216
386	IF (WORD(KK+2).NE.4HA900) GOTO387	A321 217
	OP=4HA900	A321 218
	RETURN	A321 219
387	IF (WORD(KK+2).NE.4HA902) GOTO388	A321 220
	OP=4HA902	A321 221
	RETURN	A321 222
388	IF (WORD(KK+2).NE.5HCLOCK) GO TO 390	A321 223
	OP=4HA343	A321 224
	RETURN	A321 225
390	OP=4HA308	A321 226
	RETURN	A321 227
	END	A321 228

C SUBROUTINE A212
ARGUMENT CLASSIFICATION AND NUMBERING

```

COMMON WORD(16080),KK,DUM,ID(3),TYPE(3)
REAL WORD,TYPE
INTEGER KK,ID

DO650 I=1,3,2
IF(WORD(KK+I).NE.2HYX)GOTO2
ID(I)=1
TYPE(I)=5HARRAY
GOTO650

2 IF(WORD(KK+I).NE.2HYY)GOTO3
ID(I)=2
TYPE(I)=5HARRAY
GOTO650

3 IF(WORD(KK+I).NE.2HAX)GOTO4
ID(I)=3
TYPE(I)=5HARRAY
GOTO650

4 IF(WORD(KK+I).NE.2HAY)GOTO5
ID(I)=4
TYPE(I)=5HARRAY
GOTO650

5 IF(WORD(KK+I).NE.2HBY)GOTO6
ID(I)=5
TYPE(I)=5HARRAY
GOTO650

6 IF(WORD(KK+I).NE.2HCY)GOTO7
ID(I)=6
TYPE(I)=5HARRAY
GOTO650

7 IF(WORD(KK+I).NE.2HC1)GOTO8
ID(I)=7
TYPE(I)=8HCONSTANT
GOTO650

8 IF(WORD(KK+I).NE.2HC2)GOTO9
ID(I)=8
TYPE(I)=8HCONSTANT
GOTO650

9 IF(WORD(KK+I).NE.2HC3)GOTO10
ID(I)=9
TYPE(I)=8HCONSTANT
GOTO650

10 IF(WORD(KK+I).NE.2HC4)GOTO11
ID(I)=10
TYPE(I)=8HCONSTANT
GOTO650

11 IF(WORD(KK+I).NE.2HNY)GOTO12
ID(I)=11
TYPE(I)=9HD0LOOPLMT
GOTO650

12 IF(WORD(KK+I).NE.2HNA)GOTO13
ID(I)=12
TYPE(I)=9HD0LOOPLMT
GOTO650

13 IF(WORD(KK+I).NE.2HNB)GOTO14
ID(I)=13
TYPE(I)=9HD0LOOPLMT
GOTO650

14 IF(WORD(KK+I).NE.2HNC)GOTO15
ID(I)=14
TYPE(I)=9HD0LOOPLMT
GOTO650

15 IF(WORD(KK+I).NE.2HYI)GOTO16

```

```

A212 1
A212 2
A212 3
A212 4
A212 5
A212 6
A212 7
A212 8
A212 9
A212 10
A212 11
A212 12
A212 13
A212 14
A212 15
A212 16
A212 17
A212 18
A212 19
A212 20
A212 21
A212 22
A212 23
A212 24
A212 25
A212 26
A212 27
A212 28
A212 29
A212 30
A212 31
A212 32
A212 33
A212 34
A212 35
A212 36
A212 37
A212 38
A212 39
A212 40
A212 41
A212 42
A212 43
A212 44
A212 45
A212 46
A212 47
A212 48
A212 49
A212 50
A212 51
A212 52
A212 53
A212 54
A212 55
A212 56
A212 57
A212 58
A212 59
A212 60
A212 61
A212 62
A212 63
A212 64
A212 65
A212 66
A212 67
A212 68
A212 69
A212 70
A212 71
A212 72
A212 73
A212 74
A212 75
A212 76
A212 77
A212 78
A212 79
A212 80
A212 81
A212 82
A212 83
A212 84
A212 85
A212 86
A212 87
A212 88
A212 89
A212 90
A212 91
A212 92
A212 93
A212 94
A212 95

```

```

      ID(II)=15
      TYPE(II)=9HDOOLOPLMT
      GOTO650

16 IF (WORD(KK+II).NE.2HYF)GOTO17
      ID(II)=16
      TYPE(II)=9HDOOLOPLMT
      GOTO650

17 IF (WORD(KK+II).NE.2HAI)GOTO18
      ID(II)=17
      TYPE(II)=9HDOOLOPLMT
      GOTO650

18 IF (WORD(KK+II).NE.2HAF)GOTO19
      ID(II)=18
      TYPE(II)=9HDOOLOPLMT
      GOTO650

19 IF (WORD(KK+II).NE.2HBI)GOTO20
      ID(II)=19
      TYPE(II)=9HDOOLOPLMT
      GOTO650

20 IF (WORD(KK+II).NE.2HBF)GOTO21
      ID(II)=20
      TYPE(II)=9HDOOLOPLMT
      GOTO650

21 IF (WORD(KK+II).NE.2HCI)GOTO22
      ID(II)=21
      TYPE(II)=9HDOOLOPLMT
      GOTO650

22 IF (WORD(KK+II).NE.2HCF)GOTO23
      ID(II)=22
      TYPE(II)=9HDOOLOPLMT
      GOTO650

23 IF (WORD(KK+II).NE.2HNI)GOTO24
      ID(II)=23
      TYPE(II)=9HDOOLOPLMT
      GOTO650

24 IF (WORD(KK+II).NE.2HNF)GOTO25
      ID(II)=24
      TYPE(II)=9HDOOLOPLMT
      GOTO650

25 IF (WORD(KK+II).NE.6HXLIMUP)GOTO26
      ID(II)=25
      TYPE(II)=9HPARAMETER
      GOTO650

26 IF (WORD(KK+II).NE.6HXLIMLR)GOTO27
      ID(II)=26
      TYPE(II)=9HPARAMETER
      GOTO650

27 IF (WORD(KK+II).NE.6HYLIMUP)GOTO28
      ID(II)=27
      TYPE(II)=9HPARAMETER
      GOTO650

28 IF (WORD(KK+II).NE.6HYLIMLR)GOTO29
      ID(II)=28
      TYPE(II)=9HPARAMETER
      GOTO650

29 IF (WORD(KK+II).NE.4HEMIN)GOTO30
      ID(II)=29
      TYPE(II)=9HPARAMETER
      GOTO650

30 IF (WORD(KK+II).NE.4HEMAX)GOTO31
      ID(II)=30
      TYPE(II)=9HPARAMETER
      GOTO650

```

```

A212 96
A212 97
A212 98
A212 99
A212 100
A212 101
A212 102
A212 103
A212 104
A212 105
A212 106
A212 107
A212 108
A212 109
A212 110
A212 111
A212 112
A212 113
A212 114
A212 115
A212 116
A212 117
A212 118
A212 119
A212 120
A212 121
A212 122
A212 123
A212 124
A212 125
A212 126
A212 127
A212 128
A212 129
A212 130
A212 131
A212 132
A212 133
A212 134
A212 135
A212 136
A212 137
A212 138
A212 139
A212 140
A212 141
A212 142
A212 143
A212 144
A212 145
A212 146
A212 147
A212 148
A212 149
A212 150
A212 151
A212 152
A212 153
A212 154
A212 155
A212 156
A212 157
A212 158
A212 159
A212 160
A212 161
A212 162
A212 163
A212 164
A212 165
A212 166
A212 167
A212 168
A212 169
A212 170
A212 171
A212 172
A212 173
A212 174
A212 175
A212 176
A212 177
A212 178
A212 179
A212 180
A212 181
A212 182
A212 183
A212 184
A212 185
A212 186
A212 187
A212 188
A212 189
A212 190

```


651	ID(II)=1000	A212	286
	TYPE(II)=5HERROR	A212	287
650	CONTINUE	A212	288
	RETURN	A212	289
	END	A212	290
		A212	291

	SUBROUTINE A216	A216	1
C	THE PURPOSE OF THIS SUBROUTINE IS TO WRITE THE ARRAY LOADER (A200),	A216	2
C	THE ARRAY UNLOADER (A201), AND THE ARRAY INITIALIZATION (A217)	A216	3
C	SUBROUTINES	A216	4
	DIMENSION SUB(2), GO(13), FMT(5), ZERO(3), ARR(6), DOSTMT(2)	A216	5
	COMMON DUM1(16080), DUM2(8), MAX	A216	6
	INTEGER MAX	A216	7
	DATA SUB/10H SUBR,7HOUTINE /	A216	10
	DATA GO/2H ,2H ,2H ,2HGO,2HTO,2H(1,2H,2,2H,3,2H,4,2H,5,2H,6,2H)	A216	11
	1N,2HN /	A216	12
	DATA FMT/10H(* ,11,10H,* CALL A2,10H*,11,11/* ,10H RETURN,	A216	13
	110HN*) /	A216	14
	DATA ZERO/10H RETU,2HRN,10H END /	A216	15
	DATA ARR/2HYX,2HYY,2HAX,2HAY,2HBY,2HCY /	A216	16
	DATA DOSTMT/10H1,LENGTHY ,10HNI,NF /	A216	17
		A216	18
C	SET VARIABLES TO WRITE SUBROUTINE A200	A216	19
	SUBNAME=4HA200	A216	20
	KEY=2	A216	21
C	WRITE SUBROUTINES A200 AND A201	A216	22
	10 WRITE(19,20) SUB,SUBNAME	A216	23
	20 FORMAT(A10,A7,A4)	A216	24
	WRITE(19,21)	A216	25
	21 FORMAT(6X*COMMON NTELE,LENGTHY,LENGTHA,IREADLK*/6X	A216	26
	1*COMMON C1,C2,C3,C4*/6X	A216	27
	2*COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF*/6X	A216	28
	3*COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX*/6X	A216	29
	4*COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO*/	A216	30
	56X	A216	31
	6*COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	A216	32
	7*/6X	A216	33
	8*COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL*/6X	A216	34
	9*COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)*6X	A216	35
	A*COMMON LASTTTL(2,4)*6X	A216	36
	B*COMMON LOCK(2)*	A216	37
	IF(MAX.NE.0)GOTO50	A216	38
	WRITE(19,40)ZERO	A216	39
40	FORMAT(A10,A2/A9)	A216	40
	IF(KEY.EQ.2)GOTO80	A216	41
	IF(KEY.EQ.3)GOTO100	A216	42
	RETURN	A216	43
50	IF(KEY.EQ.1)GOTO110	A216	44
	WRITE(19,52)	A216	45
52	FORMAT(* IF(TYPE(II).NE.5HARRAY)RETURN*)	A216	46
	IF(MAX.EQ.1)GOTO65	A216	47
	WRITE(19,55)	A216	48
55	FORMAT(* NN=10(II)*	A216	49
	LIM=MAX+5	A216	50
	WRITE(19,60)(GO(I),I=1,LIM),GO(12),GO(13)	A216	51
60	FORMAT(13A2)	A216	52
65	DO 70 J=1,MAX	A216	53
70	WRITE(19,FMT)J,KEY,J	A216	54
	WRITE(19,40)ZERO(3)	A216	55
	IF(KEY.EQ.3)GOTO100	A216	56
C	SET VARIABLES TO WRITE SUBROUTINE A201	A216	57
80	SUBNAME=4HA201	A216	58
	KEY=3	A216	59
	GOTO10	A216	60
C	SET VARIABLES TO WRITE SUBROUTINE A217	A216	61
100	SUBNAME=4HA217	A216	62
	KEY=1	A216	63
	GOTO10	A216	64
C	WRITE SUBROUTINE A217	A216	65
110	LIM=1	A216	66
	IF(MAX.GE.2)LIM=2	A216	67
	DO 112 I=1,LIM	A216	68
112	WRITE(19,113)ARR(I)	A216	69
116	FORMAT(* COMMON *A2*(6000)*	A216	70
113	FORMAT(* COMMON *A2*(6000)*	COM1	1
	IF(MAX.LE.2)GOTO119	COM1	2
	DO 115 I=3,MAX	A216	72
115	WRITE(19,116)ARR(I)	A216	73
119	LABEL=1	A216	74
	WRITE(19,120)LABEL,DOSTMT(LABEL)	A216	75
120	FORMAT(* DO*,11,*I=*,A10)	A216	76
	DO 130 I=1,LIM	A216	77
130	WRITE(19,140)ARR(I)	A216	78
140	FORMAT(* *,A2,*I)=0.*)	A216	79
	WRITE(19,150)LABEL	A216	80
150	FORMAT(* *,11,* CONTINUE*)	A216	81
	IF(MAX.LE.2)GOTO180	A216	82
		A216	83

	LABEL=2	A216	84
	WRITE(19,120) LABEL,DOSTMT(LABEL)	A216	85
	DO 170 I=3,MAX	A216	86
170	WRITE(19,140) ARR(I)	A216	87
	WRITE(19,150) LABEL	A216	88
180	WRITE(19,40) ZERO	A216	89
	RETURN	A216	90
	END	A216	91

	SUBROUTINE A319	A319	1
C	SETS WORD, ID AND TYPE AND CHECKS FOR ERRONEOUS ARGUMENTS	A319	2
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A319	3
	COMMON C1,C2,C3,C4	A319	4
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	3
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
	COMMON LASTTTL(2,4)	BLOC	7
	COMMON LOCK(2)	BLOC	8
		BLOC	9
		BLOC	10
	REAL C1,C2,C3,C4	SPAR	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A319	7
	REAL LASTTTL	A319	8
		REAL	1
		REAL	2
		REAL	3
		A319	10
		A319	11
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
	INTEGER LIMIT,ID,II,JJ,LL	INTG	3
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
		A319	13
	DATA KEY/1/	A319	14
		A319	15
		A319	16
		A319	17
	IF(KEY.EQ.1) READ(18,5) NTELE	A319	18
	5 FORMAT(I2)	A319	19
	CALL DATE(ZZ)	A319	20
	IF(KEY.EQ.1) WRITE(11,1) ZZ	A319	21
	IF((KEY.EQ.1).AND.(NTELE.EQ.1)) WRITE(12,1) ZZ	A319	22
	1 FORMAT(*1*13X,*PROGRAM EXECUTION*,10X,*DATE * A10//)	A319	23
	KEY=0	A319	24
C	READ INSTRUCTION, TYPES, AND IDS. PRINT THE INSTRUCTION	A319	25
	READ(18,10) WORD(1,1),TYPE1,ID1,(WORD(1,1),I=2,3),TYPE3,ID3,(WORD(1,1),J=4,8)	A319	26
	10 FORMAT(2A10,I10,3A10,I10,E15.6,4A10)	A319	27
	WRITE(11,20) (WORD(I,1),I=1,8)	A319	28
	20 FORMAT(* \$\$\$*8X,A2,2A10,E15.6,4A10,8X,* \$\$\$*)	A319	29
	IF(NTELE.EQ.1) WRITE(12,30) (WORD(1,1),I=1,8)	A319	30
	30 FORMAT(1X* * ,A2,2A10,E14.6,4A10,* * *)	A319	31
C	CHECK IDS, TYPES, AND OPERATION	A319	32
	ID(1)=ID1	A319	33
	TYPE(1)=TYPE1	A319	34
	IF(ID1.EQ.1000) GOTO50	A319	35
	40 ID(3)=ID3	A319	36
	TYPE(3)=TYPE3	A319	37
	IF(ID3.EQ.1000) GOTO70	A319	38
	45 IF(WORD(2,1).EQ.4HA308) GOTO90	A319	39
	RETURN	A319	40
C	DIAGNOSTICS FOR ILLEGAL ARGUMENT OR OPERATION	A319	41
	50 WRITE(11,60)	A319	42
	IF(NTELE.EQ.1) WRITE(12,60)	A319	43
	60 FORMAT(53H *****ILLEGAL FIRST ARGUMENT-EXECUTION CONTINUED*****)	A319	44
	GOTO40	A319	45
	70 WRITE(11,80)	A319	46
	IF(NTELE.EQ.1) WRITE(12,80)	A319	47
	80 FORMAT(54H *****ILLEGAL SECOND ARGUMENT-EXECUTION CONTINUED*****)	A319	48
	GOTO45	A319	49
	90 WRITE(11,100)	A319	50
	IF(NTELE.EQ.1) WRITE(12,100)	A319	51
	100 FORMAT(* ///ILLEGAL MODULE MODULE SHOWN IN OBJECT PROG. LISTING*//)	A319	52
	1* REPLACED BY A308. COMMON DUMPED. OPERATION SKIPPED ///*)	A319	53
	RETURN	A319	54
		A319	55
		A319	56
		A319	57
	ENTRY A318	A319	58
	WORD(4,1)=1.0	A319	59
CLIST	TABLE OF CONTENTS OF ANY TAPES WRITTEN	A319	60
	WRITE(11,570)	A319	61
	IF(NTELE.EQ.1) WRITE(12,570)	A319	62
570	FORMAT(*TAPE CONTENTS LISTINGS*)	A319	63
	TAPENO=1	A319	64
	DISKB=3	A319	65
	CALL A399	A319	66
	TAPENO=2	A319	67
	DISKB=4	A319	68

CALL A399
RETURN
END

A319 69
A319 70
A319 71

C SUBROUTINE A215
COMMON INITIALIZATION

COMMON NTELE, LENGTHY, LENGTHA, IREADLK
COMMON C1, C2, C3, C4
COMMON NY, NA, NB, NC, YI, YF, AI, AF, BI, BF, CI, CF, NI, NF
COMMON XLIMUP, XLIMLR, YLIMUP, YLIMLR, EMIN, EMAX
COMMON NSTEP, KEY100, KEY101, LINE100, LINE101, FAULTLT, DUMPLT, PRNTO
COMMON TITLE(4), COMMENT(4), TITLEA(4), TITLES(4), TITLFC(4), DUMMY(4)
COMMON WORD(8,1), ELEMENT(4), LIMIT(4), TYPE(3), IO(3), II, JJ, LL
COMMON NTITLE(2), NPTSBLK(2), TAPENO, DISKA, DISKB, NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

REAL C1, C2, C3, C4
REAL XLIMUP, XLIMLR, YLIMUP, YLIMLR, EMIN, EMAX, ELEMENT
REAL LASTTTL

INTEGER NY, NA, NB, NC, YI, YF, AI, AF, BI, BF, CI, CF, NI, NF
INTEGER NSTEP, KEY100, KEY101, LINE100, LINE101, FAULTLT, DUMPLT, PRNTO
INTEGER LIMIT, IO, II, JJ, LL
INTEGER NTITLE, NPTSBLK, TAPENO, DISKA, DISKB, NFILE

REWIND 10

C1=0.
C2=0.
C3=0.
C4=0.
NY=1
NA=1
NB=1
NC=1
YI=1
YF=1
AI=1
AF=1
BI=1
BF=1
CI=1
CF=1
NI=1
XLIMUP=0.
XLIMLR=0.
YLIMUP=0.
YLIMLR=0.
EMIN=0.
EMAX=0.
NSTEP=1
KEY100=0
KEY101=0
LINE100=0
LINE101=0
FAULTLT=0
DUMPLT=1
PRNTO=0
TAPENO=0
DISKA=0
DISKB=0
LENGTHY=6000
LENGTHA=6000
IREADLK=0
NTELE=0
NF=LENGTHA

DO1323 I=1,2
LOCK(I)=0
NTITLE(I)=0
NFILE(I)=0
1323 NPTSBLK(I)=0

DO101 I=1,4
TITLE(I)=10H
COMMENT(I)=10H
TITLEA(I)=10H
TITLES(I)=10H
TITLFC(I)=10H
DUMMY(I)=10H
ELEMENT(I)=0.

A215 1
A215 2
A215 3
A215 4
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
A215 7
A215 8
REAL 1
REAL 2
REAL 3
A215 10
A215 11
INTG 1
INTG 2
INTG 3
INTG 4
A215 13
A215 14
A215 15
A215 16
A215 17
A215 18
A215 19
A215 20
A215 21
A215 22
A215 23
A215 24
A215 25
A215 26
A215 27
A215 28
A215 29
A215 30
A215 31
A215 32
A215 33
A215 34
A215 35
A215 36
A215 37
A215 38
A215 39
A215 40
A215 41
A215 42
A215 43
A215 44
A215 45
A215 46
A215 47
A215 48
A215 49
A215 50
A215 51
COM4 1
COM4 2
A215 53
A215 54
A215 55
A215 56
A215 57
A215 58
A215 59
A215 60
A215 61
A215 62
A215 63
A215 64
A215 65
A215 66
A215 67
A215 68
A215 69
A215 70
A215 71
A215 72

10	LIMIT(I)=0	A215	73
		A215	74
	DO12=1,8	A215	75
12	WORD(I,I)=10H	A215	76
		A215	77
	DO14=1,3	A215	78
	TYPE(I)=10H	A215	79
14	ID(I)=0	A215	80
		A215	81
		A215	82
		A215	83
	II=1	A215	84
	JJ=1	A215	85
	LL=1	A215	86
		A215	87
		A215	88
	DO500K=1,2	A215	89
	DO500I=1,4	A215	90
500	LASTTTL(K,I)=10H	A215	91
	RETURN	A215	92
	END	A215	93
		A215	94

C	SUBROUTINE A202	A202	1
	CONSTANT LOADER	A202	2
		A202	3
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A202	4
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
	COMMON NSTEP,KEY100,KEY101,LIN100,LIN101,FAULTLT,DUMPLT,PRNO	BLOC	4
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	COMMON WORD(8,I),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
	COMMON LASTTTL(2,4)	BLOC	8
	COMMON LOCK(2)	BLOC	9
		SPAR	10
	REAL C1,C2,C3,C4	A202	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A202	7
	REAL LASTTTL	A202	8
		REAL	1
		REAL	2
		REAL	3
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A202	10
	INTEGER NSTEP,KEY100,KEY101,LIN100,LIN101,FAULTLT,DUMPLT,PRNO	A202	11
	INTEGER LIMIT,ID,II,JJ,LL	INTG	1
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	2
		INTG	3
		INTG	4
	IF (TYPE(II).NE.8HCONSTANT) RETURN	A202	13
	NN=ID(II)-6	A202	14
	GOTO(13,14,15,16)NN	A202	15
13	ELEMENT(II)=C1	A202	16
	RETURN	A202	17
14	ELEMENT(II)=C2	A202	18
	RETURN	A202	19
15	ELEMENT(II)=C3	A202	20
	RETURN	A202	21
16	ELEMENT(II)=C4	A202	22
	RETURN	A202	23
		A202	24
		A202	25
		A202	26
		A202	27
		A202	28
C	ENTRY A203	A202	29
	CONSTANT UNLOADER	A202	30
	IF (TYPE(II).NE.8HCONSTANT) GOTO9000	A202	31
	NN=ID(II)-6	A202	32
	GOTO(17,18,19,20)NN	A202	33
17	C1=ELEMENT(II)	A202	34
	RETURN	A202	35
18	C2=ELEMENT(II)	A202	36
	RETURN	A202	37
19	C3=ELEMENT(II)	A202	38
	RETURN	A202	39
20	C4=ELEMENT(II)	A202	40
	RETURN	A202	41
		A202	42
		A202	43
C	ENTRY A204	A202	44
	LIMIT LOADER	A202	45
	IF (TYPE(II).NE.9HDOOLOPLMT) GOTO9000	A202	46
	NN=ID(II)-10	A202	47
	GOTO(21,22,23,24,25,26,27,28,29,30,31,32,33,34)NN	A202	48
21	LIMIT(II)=NY	A202	49
	RETURN	A202	50
22	LIMIT(II)=NA	A202	51
	RETURN	A202	52
23	LIMIT(II)=NB	A202	53
	RETURN	A202	54


```

24 LIMIT(11)=NC
25 RETURN
26 LIMIT(11)=YI
27 LIMIT(11)=YF
28 LIMIT(11)=AI
29 LIMIT(11)=AF
30 LIMIT(11)=BI
31 LIMIT(11)=BF
32 LIMIT(11)=CI
33 LIMIT(11)=CF
34 LIMIT(11)=NI
35 LIMIT(11)=NF
36 RETURN
37 LIMIT(11)=NF
38 RETURN
39 LIMIT(11)=NF
40 RETURN
41 LIMIT(11)=NF
42 RETURN
43 LIMIT(11)=NF
44 RETURN
45 LIMIT(11)=NF
46 RETURN
47 LIMIT(11)=NF
48 RETURN

```

```

C
ENTRY A205      LIMIT UNLOADER
IF (TYPE(11).NE.9)HD0LOPLMT)GOTO9000
NN=10(11)-10
GOTO(35,36,37,38,39,40,41,42,43,44,45,46,47,48)NN
35 N=LIMIT(11)
36 N=LIMIT(11)
37 N=LIMIT(11)
38 N=LIMIT(11)
39 YI=LIMIT(11)
40 YF=LIMIT(11)
41 AI=LIMIT(11)
42 AI=LIMIT(11)
43 AI=LIMIT(11)
44 BE=LIMIT(11)
45 CI=LIMIT(11)
46 CE=LIMIT(11)
47 NI=LIMIT(11)
48 N=LIMIT(11)
RETURN

```

```

C
ENTRY A206      PARAMETER LOADER
IF (TYPE(11).NE.9)HPPARAMETER)GOTO9000
NN=10(11)-24
GOTO(49,50,51,52,53,54)NN
49 ELEMENT(11)=XLIMUP
50 RETURN
51 ELEMENT(11)=XLIML
52 RETURN
53 ELEMENT(11)=YIMUP
54 RETURN
55 ELEMENT(11)=YIML
56 RETURN
57 ELEMENT(11)=YIML
58 RETURN
59 ELEMENT(11)=YIML
60 RETURN

```

```

C
ENTRY A207      PARAMETER UNLOADER
IF (TYPE(11).NE.9)HPPARAMETER)GOTO9000
NN=10(11)-24
GOTO(55,56,57,58,59,60)NN
55 XLIMUP=ELEMENT(11)
56 XLIML=ELEMENT(11)
57 YIMUP=ELEMENT(11)
58 YIML=ELEMENT(11)
59 RETURN
60 RETURN

```

```

A202 56
A202 57
A202 58
A202 59
A202 60
A202 61
A202 62
A202 63
A202 64
A202 65
A202 66
A202 67
A202 68
A202 69
A202 70
A202 71
A202 72
A202 73
A202 74
A202 75
A202 76
A202 77
A202 78
A202 79
A202 80
A202 81
A202 82
A202 83
A202 84
A202 85
A202 86
A202 87
A202 88
A202 89
A202 90
A202 91
A202 92
A202 93
A202 94
A202 95
A202 96
A202 97
A202 98
A202 99
A202 100
A202 101
A202 102
A202 103
A202 104
A202 105
A202 106
A202 107
A202 108
A202 109
A202 110
A202 111
A202 112
A202 113
A202 114
A202 115
A202 116
A202 117
A202 118
A202 119
A202 120
A202 121
A202 122
A202 123
A202 124
A202 125
A202 126
A202 127
A202 128
A202 129
A202 130
A202 131
A202 132
A202 133
A202 134
A202 135
A202 136
A202 137
A202 138
A202 139
A202 140
A202 141
A202 142
A202 143
A202 144
A202 145
A202 146
A202 147
A202 148
A202 149

```

```

C      ENTRY A208
      NUMBER LOADER
      IF (TYPE(II).NE.6HNUMBER)GOTO9000
      NN=ID(II)-30
      GO TO(61,62,63,64,65,66,67,68)NN
61     LIMIT(II)=NSTEP
      RETURN
62     LIMIT(II)=KEY100
      RETURN
63     LIMIT(II)=KEY101
      RETURN
64     LIMIT(II)=LINE100
      RETURN
65     LIMIT(II)=LINE101
      RETURN
66     LIMIT(II)=FAULTLT
      RETURN
67     LIMIT(II)=DUMPLT
      RETURN
68     LIMIT(II)=PRINO
      RETURN

C      ENTRY A209
      NUMBER UNLOADER
      IF (TYPE(II).NE.6HNUMBER)GOTO9000
      NN=ID(II)-30
      GOTO(69,70,71,72,73,74,75,76)NN
69     NSTEP=LIMIT(II)
      RETURN
70     KEY100=LIMIT(II)
      RETURN
71     KEY101=LIMIT(II)
      RETURN
72     LINE100=LIMIT(II)
      RETURN
73     LINE101=LIMIT(II)
      RETURN
74     FAULTLT=LIMIT(II)
      RETURN
75     DUMPLT=LIMIT(II)
      RETURN
76     PRINO=LIMIT(II)
      RETURN

C      ENTRY A210
      AFORMAT LOADER
      IF (TYPE(II).NE.7HAFORMAT)GOTO9000
      NN=ID(II)-38
      GOTO(77,78,79,80,81,82)NN
77     DO1077I=1,4
1077    ELEMENT(I)=TITLE(I)
      RETURN
78     DO1078I=1,4
1078    ELEMENT(I)=COMMENT(I)
      RETURN
79     DO1079I=1,4
1079    ELEMENT(I)=TITLEA(I)
      RETURN
80     DO1080I=1,4
1080    ELEMENT(I)=TITLEB(I)
      RETURN
81     DO1081I=1,4
1081    ELEMENT(I)=TITLEC(I)
      RETURN
82     DO1082I=1,4
1082    ELEMENT(I)=DUMMY(I)
      RETURN

C      ENTRY A211
      AFORMAT UNLOADER
      IF (TYPE(II).NE.7HAFORMAT)GOTO9000
      NN=ID(II)-38
      GOTO(83,84,85,86,87,88)NN
83     DO1083I=1,4
1083    TITLE(I)=ELEMENT(I)
      RETURN
84     DO1084I=1,4
1084    COMMENT(I)=ELEMENT(I)
      RETURN
85     DO1085I=1,4
1085    TITLEA(I)=ELEMENT(I)
      RETURN
86     DO1086I=1,4
1086    TITLEB(I)=ELEMENT(I)
      RETURN
87     DO1087I=1,4
1087    TITLEC(I)=ELEMENT(I)
      RETURN
88     DO1088I=1,4
1088    DUMMY(I)=ELEMENT(I)

```

```

A202 150
A202 151
A202 152
A202 153
A202 154
A202 155
A202 156
A202 157
A202 158
A202 159
A202 160
A202 161
A202 162
A202 163
A202 164
A202 165
A202 166
A202 167
A202 168
A202 169
A202 170
A202 171
A202 172
A202 173
A202 174
A202 175
A202 176
A202 177
A202 178
A202 179
A202 180
A202 181
A202 182
A202 183
A202 184
A202 185
A202 186
A202 187
A202 188
A202 189
A202 190
A202 191
A202 192
A202 193
A202 194
A202 195
A202 196
A202 197
A202 198
A202 199
A202 200
A202 201
A202 202
A202 203
A202 204
A202 205
A202 206
A202 207
A202 208
A202 209
A202 210
A202 211
A202 212
A202 213
A202 214
A202 215
A202 216
A202 217
A202 218
A202 219
A202 220
A202 221
A202 222
A202 223
A202 224
A202 225
A202 226
A202 227
A202 228
A202 229
A202 230
A202 231
A202 232
A202 233
A202 234
A202 235
A202 236
A202 237
A202 238
A202 239
A202 240
A202 241
A202 242
A202 243
A202 244

```

RETURN	A202 245
	A202 246
	A202 247
C ENTRY A214	A202 248
BLANK LOADER UNLOADER	A202 249
IF(TYPE(11).NE.5HBLANK)GOTO9000	A202 250
RETURN	A202 251
	A202 252
	A202 253
C ENTRY A213	A202 254
STAR LOADER UNLOADER	A202 255
IF(TYPE(11).NE.4HSTAR)GOTO9000	A202 256
RETURN	A202 257
9000 WRITE(11,9001)	A202 258
IF(NTELE.EQ.1)WRITE(12,9001)	A202 259
9001 FORMAT(* FAULT AT LOAD,UNLOADER *)	A202 260
CALLA308	A202 261
RETURN	A202 262
END	A202 263

SUBROUTINE A221	A221 1
	A221 2
C YX ARRAY LOADER	A221 3
	A221 4
	A221 5
	A221 6
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC 1
COMMON C1,C2,C3,C4	BLOC 2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC 3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC 4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC 5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC 6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC 7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC 8
COMMON LASTTTT(2,4)	BLOC 9
COMMON LOCK(2)	BLOC 10
COMMON YX(6000)	SPAR 1
	COMY 1
REAL C1,C2,C3,C4	A221 10
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A221 11
REAL LASTTTT	REAL 1
	REAL 2
	REAL 3
	A221 13
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A221 14
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG 1
INTEGER LIMIT,ID,II,JJ,LL	INTG 2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG 3
	INTG 4
	A221 16
ELEMENT(II)=YX(LL)	A221 17
RETURN	A221 18
END	A221 19
	A221 20

SUBROUTINE A222	A222 1
	A222 2
C YY ARRAY LOADER	A222 3
	A222 4
	A222 5
	A222 6
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC 1
COMMON C1,C2,C3,C4	BLOC 2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC 3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC 4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC 5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC 6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC 7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC 8
COMMON LASTTTT(2,4)	BLOC 9
COMMON LOCK(2)	BLOC 10
COMMON YX(6000)	SPAR 1
COMMON YY(6000)	COMY 1
	COMY 1
REAL C1,C2,C3,C4	A222 11
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A222 12
REAL LASTTTT	REAL 1
	REAL 2
	REAL 3
	A222 14
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A222 15
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG 1
INTEGER LIMIT,ID,II,JJ,LL	INTG 2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG 3
	INTG 4
	A222 17
	A222 18

ELEMENT(II)=YY(LL)	A222	19
RETURN	A222	20
END	A222	21
SUBROUTINE A223	A223	1
C AX ARRAY LOADER	A223	2
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A223	3
COMMON C1,C2,C3,C4	A223	4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A223	5
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	A223	6
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	1
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	2
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	3
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	4
COMMON LASTTTL(2,4)	BLOC	5
COMMON LOCK(2)	BLOC	6
COMMON YX(6000)	BLOC	7
COMMON YY(6000)	BLOC	8
COMMON AX(6000)	BLOC	9
	SPAR	10
REAL C1,C2,C3,C4	COMY	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	COMY	1
REAL LASTTTL	COMA	1
	A223	12
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A223	13
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	REAL	1
INTEGER LIMIT,ID,II,JJ,LL	REAL	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	REAL	3
	A223	15
ELEMENT(II)=AX(LL)	A223	16
RETURN	INTG	1
END	INTG	2
	INTG	3
	INTG	4
	A223	18
	A223	19
	A223	20
	A223	21
	A223	22
SUBROUTINE A224	A224	1
C AY ARRAY LOADER	A224	2
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A224	3
COMMON C1,C2,C3,C4	A224	4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A224	5
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	A224	6
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	1
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	2
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	3
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	4
COMMON LASTTTL(2,4)	BLOC	5
COMMON LOCK(2)	BLOC	6
COMMON YX(6000)	BLOC	7
COMMON YY(6000)	BLOC	8
COMMON AX(6000)	BLOC	9
COMMON AY(6000)	BLOC	10
	SPAR	1
REAL C1,C2,C3,C4	COMY	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	COMY	1
REAL LASTTTL	COMA	1
	COMA	1
	A224	13
	A224	14
	REAL	1
	REAL	2
	REAL	3
	A224	16
	A224	17
	INTG	1
	INTG	2
	INTG	3
	INTG	4
	A224	19
	A224	20
	A224	21
	A224	22
	A224	23
SUBROUTINE A225	A225	1
C BY ARRAY LOADER	A225	2
	A225	3
	A225	4

COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A225	5
COMMON C1,C2,C3,C4	A225	6
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	3
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
COMMON LASTTTL(2,4)	BLOC	7
COMMON LOCK(2)	BLOC	8
	BLOC	9
COMMON YX(6000)	SPAR	10
COMMON YY(6000)	COMH	1
COMMON AX(6000)	COMH	1
COMMON AY(6000)	COMA	1
COMMON BY(6000)	COMA	1
	COMB	1
	A225	14
REAL C1,C2,C3,C4	A225	15
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
REAL LASTTTL	REAL	2
	REAL	3
	A225	17
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A225	18
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
	INTG	4
	A225	20
ELEMENT(II)=BY(LL)	A225	21
RETURN	A225	22
END	A225	23
	A225	24
SUBROUTINE A226	A226	1
	A226	2
C CY ARRAY LOADER	A226	3
	A226	4
	A226	5
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A226	6
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
COMMON LASTTTL(2,4)	BLOC	8
COMMON LOCK(2)	BLOC	9
	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMH	1
COMMON AX(6000)	COMH	1
COMMON AY(6000)	COMA	1
COMMON BY(6000)	COMA	1
COMMON CY(6000)	COMB	1
	COMC	1
	A226	15
REAL C1,C2,C3,C4	A226	16
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
REAL LASTTTL	REAL	2
	REAL	3
	A226	18
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A226	19
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
	INTG	4
	A226	21
ELEMENT(II)=CY(LL)	A226	22
RETURN	A226	23
END	A226	24
	A226	25
SUBROUTINE A231	A231	1
	A231	2
C YX ARRAY UNLOADER	A231	3
	A231	4
	A231	5
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A231	6
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	BLOC	6

```

COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

YX(II)=ELEMENT(II)
RETURN
END

```

```

BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
COMY 1
A231 10
A231 11
REAL 1
REAL 2
REAL 3
A231 13
A231 14
INTG 1
INTG 2
INTG 3
INTG 4
A231 16
A231 17
A231 18
A231 19
A231 20

```

SUBROUTINE A232

C YY ARRAY UNLOADER

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)
COMMON YY(6000)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

YY(II)=ELEMENT(II)
RETURN
END

```

```

A232 1
A232 2
A232 3
A232 4
A232 5
A232 6
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
COMY 1
COMY 1
A232 11
A232 12
REAL 1
REAL 2
REAL 3
A232 14
A232 15
INTG 1
INTG 2
INTG 3
INTG 4
A232 17
A232 18
A232 19
A232 20
A232 21

```

SUBROUTINE A233

C AX ARRAY UNLOADER

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)
COMMON YY(6000)
COMMON AX(6000)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF

```

```

A233 1
A233 2
A233 3
A233 4
A233 5
A233 6
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
COMY 1
COMY 1
COMA 1
A233 12
A233 13
REAL 1
REAL 2
REAL 3
A233 15
A233 16
INTG 1

```

INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A233	18
	A233	19
	A233	20
	A233	21
	A233	22
AX(LL)=ELEMENT(II)		
RETURN		
END		
SUBROUTINE A234	A234	1
	A234	2
C AY ARRAY UNLOADER	A234	3
	A234	4
	A234	5
	A234	6
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LOCK(2)	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMY	1
COMMON AX(6000)	COMY	1
COMMON AY(6000)	COMA	1
	COMA	1
	A234	13
	A234	14
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A234	16
	A234	17
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A234	19
	A234	20
	A234	21
	A234	22
	A234	23
AY(LL)=ELEMENT(II)		
RETURN		
END		
SUBROUTINE A235	A235	1
	A235	2
C BY ARRAY UNLOADER	A235	3
	A235	4
	A235	5
	A235	6
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LOCK(2)	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMY	1
COMMON AX(6000)	COMY	1
COMMON AY(6000)	COMA	1
COMMON BY(6000)	COMB	1
	A235	14
	A235	15
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A235	17
	A235	18
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A235	20
	A235	21
	A235	22
	A235	23
	A235	24
BY(LL)=ELEMENT(II)		
RETURN		
END		

SUBROUTINE A236		A236	1
C	CY ARRAY UNLOADER	A236	2
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A236	3
	COMMON C1,C2,C3,C4	A236	4
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	A236	5
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	A236	6
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC	1
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	2
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	3
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	4
	COMMON LASTTTL(2,4)	BLOC	5
	COMMON LOCK(2)	BLOC	6
		BLOC	7
	COMMON YX(6000)	BLOC	8
	COMMON YY(6000)	BLOC	9
	COMMON AX(6000)	BLOC	10
	COMMON AY(6000)	SPAR	1
	COMMON BY(6000)	COMY	1
	COMMON CY(6000)	COMY	1
		COMA	1
		COMA	1
		COMR	1
		COMC	1
		A236	15
	REAL C1,C2,C3,C4	A236	16
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
	REAL LASTTTL	REAL	2
		REAL	3
		A236	18
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	A236	19
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG	1
	INTEGER LIMIT,ID,II,JJ,LL	INTG	2
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
		INTG	4
		A236	21
	CY(LL)=ELEMENT(11)	A236	22
	RETURN	A236	23
	END	A236	24
		A236	25
SUBROUTINE A315		A315	1
C	READ INSTRUCTION	A315	2
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A315	3
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	BLOC	2
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC	4
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
	COMMON LASTTTL(2,4)	BLOC	8
	COMMON LOCK(2)	BLOC	9
		BLOC	10
		SPAR	1
		A315	6
		A315	7
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTTTL	REAL	3
		A315	9
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	A315	10
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG	1
	INTEGER LIMIT,ID,II,JJ,LL	INTG	2
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
		INTG	4
		A315	12
	II=3	A315	13
C	READ A CONSTANT	A315	14
	IF(TYPE(11).NE.8HCONSTANT)GOTO20	A315	15
	ELEMENT(11)=WORD(4,JJ)	A315	16
	CALLA203	A315	17
	GOTO650	A315	18
		A315	19
		A315	20
C	READ A DO LOOP LIMIT	A315	21
	20 IF(TYPE(11).NE.9HDOLOOPMT)GOTO30	A315	22
	LIMIT(11)=IF(X(WORD(4,JJ)))	A315	23
	CALLA205	A315	24
	GOTO650	A315	25
		A315	26
		A315	27
C	READ A PARAMETER	A315	28
	30 IF(TYPE(13).NE.9HPARAMETER)GOTO40	A315	29
	ELEMENT(11)=WORD(4,JJ)	A315	30
	CALL A207	A315	31
	GOTO650	A315	32
		A315	33
		A315	34
		A315	35
C	READ A NUMBER	A315	36
	40 IF(TYPE(11).NE.6HNUMBER)GOTO50	A315	37

LIMIT(11)=IFX(WORD(4,JJ))	A315	38
CALLA209	A315	39
GOTO650	A315	40
C READ COMMENT FIELD	A315	41
50 IF(TYPE(11).NE.7HAFORMAT)GOTO51	A315	42
DO1051(=1)4	A315	43
1051 ELEMENT(1)=WORD(4+1,JJ)	A315	44
CALLA211	A315	45
GO TO 650	A315	46
	A315	47
	A315	48
	A315	49
C READ DATA CARDS	A315	50
51 IF(TYPE(11).NE.5HBLANK)GO TO 607	A315	51
CALL TANDCRD	A315	52
IF(DUMMY(3).NE.10HTABLE READ)GOTO607	A315	53
CALL CHECKER	A315	54
IF(DUMMY(1).NE.8HTABLE OK)GOTO607	A315	55
1515 WRITE(11,1516)	A315	56
IF(INTELE.EQ.1)WRITE(12,1516)	A315	57
1516 FORMAT(* ARRAYS READ*)	A315	58
650 RETURN	A315	59
607 WRITE(11,609)	A315	60
IF(INTELE.EQ.1)WRITE(12,609)	A315	61
609 FORMAT(* ILLEGAL SECOND ARGUMENT OR PROBLEMS WITH *	A315	62
1/* DATA CARD INPUT---MODULE PARTIALLY EXECUTED*)	A315	63
GOTO650	A315	64
END	A315	65
	A315	66
SUBROUTINE A316	A316	1
C PRINT INSTRUCTION	A316	2
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A316	3
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	BLOC	2
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
COMMON NTITLE,NPTSBLK(2),TAPEND,DISKA,DISKB,NFILE(2)	BLOC	7
COMMON LASTTTL(2,4)	BLOC	8
COMMON LOCK(2)	BLOC	9
COMMON YX(6000)	BLOC	10
COMMON YY(6000)	SPAR	1
	COMY	1
	COMY	1
	A316	8
REAL C1,C2,C3,C4	A316	9
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
REAL LASTTTL	REAL	2
	REAL	3
	A316	11
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF	A316	12
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPEND,DISKA,DISKB,NFILE	INTG	3
	INTG	4
II=3	A316	14
	A316	15
	A316	16
	A316	17
	A316	18
C PRINT A CONSTANT	A316	19
IF(TYPE(11).NE.8HCONSTANT)GOTO20	A316	20
CALL A202	A316	21
WRITE(11,1020)WORD(3,JJ),ELEMENT(11)	A316	22
IF(INTELE.EQ.1)WRITE(12,1020)WORD(3,JJ),ELEMENT(11)	A316	23
1020 FORMAT(*O*A10,*=*E15.5)	A316	24
GOTO650	A316	25
	A316	26
	A316	27
C PRINT A DO LOOP LIMIT	A316	28
20 IF(TYPE(11).NE.9HDOLOOPLMT)GOTO30	A316	29
CALL A204	A316	30
WRITE(11,1030)WORD(3,JJ),LIMIT(11)	A316	31
IF(INTELE.EQ.1)WRITE(12,1030)WORD(3,JJ),LIMIT(11)	A316	32
1030 FORMAT(*O*A10,*=*E15)	A316	33
GOTO650	A316	34
	A316	35
	A316	36
	A316	37
C PRINT A PARAMETER	A316	38
30 IF(TYPE(11).NE.9HPARAMETER)GOTO40	A316	39
CALL A206	A316	40
WRITE(11,1040)WORD(3,JJ),ELEMENT(11)	A316	41
IF(INTELE.EQ.1)WRITE(12,1040)WORD(3,JJ),ELEMENT(11)	A316	42
1040 FORMAT(*O*A10,*=*E15.5)	A316	43
GOTO650	A316	44
	A316	45
	A316	46
C PRINT A NUMBER	A316	47
40 IF(TYPE(11).NE.6HNUMBER)GOTO50	A316	47

CALL A208	A316	48
WRITE(11,1150)WORD(3,JJ),LIMIT(11)	A316	49
IF(NTELE.EQ.1)WRITE(12,1150)WORD(3,JJ),LIMIT(11)	A316	50
1150 FORMAT(*0*A10,*,*15)	A316	51
GOTO650	A316	52
C PRINT A TITLE	A316	53
50 IF(TYPE(11).NE.7)HAFORMAT)GOTO51	A316	54
CALL A210	A316	55
WRITE(11,1051)WORD(3,JJ),(ELEMENT(K),K=1,4)	A316	56
IF(NTELE.EQ.1)WRITE(12,1051)WORD(3,JJ),(ELEMENT(K),K=1,4)	A316	57
1051 FORMAT(*0*A10,*,*4A10)	A316	58
GOTO650	A316	59
C PRINT YX AND YY BY INTERVAL PRNO	A316	60
51 CONTINUE	A316	61
IF(PRNO.EQ.0)CALL SELECTR	A316	62
WRITE(11,1052)YI,YF,PRNO,(YX(K),YY(K),K=YI,YF,PRNO)	A316	63
IF(NTELE.EQ.1)WRITE(12,900)YI,YF,PRNO,(YX(K),YY(K),K=YI,YF,PRNO)	A316	64
900 FORMAT(*0YI=*,I4,*,YF=*,I4,*,PRNO=*,I4,	A316	65
X* EACH LINE BELOW IS YX(I), YY(I) FOR I=YI TO YF BY PRNO*/	A316	66
X/(1H 6E12.5))	A316	67
1052 FORMAT(*0YI=*,I5,*,YF=*,I5,*,PRNO=*,	A316	68
115/1H *EACH LINE BELOW IS YX(I), YY(I)*	A316	69
1* FOR I=YI TO YF BY PRNO*/ /(1H 10E12.5))	A316	70
PRNO=0	A316	71
650 RETURN	A316	72
END	A316	73
	A316	74
	A316	75
	A316	76
	A316	77

C	SUBROUTINE A320	A320	1
	PRALLARRAY	A320	2
C	PRINT ALL ARRAYS OVER RANGE OF DIMENSIONS	A320	3
		A320	4
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A320	5
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC	4
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
	COMMON NTITLE(2),NPYSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
	COMMON LASTTTL(2,4)	BLOC	8
	COMMON LOCK(2)	BLOC	9
		BLOC	10
	COMMON YX(6000)	SPAR	1
	COMMON YY(6000)	COMY	1
	COMMON AX(6000)	COMA	1
	COMMON AY(6000)	COMA	1
	COMMON BY(6000)	COMB	1
	COMMON CY(6000)	COMC	1
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTTTL	REAL	3
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG	2
	INTEGER LIMIT,ID,II,JJ,LL	INTG	3
	INTEGER NTITLE,NPYSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
		A320	16
	ITEMP1=YI	A320	17
	ITEMP2=YF	A320	18
	YI=NI	A320	19
	YF=NF	A320	20
	WRITE(11,87)	A320	21
	IF(NTELE.EQ.1)WRITE(12,87)	A320	22
	IF(PRNO.EQ.0)CALLSELECTR	A320	23
	WRITE(11,86)(I,YX(I),YY(I),AX(I),AY(I),BY(I),CY(I),I=NI,NF,PRNO)	A320	24
	IF(NTELE.EQ.1)WRITE(12,86)(I,YX(I),YY(I),AX(I),AY(I),BY(I),CY(I),	A320	25
	XI=NI,NF,PRNO)	A320	26
	87 FORMAT(*1 PRINTING OF ALL ARRAYS OVER RANGE *	A320	27
	1*OF DIMENSIONS*/*0 PRINTED BELOW ARE *	A320	28
	2*YX(I), YY(I), AX(I), AY(I), BY(I), AND CY(I) *	A320	29
	3*FOR VALUES OF I SHOWN*////)	A320	30
	86 FORMAT(1H 16,6E12.5)	A320	31
	YI=ITEMP1	A320	32
	YF=ITEMP2	A320	33
	PRNO=0	A320	34
	RETURN	A320	35
	END	A320	36
		A320	37

C	SUBROUTINE A302	A302	1
	FUNCTION SUBROUTINE	A302	2
		A302	3
		A302	4

COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF	BLOC	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTD	BLOC	5
COMMON TTITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPEND,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LCKK(2)	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMY	1
	COMY	1
	A302	9
	A302	10
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A302	12
	A302	13
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTD	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPEND,DISKA,DISKB,NFILE	INTG	4
	A302	15
	A302	16
	A302	17
	A302	18
	A302	19
	A302	20
	A302	21
	A302	22
	A302	23
	A302	24
	A302	25
	A302	26
	A302	27
	A302	28
	A302	29
	A302	30
	A302	31
	A302	32
	A302	33
	A302	34
	A302	35
	A302	36
	A302	37
	A302	38
	A302	39
	A302	40
	A302	41
	A302	42
	A302	43
	A302	44
	A302	45
	A302	46
	A302	47
	A302	48
	A302	49
	A302	50
	A302	51
	A302	52
	A302	53
	A302	54
	A302	55
	A302	56
	A302	57
	A302	58
	A302	59
	A302	60
	A302	61
	A302	62
	A302	63
	A302	64
	A302	65
	A302	66
	A302	67
	A302	68
	A302	69
	A302	70
	A302	71
	A302	72
	A302	73
	A302	74
	A302	75
	A302	76
	A302	77
	A302	78
	A302	79
	A302	80
	A302	81
	A302	82
	A302	83
	A302	84
	A302	85


```

C      ADD
12 IF (TYPE(1).EQ.8HCONSTANT)10,12
10 IF (TYPE(1).EQ.5HARRAY)16,787
16 IF (TYPE(3).EQ.8HCONSTANT)18,14
14 IF (TYPE(3).EQ.5HARRAY)2,787
18 DO31I=1,3,2
CCONSTANT LOADER
3 CALL A202
C2=ELEMENT(1)+ELEMENT(3)
GOTO1580
2 DO100LL=YI,YF
DO102II=1,3,2
CARRAY LOADER
CALL A200
CCONSTANT LOADER
102 CALL A202
100 YY(LL)=ELEMENT(1)+ELEMENT(3)
GOTO1580
787 WRITE(11,788)
IF (NTELE.EQ.1)WRITE(12,788)
788 FORMAT(* ILLEGAL ARGUMENTS*)
GOTO1580

C      ENTRY A303
SUBTRACT
30 IF (TYPE(1).EQ.8HCONSTANT)32,30
32 IF (TYPE(1).EQ.5HARRAY)33,787
33 IF (TYPE(3).EQ.8HCONSTANT)36,34
34 IF (TYPE(3).EQ.5HARRAY)22,787
36 DO23II=1,3,2
CCONSTANT LOADER
23 CALL A202
C2=ELEMENT(1)-ELEMENT(3)
GOTO1580

22 DO25LL=YI,YF
DO26II=1,3,2
CARRAY LOADER
CALL A200
CCONSTANT LOADER
26 CALL A202
25 YY(LL)=ELEMENT(1)-ELEMENT(3)
GOTO1580

C      ENTRY A304
MULTIPLY
40 IF (TYPE(1).EQ.8HCONSTANT)42,40
42 IF (TYPE(1).EQ.5HARRAY)43,787
43 IF (TYPE(3).EQ.8HCONSTANT)46,44
44 IF (TYPE(3).EQ.5HARRAY)402,787
46 DO403II=1,3,2
CCONSTANT LOADER
403 CALL A202
C2=ELEMENT(1)*ELEMENT(3)
GOTO1580

402 DO405LL=YI,YF
DO406II=1,3,2

```

C	ARRAY LOADER	A302	86
	CALL A200	A302	87
	CCONSTANT LOADER	A302	88
	406 CALL A202	A302	89
	405 YY(II)=ELEMENT(1)*ELEMENT(3)	A302	90
	GOTO1580	A302	91
		A302	92
		A302	93
	ENTRY A305	A302	94
	DIVIDE	A302	95
	IF(TYPE(1).EQ.8HCONSTANT)52,50	A302	96
	50 IF(TYPE(1).EQ.5HARRAY)53,787	A302	97
	52 IF(TYPE(3).EQ.8HCONSTANT)56,54	A302	98
	53 IF(TYPE(3).EQ.8HCONSTANT)501,54	A302	99
	54 IF(TYPE(3).EQ.5HARRAY)501,787	A302	100
	56 D0503III=1,3,2	A302	101
	503 CALL A202	A302	102
	CDIVIDE BY ZERO PROTECTOR	A302	103
	CLOAD RESULT WITH -0. IF DIVIDE BY ZERO ENCOUNTERED	A302	104
	C2=-0.0	A302	105
	IF(ELEMENT(3).EQ.0.)WRITE(11,9050)	A302	106
	IF((ELEMENT(3).EQ.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A302	107
	9050 FORMAT(* ZERO OR NEGATIVE ARGUMENT ENCOUNTERED*)	A302	108
	IF(ELEMENT(3).NE.0.0)C2=ELEMENT(1)/ELEMENT(3)	A302	109
	GOTO1580	A302	110
	501 D0510LL=YI,YF	A302	111
	D0511II=1,3,2	A302	112
	CARRAY LOADER	A302	113
	CALL A200	A302	114
	CCONSTANT LOADER	A302	115
	511 CALL A202	A302	116
	YY(LL)=-0.0	A302	117
	IF(ELEMENT(3).EQ.0.)WRITE(11,9050)	A302	118
	IF((ELEMENT(3).EQ.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A302	119
	IF(ELEMENT(3).NE.0.0)YY(LL)=ELEMENT(1)/ELEMENT(3)	A302	120
	510 CONTINUE	A302	121
	GOTO1580	A302	122
		A302	123
		A302	124
	ENTRY A317	A302	125
	BLANK	A302	126
	1580 RETURN	A302	127
	FND	A302	128
	SUBROUTINE A312	A312	1
	RECIPROCAT	A312	2
		A312	3
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A312	4
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
	COMMON NSTEP,KEY100,KEY101,LINEL00,LINEL01,FAULTLT,DUMPLT,PRTN0	BLOC	4
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
	COMMON LASTITL(2,4)	BLOC	8
	COMMON LOCK(2)	BLOC	9
		BLOC	10
	COMMON YX(6000)	SPAR	1
	COMMON YY(6000)	COMY	1
		COMY	1
		A312	9
		A312	10
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTITL	REAL	3
		A312	12
		A312	13
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
	INTEGER NSTEP,KEY100,KEY101,LINEL00,LINEL01,FAULTLT,DUMPLT,PRTN0	INTG	2
	INTEGER LIMIT,ID,II,JJ,LL	INTG	3
	INTEGER NTITLE,NPPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
		A312	15
	II=3	A312	16
	IF(TYPE(11).EQ.5HARRAY)GOTO571	A312	17
	IF(TYPE(11).NE.8HCONSTANT)GOTO787	A312	18
	C CONSTANT LOADER	A312	19
	CALLA202	A312	20
	IF(ELEMENT(11).EQ.0.0)GOTO572	A312	21
	ELEMENT(11)=1.0/ELEMENT(11)	A312	22
	CCONSTANT UNLOADER	A312	23
	CALLA203	A312	24
	GOTO1580	A312	25
	C DIVIDE BY ZERO PROTECT	A312	26
	CFOR ILLEGAL OPERATION LOAD ANSWER WITH -0.	A312	27
	572 ELEMENT(11)=-0.0	A312	28
	IF(ELEMENT(11).EQ.0.)WRITE(11,9050)	A312	29
	IF((ELEMENT(11).EQ.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	30
	C CONSTANT LOADER	A312	31
	CALL A203	A312	32
		A312	33

GO TO1580	A312	34
571 DO1073LL=YI,YF	A312	35
CARRY LOADER	A312	36
CALL A200	A312	37
IF(ELEMENT(11).EQ.0.0)GOTO580	A312	38
ELEMENT(11)=1.0/ELEMENT(11)	A312	39
C DIVIDE BY ZERO PROTECT	A312	40
CALL A201	A312	41
GOTO1073	A312	42
580 ELEMENT(11)=-0.0	A312	43
IF(ELEMENT(11).EQ.0.)WRITE(11,9050)	A312	44
IF((ELEMENT(11).EQ.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	45
9050 FORMAT(* ZERO OR NEGATIVE ARGUMENT ENCOUNTERED*)	A312	46
CARRY UNLOADER	A312	47
CALL A201	A312	48
1073 CONTINUE	A312	49
GOTO1580	A312	50
	A312	51
	A312	52
ENTRY A306	A312	53
EXPONENTIATION,**	A312	54
C RAISE TO A POWER	A312	55
IF(TYPE(1).EQ.8HCONSTANT)10,12	A312	56
12 IF(TYPE(1).EQ.5HARRAY)16,787	A312	57
10 IF(TYPE(3).EQ.8HCONSTANT)18,14	A312	58
16 IF(TYPE(3).EQ.8HCONSTANT)602,14	A312	59
14 IF(TYPE(3).EQ.5HARRAY)602,787	A312	60
18 DO603I=1,3,2	A312	61
CCONSTANT LOADER	A312	62
603 CALL A202	A312	63
C LOAD ANSWER WITH -0. IF REAL POWER OF NEGATIVE NUMBER, OR IF NEGATIVE	A312	64
C OR ZERO POWER OF 0.	A312	65
C2=-0.	A312	66
IF((ELEMENT(1).LT.0.).OR.((ELEMENT(1).EQ.0.).AND.(ELEMENT(3).LE.0.	A312	67
1)))WRITE(11,9050)	A312	68
IF((ELEMENT(1).LT.0.).OR.((ELEMENT(1).EQ.0.).AND.(ELEMENT(3).LE.	A312	69
20.)))AND.(NTELE.EQ.1))WRITE(12,9050)	A312	70
C ANY POSITIVE POWER OF ZERO IS 0.	A312	71
IF((ELEMENT(1).EQ.0.1.AND.(ELEMENT(3).GT.0.))C2=0.	A312	72
IF(ELEMENT(1).GT.0.)C2=ELEMENT(1)**ELEMENT(3)	A312	73
GOTO1580	A312	74
602 DO6000LL=YI,YF	A312	75
DO6002I=1,3,2	A312	76
CARRY LOADER	A312	77
CALL A200	A312	78
CCONSTANT LOADER	A312	79
6002 CALL A202	A312	80
C LOAD ANSWER WITH -0. IF REAL POWER OF NEGATIVE NUMBER, OR IF NEGATIVE	A312	81
C OR ZERO POWER OF 0.	A312	82
YY(LL)=-0.	A312	83
IF((ELEMENT(1).LT.0.).OR.((ELEMENT(1).EQ.0.).AND.(ELEMENT(3).LE.0.	A312	84
1)))WRITE(11,9050)	A312	85
IF((ELEMENT(1).LT.0.).OR.((ELEMENT(1).EQ.0.).AND.(ELEMENT(3).LE.	A312	86
20.)))AND.(NTELE.EQ.1))WRITE(12,9050)	A312	87
C ANY POSITIVE POWER OF ZERO IS 0.	A312	88
IF((ELEMENT(1).EQ.0.1.AND.(ELEMENT(3).GT.0.))YY(LL)=0.	A312	89
6000 IF(ELEMENT(1).GT.0.)YY(LL)=ELEMENT(1)**ELEMENT(3)	A312	90
GOTO1580	A312	91
	A312	92
	A312	93
	A312	94
	A312	95
ENTRY A307	A312	96
C SQUARE ROOT	A312	97
II=3	A312	98
IF(TYPE(3).EQ.5HARRAY)GOTO802	A312	99
IF(TYPE(3).NE.8HCONSTANT)GOTO787	A312	100
CCONSTANT LOADER	A312	101
CALL A202	A312	102
C LOAD ANSWER WITH -0.	A312	103
C IF SQUARE ROOT OF OF NUMBER LE 0 ENCOUNTERED	A312	104
C2=-0.0	A312	105
IF(ELEMENT(3).LT.0.)WRITE(11,9050)	A312	106
IF((ELEMENT(3).LT.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	107
IF(ELEMENT(3).GE.0.0)C2=SQR(ELEMENT(3))	A312	108
GOTO1580	A312	109
802 DO803LL=YI,YF	A312	110
CARRY LOADER	A312	111
CALL A200	A312	112
YY(LL)=-0.0	A312	113
IF(ELEMENT(3).LT.0.)WRITE(11,9050)	A312	114
IF((ELEMENT(3).LT.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	115
IF(ELEMENT(3).GE.0.0)YY(LL)=SQR(ELEMENT(3))	A312	116
803 CONTINUE	A312	117
GOTO1580	A312	118
	A312	119
	A312	120
ENTRY A309	A312	121
C LOGE	A312	122
II=3	A312	123
IF(TYPE(3).EQ.5HARRAY)GOTO702	A312	124
IF(TYPE(3).NE.8HCONSTANT)GOTO787	A312	125
C CONSTANT LOADER	A312	126
CALL A202	A312	127
	A312	128

CLOAD	ANSWER WITH -0. IF ILLEGAL SITUATION PREVAILS	A312	129
	C2=-0.0	A312	130
	IF(ELEMENT(3).LE.0.)WRITE(11,9050)	A312	131
	IF((ELEMENT(3).LE.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	132
	IF(ELEMENT(3).GT.0.0)C2=ALOG(ELEMENT(3))	A312	133
	GOTO1580	A312	134
702	DO703LL=YI,YF	A312	135
CARRY	LOADER	A312	136
	CALL A200	A312	137
	YY(LL)=0.0	A312	138
	IF(ELEMENT(3).LE.0.)WRITE(11,9050)	A312	139
	IF((ELEMENT(3).LE.0.).AND.(NTELE.EQ.1))WRITE(12,9050)	A312	140
	IF(ELEMENT(3).GT.0.0)YY(LL)=ALOG(ELEMENT(3))	A312	141
703	CONTINUE	A312	142
	GOTO1580	A312	143
		A312	144
		A312	145
	ENTRY A311	A312	146
	EXP(X)	A312	147
C	II=3	A312	148
	IF(TYPE(3).EQ.5HARRAY)GOTO82	A312	149
	IF(TYPE(3).NE.8HCONSTANT)GOTO787	A312	150
C	CONSTANT LOADER	A312	151
	CALLA202	A312	152
	C2=EXP(ELEMENT(III))	A312	153
	GOTO1580	A312	154
82	DO83LL=YI,YF	A312	155
CARRY	LOADER	A312	156
	CALLA200	A312	157
83	YY(LL)=EXP(ELEMENT(III))	A312	158
	GOTO1580	A312	159
1580	RETURN	A312	160
787	WRITE(11,788)	A312	161
	IF(NTELE.EQ.1)WRITE(12,788)	A312	162
788	FORMAT(* ILLEGAL ARGUMENTS*)	A312	163
	RETURN	A312	164
	END	A312	165
		A312	166
	SUBROUTINE A322	A322	1
	SPLIT	A322	2
C	PLOT ONE PAGE GRAPH ON MICROFILM	A322	3
		A322	4
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A322	5
	COMMON C1,C2,C3,C4	BLOC	2
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	3
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	BLOC	5
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
	COMMON LASTTTL(2,4)	BLOC	9
	COMMON LOCK(2)	BLOC	10
	COMMON YX(6000)	SPAR	1
	COMMON YY(6000)	COMY	1
		COMY	1
		A322	10
		A322	11
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTTTL	REAL	3
		A322	13
		A322	14
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO	INTG	2
	INTEGER LIMIT,ID,II,JJ,LL	INTG	3
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
		A322	16
		A322	17
		A322	18
		A322	19
		A322	20
	NY=0	A322	21
	CALLGRAPH	A322	22
	GOTO1580	A322	23
		A322	24
	ENTRY A323	A322	25
	PLOT	A322	26
C	PLOT POINTS ON AN ESTABLISHED GRID ON MICROFILM	A322	27
	NY=1	A322	28
	CALL GRAPH	A322	29
	GOTO1580	A322	30
		A322	31
		A322	32
		A322	33
		A322	34
		A322	35
	ENTRY A324	A322	36
C	SET GRIDY	A322	37
C	ESTABLISH A GRID ON MICROFILM WITH AN EXTERNALLY SCALED Y-AXIS	A322	38
	ITEMP1=YI	A322	37
	ITEMP2=YF	A322	38

	YI=1	A322	39
	YF=2	A322	40
	NY=-1	A322	41
	DO 931 K=1,2	A322	42
	XBUFF(K)=YX(K)	A322	43
931	YBUFF(K)=YY(K)	A322	44
	ITEMP=LINE101	A322	45
	YX(1)=YX(ITEMP1)	A322	46
	YX(2)=YX(ITEMP2)	A322	47
	YY(1)=YLIMUP	A322	48
	YY(2)=YLIMLR	A322	49
	LINE101=0	A322	50
	CALL GRAPH	A322	51
	YI=ITEMP1	A322	52
	YF=ITEMP2	A322	53
	DO 932 K=1,2	A322	54
	YX(K)=XBUFF(K)	A322	55
932	YY(K)=YBUFF(K)	A322	56
	LINE101=ITEMP	A322	57
	GOTO1580	A322	58
	ENTRY A325	A322	59
C	SETGRIDXY	A322	60
C	ESTABLISH A GRID ON MICROFILM WITH AN EXTERNALLY SCALED X AND Y-AXIS	A322	61
	ITEMP1=YI	A322	62
	ITEMP2=YF	A322	63
	YI=1	A322	64
	YF=2	A322	65
	NY=-1	A322	66
	DO941 K=1,2	A322	67
	XBUFF(K)=YX(K)	A322	68
941	YBUFF(K)=YY(K)	A322	69
	ITEMP=LINE101	A322	70
	YY(1)=YLIMUP	A322	71
	YY(2)=YLIMLR	A322	72
	YX(1)=XLIMUP	A322	73
	YX(2)=XLIMLR	A322	74
	LINE101=0	A322	75
	CALL GRAPH	A322	76
	YI=ITEMP1	A322	77
	YF=ITEMP2	A322	78
	DO 952 K=1,2	A322	79
	YX(K)=XBUFF(K)	A322	80
952	YY(K)=YBUFF(K)	A322	81
	LINE101=ITEMP	A322	82
	GOTO 1580	A322	83
1580	RETURN	A322	84
	END	A322	85
		A322	86
C	SUBROUTINE A314	A314	1
	EMINMAX	A314	2
		A314	3
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A314	4
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	4
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
	COMMON LASTTTL(2,4)	BLOC	8
	COMMON LOCK(2)	BLOC	9
		BLOC	10
		SPAR	1
		A314	7
		A314	8
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTTTL	REAL	3
		A314	10
		A314	11
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	2
	INTEGER LIMIT,ID,II,JJ,LL	INTG	3
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
		A314	13
		A314	14
	CALL CHECKER	A314	15
	IF(DUMMY(1).NE.8HTABLE OK)GOTO9001	A314	16
	CALL MINMAX	A314	17
	RETURN	A314	18
9001	WRITE(11,9002)	A314	19
	IF(NTELE.EQ.1)WRITE(12,9002)	A314	20
9002	FORMAT(* DATA CHECKED BAD-INSTRUCTION NOT EXECUTED*)	A314	21
	RETURN	A314	22
		A314	23
		A314	24
		A314	25
		A314	26
C	ENTRY A348	A314	27
C	POINTER	A314	28
C	EMINMAX BLOCK OF Y-ARRAY DATA, TRIM IT OT FIT A-ARRAYS AND		
C	SLIDE IT TO TOP OF Y-ARRAYS.		

CALL CHECKER	A314	29
IF(DUMMY(1).NE.8HTABLE OK)GOTO9001	A314	30
CALL MINMAX	A314	31
IF(DUMMY(1).NE.8HTABLE OK)GOTO 9001	A314	32
CALL A345	A314	33
RETURN	A314	34
END	A314	35
SUBROUTINE A326	A326	1
C READTAPE1	A326	2
C READ MAGNETIC TAPE OR DISC LOGICAL NUMBER 1	A326	3
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A326	4
COMMON C1,C2,C3,C4	A326	5
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	3
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
COMMON LASTTTL(2,4)	BLOC	7
COMMON LOCK(2)	BLOC	8
REAL C1,C2,C3,C4	BLOC	9
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	SPAR	10
REAL LASTTTL	A326	1
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A326	8
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	A326	9
INTEGER LIMIT,ID,II,JJ,LL	REAL	1
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	REAL	2
DISKA=1	REAL	3
DISKB=3	A326	11
TAPENO=1	A326	12
7375 WORD(2,JJ)=8HREADTAPE	INTG	1
C CHECK FOR * IN FIRST COLUMN OF TITLE	INTG	2
7776 DECODE(1,4056,WORD(5,JJ))STAR	INTG	3
IF(ISTAR.NE.1H*)WRITE(11,4059)	A326	14
IF((ISTAR.NE.1H*).AND.(NTELE.EQ.1))WRITE(12,4059)	A326	15
IF(ISTAR.NE.1H*)GOTO1580	A326	16
CALL TANOCRD	A326	17
IF(DUMMY(3).NE.10HTABLE READ)GOTO146	A326	18
CALL CHECKER	A326	19
IF(DUMMY(1).NE.8HTABLE OK)GOTO146	A326	20
GOTO1515	A326	21
146 CONTINUE	A326	22
WRITE(11,1666)	A326	23
IF(NTELE.EQ.1)WRITE(12,1666)	A326	24
1666 FORMAT(* DATA READ FROM TAPE IS EITHER NOT THERE OR ILLEGAL*)	A326	25
1580 RETURN	A326	26
1515 WRITE(11,7787)(WORD(1,JJ),I=5,8)	A326	27
IF(NTELE.EQ.1)WRITE(12,7787)(WORD(1,JJ),I=5,8)	A326	28
7787 FORMAT(* DATA UNDER TITLE *A10 *READ AND CHECKED TO BE LEGAL*)	A326	29
GO TO 1580	A326	30
ENTRY A346	A326	31
C READTAPE2	A326	32
C READ MAGNETIC TAPE OR DISC LOGICAL NUMBER 2	A326	33
DISKA=2	A326	34
DISKB=4	A326	35
TAPENO=2	A326	36
GOTO7375	A326	37
ENTRY A327	A326	38
C WRITETAPE1	A326	39
C WRITE LOGICAL NUMBER 1 DISC OR MAGNETIC TAPE	A326	40
DISKA=1	A326	41
DISKB=3	A326	42
TAPENO=1	A326	43
7377 WORD(2,JJ)=9HWRITETAPE	A326	44
C CHECK FOR NEW OR OLD TAPE	A326	45
IF(WORD(3,JJ).EQ.2H)GOTO4037	A326	46
IF(WORD(3,JJ).EQ.2H**)GOTO4037	A326	47
WRITE(11,4052)	A326	48
IF(NTELE.EQ.1)WRITE(12,4052)	A326	49
4052 FORMAT(* TAPE NOT WRITTEN BECAUSE NEW OR OLD NOT SET*)	A326	50
GOTO1580	A326	51
C CHECK FOR * IN COLUMN ONE OF TITLE	A326	52
4037 DECODE(1,4056,WORD(5,JJ))STAR	A326	53
4056 FORMAT(A1)	A326	54
IF(ISTAR.EQ.1H*)GOTO4058	A326	55
WRITE(11,4059)	A326	56
IF(NTELE.EQ.1)WRITE(12,4059)	A326	57

4059	FORMAT(* REQUESTING TITLE INCORRECT*)	A326	70
	GOTO 1580	A326	71
4058	CALL CHECKER	A326	72
	IF((YF-YI+1).GT.LENGTHY)GOTO20	A326	73
	IF(DUMMY(1).EQ.8HTABLE OK)GOTO4031	A326	74
20	WRITE(11,4032)	A326	75
	IF(NTELE.EQ.1)WRITE(12,4032)	A326	76
4032	FORMAT(* DATA ILLEGAL-NOT WRITTEN ON TAPE*)	A326	77
	GOTO 1580	A326	78
4031	CALL TWRITE	A326	79
	IF(DUMMY(3).EQ.4HGOOD)GOTO5075	A326	80
	WRITE(11,4074)	A326	81
	IF(NTELE.EQ.1)WRITE(12,4074)	A326	82
4074	FORMAT(* TAPE NOT WRITTEN*)	A326	83
	GOTO1580	A326	84
5075	WRITE(11,4070)	A326	85
	IF(NTELE.EQ.1)WRITE(12,4070)	A326	86
4070	FORMAT(* DATA PUT ON TAPE IS IN ACCORDANCE *	A326	87
	1*WITH CHECKER RULES*)	A326	88
	GOTO1580	A326	89
		A326	90
	ENTRY A347	A326	91
	WRITETAPE2	A326	92
C	WRITE LOGICAL NUMBER 2 DISC OR MAGNETIC TAPE	A326	93
	DISKA=2	A326	94
	DISKB=4	A326	95
	TAPENO=2	A326	96
	GOTO7377	A326	97
	END	A326	98
		A326	99
	SUBROUTINE A328	A328	1
	MPL0T2	A328	2
C	PLOT A SCROLL GRAPH ON MICROFILM	A328	3
		A328	4
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A328	5
	COMMON C1,C2,C3,C4	BLOC	6
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	7
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	8
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	9
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	10
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	11
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	12
	COMMON LASTTTL(2,4)	BLOC	13
	COMMON LOCK(2)	BLOC	14
	COMMON YX(6000)	SPAR	15
	COMMON YY(6000)	COMY	16
	COMMON AX(6000)	COMY	17
	COMMON AY(6000)	COMA	18
		COMA	19
		A328	20
	REAL C1,C2,C3,C4	REAL	21
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	22
	REAL LASTTTL	REAL	23
		A328	24
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A328	25
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	26
	INTEGER LIMIT,ID,II,JJ,LL	INTG	27
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	28
		A328	29
	WRITE(11,220)	A328	30
	IF(NTELE.EQ.1)WRITE(12,220)	A328	31
220	FORMAT(* CHECKING Y-ARRAY DATA*)	A328	32
	NPRINT=PRTN0	A328	33
	CALL CHECKER	A328	34
	IF(DUMMY(1).NE.8HTABLE OK)GOTO9001	A328	35
	IF(WORD(4,JJ).NE.2.) GO TO 2040	A328	36
	WRITE(11,230)	A328	37
	IF(NTELE.EQ.1)WRITE(12,230)	A328	38
230	FORMAT(* CHECKING A-ARRAY DATA*)	A328	39
	IEXCH=0	A328	40
2038	ITEMP=YI	A328	41
	YI=AI	A328	42
	AI=ITEMP	A328	43
	ITEMP=YF	A328	44
	YF=AF	A328	45
	AF=ITEMP	A328	46
	DO 2039 ITOA=1,LENGTHA	A328	47
	TEMP=YX(ITOA)	A328	48
	YX(ITOA)=AX(ITOA)	A328	49
	AX(ITOA)=TEMP	A328	50
	TEMP=YY(ITOA)	A328	51
	YY(ITOA)=AY(ITOA)	A328	52
	AY(ITOA)=TEMP	A328	53
2039	CONTINUE	A328	54
	IEXCH=IEXCH+1	A328	55
	IF(IEXCH.EQ.2) GO TO 2040	A328	56

PRINO=NPRINT	A328	47
CALL CHECKER	A328	48
IF(DUMMY(1).EQ.8HTABLE OK) GO TO 2038	A328	49
WORD(4,JJ)=0.0	A328	50
WRITE(11,2042)	A328	51
IF(NTELE.EQ.1)WRITE(12,2042)	A328	52
2042 FORMAT(*0 DUMMY(1).NE.8HTABLE OK FOR SET (AX,AY)--WORD(4,JJ)=0.0*)	A328	53
GO TO 2038	A328	54
2040 IF(WORD(2,JJ).EQ.6HMPLT2)CALL MPLT2	A328	55
1580 RETURN	A328	56
9001 WRITE(11,9002)	A328	57
IF(NTELE.EQ.1)WRITE(12,9002)	A328	58
9002 FORMAT(* DATA CHECKED BAD-INSTRUCTION NOT EXECUTED*)	A328	59
GOTO1580	A328	60
END	A328	61
SUBROUTINE A349	A349	1
C CALCOM2	A349	2
C PLOT A SCROLL GRAPH ON PAPER	A349	3
	A349	4
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A349	5
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
COMMON LASTTTL(2,4)	BLOC	8
COMMON LOCK(2)	BLOC	9
COMMON YX(6000)	BLOC	10
COMMON YY(6000)	SPAR	1
COMMON AX(6000)	COMY	1
COMMON AY(6000)	COMY	1
	COMA	1
	CCMA	1
	A349	12
	A349	13
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A349	15
	A349	16
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	INTG	2
INTEGER LIMIT,II,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A349	18
	A349	19
WRITE(11,220)	A349	20
IF(NTELE.EQ.1)WRITE(12,220)	A349	21
220 FORMAT(* CHECKING Y-ARRAY DATA*)	A349	22
NPRINT=PRINO	A349	23
CALL CHECKER	A349	24
IF(DUMMY(1).NE.8HTABLE OK)GOTO9001	A349	25
IF(WORD(4,JJ).NE.2.) GO TO 2040	A349	26
WRITE(11,230)	A349	27
IF(NTELE.EQ.1)WRITE(12,230)	A349	28
230 FORMAT(* CHECKING A-ARRAY DATA*)	A349	29
IEXCH=0	A349	30
2038 ITEMP=YI	A349	31
YI=AI	A349	32
AI=ITEMP	A349	33
ITEMP=YF	A349	34
YF=AF	A349	35
AF=ITEMP	A349	36
DO 2039 ITOA=1,LENGTHA	A349	37
TEMP=YX(ITOA)	A349	38
YX(ITOA)=AX(ITOA)	A349	39
AX(ITOA)=TEMP	A349	40
TEMP=YY(ITOA)	A349	41
YY(ITOA)=AY(ITOA)	A349	42
AY(ITOA)=TEMP	A349	43
2039 CONTINUE	A349	44
IEXCH=IEXCH+1	A349	45
IF(IEXCH.EQ.2) GO TO 2040	A349	46
PRINO=NPRINT	A349	47
CALL CHECKER	A349	48
IF(DUMMY(1).EQ.8HTABLE OK) GO TO 2038	A349	49
WORD(4,JJ)=0.0	A349	50
WRITE(11,2042)	A349	51
IF(NTELE.EQ.1)WRITE(12,2042)	A349	52
2042 FORMAT(*0 DUMMY(1).NE.8HTABLE OK FOR SET (AX,AY)--WORD(4,JJ)=0.0*)	A349	53
GO TO 2038	A349	54
2040 IF(WORD(2,JJ).EQ.7HCALCOM2)CALL CALCOM2	A349	55
1580 RETURN	A349	56
9001 WRITE(11,9002)	A349	57
IF(NTELE.EQ.1)WRITE(12,9002)	A349	58
9002 FORMAT(* DATA CHECKED BAD-INSTRUCTION NOT EXECUTED*)	A349	59
GOTO1580	A349	60
END	A349	61

SUBROUTINE MPLTCAL
C MPLOT CALLER
C PLOT A SCROLL ON MICROFILM

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

CALL CHECKER
IF(DUMMY(1).NE.8HTABLE OK)GOTO9001
CALL MPLOT
RETURN
9001 WRITE(11,9002)
IF(NTELE.EQ.1)WRITE(12,9002)
9002 FORMAT(* DATA CHECKED BAD-INSTRUCTION NOT EXECUTED*)
RETURN
END

MPLT 1
MPLT 2
MPLT 3
MPLT 4
MPLT 5
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
MPLT 8
MPLT 9
REAL 1
REAL 2
REAL 3
MPLT 11
MPLT 12
INTG 1
INTG 2
INTG 3
INTG 4
MPLT 14
MPLT 15
MPLT 16
MPLT 17
MPLT 18
MPLT 19
MPLT 20
MPLT 21
MPLT 22
MPLT 23
MPLT 24

SUBROUTINE CALCALL
C CALCOMP CALLER
C PLOT A SCROLL ON PAPER

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

CALL CHECKER
IF(DUMMY(1).NE.8HTABLE OK)GOTO9001
CALL CALCOMP
RETURN
9001 WRITE(11,9002)
IF(NTELE.EQ.1)WRITE(12,9002)
9002 FORMAT(* DATA CHECKED BAD-INSTRUCTION NOT EXECUTED*)
RETURN
END

CALC 1
CALC 2
CALC 3
CALC 4
CALC 5
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
CALC 8
CALC 9
REAL 1
REAL 2
REAL 3
REAL 4
CALC 12
CALC 11
INTG 1
INTG 2
INTG 3
INTG 4
CALC 14
CALC 15
CALC 16
CALC 17
CALC 18
CALC 19
CALC 20
CALC 21
CALC 22
CALC 23
CALC 24

SUBROUTINE A329
C PRINT CONTENTS OF COMMON REGION OF TELERAY
C DUMP

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX

A329 1
A329 2
A329 3
A329 4
A329 5
BLOC 1
BLOC 2
BLOC 3
BLOC 4

```
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)
```

```
COMMON YX(6000)
COMMON YY(6000)
```

```
REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL
```

```
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
```

```
ITEMP=DUMPLT
DUMPLT=1
CALL A308
DUMPLT=ITEMP
GOTO1580
```

```
ENTRY A330
PUNCH
```

```
C PUNCH CARDS
CALL CHECKER
IF(DUMMY(1).EQ.8HTABLE OK)GOTO3333
WRITE(11,3332)DUMMY(1)
IF(INTELE.EQ.1)WRITE(12,3332)DUMMY(1)
3332 FORMAT(* *A10,*--ARRAYS NT PUNCHED*)
GOTO1580
3333 IF(INTELE.EQ.1)3307,3308
3308 WRITE(12,3327)COMMENT,(YX(1),YY(1),I=YI,YF)
3327 FORMAT(4A10/(6E12.5))
WRITE(11,3334)
3334 FORMAT(* COMMENT AND YX AND YY PUNCHED*)
GOTO1580
3307 CALL SELECTR
WRITE(12,3309)COMMENT,(YX(1),YY(1),I=YI,YF,PRTN0)
3309 FORMAT(*O*4A10///(* *6E12.5))
1580 RETURN
END
```

```
SUBROUTINE A332
INTEGRATE
TRAPEZOIDAL RULE INTEGRATION
```

```
COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)
```

```
COMMON YX(6000)
COMMON YY(6000)
```

```
REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL
```

```
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,N1,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
```

```
C CALL LEGALORDER AND NOEQUALX TO CHECK AND CLEANUP Y-ARRAYS
```

```
CALL A338
IF(DUMMY(1).NE.8HTABLE OK)RETURN
IUP=YF-1
SUM=0.0
DO50371=YI,IUP
DELTAX=YX(1)-YX(I+1)
```

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

	DELTAX=ABS(DELTAX)	A332	25
	SUMY=YY(I)+YY(I+1)	A332	26
5037	SUM=SUM+0.5*SUMY*DELTAX	A332	27
	C2=SUM	A332	28
	WRITE(11,5038)YX(IYI),YX(IYF),C2	A332	29
	IF(NTELE.EQ.1)WRITE(12,5038)YX(IYI),YX(IYF),C2	A332	30
5038	FORMAT('OINTEGRAL OF YY FROM YX=*E12.5/* TO YX=*E12.5,*=C2=*E12.5')	A332	31
	RETURN	A332	32
	END	A332	33
		A332	34

	SUBROUTINE SELECTR	SELE	1
CSETS	DO LOOP INTERVAL AND ESTABLISHES TYPE OF GRAPH	SELE	2
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	2
	COMMON C1,C2,C3,C4	BLOC	1
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF	BLOC	3
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	BLOC	5
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
	COMMON LASTTTL(2,4)	BLOC	9
	COMMON LOCK(2)	BLOC	10
		SPAR	1
		SELE	5
	REAL C1,C2,C3,C4	SELE	6
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
	REAL LASTTTL	REAL	2
		REAL	3
	INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF	SELE	8
	INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	INTG	1
	INTEGER LIMIT,ID,II,JJ,LL	INTG	2
	INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
	INTEGER PRINTNO	SELE	10
	INTEGER GRAPHNO,DELN	SELE	11
	EQUIVALENCE(PRTNO,PRINTNO)	SELE	12
		SELE	13
		SELE	14
		SELE	15
C	SETTING OF THE PRINTING INTERVAL	SELE	16
	DELN=YF-YI	SELE	17
	IF(WORD(2,JJ).EQ.10MPRALLARRAY)DELN=NF-NI	SELE	18
	GRAPHNO=1	SELE	19
	IF(DELN.LE.100)GOTO1	SELE	20
	IF(DELN.LE.500)GOTO2	SELE	21
	IF(DELN.LE.1500)GOTO3	SELE	22
	IF(DELN.LE.3000)GOTO4	SELE	23
	IF(DELN.LE.6000)GOTO5	SELE	24
1	PRINTNO=1	SELE	25
	RETURN	SELE	26
2	PRINTNO=4	SELE	27
	RETURN	SELE	28
3	PRINTNO=14	SELE	29
	RETURN	SELE	30
4	PRINTNO=24	SELE	31
	RETURN	SELE	32
5	PRINTNO=50	SELE	33
	RETURN	SELE	34
	END	SELE	35

		SELE	36
	SUBROUTINE A333	SELE	37
	LASTTITLE1	A333	1
C	SETUP OF TAPE UNIT FOR LAST TITLE 1	A333	2
		A333	3
		A333	4
		A333	5
	COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
	COMMON C1,C2,C3,C4	BLOC	2
	COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF	BLOC	3
	COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
	COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	BLOC	5
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
	COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
	COMMON LASTTTL(2,4)	BLOC	9
	COMMON LOCK(2)	BLOC	10
		SPAR	1
	COMMON YX(6000)	COMY	1
	COMMON YY(6000)	COMY	1
		A333	10
		A333	11
	REAL C1,C2,C3,C4	REAL	1
	REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
	REAL LASTTTL	REAL	3
		A333	13

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A333	14
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
	INTG	4
	A333	16
TAPENO=1	A333	17
11 DO12I=1,4	A333	18
K=4+1	A333	19
12 LASTTTL(TAPENO,I)=WORD(K,JJ)	A333	20
LOCK(TAPENO)=1	A333	21
13 RETURN	A333	22
	A333	23
	A333	24
ENTRY A350	A333	25
LASTTITLE2	A333	26
C SETUP OF TAPE UNIT FOR LAST TITLE 2	A333	27
TAPENO=2	A333	28
GOTO11	A333	29
	A333	30
	A333	31
ENTRY A334	A333	32
LOADYLIMUP	A333	33
C LOAD YLIMUP SO THAT IT CAN BE USED AND PRINTED	A333	34
TEMP=YY(YI)	A333	35
DO30I=YI,YF	A333	36
IF(TEMP.GE.YY(I))GOTO30	A333	37
TEMP=YY(I)	A333	38
30 CONTINUE	A333	39
YLIMUP=TEMP	A333	40
WRITE(11,50)YLIMUP	A333	41
IF(INTELE.EQ.1)WRITE(12,50)YLIMUP	A333	42
50 FORMAT(* YLIMUP=*,E12.5)	A333	43
GOTO13	A333	44
	A333	45
	A333	46
ENTRY A351	A333	47
LOADYLIMLR	A333	48
C LOAD YLIMLR SO THAT IT CAN BE USED AND PRINTED	A333	49
TEMP=YY(YI)	A333	50
DO31I=YI,YF	A333	51
IF(TEMP.LE.YY(I))GOTO31	A333	52
TEMP=YY(I)	A333	53
31 CONTINUE	A333	54
YLIMLR=TEMP	A333	55
WRITE(11,60)YLIMLR	A333	56
IF(INTELE.EQ.1)WRITE(12,60)YLIMLR	A333	57
60 FORMAT(* YLIMLR=*,E12.5)	A333	58
GOTO13	A333	59
ENG	A333	60
	A333	61
SUBROUTINE A336	A336	1
AVOVERTIME	A336	2
C TIME AVERAGING OF ORDINATE-VS-NEUTRON-ENERGY DATA	A336	3
	A336	4
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A336	5
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
COMMON XLIMUP,YLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
COMMON LASTTTL(2,4)	BLOC	8
COMMON LOCK(2)	BLOC	9
	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMY	1
COMMON AX(6000)	COMY	1
COMMON AY(6000)	COMA	1
	COMA	1
	A336	12
REAL C1,C2,C3,C4	A336	13
REAL XLIMUP,YLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
REAL LASTTTL	REAL	2
	REAL	3
	A336	15
	A336	16
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A336	18
C YX(I)=NEUTRON ENERGY IN EV	A336	19
C YY=ORDINATE	A336	20
C C1=FLGT PATH IN METERS	A336	21
C C3=TIME STEP IN MICROSECONDS	A336	22
CAX(I) IS AVERAGE TIME CONVERTED TO ENERGY AND AY(I) IS AVERAGE ORDINATE	A336	23
C AVERAGING ROUTINE WAS WRITTEN BY JOHN WARREN	A336	24
	A336	25


```

COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)
COMMON YY(6000)

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULT,DT,DUMPLT,PRTNQ
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

DIMENSION BUFF(11)

C CALL LEGALORDER AND NOEQUALX TO CHECK AND CLEANUP Y-ARRAYS
CALL A338
IF(DUMMY(1).NE.8HTABLE OK)4553,4570
4570 WRITE(11,60)
IF(INTELE.EQ.1)WRITE(12,60)
60 FORMAT(1H1,10X*PRESCRIPTION FOR FINDING AN INTEGRAL-*/
11X,*INTEGRAL=RUNSUM(XSMALL)-RUNSUM(XLARGE) + */
11X,*RUNSUMDECREMENT(XSMALL)-RUNSUMDECREMENT(XLARGE)*//)
CCON=0.
WRITE(11,6110)YX(YI),YY(YI),CCON
IF(INTELE.EQ.1)WRITE(12,6110)YX(YI),YY(YI),CCON
C C2=MAXIMUM VALUE RUNSUM IS ALLOWED TO REACH
IF((C2.EQ.0.).OR.(C2.GT.1.E+6))WRITE(11,200)
200 FORMAT(1* C2=0.,OR.C2.GT.1.E+6,RESET TO 1.E+6*//)
IF((C2.EQ.0.).OR.(C2.GT.1.E+6)).AND.(INTELE.EQ.1))WRITE(12,200)
IF((C2.EQ.0.).OR.(C2.GT.1.E+6))C2=1.E+6
C INITIALIZE VARIABLES
K=1
NPASS=1
BUFF(1)=0.
IEND=YF-1
NBUFF=10
IF(INTELE.EQ.1)NBUFF=6
C RUNNING SUM LOOP
DO 765 I=YI,IEND
DELTA=X(YI)-X(I+1)
DELTA=ABS(DELTA)
SUMY=YI+YI+1
ARG=0.5*SUMY*DELTA
BUFF(K+1)=BUFF(K)+ARG
IF(BUFF(K+1).GT.C2)6000,6020
C LIMITING BUFF(K+1) TO BE .LE. C2
6000 ICTR=0
6040 BUFF(K+1)=BUFF(K+1)-C2
CCON=CCON+C2
ICTR=ICTR+1
C INFINITE LOOP PROTECTION
IF(ICTR.GT.100)6080,6090
6080 WRITE(11,6100)
IF(INTELE.EQ.1)WRITE(12,6100)
6100 FORMAT(1* INFINITE LOOP ENCOUNTERED IN */
1* RUNSUM MODULE. EXECUTION TERMINATED*)
6090 IF(BUFF(K+1).LE. C2)6010,6040
6010 WRITE(11,6110)YX(I+1),YY(I+1),CCON
IF(INTELE.EQ.1)WRITE(12,6110)YX(I+1),YY(I+1),CCON
6110 FORMAT(1* FROM X= *E12.5,*Y= *E12.5,* TO THE X,Y *
*/ OF THE NEXT SIMILAR DIAGNOSTIC, RUNSUM IS DECREASED*
*/ BY *E17.10)
6020 IF((K.EQ.NBUFF).OR.(I.EQ.IEND))1500,1550
1550 K=K+1
GOTO 765
1500 NABC=1
MMMM=K
IF((K.NE.NBUFF).AND.(I.EQ.IEND))MMMM=MMMM+1
5000 IF(INTELE.EQ.1)8001,8002
8001 WRITE(12,8003)(YX(I-K+M),M=NABC,MMMM)
WRITE(12,8004)(YY(I-K+M),M=NABC,MMMM)
WRITE(12,8005)(BUFF(M),M=NABC,MMMM)
8002 WRITE(11,75)(YX(I-K+M),M=NABC,MMMM)
WRITE(11,76)(YY(I-K+M),M=NABC,MMMM)
WRITE(11,77)(BUFF(M),M=NABC,MMMM)
IF(NPASS.EQ.0)GOTO765
NABC=NBUFF+1
MMMM=NBUFF+1
NPASS=0
IF((K.EQ.NBUFF).AND.(I.EQ.IEND))GOTO5000
8003 FORMAT(1* X=*6F12.3)
8004 FORMAT(1* Y=*6F12.3)
8005 FORMAT(1* RSUM=*6F12.3//)
75 FORMAT(1* X=*10F12.3)
76 FORMAT(1* Y=*10F12.3)
77 FORMAT(1* RSUM=*10F12.3//)
BUFF(1)=BUFF(K+1)
K=1

```

```

BLOC 9
BLOC 10
SPAR 1
COMY 1
COMY 1
A337 9
A337 10
REAL 1
REAL 2
REAL 3
A337 12
A337 13
INTG 1
INTG 2
INTG 3
INTG 4
A337 15
A337 16
A337 17
A337 18
A337 19
A337 20
A337 21
A337 22
A337 23
A337 24
A337 25
A337 26
A337 27
A337 28
A337 29
A337 30
A337 31
A337 32
A337 33
A337 34
A337 35
A337 36
A337 37
A337 38
A337 39
A337 40
A337 41
A337 42
A337 43
A337 44
A337 45
A337 46
A337 47
A337 48
A337 49
A337 50
A337 51
A337 52
A337 53
A337 54
A337 55
A337 56
A337 57
A337 58
A337 59
A337 60
A337 61
A337 62
A337 63
A337 64
A337 65
A337 66
A337 67
A337 68
A337 69
A337 70
A337 71
A337 72
A337 73
A337 74
A337 75
A337 76
A337 77
A337 78
A337 79
A337 80
A337 81
A337 82
A337 83
A337 84
A337 85
A337 86
A337 87
A337 88
A337 89
A337 90
A337 91
A337 92
A337 93

```


NPASS=1	A337	94
765 CONTINUE	A337	95
RETURN	A337	96
	A337	97
	A337	98
ENTRY A353	A337	99
C RESINTEGRAL	A337	100
C RESONANCE INTEGRAL OF CROSS SECTION IN BARNS STORED IN YY VERSUS	A337	101
C ENERGY IN ELECTRON VOLTS STORED IN YX	A337	102
C CHECK LEGALITY AND ORDER OF Y-ARRAY DATA	A337	103
C AND AVERAGE POINTS HAVING SAME YX.	A337	104
C NEGATIVE XSECS LEFT OUT INORDER TO PREVENT BUILDUP	A337	105
C OF NEGATIVE AREAS IN BROAD ENERGY REGIONS OF SMALL	A337	106
C BACKGROUNDS.	A337	107
CALL A338	A337	108
IF(DUMMY(1).NE.8HTABLE OK)4553,4554	A337	109
4553 WRITE(11,4556)	A337	110
IF(INTELE.EQ.1)WRITE(12,4556)	A337	111
4556 FORMAT(* BAD DATA--MODULE SKIPPED*)	A337	112
RETURN	A337	113
4554 NNN=YF-1	A337	114
C2=0.	A337	115
C WORKS OK WHEN YI=NNN---ONLY TWO DATA POINTS	A337	116
DO 2000 I=YI,NNN,1	A337	117
CRS1=YY(I)	A337	118
C HANDLES NEGATIVE ZERO ALSO	A337	119
IF(CRS1.LE.0.)CRS1=0.	A337	120
CRS2=YY(I+1)	A337	121
C HANDLES NEGATIVE ZERO ALSO	A337	122
IF(CRS2.LE.0.)CRS2=0.	A337	123
CRS1=CRS1+CRS2	A337	124
CRS1=CRS1*(YX(I)-YX(I+1))	A337	125
CRS1=CRS1/(YX(I)+YX(I+1))	A337	126
2000 C2=C2+CRS1	A337	127
WRITE(11,5010)YX(YF),YX(YI),C2	A337	128
5010 FORMAT(*THE RESONANCE INTEGRAL BETWEEN E=*E12.5,*AND*/	A337	129
1* E=*E12.5,*IS=*E12.5,* AND IS STORED IN C2.*)	A337	130
IF(INTELE.EQ.1)WRITE(12,5010)YX(YF),YX(YI),C2	A337	131
13 RETURN	A337	132
END	A337	133
SUBROUTINE A338	A338	1
C NOEQUALX	A338	2
C COMBINE POINTS WITH THE SAME VALUE OF X INTO A POINT AT THIS X WITH	A338	3
C A Y-VALUE THAT IS THE ARITHMETIC AVERAGE OF THE ONES OF THE POINTS	A338	4
	A338	5
COMMON NTELE,LENGTH,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LOCK(2)	BLOC	10
COMMON YX(6000)	SPAR	1
COMMON YY(6000)	COMY	1
	COMY	1
	A338	11
	A338	12
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A338	14
	A338	15
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A338	17
	A338	18
NPRTNO=PRTN0	A338	19
C CHECK LEGALITY AND ORDER OF Y-ARRAY DATA	A338	20
CALL CHECKER	A338	21
PRTN0=NPRTNO	A338	22
IF(DUMMY(1).NE.8HTABLE OK)GOTO9000	A338	23
C IF PRTN0.NE.0 PRINT HEADINGS	A338	24
IF(PRTNO.NE.0)WRITE(11,1210)	A338	25
IF((PRTNO.NE.0).AND.(INTELE.EQ.1))WRITE(12,1210)	A338	26
1210 FORMAT(1H,15X,*DUPLICATE X COMBINING*//16X,5X,*INDEX*2X,2X,4X*YX(	A338	27
11)*4X,2X,4X*YY(11)*4X///)	A338	28
KEY=0	A338	29
K=YI-1	A338	30
DO1220I=YI,YF	A338	31
J=1	A338	32
SUM=YY(I)	A338	33
1230 IF(YX(I).EQ.YX(I+J))1240,1350	A338	34
1240 JUMP=KEY+1	A338	35
IF((JUMP.EQ.1).AND.(PRTNO.NE.0))1250,1260	A338	36

1250	WRITE(11,1270)I,YX(I),YY(I)	A338	37
	IF(INTELE.EQ.1)WRITE(12,1270)I,YX(I),YY(I)	A338	38
1270	FORMAT(16X,5X15,2X,E12.5,2X,E12.5)	A338	39
C	PRINT THE DUPLICATE POINTS AND THE RESULTING ONES	A338	40
C	ALSO PRINT THE FINAL TABLE INDICIES	A338	41
1260	JNUMBER=I+J	A338	42
	IF(PRTNO.NE.0)WRITE(11,1270)JNUMBER,YX(JNUMBER),YY(JNUMBER)	A338	43
	IF((PRTNO.NE.0).AND.(INTELE.EQ.1))WRITE(12,1270)JNUMBER,YX(JNUMBER)	A338	44
	1,YY(JNUMBER)	A338	45
C	AVERAGE YY VALUES AT EQUAL YX VALUES	A338	46
1280	SUM=SUM+YY(I+J)	A338	47
	J=J+1	A338	48
C	KEY .NE.0 MEANS DUPLICATE YX ENCOUNTERED	A338	49
	KEY=1	A338	50
	IF((I+J).GT.YF)GOTO1350	A338	51
	GOTO1230	A338	52
1350	IF(KEY.NE.0)1290,1300	A338	53
1290	SUM=SUM/FLOAT(J)	A338	54
C	IF PRTNO.NE.0 PRINT AVERAGE POINT	A338	55
	IF(PRTNO.NE.0)WRITE(11,1310)I,YX(I),SUM	A338	56
	IF((PRTNO.NE.0).AND.(INTELE.EQ.1))WRITE(12,1310)I,YX(I),SUM	A338	57
C	SLIP ARRAYS OVER DUPLICATE YX	A338	58
1300	K=K+1	A338	59
1310	FORMAT(1H 9X*RESULT*5X,15,2X,E12.5,2X,E12.5/)	A338	60
	YX(K)=YX(I)	A338	61
	YY(K)=SUM	A338	62
	IF(KEY.EQ.0)GOTO1220	A338	63
	I=J+1 -1	A338	64
	KEY=0	A338	65
1220	CONTINUE	A338	66
	WRITE(11,1330)YI,YF,YI,K	A338	67
C	PRINT FINAL RESULTS	A338	68
	IF(INTELE.EQ.1)WRITE(12,1330)YI,YF,YI,K	A338	69
1330	FORMAT(1H0,15X,*INITIAL YI=*15,*INITIAL YF=*,15/16X,*FINAL YI=	A338	70
	1*,15,* FINAL YF=*,15/16X,*NOTE---CHANGE IN YF INDICATES THAT STA	A338	71
	2RUCTURE*/15X,* OF Y-ARRAYS HAS BEEN CHANGED BY THE REMOVAL*/15X,	A338	72
	3* OF MULTIPLY DEFINED POINTS HAVING SAME YX.*)	A338	73
C	RESET YF	A338	74
	YF=K	A338	75
	IF(YF.EQ.YI)GOTO600	A338	76
	PRTNO=0	A338	77
13	RETURN	A338	78
9000	WRITE(11,9001)	A338	79
	IF(INTELE.EQ.1)WRITE(12,9001)	A338	80
9001	FORMAT(* Y-ARRAYS CONTAIN BAD DATA. INSTRUCTION MODULE SKIPPED*)	A338	81
	PRTNO=0	A338	82
	RETURN	A338	83
600	WRITE(11,610)	A338	84
	IF(INTELE.EQ.1)WRITE(12,610)	A338	85
610	FORMAT(16X,*ONLY ONE POINT REMAINING AFTER NOEQUALX */	A338	86
	116X*OPERATION*)	A338	87
	DUMMY(1)=3HBAD	A338	88
	PRTNO=0	A338	89
	RETURN	A338	90
	END	A338	91

SUBROUTINE SEARCH
 CSINGLE PASS. LINEAR INTERPOLATION ROUTINE. BOUNDS OF STANDARD TABLE B
 CMAY BE RESET. TABLE A INTERPOLATED AT EB TO FIND ORDINATE YA. TABLES
 CMUST CONFORM TO RULES GIVEN BY CHECKER.
 COORDINATES=IA,IB
 CABCISSA=EA,EB

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
 COMMON C1,C2,C3,C4
 COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
 COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
 COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULT1,DUMPLT,PRTNO
 COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
 COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
 COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
 COMMON LASTTTL(2,4)
 COMMON LOCK(2)

COMMON YX(6000)
 COMMON YY(6000)
 COMMON AX(6000)
 COMMON AY(6000)
 COMMON BY(6000)

EQUIVALENCE (YX,EA),(YY,IA),(AX,EB),(BY,YA)
 EQUIVALENCE (YI,NAI),(YF,NAF),(AI,NBI),(AF,NBF)

REAL EA(1),IA(1),EB(1),YA(1),AY
 REAL C1,C2,C3,C4
 REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT

SEAR	1
SEAR	2
SEAR	3
SEAR	4
SEAR	5
SEAR	6
SEAR	7
SEAR	8
SEAR	9
SEAR	10
BLOC	1
BLOC	2
BLOC	3
BLOC	4
BLOC	5
BLOC	6
BLOC	7
BLOC	8
BLOC	9
BLOC	10
SPAR	1
COMY	1
COMY	1
COMA	1
COMA	1
COMB	1
COMB	1
SEAR	18
SEAR	19
SEAR	20
SEAR	21
SEAR	22
SEAR	23
SEAR	24
REAL	1
REAL	2

REAL LASTTTL	REAL	3
REAL TEMP	SEAR	26
	SEAR	27
	SEAR	28
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTRNO	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
INTEGER NDEX,JUMP,J,K	SEAR	30
	SEAR	31
	SEAR	32
C NBI=INITIAL INDEX OF STANDARD TABLE	SEAR	33
C NBF=FINAL INDEX OF STANDARD TABLE	SEAR	34
C NAI=INITIAL INDEX OF INTERPOLATED TABLE	SEAR	35
C NAF=FINAL INDEX OF INTERPOLATED TABLE	SEAR	36
C CHECKING OF Y AND A ARRAY DATA	SEAR	37
NPRINT=PRTRNO	SEAR	38
WRITE(11,2200)	SEAR	39
IF(INTELE.EQ.1)WRITE(12,2200)	SEAR	40
2200 FORMAT(* CHECKING Y-ARRAY DATA*)	SEAR	41
CALL CHECKER	SEAR	42
IF(DUMMY(1).NE.8HTABLE OK)GOTO9001	SEAR	43
WRITE(11,2220)	SEAR	44
IF(INTELE.EQ.1)WRITE(12,2220)	SEAR	45
2220 FORMAT(* CHECKING A-ARRAY DATA*)	SEAR	46
C SWAP A-ARRAYS WITH Y-ARRAYS	SEAR	47
IEXCH=0	SEAR	48
2038 ITEMP=NAI	SEAR	49
NAI=NBI	SEAR	50
NBI=ITEMP	SEAR	51
ITEMP=NAF	SEAR	52
NAF=NBF	SEAR	53
NBF=ITEMP	SEAR	54
DO 2039 ITOA=1,LENGTHA	SEAR	55
TEMP=EA(ITOA)	SEAR	56
EA(ITOA)=EB(ITOA)	SEAR	57
EB(ITOA)=TEMP	SEAR	58
TEMP=IA(ITOA)	SEAR	59
IA(ITOA)=AY(ITOA)	SEAR	60
AY(ITOA)=TEMP	SEAR	61
2039 CONTINUE	SEAR	62
IEXCH=IEXCH+1	SEAR	63
IF(IEXCH.EQ.2)GOTO2040	SEAR	64
PRTRNO=NPRINT	SEAR	65
CALL CHECKER	SEAR	66
GOTO2038	SEAR	67
2040 IF(DUMMY(1).NE.8HTABLE OK)9001,9010	SEAR	68
9001 WRITE(11,9002)	SEAR	69
IF(INTELE.EQ.1)WRITE(12,9002)	SEAR	70
9002 FORMAT(* DATA CHECKED BAD-MODULE NOT EXECUTED*)	SEAR	71
RETURN	SEAR	72
9010 CONTINUE	SEAR	73
NSWH=0	SEAR	74
NBISAVE=NBI	SEAR	75
NBFSAVE=NBF	SEAR	76
C	SEAR	77
C CHECK IF BOUNDARIES OF STANDARD TABLE ARE	SEAR	78
C EQUAL OR WITHIN THOSE OF INTERPOLATED TABLE.	SEAR	79
C ALTER STANDARD TABLE INDICES IF NECESSARY.	SEAR	80
3 IF(EB(NBI).GT.EA(NAI))GOTO 5	SEAR	81
4 IF(EB(NBF).LT.EA(NAF))GOTO 6	SEAR	82
NDEX=NAF-1	SEAR	83
JUMP=NBI	SEAR	84
DO 1 J=NAI,NDEX	SEAR	85
TEMP=(IA(J)-IA(J+1))/(EA(J)-EA(J+1))	SEAR	86
K=JUMP	SEAR	87
660 CONTINUE	SEAR	88
C OUTSIDE RANGE OF EB TABLE	SEAR	89
IF(EB(K).LT.EA(J+1))GOTO19	SEAR	90
IF(EB(K).GT.EA(J))GOTO19	SEAR	91
C WITHIN RANGE OR EQUAL TO END POINTS OF RANGE OF EB TABLE	SEAR	92
YA(K)=TEMP*(EB(K)-EA(J+1))+IA(J+1)	SEAR	93
IF(K.EQ.NBF)GOTO220	SEAR	94
JUMP=K+1	SEAR	95
GOTO220	SEAR	96
19 K=NBF+1	SEAR	97
220 K=K+1	SEAR	98
IF(K.LE.NBF)GOTO660	SEAR	99
1 CONTINUE	SEAR	100
WRITE(11,330) BI,NBI,BF,NBF	SEAR	101
330 FORMAT(* INITIAL BI=*,I6,* FINAL BI=*,I6/	SEAR	102
1* INITIAL BF=*,I6,* FINAL BF=*,I6)	SEAR	103
IF(INTELE.EQ.1)WRITE(12,330)	SEAR	104
BI=NBI	SEAR	105
BF=NBF	SEAR	106
IF(BI.EQ.BF)510,150	SEAR	107
510 WRITE(11,500)	SEAR	108
IF(INTELE.EQ.1)WRITE(12,500)	SEAR	109
500 FORMAT(* ONLY ONE INTERPOLATED POINT OBTAINED*)	SEAR	110
150 IF(NSWH.GT.0)GOTO150	SEAR	111
RETURN	SEAR	112
C RESET TABLE INDICIES	SEAR	113
5 NBI=NBI+1	SEAR	114
NSWH=1	SEAR	115
IF(NBI.LE.NBF)GOTO3	SEAR	116

WRITE(11,21)	SEAR 117
IF(INTELE.EQ.1)WRITE(12,21)	SEAR 118
21 FORMAT(* AI,GT,AF,NO INTERPOLATION*)	SEAR 119
GOTO 150	SEAR 120
CRESET TABLE INDICIES	SEAR 121
6 NBF=NBF-1	SEAR 122
NSWH=1	SEAR 123
IF(NBF.GE.NBI)GOTO4	SEAR 124
WRITE(11,22)	SEAR 125
IF(INTELE.EQ.1)WRITE(12,22)	SEAR 126
22 FORMAT(* AF,LT,AI,NO INTERPOLATION*)	SEAR 127
IF(INTELE.EQ.1)WRITE(12,330)BI,NBI,BF,NBF	SEAR 128
160 WRITE(11,170)NBISAVE,NBI,NBFSAVE,NBF	SEAR 129
170 FORMAT(* A-ARRAY INDICES CHANGED, INITIAL AI=*,16,* FINAL AI=*,16/	SEAR 130
1* INITIAL AF=*,16,* FINAL AF=*,16)	SEAR 131
IF(INTELE.EQ.1)WRITE(12,170)NBISAVE,NBI,NBFSAVE,NBF	SEAR 132
RETURN	SEAR 133
END	SEAR 134

SUBROUTINE GRAPH
C PLOT AN 8.5X11 INCH GRAPH ON MICROFILM OR SET-UP A GRID FOR PLOTTING

COMMON NTELE,LENGTHY,LENGTHA,IREADLK	GRAP 1
COMMON C1,C2,C3,C4	GRAP 2
COMMON NX,NA,NB,NC,NAI,NAF,AI,AF,BI,BF,CI,CF,NI,NF	GRAP 3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	GRAP 4
COMMON NSTEP,KEY100,TYPE,LINE100,LINE,FAULTLT,DUMPLT,PRNTO	GRAP 5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	GRAP 6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),ATYPE(3),ID(3),IJ,JJ,LL	GRAP 7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	GRAP 8
COMMON LASTTTL(2,4)	GRAP 9
COMMON LOCK(2)	GRAP 10
COMMON YX(6000)	GRAP 11
COMMON YY(6000)	GRAP 12
EQUIVALENCE(YX,XX)	GRAP 13
REAL XX(1),YY	SPAR 1
REAL C1,C2,C3,C4	COMY 1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	COMY 1
REAL LASTTTL	GRAP 18
INTEGER NX,NA,NB,NC,NAI,NAF,AI,AF,BI,BF,CI,CF,NI,NF	GRAP 19
INTEGER NSTEP,KEY100,TYPE,LINE100,LINE,FAULTLT,DUMPLT,PRNTO	GRAP 20
INTEGER LIMIT,ID,IJ,JJ,LL	GRAP 21
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	REAL 3
J=NAI-1	GRAP 23
C DATA SHIFT TO ELIMINATE NEGATIVES AND ZEROS	GRAP 24
DO1K=NAI,NAF	GRAP 25
IF((TYPE.EQ.2).AND.(YY(K).LE.0.0))GOTO1	GRAP 26
IF((TYPE.EQ.3).AND.(XX(K).LE.0.0))GOTO1	GRAP 27
IF((TYPE.EQ.4).AND.(YY(K).LE.0.0).OR.(XX(K).LE.0.0))GOTO1	GRAP 28
J=J+1	GRAP 29
XX(J)=XX(K)	GRAP 30
YY(J)=YY(K)	GRAP 31
1 CONTINUE	GRAP 32
NPO=NAF-J	GRAP 33
WRITE(11,33)NPO	GRAP 34
IF(INTELE.EQ.1)WRITE(12,33)NPO	GRAP 35
33 FORMAT(*0 NUMBER OF POINTS OMITTED FROM PLOT DONE BY GRAPH=*16)	GRAP 36
IF(J.LT.2)GOTO34	GRAP 37
NAF=J	GRAP 38
NN=J-NAI+1	GRAP 39
GOTO(101,102,103,102)TYPE	GRAP 40
101 IF(NX)111,112,113	GRAP 41
C***NX=+1 PLOT=****	GRAP 42
C***NX=0 SPLOT=****	GRAP 43
C***NX=-1 SET GRID=****	GRAP 44
111 CALL SPLOT(1,NN,XX(NAI),YY(NAI),48,0)	GRAP 45
GO TO 10	GRAP 46
112 CALL SPLOT(1,NN,XX(NAI),YY(NAI),42,LINE)	GRAP 47
GO TO 10	GRAP 48
113 CALL PLOT(NN,XX(NAI),1,YY(NAI),1,42,LINE)	GRAP 49
GO TO 10	GRAP 50
102 DO2 J=NAI,NAF	GRAP 51
2 YY(J)=ALOG10(YY(J))	GRAP 52
IF(TYPE.EQ.2)GOTO7	GRAP 53
103 DO3 J=NAI,NAF	GRAP 54
3 XX(J)=ALOG10(XX(J))	GRAP 55
7 IF(NX)121,122,123	GRAP 56
121 CALL SPLOT(TYPE,NN,XX(NAI),YY(NAI),48,0)	GRAP 57
GO TO 10	GRAP 58
122 CALL SPLOT(TYPE,NN,XX(NAI),YY(NAI),42,LINE)	GRAP 59
GO TO 10	GRAP 60
	GRAP 61
	GRAP 62
	GRAP 63
	GRAP 64
	GRAP 65
	GRAP 66
	GRAP 67
	GRAP 68
	GRAP 69
	GRAP 70

123 CALL PLOT(NN,XX(NAI),1,YY(NAI),1,42,LINE)	GRAP	71
GO TO 10	GRAP	72
C EMPTY PLOTTING BUFFERS	GRAP	73
10 IF(NX.LT.0)CALL EMPTY	GRAP	74
IF(NX.LT.0)RETURN	GRAP	75
CALL WLCH(70,0,40,TITLE,2)	GRAP	76
CALL WLCH(99,28,40,COMMENT,1)	GRAP	77
CALL EMPTY	GRAP	78
RETURN	GRAP	79
34 WRITE(11,35)J	GRAP	80
IF(INTELE.EQ.1)WRITE(12,35)J	GRAP	81
35 FORMAT(*ONO OF POINTS=*I6*PLOT OMITTED*)	GRAP	82
RETURN	GRAP	83
END	GRAP	84
	GRAP	85

SUBROUTINE MINMAX	MINM	1
***SUBROUTINE FOR FINDING TABLE INDICES CORRESPONDING TO EMIN AND EMAX	MINM	2
***TABLE MUST CONFORM TO LEGAL REQUIREMENTS SPECIFIED BY CHECKER	MINM	3
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LOCK(2)	BLOC	10
COMMON YX(6000)	SPAR	1
	COMY	1
	MINM	7
REAL C1,C2,C3,C4	MINM	8
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	1
REAL LASTTTL	REAL	2
	REAL	3
	MINM	10
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	MINM	11
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
INTEGER NYI,NYF	INTG	4
	MINM	13
	MINM	14
	MINM	15
DUMMY(1)=3HBAD	MINM	16
C SPECIFICATION CHECK OF EMIN,EMAX	MINM	17
IF(EMIN.EQ.0.0).AND.(EMAX.EQ.0.0))GOTO4	MINM	18
IF(EMIN.GE.EMAX)GOTO5	MINM	19
IF(EMIN.GT.YX(YI))GOTO5	MINM	20
IF(EMAX.LT.YX(YF))GOTO5	MINM	21
IF(EMIN.LT.YX(YF))GOTO10	MINM	22
11 IF(EMAX.GT.YX(YI))90,92	MINM	23
90 WRITE(11,95)EMAX,YX(YI)	MINM	24
IF(INTELE.EQ.1)WRITE(12,95)EMAX,YX(YI)	MINM	25
95 FORMAT(*EMAX CHANGED FROM*,E12.5,* TO *E12.5)	MINM	26
EMAX=YX(YI)	MINM	27
C SEARCH FOR FINAL YI AND YF	MINM	28
92 NYI=0	MINM	29
NYF=0	MINM	30
NNN=YF-1	MINM	31
C PICK EQUAL OR INTERIOR POINTS INORDER TO BE ON OR WITHIN EMIN-EMAX	MINM	32
C RANGE	MINM	33
DO 1 K=YI,NNN	MINM	34
IF(EMAX.EQ.YX(K))NYI=K	MINM	35
IF((EMAX.LT.YX(K)).AND.(EMAX.GT.YX(K+1)))NYI=K+1	MINM	36
IF((EMIN.LT.YX(K)).AND.(EMIN.GT.YX(K+1)))GOTO 2	MINM	37
IF(EMIN.EQ.YX(K))2,1	MINM	38
2 NYF=K	MINM	39
GOTO100	MINM	40
1 CONTINUE	MINM	41
IF(EMIN.EQ.YX(YF))200,5	MINM	42
200 NYF=YF	MINM	43
C ONLY ONE VALUE	MINM	44
100 IF(NYI.EQ.NYF)GOTO5	MINM	45
WRITE(11,25)YI,YF,NYI,NYF	MINM	46
IF(INTELE.EQ.1)WRITE(12,25)YI,YF,NYI,NYF	MINM	47
25 FORMAT(* INITIAL YI=*,15,* INITIAL YF=*,15/	MINM	48
1* FINAL YI=*,15,* FINAL YF=*,15)	MINM	49
YI=NYI	MINM	50
YF=NYF	MINM	51
DUMMY(1)=8HTABLE OK	MINM	52
RETURN	MINM	53
4 WRITE(11,26)	MINM	54
IF(INTELE.EQ.1)WRITE(12,26)	MINM	55
26 FORMAT(* EMIN=EMAX=0. MODULE NOT EXECUTED*/	MINM	56
1* YI AND YF UNCHANGED*)	MINM	57
RETURN	MINM	58
5 WRITE(11,6)	MINM	59
IF(INTELE.EQ.1)WRITE(12,6)	MINM	60
6 FORMAT(* EMAX OR EMIN OUT OF RANGE*)	MINM	61

```

      RETURN
10  WRITE(11,28)YX(YF)
      IF(NTELE.EQ.1)WRITE(12,28)YX(YF)
28  FORMAT(* EMIN HAS BEEN CHANGED TO YX(YF)=*,E12.5)
      EMIN=YX(YF)
      GOTO11
      END

```

```

MINM 62
MINM 63
MINM 64
MINM 65
MINM 66
MINM 67
MINM 68

```

```

      SUBROUTINE A340
      PRINTPLOT
C  PRINT A GRAPH ON THE LINE PRINTER OR TELETYPE

```

```

      COMMON NTELE,LENGTHY,LENGTHA,IREADLK
      COMMON C1,C2,C3,C4
      COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
      COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
      COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
      COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
      COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
      COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
      COMMON LASTTTL(2,4)
      COMMON LOCK(2)

      COMMON YX(6000)
      COMMON YY(6000)

```

```

      REAL C1,C2,C3,C4
      REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
      REAL LASTTTL

```

```

      INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
      INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
      INTEGER LIMIT,ID,II,JJ,LL
      INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

```

```

7100 IF((YF-YI+1).GT.1000)7300,7400
7300 WRITE(11,7500)
7500 FORMAT(* MORE THAN 1000 POINTS-PRINTER PLOT SKIPPED*)
      IF(NTELE.EQ.1)WRITE(12,7500)
      GOTO7550
C  LINE101=0,GIVES PLOT OF INDIVIDUAL POINTS
C  =1,GIVES PLOT OF POINTS WITH CONNECTING LINES
C  C4=0,GIVES NO GRID
C  =1,GIVES + AT GRID INTERSECTIONS
C  =2,GIVES FULL GRID
7400 IIC4=IFIX(C4)
      CALL PRTPLOT(YX,YY,YI,YF,LINE101,IIC4,11)
      IF(NTELE.EQ.1)CALL PRTPLOT(YX,YY,YI,YF,LINE101,IIC4,-12)
7550 RETURN
      END

```

```

A340 1
A340 2
A340 3
A340 4
A340 5
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
COMY 1
COMY 1
A340 10
A340 11
REAL 1
REAL 2
REAL 3
A340 13
A340 14
INTG 1
INTG 2
INTG 3
INTG 4
A340 16
A340 17
A340 18
A340 19
A340 20
A340 21
A340 22
A340 23
A340 24
A340 25
A340 26
A340 27
A340 28
A340 29
A340 30
A340 31
A340 32

```

```

      SUBROUTINE A341

```

```

      COMMON NTELE,LENGTHY,LENGTHA,IREADLK
      COMMON C1,C2,C3,C4
      COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
      COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
      COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
      COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
      COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
      COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
      COMMON LASTTTL(2,4)
      COMMON LOCK(2)

```

```

      COMMON YX(6000)
      COMMON YY(6000)
      COMMON AX(6000)
      COMMON AY(6000)

```

```

      REAL C1,C2,C3,C4
      REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
      REAL LASTTTL

```

```

      INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
      INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
      INTEGER LIMIT,ID,II,JJ,LL
      INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
      INTEGER NYI,NYF

```

```

A341 1
A341 2
A341 3
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
COMY 1
COMY 1
A341 10
A341 11
REAL 1
REAL 2
REAL 3
A341 13
A341 14
INTG 1
INTG 2
INTG 3
INTG 4
A341 16
A341 17
A341 18

```

C	MELDYANDA	A341	19
C	MELD CONTENTS OF A-ARRAYS WITH THOSE OF THE Y-ARRAYS.	A341	20
C	RESULTING ARRAY RESIDES IN Y-ARRAYS	A341	21
	LONG=(YF-YI+1)+(AF-AI+1)	A341	22
	IF(LONG.GT.LENGTHY)140,100	A341	23
100	L=0	A341	24
C	SHIFT Y-ARRAY CONTENTS TO TOP OF Y-ARRAYS	A341	25
	DO1201=YI,YF	A341	26
	L=L+1	A341	27
	YY(L)=YY(I)	A341	28
120	YX(L)=YX(I)	A341	29
	YI=I	A341	30
	YF=L	A341	31
	NSTART=YF+1	A341	32
	NDONE=AF-AI+1+YF	A341	33
	LONG=(YF-YI+1)+(AF-AI+1)	A341	34
	IF(LONG.GT.LENGTHY)140,160	A341	35
160	L=AI-1	A341	36
	DO1501=NSTART,NDONE	A341	37
	L=L+1	A341	38
	YX(I)=AX(L)	A341	39
150	YY(I)=AY(L)	A341	40
	YF=NDONE	A341	41
	WRITE(11,175)YI,YF	A341	42
	IF(INTELE.EQ.1)WRITE(12,175)YI,YF	A341	43
175	FORMAT(* MELDYANDA COMPLETED. FINAL YI=*,I5,* FINAL YF=*,I5)	A341	44
	GOTO7550	A341	45
140	WRITE(11,170)	A341	46
170	FORMAT(* MELDYANDA SKIPPED---Y-ARRAY STORAGE WOULD HAVE BEEN*	A341	47
	1* EXCEEDED*)	A341	48
	IF(INTELE.EQ.1)WRITE(12,170)	A341	49
	GOTO7550	A341	50
		A341	51
	ENTRY A342	A341	52
C	DELETEx=	A341	53
C	DELETE ONE POINT ENTERED IN A E12.5 FORMAT IN THE NUMERIC FIELD	A341	54
C	FIRST OCCURENCE OF POINT DELETED.	A341	55
C	TO REMOVE ALL POINTS = SPECIFIED POINT, PRECEED INSTRUCTION	A341	56
C	BY NOEQUALX.	A341	57
C	TO RESET Y-ARRAY INDICES, CALL LEGALORDER AFTER LAST DELETEx=	A341	58
C	INSTRUCTION MODULE.	A341	59
300	CMARK=5.E-6	A341	60
	DO3101=YI,YF	A341	61
	ISAVE=I	A341	62
	IF(((WORD(4,JJ)+CMARK).GT.YX(I)).AND.((WORD(4,JJ) -CMARK).LE.YX(I))	A341	63
	1)1340,310	A341	64
310	CONTINUE	A341	65
	GOTO1300	A341	66
1300	WRITE(11,1330)	A341	67
	IF(INTELE.EQ.1)WRITE(12,1330)	A341	68
1330	FORMAT(* POINT TO BE DELETED NOT FOUND*)	A341	69
	GOTO7550	A341	70
340	WRITE(11,350)YX(ISAVE),YY(ISAVE)	A341	71
	IF(INTELE.EQ.1)WRITE(12,350)YX(ISAVE),YY(ISAVE)	A341	72
350	FORMAT(* POINT DELETED-YX=*E12.5,4X*YY=*E12.5)	A341	73
	YX(ISAVE)=0.	A341	74
	YY(ISAVE)=0.	A341	75
	GOTO7550	A341	76
		A341	77
	ENTRY A345	A341	78
C	SLIDE AND SHRINK Y-ARRAYS FOR POINTER ROUTINE A348	A341	79
C	KEY16 IS SWITCHED TO 1 IF Y-ARRAYS HAVE TO BE SHRUNK	A341	80
C	VALUE OF KEY17 DETERMINES FROM WHICH END Y-ARRAYS ARE TO BE SHRUNK	A341	81
	IF((KEY100.LT.2).OR.(KEY100.GT.LENGTHY))600,603	A341	82
600	WRITE(11,605)	A341	83
	IF(INTELE.EQ.1)WRITE(12,605)	A341	84
605	FORMAT(* KEY100 LT 2 OR GT 6000-SHRINK AND */	A341	85
	1* SLIDE OPERATIONS SKIPPED, YI AND YF NOT RESET*)	A341	86
	GOTO7550	A341	87
603	KEY16=0	A341	88
	KEY17=0	A341	89
	NYI=YI	A341	90
	NYF=YF	A341	91
800	NLENGTH=NYF-NYI+1	A341	92
	IF(NLENGTH.LE.KEY100)810,820	A341	93
810	IF(KEY16.NE.0)GOTO868	A341	94
830	L=0	A341	95
C	SLIDE UP	A341	96
	DO8351=YI,YF	A341	97
	L=L+1	A341	98
	YX(L)=YX(I)	A341	99
835	YY(L)=YY(I)	A341	100
	YI=I	A341	101
	YF=L	A341	102
	WRITE(11,533)YI,YF	A341	103
	IF(INTELE.EQ.1)WRITE(12,533)YI,YF	A341	104
533	FORMAT(* DATA BLOCK SHIFTED TO TOP OF Y-ARRAYS*/	A341	105
	1* FINAL YI=*,I5,* FINAL YF=*,I5)	A341	106
	RETURN	A341	107
820	IF(KEY17.EQ.0)870,871	A341	108
C	ADJUST Y-ARRAY BOUNDS SO THAT LENGTH .LE. LENGTHA	A341	109
870	NYI=NYI+1	A341	110
	KEY16=1	A341	111
		A341	112
		A341	113

KEY17=1	A341	114
GOTO800	A341	115
871 NYF=NYF-1	A341	116
KEY16=1	A341	117
KEY17=0	A341	118
GOTO800	A341	119
868 WRITE(11,869)YI,YF,NYI,NYF	A341	120
869 FORMAT(* Y-ARRAY INDICES ALTERED TO FIT A-ARRAYS*/	A341	121
1* INITIAL YI=*,15,* INITIAL YF=*,15,5X,*FINAL YI=*,15,	A341	122
1* FINAL YF=*,15)	A341	123
IF(INTELE.EQ.2)WRITE(12,869)YI,YF,NYI,NYF	A341	124
YI=NYI	A341	125
YF=NYF	A341	126
GOTO830	A341	127
7550 RETURN	A341	128
END	A341	129

SUBROUTINE A343		A343	1
C	CLOCK MODULE	A343	2
C	PRINT CENTRAL PROCESSOR TIME IN SECONDS	A343	3
COMMON NTELE,LENGTHY,LENGTHM,IREADLK		A343	4
COMMON C1,C2,C3,C4		A343	5
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF		BLOC	6
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX		BLOC	7
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO		BLOC	8
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMHY(4)		BLOC	9
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),IJ,JJ,LL		BLOC	10
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)		BLOC	11
COMMON LASTTTL(2,4)		BLOC	12
COMMON LOCK(2)		BLOC	13
REAL C1,C2,C3,C4		SPAR	14
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT		A343	15
REAL LASTTTL		A343	16
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF		A343	17
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO		A343	18
INTEGER LIMIT,ID,IJ,JJ,LL		A343	19
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE		A343	20
CALL SECOND(XZ)		A343	21
5000	IF(INTELE.EQ.1)WRITE(12,5000)XZ	A343	22
	FORMAT(* CLOCK TIME=*,1PE11.4,* SECONDS*)	A343	23
	WRITE(11,5000)XZ	A343	24
	RETURN	A343	25
	END	A343	26

SUBROUTINE A344		A344	1
COMMON NTELE,LENGTHY,LENGTHM,IREADLK		A344	2
COMMON C1,C2,C3,C4		A344	3
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF		BLOC	4
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX		BLOC	5
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO		BLOC	6
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMHY(4)		BLOC	7
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),IJ,JJ,LL		BLOC	8
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)		BLOC	9
COMMON LASTTTL(2,4)		BLOC	10
COMMON LOCK(2)		BLOC	11
REAL C1,C2,C3,C4		SPAR	12
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT		A344	13
REAL LASTTTL		A344	14
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF		A344	15
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNO		A344	16
INTEGER LIMIT,ID,IJ,JJ,LL		A344	17
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE		A344	18
RETURN		A344	19
END		A344	20

SUBROUTINE A870

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

```

```

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

```

```

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

```

```

RETURN
END

```

```

A870 1
A870 2
A870 3
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
A870 6
A870 7
REAL 1
REAL 2
REAL 3
A870 9
A870 10
INTG 1
INTG 2
INTG 3
INTG 4
A870 12
A870 13
A870 14
A870 15

```

SUBROUTINE A872

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

```

```

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

```

```

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

```

```

RETURN
END

```

```

A872 1
A872 2
A872 3
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
A872 6
A872 7
REAL 1
REAL 2
REAL 3
A872 9
A872 10
INTG 1
INTG 2
INTG 3
INTG 4
A872 12
A872 13
A872 14
A872 15

```

SUBROUTINE A874

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

```

```

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

```

```

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

```

```

A874 1
A874 2
A874 3
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
A874 6
A874 7
REAL 1
REAL 2
REAL 3
A874 9
A874 10
INTG 1
INTG 2
INTG 3
INTG 4
A874 12
A874 13

```

RETURN	A874	14
END	A874	15

SUBROUTINE A876	A876	1
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A876	2
COMMON C1,C2,C3,C4	A876	3
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	3
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
COMMON LASTTTL(2,4)	BLOC	7
COMMON LOCK(2)	BLOC	8
	BLOC	9
	BLOC	10
	SPAR	1
REAL C1,C2,C3,C4	A876	6
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A876	7
REAL LASTTTL	REAL	1
	REAL	2
	REAL	3
	A876	9
	A876	10
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	2
INTEGER LIMIT,IO,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A876	12
	A876	13
RETURN	A876	14
END	A876	15

SUBROUTINE A888	A888	1
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A888	2
COMMON C1,C2,C3,C4	A888	3
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	3
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
COMMON LASTTTL(2,4)	BLOC	7
COMMON LOCK(2)	BLOC	8
	BLOC	9
	BLOC	10
	SPAR	1
REAL C1,C2,C3,C4	A888	6
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A888	7
REAL LASTTTL	REAL	1
	REAL	2
	REAL	3
	A888	9
	A888	10
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	2
INTEGER LIMIT,IO,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A888	12
	A888	13
RETURN	A888	14
END	A888	15

SUBROUTINE A900	A900	1
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A900	2
COMMON C1,C2,C3,C4	A900	3
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	1
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	2
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	3
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	4
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	5
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	6
COMMON LASTTTL(2,4)	BLOC	7
COMMON LOCK(2)	BLOC	8
	BLOC	9
	BLOC	10
	SPAR	1
REAL C1,C2,C3,C4	A900	6
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A900	7
REAL LASTTTL	REAL	1
	REAL	2
	REAL	3
	A900	9

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A900	10
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	1
INTEGER LIMIT,ID,II,JJ,LL	INTG	2
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	3
	INTG	4
RETURN	A900	12
END	A900	13
	A900	14
	A900	15

SUBROUTINE A902

COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A902	1
COMMON C1,C2,C3,C4	A902	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A902	3
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	1
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	2
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	3
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	4
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	5
COMMON LASTTTL(2,4)	BLOC	6
COMMON LOCK(2)	BLOC	7
	BLOC	8
	BLOC	9
	BLOC	10
	SPAR	1
	A902	6
	A902	7
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A902	9
	A902	10
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	INTG	4
RETURN	A902	12
END	A902	13
	A902	14
	A902	15

SUBROUTINE A300
EXCHANGE

	A300	1
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A300	2
COMMON C1,C2,C3,C4	A300	3
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	A300	4
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	1
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	BLOC	2
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	3
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	4
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	5
COMMON LASTTTL(2,4)	BLOC	6
COMMON LOCK(2)	BLOC	7
	BLOC	8
	BLOC	9
	BLOC	10
	SPAR	1
	A300	7
	A300	8
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	A300	10
	A300	11
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A300	13
	A300	14
IF(TYPE(1).NE.TYPE(3))GOTO900	A300	15
IF(TYPE(1).NE.5HARRAY)GOTO100	A300	16
DO11=NI,NF,NSTEP	A300	17
DO21=1,3,2	A300	18
2 CALL A200	A300	19
TEMP1=ELEMENT(1)	A300	20
ELEMENT(1)=ELEMENT(3)	A300	21
ELEMENT(3)=TEMP1	A300	22
DO31=1,3,2	A300	23
3 CALL A201	A300	24
1 CONTINUE	A300	25
GOTO 908	A300	26
	A300	27
	A300	28
100 IF(TYPE(1).NE.8HCONSTANT)GOTO200	A300	29
DO1021=1,3,2	A300	30
102 CALL A202	A300	31
TEMP1=ELEMENT(1)	A300	32
ELEMENT(1)=ELEMENT(3)	A300	33

ELEMENT(3)=TEMP1	A300	34
DO10311=1,3,2	A300	35
103 CALLA203	A300	36
GOTO 908	A300	37
200 IF(TYPE(1).NE.9HDOOLOPLMT)GOTO900	A300	38
	A300	39
	A300	40
DO20211=1,3,2	A300	41
202 CALLA204	A300	42
ITEMP1=LIMIT(1)	A300	43
LIMIT(1)=LIMIT(3)	A300	44
LIMIT(3)=ITEMP1	A300	45
DO20311=1,3,2	A300	46
203 CALLA205	A300	47
908 RETURN	A300	48
900 WRITE(11,901)	A300	49
IF(INTELE.EQ.1)WRITE(12,901)	A300	50
901 FORMAT(* TROUBLE AT EXCHANGE OR TO*)	A300	51
CALL A308	A300	52
GO TO 908	A300	53
	A300	54
	A300	55
ENTRY A301	A300	56
C TO	A300	57
IF(TYPE(1).NE.TYPE(3))GOTO990	A300	58
IF(TYPE(1).NE.5HARRAY)GOTO300	A300	59
DO251LL=NI,NF,NSTEP	A300	60
DO25211=1,3,2	A300	61
252 CALL A200	A300	62
ELEMENT(3)=ELEMENT(1)	A300	63
DO25311=1,3,2	A300	64
253 CALL A201	A300	65
251 CCNTINUE	A300	66
GO TO 908	A300	67
	A300	68
	A300	69
300 IF(TYPE(1).NE.8HCONSTANT)GOTO400	A300	70
DO30211=1,3,2	A300	71
302 CALL A202	A300	72
ELEMENT(3)=ELEMENT(1)	A300	73
DO30311=1,3,2	A300	74
303 CALLA203	A300	75
GO TO 908	A300	76
	A300	77
	A300	78
400 IF(TYPE(1).NE.9HDOOLOPLMT)GOTO 900	A300	79
DO40211=1,3,2	A300	80
402 CALL A204	A300	81
LIMIT(3)=LIMIT(1)	A300	82
DO40311=1,3,2	A300	83
403 CALLA205	A300	84
GO TO 908	A300	85
END	A300	86
SUBROUTINE A308	A308	1
C PRINT OUT OF THE COMMON REGION OF MEMORY	A308	2
	A308	3
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	A308	4
COMMON C1,C2,C3,C4	BLOC	1
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	BLOC	2
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX	BLOC	3
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	BLOC	4
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	5
COMMON NORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	6
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	7
COMMON LASTTTL(2,4)	BLOC	8
COMMON LOCK(2)	BLOC	9
	BLOC	10
	SPAR	1
REAL C1,C2,C3,C4	A308	7
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT	A308	8
REAL LASTTTL	REAL	1
	REAL	2
	REAL	3
	A308	10
	A308	11
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO	INTG	2
INTEGER LIMIT,ID,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
	A308	13
	A308	14
	A308	15
IF(INTELE.EQ.1)100,200	A308	16
100 WRITE(12,500)	A308	17
IF(DUMPLT.LE.0)CALL EXIT	A308	18
200 WRITE(11,500)	A308	19
500 FORMAT(*0*,10X,*/DUMP OF NONARRAY COMMON*/*)	A308	20
WRITE(11,1)C1,C2,C3,C4,NY,NA,NB,NC,YI,YF,AI,AF,BI,	A308	21
1BF,CI,CF,NI,NF	A308	22
WRITE(11,2)XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,	A308	23
1NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,		

2DUMPLT,PRINO	A308	24
WRITE(11,3)TITLE,COMMENT,TITLEA,TITLEB,TITLEC,	A308	25
1DUMMY	A308	26
1 FORMAT(1H110X*CONSTANTS*/*OC1=*E15.5,	A308	27
110X*C2=*E15.5,10X*C3=*E15.5,10X*C4=*E15.5	A308	28
2//11X*DOLOOP,LIMITS*/*ONY=*I6.5X,	A308	29
3*NA=*I6.5X,*NB=*I6.5X,*NC=*I6.	A308	30
45X,*YI=*I6.5X,*YF=*I6.5X*AI=*I6.	A308	31
55X,*AF=*I6/*OB1=*I6.5X,*BF=*I6.	A308	32
65X,*CI=*I6.5X*CF=*I6.5X,*NI=*I6.	A308	33
75X,*NF=*I6)	A308	34
2 FORMAT(1H0,10X,*PARAMETERS*/*O*	A308	35
1*XLIMUP=*E15.5,10X,*XLIMLR=*E15.5,10X,	A308	36
2*YLIMUP=*E15.5,10X,*YLIHLR=*E15.5/	A308	37
3*OEMIN=*E15.5,10X,*EMAX=*E15.5/	A308	38
4*ONSTEP=*I5.5X,*KEY100=*I5.5X,	A308	39
5*CY101=*I5.5X,*LINE100=*I5.5X,	A308	40
6*LINE101=*I5.5X,*FAULTLT=*I5.3X,	A308	41
7*DUMPLT=*I5.3X,*PRINO=*I5)	A308	42
3 FORMAT(1H0,10X*AFORNATS*/*OTITLE=*A10,	A308	43
110X,*COMMENT=*A10/*OTITLEA=*A10,10X	A308	44
2*TITLEB=*A10/*OTITLEC=*A10,10X	A308	45
3*DUMMY=*A10)	A308	46
WRITE(11,1011,JJ,LL,{WORD(K,JJ),K=1,8},	A308	47
1 LIMIT,TYPE,1D	A308	48
10 FORMAT(*O11=*I6.5X,*JJ=*I6.5X,	A308	49
1*LL=*I5.5X*WORD1=*A2.5X*WORD2=*A	A308	50
2A10.5X*WORD3=*A10.5X,*WORD4=*E15.5	A308	51
3/* WORD5=*A10.5X*WORD6=*A10.5X,	A308	52
4*WORD7=*A10.5X,*WORD8=*A10/	A308	53
5* LIMIT11=*I6.2X*LIMIT(2)=*I6.2X	A308	54
6*LIMIT(4)=*I6/* TYPE(1)=*A10,2X*TYPE(2)=*A10,	A308	55
72X*TYPE(3)=*A10/* ID(1)=*I6,2X*ID(2)=*I6,	A308	56
72X*ID(3)=*I6)	A308	57
WRITE(11,27)NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE,LOCK,	A308	58
1(LASTTTL(K,J),J=1,4),K=1,2)	A308	59
27 FORMAT(* NTITLE(1)=*I5.5X,*NTITLE(2)=*I5.5X,*NPTSBLK(1)=*I10,	A308	60
15X,*NPTSBLK(2)=*I6,	A308	61
15X,*TAPENO=*I3.5X,*DISKA=*I3.5X,*DISKB=*I3/* NFILE(1)=*I5.5X	A308	62
1*NFILE(2)=*I5.5X,*LOCK(1)=*I3.5X,*LOCK(2)=*I3/	A308	63
1* LASTTTL(1)=*A10,10X,*LASTTTL(2)=*A10)	A308	64
WRITE(11,85D)NTELE,LENGTHY,LENGTHA,IREADLK	A308	65
85D FORMAT(* NTELE= *I4,* LENGTHY= *,I4,*LENGTHA= *,I4,* IREADLK= *	A308	66
14////)	A308	67
IF(DUMPLT.LE.0)CALLEXIT	A308	68
RETURN	A308	69
END	A308	70

SUBROUTINE CHECKER	CHEC	1
C ROUTINE FOR CHECKING THE LEGALITY OF THE DATA. DATA IS LEGALIZED AND	CHEC	2
C REORDERED IF NECESSARY	CHEC	3
	CHEC	4
	CHEC	5
COMMON NTELE,LENGTHY,LENGTHA,IREADLK	BLOC	1
COMMON C1,C2,C3,C4	BLOC	2
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	SLOC	3
COMMON XLIMUP,XLIHLR,YLIMUP,YLIHLR,EMIN,EMAX	BLOC	4
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	BLOC	5
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	BLOC	6
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	BLOC	7
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	BLOC	8
COMMON LASTTTL(2,4)	BLOC	9
COMMON LOCK(2)	BLOC	10
	SPAR	1
COMMON YX(6000)	COMY	1
COMMON YY(6000)	COMY	1
	CHEC	1
	CHEC	1
REAL C1,C2,C3,C4	REAL	1
REAL XLIMUP,XLIHLR,YLIMUP,YLIHLR,EMIN,EMAX,ELEMENT	REAL	2
REAL LASTTTL	REAL	3
	CHEC	13
	CHEC	14
INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF	INTG	1
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRINO	INTG	2
INTEGER LIMIT,1D,II,JJ,LL	INTG	3
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	INTG	4
INTEGER CARD	CHEC	16
	CHEC	17
	CHEC	18
C NA,NB,NC, AND NY ARE USED FOR TEMPORARY STORAGE	CHEC	19
C MINIMUM NUMBER OF LEGAL POINTS = 2	CHEC	20
C ONLY ONE REORDERING ALLOWED	CHEC	21
DUMMY(1)=9HBAD TABLE	CHEC	22
WRITE(11,93)	CHEC	23
IF(NTELE.EQ.1)WRITE(12,93)	CHEC	24
93 FORMAT(* 1ST PASS THRU LEGAL CHECKER*)	CHEC	25
C COUNTER FOR NUMBER OF TIMES TABLE REORDERED	CHEC	26
KEY=0	CHEC	27
C TOTAL NUMBER OF POINTS	CHEC	28
84 NB=YF-YI+1	CHEC	29

C	COUNTER FOR BAD POINTS	CHEC	30
	NY=0	CHEC	31
C	COUNTER FOR GOOD POINTS	CHEC	32
	NA=0	CHEC	33
C	COUNTER FOR POINTS ON A CARD	CHEC	34
	J=0	CHEC	35
C	CARD COUNTER	CHEC	36
	CARD=0	CHEC	37
	IF(WORD(2,JJ).EQ.4HREAD)CARD=1	CHEC	38
	DO7 I=YI,YF	CHEC	39
	IF(WORD(2,JJ).NE.4HREAD)GOTO200	CHEC	40
C	CARD AND CARD POINT CALCULATOR	CHEC	41
	J=J+1	CHEC	42
	IF(J.GT.3)CARD=CARD+1	CHEC	43
	IF(J.GT.3) J=1	CHEC	44
C	NEGATIVE YX	CHEC	45
	200 IF((YX(I).LT.0.0).AND.(YX(I).NE.0.0))GOTO700	CHEC	46
	701 IF(I.EQ.YF)GOTO77	CHEC	47
	IF(PRTNO.EQ.0)GOTO900	CHEC	48
C	BAD ORDER OR EMBEDDED ZERO	CHEC	49
	IF(YX(I).LT.YX(I+1))WRITE(11,40)YX(I),YY(I),CARD	CHEC	50
	IF((YX(I).LT.YX(I+1)).AND.(NTELE.EQ.1))WRITE(12,40)YX(I),YY(I),	CHEC	51
	1CARD	CHEC	52
	40 FORMAT(* BAD ORDER OR EMBEDDED ZERO AT X=* E12.5,2X,*Y=*E12.5,2X,	CHEC	53
	1*CARD=*16)	CHEC	54
	900 IF(YX(I).LT.YX(I+1))NY=NY+1	CHEC	55
C	LEGAL YX=0 NOT COUNTED	CHEC	56
	77 IF(YX(I).EQ.0.0)GOTO7	CHEC	57
	500 NA=NA+1	CHEC	58
	7 CONTINUE	CHEC	59
C**NO BAD POINTS FOUND**		CHEC	60
C**ONLY ONE ENTRY**		CHEC	61
	IF((NY.EQ.0).AND.(NA.LT.2))GOTO301	CHEC	62
	IF(NY.NE.0)GOTO300	CHEC	63
	DUMMY(1)=8HTABLE OK	CHEC	64
C	CALCULATION OF FINAL YF	CHEC	65
	NC=YI+NA-1	CHEC	66
	WRITE(11,10)NA,YI,YF,NC	CHEC	67
	IF(NTELE.EQ.1)WRITE(12,10)NA,YI,YF,NC	CHEC	68
C	FINAL YF	CHEC	69
	10 FORMAT(* FINAL NO PTS=*,16,2X,*YI=*16,2X,*INITIAL YF=*,16,2X,	CHEC	70
	1*FINAL YF=*,16)	CHEC	71
	YF=NC	CHEC	72
	PRTNO=0	CHEC	73
	RETURN	CHEC	74
301	DUMMY(1)=9H.LT. 2PTS	CHEC	75
	WRITE(11,302)DUMMY(1)	CHEC	76
	IF(NTELE.EQ.1)WRITE(12,302)DUMMY(1)	CHEC	77
302	FORMAT(* BAD TABLE*A10,5X,*YF NOT RESET*)	CHEC	78
	PRTNO=0	CHEC	79
	RETURN	CHEC	80
700	IF(PRTNO.EQ.0)GOTO901	CHEC	81
	WRITE(11,49)YX(I),YY(I),CARD	CHEC	82
	IF(NTELE.EQ.1)WRITE(12,49)YX(I),YY(I),CARD	CHEC	83
49	FORMAT(* NEG X AT X=*E12.5,2X,*Y=*E12.5,2X,*CARD=*16,	CHEC	84
	1* X SET=0.*)	CHEC	85
901	YX(I)=0.0	CHEC	86
	NY=NY+1	CHEC	87
	GO TO 500	CHEC	88
300	KEY=KEY+1	CHEC	89
	IF(KEY.GT.1)GOTO370	CHEC	90
	WRITE(11,340)	CHEC	91
	IF(NTELE.EQ.1)WRITE(12,340)	CHEC	92
340	FORMAT(* TABLE REORDERED*)	CHEC	93
C	TABLE IS SORTED IN DESCENDING ORDER	CHEC	94
C	SORTING ROUTINE IS DUE TO SHELL,D.L.,COMM ACM 2 (1959)30-32	CHEC	95
	N=YF-YI+1	CHEC	96
	M=N	CHEC	97
110	M=M/2	CHEC	98
	IF(M.EQ.0)GOTO79	CHEC	99
	K=N-M	CHEC	100
	J=YI	CHEC	101
120	I=J	CHEC	102
130	IF(YX(I).LT.YX(I+M))GOTO150	CHEC	103
140	J=J+1	CHEC	104
	IF(J.GT.(K+YI-1))GOTO110	CHEC	105
	GOTO120	CHEC	106
150	TEMPX=YX(I)	CHEC	107
	TEMPY=YY(I)	CHEC	108
	YX(I)=YX(I+M)	CHEC	109
	YY(I)=YY(I+M)	CHEC	110
	YX(I+M)=TEMPX	CHEC	111
	YY(I+M)=TEMPY	CHEC	112
	I=I+M	CHEC	113
	IF(I.LT.YI)GOTO140	CHEC	114
	GOTO130	CHEC	115
79	WRITE(11,83)	CHEC	116
	IF(NTELE.EQ.1)WRITE(12,83)	CHEC	117
83	FORMAT(* 2ND PASS THRU LEGAL CHECKER*)	CHEC	118
	GOTO84	CHEC	119
370	DUMMY(1)=10H.BAD TABLE-	CHEC	120
	WRITE(11,371)DUMMY(1),KEY	CHEC	121
	IF(NTELE.EQ.1)WRITE(12,371)DUMMY(1),KEY	CHEC	122
371	FORMAT(* *A10,*REORDERING UNSUCCESSFUL AFTER *110* TRY*)	CHEC	123
	PRTNO=0	CHEC	124

RETURN
END

CHEC 125
CHEC 126

```

SUBROUTINE MPLOT
COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,DIVDEC
COMMON NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YMAX,YMIN,EMIN,EMAX
COMMON NSTEP,KEY100,IND,LINE100,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)
COMMON YY(6000)
EQUIVALENCE (YX,X),(YY,Y)

COMMON/CNE/KEY,XL,YL,LINY,ARRAY,NTELE1

REALX(1),Y(1),AX,AY,BY,CY
REAL C1,C2,C3,DIVDEC
REAL XLIMUP,XLIMLR,YMAX,YMIN,XMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,IND,IFIRST,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
DIMENSION XL(2),YL(2)
DIMENSION AX(2),AY(2)

C NNIC=INITIAL INDEX
C NNCF=FINAL INDEX
C X AND Y ARE ABSCISSA AND ORDINATE RESPECTIVELY
C N=NO. OF PTS.
C DIVDEC IS NO. UNITS (MULTIPLE OF 5 FOR LINEAR X,AX) OR DECADES PER
C PLOT. =0 FOR ONE PLOT(SPLOT)
C YMIN AND YMAX ARE ORDINATE SCALING LIMITS- IF YMIN=YMAX=0,MPLOT WILL
C DETERMINE THE ORDINATE SCALING LIMITS ITSELF.
C J1 IS THE NUMBER OF DECADES OR LINEAR DIVISIONS FOR THE X-AXIS
C ARRAY IS THE Y ARRAY IDENTIFIER-IF SET TO 1H0, PLOTTING IS DELETED
C FOR IND=2,4-FIRST DECADE LINEAR ONLY IF LINE100 IS NON-ZERO
C LINY IS THE NUMBER OF LINEARLY SCALED POINTS IN THE Y ARRAY
C FOR IND=2,4-LINEAR PORTION IS DELETED IF LINY=0
C FOR IND=2,4-LOG PORTION DELETED IF ALL POINTS LINEAR(LE.10.0)
C IPTCTX IS THE NUMBER OF POINTS PER PLOT FOR THE X ARRAY
C NTELE1=NTELE
C ARAY=2H Y
C LINY=0
C IC=NNIC
C N=NNCF
C ARRAA=1H0
C IPLTNO=0
C KEY=0
C XL(2)=-10000.
C XL(1)=+10000.
C ARAY=2H Y
C REMOVE ILLEGAL POINTS-REORDER ARRAYS-DETERMINE MIN,MAX OF ARRAYS-
C AND PRE-SCALE ARRAYS( AS APPLICABLE )
C CALL REMOV(X,Y,IND,IC,N,LINE100)
C IPTCTX=N-IC+1
C 1 IF(ARRAY.EQ.1H0) GO TO 29
C IF(DIVDEC.EQ.0.0) GO TO 14
C MULTIPLE PLOTS SCALED ACCORDING TO YMAX,YMIN AND DIVDEC
C SCALE ORDINATE ACCORDING TO YMIN AND YMAX( IF APPLICABLE )
C IF((YMIN.EQ.0.0).AND.(YMAX.EQ.0.0)) GO TO 4
C IF((IND.EQ.2).OR.(IND.EQ.4)) GO TO 2
C YL(1)=YMIN
C YL(2)=YMAX
C GO TO 4
C 2 IF(YMAX.GT.0.0) YL(2)=ALOG10(YMAX)
C IF(YMAX.LT.0.0) YMAX=10.0**YL(2)
C IF(LINY.EQ.0) GO TO 3
C YMIN=10.0*YL(1)
C GO TO 4
C 3 IF(YMIN.GT.0.0) YL(1)=ALOG10(YMIN)
C IF(YMIN.LT.0.0) YMIN=10.0**YL(1)
C 4 CONTINUE
C NPX=IC
C IF(IND.LT.3) GO TO 5
C DEFINE ABSCISSA LIMITS AND NUMBER OF DECADES FOR LOG-AXIS
C XL(2)=INT((XL(2)+4.99999)/5.1)*5
C J1=DIVDEC+0.001
C GO TO 6
C DEFINE ABSCISSA LIMITS AND NUMBER OF DIVISIONS FOR LINEAR AXIS
C 5 XL(2)=INT((XL(2)+4.99999)/5.1)*5
C J1=(DIVDEC+0.001)/5.

```

MPL1 1
MPL1 2
MPL1 3
MPL1 4
MPL1 5
MPL1 6
MPL1 7
MPL1 8
MPL1 9
MPL1 10
MPL1 11
MPL1 12
MPL1 13
MPL1 14
MPL1 15
MPL1 16
MPL1 17
MPL1 18
MPL1 19
MPL1 20
MPL1 21
MPL1 22
MPL1 23
MPL1 24
MPL1 25
MPL1 26
MPL1 27
MPL1 28
MPL1 29
MPL1 30
MPL1 31
MPL1 32
MPL1 33
MPL1 34
MPL1 35
MPL1 36
MPL1 37
MPL1 38
MPL1 39
MPL1 40
MPL1 41
MPL1 42
MPL1 43
MPL1 44
MPL1 45
MPL1 46
MPL1 47
MPL1 48
MPL1 49
MPL1 50
MPL1 51
MPL1 52
MPL1 53
MPL1 54
MPL1 55
MPL1 56
MPL1 57
MPL1 58
MPL1 59
MPL1 60
MPL1 61
MPL1 62
MPL1 63
MPL1 64
MPL1 65
MPL1 66
MPL1 67
MPL1 68
MPL1 69
MPL1 70
MPL1 71
MPL1 72
MPL1 73
MPL1 74
MPL1 75
MPL1 76
MPL1 77
MPL1 78
MPL1 79
MPL1 80
MPL1 81
MPL1 82
MPL1 83
MPL1 84
MPL1 85
MPL1 86
MPL1 87
MPL1 88

C	LIMIT NUMBER OF DIVISIONS TO 15 OR 10	MPL1	89
	IF(J1.LE.15) GO TO 6	MPL1	90
	J1=10	MPL1	91
	6 XL(1)=INT(XL(2)-DIVDEC+0.001)	MPL1	92
	7 IPTCTX=-1	MPL1	93
	IPLTNO=IPLTNO+1	MPL1	94
C	LIMIT NUMBER OF SCROLL PLOTS TO 25	MPL1	95
	IF(IPLTNO.EQ.26) GO TO 29	MPL1	96
	WRITE(11,8)XL(1),XL(2)	MPL1	97
	8 FORMAT(20XL(1)=*,E12.5,2X,*,E12.5,2X,2E12.5)	MPL1	98
	IF(INTELE.EQ.1)WRITE(12,8)XL(1),XL(2)	MPL1	99
C	DETERMINE NUMBER OF POINTS ON CURRENT SCROLL PLOT FOR (YX,YY) ARRAYS	MPL1	100
	9 IPTCTX=IPTCTX+1	MPL1	101
	IF(X(NPX).LT.XL(1)) GO TO 10	MPL1	102
	NPX=NPX+1	MPL1	103
	IF(NPX.LE.N) GO TO 9	MPL1	104
	IPTCTX=IPTCTX+1	MPL1	105
	10 WRITE(11,11)IPLTNO,IPTCTX,ARRAY	MPL1	106
	IF(INTELE.EQ.1)WRITE(12,11)IPLTNO,IPTCTX,ARRAY	MPL1	107
	11 FORMAT(22H NO. PTS. IN PLOT NO. ,I5,1H=,I5,1H FOR ARRAY ,A2)	MPL1	108
	13 IF(LINY.EQ.0) GO TO 22	MPL1	109
	GO TO 16	MPL1	110
C	SINGLE GRAPH -SPLOT - OR FIRST DECADE LINEAR ORDINATE	MPL1	111
	14 IF(LINY.EQ.0) GO TO 22	MPL1	112
	IF(IND.EQ.2) GO TO 15	MPL1	113
C	DEFINE ABSCISSA LIMITS AND NUMBER OF DECADES FOR LOG-AXIS	MPL1	114
	XL(2)=INT(XL(2)+0.99999)	MPL1	115
	XL(1)=INT(XL(1))	MPL1	116
	J1=XL(2)-XL(1)+0.001	MPL1	117
	GO TO 16	MPL1	118
C	DEFINE ABSCISSA LIMITS AND NUMBER OF DIVISIONS FOR LINEAR AXIS	MPL1	119
	15 XL(2)=INT((XL(2)+4.99999)/5.)*5	MPL1	120
	XL(1)=INT(XL(1)/5.)*5	MPL1	121
	J1=(XL(2)-XL(1)+0.001)/5.	MPL1	122
C	LIMIT NUMBER OF DIVISIONS TO 10 AND ADJUST AXIS LIMITS ACCORDINGLY	MPL1	123
	IF(J1.LE.10) GO TO 16	MPL1	124
	SAVJ=(J1-J1/2)*2	MPL1	125
	J1=10	MPL1	126
	XL(2)=XL(1)+5.*SAVJ	MPL1	127
	16 CALL ADV(1)	MPL1	128
C	NO=NUMBER OF DECADES IN LOG PORTION OF ORDINATE	MPL1	129
	YNO=LOG OF MAX ORDINATE SCALE LABEL	MPL1	130
C	IY= PLOT COORDINATE OF LINEAR-LOG ORDINATE DIVISION	MPL1	131
	ND=YL(2)+0.99999	MPL1	132
	YND=ND	MPL1	133
	IY=860*(ND-1)/ND+50	MPL1	134
	IF(IY.EQ.50) GO TO 19	MPL1	135
C	DEFINE GRID AREA AND LABEL LOG PORTION OF ORDINATE	MPL1	136
	CALL DGA(120,980,50,IY,XL(1),XL(2),YND,1.)	MPL1	137
	IF(IND.EQ.4) GO TO 17	MPL1	138
	CALL DLNLG(J1)	MPL1	139
	GO TO 18	MPL1	140
	17 CALL DLGLG	MPL1	141
	18 CALL SLLOG	MPL1	142
C	DEFINE GRID AREA AND LABEL LINEAR PORTION OF ORDINATE	MPL1	143
	19 CALL DGA(120,980,IY,910,XL(1),XL(2),10.,0.)	MPL1	144
	CALL PLT(75,907,0)	MPL1	145
	IF(IY.EQ.50) CALL PLT(68,46,1)	MPL1	146
	IF(IY.EQ.50) CALL PLT(75,46,0)	MPL1	147
	IF(IND.EQ.4) GO TO 20	MPL1	148
	CALL DLNLN(J1,10)	MPL1	149
	CALL SBLIN(J1,0)	MPL1	150
	GO TO 21	MPL1	151
	20 CALL DLGLN(10)	MPL1	152
	CALL SBLOG	MPL1	153
C	REDEFINE GRID AREA OVER FULL POINT PLOTTING RANGE	MPL1	154
	21 CALL DGA(120,980,50,910,XL(1),XL(2),YND,0.)	MPL1	155
	GO TO 23	MPL1	156
C	USE FAKE CALL TO SPLOT TO DRAW GRID AND LABEL FOR SINGLE GRAPH OR	MPL1	157
C	NON FIRST DECADE LINEAR SCROLL GRAPH	MPL1	158
	22 CALL SPLOT(IND,2,XL,YL,48,0)	MPL1	159
	23 CALL EXL	MPL1	160
	IF(DIVDEC.EQ.0.0) GO TO 24	MPL1	161
	IF(IPTCTX.EQ.0) GO TO 28	MPL1	162
C	PLOT AND LABEL (YX,YY) POINTS	MPL1	163
	24 CALL PLOT(IPTCTX,X(IC),1,Y(IC),1,44,LINE101)	MPL1	164
	CALL EXH	MPL1	165
	CALL WLCH(840,0,10,10H(YX,YY) = ,1)	MPL1	166
	CALL PLT(965,7,44)	MPL1	167
C	WRITE TITLE AND COMMENT ON GRAPH	MPL1	168
	28 CALL WLCH(200,0,40,TITLE,1)	MPL1	169
	CALL WLCH(200,28,40,COMMENT,1)	MPL1	170
	IF(DIVDEC.EQ.0.0) GO TO 29	MPL1	171
C	IF MORE POINTS ARE TO BE PLOTTED ON A SCROLL GRAPH, THEN RESET	MPL1	172
C	NECESSARY VARIABLES AND CYCLE	MPL1	173
	IF(NPX.GT.N) GO TO 29	MPL1	174
	XL(2)=XL(1)	MPL1	175
	IC=NPX	MPL1	176
	GO TO 6	MPL1	177
29	CALL EMPTY	MPL1	178
	RETURN	MPL1	179
	END	MPL1	180


```

SUBROUTINE MPL22
COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,DIVDEC
COMMON NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YMAX,YMIN,EMIN,EMAX
COMMON NSTEP,KEY100,IND,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),IJ,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

COMMON YX(6000)
COMMON YY(6000)
COMMON AX(6000)
COMMON AY(6000)
EQUIVALENCE (YX,X),(YY,Y)

COMMON/ONE/KEY,XL,YL,LINY,ARRAY,NTELE1

REAL X(1),Y(1),AX,AY,BY,CY
REAL C1,C2,C3,DIVDEC
REAL XLIMUP,XLIMLR,YMAX,YMIN,XMIN,EMAX,ELEMENT
REAL LASTTTL

INTEGER NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,IND,IFIRST,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,IJ,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
DIMENSION XL(2),YL(2)

C  NNIC=INITIAL INDEX
C  NNCF=FINAL INDEX
C  X AND Y ARE ABCISSA AND ORDINATE RESPECTIVELY
C  N=NO. OF PTS.
C  DIVDEC IS NO. UNITS (MULTIPLE OF 5 FOR LINEAR X,AX) OR DECADES PER
C  PLOT. =0 FOR ONE PLOT(SPLOT)
C  YMIN AND YMAX ARE ORDINATE SCALING LIMITS- IF YMIN=YMAX=0,MPL22 WILL
C  DETERMINE THE ORDINATE SCALING LIMITS ITSELF.
C  WORD(4,1) = 2. FOR (AX,AY) OVERLAY
C  J1 IS THE NUMBER OF DECADES OR LINEAR DIVISIONS FOR THE X-AXIS (X,
C  AX) ARRAYS
C  ARRAY, ARRAA ARE ARRAY (Y,AY) IDENTIFIERS-IF ONE (OR BOTH) IS SET
C  TO 1H0, PLOTTING IS DELETED FOR THAT ARRAY
C  FOR IND=2,4-FIRST DECADE LINEAR ONLY IF LINE100 IS NON-ZERO
C  LINY IS THE NUMBER OF LINEARLY SCALED POINTS IN THE Y, AY ARRAYS
C  FOR IND=2,4-LINEAR PORTION IS DELETED IF LINY=0
C  FOR IND=2,4-LOG PORTION DELETED IF ALL POINTS LINEAR(LE.10.0)
C  NAI, AI AND NAF, AF ARE THE INITIAL AND FINAL INDICES FOR THE AX,AY
C  ARRAYS
C  IPTCTX,IPTCTA ARE THE NUMBER OF POINTS PER PLOT FOR THE X AND AX
C  ARRAYS, RESPECTIVELY
C  NTELE1=NTELE
C  ARAY=2H Y
C  ARAA=2HAY
C  LINY=0
C  IC=NNIC
C  N=NNCF
C  ARRAA=1H0
C  IPLTNO=0
C  KEY=0
C  XL(2)=-10000.
C  XL(1)=+10000.
C  ARAY=2H Y
C  REMOVE ILLEGAL POINTS-REORDER ARRAYS-DETERMINE MIN,MAX OF ARRAYS-
C  AND PRE-SCALE ARRAYS( AS APPLICABLE )
C  CALL REMOV(X,Y,IND,IC,N,LINE100)
C  IPTCTX=N-IC+1
C  IF(WORD(4,1).NE.2.) GO TO 1
C  SAVJ=ARRAY
C  ARRAY=2HAY
C  NAI=AI
C  NAF=AF
C  CALL REMOV(AX,AY,IND,NAI,NAF,LINE100)
C  ARRAA=ARRAY
C  ARAY=SAVJ
C  IPTCTA=NAF-NAI+1
C  1 IF((ARRAY.EQ.1H0).AND.(ARRAA.EQ.1H0)) GO TO 29
C  IF(DIVDEC.EQ.0.0) GO TO 14
C  MULTIPLE PLOTS SCALED ACCORDING TO YMAX,YMIN AND DIVDEC
C  SCALE ORDINATE ACCORDING TO YMIN AND YMAX( IF APPLICABLE )
C  IF((YMIN.EQ.0.0).AND.(YMAX.EQ.0.0)) GO TO 4
C  IF((IND.EQ.2).OR.(IND.EQ.4)) GO TO 2
C  YL(1)=YMIN
C  YL(2)=YMAX
C  GO TO 4
C  2 IF(YMAX.GT.0.0) YL(2)=ALOG10(YMAX)
C  IF(YMAX.LT.0.0) YMAX=10.0**YL(2)
C  IF(LINY.EQ.0) GO TO 3
C  YMIN=10.0**YL(1)
C  GO TO 4
C  3 IF(YMIN.GT.0.0) YL(1)=ALOG10(YMIN)
C  IF(YMIN.LT.0.0) YMIN=10.0**YL(1)

```

4	CONTINUE	MPL2	96
	NPX=IC	MPL2	97
	NPA=NAI	MPL2	98
C	IF(IND.LT.3) GO TO 5	MPL2	99
	DEFINE ABSCISSA LIMITS AND NUMBER OF DECADES FOR LOG-AXIS	MPL2	100
	XL(2)=INT(XL(2)+0.99999)	MPL2	101
	J1=DIVDEC+0.001	MPL2	102
	GO TO 6	MPL2	103
C	DEFINE ABSCISSA LIMITS AND NUMBER OF DIVISIONS FOR LINEAR AXIS	MPL2	104
	5 XL(2)=(INT((XL(2)+4.99999)/5.)*5	MPL2	105
	J1=(DIVDEC+0.001)/5.	MPL2	106
C	LIMIT NUMBER OF DIVISIONS TO 15 OR 10	MPL2	107
	IF(J1.LE.15) GO TO 6	MPL2	108
	J1=10	MPL2	109
	6 XL(1)=INT(XL(2)-DIVDEC+0.001)	MPL2	110
	7 IPTCTX=-1	MPL2	111
	IPTCTA=-1	MPL2	112
	IPLTNO=IPLTNO+1	MPL2	113
C	LIMIT NUMBER OF SCROLL PLOTS TO 25	MPL2	114
	IF(IPLTNO.EQ.26) GO TO 29	MPL2	115
	WRITE(11,8)XL(1),XL(2)	MPL2	116
	8 FORMAT(*OXL(1)=*,E12.5,2X,*XL(2)=*,E12.5,2X,2E12.5)	MPL2	117
	IF(INTELE.EQ.1)WRITE(12,8)XL(1),XL(2)	MPL2	118
	IF(ARRAY.EQ.1H0) GO TO 10	MPL2	119
C	DETERMINE NUMBER OF POINTS ON CURRENT SCROLL PLOT FOR (YX,YY) ARRAYS	MPL2	120
	9 IPTCTX=IPTCTX+1	MPL2	121
	IF(X(NPA).LT.XL(1)) GO TO 10	MPL2	122
	NPX=NPX+1	MPL2	123
	IF(NPX.LE.N) GO TO 9	MPL2	124
	IPTCTX=IPTCTX+1	MPL2	125
	10 WRITE(11,11)IPLTNO,IPTCTX,ARRAY	MPL2	126
	IF(INTELE.EQ.1)WRITE(12,11)IPLTNO,IPTCTX,ARRAY	MPL2	127
	11 FORMAT(22H NO. PTS. IN PLOT NO. ,15,11H=,15,11H FOR ARRAY ,A2)	MPL2	128
	IF(ARRAA.EQ.1H0) GO TO 13	MPL2	129
C	DETERMINE NUMBER OF POINTS ON CURRENT SCROLL PLOT FOR (AX,AY) ARRAYS	MPL2	130
	12 IPTCTA=IPTCTA+1	MPL2	131
	IF(AX(NPA).LT.XL(1)) GO TO 13	MPL2	132
	NPA=NPA+1	MPL2	133
	IF(NPA.LE.NAF) GO TO 12	MPL2	134
	IPTCTA=IPTCTA+1	MPL2	135
	13 WRITE(11,11)IPLTNO,IPTCTA,ARRAA	MPL2	136
	IF(INTELE.EQ.1)WRITE(12,11)IPLTNO,IPTCTA,ARRAA	MPL2	137
	IF(LINY.EQ.0) GO TO 22	MPL2	138
	GO TO 16	MPL2	139
C	SINGLE GRAPH -SPLOT - OR FIRST DECADE LINEAR ORDINATE	MPL2	140
	14 IF(LINY.EQ.0) GO TO 22	MPL2	141
	IF(IND.EQ.2) GO TO 15	MPL2	142
C	DEFINE ABSCISSA LIMITS AND NUMBER OF DECADES FOR LOG-AXIS	MPL2	143
	XL(2)=INT(XL(2)+0.99999)	MPL2	144
	XL(1)=INT(XL(1))	MPL2	145
	J1=XL(2)-XL(1)+0.001	MPL2	146
	GO TO 16	MPL2	147
C	DEFINE ABSCISSA LIMITS AND NUMBER OF DIVISIONS FOR LINEAR AXIS	MPL2	148
	15 XL(2)=INT((XL(2)+4.99999)/5.)*5	MPL2	149
	XL(1)=INT(XL(1)/5.)*5	MPL2	150
	J1=(XL(2)-XL(1)+0.001)/5.	MPL2	151
C	LIMIT NUMBER OF DIVISIONS TO 10 AND ADJUST AXIS LIMITS ACCORDINGLY	MPL2	152
	IF(J1.LE.10) GO TO 16	MPL2	153
	SAVJ=(J1-J1/2)*2	MPL2	154
	J1=10	MPL2	155
	XL(2)=XL(1)+5.*SAVJ	MPL2	156
	16 CALL ADV(1)	MPL2	157
C	ND=NUMBER OF DECADES IN LOG PORTION OF ORDINATE	MPL2	158
C	YND=LOG OF MAX ORDINATE SCALE LABEL	MPL2	159
C	IY=PLOT COORDINATE OF LINEAR-LOG ORDINATE DIVISION	MPL2	160
	ND=YL(2)+0.99999	MPL2	161
	YND=ND	MPL2	162
	IY=860*(ND-1)/ND+50	MPL2	163
	IF(IY.EQ.50) GO TO 19	MPL2	164
C	DEFINE GRID AREA AND LABEL LOG PORTION OF ORDINATE	MPL2	165
	CALL DGA(120,980,50,IY,XL(1),XL(2),YND,1.)	MPL2	166
	IF(IND.EQ.4) GO TO 17	MPL2	167
	CALL DLNLC(J1)	MPL2	168
	GO TO 18	MPL2	169
	17 CALL DLGLG	MPL2	170
	18 CALL SLLOG	MPL2	171
C	DEFINE GRID AREA AND LABEL LINEAR PORTION OF ORDINATE	MPL2	172
	19 CALL DGA(120,980,IY,910,XL(1),XL(2),10.,0.)	MPL2	173
	CALL PLT(75,907,0)	MPL2	174
	IF(IY.EQ.50) CALL PLT(68,46,1)	MPL2	175
	IF(IY.EQ.50) CALL PLT(75,46,0)	MPL2	176
	IF(IND.EQ.4) GO TO 20	MPL2	177
	CALL DLNLC(J1,10)	MPL2	178
	CALL SBLIN(J1,0)	MPL2	179
	GO TO 21	MPL2	180
	20 CALL DLGLN(10)	MPL2	181
	CALL SBLOG	MPL2	182
C	REDEFINE GRID AREA OVER FULL POINT PLOTTING RANGE	MPL2	183
	21 CALL DGA(120,980,50,910,XL(1),XL(2),YND,0.)	MPL2	184
	GO TO 23	MPL2	185
C	USE FAKE CALL TO SPLOT TO DRAW GRID AND LABEL FOR SINGLE GRAPH OR	MPL2	186
C	NON FIRST DECADE LINEAR SCROLL GRAPH	MPL2	187
	22 CALL SPLOT(IND,2,XL,YL,48,0)	MPL2	188
	23 CALL EXL	MPL2	189
	IF(ARRAY.EQ.1H0) GO TO 26	MPL2	190

	IF(DIVDEC.EQ.0.0) GO TO 24	MPL2	191
	IF(IPTCTX.EQ.0) GO TO 25	MPL2	192
C	PLOT AND LABEL (YX,YY) POINTS	MPL2	193
	24 CALL PLOT(IPTCTX,X(IC),1,Y(IC),1.44,LINE101)	MPL2	194
	CALL EXH	MPL2	195
	CALL WLCH(840,0,10,10H(YX,YY) = .1)	MPL2	196
	CALL PLT(965,7,44)	MPL2	197
	CALL EXL	MPL2	198
	25 IF(ARRAA.EQ.1H0) GO TO 28	MPL2	199
	26 IF(DIVDEC.EQ.0.0) GO TO 27	MPL2	200
	IF(IPTCTA.EQ.0) GO TO 28	MPL2	201
C	PLOT AND LABEL (AX,AY) POINTS	MPL2	202
	27 CALL PLOT(IPTCTA,AX(NAI),1,AY(NAI),1.63,LINE101)	MPL2	203
	CALL EXH	MPL2	204
	CALL WLCH(840,28,10,10H(AX,AY) = .1)	MPL2	205
	CALL PLT(965,35,63)	MPL2	206
	28 CALL EXH	MPL2	207
C	WRITE TITLE AND COMMENT ON GRAPH	MPL2	208
	CALL WLCH(200, 8.40,TITLE,1)	MPL2	209
	CALL WLCH(200,28.40,COMMENT,1)	MPL2	210
	IF(DIVDEC.EQ.0.0) GO TO 29	MPL2	211
C	IF MORE POINTS ARE TO BE PLOTTED ON A SCROLL GRAPH, THEN RESET	MPL2	212
C	NECESSARY VARIABLES AND CYCLE	MPL2	213
	IF(NPX.GT.N) ARRAY=1H0	MPL2	214
	IF(NPA.GT.NAF) ARRAA=1H0	MPL2	215
	IF((ARRAY.EQ.1H0).AND.(ARRAA.EQ.1H0)) GO TO 29	MPL2	216
	XL(2)=XL(1)	MPL2	217
	IC=NPX	MPL2	218
	NAI=NPA	MPL2	219
	GO TO 6	MPL2	220
	29 CALL EMPTY	MPL2	221
	RETURN	MPL2	222
	END	MPL2	223

	SUBROUTINE REMOV(T,V,IND,I,IE,LINE100)	REMI	1
	COMMON/ONE/KEY,XL,YL,LIN,ARRAY,NTELE	REMI	2
	DIMENSION T(6000), V(6000), YL(2), XL(2)	REMI	3
C	NUMBER IS THE NUMBER OF ILLEGAL (LE.0.0) POINTS FOR IND=2,4	REMI	4
	NNT IS THE NUMBER OF LEGAL POINTS	REMI	5
	FOR IND=2,4 (LOG-ORDINATE CASE), SUB REMOV REMOVES THOSE ORDINATE	REMI	6
	POINTS LESS THAN OR EQUAL TO 0.0 AND REORDERS(SHIFTS) THE ORDINATE	REMI	7
	(V(I)) AND ABSCISSA(T(I)) ARRAYS	REMI	8
	SUB REMOV ALSO PRE-SCALES THE ORDINATE AND ABSCISSA ARRAYS FOR THE	REMI	9
	LOG-AXIS CASE (AND FIRST DECADE LINEAR ORDINATE)	REMI	10
C	FOR LOG ORDINATE, FIRST DECADE LINEAR IF LINE100 IS NON-ZERO	REMI	11
	NUMBER=0	REMI	12
	J=I-1	REMI	13
	DO 12 I=I,IE	REMI	14
	IF((IND.EQ.2).OR.(IND.EQ.4)).AND.(V(I).LE.0.0) GO TO 1	REMI	15
	GO TO 2	REMI	16
	1 NUMBER=NUMBER+1	REMI	17
	GO TO 12	REMI	18
	2 J=J+1	REMI	19
C	IF(KEY.NE.0) GO TO 5	REMI	20
C	INITIALIZING MIN,MAX(YL(1),YL(2)) OF ORDINATE ON FIRST CALL TO SUB	REMI	21
C	REMOV	REMI	22
	YL(1)=V(I)	REMI	23
	IF((IND.EQ.1).OR.(IND.EQ.3)) GO TO 4	REMI	24
	IF(LINE100.EQ.0) GO TO 3	REMI	25
	IF(V(I).GT.10.) GO TO 3	REMI	26
	YL(1)=0.1*V(I)	REMI	27
	GO TO 4	REMI	28
	3 YL(1)=ALOG10(V(I))	REMI	29
	4 YL(2)=YL(1)	REMI	30
	KEY=KEY+1	REMI	31
C	REORDERING(SHIFTING) THE ARRAYS-REMOVING THE ILLEGAL POINTS	REMI	32
	5 T(J)=T(I)	REMI	33
	V(J)=V(I)	REMI	34
C	PRE-SCALING(IF NECESSARY) OF ORDINATE AND ABSCISSA VALUES FOR LOG-	REMI	35
C	AXIS CASE	REMI	36
	IF(IND.EQ.1) GO TO 8	REMI	37
	IF(IND.EQ.2) GO TO 6	REMI	38
	T(J)=ALOG10(T(J))	REMI	39
	IF(IND.NE.4) GO TO 8	REMI	40
	6 IF(LINE100.EQ.0) GO TO 7	REMI	41
	IF(V(J).GT.10.) GO TO 7	REMI	42
	V(J)=0.1*V(J)	REMI	43
	LIN=LIN+1	REMI	44
	GO TO 6	REMI	45
	7 V(J)=ALOG10(V(J))	REMI	46
C	DETERMINING MIN,MAX OF ORDINATE AND ABSCISSA FOR USE IN MLOT	REMI	47
	8 IF(YL(1).LE.V(J)) GO TO 9	REMI	48
	YL(1)=V(J)	REMI	49
	9 IF(YL(2).GE.V(J)) GO TO 10	REMI	50
	YL(2)=V(J)	REMI	51
	10 IF(XL(1).LE.T(J)) GO TO 11	REMI	52
	XL(1)=T(J)	REMI	53
	11 IF(XL(2).GE.T(J)) GO TO 12	REMI	54
	XL(2)=T(J)	REMI	55
	12 CONTINUE	REMI	56
	NNT=IE-I+1-NUMBER	REMI	57

```

WRITE(11,13)NNT,NUMBER,ARRAY                                REM1 58
IF(INTELE.EQ.1)WRITE(12,13)NNT,NUMBER,ARRAY                 REM1 59
13 FORMAT(*NUMBER OF LEGAL POINTS=*,16,*NUMBER OF ILLEGAL POINTS=*, REM1 60
116,7H ARRAY, A2)                                           REM1 61
C REDEFINING THE FINAL INDEX IE-REMOVING THE NUMBER OF ILLEGAL INVOICES REM1 62
IE=IE-NUMBER                                                 REM1 63
IF((IE-11).GT.0) GO TO 15                                    REM1 64
WRITE(11,14)ARRAY                                             REM1 65
IF(INTELE.EQ.1)WRITE(12,14)ARRAY                             REM1 66
14 FORMAT(* NUMBER OF POINTS IS LESS THAN OR EQUAL TO 0. MPLOT SKIPPED REM1 67
10 FOR SET *,A2)                                             REM1 68
ARRAY=IHO                                                    REM1 69
15 CONTINUE                                                  REM1 70
RETURN                                                       REM1 71
END                                                         REM1 72

```

```

SUBROUTINE PRTPLT (XPOINT,YPOINT,KRDBGN,KROEND,KUNTRL,INDGRD,ISIZE)PRTP 1
1) PRTP 2
C PROGRAM WRITTEN BY J.C. SOLEM, LASL GROUP T-4. PRTP 3
DIMENSION NAME(113), MATRIX(6,51), LIST(31), XPOINT(1000), YPOINT(1000) PRTP 4
11000) & DIMENSION M1(20), M2(8), N1(16), M2(4), N3(2), N4(16), N5(16) PRTP 5
21(19), M1(20)/1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384 PRTP 6
34) PRTP 7
C DATA M1(01),M1(02),M1(03),M1(04),M1(05),M1(06),M1(07),M1(08),M1(09)PRTP 8
11,M1(10),M1(11),M1(12),M1(13),M1(14),M1(15),M1(16),M1(17),M1(18),MPRT 9
21(19),M1(20)/1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384 PRTP 10
34,32768,65536,131072,262144,524288/ PRTP 11
C DATA M2(01),M2(02),M2(03),M2(04),M2(05),M2(06),M2(07),M2(08)/1,10, PRTP 12
1100,1000,10000,100000,1000000,10000000/ PRTP 13
C DATA N1(01),N1(02),N1(03),N1(04),N1(05),N1(06),N1(07),N1(08),N1(09)PRTP 14
11,N1(10),N1(11),N1(12),N1(13),N1(14),N1(15),N1(16)/4H PRTP 15
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 16
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 17
C DATA K2(01),K2(02),K2(03),K2(04)/4H PRTP 18
11,K2(05),K2(06),K2(07),K2(08),K2(09)PRTP 19
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 20
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 21
C DATA K3(01),K3(02)/4H PRTP 22
11,K3(03),K3(04),K3(05),K3(06),K3(07),K3(08),K3(09)PRTP 23
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 24
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 25
C DATA K4(01),K4(02),K4(03),K4(04),K4(05),K4(06),K4(07),K4(08),K4(09)PRTP 26
11,K4(10),K4(11),K4(12),K4(13),K4(14),K4(15),K4(16)/4H PRTP 27
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 28
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 29
C DATA K5(01),K5(02),K5(03),K5(04)/4H PRTP 30
11,K5(05),K5(06),K5(07),K5(08),K5(09)PRTP 31
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 32
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 33
C DATA L2(01),L2(02),L2(03),L2(04)/4H1 PRTP 34
11,L2(05),L2(06),L2(07),L2(08),L2(09)PRTP 35
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 36
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 37
C DATA L3(01),L3(02)/4H1 PRTP 38
11,L3(03),L3(04),L3(05),L3(06),L3(07),L3(08),L3(09)PRTP 39
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 40
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 41
C DATA L4(01),L4(02),L4(03),L4(04),L4(05),L4(06),L4(07),L4(08),L4(09)PRTP 42
11,L4(10),L4(11),L4(12),L4(13),L4(14),L4(15),L4(16)/4H PRTP 43
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 44
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 45
C DATA L5(01),L5(02),L5(03),L5(04)/4H1 PRTP 46
11,L5(05),L5(06),L5(07),L5(08),L5(09)PRTP 47
2H X,4H XX,4H X,4H X X,4H XX,4H XXX,4H X,4H X X,4H X,4H X PRTP 48
3X X,4H XX,4H XX X,4H XXX,4H XXXX/ PRTP 49
C DATA N6(01),N6(02),N6(03),N6(04),N6(05),N6(06),N6(07),N6(08),N6(09)PRTP 50
11,N6(10),N6(11),N6(12),N6(13),N6(14)/4H0 PRTP 51
2H4,4H5,4H6,4H7,4H8,4H9,4H,4H+,4H2,4H3,4H-,4H PRTP 52
3 / PRTP 53
C ITAPE=IABS(ISIZE) & IF (ISIZE.EQ.0) ITAPE=1 & IF (INDGRD-1) 1,1,4 PRTP 54
DO 2 I=1,16 PRTP 55
N4(I)=K4(I) & DO 3 I=1,4 & N5(I)=K5(I) PRTP 56
N2(I)=K2(I) & N3(01)=K3(01) & N3(02)=K3(02) & IF (INDGRD.EQ.1) N5( PRTP 57
101)=4H+ & GO TO 7 PRTP 58
DO 5 I=1,16 PRTP 59
N4(I)=L4(I) & DO 6 I=1,4 & N5(I)=L5(I) PRTP 60
N2(I)=L2(I) & N3(01)=L3(01) & N3(02)=L3(02) PRTP 61
CONTINUE PRTP 62
*****SEARCH FOR MINIMUM AND MAXIMUM COORDINATES*****PRTP 63
XFIRST=XPOINT(KRDBGN) & XFINAL=XPOINT(KROEND) & YFIRST=YPOINT(KRDBGN) PRTP 64
1GN) & YFINAL=YPOINT(KROEND) & KRD=KRDBGN PRTP 65
KRD=KRD+1 & IF (XPOINT(KRD)-XFIRST) 9,10,10 PRTP 66
XFIRST=XPOINT(KRD) PRTP 67
IF (XPOINT(KRD)-XFINAL) 12,12,11 PRTP 68
XFINAL=XPOINT(KRD) PRTP 69
IF (YPOINT(KRD)-YFIRST) 13,14,14 PRTP 70
YFIRST=YPOINT(KRD) PRTP 71
IF (YPOINT(KRD)-YFINAL) 16,16,15 PRTP 72
YFINAL=YPOINT(KRD) PRTP 73
IF (KRD-KROEND) 8,17,17 PRTP 74
WRITE (ITAPE,76) & IF (ISIZE.GT.0) GO TO 19 & IF (ISIZE.LT.0) GO PRTP 75
1TO 18 & XFINAL=2.5*(XFINAL-XFIRST)+XFIRST & YFINAL=10/3*(YFINAL-YF PRTP 76
2FIRST)+YFIRST & GO TO 19 PRTP 77
XFINAL=1.25*(XFINAL-XFIRST)+XFIRST & YFINAL=1.25*(YFINAL-YFIRST)+YF PRTP 78
1FIRST PRTP 79
IF ((XFINAL-XFIRST)/100.0) 23,23,20 PRTP 80
IF (XFIRST+XFINAL) 21,22,22 PRTP 81
IF ((XFIRST/(XFIRST-XFINAL))-100000.0) 25,25,23 PRTP 82
IF ((XFINAL/(XFINAL-XFIRST))-100000.0) 25,25,23 PRTP 83
WRITE (ITAPE,77) & XFIRST=XFIRST-ABS(0.005*XFIRST) & XFINAL=XFINA PRTP 84
1L+ABS(0.005*XFINAL) & IF ((XFINAL-XFIRST)/100.0) 24,24,25 PRTP 85
XFIRST=-0.5 & XFINAL=0.5 PRTP 86

```

```

25 IF ((YFINAL-YFIRST)/100.0) 29,29,26 PRTP 76
26 IF (YFIRST+YFINAL) 27,28,28 PRTP 77
27 IF ((YFIRST/(YFIRST-YFINAL))-100000.0) 31,31,29 PRTP 78
28 IF ((YFINAL/(YFINAL-YFIRST))-100000.0) 31,31,29 PRTP 79
29 WRITE (ITAPE,78) & YFIRST=YFIRST-ABS(0.005*YFIRST) & YFINAL=YFINAL+ABS(0.005*YFINAL) & IF ((YFINAL-YFIRST)/100.0) 30,30,31 PRTP 80
30 YFIRST=-0.5 & YFINAL=0.5 PRTP 81
31 XSCALE=100.0/(XFINAL-XFIRST) & YSCALE=50.0/(YFINAL-YFIRST) & DO 32 PRTP 82
32 1 KOLUMN=1,6 & DO 32 LINE=1,51 PRTP 83
33 MATRIX(KOLUMN,LINE)=0 & IF (KONTRL) 33,33,36 PRTP 84
34 *****KONTRL = 0, GENERATE POINT PLOT***** PRTP 85
35 DO 35 KRD=KRDBGN,KREND & LASTX=1.5*(XSCALE*(XPOINT(KRD)-XFIRST)) & LINE=51.5-(YSCALE*(YPOINT(KRD)-YFIRST)) & KOLUMN=(LASTX+19)/20 PRTP 86
36 LOCATN=(20*KOLUMN)-LASTX+1 & KOMPAR=MATRIX(KOLUMN,LINE)/M1(LOCATN) PRTP 87
37 IF (KOMPAR-(2*(KOMPAR/2))) 34,34,35 PRTP 88
38 MATRIX(KOLUMN,LINE)=MATRIX(KOLUMN,LINE)+M1(LOCATN) PRTP 89
39 CONTINUE & GO TO 48 PRTP 90
40 *****KONTRL = 1, GENERATE LINE PLOT***** PRTP 91
41 LASTX=1.5*(XSCALE*(XPOINT(KRDBGN)-XFIRST)) & LINE=51.5-(YSCALE*(YPOINT(KRDBGN)-YFIRST)) & KOLUMN=(LASTX+19)/20 & LOCATN=(20*KOLUMN)-LASTX+1 & MATRIX(KOLUMN,LINE)=M1(LOCATN) & DO 47 KRD=KRDBGN,KREND PRTP 92
42 MOVEX=1.5*(XSCALE*(XPOINT(KRD)-XFIRST)) & LASTX=LASTX+MOVEX & IF (SPR) PRTP 93
43 1.5-(YSCALE*(YPOINT(KRD)-YFIRST))-LINE & JUMPX=ABS(MOVEX) & JUMPPRTP 94
44 2Y=ABS(MOVEY) & IF (JUMPX) 37,37,39 PRTP 95
45 IF (JUMPY) 47,47,38 PRTP 96
46 LAGX=0 & LAGY=0 & MULT=JUMPY & GO TO 44 PRTP 97
47 IF (JUMPY) 40,40,41 PRTP 98
48 LAGX=0 & LAGY=0 & MULT=JUMPX & GO TO 44 PRTP 99
49 LAGX=(MOVEX*JUMPY)/(2*JUMPX) & LAGY=(JUMPX*MOVEY)/(2*JUMPY) & IF (PRTP 100
50 1 JUMPX-JUMPY) 42,43,43 PRTP 101
51 MULT=JUMPX & GO TO 44 PRTP 102
52 MULT=JUMPX PRTP 103
53 DO 46 J=1,MULT & NEWX=LASTX+((J*MOVEX)+LAGX)/MULT & NEWY=LINE+((J*MOVEY)+LAGY)/MULT & KOLUMN=(NEWX+19)/20 & LOCATN=(20*KOLUMN)-LASTX+1 & KOMPAR=MATRIX(KOLUMN,NEWY)/M1(LOCATN) & IF (KOMPAR-(2*(KOMPAR/2))) 45,45,46 PRTP 104
54 MATRIX(KOLUMN,NEWY)=MATRIX(KOLUMN,NEWY)+M1(LOCATN) PRTP 105
55 CONTINUE & LASTX=NEWX & LINE=NEWY PRTP 106
56 *****PRINT THE PLOT STORAGE ARRAY***** PRTP 107
57 LOCK=0 & FIRST=YFIRST & FINAL=YFINAL & GO TO 56 PRTP 108
58 IBGN=11 & IF (ISIZE.EQ.0) IBGN=36 & IF (ISIZE.GT.0) IBGN=1 & DO 54 PRTP 109
59 1 LINE=IBGN,51 & IF (LOCK) 52,52,50 PRTP 110
60 LOCK=LOCK-1 & DO 51 KOLUMN=1,5 & INDEX5=MATRIX(KOLUMN,LINE)/16 PRTP 111
61 INDEX4=INDEX5/16 & INDEX3=INDEX4/4 & INDEX2=INDEX3/16 & INDEX1=INDEX2/16 PRTP 112
62 1EX2/16 & INDEX6=MATRIX(KOLUMN,LINE)-16*INDEX5 & INDEX5=INDEX5-16*INDEX4 PRTP 113
63 2INDEX4 & INDEX4=INDEX4-4*INDEX3 & INDEX3=INDEX3-16*INDEX2 & INDEX2=INDEX2-16*INDEX1 PRTP 114
64 3INDEX2-16*INDEX1 & LIST(6*KOLUMN)=N1(INDEX6+1) & LIST(6*KOLUMN-1)=N1(INDEX3+1) & LIST(6*KOLUMN-2)=N2(INDEX4+1) & LIST(6*KOLUMN-3)=N1(INDEX2+1) & LIST(6*KOLUMN-4)=N1(INDEX2+1) PRTP 115
65 LIST(6*KOLUMN-5)=N2(INDEX1+1) & INDEX1=1+(MATRIX(6,LINE)/524288) PRTP 116
66 LIST(31)=N3(INDEX1) & IF (ISIZE.LT.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) PRTP 117
67 LIST(25)=N3(1) & IF (ISIZE.EQ.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) LIST(21)=N3(1) & IF (ISIZE.GT.0) WRITE (ITAPE,79) (LIST(1),I=1,31) PRTP 118
68 IF (ISIZE.LT.0) WRITE (ITAPE,83) (LIST(1),I=1,25) & IF (ISIZE.EQ.0) PRTP 119
69 1) WRITE (ITAPE,84) (LIST(1),I=1,13) & GO TO 54 PRTP 120
70 LOCK=4 & DO 53 KOLUMN=1,5 & INDEX5=MATRIX(KOLUMN,LINE)/16 & INDEX4=INDEX5/16 & INDEX3=INDEX4/4 & INDEX2=INDEX3/16 & INDEX1=INDEX2/16 PRTP 121
71 INDEX6=MATRIX(KOLUMN,LINE)-16*INDEX5 & INDEX5=INDEX5-16*INDEX4 PRTP 122
72 INDEX4=INDEX4-4*INDEX3 & INDEX3=INDEX3-16*INDEX2 & INDEX2=INDEX2-16*INDEX1 PRTP 123
73 16*INDEX1 & LIST(6*KOLUMN)=N4(INDEX6+1) & LIST(6*KOLUMN-1)=N4(INDEX3+1) & LIST(6*KOLUMN-2)=N5(INDEX4+1) & LIST(6*KOLUMN-3)=N4(INDEX2+1) & LIST(6*KOLUMN-4)=N4(INDEX2+1) PRTP 124
74 LIST(6*KOLUMN-5)=N5(INDEX1+1) & INDEX1=1+(MATRIX(6,LINE)/524288) PRTP 125
75 LIST(31)=N3(INDEX1) & IF (ISIZE.LT.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) PRTP 126
76 LIST(25)=N3(1) & IF (ISIZE.EQ.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) LIST(21)=N3(1) & IF (ISIZE.GT.0) WRITE (ITAPE,83) (LIST(1),I=1,31) PRTP 127
77 IF (ISIZE.LT.0) WRITE (ITAPE,84) (LIST(1),I=1,13) & IF (ISIZE.EQ.0) PRTP 128
78 1) WRITE (ITAPE,84) (LIST(1),I=1,13) & GO TO 54 PRTP 129
79 LOCK=4 & DO 53 KOLUMN=1,5 & INDEX5=MATRIX(KOLUMN,LINE)/16 & INDEX4=INDEX5/16 & INDEX3=INDEX4/4 & INDEX2=INDEX3/16 & INDEX1=INDEX2/16 PRTP 130
80 INDEX6=MATRIX(KOLUMN,LINE)-16*INDEX5 & INDEX5=INDEX5-16*INDEX4 PRTP 131
81 INDEX4=INDEX4-4*INDEX3 & INDEX3=INDEX3-16*INDEX2 & INDEX2=INDEX2-16*INDEX1 PRTP 132
82 16*INDEX1 & LIST(6*KOLUMN)=N4(INDEX6+1) & LIST(6*KOLUMN-1)=N4(INDEX3+1) & LIST(6*KOLUMN-2)=N5(INDEX4+1) & LIST(6*KOLUMN-3)=N4(INDEX2+1) & LIST(6*KOLUMN-4)=N4(INDEX2+1) PRTP 133
83 LIST(6*KOLUMN-5)=N5(INDEX1+1) & INDEX1=1+(MATRIX(6,LINE)/524288) PRTP 134
84 LIST(31)=N3(INDEX1) & IF (ISIZE.LT.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) PRTP 135
85 LIST(25)=N3(1) & IF (ISIZE.EQ.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) LIST(21)=N3(1) & IF (ISIZE.GT.0) WRITE (ITAPE,83) (LIST(1),I=1,31) PRTP 136
86 IF (ISIZE.LT.0) WRITE (ITAPE,84) (LIST(1),I=1,13) & IF (ISIZE.EQ.0) PRTP 137
87 1) WRITE (ITAPE,84) (LIST(1),I=1,13) & GO TO 54 PRTP 138
88 LOCK=4 & DO 53 KOLUMN=1,5 & INDEX5=MATRIX(KOLUMN,LINE)/16 & INDEX4=INDEX5/16 & INDEX3=INDEX4/4 & INDEX2=INDEX3/16 & INDEX1=INDEX2/16 PRTP 139
89 INDEX6=MATRIX(KOLUMN,LINE)-16*INDEX5 & INDEX5=INDEX5-16*INDEX4 PRTP 140
90 INDEX4=INDEX4-4*INDEX3 & INDEX3=INDEX3-16*INDEX2 & INDEX2=INDEX2-16*INDEX1 PRTP 141
91 16*INDEX1 & LIST(6*KOLUMN)=N4(INDEX6+1) & LIST(6*KOLUMN-1)=N4(INDEX3+1) & LIST(6*KOLUMN-2)=N5(INDEX4+1) & LIST(6*KOLUMN-3)=N4(INDEX2+1) & LIST(6*KOLUMN-4)=N4(INDEX2+1) PRTP 142
92 LIST(6*KOLUMN-5)=N5(INDEX1+1) & INDEX1=1+(MATRIX(6,LINE)/524288) PRTP 143
93 LIST(31)=N3(INDEX1) & IF (ISIZE.LT.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) PRTP 144
94 LIST(25)=N3(1) & IF (ISIZE.EQ.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) LIST(21)=N3(1) & IF (ISIZE.GT.0) WRITE (ITAPE,83) (LIST(1),I=1,31) PRTP 145
95 IF (ISIZE.LT.0) WRITE (ITAPE,84) (LIST(1),I=1,13) & IF (ISIZE.EQ.0) PRTP 146
96 1) WRITE (ITAPE,84) (LIST(1),I=1,13) & GO TO 54 PRTP 147
97 LOCK=4 & DO 53 KOLUMN=1,5 & INDEX5=MATRIX(KOLUMN,LINE)/16 & INDEX4=INDEX5/16 & INDEX3=INDEX4/4 & INDEX2=INDEX3/16 & INDEX1=INDEX2/16 PRTP 148
98 INDEX6=MATRIX(KOLUMN,LINE)-16*INDEX5 & INDEX5=INDEX5-16*INDEX4 PRTP 149
99 INDEX4=INDEX4-4*INDEX3 & INDEX3=INDEX3-16*INDEX2 & INDEX2=INDEX2-16*INDEX1 PRTP 150
100 16*INDEX1 & LIST(6*KOLUMN)=N4(INDEX6+1) & LIST(6*KOLUMN-1)=N4(INDEX3+1) & LIST(6*KOLUMN-2)=N5(INDEX4+1) & LIST(6*KOLUMN-3)=N4(INDEX2+1) & LIST(6*KOLUMN-4)=N4(INDEX2+1) PRTP 151
101 LIST(6*KOLUMN-5)=N5(INDEX1+1) & INDEX1=1+(MATRIX(6,LINE)/524288) PRTP 152
102 LIST(31)=N3(INDEX1) & IF (ISIZE.LT.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) PRTP 153
103 LIST(25)=N3(1) & IF (ISIZE.EQ.0.A.INDEX6.LE.7.A.KOLUMN.EQ.5) LIST(21)=N3(1) & IF (ISIZE.GT.0) WRITE (ITAPE,83) (LIST(1),I=1,31) PRTP 154
104 IF (ISIZE.LT.0) WRITE (ITAPE,84) (LIST(1),I=1,13) & IF (ISIZE.EQ.0) PRTP 155
105 1) WRITE (ITAPE,84) (LIST(1),I=
```

```

58  ROUND=5.0*(10.0**KOUNT) & IF (FIRST+FINAL) 59,60,60      PRTP 171
59  FINAL=-FIRST                                                PRTP 172
60  FINAL=FINAL+ROUND & KCOEFF=ALOG10(FINAL) & IF (FINAL-1.0) 61,62,62 PRTP 173
61  KCOEFF=KCOEFF-1 & NAME(111)=N6(13) & INDEX1=-KCOEFF/10 & INDEX2=-KCOEFF PRTP 174
62  IF-10*INDEX1 & GO TO 63                                       PRTP 175
63  NAME(111)=N6(12) & INDEX1=KCOEFF/10 & INDEX2=KCOEFF-10*INDEX1 PRTP 176
64  NAME(112)=N6(INDEX1+1) & NAME(113)=N6(INDEX2+1) & DO 64 I=1,110 PRTP 177
65  NAME(11)=N6(14) & KOUNT=KCOEFF-KOUNT & IF (KOUNT-8) 66,66,65 PRTP 178
66  KOUNT=8                                                       PRTP 179
67  KCOEFF=KOUNT-KCOEFF-1 & ADD=0.0 & DO 74 I=1,11 & LEFT=(1*101-9+((9-KOUNT)/2) & IF (FIRST+ADD) 67,68,68 PRTP 180
68  NAME(LEFT-1)=N6(13) & VALUE=(ROUND-ADD)-FIRST & GO TO 59 PRTP 181
69  VALUE=(ROUND+ADD)+FIRST PRTP 182
70  INTEGR=VALUE*(10.0**KCOEFF) PRTP 183
71  INDEX1=1+(INTEGR/M2(KOUNT)) & IF (INDEX1-11) 72,71,75 PRTP 184
72  INTEGR=INTEGR-1 & GO TO 70 PRTP 185
73  NAME(LEFT)=N6(INDEX1) & NAME(LEFT+1)=N6(11) & LEFT=LEFT+2 & DO 73 I=2,KOUNT & INDEX1=KOUNT-I+1 & INDEX1=1+(INTEGR/M2(INDEX1))-((10*(INTEGR/M2(INDEX1+1))) & NAME(LEFT)=N6(INDEX1) PRTP 186
74  LEFT=LEFT+1 PRTP 187
75  ADD=ADD+STEP PRTP 188
76  IF (LOCK) 49,49,55 PRTP 189
77  FORMAT (1H1) PRTP 190
78  FORMAT (16X,51H) PLOT ERROR, ZERO XPOINT HORIZONTAL COORDINATE RANGE PRTP 191
79  FORMAT (16X,51H) PLOT ERROR, ZERO YPOINT VERTICAL COORDINATE RANGE PRTP 192
80  FORMAT (16X,10(A2,2A4),A1) PRTP 193
81  FORMAT (* MAX Y COORD=*,10A1,1HE,3A1) PRTP 194
82  FORMAT (* MAXIMUM Y COORDINATE =*,10A1,1HE,3A1) PRTP 195
83  FORMAT (1X,10A1,1HE,3A1,1X,10(A2,2A4),A1) PRTP 196
84  FORMAT (2X,8(A2,2A4),A1) PRTP 197
85  FORMAT (2X,4(A2,2A4),A1) PRTP 198
86  FORMAT (* MIN Y COORD=*,10A1,1HE,3A1) PRTP 199
87  FORMAT (* MINIMUM Y COORDINATE =*,10A1,1HE,3A1) PRTP 200
88  FORMAT (1H0,11X,110A1/9X,11(6X,1HE,3A1)) PRTP 201
89  FORMAT (1H0,84A1/1X,1HE,3A1,8(6X,1HE,3A1)) PRTP 202
90  FORMAT (1H0,44A1/1X,1HE,3A1,4(6X,1HE,3A1)) PRTP 203
91  END PRTP 204

```

SUBROUTINE PLACE C TAPE MANIPULATOR AND POSITIONER

```

COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULT1,DUMPLT,PRNO
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)

```

```

REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTTTL

```

```

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,C1,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULT1,DUMPLT,PRNO
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE

```

```

DIMENSION KEY6(2),TTL(4),T/L2(4)
DIMENSION KEY5(2)
DATA KEY6/0.0/
DATA KEY5/0.0/

```

C TOC STANDS FOR TABLE OF CONTENTS
C KEY6=1 MEANS TOC PUT ON DISKB EITHER BY NEW TAPE WRITES OR BY TRANSFER
C FROM DISK A
C KEY7=0 MEANS FIRST PASS OR THAT TOC LISTED ON PREVIOUS PASS
C DISKA=DATA TAPE
C DISKB=TEMPORARY TOC STORAGE

```

IF (KEY5(TAPENO).EQ.1) 410,414
410 WRITE(11,413)TAPENO
IF (NTELE.EQ.1) WRITE(12,413)TAPENO
413 FORMAT(* PLACE LOCKED OUT TO TAPENO=*13 )
RETURN
414 IF (KEY6(TAPENO).EQ.0) GOTO 100
GOTO 200
100 REWIND DISKA

```

BPLA 1
BPLA 2
BPLA 3
BPLA 4
BLOC 1
BLOC 2
BLOC 3
BLOC 4
BLOC 5
BLOC 6
BLOC 7
BLOC 8
BLOC 9
BLOC 10
SPAR 1
BPLA 7
BPLA 8
REAL 1
REAL 2
REAL 3
BPLA 10
BPLA 11
INTG 1
INTG 2
INTG 3
INTG 4
BPLA 13
BPLA 14
BPLA 15
BPLA 16
BPLA 17
BPLA 18
BPLA 19
BPLA 20
BPLA 21
BPLA 22
BPLA 23
BPLA 24
BPLA 25
BPLA 26
BPLA 27
BPLA 28
BPLA 29
BPLA 30
BPLA 31
BPLA 32
BPLA 33
BPLA 34
BPLA 35
BPLA 36

```

REWIND DISK
NTITLE(TAPENO)=0
NFILE(TAPENO)=1
IF(WORD(2,JJ).EQ.9)HWRITETAPE(GOTO101
GOTO102
101 IF(WORD(3,JJ).EQ.2)H**GOTO103
GOTO104
CNEWTAPE SET UP
103 KEY6(TAPENO)=1
GOTO225
102 IF(WORD(2,JJ).EQ.8)HREADTAPE(GOTO104
WRITE(11,110)
IF(INTELE.EQ.1)WRITE(12,110)
110 FORMAT(* WORD(2,JJ) INCORRECT FOR PLACE*)
RETURN
CIOC ASSEMBLY AND RECORDING ON DISK
104 REWIND DISK
WRITE(11,511)
IF(INTELE.EQ.1)WRITE(12,511)
NTITLE(TAPENO)=0
KEY5(TAPENO)=1
NFILE(TAPENO)=1
400 READ (DISK) TTL,TTL2
401 FORMAT(4A10,4A10)
IF(EOF,DISK)402,403
402 WRITE(11,498)TAPENO
IF(INTELE.EQ.1)WRITE(12,498)TAPENO
498 FORMAT(* EOF READ ON TAPENO=*13 )
RETURN
403 DECODE(1,405,TTL(1))STAR
405 FORMAT(A1)
IF(1STAR.EQ.1)H*500,501
501 WRITE(11,502)TAPENO
IF(INTELE.EQ.1)WRITE(12,502)TAPENO
502 FORMAT(* NO STAR ON TITLE OF TAPENO=*13 )
RETURN
500 DECODE(10,503,TTL2(1))IPTSBLK
503 FORMAT(110)
IF(IPTSBLK.EQ.0)504,505
504 WRITE(11,506)TAPENO
IF(INTELE.EQ.1)WRITE(12,506)TAPENO
506 FORMAT(* BLOCK LENGTH=0 ON TAPENO=*13 )
RETURN
505 WRITE(DISK,411)TTL,IPTSBLK
WRITE(11,634)TTL,IPTSBLK
IF(INTELE.EQ.1)WRITE(12,634)TTL,IPTSBLK
634 FORMAT(* 10X,4A10,110)
NTITLE(TAPENO)=NTITLE(TAPENO)+1
IF(EOF,DISK)508,508
508 CALL SKPFIL(DISK,1,1)
NFILE(TAPENO)=NFILE(TAPENO)+1
DO509 I=1,4
IF(TTL(1).EQ.LASTTTL(TAPENO,1))509,400
509 CONTINUE
KEYA(TAPENO)=1
KEY5(TAPENO)=0
411 FORMAT(4A10,110)
412 FORMAT(1H*)TABLE OF CONTENTS OF TAPE NO=*13)
511 FORMAT(* IOC WRITTEN ON DISK*)
200 REWIND DISK
ITEMP=NTITLE(TAPENO)
DO807 I=1,ITEMP
READ(DISK,411)TTL, IPTSBLK
K=0
DO805 J=1,4
K=J+4
IF(WORD(K,JJ).NE.TTL(J))GOTO807
805 CONTINUE
IF(WORD(2,JJ).EQ.9)HWRITETAPE(GOTO118
GOTO119
118 WRITE(11,120)
IF(INTELE.EQ.1)WRITE(12,120)
120 FORMAT(* DATA BLOCK ALREADY ON TAPE*)
RETURN
119 ITEM=1-NFILE(TAPENO)
IF(ITEMP.LT.0)REWIND DISK
IF(ITEMP.LT.0)ITEMP=1-1
IF(EOF,DISK)121,121
121 CALL SKPFIL(DISK,ITEMP,1)
NFILE(TAPENO)=1
C/CHECK IF CORRECT TITLE FOUND
READ (DISK) TTL, TTL2
DECODE(10,503,TTL2(1)) IPTSBLK
K=0
DO806 J=1,4
K=J+4
IF(WORD(K,JJ).NE.TTL(J))GOTO122
806 CONTINUE
NPTSBLK(TAPENO)=IPTSBLK
GOTO225
122 WRITE(11,124)TAPENO
IF(INTELE.EQ.1)WRITE(12,124)TAPENO
124 FORMAT(* ERROR ON DATA TAPE=*13 )
RETURN
907 CONTINUE

```

```

BPLA 37
BPLA 38
BPLA 39
BPLA 40
BPLA 41
BPLA 42
BPLA 43
BPLA 44
BPLA 45
BPLA 46
BPLA 47
BPLA 48
BPLA 49
BPLA 50
BPLA 51
BPLA 52
BPLA 53
BPLA 54
BPLA 55
BPLA 56
BPLA 57
BPLA 58
BPLA 59
BPLA 60
BPLA 61
BPLA 62
BPLA 63
BPLA 64
BPLA 65
BPLA 66
BPLA 67
BPLA 68
BPLA 69
BPLA 70
BPLA 71
BPLA 72
BPLA 73
BPLA 74
BPLA 75
BPLA 76
BPLA 77
BPLA 78
BPLA 79
BPLA 80
BPLA 81
BPLA 82
BPLA 83
BPLA 84
BPLA 85
BPLA 86
BPLA 87
BPLA 88
BPLA 89
BPLA 90
BPLA 91
BPLA 92
BPLA 93
BPLA 94
BPLA 95
BPLA 96
BPLA 97
BPLA 98
BPLA 99
BPLA 100
BPLA 101
BPLA 102
BPLA 103
BPLA 104
BPLA 105
BPLA 106
BPLA 107
BPLA 108
BPLA 109
BPLA 110
BPLA 111
BPLA 112
BPLA 113
BPLA 114
BPLA 115
BPLA 116
BPLA 117
BPLA 118
BPLA 119
BPLA 120
BPLA 121
BPLA 122
BPLA 123
BPLA 124
BPLA 125
BPLA 126
BPLA 127
BPLA 128
BPLA 129
BPLA 130
BPLA 131

```

```

IF (WORD(2,JJ).EQ.8HREADTAPE)GOTO122
CCHECK LAST TITLE IN TOC=NOW POSITION
DO3333I=1,4
IF (TITLE(I).NE.LASTTTL(TAPENO,I))GOTO3334
3333 CONTINUE
ITEMP=(NTITLE(TAPENO)+1)-NFILE(TAPENO)
IF (EOF(DISK4)418,418)
418 CALL SKPFIL(DISK4,ITEMP,1)
NFILE(TAPENO)=NTITLE(TAPENO)+1
GOTO225
3334 WRITE(11,3335)
IF (NTELE.EQ.1)WRITE(12,3335)
3335 FORMAT(* LAST TITLE IN TOC AND ON DATA TAPE DONT AGREE*)
RETURN
225 DUMMY(3)=4HGOOD
RETURN

ENTRY A399
CLIST TABLE OF CONTENTS
IF (LOCK(TAPENO).EQ.0)GOTO910
CALL DATE(ZZ)
WRITE(11,412)TAPENO
IF (NTELE.EQ.1)WRITE(12,412)TAPENO
WRITE(11,863)ZZ
IF (NTELE.EQ.1)WRITE(12,863)ZZ
863 FORMAT(* DATE LISTED *A10//)
REWINDDISKB
ITEMP=NTITLE(TAPENO)
DO92I=1,ITEMP
READ(DISK4,411)TTL,IPTSBLK
IF (NTELE.EQ.1) WRITE (12,634) TTL,IPTSBLK
92 WRITE(11,634)TTL,IPTSBLK
WRITE(11,700)
IF (NTELE.EQ.1)WRITE(12,700)
700 FORMAT(1H1)
LOCK(TAPENO)=0
RETURN
910 WRITE(11,911)
IF (NTELE.EQ.1)WRITE(12,911)
911 FORMAT(* BLANK DISK OR ALREADY LISTED TOC*)
RETURN
END

```

```

SUBROUTINE TANDCRD
COMMON NTELE,LENGTHY,LENGTHA,IREADLK
COMMON C1,C2,C3,C4
COMMON NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
COMMON XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX
COMMON NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLY,DUMPLT,PRTNO
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LCK(2)

COMMON YX(6000)
COMMON YY(6000)

```

```
REAL C1,C2,C3,C4
REAL XLIMUP,XLIMLR,YLIMUP,YLIMLR,EMIN,EMAX,ELEMENT
REAL LASTITI
```

```

INTEGER NY,NA,NB,NC,YI,YF,AI,AF,BI,BF,CI,CF,NI,NF
INTEGER NSTEP,KEY100,KEY101,LINE100,LINE101,FAULTLT,DUMPLT,PRNTNO
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
INTEGER CARD

```

```
DIMENSION KEY(2),KEY4(2) ,TTL(4)
DIMENSION TRWRD(6)
DATA KEY/0,0/,KEY4/0,0/
```

C READ DATA CARDS OR MAGNETIC TAPE WRITTEN IN TELERAY FORMAT

```

CARD=1
DUMMY(3)=3HBAD
IF(IREADLK.EQ.1)900,901
900 WRITE(1,904)
IF(NTELE.EQ.1)WRITE(12,904)
904 FORMAT(* READCARDS AND READTAPE MODULE LOCKED OUT*)
RETURN
901 IF(WORD(2,JJ).EQ.8HREADTAPE)137,138
C READ TAPE ROUTINE
C POSITION TAPE
137 CALL PLACE
C CHECK IF POSITION IS OK
IF(DUMMY(3).NE.4HGOOD)RETURN

```


DUMMY(3)=3HBAD	BTAN	34
ITEMP=NPTSBLK(TAPENO)	BTAN	35
IF(NPTSBLK(TAPENO).GT.LENGTHY)GOTO141	BTAN	36
C READ DATA BLOCK	BTAN	37
READ (DISKA) (YX(I),YY(I), I=1, ITEMP)	BTAN	38
3 FORMAT(11E12.5)	BTAN	39
C LOAD YI AND YF	BTAN	40
YI=1	BTAN	41
YF=NPTSBLK(TAPENO)	BTAN	42
C NORMAL EXIT	BTAN	43
GOTO4411	BTAN	44
140 IF(CARD.EQ.1)GOTO144	BTAN	45
YI=1	BTAN	46
YF=3*CARD-3	BTAN	47
4411 DUMMY(3)=10HTABLE READ	BTAN	48
C NORMAL EXIT	BTAN	49
RETURN	BTAN	50
141 WRITE(11,143)	BTAN	51
IF(INTELE.EQ.1)WRITE(12,143)	BTAN	52
143 FORMAT(* NUMBER OF POINTS EXCEED LENGTHY. IF INSTRUCTION IS READ,*	BTAN	53
1* THEN DATA*/3X,*ARE TRUNCATED TO LENGTHY PAIRS*)	BTAN	54
WRITE(11,904)	BTAN	55
IF(INTELE.EQ.1)WRITE(12,904)	BTAN	56
IREADLK=1	BTAN	57
GOTO140	BTAN	58
144 IF(WORD(2,JJ).EQ.8HREADTAPE)RETURN	BTAN	59
WRITE(11,145)	BTAN	60
IF(INTELE.EQ.1)WRITE(12,145)	BTAN	61
145 FORMAT(* NO POINTS,ONLY -1. OR EOF CARD, YI AND YF NOT RESET*)	BTAN	62
C ERROR EXIT	BTAN	63
RETURN	BTAN	64
C READ CARDS ROUTINE	BTAN	65
138 IAB=LENGTHY/3	BTAN	66
146 READ(10,8)(TRWRD(I),I=1,6)	BTAN	67
8 FORMAT(6E12.5)	BTAN	68
IF(EOF,10)140,920	BTAN	69
920 IF(TRWRD(1).EQ.-1.)GOTO140	BTAN	70
IF(CARD.GT.IAB)GOTO141	BTAN	71
DO 804 I=1,3	BTAN	72
K=2*I-1	BTAN	73
L=2*I	BTAN	74
YX(3*CARD-3+I)=TRWRD(K)	BTAN	75
804 YY(3*CARD-3+I)=TRWRD(L)	BTAN	76
CARD=CARD+1	BTAN	77
GOTO146	BTAN	78
ENTRY TWRITE	BTAN	79
CTAPE WRITING ROUTINE FOR TAPE NO=TAPENO	BTAN	80
CDISKA=DATA TAPE	BTAN	81
CDISKB=TEMPORARY TABLE OF CONTENTS STORAGE	BTAN	82
CKEY=0,TAPE NOT WRITTEN OR FIRST PASS THRU ROUTINE OR BOTH	BTAN	83
CIPTSBLK=NUMBER OF DATA POINTS IN A DATA BLOCK	BTAN	84
CKE4=1 MEANS TAPE IS FULL	BTAN	85
DUMMY(3)=3HBAD	BTAN	86
IF(KEY4(TAPENO).NE.1)GOTO132	BTAN	87
WRITE(11,134)TAPENO	BTAN	88
IF(INTELE.EQ.1)WRITE(12,134)TAPENO	BTAN	89
134 FORMAT(* TAPENO=*I5 ,5X,*IS FULL*)	BTAN	90
RETURN	BTAN	91
C POSITION TAPE	BTAN	92
132 CALL PLACE	BTAN	93
IF(DUMMY(3).NE.4HGOOD)RETURN	BTAN	94
C IF TAPE PROPERLY POSITIONED,CONTINUE	BTAN	95
DUMMY(3)=3HBAD	BTAN	96
C REWIND TABLE OF CONTENTS AND CHECK FOR NEW OR OLD DATA TAPE	BTAN	97
REWIND DISKB	BTAN	98
IF(WORD(3,JJ).EQ.2H**)FTELEFT=2200.0	BTAN	99
IF(WORD(3,JJ).EQ.2H**)GOTO320	BTAN	100
IF(NTITLE(TAPENO).EQ.0)GOTO133	BTAN	101
CAMOUNT OF TAPE ALREADY WRITTEN	BTAN	102
SUM=0.	BTAN	103
C CALCULATE LENGTH OF TAPE IN USE AND POSITION TABLE OF CONTENTS	BTAN	104
LIMIT=NTITLE(TAPENO)	BTAN	105
DO1002I=1,LIMIT	BTAN	106
READ(DISK8,1001)TTL,IPTSBLK	BTAN	107
C THE 10H CONSTANT IS A PATCH TO COVER A SYSTEMS BUG IN THE COMPUTER.	BTAN	108
1001 FORMAT(4A10,110,10HABCDEFGHJ)	BTAN	109
IF(EOF,DISKB)133,448	BTAN	110
133 WRITE(11,449)TAPENO	BTAN	111
IF(INTELE.EQ.1)WRITE(12,449)TAPENO	BTAN	112
449 FORMAT(* FOR TAPE*I3,* NTITLE=0 OR DISK INCORRECTLY WRITTEN*)	BTAN	113
RETURN	BTAN	114
448 IF(IPTSBLK.EQ.0)GOTO133	BTAN	115
C NUMBER OF DATA WORDS PER DATA BLOCK	BTAN	116
TEMP7=FLOAT(IPTSBLK)*2.	BTAN	117
C NUMBER OF CHARACTERS OF BINARY FILE	BTAN	118
TEMP=TEMP7*10.+80.	BTAN	119
CFILE LENGTH IN INCHES FOR 556 BIT DENSITY	BTAN	120
TEMP=TEMP/556.+(TEMP7/512.+2.)*0.75+4.0	BTAN	121
CINCHES OF TAPE USED	BTAN	122
1002 SUM=SUM+TEMP	BTAN	123
CASSUME 2200 USEFUL F. ON NEW,CERTIFIED REEL	BTAN	124
SUM=SUM/12.	BTAN	125
FTELEFT=2200.0-SUM	BTAN	126
	BTAN	127
	BTAN	128

```

WRITE(11,500)TAPENO,FTLEFT
IF(INTELE.EQ.1)WRITE(12,500)TAPENO,FTLEFT
500 FORMAT(* USEFUL TAPE LEFT ON TAPENO=*I3,* BEFORE NEW BLOCK IS *
C 1*WRITTEN*/ * IS=*,E12.5,* FT*)
C AMOUNT OF TAPE NEEDED FOR NEW DATA BLOCK
C TOTAL NUMBER OF WORDS IN NEW DATA BLOCK
320 TEMP7=FLOAT(YF-YI+1)*2.
C NUMBER OF CHARACTERS IN DATA BLOCK
TEMP=TEMP7*10.+80.
C NUMBER OF INCHES OF TAPE REQUIRED
TEMP=TEMP/556.+(TEMP7/512.+2.)*0.75+4.0
TEMP=TEMP/12.0
IF(TEMP.LT.FTLEFT)GOTO438
WRITE(11,98)TAPENO
IF(INTELE.EQ.1)WRITE(12,98)TAPENO
98 FORMAT(* NEW BLOCK WON'T FIT ON TAPE*I3)
KEY4(TAPENO)=1
C LIST TABLE OF CONTENTS-NO SPACE LEFT ON TAPE
WORD(4,JJ)=1.
GOTO90
C NUMBER OF POINTS PER BLOCK
438 IPTSBLK=YF-YI+1
ENCODE (10,1010,TTL(1)) IPTSBLK
1010 FORMAT(110)
C WRITE 8-WORD RECORD OF DATA BLOCK TITLE AND NUMBER OF POINTS
WRITE (DISKA) (WORD(K,JJ), K=5,8), TTL(1), (WORD(KK,JJ), KK=5,7)
C WRITE DATA BLOCK AS ONE LOGICAL RECORD
WRITE (DISKA) (YX(K),YY(K), K=YI,YF)
C ENDFILE DATA RECORD
ENDFILEDISKA
C WRITE DATA BLOCK TITLE IN TABLE OF CONTENTS
WRITE (DISKB,1001) (WORD(K,JJ), K=5,8),IPTSBLK
K=0
C LOAD LAST TITLE WITH DATA BLOCK TITLE
D07501=1,4
K=1+4
750 LASTTTL(TAPENO,1)=WORD(K,JJ)
C UNLOCK TABLE OF CONTENTS LISTER OF ENTRY A399
LOCK(TAPENO)=1
C INCREMENT NUMBER OF TITLES AND NUMBER OF FILES
NTITLE(TAPENO)=NTITLE(TAPENO)+1
NFILE(TAPENO)=NFILE(TAPENO)+1
C SET TAPE STATUS SWITCH TO TAPE WRITTEN
KEY(TAPENO)=1
DUMMY(3)=4HGOOD
90 IF(KEY(TAPENO).EQ.0)RETURN
IF(WORD(4,JJ).EQ.1.)CALLA399
RETURN
END

```

```

BTAN 129
BTAN 130
BTAN 131
BTAN 132
BTAN 133
BTAN 134
BTAN 135
BTAN 136
BTAN 137
BTAN 138
BTAN 139
BTAN 140
BTAN 141
BTAN 142
BTAN 143
BTAN 144
BTAN 145
BTAN 146
BTAN 147
BTAN 148
BTAN 149
BTAN 150
BTAN 151
BTAN 152
BTAN 153
BTAN 154
BTAN 155
BTAN 156
BTAN 157
BTAN 158
BTAN 159
BTAN 160
BTAN 161
BTAN 162
BTAN 163
BTAN 164
BTAN 165
BTAN 166
BTAN 167
BTAN 168
BTAN 169
BTAN 170
BTAN 171
BTAN 172
BTAN 173
BTAN 174
BTAN 175
BTAN 176
BTAN 177

```

```

SUBROUTINE CALCOMP
COMMON MTELE,LENGTHY,LENGTHA,IREADLK
COMMON GRID,C2,C3,DIVDEC
COMMON NY,NA,NB,NC,NNIC,NNCF,A1,AF,B1,BF,C1,CF,N1,NF
COMMON XLIMUP,XLIMLR,YMAX,YMIN,EMIN,EMAX
COMMON NSTEP,KEY100,IND,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0
COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)
COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL
COMMON NTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)
COMMON LASTTTL(2,4)
COMMON LOCK(2)
COMMON X(6000),Y(6000)
COMMON ONE/ KEY,XL,YL,LINY,ARRAY,MTELE
REAL X,Y,AX,AY,BY,CY
REAL GRID,C2,C3,DIVDEC
REAL XLIMUP,XLIMLR,YMAX,YMIN,XMIN,EMAX,ELEMENT
REAL LASTTTL
INTEGER NY,NA,NB,NC,NNIC,NNCF,A1,AF,B1,BF,C1,CF,N1,NF
INTEGER NSTEP,KEY100,IND,IFIRST,LINE101,FAULTLT,DUMPLT,PRTN0
INTEGER LIMIT,ID,II,JJ,LL
INTEGER NTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE
DIMENSION XL(2),YL(2)
DIMENSION AX(2),AY(2)
C NEGATIVE LINEAR OPTION
C NNIC=INITIAL INDEX
C NNCF=FINAL INDEX
C X AND Y ARE ABSCISSA AND ORDINATE RESPECTIVELY
C N=NO. OF PTS.
C DIVDEC IS INCHES/DECADE OR INCHES/(5 UNITS) FOR EXTENDED ABSCISSA. =0CALL
C FOR STANDARD LENGTH ABSCISSA(17 INCHES)
C YMIN AND YMAX ARE ORDINATE SCALING LIMITS- IF YMIN=YMAX=0,MPL0T WILL
C DETERMINE THE ORDINATE SCALING LIMITS ITSELF.
C XJ1 IS THE LENGTH OF THE ABSCISSA IN INCHES
C YJ1 IS THE LENGTH OF THE ORDINATE IN INCHES
C ARRAY IS Y ARRAY IDENTIFIER-IF SET TO 1HO, PLOTTING IS DELETED
C FOR IND=2,4-FIRST DECADE POSITIVE LINEAR IF LINE100 IS +1--FIRST
C DECADE NEGATIVE OPTION IF LINE100=-1
C LINY INDICATES FIRST DECADE LINEAR DATA(NEGATIVE,POSITIVE) FOR
C LOGARITHMIC ORDINATE
C FOR IND=2,4-LINEAR PORTION IS DELETED IF LINY=0
C FOR IND=2,4-LOG PORTION DELETED IF ALL POINTS LINEAR(.LE.10.0)

```

```

CALL 1
CALL 2
CALL 3
CALL 4
CALL 5
CALL 6
CALL 7
CALL 8
CALL 9
CALL 10
CALL 11
COM2 1
CALL 13
CALL 14
CALL 15
CALL 16
CALL 17
CALL 18
CALL 19
CALL 20
CALL 21
CALL 22
CALL 23
CALL 24
CALL 25
CALL 26
CALL 27
CALL 28
CALL 29
CALL 30
CALL 31
CALL 32
CALL 33
CALL 34
CALL 35
CALL 36
CALL 37
CALL 38
CALL 39
CALL 40
CALL 41

```

	IF (TYPE(2).NE.7HCALCOMP) CALL PLOTS (7)	CALI	42
	TYPE(2)=7HCALCOMP	CALI	43
	MTELE=NTELE	CALI	44
	XJ1=17.	CALI	45
	J1=XJ1+0.001	CALI	46
	YJ1=10.	CALI	47
	J2=YJ1+0.001	CALI	48
	FY=0.	CALI	49
	YY=0.	CALI	50
	ARRAY=2H Y	CALI	51
	LINY=0	CALI	52
	IC=NNIC	CALI	53
	N=NNCF	CALI	54
	KEY=0	CALI	55
	XL(2)=-10000.	CALI	56
	XL(1)=+10000.	CALI	57
	ARRAY=2H Y	CALI	58
C	REMOVE ILLEGAL POINTS-REORDER ARRAYS-DETERMINE MIN,MAX OF ARRAYS-	CALI	59
C	AND PRE-SCALE ARRAYS(AS APPLICABLE)	CALI	60
	CALL REMOV (X,Y,IND,IC,N,LINE100)	CALI	61
	IPCTCX=N-IC+1	CALI	62
C	IF DATA SET IS EMPTY, SKIP PLOT	CALI	63
	IF (ARRAY.EQ.1H0) GO TO 52	CALI	64
	IF (DIVDEC.EQ.0.0) GO TO 6	CALI	65
C	VARIABLE LENGTH PLOT DETERMINED BY DIVDEC	CALI	66
C	SCALE ORDINATE ACCORDING TO YMIN AND YMAX(IF APPLICABLE)	CALI	67
	IF ((YMIN.EQ.0.0).AND.(YMAX.EQ.0.0)) GO TO 3	CALI	68
	IF ((IND.EQ.2).OR.(IND.EQ.4)) GO TO 2	CALI	69
	YL(1)=YMIN	CALI	70
	YL(2)=YMAX	CALI	71
	GO TO 3	CALI	72
2	IF (YMAX.GT.0.0) YL(2)=YMAX	CALI	73
	IF (LINY.NE.0) GO TO 3	CALI	74
	IF (YMIN.GT.0.0) YL(1)=YMIN	CALI	75
3	IF (IND.LT.3) GO TO 4	CALI	76
C	DEFINE ABSCISSA LIMITS AND NUMBER OF INCHES FOR LOG-AXIS	CALI	77
C	DIVDEC IS IN INCHES/DECADE	CALI	78
	XL2=INT(ALOG10(XL(2))+0.99999)	CALI	79
	XL1=INT(ALOG10(XL(1)))	CALI	80
	J1=(XL2-XL1+0.001)*DIVDEC	CALI	81
	GO TO 5	CALI	82
C	DEFINE ABSCISSA LIMITS AND NUMBER OF INCHES FOR LINEAR AXIS	CALI	83
C	DIVDEC IS IN INCHES/(5 UNITS)	CALI	84
4	XL2=INT((XL(2)+4.99999)/5.)*5	CALI	85
	XL1=INT(XL(1)/5.)*5	CALI	86
	J1=((XL2-XL1+0.001)/5.)*DIVDEC	CALI	87
C	LIMIT LENGTH OF ABSCISSA TO 156 INCHES(13 FEET)	CALI	88
5	IF (J1.GT.156) J1=156	CALI	89
	XJ1=J1	CALI	90
C	FOR LOG ORDINATE, IF LINE100=0, OR IF NO POINTS IN FIRST DECADE(0.	CALI	91
C	10.), GO TO NORMAL PLOT	CALI	92
6	IF (LINY.NE.0) GO TO 7	CALI	93
	INDY=+2	CALI	94
	IF (IND.EQ.2.OR.IND.EQ.4) INDY=-2	CALI	95
C	DETERMINE SCALE AND CONSTANTS FOR ORDINATE AXIS	CALI	96
	CALL SCALE (YL,FY,YIY,YJ1,INDY,1)	CALI	97
	IF (INDY.LT.0) J2=YIY*YJ1+0.001	CALI	98
	IF (INDY.LT.0) GO TO 11	CALI	99
	YINCR=1.	CALI	100
	JY1=YJ1	CALI	101
C	REDUCE LINEAR SCALE TO SMALLEST POSSIBLE INCREMENTS AND CORRESPOND	CALI	102
C	POWER OF TEN	CALI	103
	YLOG=ALOG10(ABS(YIY))	CALI	104
	IF (YLOG.LT.0.) YLOG=YLOG-1.	CALI	105
	YLOG=AIN(YLOG)	CALI	106
	YIY=YIY*10.**(-YLOG)	CALI	107
	FY=FY*10.**(-YLOG)	CALI	108
	CALL NUMBER (-0.35,-0.05,.10,FY,0.,-1)	CALI	109
	GO TO 9	CALI	110
C	DETERMINE SCALE AND CONSTANTS-FIRST DECADE LINEAR ORDINATE	CALI	111
7	IF (YL(2).GT.1) GO TO 8	CALI	112
	J2=-1	CALI	113
	FY=1.	CALI	114
	YLOG=0.	CALI	115
	YIY=YJ1	CALI	116
	YINCR=YJ1/10.	CALI	117
	JY1=10	CALI	118
	CALL SYMBOL (-0.35,-0.05,.10,3H-10,0.,3)	CALI	119
	GO TO 9	CALI	120
8	J2=ALOG10(YL(2))-0.000001	CALI	121
	FY=1.	CALI	122
	YLOG=0.	CALI	123
	YIY=YJ1	CALI	124
	K2=J2+1	CALI	125
	JY1=10	CALI	126
	IF (LINY.LT.0) JY1=20	CALI	127
	IF (LINY.LT.0) CALL SYMBOL (-0.35,-0.05,.10,3H-10,0.,3.	CALI	128
	IF (LINY.GT.0) CALL SYMBOL (-0.17,-0.05,.10,1H0,0.,1)	CALI	129
	IF (LINY.LT.0) K2=K2+1	CALI	130
	YIY=YIY/FLOAT(K2)	CALI	131
	YINCR=YIY/10.	CALI	132
	IF (J2.GE.1) YIY=1./YIY	CALI	133
C	DRAW AND SCALE LEFT-HAND ORDINATE--LINEAR PORTION	CALI	134
9	CALL PLT2 (0.,0.,3)	CALI	135
	DO 10 I=1,JY1	CALI	136

	IF (L1NY.EQ.0) FY=FY+YIY	CALL 137
	YY=YY+YINCR	CALL 138
	CALL PLTZ (0.,YY,2)	CALL 139
	CALL PLTZ (0.1,YY,2)	CALL 140
	IF (1.EQ.JY1) GO TO 10	CALL 141
	IF (L1NY.EQ.-1.AND.1.EQ.10) CALL SYMBOL (-0.17,YY-0.05,.10,1H0,0.,	CALL 142
	11)	CALL 143
	IF (L1NY.EQ.0) CALL NUMBER (-0.35,YY-.05,.10,FY,0.,-1)	CALL 144
	CALL PLTZ (0.,YY,3)	CALL 145
10	CONTINUE	CALL 146
	IF (L1NY.NE.0.AND.J2.GT.0) GO TO 11	CALL 147
	IF (J2.EQ.-1) CALL SYMBOL (-0.17,YJ1-.05,.10,1H0,0.,1)	CALL 148
	IF (J2.EQ.0) CALL SYMBOL (-0.25,YJ1-.05,.10,2H10,0.,2)	CALL 149
	IF (L1NY.EQ.0) CALL SYMBOL (-0.45,YJ1-.05,.10,1HE,0.,1)	CALL 150
	IF (L1NY.EQ.0) CALL NUMBER (-0.30,YJ1-.05,.10,YLOG,0.,-1)	CALL 151
	CALL PLTZ (0.,YJ1,3)	CALL 152
	GO TO 14	CALL 153
C	DRAW AND SCALE LEFT-HAND ORDINATE-LOG PORTION	CALL 154
11	YLOG=FY+0.0001	CALL 155
	CALL SYMBOL (-0.35,YY-.05,.10,2H10,0.,2)	CALL 156
	CALL NUMBER (-0.15,YY,.07,YLOG,0.,-1)	CALL 157
	CALL PLTZ (0.,YY,3)	CALL 158
	YIY=1./YIY	CALL 159
	DO 13 I=1,J2	CALL 160
	YLOG=YLOG+1.	CALL 161
	YD=YY	CALL 162
	DO 12 K=2,10	CALL 163
	YY=YD+ALOG10(FLOAT(K))*YIY	CALL 164
	CALL PLTZ (0.,YY,2)	CALL 165
	CALL PLTZ (0.1,YY,2)	CALL 166
12	CALL PLTZ (0.,YJ1,2)	CALL 167
	CALL SYMBOL (-0.35,YY-.05,.10,2H10,0.,2)	CALL 168
	CALL NUMBER (-0.15,YY,.07,YLOG,0.,-1)	CALL 169
	CALL PLTZ (0.,YY,3)	CALL 170
13	CONTINUE	CALL 171
14	XX=0.	CALL 172
	CALL SYMBOL (1.0,YJ1+0.25,.10,TITLE,0.,40)	CALL 173
	CALL SYMBOL (1.0,YJ1+0.05,.10,COMMENT,0.,40)	CALL 174
	CALL PLTZ (0.,YJ1,3)	CALL 175
	IF (IND.GT.2) GO TO 16	CALL 176
C	SCALE AND DRAW UPPER ABSCISSA AXIS-LINEAR	CALL 177
	CALL SCALE (XL,FX,XIX,XJ1,2,-1)	CALL 178
	DO 15 I=1,J1	CALL 179
	XX=XX+1.	CALL 180
	CALL PLTZ (XX,YJ1,2)	CALL 181
	CALL PLTZ (XX,YJ1-0.1,2)	CALL 182
	CALL PLTZ (XX,YJ1,2)	CALL 183
15	CONTINUE	CALL 184
	GO TO 19	CALL 185
C	SCALE AND DRAW UPPER ABSCISSA AXIS-LOG	CALL 186
16	CALL SCALE (XL,FX,XIX,XJ1,-2,-1)	CALL 187
	XIX=1./XIX	CALL 188
	LOGG=XJ1/XIX+0.001	CALL 189
	DO 18 I=1,LOGG	CALL 190
	FX=FX+1.	CALL 191
	XD=XX	CALL 192
	DO 17 K=2,10	CALL 193
	XX=XD+ALOG10(FLOAT(K))*XIX	CALL 194
	CALL PLTZ (XX,YJ1,2)	CALL 195
	CALL PLTZ (XX,YJ1-0.1,2)	CALL 196
	CALL PLTZ (XX,YJ1,2)	CALL 197
17	CONTINUE	CALL 198
18	CCONTINUE	CALL 199
19	CALL PLTZ (XJ1,YJ1,3)	CALL 200
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 22	CALL 201
	IF (L1NY.NE.0.AND.J2.LE.0) GO TO 22	CALL 202
C	DRAW AND SCALE RIGHT-HAND ORDINATE-LOG PORTION	CALL 203
	CALL SYMBOL (XJ1+0.1,YJ1-.05,.10,2H10,0.,2)	CALL 204
	CALL NUMBER (XJ1+0.30,YJ1,.07,YLOG,0.,-1)	CALL 205
	CALL PLTZ (XJ1,YJ1,3)	CALL 206
	DO 21 I=1,J2	CALL 207
	YLOG=YLOG-1.	CALL 208
	YD=YY	CALL 209
	DO 20 K=1,9	CALL 210
	YY=YD-(1.-ALOG10(10.-FLOAT(K)))*YIY	CALL 211
	CALL PLTZ (XJ1,YY,2)	CALL 212
	CALL PLTZ (XJ1-0.1,YY,2)	CALL 213
	CALL PLTZ (XJ1,YY,2)	CALL 214
20	CONTINUE	CALL 215
	CALL SYMBOL (XJ1+0.1,YY-.05,.10,2H10,0.,2)	CALL 216
	CALL NUMBER (XJ1+0.30,YY,.07,YLOG,0.,-1)	CALL 217
	CALL PLTZ (XJ1,YY,3)	CALL 218
21	CONTINUE	CALL 219
	IF (L1NY.EQ.0) GO TO 25	CALL 220
	YLOG=YLOG-1.	CALL 221
	GO TO 23	CALL 222
C	DRAW AND SCALE RIGHT-HAND ORDINATE-LINEAR PORTION	CALL 223
22	IF (J2.EQ.-1) CALL SYMBOL (XJ1+.17,YJ1-.05,.10,1H0,0.,1)	CALL 224
	IF (J2.EQ.0) CALL SYMBOL (XJ1+.1,YJ1-.05,.10,2H10,0.,2)	CALL 225
	IF (L1NY.EQ.0) CALL SYMBOL (XJ1+.1,YJ1-.05,.10,1HE,0.,1)	CALL 226
	IF (L1NY.EQ.0) CALL NUMBER (XJ1+.25,YJ1-.05,.10,YLOG,0.,-1)	CALL 227
	CALL PLTZ (XJ1,YJ1,3)	CALL 228
23	DO 24 I=1,YJ1	CALL 229
	IF (L1NY.EQ.0) FY=FY-YIY	CALL 230
	YY=YY-YINCR	CALL 231

	CALL PLTZ (XJ1,YY,2)	CALL 232
	CALL PLTZ (XJ1-0.1,YY,2)	CALL 233
	IF (LINY.EQ.0) CALL NUMBER (XJ1+0.1,YY-.05,.10,FY,0.,-1)	CALL 234
	IF (I.EQ.JY1) GO TO 24	CALL 235
	IF (LINY.EQ.-1.AND.I.EQ.10) CALL SYMBOL (XJ1+.1,YY-.05,.10,1H0,0.,	CALL 236
	11)	CALL 237
	CALL PLTZ (XJ1,YY,3)	CALL 238
24	CONTINUE	CALL 239
	IF (LINY.EQ.0) GO TO 25	CALL 240
	IF (LINY.LT.0) CALL SYMBOL (XJ1+.2,-.05,.10,3H-10,0.,3)	CALL 241
	IF (LINY.GT.0) CALL SYMBOL (XJ1+.2,-.05,.10,1H0,0.,1)	CALL 242
25	IF (IND.GE.3) GO TO 27	CALL 243
	DRAW AND SCALE LOWER ABSCISSA AXIS-LINEAR	CALL 244
C	REDUCE LINEAR SCALE TO SMALLEST POSSIBLE INCREMENTS AND CORRESPOND	CALL 245
	POWER OF TEN	CALL 246
	XLOG=ALOG10(ABS(XIX))	CALL 247
	IF (XLOG.LT.0.) XLOG=XLOG-1.	CALL 248
	XLOG=AINT(XLOG)	CALL 249
	XIX=XIX*10.**(-XLOG)	CALL 250
	FX=FX*10.**(-XLOG)	CALL 251
	CALL SYMBOL (XJ1-0.2,-0.2,.10,1HE,0.,1)	CALL 252
	CALL NUMBER (XJ1-.05,-0.2,.10,XLOG,0.,-1)	CALL 253
	CALL PLTZ (XX,0.,3)	CALL 254
	DO 26 I=1,J1	CALL 255
	FX=FX+XIX	CALL 256
	XX=XX-1.	CALL 257
	CALL PLTZ (XX,0.,2)	CALL 258
	CALL PLTZ (XX,0.1,2)	CALL 259
	IF (I.EQ.J1) GO TO 26	CALL 260
	CALL NUMBER (XX-0.05,-0.2,.10,FX,0.,-1)	CALL 261
	CALL PLTZ (XX,0.,3)	CALL 262
26	CONTINUE	CALL 263
	CALL SYMBOL (-0.1,-0.2,.10,1HE,0.,1)	CALL 264
	CALL NUMBER (.05,-0.2,.10,XLOG,0.,-1)	CALL 265
	GO TO 30	CALL 266
27	XLOG=FX+0.0001	CALL 267
C	DRAW AND SCALE LOWER ABSCISSA AXIS-LOG	CALL 268
	CALL SYMBOL (XX-.15,-.20,.10,2H10,0.,2)	CALL 269
	CALL NUMBER (XX+0.1,-.15,.07,XLOG,0.,-1)	CALL 270
	CALL PLTZ (XX,0.,3)	CALL 271
	DO 29 I=1,LOGG	CALL 272
	XD=XX	CALL 273
	XLOG=XLOG-1.	CALL 274
	DO 28 K=1,9	CALL 275
	XX=XD-(1.-ALOG10(10.-FLOAT(K)))*XIX	CALL 276
	CALL PLTZ (XX,0.,2)	CALL 277
	CALL PLTZ (XX,0.1,2)	CALL 278
	CALL PLTZ (XX,0.,2)	CALL 279
28	CONTINUE	CALL 280
	CALL SYMBOL (XX-.15,-.20,.10,2H10,0.,2)	CALL 281
	CALL NUMBER (XX+0.1,-.15,.07,XLOG,0.,-1)	CALL 282
	CALL PLTZ (XX,0.,3)	CALL 283
29	CONTINUE	CALL 284
30	CONTINUE	CALL 285
C	SCALE AND PLOT DATA ARRAYS	CALL 286
	IF (LINE101.NE.-2) LINE101=-1	CALL 287
	IF (IND.LE.2) XLOG=10.**(-XLOG)	CALL 288
	IF (IND.EQ.1.OR.IND.EQ.3) YLOG=10.**(-YLOG)	CALL 289
C	SCALE YX,YY ARRAYS AND PLOT WITH A PLUS(+) SYMBOL	CALL 290
	DO 35 I=IC,N	CALL 291
	INDX=N-I+IC	CALL 292
	IF (IND.GT.2) GO TO 31	CALL 293
	XX=(X(INDX)*XLOG-FX)/(-XIX)	CALL 294
	GO TO 32	CALL 295
31	XX=(ALOG10(X(INDX))-XLOG)*XIX	CALL 296
32	IF (IND.EQ.2.OR.IND.EQ.4) GO TO 33	CALL 297
	YY=(Y(INDX)*YLOG-FY)/YIY	CALL 298
	GO TO 34	CALL 299
33	YY=(ALOG10(Y(INDX))-YLOG)*YIY	CALL 300
	IF (LINY.EQ.-1) YY=YY+YIY	CALL 301
34	IF (YY.LT.0.) YY=0.	CALL 302
	IF (YY.GT.YJ1) YY=YJ1	CALL 303
	IF (I.EQ.IC) CALL PLTZ (XX,YY,3)	CALL 304
	CALL SYMBOL (XX,YY,.10,3,0.,LINE101)	CALL 305
35	CONTINUE	CALL 306
	CALL SYMBOL (XJ1-1.5,YJ1+0.3,.10,10H(YX,YY) = ,0.,10)	CALL 307
	CALL SYMBOL (XJ1-.60,YJ1+0.35,.10,3,0.,-1)	CALL 308
42	IF (GRID.NE.-1.) GO TO 51	CALL 309
	YY=0.	CALL 310
	XD=0.	CALL 311
	XX=XJ1	CALL 312
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 45	CALL 313
	IF (LINY.NE.0) GO TO 45	CALL 314
43	DO 44 I=1,J2	CALL 315
	YD=YY	CALL 316
	DO 44 K=2,10	CALL 317
	YY=YD+ALOG10(FLOAT(K))*YIY	CALL 318
	CALL PLTZ (XD,YY,3)	CALL 319
	CALL PLTZ (XX,YY,2)	CALL 320
	X11=XD	CALL 321
	XD=XX	CALL 322
	XX=X11	CALL 323
44	CONTINUE	CALL 324
	GO TO 47	CALL 325
45	DO 46 I=1,JY1	CALL 326

	YY=YY+YINCR	CAL1	327
	CALL PLTZ (XD,YY,3)	CAL1	328
	CALL PLTZ (XX,YY,2)	CAL1	329
	XL1=XD	CAL1	330
	XD=XX	CAL1	331
	XX=XL1	CAL1	332
46	CONTINUE	CAL1	333
	IF (LINY.NE.0.AND.J2.GT.0) GO TO 43	CAL1	334
47	XX=0.	CAL1	335
	YD=YJ1	CAL1	336
	YY=0.	CAL1	337
	IF (IND.GT.2) GO TO 49	CAL1	338
	DO 48 I=1,J1	CAL1	339
	XX=XX+1.	CAL1	340
	CALL PLTZ (XX,YD,3)	CAL1	341
	CALL PLTZ (XX,YY,2)	CAL1	342
	XL1=YD	CAL1	343
	YD=YY	CAL1	344
	YY=XL1	CAL1	345
48	CONTINUE	CAL1	346
	GO TO 51	CAL1	347
49	DO 50 I=1,LOGG	CAL1	348
	XD=XX	CAL1	349
	DO 50 K=2,10	CAL1	350
	XX=XD+ALOG10(FLOAT(K))*XIX	CAL1	351
	CALL PLTZ (XX,YD,3)	CAL1	352
	CALL PLTZ (XX,YY,2)	CAL1	353
	XL1=YD	CAL1	354
	YD=YY	CAL1	355
	YY=XL1	CAL1	356
50	CONTINUE	CAL1	357
51	CALL PLTZ (XJ1+5.0,0.,-3)	CAL1	358
52	CONTINUE	CAL1	359
	DIVDEC=0.	CAL1	360
	RETURN	CAL1	361
	END	CAL1	362

	SUBROUTINE CALCOM2	CAL2	1
	COMMON NTELE,LENGTHY,LENGTHX,HA,IREADLK	CAL2	2
	COMMON GRID,C2,C3,DIVDEC	CAL2	3
	COMMON NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF	CAL2	4
	COMMON XLIMUP,XLIMLR,YMAX,YMIN,EMIN,EMAX	CAL2	5
	COMMON NSTEP,KEY100,IND,LINE100,LINE101,FAULTLT,DUMPLT,PRTN0	CAL2	6
	COMMON TITLE(4),COMMENT(4),TITLEA(4),TITLEB(4),TITLEC(4),DUMMY(4)	CAL2	7
	COMMON WORD(8,1),ELEMENT(4),LIMIT(4),TYPE(3),ID(3),II,JJ,LL	CAL2	8
	COMMON MTITLE(2),NPTSBLK(2),TAPENO,DISKA,DISKB,NFILE(2)	CAL2	9
	COMMON LASTTTL(2,4)	CAL2	10
	COMMON LOCK(2)	CAL2	11
	COMMON X(6000),Y(6000)	COM2	1
	COMMON AX(6000),AY(6000)	COM3	1
	COMMON /ONE/ KEY,XL,YL,LINX,ARRAY,MTELE	CAL2	14
	REAL X,Y,AX,AY,BY,CY	CAL2	15
	REAL GRID,C2,C3,DIVDEC	CAL2	16
	REAL XLIMUP,XLIMLR,YMAX,YMIN,XMIN,EMAX,ELEMENT	CAL2	17
	REAL LASTTTL	CAL2	18
	INTEGER NY,NA,NB,NC,NNIC,NNCF,AI,AF,BI,BF,CI,CF,NI,NF	CAL2	19
	INTEGER NSTEP,KEY100,IND,IFIRST,LINE101,FAULTLT,DUMPLT,PRTN0	CAL2	20
	INTEGER LIMIT,ID,II,JJ,LL	CAL2	21
	INTEGER MTITLE,NPTSBLK,TAPENO,DISKA,DISKB,NFILE	CAL2	22
	DIMENSION XL(2),YL(2)	CAL2	23
C	NEGATIVE LINEAR OPTION	CAL2	24
CCC	NNIC=INITIAL INDEX	CAL2	25
CCC	NNCF=FINAL INDEX	CAL2	26
CCC	X AND Y ARE ABSCISSA AND ORDINATE RESPECTIVELY	CAL2	27
CCC	N=NO. OF PTS.	CAL2	28
CCC	DIVDEC IS INCHES/DECADE OR INCHES/(5 UNITS) FOR EXTENDED ABSCISSA. =0	CAL2	29
CCC	FOR STANDARD LENGTH ABSCISSA(17 INCHES)	CAL2	30
CCC	YMIN AND YMAX ARE ORDINATE SCALING LIMITS- IF YMIN=YMAX=0,MPL0T WILL	CAL2	31
CCC	DETERMINE THE ORDINATE SCALING LIMITS ITSELF.	CAL2	32
CCC	WORD(4,1) = 2. FOR (AX,AY) OVERLAY	CAL2	33
CCC	XJ1 IS THE LENGTH OF THE ABSCISSA IN INCHES	CAL2	34
CCC	YJ1 IS THE LENGTH OF THE ORDINATE IN INCHES	CAL2	35
CCC	ARRAY, ARRAA ARE ARRAY (Y,AY) IDENTIFIERS-IF ONE (OR BOTH) IS SET	CAL2	36
CCC	TO 1H0, PLOTTING IS DELETED FOR THAT ARRAY	CAL2	37
CCC	FOR IND=2,4-FIRST DECADE POSITIVE LINEAR IF LINE100 IS +1--FIRST	CAL2	38
CCC	DECADE NEGATIVE OPTION IF LINE100=-1	CAL2	39
CCC	LINX INDICATES FIRST DECADE LINEAR DATA(NEGATIVE,POSITIVE) FOR	CAL2	40
CCC	LOGARITHMIC ORDINATE	CAL2	41
CCC	FOR IND=2,4-LINEAR PORTION IS DELETED IF LINX=0	CAL2	42
CCC	FOR IND=2,4-LOG PORTION DELETED IF ALL POINTS LINEAR(LE.10.0)	CAL2	43
CCC	NA1,AI AND NAF,AF ARE THE INITIAL AND FINAL INDICES FOR THE AX,AY	CAL2	44
CCC	ARRAYS	CAL2	45
CCC	IF (TYPE(2).NE.7HCALCOMP) CALL PLOTS (7)	CAL2	46
CCC	TYPE(2)=7HCALCOMP	CAL2	47
CCC	MTELE=NTELE	CAL2	48
CCC	XJ1=17.	CAL2	49
CCC	J1=XJ1+0.001	CAL2	50
CCC	YJ1=10.	CAL2	51
CCC	J2=YJ1+0.001	CAL2	52
CCC	FY=0.	CAL2	53
CCC	YY=0.	CAL2	54

	ARRAY=2H Y	CAL2	55
	ARRA=2HAY	CAL2	56
	LINY=0	CAL2	57
	IC=NNIC	CAL2	58
	N=NNCF	CAL2	59
	ARRAA=1H0	CAL2	60
	KEY=0	CAL2	61
	XL(2)=-10000.	CAL2	62
	XL(1)=+10000.	CAL2	63
	ARRAY=2H Y	CAL2	64
C	REMOVE ILLEGAL POINTS-REORDER ARRAYS-DETERMINE MIN,MAX OF ARRAYS-	CAL2	65
C	AND PRE-SCALE ARRAYS(AS APPLICABLE)	CAL2	66
	CALL REMCOV (X,Y,IND,IC,N,LINE100)	CAL2	67
	IPCTCX=N-IC+1	CAL2	68
	IF (WORD(4,1).NE.2.) GO TO 1	CAL2	69
	SAVJ=ARRAY	CAL2	70
	ARRAY=2HAY	CAL2	71
	NAI=AI	CAL2	72
	NAF=AF	CAL2	73
	CALL REMCOV (AX,AY,IND,NAI,NAF,LINE100)	CAL2	74
	ARRAA=ARRAY	CAL2	75
	ARRAY=SAVJ	CAL2	76
	IPCTCA=NAF-NAI+1	CAL2	77
C	IF BOTH DATA SETS ARE EMPTY, SKIP PLOT	CAL2	78
1	IF (ARRAY.EQ.1H0.AND.ARRAA.EQ.1H0) GO TO 52	CAL2	79
	IF (DIVDEC.EQ.0.0) GO TO 6	CAL2	80
C	VARIABLE LENGTH PLOT DETERMINED BY DIVDEC	CAL2	81
C	SCALE ORDINATE ACCORDING TO YMIN AND YMAX(IF APPLICABLE)	CAL2	82
	IF ((YMIN.EQ.0.0).AND.(YMAX.EQ.0.0)) GO TO 3	CAL2	83
	IF ((IND.EQ.2).OR.(IND.EQ.4)) GO TO 2	CAL2	84
	YL(1)=YMIN	CAL2	85
	YL(2)=YMAX	CAL2	86
	GO TO 3	CAL2	87
2	IF (YMAX.GT.0.0) YL(2)=YMAX	CAL2	88
	IF (LINY.NE.0) GO TO 3	CAL2	89
	IF (YMIN.GT.0.0) YL(1)=YMIN	CAL2	90
3	IF (IND.LT.3) GO TO 4	CAL2	91
C	DEFINE ABSCISSA LIMITS AND NUMBER OF INCHES FOR LOG-AXIS	CAL2	92
C	DIVDEC IS IN INCHES/DECADE	CAL2	93
	XL2=INT(ALOG10(XL(2))+0.99999)	CAL2	94
	XL1=INT(ALOG10(XL(1)))	CAL2	95
	J1=(XL2-XL1+0.001)*DIVDEC	CAL2	96
	GO TO 5	CAL2	97
C	DEFINE ABSCISSA LIMITS AND NUMBER OF INCHES FOR LINEAR AXIS	CAL2	98
C	DIVDEC IS IN INCHES/(5 UNITS)	CAL2	99
4	XL2=INT((XL(2)+4.99999)/5.)*5	CAL2	100
	XL1=INT(XL(1)/5.)*5	CAL2	101
	J1=((XL2-XL1+0.001)/5.)*DIVDEC	CAL2	102
C	LIMIT LENGTH OF ABSCISSA TO 156 INCHES(13 FEET)	CAL2	103
5	IF (J1.GT.156) J1=156	CAL2	104
	XJ1=J1	CAL2	105
C	FOR LOG ORDINATE, IF LINE100=0, OR IF NO POINTS IN FIRST DECADE(0.	CAL2	106
C	10.), GO TO NORMAL PLOT	CAL2	107
6	IF (LINY.NE.0) GO TO 7	CAL2	108
	INDY=+2	CAL2	109
	IF (IND.EQ.2.OR.IND.EQ.4) INDY=-2	CAL2	110
C	DETERMINE SCALE AND CONSTANTS FOR ORDINATE AXIS	CAL2	111
	CALL SCALE (YL,FY,YIY,YJ1,INDY,1)	CAL2	112
	IF (INDY.LT.0) J2=YIY*YJ1+0.001	CAL2	113
	IF (INDY.LT.0) GO TO 11	CAL2	114
	YINCR=1.	CAL2	115
	YJ1=YJ1	CAL2	116
C	REDUCE LINEAR SCALE TO SMALLEST POSSIBLE INCREMENTS AND CORRESPOND	CAL2	117
C	POWER OF TEN	CAL2	118
	YLOG=ALOG10(ABS(YIY))	CAL2	119
	IF (YLOG.LT.0.) YLOG=YLOG-1.	CAL2	120
	YLOG=AINI(YLOG)	CAL2	121
	YIY=YIY*10.**(-YLOG)	CAL2	122
	FY=FY*10.**(-YLOG)	CAL2	123
	CALL NUMBER (-0.35,-0.05,.10,FY,0.,-1)	CAL2	124
	GO TO 9	CAL2	125
C	DETERMINE SCALE AND CONSTANTS-FIRST DECADE LINEAR ORDINATE	CAL2	126
7	IF (YL(2).GT.1) GO TO 8	CAL2	127
	J2=-1	CAL2	128
	FY=1.	CAL2	129
	YLOG=0.	CAL2	130
	YIY=YJ1	CAL2	131
	YINCR=YJ1/10.	CAL2	132
	JY1=10	CAL2	133
	CALL SYMBOL (-0.35,-0.05,.10,3H-10,0.,3)	CAL2	134
	GO TO 9	CAL2	135
8	J2=ALOG10(YL(2))-0.000001	CAL2	136
	FY=1.	CAL2	137
	YLOG=0.	CAL2	138
	YIY=YJ1	CAL2	139
	K2=J2+1	CAL2	140
	JY1=10	CAL2	141
	IF (LINY.LT.0) JY1=20	CAL2	142
	IF (LINY.LT.0) CALL SYMBOL (-0.35,-0.05,.10,3H-10,0.,3)	CAL2	143
	IF (LINY.GT.0) CALL SYMBOL (-0.17,-0.05,.10,1H0,0.,1)	CAL2	144
	IF (LINY.LT.0) K2=K2+1	CAL2	145
	YIY=YIY/10	CAL2	146
	YINCR=YIY/10.	CAL2	147
	IF (J2.GE.1) YIY=1./YIY	CAL2	148
C	DRAW AND SCALE LEFT-HAND ORDINATE--LINEAR PORTION	CAL2	149

9	CALL PLTZ (0.,0.,3)	CAL2 150
	DO 10 I=1,JY1	CAL2 151
	IF (LINY.EQ.0) FY=FY+YIY	CAL2 152
	YY=YY+YINCR	CAL2 153
	CALL PLTZ (0.,YY,2)	CAL2 154
	CALL PLTZ (0.1,YY,2)	CAL2 155
	IF (I.EQ.JY1) GO TO 10	CAL2 156
	IF (LINY.EQ.-1.AND.I.EQ.10) CALL SYMBOL (-0.17,YY-0.05,.10,1H0,0.,	CAL2 157
	11)	CAL2 158
	IF (LINY.EQ.0) CALL NUMBER (-0.35,YY-.05,.10,FY,0.,-1)	CAL2 159
10	CALL PLTZ (0.,YY,3)	CAL2 160
	CONTINUE	CAL2 161
	IF (LINY.NE.0.AND.J2.GT.0) GO TO 11	CAL2 162
	IF (J2.EQ.-1) CALL SYMBOL (-0.17,YJ1-.05,.10,1H0,0.,1)	CAL2 163
	IF (J2.EQ.0) CALL SYMBOL (-0.25,YJ1-.05,.10,2H10,0.,2)	CAL2 164
	IF (LINY.EQ.0) CALL SYMBOL (-0.45,YJ1-.05,.10,1HE,0.,1)	CAL2 165
	IF (LINY.EQ.0) CALL NUMBER (-0.30,YJ1-.05,.10,YLOG,0.,-1)	CAL2 166
	CALL PLTZ (0.,YJ1,3)	CAL2 167
	GO TO 14	CAL2 168
C	DRAW AND SCALE LEFT-HAND ORDINATE-LOG PORTION	CAL2 169
11	YLOG=FY+0.0001	CAL2 170
	CALL SYMBOL (-0.35,YY-.05,.10,2H10,0.,2)	CAL2 171
	CALL NUMBER (-0.15,YY,.07,YLOG,0.,-1)	CAL2 172
	CALL PLTZ (0.,YY,3)	CAL2 173
	YIY=1./YIY	CAL2 174
	DO 13 I=1,J2	CAL2 175
	YLOG=YLOG+1.	CAL2 176
	YD=YY	CAL2 177
	DO 12 K=2,10	CAL2 178
	YY=YD+ALOG10(FLOAT(K))*YIY	CAL2 179
	CALL PLTZ (0.,YY,2)	CAL2 180
	CALL PLTZ (0.1,YY,2)	CAL2 181
12	CALL PLTZ (0.,YY,2)	CAL2 182
	CALL SYMBOL (-0.35,YY-.05,.10,2H10,0.,2)	CAL2 183
	CALL NUMBER (-0.15,YY,.07,YLOG,0.,-1)	CAL2 184
	CALL PLTZ (0.,YY,3)	CAL2 185
13	CONTINUE	CAL2 186
14	XX=0.	CAL2 187
	CALL SYMBOL (1.0,YJ1+0.25,.10,TITLE,0.,40)	CAL2 188
	CALL SYMBOL (1.0,YJ1+0.05,.10,COMMENT,0.,40)	CAL2 189
	CALL PLTZ (0.,YJ1,3)	CAL2 190
	IF (IND.GT.2) GO TO 16	CAL2 191
C	SCALE AND DRAW UPPER ABSCISSA AXIS-LINEAR	CAL2 192
	CALL SCALE (XL,FX,XIX,XJ1,2,-1)	CAL2 193
	DO 15 I=1,J1	CAL2 194
	XX=XX+1.	CAL2 195
	CALL PLTZ (XX,YJ1,2)	CAL2 196
	CALL PLTZ (XX,YJ1-0.1,2)	CAL2 197
	CALL PLTZ (XX,YJ1,2)	CAL2 198
15	CONTINUE	CAL2 199
	GO TO 19	CAL2 200
C	SCALE AND DRAW UPPER ABSCISSA AXIS-LOG	CAL2 201
16	CALL SCALE (XL,FX,XIX,XJ1,-2,-1)	CAL2 202
	XIX=1./XIX	CAL2 203
	LOGG=XJ1/XIX+0.001	CAL2 204
	DO 18 I=1,LOGG	CAL2 205
	FX=FX+1.	CAL2 206
	XD=XX	CAL2 207
	DO 17 K=2,10	CAL2 208
	XX=XD+ALOG10(FLOAT(K))*XIX	CAL2 209
	CALL PLTZ (XX,YJ1,2)	CAL2 210
	CALL PLTZ (XX,YJ1-0.1,2)	CAL2 211
	CALL PLTZ (XX,YJ1,2)	CAL2 212
17	CONTINUE	CAL2 213
18	CONTINUE	CAL2 214
19	CALL PLTZ (XJ1,YJ1,3)	CAL2 215
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 22	CAL2 216
	IF (LINY.NE.0.AND.J2.LE.0) GO TO 22	CAL2 217
C	DRAW AND SCALE RIGHT-HAND ORDINATE-LOG PORTION	CAL2 218
	CALL SYMBOL (XJ1+0.1,YJ1-.05,.10,2H10,0.,2)	CAL2 219
	CALL NUMBER (XJ1+0.30,YJ1,.07,YLOG,0.,-1)	CAL2 220
	CALL PLTZ (XJ1,YJ1,3)	CAL2 221
	DO 21 I=1,J2	CAL2 222
	YLOG=YLOG-1.	CAL2 223
	YD=YY	CAL2 224
	DO 20 K=1,9	CAL2 225
	YY=YD-(1.-ALOG10(10.-FLOAT(K)))*YIY	CAL2 226
	CALL PLTZ (XJ1,YY,2)	CAL2 227
	CALL PLTZ (XJ1-0.1,YY,2)	CAL2 228
	CALL PLTZ (XJ1,YY,2)	CAL2 229
20	CONTINUE	CAL2 230
	CALL SYMBOL (XJ1+0.1,YY-.05,.10,2H10,0.,2)	CAL2 231
	CALL NUMBER (XJ1+0.30,YY,.07,YLOG,0.,-1)	CAL2 232
	CALL PLTZ (XJ1,YY,3)	CAL2 233
21	CONTINUE	CAL2 234
	IF (LINY.EQ.0) GO TO 25	CAL2 235
	YLOG=YLOG-1.	CAL2 236
	GO TO 23	CAL2 237
C	DRAW AND SCALE RIGHT-HAND ORDINATE-LINEAR PORTION	CAL2 238
22	IF (J2.EQ.-1) CALL SYMBOL (XJ1+.17,YJ1-.05,.10,1H0,0.,1)	CAL2 239
	IF (J2.EQ.0) CALL SYMBOL (XJ1+.1,YJ1-.05,.10,2H10,0.,2)	CAL2 240
	IF (LINY.EQ.0) CALL SYMBOL (XJ1+.1,YJ1-.05,.10,1HE,0.,1)	CAL2 241
	IF (LINY.EQ.0) CALL NUMBER (XJ1+.25,YJ1-.05,.10,YLOG,0.,-1)	CAL2 242
	CALL PLTZ (XJ1,YJ1,3)	CAL2 243
23	DO 24 I=1,JY1	CAL2 244

	IF (LINY.EQ.0) FY=FY-YIY	CAL2	245
	YY=YY-YINCR	CAL2	246
	CALL PLTZ (XJ1,YY,2)	CAL2	247
	CALL PLTZ (XJ1-0.1,YY,2)	CAL2	248
	IF (LINY.EQ.0) CALL NUMBER (XJ1+0.1,YY-.05,.10,FY,0.,-1)	CAL2	249
	IF (I.EQ.JY1) GO TO 24	CAL2	250
	IF (LINY.EQ.-1.AND.I.EQ.10) CALL SYMBOL (XJ1+.1,YY-.05,.10,1H0,0.,	CAL2	251
	1)	CAL2	252
24	CALL PLTZ (XJ1,YY,3)	CAL2	253
	CONTINUE	CAL2	254
	IF (LINY.EQ.0) GO TO 25	CAL2	255
	IF (LINY.LT.0) CALL SYMBOL (XJ1+.2,-.05,.10,3H-10,0.,3)	CAL2	256
	IF (LINY.GT.0) CALL SYMBOL (XJ1+.2,-.05,.10,1H0,0.,1)	CAL2	257
25	IF (IND.GE.3) GO TO 27	CAL2	258
C	DRAW AND SCALE LOWER ABSCISSA AXIS-LINEAR	CAL2	259
C	REDUCE LINEAR SCALE TO SMALLEST POSSIBLE INCREMENTS AND CORRESPOND	CAL2	260
	POWER OF TEN	CAL2	261
	XLOG=ALOG10(ABS(XIX))	CAL2	262
	IF (XLOG.LT.0.) XLOG=XLOG-1.	CAL2	263
	XLOG=AINT(XLOG)	CAL2	264
	XIX=XIX*10.**(-XLOG)	CAL2	265
	FX=FX*10.**(-XLOG)	CAL2	266
	CALL SYMBOL (XJ1-0.2,-0.2,.10,1HE,0.,1)	CAL2	267
	CALL NUMBER (XJ1-.05,-0.2,.10,XLOG,0.,-1)	CAL2	268
	CALL PLTZ (XX,0.,3)	CAL2	269
	DO 26 I=1,J1	CAL2	270
	FX=FX+XIX	CAL2	271
	XX=XX-1.	CAL2	272
	CALL PLTZ (XX,0.,2)	CAL2	273
	CALL PLTZ (XX,0.1,2)	CAL2	274
	IF (I.EQ.J1) GO TO 26	CAL2	275
	CALL NUMBER (XX-0.05,-0.2,.10,FX,0.,-1)	CAL2	276
26	CALL PLTZ (XX,0.,3)	CAL2	277
	CONTINUE	CAL2	278
	CALL SYMBOL (-0.1,-0.2,.10,1HE,0.,1)	CAL2	279
	CALL NUMBER (.05,-0.2,.10,XLOG,0.,-1)	CAL2	280
	GO TO 30	CAL2	281
27	XLOG=FX+0.0001	CAL2	282
C	DRAW AND SCALE LOWER ABSCISSA AXIS-LOG	CAL2	283
	CALL SYMBOL (XX-.15,-.20,.10,2H10,0.,2)	CAL2	284
	CALL NUMBER (XX+0.1,-.15,.07,XLOG,0.,-1)	CAL2	285
	CALL PLTZ (XX,0.,3)	CAL2	286
	DO 29 I=1,LOGG	CAL2	287
	XD=XX	CAL2	288
	XLOG=XLOG-1.	CAL2	289
	DO 28 K=1,9	CAL2	290
	XX=XD-(1.-ALOG10(10.-FLOAT(K)))*XIX	CAL2	291
	CALL PLTZ (XX,0.,2)	CAL2	292
	CALL PLTZ (XX,0.1,2)	CAL2	293
	CALL PLTZ (XX,0.,2)	CAL2	294
28	CONTINUE	CAL2	295
	CALL SYMBOL (XX-.15,-.20,.10,2H10,0.,2)	CAL2	296
	CALL NUMBER (XX+0.1,-.15,.07,XLOG,0.,-1)	CAL2	297
	CALL PLTZ (XX,0.,3)	CAL2	298
29	CONTINUE	CAL2	299
30	IOIR=-1	CAL2	300
C	SCALE AND PLOT DATA ARRAYS	CAL2	301
	IF (LINE101.NE.-2) LINE101=-1	CAL2	302
	IF (IND.LE.2) XLOG=10.**(-XLOG)	CAL2	303
	IF (IND.EQ.1.OR.IND.EQ.3) YLOG=10.**(-YLOG)	CAL2	304
	IF (ARRAY.EQ.1H0) GO TO 36	CAL2	305
	SCALE YX,YY ARRAYS AND PLOT WITH A PLUS(+) SYMBOL	CAL2	306
	IOIR=+1	CAL2	307
	DO 35 I=1,C,N	CAL2	308
	INDX=N-I+1	CAL2	309
	IF (IND.GT.2) GO TO 31	CAL2	310
	XX=(X(INDX)*XLOG-FX)/(-XIX)	CAL2	311
	GO TO 32	CAL2	312
31	XX=(ALOG10(X(INDX))-XLOG)*XIX	CAL2	313
32	IF (IND.EQ.2.OR.IND.EQ.4) GO TO 33	CAL2	314
	YY=(Y(INDX)*YLOG-FY)/YIY	CAL2	315
	GO TO 34	CAL2	316
33	YY=(ALOG10(Y(INDX))-YLOG)*YIY	CAL2	317
	IF (LINY.EQ.-1) YY=YY+YIY	CAL2	318
34	IF (YY.LT.0.) YY=0.	CAL2	319
	IF (YY.GT.YJ1) YY=YJ1	CAL2	320
	IF (I.EQ.IC) CALL PLTZ (XX,YY,3)	CAL2	321
	CALL SYMBOL (XX,YY,.10,3,0.,LINE101)	CAL2	322
35	CONTINUE	CAL2	323
	CALL SYMBOL (XJ1-1.5,YJ1+0.3,.10,10H(YX,YY) = ,0.,10)	CAL2	324
	CALL SYMBOL (XJ1-.60,YJ1+0.35,.10,3,0.,-1)	CAL2	325
C	SCALE AX,AY ARRAYS AND PLOT WITH A CIRCLE(O) SYMBOL	CAL2	326
	IF (ARAA.EQ.1H0) GO TO 42	CAL2	327
36	DO 41 I=NAI,NAF	CAL2	328
	INDX=I	CAL2	329
	IF (IOIR.LT.0) INDX=NAF-I+NAI	CAL2	330
	IF (IND.GT.2) GO TO 37	CAL2	331
	XX=(AX(INDX)*XLOG-FX)/(-XIX)	CAL2	332
	GO TO 38	CAL2	333
37	XX=(ALOG10(AX(INDX))-XLOG)*XIX	CAL2	334
38	IF (IND.EQ.2.OR.IND.EQ.4) GO TO 39	CAL2	335
	YY=(AY(INDX)*YLOG-FY)/YIY	CAL2	336
	GO TO 40	CAL2	337
39	YY=(ALOG10(AY(INDX))-YLOG)*YIY	CAL2	338
	IF (LINY.EQ.-1) YY=YY+YIY	CAL2	339

40	IF (YY.LT.0.) YY=0.	CAL2	340
	IF (YY.GT.YJ1) YY=YJ1	CAL2	341
	IF (I.EQ.NA1) CALL PLTZ (XX,YY,3)	CAL2	342
	CALL SYMBOL (XX,YY,.10,1,0.,LINE101)	CAL2	343
41	CONTINUE	CAL2	344
	CALL SYMBOL (XJ1-1.5,YJ1+0.1,.10,10H(AX,AY; = ,0.,10)	CAL2	345
	CALL SYMBOL (XJ1-.60,YJ1+0.15,.10,1,0.,-1)	CAL2	346
42	IF (GRID.NE.-1.) GO TO 51	CAL2	347
	YY=0.	CAL2	348
	XD=0.	CAL2	349
	XX=XJ1	CAL2	350
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 45	CAL2	351
	IF (LINY.NE.0) GO TO 45	CAL2	352
43	DO 44 I=1,J2	CAL2	353
	YD=YY	CAL2	354
	DO 44 K=2,10	CAL2	355
	YY=YD+ALOG10(FLOAT(K))*YIY	CAL2	356
	CALL PLTZ (XD,YY,3)	CAL2	357
	CALL PLTZ (XX,YY,2)	CAL2	358
	XL1=XD	CAL2	359
	XD=XX	CAL2	360
	XX=XL1	CAL2	361
44	CONTINUE	CAL2	362
	GO TO 47	CAL2	363
45	DO 46 I=1,JY1	CAL2	364
	YY=YY+YINCR	CAL2	365
	CALL PLTZ (XD,YY,3)	CAL2	366
	CALL PLTZ (XX,YY,2)	CAL2	367
	XL1=XD	CAL2	368
	XD=XX	CAL2	369
	XX=XL1	CAL2	370
46	CONTINUE	CAL2	371
	IF (LINY.NE.0.AND.J2.GT.0) GO TO 43	CAL2	372
47	XX=0.	CAL2	373
	YD=YJ1	CAL2	374
	YY=0.	CAL2	375
	IF (IND.GT.2) GO TO 49	CAL2	376
	DO 48 I=1,J1	CAL2	377
	XX=XX+1.	CAL2	378
	CALL PLTZ (XX,YD,3)	CAL2	379
	CALL PLTZ (XX,YY,2)	CAL2	380
	XL1=YD	CAL2	381
	YD=YY	CAL2	382
	YY=XL1	CAL2	383
48	CONTINUE	CAL2	384
	GO TO 51	CAL2	385
49	DO 50 I=1,LOGG	CAL2	386
	XD=XX	CAL2	387
	DO 50 K=2,10	CAL2	388
	XX=XD+ALOG10(FLOAT(K))*XIX	CAL2	389
	CALL PLTZ (XX,YD,3)	CAL2	390
	CALL PLTZ (XX,YY,2)	CAL2	391
	XL1=YD	CAL2	392
	YD=YY	CAL2	393
	YY=XL1	CAL2	394
50	CONTINUE	CAL2	395
51	CALL PLTZ (XJ1+5.0,0.,-3)	CAL2	396
52	CONTINUE	CAL2	397
	DIVDEC=0.0	CAL2	398
	RETURN	CAL2	399
	END	CAL2	400

	SUBROUTINE REMOV (I,V,IND,I1,IE,LINE100)	REM2	1
	COMMON/ONE/ KEY,XL,YL,LIN,ARRAY,NTELE	REM2	2
	DIMENSION I(6000), V(6000), YL(2), XL(2)	REM2	3
	NUMBER IS THE NUMBER OF ILLEGAL(.LE.C.0) POINTS FOR IND=2,4	REM2	4
	NNT IS THE NUMBER OF LEGAL POINTS	REM2	5
	FOR IND=2,4 (LOG-ORDINATE CASE), SUB REMOV REMOVES THOSE ORDINATE	REM2	6
	POINTS LESS THAN OR EQUAL TO 0.0 AND REORDERS(SHIFTS) THE ORDINATE	REM2	7
	(V(I)) AND ABSCISSA(I(I)) ARRAYS	REM2	8
	SUB REMOV ALSO PRE-SCALES THE ORDINATE ARRAYS FOR THE FIRST DECADE	REM2	9
	LINEAR CASE(LINE100 NON-ZERO)	REM2	10
	FOR LOG ORDINATE, FIRST DECADE LINEAR IF LINE100 IS NON-ZERO	REM2	11
	NUMBER=0	REM2	12
	J=I1-1	REM2	13
	DO 9 I=1,IE	REM2	14
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 2	REM2	15
	IF (V(I).LE.-10.) GO TO 1	REM2	16
	IF (LINE100.NE.-1.AND.V(I).LE.0.) GO TO 1	REM2	17
	GO TO 2	REM2	18
1	NUMBER=NUMBER+1	REM2	19
	GO TO 9	REM2	20
2	J=J+1	REM2	21
	IF (KEY.NE.0) GO TO 4	REM2	22
	INITIALIZING MIN,MAX(YL(1),YL(2)) OF ORDINATE ON FIRST CALL TO SUB	REM2	23
	REMOV	REM2	24
	YL(1)=V(I)	REM2	25
	IF ((IND.EQ.1).OR.(IND.EQ.3)) GO TO 3	REM2	26
	IF (LINE100.EQ.0) GO TO 3	REM2	27
	IF (V(I).GT.10.) GO TO 3	REM2	28
	YL(1)=10.**(.01*V(I))	REM2	29

3	YL(2)=YL(1)	REM2	30
	KEY=KEY+1	REM2	31
C	REORDERING(SHIFTING) THE ARRAYS-REMOVING THE ILLEGAL POINTS	REM2	32
4	T(J)=T(I)	REM2	33
	V(J)=V(I)	REM2	34
C	PRE-SCALING(IF NECESSARY) OF ORDINATE AND ABSCISSA VALUES FOR LOG-	REM2	35
C	AXIS CASE	REM2	36
	IF (IND.EQ.1.OR.IND.EQ.3) GO TO 5	REM2	37
	IF (LINE100.EQ.0) GO TO 5	REM2	38
	IF (V(J).GT.10.) GO TO 5	REM2	39
	VJ=V(J)	REM2	40
	V(J)=10.**(.01*V(J))	REM2	41
	IF (LINY.EQ.-1) GO TO 5	REM2	42
	LINY=+1	REM2	43
	IF (VJ.LT.0.) LINY=-1	REM2	44
C	DETERMINING MIN,MAX OF ORDINATE AND ABSCISSA FOR USE IN MPLUT	REM2	45
5	IF (YL(1).LE.V(J)) GO TO 6	REM2	46
	YL(1)=V(J)	REM2	47
6	IF (YL(2).GE.V(J)) GO TO 7	REM2	48
	YL(2)=V(J)	REM2	49
7	IF (XL(1).LE.T(J)) GO TO 8	REM2	50
	XL(1)=T(J)	REM2	51
8	IF (XL(2).GE.T(J)) GO TO 9	REM2	52
	XL(2)=T(J)	REM2	53
9	CONTINUE	REM2	54
	NNT=IE-I+1-NUMBER	REM2	55
	WRITE(11,10)NNT,NUMBER,ARRAY	REM2	56
	IF (INTEL.EQ.1)WRITE(12,10)NNT,NUMBER,ARRAY	REM2	57
10	FORMAT (24HNUMBER OF LEGAL POINTS=,16,25HNUMBER OF ILLEGAL POINTS	REM2	58
	I=16,7H ARRAY ,A2)	REM2	59
C	REDEFINING THE FINAL INDEX IE-REMOVING THE NUMBER OF ILLEGAL INDIC	REM2	60
	IE=IE-NUMBER	REM2	61
	IF ((IE-I).GT.0) GO TO 12	REM2	62
	WRITE(11,11)ARRAY	REM2	63
	IF (INTEL.EQ.1)WRITE(12,11)ARRAY	REM2	64
11	FORMAT (68H NUMBER OF POINTS IS LESS THAN OR EQUAL TO 0. CLCMP SKI	REM2	65
	PPED FOR SET ,A2)	REM2	66
	ARRAY=IHO	REM2	67
12	CONTINUE	REM2	68
	RETURN	REM2	69
	END	REM2	70

	SUBROUTINE BLOCK (IA)	BLOK	1
	COMMON/CALCOMP/A(520),NPX,NPY,FACT,I,J,JMAX,PENZ,F200,XMIN,DX,YMIN	BLOK	2
	1,DY, IBA,IWORD(10),RECNO,JXY(4),IENT,LTAPE,C200	BLOK	3
	DIMENSION BN(2,10)	BLOK	4
	DATA BN/004,004,004,004,006,004,007,005,004,005,005,006,	BLOK	5
	1005,007,006,004,006,005/	BLOK	6
	DATA SYN,CO,8A/017,011,014/	BLOK	7
	IB=IA	BLOK	8
	IF (IB.LT.1000) GO TO 20	BLOK	9
10	IB=IB-(IB/1000)*1000	BLOK	10
20	N1=1	BLOK	11
	N2=1	BLOK	12
	IF (IB.LT.100) GO TO 40	BLOK	13
30	N1=IB/100+1	BLOK	14
	IB=IB-N1*100+100	BLOK	15
40	IF (IB.LT.10) GO TO 60	BLOK	16
50	N2=IB/10+1	BLOK	17
	IB=IB-N2*10+10	BLOK	18
60	N3=IB+1	BLOK	19
	DO 100 L=1,9	BLOK	20
100	CALL STORE(CD)	BLOK	21
	DO 110 L=1,7	BLOK	22
110	CALL STORE(SYN)	BLOK	23
	CALL STORE (BA)	BLOK	24
	CALL STORE (BN(1,N1))	BLOK	25
	CALL STORE (BN(2,N1))	BLOK	26
	CALL STORE (BN(1,N2))	BLOK	27
	CALL STORE (BN(2,N2))	BLOK	28
	CALL STORE (BN(1,N3))	BLOK	29
	CALL STORE (BN(2,N3))	BLOK	30
	CALL STORE (BA)	BLOK	31
	DO 120 L=1,7	BLOK	32
120	CALL STORE (SYN)	BLOK	33
	DO 130 L=1,9	BLOK	34
130	CALL STORE(CD)	BLOK	35
135	CALL TAPWRI(13,20)	BLOK	36
	J=13	BLOK	37
999	RETURN	BLOK	38
	END	BLOK	39

	SUBROUTINE TAPWRI (IS,IMAX)	TAPW	1
	COMMON/CALCOMP/A(520),NPX,NPY,FACT,I,J,JMAX,PENZ,F200,XMIN,DX,YMIN	TAPW	2
	1,DY, IBA,IWORD(10),RECNO,JXY(4),IENT,LTAPE,C200	TAPW	3
	DIMENSION NEWTP(5)	TAPW	4
	DATA NEWTP/5*0/	TAPW	5

5	FORMAT(*MOUNT NUL BLANK ON*13,* TYPE TAPE NO + GO*)	TAPW	6
10	I1=IS	TAPW	7
	I2=IMAX	TAPW	8
20	WRITE (LTAPE) (A(K),K=1,I2)	TAPW	9
	RECNO=RECNO+1	TAPW	10
	IF (RECNO,LT.2500.) GO TO 40	TAPW	11
30	WRITE(LTAPE) (A(K),K=1,9)	TAPW	12
	RECNO=0.0	TAPW	13
	CALL UNLODE(LTAPE)	TAPW	14
	CALL GETEQN(LTAPE,NUNIT)	TAPW	15
	NUNIT=10*(NUNIT/8)+MOD(NUNIT,8)	TAPW	16
	ENCODE(39,5,NEWTP) NUNIT	TAPW	17
	CALL REMARK(NEWTP)	TAPW	18
40	WRITE(LTAPE) (A(K),K=1,9)	TAPW	19
	RETURN	TAPW	20
	END	TAPW	21

	SUBROUTINE STORE(IARG)	STOR	1
C	THE FOLLOWING SECTION OF CODE PACKS UP 10 COMMANDS INTO THE PROPER	STOR	2
C	TAPE FORMAT AND CHECKS FOR FULL BUFFER, WRITES TAPE IF NECESSARY,	STOR	3
C	AND ROTATES BUFFERS. THE RETURN IS AN ASSIGNED GO TO STATEMENT	STOR	4
	COMMON/CALCOMP/A(520),NPX,NPY,FACT,I,J,JMAX,PENZ,F200,XMIN,DX,YMIN	STOR	5
	I,DY,IBA,IWORD(10),RECNO,JXY(4),IENT,LTAPE,C200	STOR	6
	DATA ENDPIT/0170011000000000000000/	STOR	7
	IWORD(1) = IARG	STOR	8
	I=I+1	STOR	9
	IF (I.LE.10) GO TO 999	STOR	10
100	CALL ENCODE (IWORD,A(J))	STOR	11
	J=J+2	STOR	12
	I=1	STOR	13
	IF (J.LT.JMAX) GO TO 999	STOR	14
320	A(J)=ENDPIT	STOR	15
	CALL TAPWR(9,J)	STOR	16
	J=J+3	STOR	17
	JMAX=519	STOR	18
999	RETURN	STOR	19
	END	STOR	20

	IDENT	ENCODE	STOR	21
	ENTRY	ENCODE	ENCO	1
	VFD	42/OH	ENCO	2
	ENCODE PACKS 1 CHARACTER FROM 10 CONSECUTIVE WORDS	ENCO	3	
*	BEGINNING WITH 1 TO 1 WORD OF 10 CHARACTERS EACH BEG. WITH A.	ENCO	4	
ENCODE	DATA 0	ENCO	5	
	SB5 2 .OUTPUT POINTER	ENCO	6	
LOOP	SB4 54	ENCO	7	
	SX6 B0 .ZERO OUTPUT	ENCO	8	
LOOP	SA3 B1	ENCO	9	
	LX3 B4,X3	ENCO	10	
	BX6 X6+X3 .IN X6	ENCO	11	
	SB4 B4-12 .DECREMENT SHIFT COUNTER	ENCO	12	
	SB1 B1+1 .INCR I WORD	ENCO	13	
	LT B4,B0,OUT	ENCO	14	
	ZR B3,LOOP	ENCO	15	
OUT	SA6 B2 .STORE FULL WD.	ENCO	16	
	SB5 B5-1 .DECR FULL WD CNT	ENCO	17	
	SB2 B2+1 .INCR I CNT	ENCO	18	
	NZ B5,LOOP	ENCO	19	
	ZR B0,ENCODE .EXIT	ENCO	20	
	END	ENCO	21	
	IDENT	SYMB	SYMB	1
*	SYMBOL ROUTINE FOR THE CALCOMP ELECTRO-MECHANICAL PLOTTER	SYMB	2	
*	CALL SYMBOL (X,Y,HT,BCD,TH,NS)	SYMB	3	
*	WHERE (X,Y) IS THE COORDINATE OF THE LOWER LEFT HAND CORNER	SYMB	4	
*	OF THE FIRST SYMBOL TO BE ANNOTATED.	SYMB	5	
*	HT IS THE HEIGHT OF THE SYMBOL IN INCHES.	SYMB	6	
*	BCD IS THE ARRAY CONTAINING BCD INFORMATION TO BE ANNOTATED.	SYMB	7	
*	TH IS THE ANGLE IN DEGREES AT WHICH THE ANNOTATION WILL APPEAR	SYMB	8	
*	NS IF POSITIVE, IS A COUNT OF THE NUMBER OF CHARACTERS	SYMB	9	
*	TO BE ANNOTATED.	SYMB	10	
*	IF NEGATIVE, ONE CHARACTER WILL BE ANNOTATED.	SYMB	11	
*	BCD MUST BE AN INTEGER VALUE WHICH CORRESPONDS TO THE	SYMB	12	
*	SYMBOL DESIRED. (SEE SYMBOL TABLE FOR CORRESPONDENCE.)	SYMB	13	
*	IF NS IS -1, THE MOVE TO (X,Y) IS WITH PEN UP.	SYMB	14	
*	IF NS IS LESS THAN -1, THE MOVE IS WITH PEN DOWN.	SYMB	15	
*	IF BCD IS LESS THAN 15, (X,Y) REFERENCES THE CENTER OF	SYMB	16	
*	THE SYMBOL.	SYMB	17	
	EXT SDG,CDG,PLTZ	SYMB	18	
	ENTRY	SYMB	19	
	VFD	SYMB	20	
	42/OH	SYMB	21	
SYMBOL	DATA 0	SYMB	22	
D1	SB7 6	SYMB	23	
	SA3 B1	SYMB	24	
	SA4 B2	SYMB	25	
	SA5 NOCHN .PICK UP NEW Y COORDINATE	SYMB	26	
	FX0 X5-X3 .GET COMPARE FOR NO NEW COORDINATE	SYMB	27	
	NG X0,GETX .SHOULD X CONTINUE FROM LAST	SYMB	28	
		SYMB	29	
		SYMB	30	
		SYMB	31	
		SYMB	32	
		SYMB	33	
		SYMB	34	
		SYMB	35	
		SYMB	36	
		SYMB	37	
		SYMB	38	
		SYMB	39	
		SYMB	40	
		SYMB	41	
		SYMB	42	
		SYMB	43	
		SYMB	44	
		SYMB	45	
		SYMB	46	
		SYMB	47	
		SYMB	48	
		SYMB	49	
		SYMB	50	
		SYMB	51	
		SYMB	52	
		SYMB	53	
		SYMB	54	
		SYMB	55	
		SYMB	56	
		SYMB	57	
		SYMB	58	
		SYMB	59	
		SYMB	60	
		SYMB	61	
		SYMB	62	
		SYMB	63	
		SYMB	64	
		SYMB	65	
		SYMB	66	
		SYMB	67	
		SYMB	68	
		SYMB	69	
		SYMB	70	
		SYMB	71	
		SYMB	72	
		SYMB	73	
		SYMB	74	
		SYMB	75	
		SYMB	76	
		SYMB	77	
		SYMB	78	
		SYMB	79	
		SYMB	80	
		SYMB	81	
		SYMB	82	
		SYMB	83	
		SYMB	84	
		SYMB	85	
		SYMB	86	
		SYMB	87	
		SYMB	88	
		SYMB	89	
		SYMB	90	
		SYMB	91	
		SYMB	92	
		SYMB	93	
		SYMB	94	
		SYMB	95	
		SYMB	96	
		SYMB	97	
		SYMB	98	
		SYMB	99	
		SYMB	100	

	BX6	X3	.STORE NRW X	SYMB	28
	SA6	X0	.IN X0	SYMB	29
GETX	FX0	X5-X4	.COMPARE FOR NEW Y COORDINATE	SYMB	30
	NG	X0,GETHT	.LEAVE Y AT LAST SETTING	SYMB	31
	BX7	X4	.SET Y0 TO NRW Y COORDINATE	SYMB	32
	SA7	Y0	.AND STORE IT	SYMB	33
GETHT	SA1	B5	.PICK UP THETA	SYMB	34
	SA3	B3	.PICK UP NEW HEIGHT	SYMB	35
	SA4	B6		SYMB	36
	SA2	B4	.PICK UP LOCATION OF BCD	SYMB	37
	BX6	X3	.SAVE NEW HEIGHT	SYMB	38
	BX7	X2	.SAVE NEW BCD LOCATION	SYMB	39
	SA6	HT	.STORE HEIGHT IN HT	SYMB	40
	SA7	BCD	.STORE BCD LOCATION IN BCD	SYMB	41
	BX6	X4		SYMB	42
	SA6	NT	.SAVE CHARACTER COUNT	SYMB	43
	BX6	X1	.PICK UP ANGLE IN DEGREES	SYMB	44
	SA6	THETA	.STORE DEGREES	SYMB	45
	SB1	THETA	.SET PARAMETER FOR JUMP TO COS	SYMB	46
	RJ	CDG	.COMPUTE COSINE OF THETA	SYMB	47
	SA6	COST	.SAVE COSINE	SYMB	48
	SB1	THETA	.SET PARAMETER FOR JUMP TO SINE	SYMB	49
	RJ	SDG	.COMPUTE SIN	SYMB	50
	SA6	SINT	.STORE SIN IN SINT	SYMB	51
BIOCT	SA6	3	.SET FOR PEN UP	SYMB	52
	SA6	IC	.SET IC FOR PEN UP (BEAM OFF)	SYMB	53
	SA1	NT	.PICK UP CHARACTER COUNT	SYMB	54
	SA5	HT	.PICK UP HEIGHT	SYMB	55
	SB5	0	.INITIALIZE SHIFT COUNT FOR RIGHT CHARACTER	SYMB	56
	SA6	1	.INITIALIZE CHARACTER COUNT FOR 1 CHAR	SYMB	57
	SA6	NT	.SAVE IT IN NT	SYMB	58
	SB3	TAB2	.SET TABLE ADDRESS FOR BCD CHARACTERS	SYMB	59
	ZR	X1,ONECH	.SKIP FOR ONE RIGHT BCD	SYMB	60
	NG	X1,SPECS	.SKIP FOR SPECIAL SYMBOLS	SYMB	61
	BX6	X1	.PLACE CHARACTER COUNT IN X6	SYMB	62
	SA6	NT	.SAVE IT IN NT	SYMB	63
	SB5	-54	.SET SHIFT COUNT FOR LEFT CHARACTER	SYMB	64
ONECH	SA2	SEVEN	.PICK UP HEIGHT FACTOR	SYMB	65
HTBY7	FX2	X5/X2	.COMPUTE GRID SIZE	SYMB	66
CXA	SA5	COST	.PICK UP COSINE OF THETA	SYMB	67
	SA3	SINT	.PICK UP SIN OF THETA	SYMB	68
	FX6	X2*X5	.COMPUTE SIN*HT FACTOR	SYMB	69
	SA6	XA	.STORE AS GRID MOVE IN X	SYMB	70
	SA4	FSIX	.PICK UP FACTOR FOR NEXT CHARACTER	SYMB	71
	FX6	X6*X4	.COMPUTE NEXT CHAR LOCATION	SYMB	72
	SA6	XT	.XT IS X COORDINATE OF NEXT CHAR	SYMB	73
	FX6	X2*X3	.COMPUTE FACTOR * SIN	SYMB	74
	SA6	YA	.STORE FOR Y GRID SIZE	SYMB	75
	FX6	X6*X4	.COMPUTE Y COORDINATE OF NEXT CHAR	SYMB	76
	SA6	YT	.YT IS Y LOCATION OF NEXT CHAR	SYMB	77
PICK	SA1	BCD	.PICK UP ADDRESS OF BCE	SYMB	78
	SA3	MASK1	.BRING IN MASK FOR CHARACTER	SYMB	79
	SA1	X1		SYMB	80
	LX1	B5,X1	.PLACE CURRENT CHAR IN LOW 6 BITS OF X1	SYMB	81
	BX2	X1*X3	.MASK OFF REST OF WORD	SYMB	82
	SB2	X2	.PLACE CHARACTER IN REGISTER	SYMB	83
	SB2	B2+B3	.COMPUTE ADDRESS TABLE LOCATION	SYMB	84
	SA4	B2	.PICK UP CONTENTS OF ADDRESS TABLE	SYMB	85
	BX0	X4*X3	.MASK FOR OFFSET TABLE LOCATION	SYMB	86
	SB2	TAB3	.PICK UP ADDRESS OF OFFSET TABLE	SYMB	87
	SB6	X0	.PLACE TABLE LOCATION IN B6	SYMB	88
	SB2	B2+B6	.COMPUTE LOCATION OF OFFSETS	SYMB	89
	AX4	6	.SHIFT TO GET OFFSET COUNT	SYMB	90
	BX0	X4*X3	.MASK OFF THE REST OF WORD	SYMB	91
	SB1	X0	.PLACE OFFSET COUNT IN INDEX 1	SYMB	92
	ZR	B1,NEWCH	.JUMP IF NO OFFSETS IN TABLE	SYMB	93
	AX4	6	.PICK UP LOCATION WITHIN OFFSET WORD	SYMB	94
	BX0	X3*X4	.MASK OFF THE REST OF ADDRESS TABLE	SYMB	95
	SB4	X0	.SAVE INTERNAL LOCATION IN SHIFT COUNT	SYMB	96
	SB4	B4-57	.COMPUTE START OF OFFSETS	SYMB	97
OFFFS1	SA2	B2	.PICK UP BCD WORD	SYMB	98
	SA1	X0		SYMB	99
	SA4	Y0	.PICK UP Y COORDINATE	SYMB	100
	BX6	X1	.SAVE X COORDINATE IN X6	SYMB	101
	BX7	X4	.SAVE Y COORDINATE	SYMB	102
	SA1	XA	.PICK UP X GRID SIZE	SYMB	103
	SA4	YA	.PICK UP Y GRID SIZE	SYMB	104
OFFS	LX0	B4,X2	.PICK UP CURRENT OFFSET PAIR	SYMB	105
	SB7	7	.PICK UP PEN UP CHECK	SYMB	106
	SA3	MASK1	.BRING IN MASK FOR CHARACTER	SYMB	107
	SA5	MASK2	.BRING IN MASK FOR CHARACTER	SYMB	108
	BX0	X0*X5		SYMB	109
	SB6	X0	.PICK UP X OFFSET COUNT	SYMB	110
	EQ	B6,B7,PUP	.SKIP IF PEN UP COMMAND(NX=7)	SYMB	111
	SB4	B4+3	.INCREMENT FOR Y COUNT	SYMB	112
	LX0	B4,X2	.PICK UP Y OFFSET	SYMB	113
	SB4	B4+3	.INCREMENT OFFSET LOCATION COUNT	SYMB	114
	BX0	X0*X5	.MASK Y LOOP COUNT (NY)	SYMB	115
	SB7	X0	.PLACE COUNT IN INDEX	SYMB	116
	ZR	B6,YLOOP	.SKIP COMPUTATION IF NX=0	SYMB	117
XLOOP	FX6	X6+X1	.XN=XN+XA	SYMB	118
	FX7	X7+X4	.YN=YN+YA	SYMB	119
	SD6	B6-1	.DECREMENT NX	SYMB	120
	NZ	B6,XLOOP	.GO BACK IF NOT FINISHED	SYMB	121
YLOOP	ZR	B7,SET	.SKIP NY LOOP IF NY=0	SYMB	122

	FX6 X6-X4	.XN=XN-YA	SYMB 123
	FX7 X7+X1	.YN=YN+XA	SYMB 124
	SB7 B7-1	.DECREMENT NY	SYMB 125
SET	NZ B7,YLOOP	.GO BACK IF NOT FINISHED WITH NY	SYMB 126
	SA7 YN	.STORE NEW Y COORDINATE	SYMB 127
	SA6 XN	.STORE NEW X COORDINATE	SYMB 128
	SX6 B1	.SAVE B1	SYMB 129
	SX7 B2	.SAVE B2	SYMB 130
	SA6 SAV1		SYMB 131
	SA7 SAV2		SYMB 132
	SX6 B3	.SAVE B3	SYMB 133
	SX7 B4	.SAVE B4	SYMB 134
	SA6 SAV3		SYMB 135
	SA7 SAV4		SYMB 136
	SX6 B5	.SAVE B5	SYMB 137
	SA6 SAV5		SYMB 138
	SB1 XN	.SET PARAMETER FOR BRANCH TO PLOT	SYMB 139
	SB2 YN	.SET Y PARAMETER FOR BRANCH TO PLOT	SYMB 140
	SB3 IC	.SET PEN COMMAND PARAM FOR BRANCH TO PLOT	SYMB 141
	RJ PLTZ	.GO TO MAKE MOVE	SYMB 142
	SA1 SAV1	.RESTORE B2	SYMB 143
	SA2 SAV2	.RESTORE B2	SYMB 144
	SA3 SAV3	.RESTORE B3	SYMB 145
	SA4 SAV4	.RESTORE B4	SYMB 146
	SA5 SAV5	.RESTORE B5	SYMB 147
	SB1 X1		SYMB 148
	SB2 X2		SYMB 149
	SB3 X3		SYMB 150
	SB4 X4		SYMB 151
	SB5 X5		SYMB 152
CONT	SX6 2	.SET FOR PEN DOWN	SYMB 153
	SA6 IC	.SET PEN COMMAND TO ON	SYMB 154
	SB1 B1-1	.DECREMENT OFFSET COUNT	SYMB 155
	ZR B1,NEWCH	.SKIP IF FINISHED WITH OFFSETS	SYMB 156
	NG B4,OFFS1	.IS OFFSET WORD FINISHED	SYMB 157
	SB4 -57	.YES,RESTORE SHIFT COUNT OF OFFSET WORD	SYMB 158
	SB2 B2+1	.GET LOCATION OF NEXT OFFSET WORD	SYMB 159
NEWCH	ZR B0,OFFS1	.GOBACK FOR NEXT OFFSET PAIR	SYMB 160
	SA1 NT	.PICK UP CHARACTER COUNT	SYMB 161
	SX7 3		SYMB 162
	SA7 IC	.SET PEN COMMAND TO BEAM OFF	SYMB 163
	SA4 X0	.PICK UP LAST COORDINATE	SYMB 164
	BX6 X4	.PLACE IN X6 (SAVE TEMP)	SYMB 165
	SA4 XT	.PICK UP START OF NEXT CHARACTER (X)	SYMB 166
	FX6 X4+X6	.COMPUTE LOCATION OF NEXT CHARACTER IN X	SYMB 167
	SA6 X0	.SAVE IT IN X0	SYMB 168
	SX6 X1-1	.DECREMENT CHARACTER COUNT	SYMB 169
	SA4 Y0	.PICK UP LAST Y COORDINATE	SYMB 170
	BX7 X4	.SAVE IT IN X7 (TEMPORARILY)	SYMB 171
	SA6 NT	.PICK UP Y LOCATION OF NEXT CHAR.	SYMB 172
	SA4 YT	.PICK UP Y LOCATION OF NEXT CHARACTER	SYMB 173
	FX7 X4+X7	.COMPUTE Y COORDINATE OF NEXT CHARACTER	SYMB 174
	SA7 Y0	.SAVE IT IN Y0	SYMB 175
	ZR X6,SYMBOL	.EXIT IF NO MORE CHARACTERS	SYMB 176
	SB5 B5+6	.INCREMENT SHIFT COUNT FOR BCD	SYMB 177
	SB6 1		SYMB 178
	LT B5,B6,PICK		SYMB 179
	SB5 -54	.REINITIALIZE LCD SHIFT COUNT	SYMB 180
	SA1 BCD	.PICK UP ADDRESS OF BCD	SYMB 181
	SX6 X1+1	.INCREMENT ADDRESS BY 1	SYMB 182
	SA6 BCD	.SAVE NEW ADDRESS	SYMB 183
SPECS	ZR B0,PICK	.GO BACK FOR NEXT CHARACTER	SYMB 184
	SX1 X1+1	.CHECK FOR NS=	SYMB 185
	SB3 TAB1		SYMB 186
	ZR X1,PENUP	.SKIP TO LEAVE BEAM ON FOR FIRST MOVE	SYMB 187
	SX6 2	.SET FOR PEN DOWN	SYMB 188
PENUP	SA6 IC	.SAVE IT	SYMB 189
	SA1 BCD		SYMB 190
	SA1 X1		SYMB 191
	SB1 B0	.SET SHIFT COUNT FOR RIGHT JUSTIFIED CHAR	SYMB 192
	SX1 X1-15		SYMB 193
	PL X1,ONECH	.LEAVE AT NORMAL SIZE WITH OUT 0 CENTERING	SYMB 194
	SA1 FFOUR	.4 X 4 GRID SIZE	SYMB 195
	FX2 X5/X1	.COMPUTE HEIGHT FACTOR	SYMB 196
	SA1 SINT	.PICK UP SIN OF THETA	SYMB 197
	SA5 COST	.PICK UP COS OF THETA	SYMB 198
	SA3 X0	.PICK UP X COORDINATE	SYMB 199
	SA4 Y0	.PICK UP Y COORDINATE	SYMB 200
	FX0 X2+X2	.COMPUTE 2*FACTOR	SYMB 201
	FX6 X5-X1	.COMPUTE COS-SIN	SYMB 202
	FX7 X5+X1	.COMPUTE COS+SIN	SYMB 203
	FX6 X6*X0	.COMPUTE X OFFSET ORIGIN	SYMB 204
	FX7 X7*X0		SYMB 205
	FX6 X3-X6		SYMB 206
	FX7 X4-X7		SYMB 207
	SA6 X0	.STORE X OFFSET ORIGIN	SYMB 208
	SA7 Y0	.STORE Y OFFSET ORIGIN	SYMB 209
	ZR B0,CXA	.GO BACK FOR MAIN FLOW	SYMB 210
PUP	SX6 3	.SET FOR PEN UP	SYMB 211
	SB4 B4+6	.INCREMENT OFFSET SHIFT COUNT	SYMB 212
	ZR B0,CONT	.GO FOR NEXT OFF SET PAIR	SYMB 213
SEVEN	DATA 7.0		SYMB 214
FFOUR	DATA 4.0		SYMB 215
XA	DATA 0		SYMB 216
YA	DATA 0		SYMB 217

TAB2

DATA	0
DATA	0
DATA	0
DATA	0
DATA	0
DATA	78
DATA	778
DATA	1
DATA	0
DATA	0
DATA	0
DATA	0
DATA	6-0
DATA	0
DATA	0
DATA	0
DATA	0
DATA	C
DATA	998.
DATA	1000B
DATA	14018
DATA	06178
DATA	07138
DATA	4407138
DATA	07038
DATA	07048
DATA	10058
DATA	13068
DATA	07108
DATA	15118
DATA	15138
DATA	1406148
DATA	04138
DATA	14178
DATA	02238
DATA	10238
DATA	0602448
DATA	05218
DATA	05228
DATA	10238
DATA	05248
DATA	15258
DATA	3008348
DATA	03308
DATA	1410028
DATA	1410208
DATA	16338
DATA	11358
DATA	1414368
DATA	1410468
DATA	07368
DATA	07408
DATA	06408
DATA	14418
DATA	0606448
DATA	06438
DATA	3008478
DATA	2206458
DATA	03448
DATA	05458
DATA	1404428
DATA	12468
DATA	2207508
DATA	0614468
DATA	2211508
DATA	15608
DATA	04538
DATA	2207478
DATA	5203408
DATA	05508
DATA	05528
DATA	3605528
DATA	1410538
DATA	0611468
DATA	3605228
DATA	11628
DATA	15548
DATA	0604648
DATA	6012558
DATA	14568
DATA	3605648
DATA	21548
DATA	13638
DATA	6605678
DATA	1402708
DATA	6613678
DATA	02528
DATA	1404518
DATA	4404518
DATA	5215308
DATA	1405028

H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

STUVWXYZ0123456789+!*#

SYMB	218
SYMB	219
SYMB	220
SYMB	221
SYMB	222
SYMB	223
SYMB	224
SYMB	225
SYMB	226
SYMB	227
SYMB	228
SYMB	229
SYMB	230
SYMB	231
SYMB	232
SYMB	233
SYMB	234
SYMB	235
SYMB	236
SYMB	237
SYMB	238
SYMB	239
SYMB	240
SYMB	241
SYMB	242
SYMB	243
SYMB	244
SYMB	245
SYMB	246
SYMB	247
SYMB	248
SYMB	249
SYMB	250
SYMB	251
SYMB	252
SYMB	253
SYMB	254
SYMB	255
SYMB	256
SYMB	257
SYMB	258
SYMB	259
SYMB	260
SYMB	261
SYMB	262
SYMB	263
SYMB	264
SYMB	265
SYMB	266
SYMB	267
SYMB	268
SYMB	269
SYMB	270
SYMB	271
SYMB	272
SYMB	273
SYMB	274
SYMB	275
SYMB	276
SYMB	277
SYMB	278
SYMB	279
SYMB	280
SYMB	281
SYMB	282
SYMB	283
SYMB	284
SYMB	285
SYMB	286
SYMB	287
SYMB	288
SYMB	289
SYMB	290
SYMB	291
SYMB	292
SYMB	293
SYMB	294
SYMB	295
SYMB	296
SYMB	297
SYMB	298
SYMB	299
SYMB	300
SYMB	301
SYMB	302
SYMB	303
SYMB	304
SYMB	305
SYMB	306
SYMB	307
SYMB	308
SYMB	309
SYMB	310
SYMB	311
SYMB	312

TAB3

```

DATA 0
DATA 2206678 . . .
DATA 2205678 . . .
DATA 3006128 . . .
DATA 5202578 . . .
DATA 5203348 . . .
DATA 5213668 . . .
DATA 3605378 . . .
DATA 0016658 . . .
DATA 4404428 . . .
DATA 5203128 . . .
DATA 5215268 . . .
DATA 0007668 . . .
DATA 4404438 . . .
DATA 5214668 . . .
DATA 0004308 . . .
DATA 5211268 . . .
DATA 2207618 . . .
DATA 2202528 . . .
DATA 222404004044242200008
DATA 222414030110304143348
DATA 242204447001417035108
DATA 222402204224220000008
DATA 220224202442220000008
DATA 220444220022402200008
DATA 221232224404440040008
DATA 220506173746450100408
DATA 220422442220220000008
DATA 224433130413110011318
DATA 403133220141704204468
DATA 222420220242220440228
DATA 004422044400402200008
DATA 0
DATA 0
DATA 222401412422230320438
DATA 232201417004442426222
DATA 272011317043031412038
DATA 20271636730102027168
DATA 004070034370064600008
DATA 034334324300000000008
DATA 0616170706700004770408
DATA 413130140000002021118
DATA 312126251535252313338
DATA 012541013010000211318
DATA 424304051646457027208
DATA 044470014170351000008
DATA 031312141323222423338
DATA 323433430141700244068
DATA 000343030617374640008
DATA 374641300007374645348
DATA 043443410001122027478
DATA 470704340400400720478
DATA 533343413010010617378
DATA 464500074047301017378
DATA 301020271737103037178
DATA 400007044447400000008
DATA 404724070003472540008
DATA 354641301001061737468
DATA 702240070201103041478
DATA 474024000737464534048
DATA 244020111627203136278
DATA 004770074007242024478
DATA 202707472414342400408
DATA 061737464534143443418
DATA 301001031405060047078
DATA 043443413010010206178
DATA 374606074746202527008
DATA 020110304143341405068
DATA 173746061726202637468
DATA 050617374645010040008
DATA 011030414637170604138
DATA 433037024206074746208
DATA 050617374645342422708
DATA 103021107022272524348
DATA 352570312122323120218
DATA 252303432301452305418
END

```

```

SYMB 313
SYMB 314
SYMB 315
SYMB 316
SYMB 317
SYMB 318
SYMB 319
SYMB 320
SYMB 321
SYMB 322
SYMB 323
SYMB 324
SYMB 325
SYMB 326
SYMB 327
SYMB 328
SYMB 329
SYMB 330
SYMB 331
SYMB 332
SYMB 333
SYMB 334
SYMB 335
SYMB 336
SYMB 337
SYMB 338
SYMB 339
SYMB 340
SYMB 341
SYMB 342
SYMB 343
SYMB 344
SYMB 345
SYMB 346
SYMB 347
SYMB 348
SYMB 349
SYMB 350
SYMB 351
SYMB 352
SYMB 353
SYMB 354
SYMB 355
SYMB 356
SYMB 357
SYMB 358
SYMB 359
SYMB 360
SYMB 361
SYMB 362
SYMB 363
SYMB 364
SYMB 365
SYMB 366
SYMB 367
SYMB 368
SYMB 369
SYMB 370
SYMB 371
SYMB 372
SYMB 373
SYMB 374
SYMB 375
SYMB 376
SYMB 377
SYMB 378
SYMB 379
SYMB 380
SYMB 381
SYMB 382
SYMB 383
SYMB 384
SYMB 385
SYMB 386
SYMB 387
SYMB 388
SYMB 389

```

```

SUBROUTINE NUMBER( XPAGE, YPAGE, HEIGHT, FPN, ANGLE, NDEC )
C..... XPAGE,YPAGE COORDINATES OF LOWER LEFT CORNER OF NUMBER.
C..... HEIGHT HEIGHT OF PLOTTED NUMBER.
C..... FPN FLOATING POINT NUMBER TO BE PLOTTED.
C..... ANGLE ANGLE AT WHICH NUMBER IS PLOTTED, IN DEGREES.
C..... NDEC NUMBER OF DECIMAL PLACES TO BE DRAWN.
C..... THIS VERSION OF NUMBER REQUIRES THE SYMBOL VERSION WITH
C..... 999. X, Y FEATURE, AND NC = 0 FEATURE.
DATA MAXN/9/,SAMEV/999./
X = XPAGE
Y = YPAGE
N = NDEC
IF( IABS(N).GT.MAXN ) N=ISIGN(MAXN,N)
IF( FPN.GE.0. ) GO TO 30
21 CALL SYMBOL (X,Y,HEIGHT,1H-,ANGLE,1)
X = SAMEV
Y = SAMEV
30 MN = -N

```


NUMB	19
NUMB	20
NUMB	21
NUMB	22
NUMB	23
NUMB	24
NUMB	25
NUMB	26
NUMB	27
NUMB	28
NUMB	29
NUMB	30
NUMB	31
NUMB	32
NUMB	33
NUMB	34
NUMB	35
NUMB	36
NUMB	37
NUMB	38
NUMB	39
NUMB	40
NUMB	41
NUMB	42
NUMB	43

SCAL	1
SCAL	2
SCAL	3
SCAL	4
SCAL	5
SCAL	6
SCAL	7
SCAL	8
SCAL	9
SCAL	10
SCAL	11
SCAL	12
SCAL	13
SCAL	14
SCAL	15
SCAL	16
SCAL	17
SCAL	18
SCAL	19
SCAL	20
SCAL	21
SCAL	22
SCAL	23
SCAL	24
SCAL	25
SCAL	26
SCAL	27
SCAL	28
SCAL	29
SCAL	30
SCAL	31
SCAL	32
SCAL	33
SCAL	34
SCAL	35
SCAL	36
SCAL	37
SCAL	38
SCAL	39
SCAL	40
SCAL	41
SCAL	42
SCAL	43
SCAL	44
SCAL	45
SCAL	46
SCAL	47
SCAL	48
SCAL	49
SCAL	50
SCAL	51
SCAL	52
SCAL	53
SCAL	54
SCAL	55
SCAL	56
SCAL	57
SCAL	58
SCAL	59
SCAL	60
SCAL	61
SCAL	62
SCAL	63
SCAL	64
SCAL	65

100	T=0.	SCAL	66
	IF(AXLE.LE.0)GOTO110	SCAL	67
	IF(YO.LT.1.) T=.999999999	SCAL	68
	FIRSTV=AIN(T(ALOG10(YO)-T)	SCAL	69
	T=0.	SCAL	70
	IF(YN.GT.1.) T=.999999999	SCAL	71
	DELTAV=(AINT(ALOG10(YN)+T)-FIRSTV)/AXLEN	SCAL	72
	GO TO 65	SCAL	73
110	FIRSTV=ALOG10(YO)	SCAL	74
	ENDV=ALOG10(YN)	SCAL	75
	FN=AINT(FIRSTV)	SCAL	76
	IF(FIRSTV.LT.0.)FN=FN-1.	SCAL	77
	X=FIRSTV-FN	SCAL	78
	DO115I=2,10	SCAL	79
	IF(X.LT.BLOG(I)-.000001) GO TO 120	SCAL	80
115	CONTINUE	SCAL	81
	I=11	SCAL	82
120	FIRSTV=FN+BLOG(I-1)	SCAL	83
	FN=AINT(ENOV)	SCAL	84
	IF(ENOV.LT.0.) FN=FN-1.	SCAL	85
	X=ENOV-FN	SCAL	86
	DO125I=1,10	SCAL	87
	IF(X.LE.BLOG(I)+.000001) GO TO 130	SCAL	88
125	CONTINUE	SCAL	89
130	ENDV=FN+BLOG(I)	SCAL	90
	IF(AXLE.EQ.0)GOTO140	SCAL	91
	DELTAV=(ENDV-FIRSTV)/AXLEN	SCAL	92
	GOTO65	SCAL	93
140	AXLE=(ENDV-FIRSTV)/DELTAV	SCAL	94
	GOTO65	SCAL	95
	END	SCAL	96

	SUBROUTINE PLTZ(X1,Y1,IC)	PLTZ	1
	THIS ROUTINE GENERATES A TAPE FOR THE CALCOMP 670 TAPE UNIT,	PLTZ	2
	AND A CALCOMP 500 SERIES PLOTTER,	PLTZ	3
	WITH A TAPE DENSITY OF 556 BPI.	PLTZ	4
	THIS PLOT ROUTINE HAS THE ENTRIES AND PARAMETERS DESCRIBED BELOW.	PLTZ	5
	CALL PLTZ (X,Y,IC)	PLTZ	6
	(X,Y) IS THE COORDINATE IN INCHES TO WHICH THE PEN WILL	PLTZ	7
	MOVE.	PLTZ	8
	IF X,Y IS 999. THEN PEN IS NOT MOVED AT ALL	PLTZ	9
	IC IS A PEN COMMAND.	PLTZ	10
	IF IC IS LESS THAN 14 AND.....	PLTZ	11
	IF THE UNITS DIGIT OF IC IS 2,MOVE MADE WITH PEN DOWN.	PLTZ	12
	IF THE UNITS DIGIT OF IC IS 3,MOVE MADE WITH PEN UP.	PLTZ	13
	IF THE TENS DIGIT IS 1, OFFSET FACTORS ARE USED.	PLTZ	14
	(SEE DESCRIPTION OF OFFSET)	PLTZ	15
	BUT IF IC IS GT 14, BUFFER IS DUMPED, AND BLOCK NO.=IC	PLTZ	16
	IS WRITTEN ON THE PLOT TAPE.(MOVE IS MADE TO (X,Y)	PLTZ	17
	WITH PEN UP.)	PLTZ	18
	IF IC IS NEGATIVE A NEW ORIGIN IS ESTABLISHED AT (X,Y).	PLTZ	19
	CALL PLOTS(LDV)	PLTZ	20
	LDV IS THE FORTRAN LOGICAL DEVICE NUMBER OF THE PLOT TAPE	PLTZ	21
	CALL WHERE(XN,YN,FACT)	PLTZ	22
	(XN,YN) IS THE COORDINATE(IN INCHES) OF THE CURRENT PEN	PLTZ	23
	POSITION.	PLTZ	24
	FACT IS THE CURRENT SCALING FACTOR.	PLTZ	25
	CALL FACTOR(FACT)	PLTZ	26
	FACT IS A SCALING FACTOR FOR ALL SUBSEQUENT PLOTTING.	PLTZ	27
	EG. CALL FACTOR (0.5) PRODUCES A HALF SIZE DRAWING.	PLTZ	28
	CALL OFFSET(XMIN,DX,YMIN,DY)	PLTZ	29
	PROVIDES SCALING FACTORS FOR SUBSEQUENT CALLS TO PLOT	PLTZ	30
	WHEN IC EQUALS 12,13,-12, OR -13.	PLTZ	31
	THE COORDINATE TO WHICH THE PEN MOVES (XT,YT) IS COMPUTED	PLTZ	32
	XT = (X-XMIN)/DX YT = (Y-YMIN)/DY	PLTZ	33
	COMMON/CALCOMP/A(520),NPX,NPY,FACT,I,J,JMAX,PENZ,F200,XMIN,DX,YMIN	PLTZ	34
	1,DY,IBA,IWORD(10),RECNO,JXY(4),IENT,LTAPE,C200	PLTZ	35
	EQUIVALENCE (STEPS,F200)	PLTZ	36
	DATA IENT/1/	PLTZ	37
	IF(IENT.EQ.1) GO TO 300	PLTZ	38
	Y=Y1	PLTZ	39
	IF(Y1.EQ.999.) Y=FLOAT(NPY)/F200	PLTZ	40
	X=X1	PLTZ	41
	IF(X1.EQ.999.) X=FLOAT(NPX)/F200	PLTZ	42
	IPEN=1ABS(IC)	PLTZ	43
20	IF (IPEN.LT.10) GO TO 70	PLTZ	44
3000	IF(IPEN.LE.13) GO TO 50	PLTZ	45
	IBA=IPEN	PLTZ	46
	GO TO 80	PLTZ	47
50	I2=IPEN/10	PLTZ	48
	IPEN=IPEN-I2*10	PLTZ	49
	IF (I2.NE.1) GO TO 70	PLTZ	50
60	X=(X-XMIN)/DX	PLTZ	51
	Y=(Y-YMIN)/DY	PLTZ	52
C	THE FOLLOWING CODE EXAMINES PEN COMMANDS ****	PLTZ	53
70	IF (IPEN-2) 130,90,80	PLTZ	54
80	IPDELAY=3	PLTZ	55
	IPCOMM=0000011	PLTZ	56
	IF (IPEN2.GT.0.) GO TO 130	PLTZ	57
85	PENZ=1.0	PLTZ	58
	GO TO 110	PLTZ	59

C	PENZ=1 MEANS PEN IS UP, PENZ=-1 MEANS PEN IS DOWN. ***	PLTIZ	60
90	IPDELAY=20	PLTIZ	61
	IPCOMM=0000012	PLTIZ	62
	IF (PENZ.LT.0.) GO TO 130	PLTIZ	63
100	PENZ=-1.0	PLTIZ	64
110	CALL STORE (IPCOMM)	PLTIZ	65
120	IPDELAY=IPDELAY-1	PLTIZ	66
	IF (IPDELAY.GT.0) GO TO 110	PLTIZ	67
C	THIS SECTION SETS UP THE BASIC LOOP***	PLTIZ	68
130	NX=X*STEPS+SIGNF(.5,X)	PLTIZ	69
	NY=Y*STEPS+SIGNF(.5,Y)	PLTIZ	70
	IDX=NX-NPX	PLTIZ	71
	IDY=NY-NPY	PLTIZ	72
	NPY=NY	PLTIZ	73
	NPX=NX	PLTIZ	74
	IX=0000003	PLTIZ	75
	LL=1	PLTIZ	76
	IF (IDX.GE.0) GO TO 150	PLTIZ	77
140	IX=0000007	PLTIZ	78
	LL=2	PLTIZ	79
	IDX=-IDX	PLTIZ	80
150	IY=0000001	PLTIZ	81
	IF (IDY.GE.0) GO TO 170	PLTIZ	82
160	IY=0000005	PLTIZ	83
	LL=LL+2	PLTIZ	84
	IDY=-IDY	PLTIZ	85
170	IXY=JXY(LL)	PLTIZ	86
	IF (IDX.GE.IDY) GO TO 190	PLTIZ	87
180	IX=IY	PLTIZ	88
	NC=IDY+IDY	PLTIZ	89
	NA=IDY	PLTIZ	90
	NB=IDX+IDX	PLTIZ	91
	ICTR=IDY	PLTIZ	92
	GO TO 250	PLTIZ	93
190	NC=IDX+IDX	PLTIZ	94
	NA=IDX	PLTIZ	95
	NB=IDY+IDY	PLTIZ	96
	ICTR=IDX	PLTIZ	97
	GO TO 250	PLTIZ	98
C	END OF CODING FOR SETUP OF BASIC LOOP***	PLTIZ	99

C	THE FOLLOWING SECTION OF CODE IS THE BASIC ALGORITHM**	PLTIZ	100
200	NA=NA+NB	PLTIZ	101
	IF (NA.LT.NC) GO TO 220	PLTIZ	102
210	NA=NA-NC	PLTIZ	103
	CALL STORE (IXY)	PLTIZ	104
	GO TO 240	PLTIZ	105
220	CALL STORE (IX)	PLTIZ	106
240	ICTR=ICTR-1	PLTIZ	107
250	IF (ICTR.GT.0) GO TO 200	PLTIZ	108
C	** END OF BASIC ALGORITHM ***	PLTIZ	109
C	FOLLOWING IS END OF PICTURE TEST	PLTIZ	110
260	IF (IC.LT.0.OR.IC.GT.13) 270,300	PLTIZ	111
270	IF (I.EQ.10) GO TO 275	PLTIZ	112
2700	CALL STORE (IPCOMM)	PLTIZ	113
	GO TO 270	PLTIZ	114
275	JMAX=J+2	PLTIZ	115
	CALL STORE (IPCOMM)	PLTIZ	116
290	CALL BLOCK (IBA)	PLTIZ	117
	IBA=IBA+1	PLTIZ	118
	IF (IC.GE.0) GO TO 300	PLTIZ	119
295	NPX=0	PLTIZ	120
	NPY=0	PLTIZ	121
	PENZ=0.	PLTIZ	122
300	RETURN	PLTIZ	123
	END	PLTIZ	124

	SUBROUTINE PLOTS (IC)	PLOT	1
	COMMON/CALCOMP/A(520),NPX,NPY,FACT,I,J,JMAX,PENZ,F200,XMIN,DX,YMIN	PLOT	2
	1,DY, IBA,IWORD(10),RECNO,JXY(4),IENT,LTAPE,C200	PLOT	3
	DATA C200/100.0/	PLOT	4
	DATA (A(K),K=1,8)/011001100110011001100,011001100110011001700,	PLOT	5
	1017001700170017001700,017001400060005000600,005000600050014001700,	PLOT	6
	2017001700170017001700,017001100110011001100,011001100110011001100/	PLOT	7
	DATA (A(K),K= 9,12)/011001100110011001100,	PLOT	8
	1011001100110011001100,011001100170017001700,017001700170017001500/	PLOT	9
	DATA NPX,NPY,FACT,F200,XMIN,DX,YMIN,DY,PENZ,INCR,RECNO,1/	PLOT	10
	10,0,1,0,100.0,0.0,1.0,0.0,1.0,0.0,0.036,0.0,1/	PLOT	11
	DATA (JXY(K),K=1,4)/002,010,004,006/	PLOT	12
	IENT=0	PLOT	13
	JMAX=519	PLOT	14
	J=13	PLOT	15
	IBA=1	PLOT	16
	LTAPE=IC	PLOT	17
10	CALL BLOCK (IBA)	PLOT	18
	IBA=IBA+1	PLOT	19
	RETURN	PLOT	20

END

PLOT 21

	IDENT	SINCOS		SINC	1
	ENTRY	SDG,CDC		SINC	2
	ENTRY	SIN,COS		SINC	3
CERR	EXT	SYSTEM		SINC	4
SERR	EQU	13		SINC	5
NSDG	EQU	36		SINC	6
SDG	VFD	42/3LSDG,18/1		SINC	7
	DATA	0		SINC	8
	SA2	B1	ARG IN DEGREES = 90N + Y	SINC	9
	SA3	SDG	PICK UP RETURN ADDRESS	SINC	10
	SB7	B0		SINC	11
	BX6	X3		SINC	12
	BX0	X2		SINC	13
	AX2	42		SINC	14
	SB6	X2	SAVE SIGN	SINC	15
	SA6	SIN	SET RETURN IN SIN	SINC	16
NCDG	EQ	SCDG	GO START ARG REDUCTION	SINC	17
CDG	VFD	42/3LCDG,18/1		SINC	18
	DATA	0		SINC	19
	SA2	B1	ARG IN DEGREES = 90N + Y	SINC	20
	SA3	CDG	PICK UP RETURN	SINC	21
	SB7	1		SINC	22
	BX6	X3		SINC	23
	SB6	B7	SET SIGN POSITIVE	SINC	24
	BX0	X2		SINC	25
SCDG	SA6	COS	SET RETURN IN COS	SINC	26
	AX2	59	START ARG REDUCTION	SINC	27
	SA5	N90		SINC	28
	BX0	X0-X2	ABS(ARG)	SINC	29
	MX4	1	4000000000000000000B	SINC	30
	SB1	1	(ARG/90)H = N + (Y/90)	SINC	31
	FX6	X0/X5	200000000000000000000B	SINC	32
	LX4	59	HIGH ORDER(Y/90)	SINC	33
	DX7	X4+X6	N	SINC	34
	FX2	X4+X6	AH=([ARG/90]*90)H	SINC	35
	FX3	X6*X5	AL=([ARG/90]*90)L	SINC	36
	DX4	X6*X5	ARG - AH	SINC	37
	FX1	X0-X3		SINC	38
	NX4	X4		SINC	39
	NX1	X1		SINC	40
				SINC	41
	FX3	X1-X4	ARG - AH - AL	SINC	42
	NX6	X3		SINC	43
	FX3	X6/X5	LOW ORDER(Y/90)	SINC	44
	NX7	X7		SINC	45
	SA1	P12	P12	SINC	46
	SA5	TDP	TDP	SINC	47
	FX4	X3*X1		SINC	48
	DX5	X7*X1		SINC	49
	RX0	X7*X1	(Y/90)*(PI/2)=ARG IN RAD	SINC	50
	FX4	X4+X5		SINC	51
	XS5	B7	SET X5=1 IF IF ENTRY WAS AT CDG X5=0 IF SDG.	SINC	52
	RX0	X4+X0		SINC	53
	EQ	REG		SINC	54
N90	DATA	90.		SINC	55
NSIN	VFD	42/3CSIN,18/1		SINC	56
SIN	DATA	0	ENTRY	SINC	57
	SA2	B1	ARG	SINC	58
	SB7	B0		SINC	59
	BX0	X2		SINC	60
	AX2	42		SINC	61
	SB6	X2	SAVE SIGN OF ARG	SINC	62
	AX2	59		SINC	63
	EQ	SINCOS		SINC	64
NCOS	VFD	42/3CCOS,18/1		SINC	65
COS	DATA	0	ENTRY	SINC	66
	SA2	B1	ARG	SINC	67
	SB7	1		SINC	68
	SB6	B7	SET SIGN POSITIVE	SINC	69
	BX0	X2		SINC	70
	AX2	59		SINC	71
	SA5	TDP	(2/PI) HIGH ORDER	SINC	72
	SA1	TDP2	(2/PI) LOW ORDER	SINC	73
	BX0	X0-X2	X = ABS(ARG)	SINC	74
	MX4	1	4000000000000000000B	SINC	75
	SB1	1		SINC	76
	FX6	X0*X5	HIGH ORDER X*(2/PI)H	SINC	77
	DX3	X0*X5	LOW ORDER X*(2/PI)L	SINC	78
	LX4	59	GET MASK 2000000000000000000B	SINC	79
	XS5	B7	SET X5=1 IF ENTRY WAS AT COS AND X5=0 IF SIN	SINC	80
	FX1	X0*X1	X * (2/PI)L	SINC	81
	FX2	X6+X4	N = INTEGER PART OF X*(2/PI)	SINC	82
	DX7	X6+X4	YH = FRACTIONAL PART OF {HIGH ORDER X*(2/PI)H}	SINC	83
	FX3	X3+X1	YL = LOW ORDER X*(2/PI)L + X*(2/PI)L	SINC	84
	SB2	B1+B1	SET B2=2	SINC	85
	MX4	58	SET MASK 7777777777777777778	SINC	86
	NX1	B3,X2		SINC	87
	SB5	B7+B7		SINC	88
REG	UX2	B0,X2	CHANGE N TO INTEGER FORMAT	SINC	89

```

SB7 B5+B5      SET RETURN INDEX TO 4 IF COS
EO 83+B0,ERR1 JUMP IF ARG GE. (2**48)*PI/2)
1 X2+X7      ADD PI/2 IF ENTRY WAS AT COS
N17 X0,X7     NO REDUCTION NECESSARY
N18 X0,NBR    PI/2
N19 X0,X3     M=N(MOD4)
N20 P12+X2    Y = YH+YL = FRACTIONAL PART OF X * (2/PI)
N21 X3+X7     M TO B4
N22 X2        (PI/2)*Y
N23 X1*X0     JUMP IF SIGN
N24 B4-B6     M-2
N25 B4-B2     JUMP IF M.EQ.0
N26 B4-B0,NRR ONE
N27 A5+B1     HIGH ORDER 1-YH
N28 X4-X7     LOW ORDER 1-YH
N29 X0,X0     (HIGH ORDER 1-YH) - YL
N30 X4,X7     (HIGH ORDER 1-YH)*(HIGH ORDER 1-YH) - YL
N31 X0,X3     X = (PI/2) * (1-Y)
N32 A0,X0     Z = X*X
N33 X5+X4     C2
N34 X1+X2     Z**2
N35 X0,X0     Q2+Z
N36 A5+B2     Q1*Z
N37 A1+B1     Q0
N38 X6+X6     (Q2+Z)*(Z**2)
N39 X1+X6     Q1*Z + (Q2*Z)*(Z**2)
N40 X2+X6     BOTTOM = Q0 + Q1*Z + Q2*Z**2 + Z**3
N41 X3+X4     X/BOTTOM
N42 X4+X5     I4
N43 X0+X3     P3
N44 A3+B2     P4*Z**2
N45 X1+X7     P2*Z
N46 X2+X6     P1
N47 A2+B1     P3*Z + P4*Z**2
N48 X3+X4     P2*Z**2
N49 X1+X7     (P3*Z + P4*Z**2) * Z**2
N50 X2+X6     P1*Z
N51 X3+X4     P2*Z**2 + P3*Z**3 + P4*Z**4
N52 X4+X5     TOP
N53 X0+X3     TOP * (X/BOTTOM)
N54 A3+B1     SIN(ABS(ARG)) OR COS(ABS(ARG))
N55 X1+X7     GET SIGN
N56 X2+X6     CHECK RANGE OF FUNCTION
N57 X0-X5     SIN(ARG) OR COS(ARG)
N58 X4-X3     INDEXED RETURN
N59 B6        SIN ERROR NUMBER
N60 X0,DONE   JUMP IF SIN ERROR
N61 X3        COS ERROR NUMBER
N62 X4,DONE   JUMP IF SIN ERROR
N63 X0,X3     GET RETURN INFO
N64 X4,DONE   NAME OF ENTRY
N65 X3        SET UP PSEUDO ENTRY
N66 X2+X5     RETURN+1
N67 X1,X1     X3
N68 X4-X1     X2-X5
N69 X4-X1     X1-X1
N70 X4-X1     X4-X1
N71 X4-X1     X4-X1
N72 X4-X1     X4-X1
N73 X4-X1     X4-X1
N74 X4-X1     X4-X1
N75 X4-X1     X4-X1
N76 X4-X1     X4-X1
N77 X4-X1     X4-X1
N78 X4-X1     X4-X1
N79 X4-X1     X4-X1
N80 X4-X1     X4-X1
N81 X4-X1     X4-X1
N82 X4-X1     X4-X1
N83 X4-X1     X4-X1
N84 X4-X1     X4-X1
N85 X4-X1     X4-X1
N86 X4-X1     X4-X1
N87 X4-X1     X4-X1
N88 X4-X1     X4-X1
N89 X4-X1     X4-X1
N90 X4-X1     X4-X1
N91 X4-X1     X4-X1
N92 X4-X1     X4-X1
N93 X4-X1     X4-X1
N94 X4-X1     X4-X1
N95 X4-X1     X4-X1
N96 X4-X1     X4-X1
N97 X4-X1     X4-X1
N98 X4-X1     X4-X1
N99 X4-X1     X4-X1
N100 X4-X1     X4-X1
N101 X4-X1     X4-X1
N102 X4-X1     X4-X1
N103 X4-X1     X4-X1
N104 X4-X1     X4-X1
N105 X4-X1     X4-X1
N106 X4-X1     X4-X1
N107 X4-X1     X4-X1
N108 X4-X1     X4-X1
N109 X4-X1     X4-X1
N110 X4-X1     X4-X1
N111 X4-X1     X4-X1
N112 X4-X1     X4-X1
N113 X4-X1     X4-X1
N114 X4-X1     X4-X1
N115 X4-X1     X4-X1
N116 X4-X1     X4-X1
N117 X4-X1     X4-X1
N118 X4-X1     X4-X1
N119 X4-X1     X4-X1
N120 X4-X1     X4-X1
N121 X4-X1     X4-X1
N122 X4-X1     X4-X1
N123 X4-X1     X4-X1
N124 X4-X1     X4-X1
N125 X4-X1     X4-X1
N126 X4-X1     X4-X1
N127 X4-X1     X4-X1
N128 X4-X1     X4-X1
N129 X4-X1     X4-X1
N130 X4-X1     X4-X1
N131 X4-X1     X4-X1
N132 X4-X1     X4-X1
N133 X4-X1     X4-X1
N134 X4-X1     X4-X1
N135 X4-X1     X4-X1
N136 X4-X1     X4-X1
N137 X4-X1     X4-X1
N138 X4-X1     X4-X1
N139 X4-X1     X4-X1
N140 X4-X1     X4-X1
N141 X4-X1     X4-X1
N142 X4-X1     X4-X1
N143 X4-X1     X4-X1
N144 X4-X1     X4-X1
N145 X4-X1     X4-X1
N146 X4-X1     X4-X1
N147 X4-X1     X4-X1
N148 X4-X1     X4-X1
N149 X4-X1     X4-X1
N150 X4-X1     X4-X1
N151 X4-X1     X4-X1
N152 X4-X1     X4-X1
N153 X4-X1     X4-X1
N154 X4-X1     X4-X1
N155 X4-X1     X4-X1
N156 X4-X1     X4-X1
N157 X4-X1     X4-X1
N158 X4-X1     X4-X1
N159 X4-X1     X4-X1
N160 X4-X1     X4-X1
N161 X4-X1     X4-X1
N162 X4-X1     X4-X1
N163 X4-X1     X4-X1
N164 X4-X1     X4-X1
N165 X4-X1     X4-X1
N166 X4-X1     X4-X1
N167 X4-X1     X4-X1
N168 X4-X1     X4-X1
N169 X4-X1     X4-X1
N170 X4-X1     X4-X1
N171 X4-X1     X4-X1
N172 X4-X1     X4-X1
N173 X4-X1     X4-X1
N174 X4-X1     X4-X1
N175 X4-X1     X4-X1
N176 X4-X1     X4-X1
N177 X4-X1     X4-X1
N178 X4-X1     X4-X1
N179 X4-X1     X4-X1
N180 X4-X1     X4-X1
N181 X4-X1     X4-X1
N182 X4-X1     X4-X1
N183 X4-X1     X4-X1
N184 X4-X1     X4-X1
N185 X4-X1     X4-X1
N186 X4-X1     X4-X1
N187 X4-X1     X4-X1
N188 X4-X1     X4-X1
N189 X4-X1     X4-X1
N190 X4-X1     X4-X1
N191 X4-X1     X4-X1
N192 X4-X1     X4-X1
N193 X4-X1     X4-X1
N194 X4-X1     X4-X1
N195 X4-X1     X4-X1
N196 X4-X1     X4-X1
N197 X4-X1     X4-X1
N198 X4-X1     X4-X1
N199 X4-X1     X4-X1
N200 X4-X1     X4-X1

```


	SB6	REF		UNLO	43
	BX2	-X3		UNLO	44
	EQ	C101.		UNLO	45
REE	SX3	B7+348	.PERFORM OP AND EXIT	UNLO	46
	SB6	REG	.SET EOF WRITE (B7 IS MODE)	UNLO	47
	SX6	B7		UNLO	48
	BX2	-X3		UNLO	49
	SA6	MODE		UNLO	50
	EQ	C101.	.GO WRITE EOF	UNLO	51
REG	SA3	MODE		UNLO	52
	SB7	X3		UNLO	53
	EQ	RED		UNLO	54
REF	SA3	FNAME		UNLO	55
	SX7	B0		UNLO	56
	SA7	X1+5	.CLEAR RECORD COUNT	UNLO	57
	SA4	X1+14		UNLO	58
	PL	X4,REH	.JMP EOF FLAG NOT SET	UNLO	59
	BX6	-X4		UNLO	60
	SA6	A4	.CLEAR EOF FLAG	UNLO	61
REH	SA2	X1+1		UNLO	62
	SX6	X2		UNLO	63
	SX7	X2		UNLO	64
	SA6	A2+1	.SET IN = FIRST	UNLO	65
	SA7	A2+2	.SET OUT = FIRST	UNLO	66
	ZR	X3,UNLODE	.EXIT FLAG NOT SET	UNLO	67
	SA2	X1		UNLO	68
	SX7	B0		UNLO	69
	AX2	18		UNLO	70
	SA7	FNAME	.CLEAR FLAG	UNLO	71
	LX2	18		UNLO	72
	BX6	X2		UNLO	73
	SA6	X1	.CLEAR BUFFER STATUS	UNLO	74
	EQ	UNLODE	.EXIT	UNLO	75
ERR	SX2	MSG		UNLO	76
	SX1	ERRNUM		UNLO	77
	BX7	X3		UNLO	78
	SA7	FNAME	.SAVE FILE NAME	UNLO	79
	RJ	SYSTEM	.GO PROCESS ERROR	UNLO	80
+	LT	B0,NAME		UNLO	81
-	RJ	ABNORML		UNLO	82
+	LT	B0,NAME		UNLO	83
MSG	DIS	3, UNASSIGNED MEDIUM, FILE		UNLO	84
FNAME	BSS	1		UNLO	85
MODE	BSS	1		UNLO	86
	END			UNLO	87

REFERENCES AND NOTES

- N. W. Glass, J. K. Theobald, A. D. Schelberg, J. W. Warren, and L. D. Tatro, "Measurements on the Capture Cross-Section of U^{238} with Bomb Source Neutrons," Conference on Neutron Cross Section Technology, CONF-660303, (1966) pp. 766-772.
- M. V. Harlow, K. D. Ferrell, R. Plaisted, and A. N. Phillips, "RAYMATH, A Problem-Oriented Computer Language Designed for Processing Neutron Cross-Section Data," Third Conference on Neutron Cross Sections and Technology, CONF-710301, (1971) pp. 360-366.
- A. Hammendinger, B. C. Diven, W. K. Brown, A. Ellis, A. Furnish, E. R. Shunk, and R. R. Fullwood, "Time-of-Flight Neutron Cross-Section Measurements Made with Neutrons from Nuclear Explosions," LASL report LA-3478, Part I (1968).
- M. V. Harlow, Jr., and J. R. Neergaard, "RAYMATH, a FORTRAN IV Code to Compile Simple Instructions for Array Mathematics and Manipulation," LASL report LA-4049, (1969).
- L. W. Weston, G. de Saussure, and R. Gwinn, private communication, Oak Ridge National Laboratory.
- N. H. Marshall, J. W. Coddling, O. D. Simpson, and J. R. Smith, "An Automated Cross Section Analysis Program (ACSAP)," Third Conference on Neutron Cross Section and Technology, CONF-710301, (1971) pp. 354-359.
- J. T. Golden, FORTRAN IV Programming and Computing, Prentice-Hall, Inc., (1965) pp. 245-253.
- A. Lapidus, M. Goldstein, and S. S. Hoffberg, "A Computer Program for Doing Tedious Algebra (SYMB66)," USARC Report NYO-1480-88 (1968).
- J. Hilsenrath and G. Galler, "OMNIFORM 1: A General Purpose Machine Program for the Calculation of Tables of Functions Given Explicitly in Terms of One Variable," NBS Technical Note 125, May 1962. See also NBS Technical News Bulletin 43, 8 (1960).
- J. Hilsenrath, G. G. Ziegler, C. G. Messina, P. J. Walsh, and R. J. Herbold, OMNITAB A Computer Program for Statistical and Numerical Analysis, NBS Handbook 101, March 1966. See pp. 10-12 for a bibliography of general-purpose computer programs.
- A. E. Beam and J. Hilsenrath, "PRECISE: A Multiple Precision Version of OMNITAB," NBS Technical Note 446, June 1968.
- H. C. Honeck, J. E. Suich, J. C. Jensen, C. E. Bailey, and J. W. Stewart, "JOSHUA - A Reactor Physics Computational System," Proc. Conf. Effective Use of Computers in the Nuclear Industry, CONF-690401, (1969) pp. 324-336.

13. J. E. Suich, J. C. Jensen, and H. C. Honeck, "Data Management for Multiaccess Computational Systems," *Ibid.*, pp. 471-481.
14. H. J. Kopp and D. S. Selengut, "DATATran - A Data Handling Computer Language for a Large Modular Reactor Design System," Proc. Intern. Conf. Utilization of Research Reactors and Reactor Mathematics and Computation, CNM-R-2, (1967) pp. 1460-1472.
15. C. N. Kelber, G. Jensen, L. Just, and B. J. Toppel, "The Argonne Reactor Computation System ARC," *Ibid.*, pp. 1428-1445.
16. E. E. Reilly, Jr., and W. H. Turner, "The Automation of Reactor Design Calculations at the Knolls Atomic Power Laboratory," Proc. Conf. Application of Computing Methods to Reactor Problems, ANL-7050, (1965) pp. 251-263.
17. J. Birnbaum, "Design, Usage, and Redesign of a Nuclear Physics Data Acquisition System," Proc. Skytop Conf. Computer Systems in Experimental Nuclear Physics, CONF-690301, (1969) pp. 91-103.
18. P. D. Summers, "A Precompilation Scheme for the Creation of Special Purpose Laboratory Languages," *Ibid.*, pp. 507-514.
19. S. Cohen, "The Delphi-Speakeasy System. I. Overall Description," *Computer Phys. Commun.* 2, 1-10 (1971).
20. C. L. Miller, "Man-Machine Communication in Civil Engineering," MIT Department of Civil Engineering report T63-3 (1963).
21. D. G. Bobrow and J. Weizenbaum, "List Processing and Extension of Language Facility by Embedding," *IEEE Trans. on Electronic Computers* EC-13, pp. 395-400 (1964).
22. J. Cocke and J. T. Schwartz, "Programming Languages and Their Compilers," Preliminary Notes (2nd revised version, April 1970), Courant Institute of Mathematical Sciences, New York University, New York, pp. 6-13.
23. K. D. Ferrell and M. V. Harlow, "The J-16 Data Reduction Program 'DATA'," EG&G Report AL-504 (1970).
24. H. A. Grench and M. V. Harlow, Jr., "SIGM7, a FORTRAN IV Program for the Resonance Analysis of Neutron Capture Data," to be published.
25. UPDATE is a CDC-6600 computer systems program that provides for creating, correcting, and manipulating files contained in a program library. As in other symbolic programs, UPDATE data may be source cards for a compiler or assembler run, data cards, or any other symbolic information desired by the user. Further information about UPDATE is available from the CONTROL DATA 6600 COMPUTER SYSTEMS, SCOPE 3 Reference Manual (1968 with revisions).