NASA TECHNICAL NOTE

NASA TN D-5350

c.1

LOAN COPY: RETURN TO
AFWL (WLIL-2)
KIRTLAND AFB, N MEX

TECH LIBRARY KAFB, NM
0132312

NASA TN D-5350

IMPLEMENTATION AND STRUCTURE OF COMPUTE, A TIME-SHARING CALCULATOR PROGRAM

by R. Bruce Canright, Jr., and Paul Swigert

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JULY 1969

TECH LIBRARY KAFB, NM
0132312

IMPLEMENTATION AND STRUCTURE OF COMPUTE, A TIME-SHARING CALCULATOR PROGRAM

By R. Bruce Canright, Jr., and Paul Swigert

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

ABSTRACT

COMPUTE is a computer program, written primarily in FORTRAN IV and operating under the IBM 360/67 Time-Sharing System. COMPUTE allows users to perform various numerical calculations solely by interacting with COMPUTE. This report gives details on the computer system, and outlines the structure of the computer program. This report is intended to serve as a guide for adapting and implementing COMPUTE. Complete source listings are included.

IMPLEMENTATION AND STRUCTURE OF COMPUTE, A TIME-SHARING CALCULATOR PROGRAM

by R. Bruce Canright, Jr., and Paul Swigert

Lewis Research Center

SUMMARY

COMPUTE is a computer program, written primarily in FORTRAN IV and operating under the IBM 360/67 Time-Sharing System. This report gives details on the computer system and outlines the structure of the computer program. It is intended to serve as a guide for adapting and implementing COMPUTE. Access to the system, program creation and execution, the logical structure of COMPUTE, and the functions of the subroutines and COMMON blocks are discussed. Complete source listings are included.

INTRODUCTION

A calculator program, COMPUTE, has been developed and is running under a time-sharing computer system, the IBM 360/67 Time-Sharing System (TSS). The capabilities and use of this program have been described previously (ref. 1). This report describes the computer system and COMPUTE in enough detail to guide implementation on other computers. This system encompasses a powerful command language and a virtual storage concept. Details on creating and executing COMPUTE under TSS are given in the first section.

The program consists of 40 routines, 37 in FORTRAN. The structure of these routines is outlined in the second section so that COMPUTE may easily be modified.

COMPUTER SYSTEM

COMPUTE was developed on the IBM 360 Model 67 under the Time-Sharing System (TSS). The purpose of this section is to describe TSS only as it affects the structure and execution of COMPUTE. This description will also show the areas to consider when

implementing COMPUTE under other computer systems. The TSS concepts mentioned herein are explained in more detail in references 2 to 4.

Access to TSS

Computer users gain access to TSS through card readers for batch tasks and through on-line devices for interactive, conversational tasks. COMPUTE is meant to be run conversationally. Access to the program is by on-line devices such as typewriter, teletype, and cathode ray tube (CRT) display terminals. Users are recognized by the system through unique names or codes. This process is called LOGON.

Computer Program Creation

Users who are LOGged ON become interactive with TSS and have available to them all data sets (ref. 2) either created by them or shared with them by other users. These data sets can include, among other things, source programs and special data sets called job libraries. Job libraries contain the object programs produced by language processors, for example, by a FORTRAN compiler. To create an object program, that is, one suitable for loading and execution, a user (1) defines a job library to contain the output object program, (2) creates the input source program, and (3) feeds it to the appropriate language processor, which stores the object program in the job library.

Computer Program Execution

To execute a program, an active user (1) defines the job library (or libraries) containing the program and all other programs it calls (via the DATA DEFINITION Command), (2) loads the program into his active storage (via the LOAD Command), and (3) runs the program (via the RUN Command). As we will see, COMPUTE programs are contained in the system library (SYSLIB) and in a private library called COMPTLIB.

When programs require input/output streams, the user may define them or default and let the system define them. For running COMPUTE and for other conversational tasks, these default input/output streams are from or to the on-line user's terminal. With one exception, COMPUTE is meant to interact with user solely via this terminal. The exception is in input/output of the data set KEYWORDS, to be discussed later.

TSS Computer Words

The computer words under TSS and expected by COMPUTE are short precision words (32 bits). This length permits four characters or approximately six significant figures for floating point numbers. The full EBCDIC character set (appendix H of ref. 4) is assumed by COMPUTE.

Machine-Dependent Routines Used in COMPUTE

Although written mainly in FORTRAN IV (ref. 5), COMPUTE uses extensively two shift routines contained in the IBM/360 shift package. ISRL (N, IWORD) shifts the full integer word, IWORD, N bits to the right. ISLL (N, IWORD) shifts the full integer word, IWORD, N bits to the left. These shifts are used extensively for character manipulation. COMPUTE reads all input in A1 format (ref. 5) and then translates into A4 format, short precision numbers, etc. COMPUTE only requires shifts of 8 bits to the left and 24 bits to the right; therefore, on the IBM 360/67 the FORTRAN coding can be substituted for the assembler language shift routines. The IBM/360 TSS shift functions are listed in the appendix.

FORTRAN Shifts for COMPUTE are as follows:

```
C       SPECIAL FORTRAN SHIFTS FOR COMPUTE
        FUNCTION ISLL (NN, IWRD)
        INTEGER MASK/Z80 00 00 00/
        IWORD = IWRD
        DO 1 I=1, NN
    1   IWORD=IWORD *2
        GO TO 3
        ENTRY ISRL (NN, IWRD)
        INTEGER MASKR/Z00 00 00 80/
        IWORD=IWRD
C       SET SIGN BIT TO ZERO
        IF (IWORD. LT. 0) IWORD+MASK
        DO 2 I=1, NN
    2   IWORD=IWORD/2
C       RESTORE BIT IF NEEDED
        IF (IWRD. LT. 0) IWORD=IWORD+MASKR
    3   ISLL=IWORD
        RETURN
        END
```

COMPUTE uses a special routine developed at Lewis for loading user object programs dynamically while COMPUTE is executing. By dynamic loading, we mean herein that user input to COMPUTE can cause user object programs to be loaded into the user's active storage or executed after loading. To do this the user inputs a program entry name, or a keyword previously stored in COMPUTE, which corresponds to a program entry name (see section Discussion of KEYWORDS). It is expected that this feature will be difficult to implement under systems other than TSS.

COMPUTE uses three ENTRY points in this routine. LOADED(NAME, KODE) returns KODE=1 if entry NAME is already loaded into the user's active storage, KODE=2 if not. LOAD (NAME, KODE) returns KODE=1 if it is able to load NAME, KODE=2 if not. RUNIT(NAME, NARG, ARG, ANS, KODE) executes user routine NAME, which has NARG arguments in array ARG, and returns one numerical result (if any) in ANS, KODE=1 if everything is successful, and KODE=2 if there are any error conditions. This routine is listed for completeness in the appendix. To run COMPUTE without the loading feature, dummy ENTRY points must be provided.

One routine in COMPUTE, SUBROUTINE EXPON, tests a system switch, OVERFL(J) where J=1 indicates overflow, J=3 indicates underflow, and J=2 means neither condition encountered. TSS will not indicate either condition without this test. However, error returns from FORTRAN - supplied routines, for example, SIN(X) are indicated (and, in fact, presently stop execution of COMPUTE). An ENTRY OVERFL(J) is required if such a switch is not available in a system.

Finally, the main routine for COMPUTE itself can be written in assembler language (ref. 6). This has the advantage that the main program can define (via ENTRY LIBDEF, see listings) and close (via ENTRY LIBREL) the two data sets required to run COMPUTE, KEYWORDS, and COMPTLIB. KEYWORDS is a set of tables, and COMPTLIB is the job library containing the COMPUTE routines; these are owned by one user and may be obtained by other users via the SHARE command. The listing of this routine is given in the appendix. This routine is in the system library and available to all users.

When the main program is in FORTRAN, the user must use DATA DEFINITION commands to describe KEYWORDS and COMPTLIB to the system. These commands simply define KEYWORDS to be a FORTRAN input/output unit, and COMPTLIB to be a library of object programs.

STRUCTURE OF COMPUTE

COMPUTE can be thought of as a set of FORTRAN routines, an interpretive user language, or both. The language has been described previously (ref. 1). The purpose of this section is to describe the working of the FORTRAN routines briefly (flow charts

of each deck will not be given) but in enough detail to show the way for changes or additions.

Coded Strings

Users can define functions and procedures within COMPUTE (see ref. 1). These are stored in arrays, in coded strings. These strings are the heart of COMPUTE processing. The method of storing functions and procedures will now be sketched.

The string for a user function begins with two locations for the function name (therefore, $\leq$ eight characters), then a pointer to the next string, then a counter of the arguments, then the argument names (two locations per name), and finally the arithmetic expression of the function (after interpretation). This structure is presented in table I. These strings are contained in the array IUSFCT (1000).

TABLE I. - STRING FOR USER FUNCTION [f(x) = x*x]

Location	Contains	Symbolic name	Example
i	Function name (1)	-------	bbbb
i+1	Function name (2)	-------	bbbF
i+2	Pointer to beginning of next name	NPOINT	----
i+3	Number of arguments for this function	NARGS	1
i+4	Argument name (1)	-------	bbbb
i+5	Argument name (2)	-------	bbbx
:			
:			
:	Argument name (1)	-------	----
i+3+2*NARGS	Argument name (2)	-------	----
i+4+2*NARGS	Arithmetic expression for function;	-------	x *
:	Coded, may include other functions, constants,	-------	x
:	arguments, etc.		
:			
NPOINT	Next function name (1)	-------	----
	Next function name (2)	-------	----

The string for a user procedure begins with two locations for the procedure name, then a pointer to the next name, then a counter of the lines in this procedure, and then substrings for each line. These substrings contain a length pointer, a code to indicate their type, and the packed expression of the line itself. The types of substrings include:

(1) name = ? TYPE = 2

(2) name = expression TYPE = 1

(3) expression = ? TYPE = 3

(4) COMPUTE commands except PRINT TYPE = 5

(5) PRINT TYPE = 4

(6) END statement (no TYPE)

This structure is shown in table II for a three-line procedure. These strings are contained in the array IUSPCD (2000).

TABLE II. - STRING FOR USER PROCEDURE

Location	Contains	Symbolic name	Example
i	Procedure name (1)	-------	-------
i+1	Procedure name (2)	-------	-------
i+2	Pointer to beginning of next name	NPOINT	-------
i+3	Number of lines for this procedure	NLINES	3
i+4	Length of first line	NLEN1	6
i+5	Code for first line	TYPE	1
i+6	Arithmetic expression for this line, depending on code	-------	x=x+1
:			
:			
i+6+NLEN1	Length of this line	NLEN2	2
i+7+NLEN1	Code for this line	TYPE	4
i+8+NLEN1	Arithmetic expression for this line	-------	PRINT(X)
i+8+NLEN1+ NLEN2	Length of expression on left hand side of < or >	-------	1
:	Arithmetic expression for this last line (END statement)	-------	X < 14
NPOINT	Procedure name (1)	-------	-------
	Procedure name (2)	-------	-------

Functions of the Subroutines

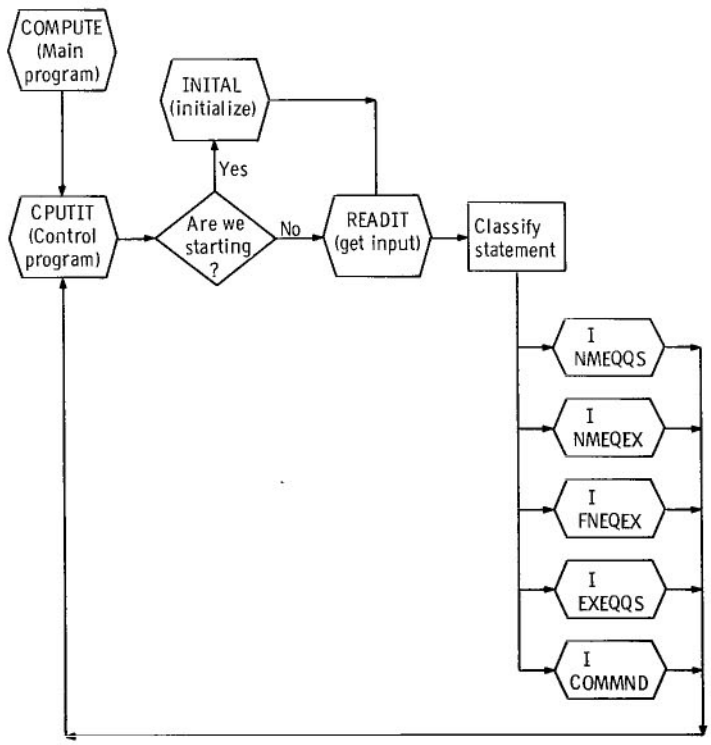
There are 40 subroutines within COMPUTE. We will not attempt to analyze the logic of each; it is hoped the listings in the appendix will serve that purpose, where necessary. Instead, we will explain briefly what each subroutine does, and then which routines call which. The routines are described in alphabetic order for each in reference. Entry points are listed with the subroutines they are contained in.

BEGNIT	begins new procedures, stores new name
CNVRT	converts input numbers in A1 format to floating point numbers for use in arithmetic
COMMND	analyzes input for valid COMPUTE commands. If one is found, calls appropriate routine, for example, DOIT
COMPUTE	main program; defines data sets if in assembler language, otherwise dummy
CPUTIT	control program; decides what to call after first scan of input (by READIT)
CREATE	small main program for initializing KEYWORDS; required only once, listed for completeness
DOIT	processes commands to do user procedures
DUMPIT	dumps out requested operands, for example, values
ENDIT	in creating procedures, processes end statement
ERASIT	erases requested operands, frees storage they used
EVAL, EVALI	processes string of expressions produced by PRESS1, PRESS2, by calling EXPR for each expression
EXEQQS	processes form expression = ?, either as command or line of procedure
EXPON, ADD, DVD, SUB, MULT	performs arithmetic for one operator
EXPR	processes one expression by finding operators and calling arithmetic routine

FNDNAM	searches lists and classifies user's names
FNEQEX	defines user function
FOFX	evaluates user function for integration routine, SIMPS1
INITAL	initializes all arrays, does bookkeeping on KEYWORDS
INTGRL	processes INTEGRATE command
INTIT	processes INT function, integration function in COMPUTE
ISRL, ISLL	shift routines in assembler language (discussed in first section)
LISTIT	dummy routine presently intended to serve similarly to DUMPIT, not developed
LOAD, LOADED, RUNIT	execution time loading routine in assembler language (discussed in first section)
MODEIT	checks and sets mode following MODE command
MSG	issues all output messages
NMBR	processes expressions assumed to be numbers, calls CNVRT
NMEQEX	processes form, name=expression, either as command or line of procedure
NMEQNU	stores user values and value names
NMEQQS	processes form, name=?, either as command or line of procedure
OVERFL	system switch for overflow and underflow conditions (discussed in first section)
PGMEVL	loads user programs and prepares arguments
PGMFCT	calls system functions whenever references to them appear
PGMNAM, PGMLST	outputs information on system functions

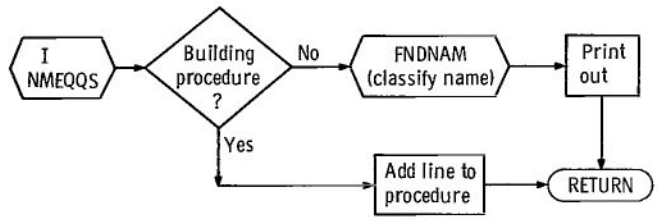
PRESS1, PRESS2	packs and factors user input strings into chains of expressions
PRNTIT, PRNT	responds to PRINT commands. PRNT produces no output, is called following PRINT commands in procedures
READIT, READT1	reads lines of user input, performs first scan on punctuation, operators, etc.
RESLV1, RESLV2	resolves names referenced by user
SIMPS1	performs numerical integration of functions. A powerful routine developed at Lewis
USRFCT	performs evaluation of user functions
USRPGM	causes execution of user object program

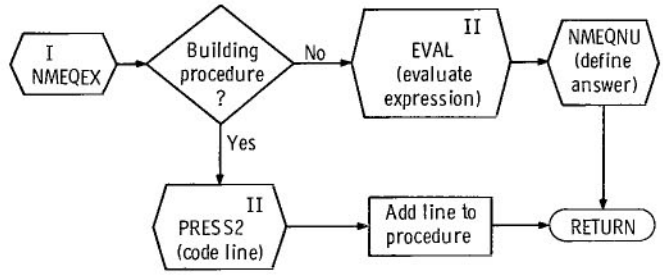
Flow of control among these routines is illustrated in figure 1. Roman numerals in program blocks indicate that the flow is continued elsewhere in the figure.

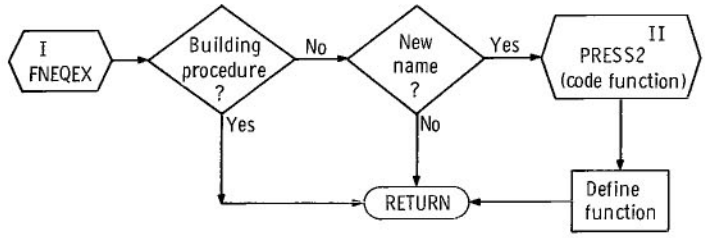


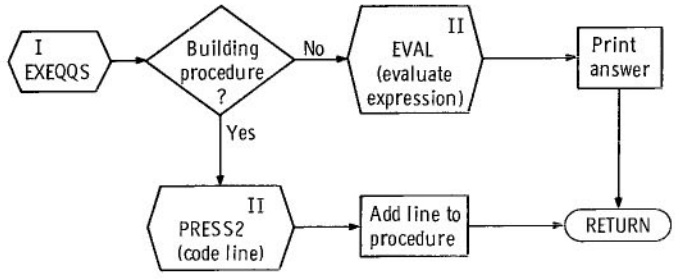
(a) Main flow of control.

Figure 1. - Flow map of COMPUTE program.



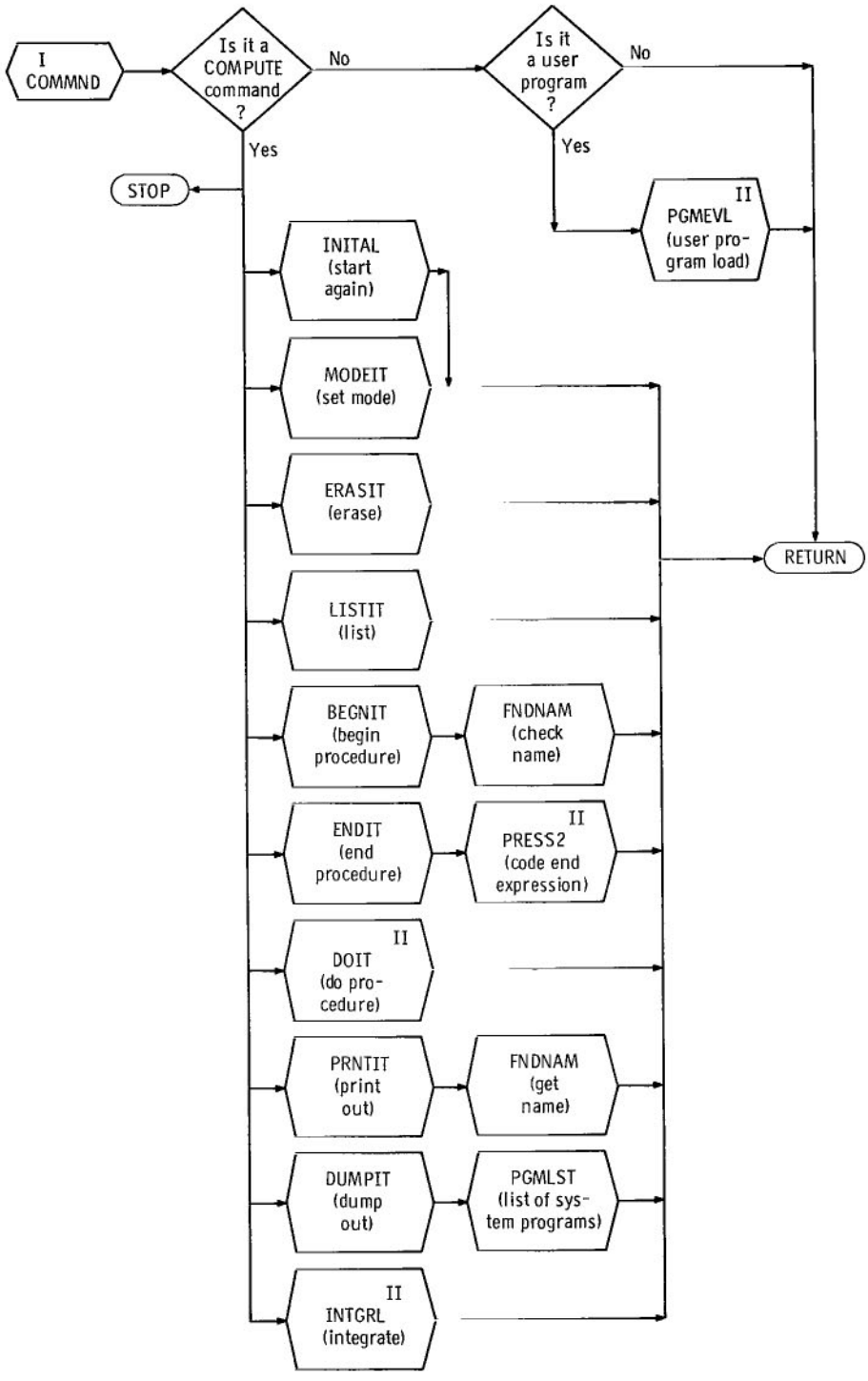






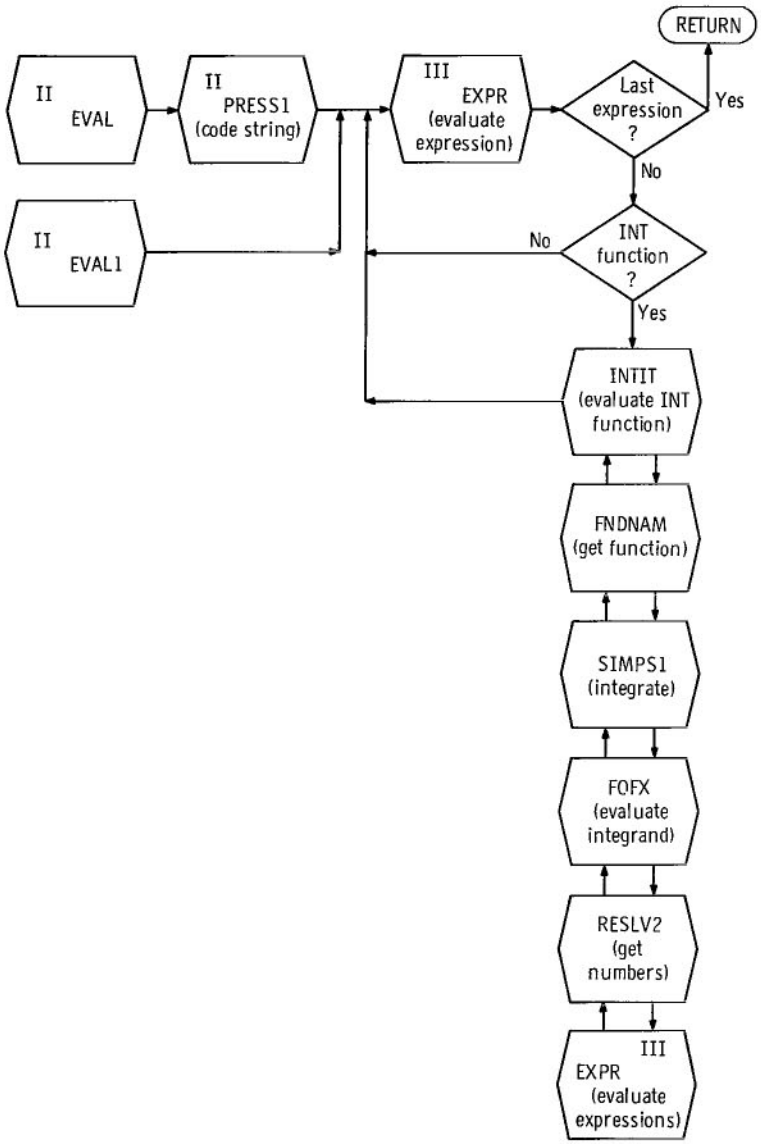
(b) Processing five types of COMPUTE statements.

Figure 1. - Continued.



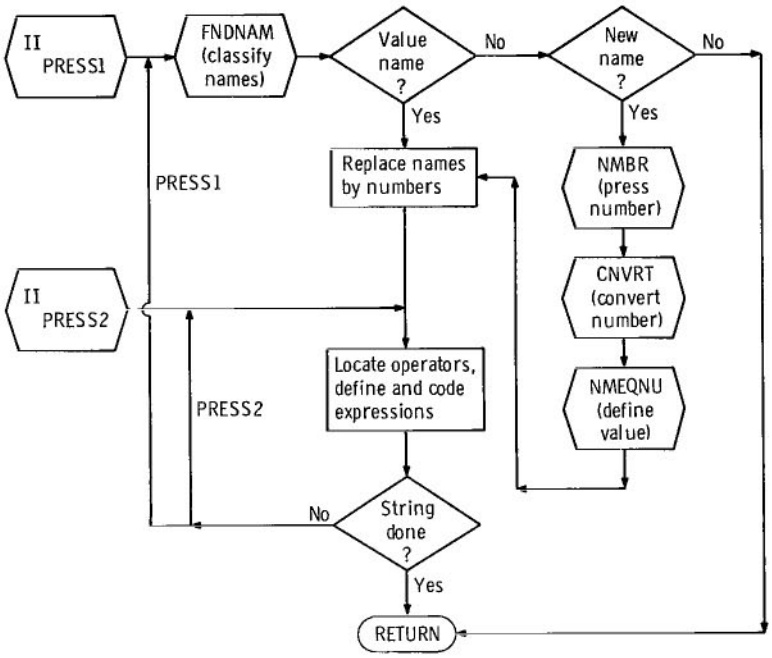
(b) Concluded.

Figure 1. - Continued.

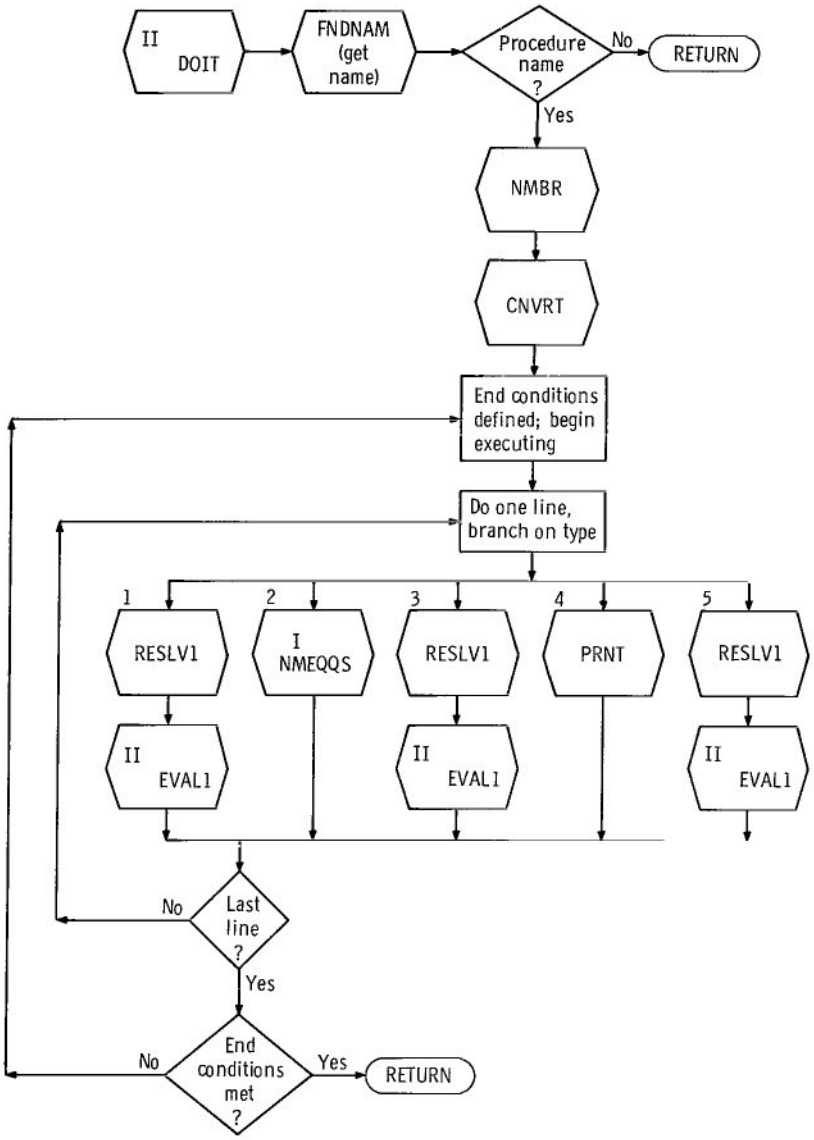


(c) Processing coded strings.

Figure 1. - Continued.

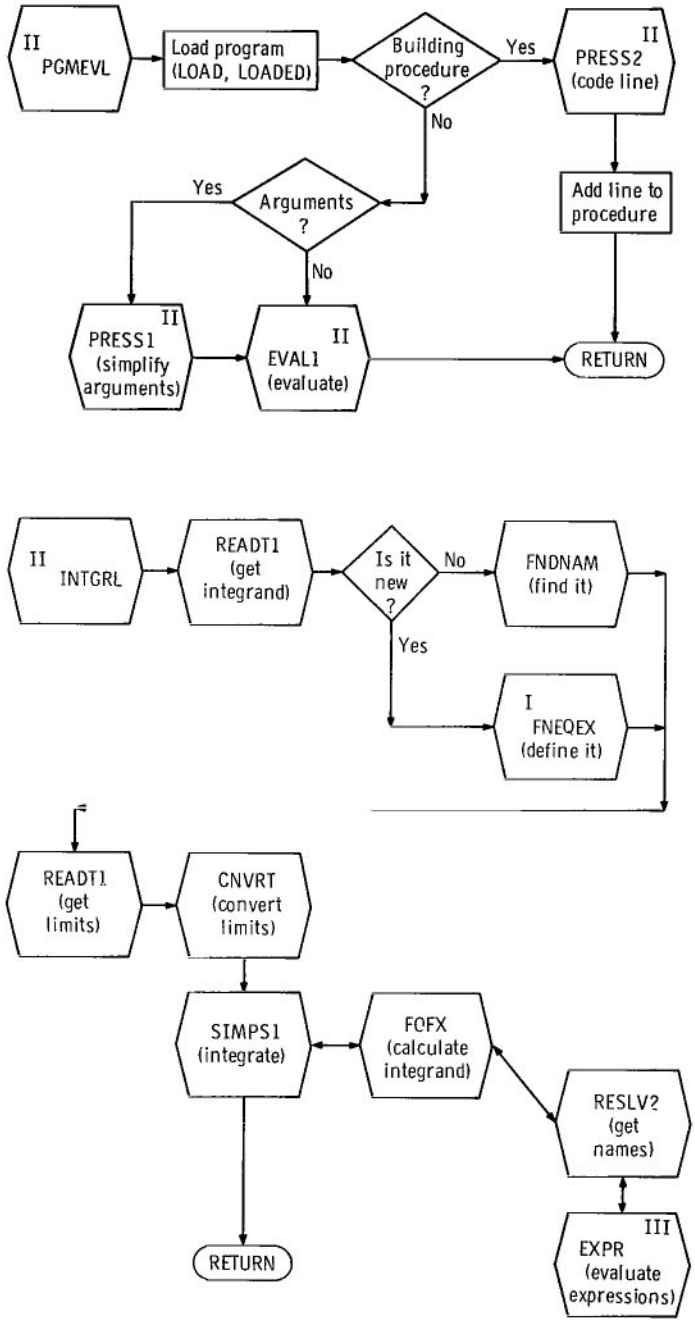


(c) Continued.
Figure 1. - Continued.



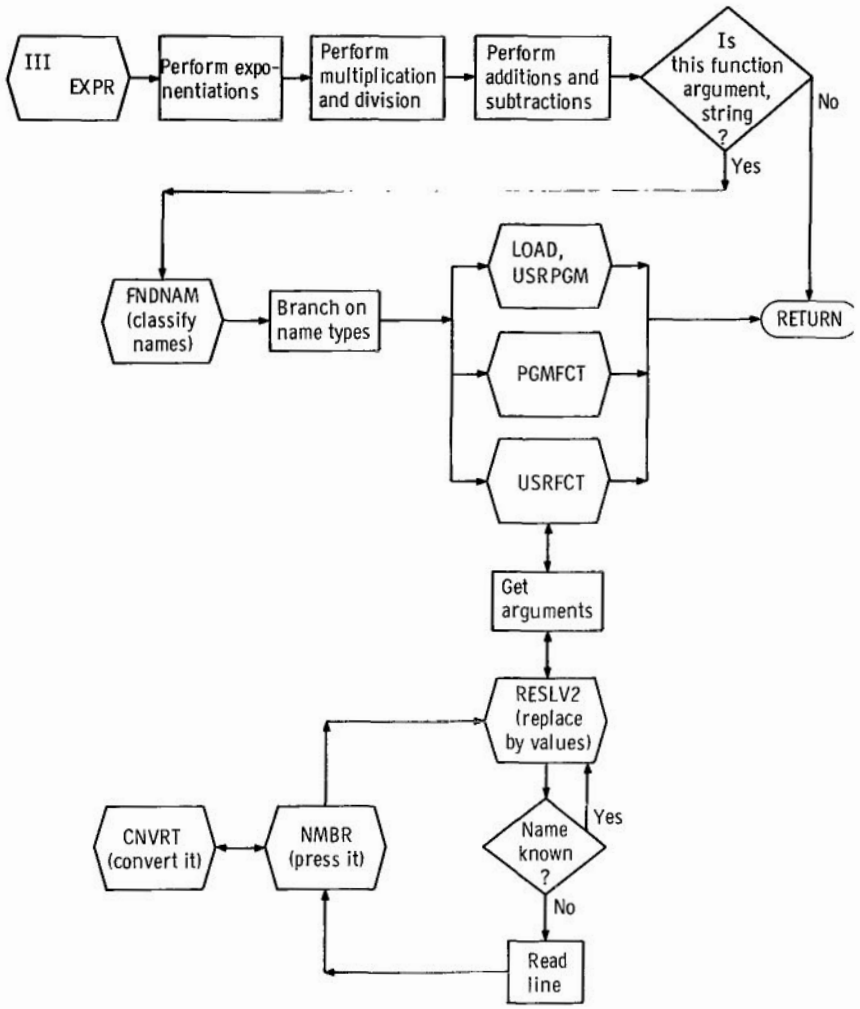
(c) Continued.

Figure 1. - Continued.



(c) Continued.

Figure 1. - Continued.



(c) Concluded.

Figure 1. - Concluded.

Functions of COMMON Blocks

There are six blocks of COMMON storage used in COMPUTE. The Common blocks are further described as follows and in table III.

COMMON/NAMES/	contains user values and value names
COMMON/FCTS/	contains all user functions
COMMON/PROCDS/	contains all user procedures
COMMON/MODE1/	contains information on the current mode

COMMON/ACMDS/ contains table, created by use of KEYWORDS, which allows user programs to be called by key names, other than program or entry names

COMMON/NANCY/ communicates information to routines which do integration

TABLE III. - COMMON BLOCK STRUCTURE

COMMON block	Appears in	Variable	Contents of variable
/NAMES/	INITAL, DUMPIT, PRNTIT, ERASIT, NMEQQS, NMEQNU, PRESS1, RESLV1, FNDNAM	NMLT	Number of user defined value names
		NAME(100)	All user defined value names
		VALUE(50)	The corresponding values
/FCTS/	INITAL, DUMPIT, ERASIT, NMEQQS, FNEQEX, USRFCT, FNDNAM, INTIT, INTGRL, FOFX	NFCT	Number of user defined functions
		LSTI	Pointer to end of IUSFCT list
		IUSFCT(1000)	List of all user defined function names and coded strings
/PROCDS/	INITAL, READIT, PGMEVL, DUMPIT, PRNTIT, ERASIT, BEGNIT, ENDIT, DOIT, NMEQQS, EXEQQS, NMEQEX, FNDNAM	NPCD	Number of user procedures
		LASTI	Pointer to end of IUSPCD list
		ICNT	Pointer to line of current procedure; number of lines
		IUSPCD(2000)	List of all user procedure names and their coded strings
/MODE1/	INITAL, COMPUTE, CPUTIT, READIT, PGMEVL, DUMPIT PRNTIT, ERASIT, MODEIT, BEGNIT, ENDIT, DOIT, NMEQQS, NMEQNU, EXEQQS, NMEQEX, FNEQEX FNDNAM, INTGRL	DEBUG	LOGICAL, T means in DEBUG mode, F means no DEBUG
		PROCED	LOGICAL, T means procedure being built, F means not
/ACMDS/	INITAL, COMMND	NAXCMD	Number of current keyword calls to user programs possible, ≤ 33
		AUXCMD (3, 33)	List of keywords current
		PGM(2, 33)	List of corresponding program or entry names current
/NANCY/	INTGRL, INTIT, FOFX	IND	Pointer to name of function in question
		NARG	Number of arguments for function in question
		NA	Pointer to beginning of functions coded string
		NB	Pointer to end of functions coded string

Explanation of KEYWORDS

A special data set, KEYWORDS, is maintained within COMPUTE by SUBROUTINE INITAL solely to extend the dynamic loading feature discussed in the section COMPUTER SYSTEM. KEYWORDS contain tables which map user-chosen names (of up to 12 characters) onto ENTRY point names (up to 6 characters). These tables allow users to load and execute programs with the dynamic loading routine by names other than ENTRY names.

An application of this is as follows: A user owns some routines of general application, which he shares with other users. He wants to make them accessible by mnemonic names, and yet be able to modify ENTRY points. He could do this by building a table and a keyword pointing to it in KEYWORDS. It should be emphasized that KEYWORDS has nothing to do with program loading and execution and that it simply allows greater freedom in naming.

For example, the COMPUTE main program (see listing) defines a job library (MATLIB) containing a set of conversational matrix routines (ref. 7). Suppose these routines are accessed by the ENTRY name MATAR. The owner of MATLIB could place in KEYWORDS the keyword MATRIX, which could invoke the table:

Keyword	Entry executed
MATRIX	MATAR
INVERSE	MATAR
DETERMINANT	MATAR
EIGENVALUES	MATAR

Then the matrix program could be executed in COMPUTE as in the following session.

```
User     :  starts execution of COMPUTE
COMPUTE:  ENTER USE KEYWORD
User     :  MATRIX
COMPUTE:  (load MATAR)
COMPUTE:  READY
User     :  EIGENVALUES
COMPUTE:  Starts execution of MATAR
```

Of course, user programs can be obtained by COMPUTE by their ENTRY names also, as previously discussed.

To build and maintain tables in KEYWORDS, a user enters the SYSKEY (b. LAMPOON. bb in listing of INITAL) when COMPUTE outputs ENTER USE KEYWORD.

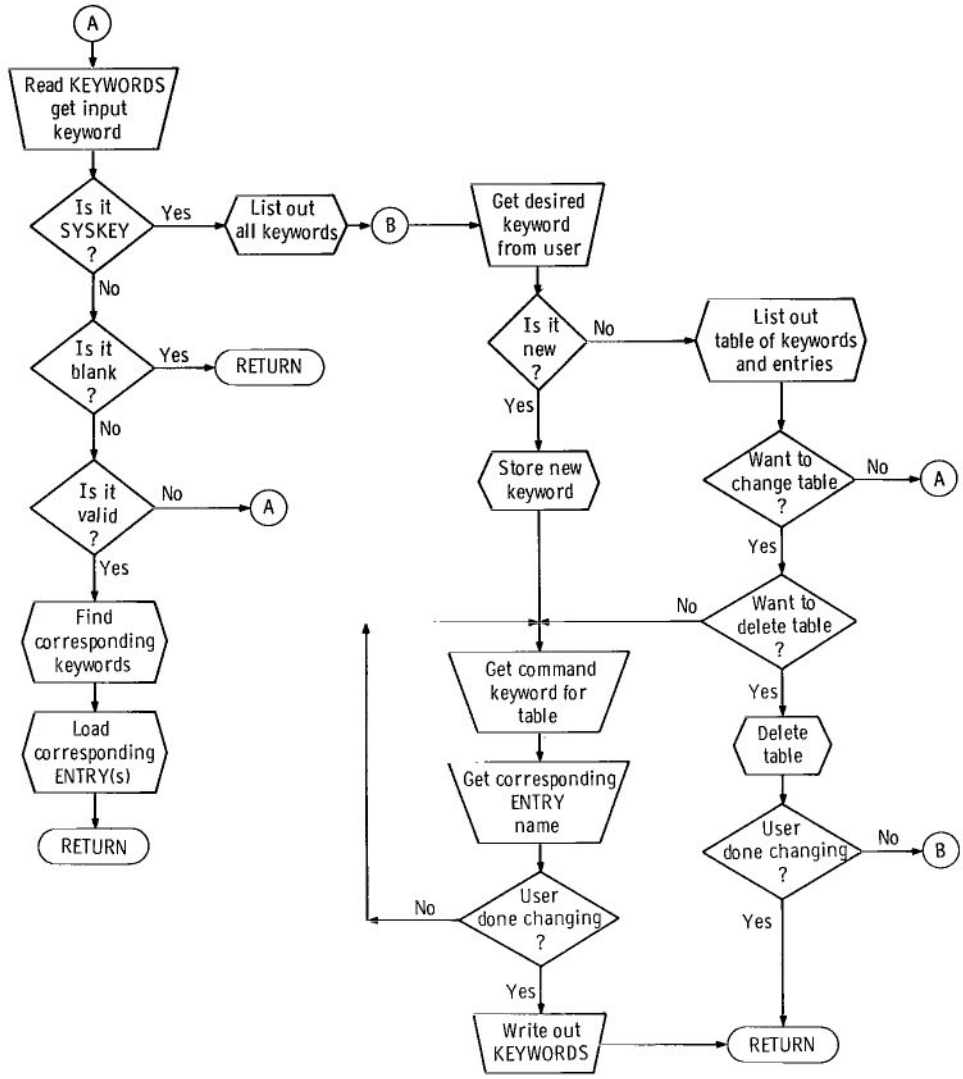


Figure 2. - Flow map of INITAL.

It is intended that the SYSKEY be known by few users. The logic of INITAL is shown in figure 2.

CONCLUDING REMARKS

A time-sharing calculator program, COMPUTE, has been developed under a particular computer system (IBM 360-67 Time-Sharing System). Many users at Lewis (not computer professionals) have successfully applied COMPUTE in their work. Because this

program can have widespread applications, details about the system and the program, written primarily in FORTRAN IV, were presented herein. These details should guide the implementation of COMPUTE at other computer installations and with other computer systems.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, April 23, 1969,
129-04-06-03-22.

APPENDIX - SOURCE LISTINGS FOR COMPUTE

Complete listings of the COMPUTE routines are included herein. Those interested in obtaining source decks should contact COSMIC, The University of Georgia, Athens, Georgia.

COMPUTE, LIBDEF, LIBREL

```
*         MAIN PROGRAM FOR COMPUTE
COMPUT#P  PSECT
          ENTRY     COMPUTE              ENTRY NAME
          ENTRY     LIBDEF
          ENTRY     LIBREL
SAVE      DC        F'76'                SAVE AREA
          DC        18F'0'
SAVE2     DC        F'76'
          DC        18F'0'
SWITCH    DC        F'0'       FIRST TIME SWITCH
ADCOMPT   DC        A(COMPUTE)
KEYS      DCB       DDNAME=FT05F001,RECFM=F,LRECL=84
MODE1     CCM
DEBUG     DC        F'0'
PRCCED    CC        F'0'
COMPUT#C  CSECT     READONLY,PUBLIC      START OF CSECT
          USING     COMPUTE,15
COMPUTE   SAVE      (14,12)
          L         14,72(0,13)          GET PSECT COVER REG
          ST        14,8(0,13)
          ST        13,4(0,14)           STORE BACKWARD LINK
          LR        13,14                SET REG 13 TO ADDRESS OF PSECT
          USING     COMPUT#P,13
          LR        12,15
          DROP      15
          USING     COMPUTE,12
          CALL      CHCBD1               SET INTERRUPTS LIKE FORTRAN
          L         7,SWITCH             FIRST
          C         7,=F'0'              TIME?
          BNE       CALLIT               NO
          L         7,=F'1'              YES
          ST        7,SWITCH             SWITCH=1
          DDEF      'DDCOMPT,VP,DSNAME=COMPTLIB,OPTION=JOBLIB'
          DDEF      'DDMAT,VP,DSNAME=MATLIB,OPTION=JOBLIB'
CALLIT    CALL      CPUTIT,,,E
          CALL      CHCIW1               FORTRAN RETURN TO SYSTEM
          L         13,4(0,13)
          RETURN    (14,12)
          USING     LIBDEF,15
LIBDEF    SAVE      (14,12)
          L         14,72(0,13)          GET PSECT COVER REG
          USING     COMPUT#P,14
          LA        12,SAVE2
          ST        12,8(0,13)
          ST        13,4(0,12)
          LR        13,12
          L         12,ADCOMPT
          DROP      15
          USING     COMPUTE,12
*                                        START DATA DEFS FOR INITIAL
          DDEF      'FT05F001,VS,DSNAME=KEYWORDS'
*                                        END DATA DEFS FOR INITIAL
          L         13,4(0,13)
          RETURN    (14,12)
```

```
         USING     LIBREL,15
LIBREL   SAVE      (14,12)
         L         14,72(0,13)          GET PSECT COVER REG
         LA        12,SAVE2
         ST        12,8(0,13)
         ST        13,4(0,12)
         LR        13,12
         USING     COMPUT#P,14
         L         12,ADCOMPT
         DROP      15
         USING     COMPUTE,12
*                                       START DATA RELS FOR INITIAL
         REL       'FT05F001'
*                                       END DATA RELS FOR INITIAL
         L         13,4(0,13)
         RETURN    (14,12)
         END
.
.
.
.                                                              COMPUTE
C        FORTRAN MAIN PROGRAM, DECK NAME COMPUTE
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      PROCED = .FALSE.
      CALL CPUTIT
      RETURN
      END
.
.
.
.
C                                                              CPUTIT
      SUBROUTINE CPUTIT
      DIMENSION INPUT(441),NM(3)
      COMMON/MODE1/DEBUG,PROCED
      LOGICAL NUM,ISTART,DEBUG,PROCED
      DATA ISTART/.TRUE./
      PROCED = .FALSE.
      IF(ISTART)  CALL INITAL
      ISTART=.FALSE.
    2 CALL READIT(INPUT,NM,K,NUM,ILP,IRP,IEQ,ICM,IOP,ILST)
      IF(ILST.EQ.0)  GO TO 20
   10 IF(ILP.EQ.0.AND.IRP.NE.0)  GO TO 11
      IF(IRP.GE.ILP)  GO TO 12
   11 CALL MSG(4,NM(1),NM(2),NM(3))
      GO TO 2
   12 IF(IEQ.NE.0)  GO TO 14
      IF(NUM.OR.ICM.NE.0.OR.(IOP.NE.0.AND.IOP.LT.ILP))  GO TO 18
      IF(K.LE.12)  GO TO 121
      CALL MSG(7,NM(1),NM(2),NM(3))
      GO TO 2
  121 CALL COMMND(NM,INPUT,ILP,ILST,&13)
      GO TO 2
```

```
   13 IF(.NOT.PROCED)  RETURN
      CALL MSG(8,NM(1),NM(2),NM(3))
      GO TO 2
   14 IF(IQM.NE.0)  GO TO 16
      IF(K.GT.8)  GO TO 19
      IF(NUM)  GO TO 18
      IF(ICP.NE.0.AND.IOP.LT.IEQ)  GO TO 18
      IF(ILP.EQ.0.AND.IRP.EQ.0)  GO TO 141
      IF(ILP.LT.IEQ.AND.IRP.LT.IEQ)  GO TO 15
  141 CALL NMEQEX(NM(2),INPUT,IEQ+1,ILST)
      GO TO 2
   15 CALL FNEQEX(NM(2),INPUT,ILP,IRP,IEQ+1,ILST)
      GO TO 2
   16 IF(ILST.GT.IQM)  CALL MSG(6,NM(1),NM(2),NM(3))
      IF(NUM)  GO TO 17
      IF(IRP.NE.0.AND.IRP.LT.IEQ)  GO TO 17
      IF(IOP.NE.0.AND.ICP.LT.IEQ)  GO TO 17
      IF(K.GT.8)  GO TO 19
      CALL NMEQQS(NM(2))
      GO TO 2
   17 CALL EXEQQS(INPUT,IEQ-1)
      GO TO 2
   18 CALL MSG(2,NM(1),NM(2),NM(3))
      GO TO 2
   19 CALL MSG(3,NM(1),NM(2),NM(3))
      GO TO 2
   20 CALL MSG(111,A,B,C)
      GO TO 2
      END
.
.
.
.                                                    READIT, READIT1
C
      SUBROUTINE READIT(INPUT,NM,K,NUM,ILP,IRP,IEQ,IQM,IOP,ILST)
      DIMENSION INPUT(1),NM(1)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER CAT,BLK,EQ,QM,AS,SH,PL,TEMP,TEMP1,RP,DP,AP
      LOGICAL NUM,APS
      DATA RP,DP,AP/')','.',1H'/
      DATA CAT,BLK,EQ,QM,LP,AS,SH,PL,MI,MASK /'|',' ','=','?','(','*',
     *'/','+','-',ZF0404040/
      KALL = 1
      GO TO 2
      ENTRY READT1(INPUT,NM,K,NUM,ILP,IRP,IEQ,IQM,IOP,ILST)
      KALL = 2
    2 K = 0
      NM(1)=BLK
      NM(2)=BLK
      NM(3)=BLK
      NUM = .FALSE.
      APS = .FALSE.
```

```
      ILP = 0
      IRP = 0
      IEQ = 0
      IQM = 0
      IOP = 0
      ILST = 0
      KOM = 0
      DO 1 I=1,441
    1 INPUT(I) = BLK
      IF(KALL.EQ.2)  GO TO 21
      IF(.NOT.PROCED)  CALL MSG(1,A,B,C)
      IF(PROCED)  CALL MSG(86,IUSPCD(ICNT),A,B)
   21 DO 9 J=1,3
      N1= 120*(J-1)+1
      N2 = 120*J
      READ 101,(INPUT(I),I=N1,N2)
      DO 8 I=N1,N2
      TEMP=INPUT(I)
      IF(TEMP.NE.CAT)  GO TO 4
      INPUT(I) = BLK
      GO TO 9
    4 IF(TEMP.EQ.BLK)  GO TO 8
      IF(TEMP.NE.AP)  GO TO 41
      INPUT(I) = BLK
      APS = .NOT.APS
      KOM = 1
      GO TO 8
   41 IF(.NOT.APS)  GO TO 42
      INPUT(I) = BLK
      GO TO 8
   42 IF(ILST.NE.0)  GO TO 3
   11 TEMP1 = ISRL(24,TEMP-MASK)
      IF(TEMP.EQ.DP.OR.(TEMP1.GE.0.AND.TEMP1.LE.9))  NUM = .TRUE.
    3 ILST = I
      IF(ILP.EQ.0.AND.TEMP.EQ.LP)  ILP = I
      IF(IRP.EQ.0.AND.TEMP.EQ.RP)  IRP = I
      IF(IEQ.EQ.0.AND.TEMP.EQ.EQ)  IEQ = I
      IF(IQM.EQ.0.AND.TEMP.EQ.QM)  IQM = I
      IF(IOP.EQ.0.AND.(TEMP.EQ.AS.OR.TEMP.EQ.SH.OR.TEMP.EQ.PL.OR.TEMP
     *.EQ.MI))  IOP = I
      IF(ILP.NE.0.OR.IEQ.NE.0)  GO TO 8
      K = K+1
      IF(K.GT.12)  GO TO 8
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,NM(3))
      NM(3) = ISLL(8,NM(3))+ISRL(24,TEMP)
    8 CONTINUE
      GO TO 10
    9 CONTINUE
      CALL MSG(9,NM(1),NM(2),NM(3))
      GO TO 2
   10 IF(ILST.EQ.C.AND.KOM.EQ.1)  GO TO 2
      RETURN
  101 FORMAT (120A1)
      END
```

```
C
      SUBROUTINE INITAL
      IMPLICIT INTEGER(A-Z)
      COMMON /NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      COMMON /ACMDS/NAXCMD,AUXCMD(3,33),PGM(2,33)
      DIMENSION INKEY(3),SYSKEY(3),INPUT(120),COMMND(3),PROG(2),
     *KEYWRD(3,20),NCMNDS(20),CMNDS(3,33,20),PGMS(2,33,20)
      DATA SYSKEY,YES,NO,BLK/' .LA','MPOD','N.  ','YES','NO',' '/
      LOGICAL DEBUG,PROCED,HERE,MOD
      DATA HERE/.FALSE./
      IF(PROCED)  GO TO 101
      DEBUG =.FALSE.
      NMLT=0
      NFCT = 0
      LSTI = 0
      NPCD = 0
      LASTI = 0
      MOD = .FALSE.
      CALL LIBDEF
      IF(HERE)  GO TO 1
      HERE = .TRUE.
      REWIND 5
      READ (5,201,END=1) NKEYS,KEYWRD,NCMNDS,CMNDS,PGMS
  201 FORMAT (I4,3(/,20A4),/,20I4,165(/,20A4))
    1 IF(MOD)  GO TO 37
      CALL MSG(139,A,B,C)
      NCMND = 0
      READ 202,INKEY
  202 FORMAT (3A4)
      IF(INKEY(1).EQ.SYSKEY(1).AND.INKEY(2).EQ.SYSKEY(2).AND.INKEY(3)
     *.EQ.SYSKEY(3))  GO TO 7
      IF(INKEY(1).NE.BLK.OR.INKEY(2).NE.BLK.OR.INKEY(3).NE.BLK)  GO TO 2
      NAXCMD = 0
      GO TO 99
    2 IF(NKEYS.EQ.0)  GO TO 4
      DO 3 I=1,NKEYS
      II = I
      IF(INKEY(1).EQ.KEYWRD(1,I).AND.INKEY(2).EQ.KEYWRD(2,I).AND.
     *INKEY(3).EQ.KEYWRD(3,I))  GO TO 5
    3 CONTINUE
    4 CALL MSG(140,A,B,C)
      GO TO 1
    5 NAXCMD = NCMNDS(II)
      DO 6 J=1,NAXCMD
      AUXCMD(1,J) = CMNDS(1,J,II)
      AUXCMD(2,J) = CMNDS(2,J,II)
```

```
      AUXCMD(3,J) = CMNDS(3,J,II)
      PGM(1,J) = PGMS(1,J,II)
      PGM(2,J) = PGMS(2,J,II)
      CALL LOAD(PGM(1,J),KODE)
      GO TO (6,51),KODE
   51 CALL MSG(134,PGM(1,J),PGM(2,J),A)
    6 CONTINUE
      GO TO 99
    7 IF(NKEYS.EQ.0)  GO TO 9
      CALL MSG(141,A,B,C)
      PRINT 203,(I,KEYWRD(1,I),KEYWRD(2,I),KEYWRD(3,I),I=1,NKEYS)
  203 FORMAT (' ',I5,2H '3A4,1H')
      GO TO 10
    9 CALL MSG(142,A,B,C)
   10 CALL MSG(143,A,B,C)
      READ 202,INKEY
      IF(INKEY(1).EQ.SYSKEY(1).AND.INKEY(2).EQ.SYSKEY(2).AND.INKEY(3)
     *.EQ.SYSKEY(3))  GO TO 11
      IF(INKEY(1).EQ.BLK.AND.INKEY(2).EQ.BLK.AND.INKEY(3).EQ.BLK)  GO TO
     *11
      GO TO 12
   11 CALL MSG(140,A,B,C)
      GO TO 10
   12 IF(NKEYS.EQ.0)  GO TO 14
      DO 13 I=1,NKEYS
      II = I
      IF(INKEY(1).EQ.KEYWRD(1,I).AND.INKEY(2).EQ.KEYWRD(2,I).AND.
     *INKEY(3).EQ.KEYWRD(3,I))  GO TO 15
   13 CONTINUE
   14 CALL MSG(144,A,B,C)
      MOD = .TRUE.
      NKEYS = NKEYS+1
      KEYWRD(1,NKEYS) = INKEY(1)
      KEYWRD(2,NKEYS) = INKEY(2)
      KEYWRD(3,NKEYS) = INKEY(3)
      II = NKEYS
      GO TO 22
   15 NCMND = NCMNDS(II)
      IF(NCMND.EQ.0)  GO TO 16
      CALL MSG(145,INKEY(1),INKEY(2),INKEY(3))
      PRINT 204,(J,CMNDS(1,J,II),CMNDS(2,J,II),CMNDS(3,J,II),
     *PGMS(1,J,II),PGMS(2,J,II),J=1,NCMND)
  204 FORMAT (' ',I5,2H '3A4,1H',4X,'CALLS',4X,1H',2A4,1H')
      GO TO 161
   16 CALL MSG(146,INKEY(1),INKEY(2),INKEY(3))
  161 CALL MSG(151,INKEY(1),INKEY(2),INKEY(3))
      READ 202,TEST
      IF(TEST.EQ.YES.OR.TEST.EQ.NO)  GO TO 162
      CALL MSG(140,A,B,C)
      GO TO 161
  162 IF(TEST.EQ.NO)  GO TO 1
      MOD = .TRUE.
  163 CALL MSG(147,INKEY(1),INKEY(2),INKEY(3))
      READ 202,TEST
```

```
      IF(TEST.EQ.YES.OR.TEST.EQ.NO)  GO TO 17
      CALL MSG(140,A,B,C)
      GO TO 163
   17 IF(TEST.EQ.NO)  GO TO 22
      IF(II.EQ.NKEYS)  GO TO 19
      DO 18 I=II,NKEYS
      KEYWRD(1,I) = KEYWRD(1,I+1)
      KEYWRD(2,I) = KEYWRD(2,I+1)
      KEYWRD(3,I) = KEYWRD(3,I+1)
      NCMNDS(I) = NCMNDS(I+1)
      DO 18 J=1,33
      CMNDS(1,J,I) = CMNDS(1,J,I+1)
      CMNDS(2,J,I) = CMNDS(2,J,I+1)
      CMNDS(3,J,I) = CMNDS(3,J,I+1)
      PGMS(1,J,I) = PGMS(1,J,I+1)
   18 PGMS(2,J,I) = PGMS(2,J,I+1)
   19 NKEYS = NKEYS-1
   20 CALL MSG(148,A,B,C)
      READ 202,TEST
      IF(TEST.EQ.YES.OR.TEST.EQ.NO)  GO TO 21
      CALL MSG(140,A,B,C)
      GO TO 20
   21 IF(TEST.EQ.YES)  GO TO 37
      MOD = .TRUE.
      GO TO 10
   22 CALL MSG(149,A,B,C)
      READ 205,INPUT
  205 FORMAT (120A1)
      COMMND(1) = BLK
      COMMND(2) = BLK
      COMMND(3) = BLK
      K = 0
      DO 23 I=1,120
      IF(INPUT(I).EQ.BLK)  GO TO 23
      K = K+1
      IF(K.GT.12)  GO TO 24
      COMMND(1) = ISLL(8,COMMND(1))+ISRL(24,COMMND(2))
      COMMND(2) = ISLL(8,COMMND(2))+ISRL(24,COMMND(3))
      COMMND(3) = ISLL(8,COMMND(3))+ISRL(24,INPUT(I))
   23 CONTINUE
      IF(COMMND(3).NE.BLK)  GO TO 25
   24 CALL MSG(140,A,B,C)
      GO TO 22
   25 CALL MSG(150,A,B,C)
      READ 205,INPUT
      PROG(1) = BLK
      PROG(2) = BLK
      K = 0
      DO 26 I=1,120
      IF(INPUT(I).EQ.BLK) GO TO 26
      K = K+1
      IF(K.GT.8)  GO TO 27
      PROG(1) = ISLL(8,PROG(1))+ISRL(24,PROG(2))
      PROG(2) = ISLL(8,PROG(2))+ISRL(24,INPUT(I))
```

```
 26 CONTINUE
    GO TO 28
 27 CALL MSG(140,A,B,C)
    GO TO 25
 28 IF(NCMND.EQ.0)  GO TO 30
    DO 29 J=1,NCMND
    JJ = J
    IF(COMMND(1).EQ.CMNDS(1,J,II).AND.COMMND(2).EQ.CMNDS(2,J,II)
   *.AND.COMMND(3).EQ.CMNDS(3,J,II))  GO TO 31
 29 CONTINUE
 30 NCMND = NCMND+1
    NCMNDS(II) = NCMND
    JJ = NCMND
    CMNDS(1,JJ,II) = COMMND(1)
    CMNDS(2,JJ,II) = COMMND(2)
    CMNDS(3,JJ,II) = COMMND(3)
 31 IF(PROG(2).EQ.BLK)  GO TO 32
    PGMS(1,JJ,II) = PROG(1)
    PGMS(2,JJ,II) = PROG(2)
    GO TO 35
 32 IF(JJ.EQ.NCMND)  GO TO 34
    DO 33 J=JJ,NCMND
    CMNDS(1,J,II) = CMNDS(1,J+1,II)
    CMNDS(2,J,II) = CMNDS(2,J+1,II)
    CMNDS(3,J,II) = CMNDS(3,J+1,II)
    PGMS(1,J,II) = PGMS(1,J+1,II)
 33 PGMS(2,J,II) = PGMS(2,J+1,II)
 34 NCMND = NCMND-1
    NCMNDS(II) = NCMND
 35 CALL MSG(148,A,B,C)
    READ 202,TEST
    IF(TEST.EQ.YES.OR.TEST.EQ.NO)  GO TO 36
    CALL MSG(140,A,B,C)
    GO TO 35
 36 IF(TEST.EQ.YES)  GO TO 37
    GO TO 22
 37 MOD = .FALSE.
    REWIND 5
    WRITE (5,201) NKEYS,KEYWRD,NCMNDS,CMNDS,PGMS
    END FILE 5
    CALL MSG(152,A,B,C)
    GO TO 1
 99 CALL MSG(11,A,B,C)
    CALL LIBREL
100 RETURN
101 CALL MSG(12,A,B,C)
    GO TO 100
    END
```

```
C                                                                NMEQQS
      SUBROUTINE NMEQQS(NM)
      DIMENSION NM(1)
      COMMON/NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      IF(PROCED)  GO TO 7
      CALL FNDNAM(NM,I,KODE)
      GO TO(1,2,3,4,5,6),KODE
    1 CALL MSG(41,NM(1),NM(2),NM(3))
      GO TO 100
    2 I2=I/2+1
      CALL MSG(77,NM(1),NM(2),VALUE(I2))
      GO TO 100
    3 CALL MSG(42,NM(1),NM(2),IUSFCT(I+3)/2)
      GO TO 100
    4 CALL MSG(43,NM(1),NM(2),NM(3))
      GO TO 100
    5 CALL MSG(44,NM(1),NM(2),NM(3))
      GO TO 100
    6 CALL MSG(133,NM(1),NM(2),NM(3))
      GO TO 100
    7 IF(LASTI+4.GT.1996)  GO TO 8
      IUSPCD(ICNT) = IUSPCD(ICNT)+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = 2
      LASTI = LASTI+1
      IUSPCD(LASTI) = 2
      LASTI = LASTI+2
      IUSPCD(LASTI-1) = NM(1)
      IUSPCD(LASTI) = NM(2)
      GO TO 100
    8 PROCED = .FALSE.
      LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
  100 RETURN
      END
C                                                                EXEQQS
      SUBROUTINE EXEQQS(INPUT,IEXND)
      DIMENSION INPUT(1)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      IS = 1
      IF(PROCED)  GO TO 2
      CALL EVAL(INPUT,IS,IEXND,ANS,&1)
      CALL MSG(78,ANS,A,B)
      GO TO 100
    1 CALL MSG(45,A,B,C)
```

```
      GO TO 100
    2 CALL PRESS2(INPUT,IS,IEXND,&100)
      IF(LASTI+IEXND+2.GT.1996)  GO TO 4
      IUSPCD(ICNT) = IUSPCD(ICNT)+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = IEXND
      LASTI = LASTI+1
      IUSPCD(LASTI) = 3
      DO 3 I=1,IEXND
      LASTI = LASTI+1
    3 IUSPCD(LASTI) = INPUT(I)
      GO TO 100
    4 PROCED = .FALSE.
      LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
  100 RETURN
      END
.
.
.
.
C                                                                    NMEQEX
      SUBROUTINE NMEQEX(NM,INPUT,IEXST,IEXND)
      DIMENSION NM(1),INPUT(1)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      IF(PROCED)  GO TO 2
      CALL EVAL (INPUT,IEXST,IEXND,ANS,&1)
      CALL NMEQNU(NM,ANS)
      GO TO 100
    1 CALL MSG(46,NM(1),NM(2),NM(3))
      GO TO 100
    2 CALL PRESS2(INPUT,IEXST,IEXND,&100)
      IF(LASTI+IEXND+4.GT.1996)  GO TO 4
      IUSPCD(ICNT) = IUSPCD(ICNT)+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = IEXND+2
      LASTI = LASTI+1
      IUSPCD(LASTI) = 1
      LASTI = LASTI+2
      IUSPCD(LASTI-1) = NM(1)
      IUSPCD(LASTI) = NM(2)
      DO 3 I=1,IEXND
      LASTI = LASTI+1
    3 IUSPCD(LASTI) = INPUT(I)
      GO TO 100
    4 PROCED = .FALSE.
      LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
  100 RETURN
      END
```

```
C
      SUBROUTINE FNEQEX(NM,INPUT,ILP,IRP,IS,IE)
      DIMENSION NM(1),INPUT(1),NM1(2)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER BLK,TEMP,CM
      DATA BLK,CM/' ',','/
      IF(PROCED)  GO TO 8
      CALL FNDNAM(NM,I,KODE)
      GO TO(2,12,9,10,11,13),KODE
   12 CALL MSG(47,NM(1),NM(2),NM(3))
      GO TO 100
    9 CALL MSG(48,NM(1),NM(2),NM(3))
      GO TO 100
   10 CALL MSG(53,NM(1),NM(2),NM(3))
      GO TO 100
   11 CALL MSG(54,NM(1),NM(2),NM(3))
      GO TO 100
   13 CALL MSG(136,NM(1),NM(2),NM(3))
      GO TO 100
    2 NFCT = NFCT+1
      LSTI = LSTI+2
      IUSFCT(LSTI-1) = NM(1)
      IUSFCT(LSTI) = NM(2)
      LSTI = LSTI+1
      LSTS = LSTI
      N1 = ILP+1
      N3 = IRP-1
      IF(N1.GT.N3)  GO TO 6
      LSTI = LSTI+1
      LSTA = LSTI
      I = 0
    3 NM1(1) = BLK
      NM1(2) = BLK
      K = 0
      N2 = N1
      DO 32 J=N2,N3
      N1 = N1+1
      TEMP = INPUT(J)
      IF(TEMP.EQ.BLK)  GO TO 32
      IF(TEMP.EQ.CM)  GO TO 33
      K = K+1
      IF(K.GT.8)  GO TO 7
      NM1(1) = ISLL(8,NM1(1))+ISRL(24,NM1(2))
      NM1(2) = ISLL(8,NM1(2))+ISRL(24,TEMP)
   32 CONTINUE
   33 IF(K.EQ.0)  GO TO 6
      IF(I.EQ.20)  GO TO 14
      I = I+2
```

```
      LSTI = LSTI+2
      IUSFCT(LSTI-1) = NM1(1)
      IUSFCT(LSTI) = NM1(2)
      IF(N1.LE.N3)  GO TO 3
      IUSFCT(LSTA) = I
      CALL PRESS2(INPUT,IS,IE,&31)
      IF(LSTI+IE.LE.1000)  GO TO 4
      CALL MSG(49,NM(1),NM(2),NM(3))
   31 LSTI = LSTS-3
      NFCT = NFCT-1
      GO TO 100
    4 DO 5 I=1,IE
      LSTI = LSTI+1
      IUSFCT(LSTI) = INPUT(I)
    5 CONTINUE
      IUSFCT(LSTS) = LSTI+1
      GO TO 100
    6 CALL MSG(50,NM(1),NM(2),NM(3))
      GO TO 31
    7 CALL MSG(51,NM(1),NM(2),NM(3))
      GO TO 31
    8 CALL MSG(52,NM(1),NM(2),NM(3))
      GO TO 100
   14 CALL MSG(126,A,B,C)
      GO TO 31
  100 RETURN
      END
.
.
.
.
.
C                                                          NMEQNU
      SUBROUTINE NMEQNU(NM,ANS)
      DIMENSION NM(1)
      COMMON/NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      CALL FNDNAM(NM,I,KODE)
      GO TO(1,2,4,5,6,7),KODE
    2 I2 = I/2+1
      VALUE(I2)=ANS
      IF(DEBUG)  CALL MSG(77,NM(1),NM(2),VALUE(I2))
      GO TO 100
    1 NMLT=NMLT+2
      IF(NMLT.EQ.98)  CALL MSG(66,NM(1),NM(2),NM(3))
      IF(NMLT.GT.100) GO TO 3
      NAME(NMLT-1)=NM(1)
      NAME(NMLT)=NM(2)
      JJ = NMLT/2
      VALUE(JJ) = ANS
      IF(DEBUG)  CALL MSG(77,NM(1),NM(2),VALUE(JJ))
      GO TO 100
    3 NMLT= 100
      CALL MSG(67,NM(1),NM(2),NM(3))
```

```
      GO TO 100
    4 CALL MSG(79,ANS,NM(1),NM(2))
      GO TO 100
    5 CALL MSG(80,ANS,NM(1),NM(2))
      GO TO 100
    6 CALL MSG(81,ANS,NM(1),NM(2))
      GO TO 100
    7 CALL MSG(131,ANS,NM(1),NM(2))
  100 RETURN
      END
.
.
.
.
C                                                                    NMBR
      SUBROUTINE NMBR(INPUT,N1,N2,ANS,*)
      DIMENSION INPUT(1),NBR(40)
      INTEGER BLK,RP,TEMP,CHAR
      REAL*8 ARG2
      DATA BLK,LP,RP/' ','(',')'/
      J=1
      NBR(1)=LP
      DO 1 I=N1,N2
      TEMP=INPUT(I)
      IF(TEMP.EQ.BLK) GO TO 1
      J=J+1
      IF(J.GT.39) GO TO 2
      NBR(J)=INPUT(I)
    1 CONTINUE
      J=J+1
      NBR(J)=RP
      KTR = 1
      CALL CNVRT(NBR,KTR,40,1,ARG1,ARG2,CHAR,&3,&3,&4)
      ANS=ARG1
      RETURN
    2 CALL MSG(74,A,B,C)
      RETURN 1
    3 CALL MSG(75,A,B,C)
      RETURN 1
    4 CALL MSG(76,CHAR,A,B)
      RETURN 1
      END
.
.
.                                                                    CNVRT
.
C     SUBROUTINE TO CONVERT A TYPE INPUT TO FLOATING POINT NUMBERS
C               INPUT    ARRAY OF CHARACTERS.  ONE PER WORD LEFT ADJ.
C               KTR      INDEX OF FIRST WORD OF ARRAY TO BE CONSIDERED
C               MAX      DIMENSION OF INPUT
C               TYPE     =1 REAL*4, =2 REAL*8
C               ARG1     REAL*4 ANS
C               ARG2     REAL*8 ANS
C               CHAR     BAD CHARACTER IF FOUND
```

```
C           RETURN        NEXT CHARACTER ) RESULT GOOD
C           RETURN 1      NO , OR ) FOUND RESULT ZERO
C           RETURN 2      NEXT CHARACTER , RESULT GOOD
C           RETURN 3      BAD CHARACTER RESULT ZERO
      SUBROUTINE CNVRT(INPUT,KTR,MAX,TYPE,ARG1,ARG2,CHAR,*,*,*)
      INTEGER INPUT(1),TYPE,TEMP,TEMP1,CHAR
      REAL*4 ARG1
      REAL*8 ARG2
      INTEGER BLK,RP,PL,DP,E,D,CM
      DATA MASK1,BLK,LP,RP,PL,MI,DP,E,D,CM/ZF0404040,' ','(',')','+',
     *'-','.','E','D',',','/
      IEXP = 0
      IP = 1
      ISGN = 1
      ARG1 = 0.0
      ARG2 = 0.0D0
      SIGN = 1.0
      NUM = 1
      ISTART = KTR
      DO 20 I=ISTART,MAX
      KTR = I
      TEMP = INPUT(I)
      IF(TEMP.EQ.BLK)  GO TO 20
      GO TO (8,9,9,9,9,9),NUM
    8 IF(TEMP.NE.LP.AND.TEMP.NE.CM)  GO TO 23
      NUM = 2
      GO TO 20
    9 IF(TEMP.EQ.RP.OR.TEMP.EQ.CM)  GO TO 22
      GO TO (10,10,13,15,17,19),NUM
   10 NUM = 3
      IF(TEMP.NE.PL)  GO TO 11
      GO TO 20
   11 IF(TEMP.NE.MI)  GO TO 13
      SIGN = -1.0
      GO TO 20
   13 IF(TEMP.EQ.E.OR.TEMP.EQ.D)  GO TO 151
      IF(TEMP.NE.DP)  GO TO 14
      NUM = 4
      GO TO 20
   14 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IF(TYPE.EQ.1)  ARG1 = ARG1*10.0+FLOAT(TEMP1)
      IF(TYPE.EQ.2)  ARG2 = ARG2*10.0D0+DBLE(FLOAT(TEMP1))
      GO TO 20
   15 IF(TEMP.NE.E.AND.TEMP.NE.D)  GO TO 16
  151 NUM = 5
      GO TO 20
   16 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IF(TYPE.EQ.1)  ARG1 = ARG1+FLOAT(TEMP1)/10.0**IP
      IF(TYPE.EQ.2)  ARG2 = ARG2+DBLE(FLOAT(TEMP1))/10.0D0**IP
      IP = IP+1
      GO TO 20
   17 NUM = 6
```

```
      IF(TEMP.NE.PL)  GO TO 18
      GO TO 20
   18 IF(TEMP.NE.MI)  GO TO 19
      ISGN = -1
      GO TO 20
   19 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IEXP = IEXP*10+TEMP1
   20 CONTINUE
      ARG1 = 0.0
      ARG2 = 0.0D0
      RETURN 1
   22 IF(TYPE.EQ.1)  ARG1 = SIGN*ARG1*10.0**(ISGN*IEXP)
      IF(TYPE.EQ.2)  ARG2 = SIGN*ARG2*10.0D0**(ISGN*IEXP)
      IF(TEMP.EQ.CM)  RETURN 2
      RETURN
   23 ARG1 = 0.0
      ARG2 = 0.0D0
      CHAR = TEMP
      RETURN 3
      END
.
.
.
.
C                                                                    FNDNAM
      SUBROUTINE FNDNAM(NM,I,KODE)
C     NM          NAME TO SEARCH FOR
C     I           INDEX OF NAME
C      KODE = 1   NAME NOT FOUND
C             2   VALUE NAME FOUND
C             3   USER FUNCTION NAME FOUND
C             4   PROCEDURE NAME FOUND
C             5   SYSTEM FUNCTION NAME FOUND
C             6   USER PROGRAM NAME FOUND
      DIMENSION NM(1)
      COMMON /NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      IF(NMLT.EQ.0)  GO TO 3
      DO 1 J=1,NMLT,2
      I=J
      IF(NM(1).EQ.NAME(J).AND.NM(2).EQ.NAME(J+1)) GO TO 2
    1 CONTINUE
      GO TO 3
    2 I2= I/2+1
      IF(DEBUG)  CALL MSG(77,NM(1),NM(2),VALUE(I2))
      KODE = 2
      GO TO 100
    3 IF(NFCT.EQ.0)  GO TO 6
      I = 1
      DO 4 J=1,NFCT
```

```
      IF(NM(1).EQ.IUSFCT(I).AND.NM(2).EQ.IUSFCT(I+1))  GO TO 5
      I = IUSFCT(I+2)
    4 CONTINUE
      GO TO 6
    5 KODE = 3
      GO TO 100
    6 IF(NPCD.EQ.0)  GO TO 9
      I = 1
      DO 7 J=1,NPCD
      IF(NM(1).EQ.IUSPCD(I).AND.NM(2).EQ.IUSPCD(I+1))  GO TO 8
      I = IUSPCD(I+2)
    7 CONTINUE
      GO TO 9
    8 KODE = 4
      GO TO 100
    9 CALL PGMNAM(NM,I)
      IF(I.EQ.0)  GO TO 11
      KODE = 5
      GO TO 100
   11 CALL LOADED(NM,I)
      GO TO (12,13),I
   12 KODE = 6
      GO TO 100
   13 KODE = 1
  100 RETURN
  201 FORMAT (' ',2A4,'=',G14.6)
      END
.
.
.
.                                                          PRESS1, PRESS2
C
      SUBROUTINE PRESS1(INPUT,IS,IE,*)
      DIMENSION INPUT(1),NM(2),IN1(88),INTMP(441)
      LOGICAL NBR,RESLV,EF
      INTEGER X,TEMP,BLK,RP,SH,AS,AS2,PL,DP,E,D,CM
      COMMON/NAMES/NMLT,NAME(100),IVALUE(50)
      DATA MASK,BLK,LP,RP,SH,AS,PL,MI,AS2/ZF0404040,' ','(',')','/','*',
     1'+','-','**'/
      DATA INT/' INT'/
      DATA DP,E,D,CM/'.','E','D',','/
      NBR(X) = ISRL(24,X-MASK).GE.0.AND.ISRL(24,X-MASK).LE.9
      RESLV = .TRUE.
      GO TO 22
      ENTRY PRESS2(INPUT,IS,IE,*)
      RESLV = .FALSE.
   22 IPRN = 0
      K=0
      NUMST=0
      EF = .FALSE.
      J = 1
      INTMP(1) = LP
      NM(1)=BLK
      NM(2)=BLK
```

```
      IEND = IE+1
      INPUT(IEND) = BLK
      DO 8 I=IS,IEND
      IF(I.EQ.IEND)  GO TO 1
      IF(J.GT.438)  GO TO 15
      IF(INPUT(I).EQ.BLK)  GO TO 8
      TEMP=INPUT(I)
      IF(TEMP.EQ.LP)  IPRN = IPRN+1
      IF(TEMP.EQ.RP)  IPRN = IPRN-1
      IF(IPRN.LT.0)  GO TO 14
      IF(TEMP.NE.PL.AND.TEMP.NE.MI)  GO TO 17
      IF(INTMP(J).NE.LP.AND.INTMP(J).NE.CM.OR.K.NE.0)  GO TO 16
      J = J+1
      INTMP(J) = 0
      GO TO 7
   16 IF(NUMST.EQ.0.OR..NOT.EF)  GO TO 1
      EF = .FALSE.
      K = K+1
      GO TO 8
   17 IF(TEMP.EQ.LP.OR.TEMP.EQ.RP.OR.TEMP.EQ.SH.OR.TEMP.EQ.AS.OR.
     *TEMP.EQ.CM)  GO TO 1
      IF((NBR(TEMP).OR.TEMP.EQ.DP).AND.K.EQ.0)  NUMST = I
      K=K+1
      IF(NUMST.EQ.0)  GO TO 18
      IF(TEMP.EQ.E.OR.TEMP.EQ.D)  EF = .TRUE.
      GO TO 8
   18 IF(K.GT.8)  GO TO 9
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,TEMP)
      GO TO 8
    1 IF(NUMST.EQ.0)  GO TO 2
      NUMEND = NUMST+K-1
      CALL NMBR(INPUT,NUMST,NUMEND,IANSR,&10)
      IF(INTMP(J-2).EQ.BLK.AND.INTMP(J-1).EQ.INT.AND.INTMP(J).EQ.LP)
     * GO TO 32
      IF(INTMP(J).NE.LP.AND.INTMP(J).NE.AS.AND.INTMP(J).NE.AS2.AND.
     * INTMP(J).NE.PL.AND.INTMP(J).NE.MI.AND.INTMP(J).NE.SH.AND.INTMP(J)
     * .NE.CM)  GO TO 34
      J=J+1
      INTMP(J) = IANSR
      GO TO 7
    2 IF(NM(2).EQ.BLK)  GO TO 61
      IF(INTMP(J).NE.LP.AND.INTMP(J).NE.AS.AND.INTMP(J).NE.AS2.AND.
     * INTMP(J).NE.PL.AND.INTMP(J).NE.MI.AND.INTMP(J).NE.SH.AND.INTMP(J)
     * .NE.CM)  GO TO 28
      IF(TEMP.EQ.LP.OR..NOT.RESLV)  GO TO 6
      CALL FNDNAM(NM,L,KODE)
      IF(INTMP(J-2).EQ.BLK.AND.INTMP(J-1).EQ.INT.AND.INTMP(J).EQ.LP)
     * GO TO 3
      GO TO(31,26,23,24,25,27),KODE
   26 L2 = L/2+1
      J=J+1
      INTMP(J) = IVALUE(L2)
      GO TO 7
```

```
  3 IF(TEMP.NE.CM.AND.TEMP.NE.RP)  GO TO 33
    GO TO (41,41,41,41,29,30),KODE
 41 J = J+2
    INTMP(J-1) = NM(1)
    INTMP(J) = NM(2)
    GO TO 7
 31 CALL MSG(55,NM(1),NM(2),NM(3))
    READ 101,IN1
    DO 4 I1=1,88
    IF(IN1(I1).NE.BLK) GO TO 5
  4 CONTINUE
    GO TO 99
  5 CALL NMBR(IN1,1,88,IANSR,&11)
    J=J+1
    INTMP(J) = IANSR
    CALL NMEQNU (NM,IANSR)
    GO TO 7
 11 CALL MSG(57,NM(1),NM(2),NM(3))
    GO TO 31
  6 J=J+2
    INTMP(J-1) = NM(1)
    INTMP(J) = NM(2)
    GO TO 7
 61 IF(TEMP.NE.AS.OR.INTMP(J).NE.AS)  GO TO 62
    INTMP(J) = AS2
    GO TO 72
 62 IF(TEMP.NE.RP.AND.TEMP.NE.AS.AND.TEMP.NE.SH.AND.TEMP.NE.PL.AND.
   *TEMP.NE.MI.AND.TEMP.NE.CM)  GO TO 63
    IF(INTMP(J).NE.RP)  GO TO 19
    GO TO 64
 63 IF(TEMP.NE.LP)  GO TO 20
    IF(INTMP(J).NE.AS2.AND.INTMP(J).NE.AS.AND.INTMP(J).NE.SH.AND.
   *INTMP(J).NE.PL.AND.INTMP(J).NE.MI.AND.INTMP(J).NE.LP.AND.
   *INTMP(J).NE.CM)  GO TO 19
 64 IF(I.EQ.IEND.AND.TEMP.NE.RP.AND.TEMP.NE.LP)  GO TO 21
  7 IF(I.EQ.IEND)  GO TO 12
    J = J+1
    INTMP(J) = TEMP
 72 NM(1)=BLK
    NM(2)=BLK
    K=0
    NUMST=0
    EF = .FALSE.
  8 CONTINUE
    STOP 100
  9 CALL MSG(56,NM(1),NM(2),NM(3))
    GO TO 99
 10 CALL MSG(57,NM(1),NM(2),NM(3))
    GO TO 99
 12 IF(IPRN.NE.C)  GO TO 14
    J = J+1
    INTMP(J) = RP
    DO 13 I=1,J
 13 INPUT(I) = INTMP(I)
```

```
      IE = J
      GO TO 100
   14 CALL MSG(58,NM(1),NM(2),NM(3))
      GO TO 99
   15 CALL MSG(60,NM(1),NM(2),NM(3))
      GO TO 99
   19 CALL MSG(61,INTMP(J),TEMP,A)
      GO TO 99
   20 CALL MSG(62,TEMP,A,B)
      GO TO 99
   21 CALL MSG(63,TEMP,A,B)
      GO TO 99
   23 CALL MSG(59,NM(1),NM(2),NM(3))
      GO TO 99
   24 CALL MSG(64,NM(1),NM(2),NM(3))
      GO TO 99
   25 CALL MSG(65,NM(1),NM(2),NM(3))
      GO TO 99
   27 CALL MSG(132,NM(1),NM(2),NM(3))
      GO TO 99
   28 CALL MSG(155,INTMP(J),NM(1),NM(2))
      GO TO 99
   29 CALL MSG(156,NM(1),NM(2),NM(3))
      GO TO 99
   30 CALL MSG(157,NM(1),NM(2),NM(3))
      GO TO 99
   32 CALL MSG(158,IANSR,A,B)
      GO TO 99
   33 CALL MSG(159,A,B,C)
      GO TO 99
   34 CALL MSG(164,INTMP(J),IANSR,A)
   99 RETURN 1
  100 RETURN
  101 FORMAT (88A1)
      END
```

```
C
      SUBROUTINE COMMND(NM,INPUT,IS,IE,*)
      DIMENSION NM(1),INPUT(1)
      COMMON /ACMDS/NAXCMD,AUXCMD(3,33),PGM(2,33)
      INTEGER AUXCMD,PGM
      INTEGER CMDS(66)/
     *             '    ','    ','DUMP','    ','    ','   D',
     *             '    ','   P','RINT','    ','    ','   P',
     *             '    ','   E','RASE','    ','    ','   E',
     *             '    ',' RES','TART','    ','    ','   R',
     *             '    ','    ','MODE','    ','    ','   M',
     *             '    ','    ',' END','    ','    ',' END',
     *             '    ','   B','EGIN','    ','    ','   B',
     *             '    ','    ','STOP','    ','    ','   S',
     *             '    ','    ','  DO','    ','    ','  DO',
     *             '    ','    ','LIST','    ','    ','   L',
     *             '   I','NTEG','RATE','    ','    ','   I'/
      DO 1 I=1,66,3
      K = I/3+1
      IF(NM(1).EQ.CMDS(I).AND.NM(2).EQ.CMDS(I+1).AND.NM(3).EQ.
     *CMDS(I+2))  GO TO 2
    1 CONTINUE
      GO TO 14
    2 GO TO(3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13),K
    3 CALL DUMPIT(INPUT,IS,IE)
      GO TO 100
    4 CALL PRNTIT(INPUT,IS,IE)
      GO TO 100
    5 CALL ERASIT(INPUT,IS,IE)
      GO TO 100
    6 CALL INITAL
      GO TO 100
    7 CALL MODEIT(INPUT,IS,IE)
      GO TO 100
    8 CALL ENDIT(INPUT,IS,IE)
      GO TO 100
    9 CALL BEGNIT(INPUT,IS,IE)
      GO TO 100
   10 RETURN 1
   11 CALL DOIT(INPUT,IS,IE)
      GO TO 100
   12 CALL LISTIT(INPUT,IS,IE)
      GO TO 100
   13 CALL INTGRL
      GO TO 100
   14 IF(NAXCMD.EQ.0)  GO TO 17
      DO 15 I=1,NAXCMD
      II = I
      IF(NM(1).EQ.AUXCMD(1,I).AND.NM(2).EQ.AUXCMD(2,I).AND.NM(3).EQ.
     *AUXCMD(3,I))  GO TO 16
```

```
   15 CONTINUE
      GO TO 17
   16 CALL PGMEVL(INPUT,IS,IE,PGM(1,II),&100)
      GO TO 100
   17 CALL PGMEVL(INPUT,IS,IE,NM(2),&101)
  100 RETURN
  101 CALL MSG(10,NM(1),NM(2),NM(3))
      GO TO 100
      END
.
.
.
.                                                                    LISTIT
C
      SUBROUTINE LISTIT(INPUT,IS,IE)
      PRINT 201
      RETURN
  201 FORMAT (' LIST NOT WORKING AT THIS TIME.')
      END
.
.
.
.                                                                    DUMPIT
C
      SUBROUTINE DUMPIT(INPUT,IS,IE)
      DIMENSION INPUT(1),NM(3)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON/NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER BLK,TEMP,RP
      INTEGER CMP(24)/
     *           '    ',' VA','LUES','    ','    ','   V',
     *           'USRF','UNCT','IONS','    ','    ',' UF',
     *           'SYSF','UNCT','IONS','    ','    ',' SF',
     *           ' PR','OCED','URES','    ','    ','  P'/
      DATA BLK,RP/' ',')'/
      IF(PROCED)  GO TO 10
      IF(IS.EQ.0)  GO TO 19
      N1 = IS+1
      NM(1) = BLK
      NM(2) = BLK
      NM(3) = BLK
      K = 0
      DO 11 I=N1,IE
      TEMP = INPUT(I)
      IF(TEMP.EQ.BLK)  GO TO 11
      IF(TEMP.EQ.RP)  GO TO 12
      K = K+1
      IF(K.GT.12)  GO TO 20
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,NM(3))
      NM(3) = ISLL(8,NM(3))+ISRL(24,TEMP)
```

```
11 CONTINUE
   CALL MSG(82,A,B,C)
   GO TO 100
12 IF(K.EQ.0)  GO TO 19
   DO 13 I=1,24,3
   K = I/3+1
   IF(NM(1).EQ.DMP(I).AND.NM(2).EQ.DMP(I+1).AND.NM(3).EQ.DMP(I+2))
  *  GO TO 14
13 CONTINUE
   CALL MSG(83,NM(1),NM(2),NM(3))
   GO TO 100
14 GO TO(15,15,16,16,17,17,18,18),K
15 IF(NMLT.EQ.0)  GO TO 3
   CALL MSG(32,A,B,C)
   N1 = 1
   N2 = 8*(NMLT/8)
   IF(N2.EQ.0)  GO TO 22
   DO 21 I=N1,N2,8
   J = I/2+1
21 PRINT 202,NAME(I),NAME(I+1),VALUE(J),NAME(I+2),NAME(I+3),
  *VALUE(J+1),NAME(I+4),NAME(I+5),VALUE(J+2),NAME(I+6),NAME(I+7),
  *VALUE(J+3)
   IF(N2.EQ.NMLT)  GO TO 100
22 N1 = N2+1
   N3 = 6*((NMLT-N2)/6)+N2
   IF(N3.EQ.N2)  GO TO 24
   N2 = N3
   DO 23 I=N1,N2,6
   J = I/2+1
23 PRINT 202,NAME(I),NAME(I+1),VALUE(J),NAME(I+2),NAME(I+3),
  *VALUE(J+1),NAME(I+4),NAME(I+5),VALUE(J+2)
   IF(N2.EQ.NMLT)  GO TO 100
24 N1 = N2+1
   N3 = 4*((NMLT-N2)/4)+N2
   IF(N3.EQ.N2)  GO TO 2
   N2 = N3
   DO 1 I=N1,N2,4
   J= I/2+1
 1 PRINT 202,NAME(I),NAME(I+1),VALUE(J),NAME(I+2),NAME(I+3),
  * VALUE(J+1)
   IF(N2.EQ.NMLT)  GO TO 100
 2 J = NMLT/2
   PRINT 202,NAME(NMLT-1),NAME(NMLT),VALUE(J)
   GO TO 100
 3 CALL MSG(33,A,B,C)
   GO TO 100
16 IF(NFCT.EQ.0)  GO TO 6
   CALL MSG(35,A,B,C)
   J = 1
   DO 5 I=1,NFCT
   NARG = IUSFCT(J+3)/2
   PRINT 207,IUSFCT(J),IUSFCT(J+1),NARG
   J = IUSFCT(J+2)
 5 CONTINUE
```

```
      GO TO 100
    6 CALL MSG(34,A,B,C)
      GO TO 100
   17 CALL PGMLST
      GO TO 100
   18 IF(NPCD.EQ.0)  GO TO 9
      CALL MSG(37,A,B,C)
      J = 1
      DO 8 I=1,NPCD
      PRINT 207,IUSPCD(J),IUSPCD(J+1)
      J = IUSPCD(J+2)
    8 CONTINUE
      GO TO 100
    9 CALL MSG(39,A,B,C)
      GO TO 100
   10 CALL MSG(40,A,B,C)
      GO TO 100
   19 CALL MSG(84,A,B,C)
      GO TO 100
   20 CALL MSG(85,A,B,C)
  100 RETURN
  202 FORMAT (4(2X,2A4,'=',G14.6))
  207 FORMAT (' ',2A4,5X,I3,' ARGUMENTS.')
      END
```

PGMNAM, PGMLST

```
C        SUBROUTINE TO MATCH INPUT NAME WITH FUNCTION NAME
      SUBROUTINE PGMNAM(PGMI,IFN)
      INTEGER PGMI(1)
      DATA NPGM/38/
      INTEGER PGM(38)
      DATA PGM/
     * '    ',' EXP','    ',' LN','    ',' LOG',' AR','CSIN',
     * ' AR','CCOS',' AR','CTAN','    ',' SIN','    ',' COS',
     * '    ',' TAN','    ','SQRT','    ','TANH','    ','SINH',
     * '    ','COSH','    ',' ERF','    ','ERFC','   G','AMMA',
     * ' LNG','AMMA','    ',' ABS','    ',' INT'/
    6 DO 7 IFN=1,NPGM,2
      IFN1 = IFN
      IF(PGMI(1).EQ.PGM(IFN).AND.PGMI(2).EQ.PGM(IFN+1))  GO TO 8
    7 CONTINUE
      IFN = 0
      GO TO 100
    8 IFN = IFN1/2+1
      GO TO 100
      ENTRY PGMLST
      PRINT 201
      PRINT 202
      PRINT 203
  100 RETURN
  201 FORMAT (' LIST OF AVAIABLE FUNCTIONS.',/,
     *'      NAME        DEFINITION                  ARGUMENT RANGE')
```

```
  202 FORMAT (
     *'      EXP         EXPONENTIAL                X<174.673',/,
     *'      LN          NATURAL LOGARITHM          X>0',/,
     *'      LOG         COMMON LOGARITHM           X>0',/,
     *'      SIN         SINE                       |X|<(2**18)*PI',/,
     *'      COS         COSINE                     |X|<(2**18)*PI',/,
     *'      TAN         TANGENT                    |X|<(2**18)*PI',/,
     *'      ARCSIN      ARCSINE                    |X|<1',/,
     *'      ARCCOS      ARCCOSINE                  |X|<1',/,
     *'      ARCTAN      ARCTANGENT                 NO RESTRICTION')
  203 FORMAT (
     *'      SINH        HYPERBOLIC SINE            X<174.673',/,
     *'      COSH        HYPERBOLIC COSINE          X<174.673',/,
     *'      TANH        HYPERBOLIC TANGENT         NO RESTRICTION',/,
     *'      SQRT        SQUARE ROOT                X>=0',/,
     *'      ERF         ERROR FUNCTION             NO RESTRICTION',/,
     *'      ERFC        COMPLEMENTED               NO RESTRICTION',/,
     *'                  ERROR FUNCTION',/,
     *'      GAMMA       GAMMA FUNCTION             2**(-252)<X<57.574',/,
     *'      LNGAMMA     NATURAL LOGARITHM          0<X<4.2913E+73',/,
     *'                  OF GAMMA FUNCTION',/,
     *'      ABS         ABSOLUTE VALUE             NO RESTRICTION',/,
     *'      INT         INTEGRATION                NO RESTRICTION',/,
     *'                  3 ARGUMENTS',/,
     *'                  (USR FUNCTION,LIMIT,LIMIT)')
      END
.
.
.
.
C                                                          PRNTIT, PRNT
      SUBROUTINE PRNTIT(INPUT,IS,IE)
      DIMENSION INPUT(1),NM(2)
      COMMON/NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER BLK,CM,TEMP
      DATA BLK,CM/' ',','/
      N1 = IS+1
      N3 = IE-1
      IF(N3.LT.N1)  GO TO 101
      I = 0
    1 NM(1) = BLK
      NM(2) = BLK
      K = 0
      N2 = N1
      DO 2 J=N2,N3
      N1 = N1+1
      TEMP = INPUT(J)
      IF(TEMP.EQ.BLK)  GO TO 2
      IF(TEMP.EQ.CM)  GO TO 3
      K = K+1
      IF(K.GT.8)  GO TO 2
```

```
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,TEMP)
    2 CONTINUE
    3 IF(K.EQ.0)  GO TO 52
      IF(K.GT.8)  GO TO 51
      IF(I.EQ.18)  GO TO 53
      I = I+1
      INPUT(I) = NM(1)
      I = I+1
      INPUT(I) = NM(2)
      GO TO 4
   51 CALL MSG(24,NM(1),NM(2),A)
      GO TO 4
   52 CALL MSG(124,A,B,C)
    4 IF(N1.LE.N3)  GO TO 1
      GO TO 5
   53 CALL MSG(125,A,B,C)
    5 IF(I.EQ.0)  GO TO 101
      IF(PROCED)  GO TO 70
      N1 = 1
      N2 = I
      GO TO 6
      ENTRY PRNT(INPUT,IS,IE)
      N1 = IS
      N2 = IE
    6 I1 = N1
    7 IF(I1.GT.N2)  GO TO 100
      CALL FNDNAM(INPUT(I1),INDEX,KODE)
      GO TO (8,9,8,8,8,8),KODE
    8 I1 = I1+2
      GO TO 7
    9 J1 = INDEX/2+1
      I2 = I1+2
   10 IF(I2.GT.N2)  GO TO 17
      CALL FNDNAM(INPUT(I2),INDEX,KODE)
      GO TO (11,12,11,11,11,11),KODE
   11 I2 = I2+2
      GO TO 10
   12 J2 = INDEX/2+1
      I3 = I2+2
   13 IF(I3.GT.N2)  GO TO 16
      CALL FNDNAM(INPUT(I3),INDEX,KODE)
      GO TO (14,15,14,14,14,14),KODE
   14 I3 = I3+2
      GO TO 13
   15 J3 = INDEX/2+1
      I4 = I3+2
  151 IF(I4.GT.N2)  GO TO 154
      CALL FNDNAM(INPUT(I4),INDEX,KODE)
      GO TO (152,153,152,152,152,152),KODE
  152 I4 = I4+2
      GO TO 151
  153 J4 = INDEX/2+1
      PRINT 202,INPUT(I1),INPUT(I1+1),VALUE(J1),INPUT(I2),INPUT(I2+1),
```

```
     *VALUE(J2),INPUT(I3),INPUT(I3+1),VALUE(J3),INPUT(I4),INPUT(I4+1),
     *VALUE(J4)
      N1 = I4+2
      GO TO 6
  154 PRINT 202,INPUT(I1),INPUT(I1+1),VALUE(J1),INPUT(I2),INPUT(I2+1),
     *VALUE(J2),INPUT(I3),INPUT(I3+1),VALUE(J3)
      GO TO 100
   16 PRINT 202,INPUT(I1),INPUT(I1+1),VALUE(J1),INPUT(I2),INPUT(I2+1),
     *VALUE(J2)
      GO TO 100
   17 PRINT 202,INPUT(I1),INPUT(I1+1),VALUE(J1)
      GO TO 100
   70 IF(LASTI+I+2.GT.1996)  GO TO 72
      IUSPCD(ICNT) = IUSPCD(ICNT)+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = I
      LASTI = LASTI+1
      IUSPCD(LASTI) = 4
      DO 71 J=1,I
      LASTI = LASTI+1
   71 IUSPCD(LASTI) = INPUT(J)
      GO TO 100
   72 PROCED = .FALSE.
      LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
      GO TO 100
  101 CALL MSG(123,A,B,C)
  100 RETURN
  202 FORMAT (4(' ',2A4,'=',G14.6))
      END
.
.
.
.                                                                    BEGNIT
C
      SUBROUTINE BEGNIT(INPUT,IS,IE)
      DIMENSION INPUT(1),NM(2)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER BLK,TEMP
      DATA BLK/' '/
      IF(PROCED)  GO TO 101
      IF(IS.EQ.0)  GO TO 102
      N1 = IS+1
      N2 = IE-1
      NM(1) = BLK
      NM(2) = BLK
      K = 0
      DO 1 I=N1,N2
      TEMP = INPUT(I)
      IF(TEMP.EQ.BLK)  GO TO 1
      K = K+1
```

```
      IF(K.GT.8)  GO TO 103
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,TEMP)
    1 CONTINUE
      IF(K.EQ.0)  GO TO 102
      CALL FNDNAM(NM,I,KODE)
      GO TO (2,105,106,107,108,109),KODE
    2 PROCED = .TRUE.
      NPCD = NPCD+1
      LASTI = LASTI+4
      IUSPCD(LASTI-3) = NM(1)
      IUSPCD(LASTI-2) = NM(2)
      ICNT = LASTI
      IUSPCD(ICNT) = 1
      GO TO 100
  101 CALL MSG(87,A,B,C)
      GO TO 100
  102 CALL MSG(88,A,B,C)
      GO TO 100
  103 CALL MSG(24,NM(1),NM(2),A)
      GO TO 100
  105 CALL MSG(89,NM(1),NM(2),A)
      GO TO 100
  106 CALL MSG(90,NM(1),NM(2),A)
      GO TO 100
  107 CALL MSG(91,NM(1),NM(2),A)
      GO TO 100
  108 CALL MSG(92,NM(1),NM(2),A)
      GO TO 100
  109 CALL MSG(130,NM(1),NM(2),A)
  100 RETURN
      END
.
.
.
.
C                                                                    ENDIT
      SUBROUTINE ENDIT(INPUT,IS,IE)
      DIMENSION INPUT(1),INTMP(441)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER GT,BLK
      DATA GT,LT,BLK/'>','<',' '/
      IF(PROCED)  GO TO 1
      CALL MSG(93,A,B,C)
      RETURN
    1 PROCED = .FALSE.
      IF(IS.NE.0)  GO TO 3
    2 IUSPCD(ICNT-1) = LASTI+1
      RETURN
    3 N1 = IS+1
      N2 = IE-1
      K = 0
```

```
      IREL = 0
      LS = LASTI
      DO 4 I=N1,N2
      INTMP(I) = INPUT(I)
      IF(INPUT(I).EQ.BLK)  GO TO 4
      K = K+1
      IF(INPUT(I).NE.LT.AND.INPUT(I).NE.GT)  GO TO 4
      IREL = I
    4 CONTINUE
      IF(IREL.NE.0)  GO TO 7
      IF(K.NE.0)  GO TO 5
      CALL MSG(94,A,B,C)
      GO TO 2
    5 CALL MSG(95,A,B,C)
    6 PROCED = .TRUE.
      RETURN
    7 IST = IS+1
      IND = IREL-1
      IF(IST.LE.IND)  GO TO 71
      CALL MSG(96,A,B,C)
      GO TO 6
   71 CALL PRESS2(INPUT,IST,IND,&8)
      GO TO 9
    8 CALL MSG(97,A,B,C)
      GO TO 6
    9 IF(LASTI+1+IND.GT.1996)  GO TO 14
      LASTI = LASTI+1
      IUSPCD(LASTI) = IND
      DO 10 I=1,IND
      LASTI = LASTI+1
   10 IUSPCD(LASTI) = INPUT(I)
      LASTI = LASTI+1
      IUSPCD(LASTI) = INTMP(IREL)
      IST = IREL+1
      IND = IE-1
      IF(IST.LE.IND)  GO TO 101
      CALL MSG(98,A,B,C)
      LASTI = LS
      GO TO 6
  101 CALL PRESS2(INTMP,IST,IND,&11)
      GO TO 12
   11 LASTI = LS
      GO TO 8
   12 IF(LASTI+IND.GT.1996)  GO TO 14
      DO 13 I=1,IND
      LASTI = LASTI+1
   13 IUSPCD(LASTI) = INTMP(I)
      GO TO 2
   14 LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
      RETURN
      END
```

```
C
      SUBROUTINE ERASIT (INPUT,IOPTST,IOPTND)
      DIMENSION INPUT(1), NM(2)
      COMMON /NAMES/NMLT,NAME(100),VALUE(50)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER CM,BLK,RP,TEMP,SWT
      DATA CM,BLK,LP,RP/',',' ','(',')'/
      IF(PROCED)  GO TO 14
      IF(IOPTST.EQ.0)  GO TO 13
      IF(NMLT+NFCT+NPCD.EQ.0)  GO TO 12
      SWT=1
      K=0
      NM(1)=BLK
      NM(2)=BLK
      N1 = IOPTST+1
      DO 10 I= N1,IOPTND
      TEMP= INPUT(I)
      IF(TEMP.EQ.BLK)  GO TO 10
      IF(TEMP.EQ.CM.OR.TEMP.EQ.RP) GO TO (3,7),SWT
      K=K+1
      IF(K.EQ.9) SWT=2
      IF(K.GT.8) GO TO 10
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,TEMP)
      GO TO 10
    3 IF(NM(1).EQ.BLK.AND.NM(2).EQ.BLK) GO TO 8
      CALL FNDNAM(NM,J,KODE)
      GO TO(73,31,5,71,75,76),KODE
   31 J2 = J/2+1
      IF(J.EQ.NMLT)  GO TO 41
      DO 4 K=J,NMLT,2
      NAME(K) = NAME(K+2)
      NAME(K+1) = NAME(K+3)
      K2 = K/2+1
    4 VALUE(K2) = VALUE(K2+1)
   41 NMLT = NMLT-2
      GO TO 72
    5 J = J-1
      K = IUSFCT(J+3)-1
      IF(K.EQ.LSTI)  GO TO 61
      L = K-J
      LL = LSTI-L-J
      N = IUSFCT(K+3)+2
      IUSFCT(K+3) = IUSFCT(K+3)-L
      DO 6 M=1,LL
      J = J+1
      K = K+1
      IF(K.NE.N)  GO TO 6
```

```
      N = IUSFCT(K)+2
      IUSFCT(K) = IUSFCT(K)-L
    6 IUSFCT(J) = IUSFCT(K)
   61 LSTI = J
      NFCT = NFCT-1
      GO TO 72
   71 J = J-1
      K = IUSPCD(J+3)-1
      IF(K.EQ.LASTI)  GO TO 81
      L = K-J
      LL = LASTI-L-J
      N = IUSPCD(K+3)+2
      IUSPCD(K+3) = IUSPCD(K+3)-L
      DO 15 M=1,LL
      J = J+1
      K = K+1
      IF(K.NE.N)  GO TO 15
      N = IUSPCD(K)+2
      IUSPCD(K) = IUSPCD(K)-L
   15 IUSPCD(J) = IUSPCD(K)
   81 LASTI = J
      NPCD = NPCD-1
   72 IF(NMLT+NFCT+NPCD.EQ.0)  GO TO 11
      GO TO 74
   73 CALL MSG(23,NM(1),NM(2),NM(3))
      GO TO 74
    7 CALL MSG(24,NM(1),NM(2),NM(3))
      GO TO 74
    8 CALL MSG(25,NM(1),NM(2),NM(3))
      GO TO 74
   75 CALL MSG(31,NM(1),NM(2),NM(3))
      GO TO 74
   76 CALL MSG(135,NM(1),NM(2),NM(3))
   74 NM(1)=BLK
      NM(2)=BLK
      SWT=1
      K=0
    9 IF(TEMP.EQ.RP) RETURN
   10 CONTINUE
      CALL MSG(26,NM(1),NM(2),NM(3))
      GO TO 100
   11 CALL MSG(27,NM(1),NM(2),NM(3))
      GO TO 100
   12 CALL MSG(28,NM(1),NM(2),NM(3))
      GO TO 100
   13 CALL MSG(29,NM(1),NM(2),NM(3))
      GO TO 100
   14 CALL MSG(30,NM(1),NM(2),NM(3))
  100 RETURN
      END
```

```
C
      SUBROUTINE MODEIT(INPUT,IOPTST,IOPTND)
      DIMENSION INPUT(1),NM(3)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED
      INTEGER BLK,TEMP,RP
      DATA BLK,RP/' ',')'/
      INTEGER MOD(18)/
     *            '    ',' D','EBUG',' ',' ',' D',
     *            '    ',' N','OBUG',' ',' ',' N',
     *            '    ',' RE','AL*4',' ',' ',' R*4'/
      IF(PROCED) GO TO 8
      IF(IOPTST.EQ.0) GO TO 7
      N1 = IOPTST+1
      NM(1)=BLK
      NM(2)=BLK
      NM(3)=BLK
      K=0
      DO 1 I=N1,IOPTND
      TEMP= INPUT(I)
      IF(TEMP.EQ.BLK) GO TO 1
      IF(TEMP.EQ.RP) GO TO 2
      K= K+1
      IF(K.GT.12) GO TO 6
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,NM(3))
      NM(3) = ISLL(8,NM(3))+ISRL(24,TEMP)
    1 CONTINUE
      CALL MSG(13,NM(1),NM(2),NM(3))
      GO TO 100
    2 IF(K.EQ.0) GO TO 7
      DO 21 I=1,18,3
      K = I/3+1
      IF(NM(1).EQ.MOD(I).AND.NM(2).EQ.MOD(I+1).AND.NM(3).EQ.MOD(I+2))
     * GO TO 22
   21 CONTINUE
      CALL MSG(14,NM(1),NM(2),NM(3))
      GO TO 100
   22 GO TO(3,3,4,4,5,5),K
    3 IF(DEBUG) CALL MSG(15,NM(1),NM(2),NM(3))
      IF(.NOT.DEBUG) CALL MSG(16,NM(1),NM(2),NM(3))
      DEBUG= .TRUE.
      GO TO 100
    4 IF(DEBUG) CALL MSG(17,NM(1),NM(2),NM(3))
      IF(.NOT.DEBUG) CALL MSG(18,NM(1),NM(2),NM(3))
      DEBUG =.FALSE.
      GO TO 100
    5 CALL MSG(19,NM(1),NM(2),NM(3))
      GO TO 100
    6 CALL MSG(20,NM(1),NM(2),NM(3))
      GO TO 100
    7 CALL MSG(21,NM(1),NM(2),NM(3))
      GO TO 100
    8 CALL MSG(22,NM(1),NM(2),NM(3))
  100 RETURN
      END
```

```
C
      SUBROUTINE DOIT(INPUT,IS,IE)
      DIMENSION INPUT(1),NM(2)
      COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
      COMMON /MODE1/DEBUG,PROCED
      LOGICAL DEBUG,PROCED,ASQ
      INTEGER BLK,AS,TEMP,TYPE
      DATA BLK,AS,LT/' ','*','<'/
      IF(PROCED)  GO TO 101
      IF(IS.EQ.0)  GO TO 102
      N1 = IS+1
      N2 = IE-1
      NM(1) = BLK
      NM(2) = BLK
      K = 0
      ASQ = .TRUE.
      DO 1 I=N1,N2
      TEMP = INPUT(I)
      IAS = I
      IF(TEMP.EQ.BLK)  GO TO 1
      IF(TEMP.EQ.AS)  GO TO 2
      K = K+1
      IF(K.GT.8)  GO TO 103
      NM(1) = ISLL(8,NM(1))+ISRL(24,NM(2))
      NM(2) = ISLL(8,NM(2))+ISRL(24,TEMP)
    1 CONTINUE
      IF(K.EQ.0)  GO TO 104
      ASQ = .FALSE.
    2 CALL FNDNAM(NM,INDEX,KODE)
      GO TO(106,107,108,3,109,114),KODE
    3 IF(ASQ)  GO TO 31
      ITIMES = 1
      GO TO 32
   31 NUMST = IAS+1
      NUMND = IE-1
      CALL NMBR(INPUT,NUMST,NUMND,ANS,&100)
      ITIMES = ANS
      IF(ITIMES.LE.0.OR.ITIMES.GT.50)  GO TO 110
   32 LAST = IUSPCD(INDEX+2)-1
      NMBRST = IUSPCD(INDEX+3)
      INDEX = INDEX+4
      DO 11 LOOP=1,ITIMES
      IDX = INDEX
      DO 8 ISTMT=1,NMBRST
      LENGTH = IUSPCD(IDX)
      IF(ISTMT.EQ.NMBRST)  GO TO 9
      TYPE = IUSPCD(IDX+1)
      GO TO(4,5,6,7,71),TYPE
```

```
  4 IST = IDX+4
    IND = IST+LENGTH-3
    CALL RESLV1(IUSPCD,IST,IND,INPUT,&111)
    IST = 1
    CALL EVAL1(INPUT,IST,IND,ANS,&111)
    CALL NMEQNU(IUSPCD(IDX+2),ANS)
    GO TO 8
  5 CALL NMEQQS(IUSPCD(IDX+2))
    GO TO 8
  6 IST = IDX+2
    IND = IST+LENGTH-1
    CALL RESLV1(IUSPCD,IST,IND,INPUT,&111)
    IST = 1
    CALL EVAL1(INPUT,IST,IND,ANS,&111)
    CALL MSG(78,ANS,A,B)
    GO TO 8
  7 IST = IDX+2
    IND = IST+LENGTH-1
    CALL PRNT(IUSPCD,IST,IND)
    GO TO 8
 71 IST = IDX+2
    IND = IST+LENGTH-1
    CALL RESLV1(IUSPCD,IST,IND,INPUT,&111)
    IST = 1
    CALL EVAL1(INPUT,IST,IND,ANS,&111)
  8 IDX = IDX+LENGTH+2
  9 IF(IDX.GE.LAST)  GO TO 11
    IST = IDX+1
    IND = IST+IUSPCD(IDX)-1
    IREL = IND+1
    CALL RESLV1(IUSPCD,IST,IND,INPUT,&111)
    IST = 1
    CALL EVAL1(INPUT,IST,IND,ANS1,&111)
    IST = IREL+1
    IND = LAST
    CALL RESLV1(IUSPCD,IST,IND,INPUT,&111)
    IST = 1
    CALL EVAL1(INPUT,IST,IND,ANS2,&111)
    IF(IUSPCD(IREL).EQ.LT)  GO TO 10
    IF(ANS1.GT.ANS2)  GO TO 112
    GO TO 11
 10 IF(ANS1.LT.ANS2)  GO TO 112
 11 CONTINUE
    GO TO 113
101 CALL MSG(99,A,B,C)
    GO TO 100
102 CALL MSG(100,A,B,C)
    GO TO 100
103 CALL MSG(24,NM(1),NM(2),A)
    GO TO 100
104 CALL MSG(101,A,B,C)
    GO TO 100
105 CALL MSG(102,A,B,C)
    GO TO 100
```

```
106 IF(.NOT.ASQ)  GO TO 105
    CALL MSG(41,NM(1),NM(2),A)
    GO TO 100
107 CALL MSG(103,NM(1),NM(2),A)
    GO TO 100
108 CALL MSG(42,NM(1),NM(2),A)
    GO TO 100
109 CALL MSG(44,NM(1),NM(2),A)
    GO TO 100
110 CALL MSG(104,ITIMES,A,B)
    GO TO 100
111 CALL MSG(105,ISTMT,NM(1),NM(2))
    GO TO 100
112 CALL MSG(106,ANS1,IUSPCD(IREL),ANS2)
    GO TO 100
113 CALL MSG(107,ITIMES,A,B)
    GO TO 100
114 CALL MSG(133,NM(1),NM(2),A)
100 RETURN
    END
```

PGMEVL

```
    SUBROUTINE PGMEVL(INPUT,ILP,IEND,NM,*)
    DIMENSION NM(1),INPUT(1)
    COMMON /PROCDS/NPCD,LASTI,ICNT,IUSPCD(2000)
    COMMON /MODE1/DEBUG,PROCED
    INTEGER RP,BLK
    LOGICAL DEBUG,PROCED
    DATA LP,RP,BLK/'(',')',' '/
    CALL LOADED(NM,KODE)
    GO TO (3,1),KODE
  1 CALL LOAD(NM,KODE)
    GO TO (2,200),KODE
  2 CALL MSG(129,NM(1),NM(2),A)
  3 IF(PROCED)  GO TO 6
    IF(ILP.EQ.0)  GO TO 4
    IST = 1
    IEND = IEND-ILP+1
    CALL PRESS1(INPUT(ILP),IST,IEND,&100)
    IEND = ILP+IEND-2
    IST = ILP-1
    INPUT(IST) = NM(1)
    INPUT(IST+1) = NM(2)
    GO TO 5
  4 IST = 1
    IEND = 5
    INPUT(1) = NM(1)
    INPUT(2) = NM(2)
    INPUT(3) = LP
    INPUT(4) = BLK
    INPUT(5) = RP
  5 CALL EVAL1(INPUT(IST),IST,IEND,ANS,&100)
```

```
      GO TO 100
    6 IF(ILP.EQ.0)  GO TO 7
      IST = 1
      IEND = IEND-ILP+1
      CALL PRESS2(INPUT(ILP),IST,IEND,&100)
      IEND = ILP+IEND-2
      IST = ILP-1
      INPUT(IST) = NM(1)
      INPUT(IST+1) = NM(2)
      GO TO 8
    7 IST = 1
      IEND = 5
      INPUT(1) = NM(1)
      INPUT(2) = NM(2)
      INPUT(3) = LP
      INPUT(4) = BLK
      INPUT(5) = RP
    8 IF(LASTI+(IEND-IST+1)+2.GT.1996)  GO TO 10
      IUSPCD(ICNT) = IUSPCD(ICNT)+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = IEND-IST+1
      LASTI = LASTI+1
      IUSPCD(LASTI) = 5
      DO 9 I=IST,IEND
      LASTI = LASTI+1
    9 IUSPCD(LASTI) = INPUT(I)
      GO TO 100
   10 PROCED = .FALSE.
      LASTI = ICNT-4
      NPCD = NPCD-1
      CALL MSG(110,IUSPCD(LASTI+1),IUSPCD(LASTI+2),A)
  100 RETURN
  200 RETURN 1
      END
```

USRPGM

```
      SUBROUTINE USRPGM(INPUT,N1,N2,ILP,ANS,*)
      INTEGER INPUT(1),ARG(10),ANS,BLK,CM
      DATA BLK,CM/' ',','/
      NARG = 0
      IF(N1.GT.N2)  GO TO 10
      DO 1 I=N1,N2
      IF(INPUT(I).EQ.BLK.OR.INPUT(I).EQ.CM)  GO TO 1
      NARG = NARG+1
      IF(NARG.GT.10)  GO TO 3
      ARG(NARG) = INPUT(I)
    1 CONTINUE
   10 CALL RUNIT(INPUT(ILP-2),NARG,ARG,ANS,KODE)
      GO TO (2,4),KODE
    2 RETURN
    3 CALL MSG(137,INPUT(ILP-2),INPUT(ILP-1),A)
      GO TO 5
    4 CALL MSG(138,INPUT(ILP-2),INPUT(ILP-1),A)
    5 RETURN 1
      END
```

```
      SUBROUTINE RESLV1(IN,IS,IE,OUT,*)
      COMMON /NAMES/NMLT,NAME(100),IVALUE(50)
      INTEGER OUT(1)
      DIMENSION IN(1),IN1(88)
      INTEGER ARG,TEMP1,TEMP,AS,AS2,SH,PL,RP,BLK,CM
      DATA INT/' INT'/
      DATA BLK,CM/' ',','/
      DATA AS,AS2,SH,PL,MI,RP,LP/'*','**','/','+','-',')','('/
      KALL = 1
      GO TO 1
      ENTRY RESLV2(IN,IS,IE,OUT,NARG,NM,ARG,*)
      DIMENSION NM(1),ARG(1)
      KALL = 2
    1 JJ = 0
      DO 8 I=IS,IE
      TEMP = IN(I)
      IF(TEMP.NE.AS.AND.TEMP.NE.AS2.AND.TEMP.NE.SH.AND.TEMP.NE.PL.AND.
     *TEMP.NE.MI.AND.TEMP.NE.RP.AND.TEMP.NE.CM)  GO TO 6
      IF(I-2.LT.IS)  GO TO 6
      IF(IN(I-1).EQ.RP)  GO TO 6
      TEMP1 = IN(I-2)
      IF(TEMP1.EQ.AS.OR.TEMP1.EQ.AS2.OR.TEMP1.EQ.SH.OR.TEMP1.EQ.PL.OR.
     *TEMP1.EQ.MI.OR.TEMP1.EQ.RP.OR.TEMP1.EQ.LP.OR.TEMP1.EQ.CM)  GO TO 6
      IF(KALL.EQ.1)  GO TO 21
      DO 10 II=1,NARG,2
      I3 = (II+1)/2
      IF(IN(I-2).EQ.NM(II).AND.IN(I-1).EQ.NM(II+1))  GO TO 14
   10 CONTINUE
      GO TO 21
   14 JJ = JJ-1
      OUT(JJ) = ARG(I3)
      GO TO 6
   21 CALL FNDNAM(IN(I-2),L,KODE)
      IF(OUT(JJ-4).EQ.BLK.AND.OUT(JJ-3).EQ.INT.AND.OUT(JJ-2).EQ.LP)
     * GO TO 3
      GO TO(31,22,11,12,13,15),KODE
   22 L2 = L/2+1
      JJ = JJ-1
      OUT(JJ) = IVALUE(L2)
      GO TO 6
    3 IF(TEMP.NE.CM.AND.TEMP.NE.RP)  GO TO 41
      GO TO(42,43,6,44,45,46),KODE
   31 CALL MSG(55,IN(I-2),IN(I-1),A)
      READ 101,IN1
      DO 4 II=1,88
      IF(IN1(II).NE.BLK)  GO TO 5
    4 CONTINUE
      GO TO 99
    5 CALL NMBR(IN1,1,88,IANSR,&16)
      JJ = JJ-1
      OUT(JJ) = IANSR
      CALL NMEQNU(IN(I-2),IANSR)
      GO TO 6
```

```
   16 CALL MSG(57,A,B,C)
      GO TO 31
    6 JJ = JJ+1
      OUT(JJ) = TEMP
    8 CONTINUE
      IE = JJ
      GO TO 100
   11 CALL MSG(59,IN(I-2),IN(I-1),A)
      GO TO 99
   12 CALL MSG(64,IN(I-2),IN(I-1),A)
      GO TO 99
   13 CALL MSG(65,IN(I-2),IN(I-1),A)
      GO TO 99
   15 CALL MSG(132,IN(I-2),IN(I-1),A)
      GO TO 99
   41 CALL MSG(159,A,B,C)
      GO TO 99
   42 CALL MSG(160,IN(I-2),IN(I-1),A)
      GO TO 99
   43 CALL MSG(161,IN(I-2),IN(I-1),A)
      GO TO 99
   44 CALL MSG(162,IN(I-2),IN(I-1),A)
      GO TO 99
   45 CALL MSG(156,IN(I-2),IN(I-1),A)
      GO TO 99
   46 CALL MSG(157,IN(I-2),IN(I-1),A)
   99 RETURN 1
  100 RETURN
  101 FORMAT (88A1)
      END
.
.
.
.
C                                                          EVAL, EVAL1
      SUBROUTINE EVAL(INPUT,IS,IE,IANS,*)
      DIMENSION INPUT(1)
      INTEGER TEMP,RP
      DATA LP,RP/'(',')'/
      CALL PRESS1(INPUT,IS,IE,&11)
      ENTRY EVAL1(INPUT,IS,IE,IANS,*)
   12 IEND = IE
   13 ILP = 0
      IRP=0
      DO 15 I=1,IEND
      TEMP=INPUT(I)
      IF(TEMP.NE.LP) GO TO 14
      ILP=I
      GO TO 15
   14 IF(TEMP.NE.RP) GO TO 15
      IRP=I
      CALL EXPR(INPUT,ILP,IRP,IEND,&50,&11)
      IF(IEND.GT.1)  GO TO 13
      GO TO 16
```

```
   15 CONTINUE
   10 IF(ILP.NE.0.OR.IRP.NE.0) GO TO 17
      IF(IEND.EQ.1) GO TO 16
      ILP=0
      IRP=IEND+1
      CALL EXPR(INPUT,ILP,IRP,IEND,&50,&11)
   16 IANS = INPUT(IEND)
      RETURN
   17 CALL MSG(58,A,B,C)
   11 RETURN 1
   50 IST = ILP+1
      CALL INTIT(INPUT(IST),IANS,&11)
      INPUT(ILP-2) = IANS
      IST = ILP-1
      N1 = IRP+1
      J = IST-1
      IF(N1.GT.IEND)  GO TO 52
      DO 51 I=N1,IEND
      J = J+1
   51 INPUT(J) = INPUT(I)
   52 IEND = J
      GO TO 13
      END
.
.
.
.                                                                    USRFCT
C
      SUBROUTINE USRFCT(INPUT,N1,N1,ILP,IRP,LST,INDEX,*)
C       RETURN  OK
C       RETURN 1
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      DIMENSION INPUT(1),NM(20)
      INTEGER ARG(10),CM,BLK
      DATA BLK,CM/' ',','/
      NARG1 = 0
      DO 3 J=N1,N2
      IF(INPUT(J).EQ.BLK.OR.INPUT(J).EQ.CM)  GO TO 3
      NARG1 = NARG1+1
      IF(NARG1.GT.10)  GO TO 15
      ARG(NARG1) = INPUT(J)
      I1 = 2*NARG1+2+INDEX
      NM(2*NARG1-1) = IUSFCT(I1)
      NM(2*NARG1) = IUSFCT(I1+1)
    3 CONTINUE
      J = 441
      K = LST+1
      N = LST-IRP
      IF(N.EQ.0)  GO TO 2
      DO 1 I=1,N
      J = J-1
      K = K-1
    1 INPUT(J) = INPUT(K)
    2 NARG = IUSFCT(INDEX+3)
```

```
      IF(NARG.NE.2*NARG1)  GO TO 15
      N1 = INDEX+NARG+4
      N2 = IUSFCT(INDEX+2)-1
      CALL RESLV2(IUSFCT,N1,N2,INPUT(ILP-2),NARG,NM,ARG,&99)
      JJ = N2+ILP-3
      IF(JJ.GT.J)  GO TO 14
      IF(JJ.EQ.J)  GO TO 10
      IF(J.EQ.441)  GO TO 10
      DO 9 I=J,440
      JJ = JJ+1
    9 INPUT(JJ) = INPUT(I)
   10 LST = JJ
      GO TO 100
   14 CALL MSG(60,A,B,C)
      GO TO 99
   15 CALL MSG(128,IUSFCT(INDEX),IUSFCT(INDEX+1),IUSFCT(INDEX+3)/2)
   99 RETURN 1
  100 RETURN
      END
.
.
.
.                                                                  PGMFCT
C
      SUBROUTINE PGMFCT(IFN,ARG1,ANS1)
    3 GO TO (501,502,503,504,505,506,507,508,509,510,
     *        511,512,513,514,515,516,517,518),IFN
  501 ANS1 =     EXP(ARG1)
      GO TO 600
  502 ANS1 =    ALOG(ARG1)
      GO TO 600
  503 ANS1 = ALOG10(ARG1)
      GO TO 600
  504 ANS1 =   ARSIN(ARG1)
      GO TO 600
  505 ANS1 =   ARCOS(ARG1)
      GO TO 600
  506 ANS1 =    ATAN(ARG1)
      GO TO 600
  507 ANS1 =     SIN(ARG1)
      GO TO 600
  508 ANS1 =     COS(ARG1)
      GO TO 600
  509 ANS1 =     TAN(ARG1)
      GO TO 600
  510 ANS1 =    SQRT(ARG1)
      GO TO 600
  511 ANS1 =    TANH(ARG1)
      GO TO 600
  512 ANS1 =    SINH(ARG1)
      GO TO 600
  513 ANS1 =    COSH(ARG1)
      GO TO 600
  514 ANS1 =     ERF(ARG1)
```

```
      GO TO 600
  515 ANS1 =    ERFC(ARG1)
      GO TO 600
  516 ANS1 =  GAMMA(ARG1)
      GO TO 600
  517 ANS1 = ALGAMA(ARG1)
      GO TO 600
  518 ANS1 =    ABS(ARG1)
  600 RETURN
      END
.
.
.
.
C                                                                    EXPR
      SUBROUTINE EXPR(INPUT,ILP,IRP,LST,*,*)
      DIMENSION INPUT(1)
      LOGICAL FCT
      INTEGER TEMP,AS,SH,PL,AS2,BLK,CM
      DATA LP,CM/'(',','/
      DATA AS,SH,PL,MI,AS2,BLK/'*','/','+','-','**',' '/
      FCT = .FALSE.
      IST=ILP+1
      IND=IRP-1
      IF(IST.LT.4) GO TO 1
      TEMP=INPUT(IST-2)
      IF(TEMP.EQ.AS.OR.TEMP.EQ.SH.OR.TEMP.EQ.PL.OR.TEMP.EQ.MI.OR.
     *TEMP.EQ.AS2.OR.TEMP.EQ.LP.OR.TEMP.EQ.CM)  GO TO 1
      FCT = .TRUE.
    1 IF(IST.GE.IND)  GO TO 6
      J = IST-1
      DO 2 I=IST,IND
      J=J+1
      TEMP=INPUT(I)
      IF(TEMP.NE.BLK)INPUT(J)=TEMP
      IF(TEMP.NE.AS2) GO TO 2
      CALL EXPON(INPUT(J-1),INPUT(I+1),IANS,&99)
      INPUT(J-1)=IANS
      INPUT(I)=BLK
      INPUT(I+1)=BLK
      J=J-2
    2 CONTINUE
      IND=J
      IF(IST.EQ.IND) GO TO 6
      J=IST-1
      DO 3 I=IST,IND
      J=J+1
      TEMP=INPUT(I)
      IF(TEMP.NE.BLK)INPUT(J)=TEMP
      IF(TEMP.NE.SH.AND.TEMP.NE.AS) GO TO 3
      IF(TEMP.EQ.AS)CALL MULT(INPUT(J-1),INPUT(I+1),IANS,&99)
      IF(TEMP.EQ.SH) CALL DVD(INPUT(J-1),INPUT(I+1),IANS,&99)
      INPUT(J-1)=IANS
      INPUT(I)=BLK
```

```
      INPUT(I+1)=BLK
      J=J-2
    3 CONTINUE
      IND=J
      IF(IST.EQ.IND) GO TO 6
      J=IST-1
      DO 4 I=IST,IND
      J=J+1
      TEMP=INPUT(I)
      IF(TEMP.NE.BLK) INPUT(J)=TEMP
      IF(TEMP.NE.PL.AND.TEMP.NE.MI) GO TO 4
      IF(TEMP.EQ.PL)CALL ADD(INPUT(J-1),INPUT(I+1),IANS,&99)
      IF(TEMP.EQ.MI)CALL SUB(INPUT(J-1),INPUT(I+1),IANS,&99)
      INPUT(J-1)=IANS
      INPUT(I)=BLK
      INPUT(I+1)=BLK
      J=J-2
    4 CONTINUE
      IND=J
      IF(FCT)  GO TO 7
      IF(IST.EQ.IND) GO TO 6
      CALL MSG(68,A,B,C)
      GO TO 99
    6 IF(ILP.NE.0) GO TO 7
      LST=1
      INPUT(1)= INPUT(IST)
      GO TO 100
    7 INPUT(ILP)=INPUT(IST)
      IF(.NOT.FCT)  GO TO 72
      CALL FNDNAM(INPUT(ILP-2),INDEX,KODE)
      GO TO (73,102,70,103,71,75),KODE
   70 CALL USRFCT(INPUT,IST,IND,ILP,IRP,LST,INDEX,&99)
      GO TO 100
   71 IF(INDEX.EQ.19)  RETURN 1
      CALL PGMFCT(INDEX,INPUT(IST),IANS)
      INPUT(ILP-2) = IANS
      IST = ILP-1
      GO TO 72
   73 CALL LOAD(INPUT(ILP-2),KODE)
      GO TO (74,101),KODE
   74 CALL MSG(129,INPUT(ILP-2),INPUT(ILP-1),A)
   75 CALL USRPGM(INPUT,IST,IND,ILP,IANS,&99)
      INPUT(ILP-2) = IANS
      IST = ILP-1
   72 N1 = IRP+1
      J=IST-1
      IF(N1.GT.LST)  GO TO 9
      DO 8 I=N1,LST
      J=J+1
    8 INPUT(J)=INPUT(I)
    9 LST = J
      GO TO 100
  101 CALL MSG(69,INPUT(ILP-2),INPUT(ILP-1),A)
      GO TO 99
```

```
  102 CALL MSG(108,INPUT(ILP-2),INPUT(ILP-1),A)
      GO TO 99
  103 CALL MSG(109,INPUT(ILP-2),INPUT(ILP-1),A)
   99 RETURN 2
  100 RETURN
      END
.
.
.
.                                                    EXPON, DVD, MULT, ADD, SUB
C
      SUBROUTINE EXPON(X1,X2,ANS,*)
      INTEGER OP(5)/'**','/','*','+','-'/
      IOP = 1
      I = X2
      IF(FLOAT(I).EQ.X2)  GO TO 1
      ANS = X1**X2
      CALL OVERFL(J)
      GO TO (2,100,3),J
    1 ANS = X1**I
      CALL OVERFL(J)
      GO TO (2,100,3),J
      ENTRY DVD(X1,X2,ANS,*)
      IOP = 2
      IF(X2.EQ.0)  GO TO 4
      ANS = X1/X2
      CALL OVERFL(J)
      GO TO (2,100,3),J
      ENTRY MULT(X1,X2,ANS,*)
      IOP = 3
      ANS = X1*X2
      CALL OVERFL(J)
      GO TO (2,100,3),J
      ENTRY ADD(X1,X2,ANS,*)
      IOP = 4
      ANS = X1+X2
      CALL OVERFL(J)
      GO TO (2,100,3),J
      ENTRY SUB(X1,X2,ANS,*)
      IOP = 5
      ANS = X1-X2
      CALL OVERFL(J)
      GO TO (2,100,3),J
    2 CALL MSG(153,X1,OP(IOP),X2)
      GO TO 99
    3 CALL MSG(154,X1,OP(IOP),X2)
      GO TO 99
    4 CALL MSG(70,A,B,C)
   99 RETURN 1
  100 RETURN
      END
```

```
C                                                                    FOFX
      SUBROUTINE FOFX(ARG,IANS,*)
      DIMENSION INTMP(441)
      INTEGER TEMP,RP
      DATA LP,RP/'(',')'/
      COMMON/FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON/NANCY/ INDEX,NARG,NA,NB
      N1 = NA
      N2 = NB
      CALL RESLV2(IUSFCT,N1,N2,INTMP,NARG,IUSFCT(INDEX+4),ARG,&98)
   12 IEND = N2
   13 ILP = 0
      IRP=0
      DO 15 I=1,IEND
      TEMP=INTMP(I)
      IF(TEMP.NE.LP) GO TO 14
      ILP=I
      GO TO 15
   14 IF(TEMP.NE.RP) GO TO 15
      IRP=I
      CALL EXPR(INTMP,ILP,IRP,IEND,&50,&98)
      IF(IEND.GT.1)  GO TO 13
      GO TO 16
   15 CONTINUE
      IF(ILP.NE.0.OR.IRP.NE.0) GO TO 17
      IF(IEND.EQ.1) GO TO 16
      ILP=0
      IRP=IEND+1
      CALL EXPR(INTMP,ILP,IRP,IEND,&50,&98)
   16 IANS = INTMP(IEND)
      RETURN
   17 CALL MSG(58,A,B,C)
      GO TO 99
   50 CALL MSG(165,IUSFCT(INDEX),IUSFCT(INDEX+1),A)
      GO TO 99
   98 CALL MSG(120,IUSFCT(INDEX),IUSFCT(INDEX+1),A)
   99 RETURN 1
      END
.
.                                                                    INTIT
.
      SUBROUTINE INTIT(INPUT,ANS,*)
      DIMENSION INPUT(1)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON/NANCY/ INDEX,NARG,NA,NB
      INTEGER CM,BLK,RP
      DATA CM,BLK,RP/',',' ',')'/
      IF(INPUT(3).NE.CM.OR.INPUT(5).NE.CM.OR.(INPUT(7).NE.BLK.AND.
     *INPUT(7).NE.RP))  GO TO 1
```

```
      CALL FNDNAM(INPUT(1),INDEX,KODE)
      GO TO(2,3,4,5,6,7),KODE
    4 NARG = IUSFCT(INDEX+3)
      IF(NARG.GT.2)  GO TO 8
      NA = INDEX+NARG+4
      NB = IUSFCT(INDEX+2)-1
      CALL SIMPS1(INPUT(4),INPUT(6),ANS,&98,&99)
      GO TO 100
   98 CALL MSG(122,INPUT(1),INPUT(2),A)
      GO TO 100
    1 CALL MSG(163,INPUT(1),INPUT(2),A)
      GO TO 99
    2 CALL MSG(160,INPUT(1),INPUT(2),A)
      GO TO 99
    3 CALL MSG(161,INPUT(1),INPUT(2),A)
      GO TO 99
    5 CALL MSG(162,INPUT(1),INPUT(2),A)
      GO TO 99
    6 CALL MSG(156,INPUT(1),INPUT(2),A)
      GO TO 99
    7 CALL MSG(157,INPUT(1),INPUT(2),A)
      GO TO 99
    8 CALL MSG(127,IUSFCT(INDEX),IUSFCT(INDEX+1),A)
   99 RETURN 1
  100 RETURN
      END
.
.
.
.                                                                    INTGRL
C
C        SUBROUTINE TO DO INTEGRALS FOLLOWING INTEGRATE COMMAND
      SUBROUTINE INTGRL
      LOGICAL NUM,EQSIGN,DEBUG,PROCED
      REAL*8 GARB
      DIMENSION INPUT(441),NM(3),INPCT(90)
      COMMON /FCTS/NFCT,LSTI,IUSFCT(1000)
      COMMON/MODE1/ DEBUG,PROCED
      COMMON /NANCY/IND,NARG,NA,NB
      DATA INPCT(1),INPCT(90) / '(    ' , ')    ' /
      INTEGER BLK
      DATA BLK /' '/
      IF(.NOT. PROCED) GO TO 10
      CALL MSG(121,G,G,G)
      RETURN
C        ASK FOR INTEGRAND
   10 CALL MSG(5,G,G,G)
    1 CALL READT1(INPUT,NM,K,NUM,ILP,IRP,IEQ,IQM,IOP,ILST)
      EQSIGN = .FALSE.
      IF(ILST.EQ.C)  GO TO 100
   11 IF(.NOT.NUM) GO TO 12
C        NUMBER ENCOUNTERED
      CALL MSG(112,G,G,G)
      GO TO 1
```

```
   12 IF(K .LE. 8) GO TO 13
C        TOO MANY CHARACTERS
      CALL MSG(3,NM(1),NM(2),NM(3))
      GO TO 1
   13 IF(IEQ .EQ. 0) GO TO 2
C        = SIGN FOUND, ASSUME DEFINITION
      EQSIGN = .TRUE.
      IF(IOP.EQ.0 .OR. IOP.GT.IEQ) GO TO 14
C        OPERATOR BEFORE = SIGN
      CALL MSG(113,G,G,G)
      GO TO 1
   14 IF(IQM .EQ. 0) GO TO 15
C        QUESTION MARK FOUND
      CALL MSG(114,G,G,G)
      GO TO 1
   15 IF(IRP.GT.0 .AND. IRP.LT.IEQ) GO TO 16
C        NO RIGHT PAREN BEFORE = SIGN
      CALL MSG(115,G,G,G)
      GO TO 1
   16 IF(ILP.GT.0 .AND. ILP.LT.IRP) GO TO 2
C        NO LEFT PAREN BEFORE RIGHT PAREN
      CALL MSG(116,G,G,G)
      GO TO 1
C        GET NAME SET UP
    2 CALL FNDNAM(NM(2),INDEX,KODE)
      IND = INDEX
      IF(KODE.NE.3) GO TO 21
      IF(.NOT.EQSIGN) GO TO 21
C        USED NAME ALREADY DEFINED
      CALL MSG(48,NM(2),NM(3),G)
      GO TO 1
   21 IF(KODE.NE.1) GO TO 22
      IF(EQSIGN) GO TO 22
C        NAME NOT DEFINED
      CALL MSG(41,NM(2),NM(3),G)
      GO TO 1
   22 IF(KODE.NE. 2) GO TO 23
C        VALUE
      CALL MSG(47,NM(2),NM(3),G)
      GO TO 1
   23 IF(KODE.NE.4) GO TO 24
C        PROCEDURE
      CALL MSG(53,NM(2),NM(3),G)
      GO TO 1
   24 IF(KODE.NE.5) GO TO 25
C        SYSTEM FUNCTION
      CALL MSG(54,NM(2),NM(3),G)
      GO TO 1
   25 IF(KODE.NE.6)  GO TO 26
      CALL MSG(136,NM(2),NM(3),G)
      GO TO 1
   26 IF(KODE.EQ.3) GO TO 4
    3 IE = IEQ + 1
      CALL FNEQEX(NM(2),INPUT,ILP,IRP,IE,ILST)
```

```
      CALL FNDNAM(NM(2),INDEX,KODE)
      IND = INDEX
      IF(KODE.EQ.3) GO TO 4
      GO TO 1
C        FUNCTION NOW DEFINED
C        ASK FOR ENDPOINTS
    4 CALL MSG(117,G,G,G)
      READ 200,(INPUT(I),I=2,89)
  200 FORMAT(88A1)
      DO 17 I=2,89
      IF(INPUT(I).NE.BLK)  GO TO 18
   17 CONTINUE
      GO TO 100
C        CONVERT THE ENDPOINTS
   18 KKK = 1
    5 CALL CNVRT(INPUT,KKK,90,1,A,CARB,CHAR,&7,&6,&8)
      GO TO 7
    6 CALL CNVRT(INPUT,KKK,90,1,B,CARB,CHAR,&7,&7,&8)
C        BOTH A AND B OK
      NARG = IUSFCT(IND+3)
      IF(NARG.GT.2)  GO TO 27
      NA = IND+NARG+4
      NB = IUSFCT(IND+2)-1
      CALL SIMPS1(A,B,ANSW,&98,&100)
      GO TO 99
   98 CALL MSG(122,NM(2),NM(3),G)
   99 CALL MSG(118,ANSW,G,G)
  100 RETURN
    7 CALL MSG(119,G,G,G)
      GO TO 4
    8 CALL MSG(76,CHAR,G,G)
      GO TO 4
   27 CALL MSG(127,IUSFCT(IND),IUSFCT(IND+1),G)
      GO TO 100
      END
```

SIMPS1

```
C
      SUBROUTINE SIMPS1(XMIN,XMAX,ANS,*,*)
      DIMENSION V(200),H(200),A(200),B(200),C(200),P(200),E(200)
      DATA T/3.0E-4/
      IF(XMIN.EQ.XMAX)  GO TO 18
      V(1)=XMIN
      H(1)=0.5*(XMAX-XMIN)
      CALL FOFX(XMIN,A(1),&100)
      CALL FOFX(XMIN+H(1),B(1) ,&100)
      CALL FOFX(XMAX,C(1),&100)
      P(1)=H(1)*(A(1)+4.0*B(1)+C(1))
      E(1)=P(1)
      ANS=P(1)
      N=1
      FRAC=2.0*T
```

```
    1 FRAC=0.5*FRAC
    2 TEST=ABS(FRAC*ANS)
      K=N
    3 DO 7 I=1,K
    4 IF(ABS(E(I)).LE.TEST)  GO TO 7
    5 N = N+1
      V(N)=V(I)+H(I)
      H(N)=0.5*H(I)
      A(N)=B(I)
      CALL FOFX(V(N)+H(N),B(N),&100)
      C(N)=C(I)
      P(N)=H(N)*(A(N)+4.0*B(N)+C(N))
      Q=P(I)
      H(I)=H(N)
      CALL FOFX(V(I)+H(I),B(I),&100)
      C(I)=A(N)
      P(I)=H(I)*(A(I)+4.0*B(I)+C(I))
      Q=P(I)+P(N)-Q
      ANS=ANS+Q
      E(I)=Q
      E(N)=Q
    6 IF(N-200) 7,13,13
    7 CONTINUE
    8 IF (N-K) 9,9,2
    9 Q = 0.0
   10 DO 11 I=1,N
   11 Q=Q+E(I)
   12 IF (ABS(Q)-T*ABS(ANS)) 14,14,1
   13 RETURN 1
   14 ANS=0.0
   15 DO 16 I=1,N
   16 ANS=ANS+P(I)
      ANS = (ANS+Q/30.0)/3.0
   17 RETURN
   18 ANS = 0.0
      RETURN
  100 RETURN 2
      END
.
.
.
.                                                                    CREATE
C        MAIN PROGRAM, DECK NAME CREATE, RUN TO CREATE KEYWORDS FOR INITAL
      IMPLICIT INTEGER(A-Z)
      DIMENSION KEYWRD(3,20),NCMNDS(20),CMNDS(3,33,20),PGMS(2,33,20)
      DATA NKEYS,KEYWRD,NCMNDS,CMNDS,PGMS/0,60*' ',20*0,3300*' '/
      REWIND 5
      WRITE (5,201) NKEYS,KEYWRD,NCMNDS,CMNDS,PGMS
      END FILE 5
      REWIND 5
      READ (5,201) NKEYS,KEYWRD,NCMNDS,CMNDS,PGMS
      STOP
  201 FORMAT (I4,3(/,20A4),/,20I4,165(/,20A4))
      END
```

```
C
      SUBROUTINE MSG(I,A,B,C)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
     1,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45
     2,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67
     3,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89
     4,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,
     5108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,
     6124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,
     7140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,
     8156,157,158,159,160,161,162,163,164,165),I
    1 PRINT 501
      GO TO 1000
    2 PRINT 502
      GO TO 1000
    3 PRINT 503,A,B,C
      GO TO 1000
    4 PRINT 504
      GO TO 1000
    5 PRINT 505
      GO TO 1000
    6 PRINT 506
      GO TO 1000
    7 PRINT 507
      GO TO 1000
    8 PRINT 508
      GO TO 1000
    9 PRINT 509
      GO TO 1000
   10 PRINT 510,A,B,C
      GO TO 1000
   11 PRINT 511
      GO TO 1000
   12 PRINT 512
      GO TO 1000
   13 PRINT 513
      GO TO 1000
   14 PRINT 514,A,B,C
      GO TO 1000
   15 PRINT 515
      GO TO 1000
   16 PRINT 516
      GO TO 1000
   17 PRINT 517
      GO TO 1000
   18 PRINT 518
      GO TO 1000
   19 PRINT 519
      GO TO 1000
   20 PRINT 520
      GO TO 1000
   21 PRINT 521
      GO TO 1000
   22 PRINT 522
```

```
      GO TO 1000
23    PRINT 523,A,B
      GO TO 1000
24    PRINT 524,A,B
      GO TO 1000
25    PRINT 525
      GO TO 1000
26    PRINT 526
      GO TO 1000
27    PRINT 527
      GO TO 1000
28    PRINT 528
      GO TO 1000
29    PRINT 529
      GO TO 1000
30    PRINT 530
      GO TO 1000
31    PRINT 531,A,B
      GO TO 1000
32    PRINT 532
      GO TO 1000
33    PRINT 533
      GO TO 1000
34    PRINT 534
      GO TO 1000
35    PRINT 535
      GO TO 1000
36    CONTINUE
      GO TO 1000
37    PRINT 537
      GO TO 1000
38    CONTINUE
      GO TO 1000
39    PRINT 539
      GO TO 1000
40    PRINT 540
      GO TO 1000
41    PRINT 541,A,B
      GO TO 1000
42    PRINT 542,A,B,C
      GO TO 1000
43    PRINT 543,A,B
      GO TO 1000
44    PRINT 544,A,B
      GO TO 1000
45    PRINT 545
      GO TO 1000
46    PRINT 546,A,B
      GO TO 1000
47    PRINT 547,A,B
      GO TO 1000
48    PRINT 548,A,B
      GO TO 1000
49    PRINT 549
```

```
      GO TO 1000
50 PRINT 550
      GO TO 1000
51 PRINT 551,A,B
      GO TO 1000
52 PRINT 552
      GO TO 1000
53 PRINT 553,A,B
      GO TO 1000
54 PRINT 554,A,B
      GO TO 1000
55 PRINT 555,A,B
      GO TO 1000
56 PRINT 556,A,B
      GO TO 1000
57 PRINT 557
      GO TO 1000
58 PRINT 558
      GO TO 1000
59 PRINT 559,A,B
      GO TO 1000
60 PRINT 560
      GO TO 1000
61 PRINT 561,A,B
      GO TO 1000
62 PRINT 562,A
      GO TO 1000
63 PRINT 563,A
      GO TO 1000
64 PRINT 564,A,B
      GO TO 1000
65 PRINT 565,A,B
      GO TO 1000
66 PRINT 566
      GO TO 1000
67 PRINT 567,A,B
      GO TO 1000
68 PRINT 568
      GO TO 1000
69 PRINT 569,A,B
      GO TO 1000
70 PRINT 570
      GO TO 1000
71 CONTINUE
      GO TO 1000
72 CONTINUE
      GO TO 1000
73 CONTINUE
      GO TO 1000
74 PRINT 574
      GO TO 1000
75 PRINT 575
      GO TO 1000
76 PRINT 576,A
```

```
    GO TO 1000
 77 PRINT 577,A,B,C
    GO TO 1000
 78 PRINT 578,A
    GO TO 1000
 79 PRINT 579,A,B,C
    GO TO 1000
 80 PRINT 580,A,B,C
    GO TO 1000
 81 PRINT 581,A,B,C
    GO TO 1000
 82 PRINT 582
    GO TO 1000
 83 PRINT 583,A,B,C
    GO TO 1000
 84 PRINT 584
    GO TO 1000
 85 PRINT 585
    GO TO 1000
 86 PRINT 586,A
    GO TO 1000
 87 PRINT 587
    GO TO 1000
 88 PRINT 588
    GO TO 1000
 89 PRINT 589,A,B
    GO TO 1000
 90 PRINT 590,A,B
    GO TO 1000
 91 PRINT 591,A,B
    GO TO 1000
 92 PRINT 592,A,B
    GO TO 1000
 93 PRINT 593
    GO TO 1000
 94 PRINT 594
    GO TO 1000
 95 PRINT 595
    GO TO 1000
 96 PRINT 596
    GO TO 1000
 97 PRINT 597
    GO TO 1000
 98 PRINT 598
    GO TO 1000
 99 PRINT 599
    GO TO 1000
100 PRINT 600
    GO TO 1000
101 PRINT 601
    GO TO 1000
102 PRINT 602
    GO TO 1000
103 PRINT 603,A,B
```

```
      GO TO 1000
104 PRINT 604,A
      GO TO 1000
105 PRINT 605,A,B,C
      GO TO 1000
106 PRINT 606,A,B,C
      GO TO 1000
107 PRINT 607,A
      GO TO 1000
108 PRINT 608,A,B
      GO TO 1000
109 PRINT 609,A,B
      GO TO 1000
110 PRINT 610,A,B
      GO TO 1000
111 PRINT 611
      GO TO 1000
112 PRINT 612
      GO TO 1000
113 PRINT 613
      GO TO 1000
114 PRINT 614
      GO TO 1000
115 PRINT 615
      GO TO 1000
116 PRINT 616
      GO TO 1000
117 PRINT 617
      GO TO 1000
118 PRINT 618,A
      GO TO 1000
119 PRINT 619
      GO TO 1000
120 PRINT 620,A,B
      GO TO 1000
121 PRINT 621
      GO TO 1000
122 PRINT 622,A,B
      GO TO 1000
123 PRINT 623
      GO TO 1000
124 PRINT 624
      GO TO 1000
125 PRINT 625
      GO TO 1000
126 PRINT 626
      GO TO 1000
127 PRINT 627,A,B
      GO TO 1000
128 PRINT 628,A,B,C
      GO TO 1000
129 PRINT 629,A,B
      GO TO 1000
130 PRINT 630,A,B
```

```
    GO TO 1000
131 PRINT 631,A,B,C
    GO TO 1000
132 PRINT 632,A,B
    GO TO 1000
133 PRINT 633,A,B
    GO TO 1000
134 PRINT 634,A,B
    GO TO 1000
135 PRINT 635,A,B
    GO TO 1000
136 PRINT 636,A,B
    GO TO 1000
137 PRINT 637,A,B
    GO TO 1000
138 PRINT 638,A,B
    GO TO 1000
139 PRINT 639
    GO TO 1000
140 PRINT 640
    GO TO 1000
141 PRINT 641
    GO TO 1000
142 PRINT 642
    GO TO 1000
143 PRINT 643
    GO TO 1000
144 PRINT 644
    GO TO 1000
145 PRINT 645,A,B,C
    GO TO 1000
146 PRINT 646,A,B,C
    GO TO 1000
147 PRINT 647,A,B,C
    GO TO 1000
148 PRINT 648
    GO TO 1000
149 PRINT 649
    GO TO 1000
150 PRINT 650
    GO TO 1000
151 PRINT 651,A,B,C
    GO TO 1000
152 PRINT 652
    GO TO 1000
153 PRINT 653,A,B,C
    GO TO 1000
154 PRINT 654,A,B,C
    GO TO 1000
155 PRINT 655,A,B,C
    GO TO 1000
156 PRINT 656,A,B
    GO TO 1000
157 PRINT 657,A,B
```

```
      GO TO 1000
  158 PRINT 658,A
      GO TO 1000
  159 PRINT 659
      GO TO 1000
  160 PRINT 660,A,B
      GO TO 1000
  161 PRINT 661,A,B
      GO TO 1000
  162 PRINT 662,A,B
      GO TO 1000
  163 PRINT 663,A,B
      GO TO 1000
  164 PRINT 664,A,B
      GO TO 1000
  165 PRINT 665,A,B
 1000 RETURN
  501 FORMAT (' READY')
  502 FORMAT (' STATEMENT CANNOT BE CLASSIFIED.')
  503 FORMAT (' ',3A4,' HAS MORE THAN 8 CHARACTERS.')
  504 FORMAT (' PARENS DO NOT BALANCE.')
  505 FORMAT(' ENTER USER FUNCTION NAME, DEFINE USER FUNCTION, OR PRESS
     *RETURN TO CANCEL.')
  506 FORMAT (' CHARACTERS AFTER ? IGNORED.')
  507 FORMAT (' MORE THAN 12 CHARACTERS IN A COMMAND.')
  508 FORMAT (' STOP COMMAND NOT ALLOWED IN PROCEDURES.')
  509 FORMAT (' MORE THAN 2 CONTINUE LINES.')
  510 FORMAT (' ',3A4,' NOT A COMMAND OR PROGRAM NAME.')
  511 FORMAT (' INITIALIZATION COMPLETE.')
  512 FORMAT (' RESTART COMMAND NOT ALLOWED IN PROCEDURES.')
  513 FORMAT (' NO RIGHT PAREN. MODE IGNORED.')
  514 FORMAT (' ',3A4,' IS NOT A VALID MODE.')
  515 FORMAT (' DEBUG MODE IS ALREADY ON.')
  516 FORMAT (' DEBUG MODE TURNED ON.')
  517 FORMAT (' DEBUG MODE TURNED OFF.')
  518 FORMAT (' DEBUG MODE IS ALREADY OFF.')
  519 FORMAT (' REAL*4 MODE IS ALREADY ON.')
  520 FORMAT (' MORE THAN 12 CHARACTERS IN MODE OPTION. OPTION IGNORED')
  521 FORMAT (' NO OPTION FOUND IN MODE COMMAND.')
  522 FORMAT (' MODE COMMAND NOT ALLOWED IN PROCEDURES.')
  523 FORMAT (' ',2A4,' WAS NOT FOUND')
  524 FORMAT (' MORE THAN 8 CHARACTERS IN ',2A4)
  525 FORMAT (' BLANK FIELD FOUND IN ERASE COMMAND.')
  526 FORMAT (' NO RIGHT PAREN FOUND. LAST NAME MAY NOT BE ERASED')
  527 FORMAT (' YOU ERASED ALL NAMES')
  528 FORMAT (' THERE ARE NO NAMES TO ERASE.')
  529 FORMAT (' NO NAMES FOUND IN ERASE COMMAND.')
  530 FORMAT (' ERASE COMMAND NOT ALLOWED IN PROCEDURES.')
  531 FORMAT (' ',2A4,' MAY NOT BE ERASED.  IT IS A SYSTEM SUPPLIED',
     *' FUNCTION.')
  532 FORMAT (' DUMP OF VALUES.')
  533 FORMAT (' NO VALUES DEFINED.')
  534 FORMAT (' NO USER FUNCTIONS DEFINED.')
  535 FORMAT (' DUMP OF USER FUNCTION NAMES.')
```

```
537 FORMAT (' DUMP OF PROCEDURE NAMES.')
539 FORMAT (' NO PROCEDURES DEFINED.')
540 FORMAT (' DUMP COMMAND NOT ALLOWED IN PROCEDURES.')
541 FORMAT (' ',2A4,' COULD NOT BE FOUND.')
542 FORMAT (' ',2A4,' IS A USER FUNCTION OF ',I3,' ARGUMENTS.')
543 FORMAT (' ',2A4,' IS A PROCEDURE.')
544 FORMAT (' ',2A4,' IS A SYSTEM SUPPLIED FUNCTION.')
545 FORMAT (' NO ANSWER.')
546 FORMAT (' ',2A4,' NOT STORED.')
547 FORMAT (' ',2A4,' ALREADY STORED AS A VALUE, FUNCTION NOT STORED')
548 FORMAT (' ',2A4,' ALREADY STORED AS A USER FUNCTION.')
549 FORMAT (' NOT ENOUGH ROOM TO STORE FUNCTION.')
550 FORMAT (' NO ARGUMENT FOUND.')
551 FORMAT (' THE ARGUMENT ',2A4,' HAS MORE THAN 8 CHARACTERS.')
552 FORMAT (' FUNCTION DEFINITIONS NOT ALLOWED IN PROCEDURES.')
553 FORMAT (' ',2A4,' ALREADY STORED AS A PROCEDURE, FUNCTION NOT',
   *' STORED.')
554 FORMAT (' ',2A4,' IS A SYSTEM FUNCTION.  USER FUNCTION NOT',
   *' STORED.')
555 FORMAT (' ',2A4,' UNKNOWN.  ENTER NUMBER OR PRESS RETURN TO',
   *' CANCEL.')
556 FORMAT (' ',2A4,' MORE THAN 8 CHARACTERS.  EVALUATION CANCELED.')
557 FORMAT (' ERROR IN NUMBER CONVERSION.')
558 FORMAT (' PARENS DO NOT BALANCE.')
559 FORMAT (' USER FUNCTION NAME ',2A4,' MAY NOT BE USED AS A VALUE',
   *' NAME.')
560 FORMAT (' EXPRESSION TOO LONG.')
561 FORMAT (' TWO CONSECUTIVE OPERATORS, ',A4,' AND ',A4)
562 FORMAT (' ILLEGAL OPERATOR ',A4)
563 FORMAT (' ONLY ONE OPERAND FOR ',A4)
564 FORMAT (' PROCEDURE NAME ',2A4,' MAY NOT BE USED AS A VALUE NAME')
565 FORMAT (' SYSTEM FUNCTION NAME ',2A4,' MAY NOT BE USED AS A',
   *' VALUE NAME.')
566 FORMAT (' YOU HAVE ROOM FOR ONLY ONE MORE VALUE NAME.')
567 FORMAT (' ',2A4,' NOT STORED.  NO ROOM.')
568 FORMAT(' ERROR IN EXPR,IST.NE.IND')
569 FORMAT (' FUNCTION ',2A4,' UNKNOWN.')
570 FORMAT (' DIVIDE BY ZERO.')
574 FORMAT (' TOO MANY CHARACTERS IN A NUMBER.  LIMIT IS 38.')
575 FORMAT (' THE CHARACTER '','' FOUND IN A NUMBER.')
576 FORMAT (' BAD CHARACTER ',1H',A1,1H',' IN A NUMBER.')
577 FORMAT (' ',2A4,'=',G14.6)
578 FORMAT (' =',G14.6)
579 FORMAT (' ATTEMPT TO STORE',G14.6,' IN USER FUNCTION NAME',
   *2A4)
580 FORMAT (' ATTEMPT TO STORE',G14.6,' IN PROCEDURE NAME',2A4)
581 FORMAT (' ATTEMPT TO STORE',G14.6,' IN SYSTEM FUNCTION NAME',
   *2A4)
582 FORMAT (' NO RIGHT PAREN.  DUMP IGNORED.')
583 FORMAT (3A4,' IS NOT A VALID DUMP OPTION.')
584 FORMAT (' NO OPTION FOUND IN DUMP COMMAND.')
585 FORMAT (' MORE THAN 12 CHARACTERS IN DUMP OPTION.')
586 FORMAT (' READY ',I3)
587 FORMAT (' BEGIN COMMAND NOT ALLOWED IN PROCEDURES.')
```

```
588 FORMAT (' NO NAME FOUND IN BEGIN COMMAND.  RE-ENTER.')
589 FORMAT (' VALUE NAME ',2A4,' MAY NOT BE USED AS A PROCEDURE',
   *' NAME.')
590 FORMAT (' USER FUNCTION NAME ',2A4,' MAY NOT BE USED AS A',
   *' PROCEDURE NAME.')
591 FORMAT (' ',2A4,' IS ALREADY A PROCEDURE.')
592 FORMAT (' SYSTEM FUNCTION NAME ',2A4,' MAY NOT BE USED AS A',
   *' PROCEDURE NAME.')
593 FORMAT (' END COMMAND BEFORE BEGIN COMMAND.')
594 FORMAT (' BLANK FIELD IN END COMMAND OPTION.  ASSUME NO OPTION.')
595 FORMAT (' NO < OR > FOUND IN END COMMAND OPTION.  RE-ENTER COMMAND
   *.')
596 FORMAT (' FIRST EXPRESSION IN END COMMAND WAS NOT FOUND.  RE-ENTER
   * COMMAND.')
597 FORMAT (' RE-ENTER END COMMAND.')
598 FORMAT (' SECOND EXPRESSION IN END COMMAND WAS NOT FOUND.',
   *'  RE-ENTER COMMAND.')
599 FORMAT (' DO COMMAND NOT ALLOWED IN PROCEDURES.')
600 FORMAT (' NO PARENS FOUND IN DO COMMAND.  RE-ENTER.')
601 FORMAT (' BLANK FIELD IN DO COMMAND OPTION.  RE-ENTER COMMAND.')
602 FORMAT (' NO * FOUND IN DO COMMAND OPTION.  RE-ENTER COMMAND.')
603 FORMAT (' ',2A4,' IS A VALUE NAME.')
604 FORMAT (' ITERATION NUMBER',I10,' IS LESS THAN 1 OR GREATER',
   *' THAN 50.')
605 FORMAT (' ERROR IS IN STATEMENT ',I3,' OF PROCEDURE ',2A4)
606 FORMAT (' END OF DO. ',G13.6,A1,G13.6)
607 FORMAT (' END OF DO.  ',I2,' ITERATIONS.')
608 FORMAT (' VALUE NAME ',2A4,' MAY NOT BE USED AS A FUNCTION',
   *' NAME.')
609 FORMAT (' PROCEDURE NAME ',2A4,' MAY NOT BE USED AS A FUNCTION',
   *' NAME.')
610 FORMAT (' YOU HAVE EXCEEDED THE ROOM FOR PROCEDURES.  ',2A4,
   *' NOT STORED.')
611 FORMAT('  LINE HAS NO CONTENTS. RE-ENTER.')
612 FORMAT('  NUMBER FOUND WHERE NAME REQUIRED. RE-ENTER.')
613 FORMAT('  OPERATOR FOUND BEFORE = SIGN. RE-ENTER.')
614 FORMAT('  QUESTION MARK ILLEGAL IN CONTEXT. RE-ENTER.')
615 FORMAT('  NO RIGHT PAREN BEFORE = SIGN. RE-ENTER.')
616 FORMAT('  NO LEFT PAREN BEFORE RIGHT PAREN. RE-ENTER.')
617 FORMAT(' ENTER LOWER LIMIT, A COMMA, UPPER LIMIT, OR PRESS RETURN
   *TO CANCEL.')
618 FORMAT('  THE INTEGRAL IS  ' , G15.6 )
619 FORMAT('  NO COMMA FOUND, OR TOO MANY COMMAS FOUND.')
620 FORMAT (' ERROR IN EVALUATION OF USER FUNCTION ',2A4,'.')
621 FORMAT('  INTEGRATE COMMAND NOT ALLOWED IN PROCEDURES.')
622 FORMAT (' INTEGRAL OF ',2A4,' MAY NOT BE ACCURATE TO FIVE SIGNIFIC
   *ANT FIGURES.')
623 FORMAT (' NO LIST FOUND IN PRINT COMMAND.  COMMAND IGNORED.')
624 FORMAT (' BLANK FIELD FOUND IN PRINT COMMAND.')
625 FORMAT (' MORE THAN 9 NAMES IN PRINT COMMAND.  EXTRA NAMES',
   *' IGNORED.')
626 FORMAT (' MORE THAT 10 NAMES IN FUNCTION ARGUMENT LIST.')
627 FORMAT (' MORE THAN ONE ARGUMENT IN USER FUNCTION ',2A4,'.')
628 FORMAT (' FUNCTION ',2A4,' SHOULD CONTAIN',I3,' ARGUMENTS.')
```

```
629 FORMAT (' PROGRAM ',2A4,' LOADED.')
630 FORMAT (' PROGRAM NAME ',2A4,' MAY NOT BE USED AS A PROCEDURE NAME
   *.')
631 FORMAT (' ATTEMPT TO STORE ',G14.6,' IN PROGRAM NAME ',2A4)
632 FORMAT (' PROGRAM NAME ',2A4,' MAY NOT BE USED AS A VALUE NAME.')
633 FORMAT (' ',2A4,' IS A PROGRAM NAME.')
634 FORMAT (' PROGRAM ',2A4,' NOT LOADED.')
635 FORMAT (' ',2A4,' MAY NOT BE ERASED.  IT IS A PROGRAM NAME.')
636 FORMAT (' ',2A4,' IS A PROGRAM NAME.  USER FUNCTION NOT STORED.')
637 FORMAT (' MORE THAN 10 ARGUMENTS IN ',2A4)
638 FORMAT (' ERROR IN PROGRAM ',2A4)
639 FORMAT (' ENTER USE KEYWORD.')
640 FORMAT (' ILLEGAL.')
641 FORMAT (' LIST OF KEYWORDS.')
642 FORMAT (' NO KEYWORD IN TABLE.')
643 FORMAT (' ENTER KEYWORD YOU WISH TO SEE.')
644 FORMAT (' THIS IS A NEW KEYWORD.')
645 FORMAT (' LIST OF COMMANDS FOR ',1H',3A4,1H')
646 FORMAT (' NO COMMANDS IN TABLE FOR KEYWORD ',1H',3A4,1H')
647 FORMAT (' DO YOU WISH TO DELETE KEYWORD ',1H',3A4,2H'?)
648 FORMAT (' ARE YOU FINISHED MODIFYING?')
649 FORMAT (' ENTER COMMAND.')
650 FORMAT (' ENTER PROGRAM ENTRY POINT.')
651 FORMAT (' DO YOU WISH TO MODIFY KEYWORD ',1H',3A4,2H'?)
652 FORMAT (' MODIFICATIONS MADE.')
653 FORMAT (' OVERFLOW IN COMPUTING (',G14.6,')',A2,'(',G14.6,')')
654 FORMAT (' UNDERFLOW IN COMPUTING (',G14.6,')',A2,'(',G14.6,')')
655 FORMAT (' AN OPERATOR IS REQUIRED BETWEEN ',A2,' AND ',2A4)
656 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME.  NOT
   *THE SYSTEM FUNCTION ',2A4)
657 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME.  NOT
   *THE USER PROGRAM ',2A4)
658 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME.  NOT
   *THE VALUE ',G14.6)
659 FORMAT (' FIRST ARGUMENT FOR THE INTEGRATION FUNCTION IS NOT A NAM
   *E.')
660 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME. ',
   *2A4,' UNKNOWN.')
661 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME.  NOT
   *THE VALUE NAME ',2A4)
662 FORMAT (' INTEGRATION FUNCTION EXPECTS A USER FUNCTION NAME.  NOT
   *THE PROCEDURE NAME ',2A4)
663 FORMAT (' INTEGRATION FUNCTION WITH USER FUNCTION ',2A4,' DOES NOT
   * CONTAIN THREE ARGUMENTS.')
664 FORMAT (' AN OPERATOR IS REQUIRED BETWEEN ',A2,' AND ',G14.6)
665 FORMAT (' USER FUNCTION ',2A4,' MAY NOT APPEAR IN THE INT FUNCTION
   *.  IT CONTAINS THE INT FUNCTION.')
    END
```

```
.
.
.
.
         ADCOND
GNCL#P   PSECT
         ENTRY LOAD
         ENTRY RUNIT
         ENTRY LOADED
SAVCAL   DC F'76'     LENGTH OF SAVE AREA
         DC 18F'0'        ZERO OUT SAVE AREA EXCEPT FOR LENGTH
RTCHAR   DS CL8    NAME OF ROUTINE TO BE LOADED OR CALLED
BCDNAM   DS F        ADDRESS OF THE NAME OF THE ROUTINE
NPARM    DS F        ADDRESS OF NUMBER OF ARGUMENTS TO BE PASSED TO FUNC
ARAPAR   DS F        ADDRESS OF THE FUNCTION ARGUMENTS
ANS      DS F        ADDRESS FOR RESULT OF FUNCTION CALL
ERRCOD   DS F        ADDRESS OF ARGUMENT FOR ERRCOD
EIGHT    DC F'8'
BLANK    DC X'40C0C0000'
CHRMSK   DC X'FF0C0000'
PARLST   DS 20F      PARAMETER LIST TO BE BUILT FOR CALLED FUNCTION
HSHIN1   DS F        TEMP LOCS FOR HASH RESULTS
HSHIN2   DS F        TEMP LOCS FOR HASH RESULTS
HSHIN3   DS F        HASH SWITCH INDICATES WHICH TABLE LAST SEARCHED
BLANKS   DS F
MASPOS   DC X'0FFFFFFF'  MASK FOR HASH TO PREVENT OVERFLOW AND SIGNS
ONE      DC F'1'    RETURN CODE AND CONSTANT
TWO      DC F'2'    RETURN CODE FOR ERROR
HSHDIV   DC F'128'    HASH DIVISOR ALSO LENGTH OF MAIN ADCON TABLE
FOURF    DC F'4'    INCREMENTS FOR TABLE SEARCHES
SIXF     DC F'24'   ENTRY SIZE FOR ADCON TABLES
LASTCL   DS F   NAME OF LAST ROUTINE LOADED OR CALLED
LASTC1   DS F        CONTINUATION OF NAME
LADPT    DS F        INDEX POINTER TO ADCON FOR LAST ROUTINE CALLED
LADPT1   DS F   SWITCH TO INDICATE MAIN ADCON TABLE OR OVERFLOW TABLE
OVFCPN   DC F'0'    NEXT AVAILABLE ENTRY LOC IN OVERFLOW ADCON TABLE
TABFUL   DC F'4080'  OVERFLOW TABLE FULL TEST CONSTANT
BASADT   DC A(ADTABL)  BASE ADDRESS OF ADTABLE
BASOAD   DC A(OVADTB)  BASE ADDRESS OF OVERFLOW ADCON TABLE
ADREND   DC A(ENDPST)      ADRESS FOR END OF PSECT NEEDED FOR BASE
BASADR   DC A(LOAD)    BASE ADRESS FOR CSECT COVER
ENTPT    DS F     ENTRY POINT INDICATOR USED FOR RETURNS OF INTER SUBS
MAXARG   DC F'20'  MAXIMUM NUMBER OF ARGUMENTS ALLOWED
ADTABL   DS 5F     MAIN ADCON TABLE  128 ENTRIES
PTOVF    DS F    POINTER TO OVERFLOW ADCON TABLE ENTRY
TABADN   DS 762F    6 WORDS PER ENTRY
ENDPST   DS F    END OF PSECT
OVF#AD   COM
OVADTB   DS F    OVERFLOW ADCON TABLE
NXCHPT   DS F   POINTER TO NEXT ADCON IN OVERFLOW ADCON CHAIN
TABOVF   DS 1018F    OVERFLOW ADCON TABLE IS ONE PAGE LONG
GNCL#C   CSECT READONLY,PUBLIC
         USING LOAD,15
LOAD     SAVE (14,12)
```

```
         L 14,72(C,13)  LOAD 14 WITH OUT PSECT
         ST 14,8(C,13)  SAVE IT BACK IN CALLING PROGRAM
         ST 13,4(0,14)  SAVE CALLERS PSECT PT IN OUR SAVE AREA
         LR 13,14  SWITCH 13 TO OUR PSECT FOR BASE
         USING GNCL#P,13
         LR 12,15  SWITCH TO 12 FOR CSECT BASE REG
         DROP 15
         USING LOAD,12
         SR 10,10
         ST 10,ENTPT
         L 2,0(0,1)  PICK UP ADDRESS OF FIRST ARG
         ST 2,BCDNAM  SAVE ADDRESS
         L 3,4(0,1)  PICK UP ADDRESS OF SECOND ARG
         ST 3,ERRCOD  SAVE ADDRESS
         LM 4,5,0(2)  LOAD 4 AND 5 WITH NAME OF FUNC TO BE LOADED
         B ADJNAM  MAKE RIGHT ADJUSTED NAME LEFT ADJUSTED
CHRLOD   STM 4,5,RTCHAR   SAVE LEFT ADJUSTED NAME
         C 4,LASTCL  CHECK TO SEE IF NAME IS THE  SAME AS LAST TIME
         BNE HASH
         C 5,LASTC1
         BE RET1   RETURN, NOTHING NEED BE DONE
HASH     LR 6,4
         LR 7,5    THE FOLLOWING SECTION OF CODING IS AN INTERNAL
         L 9,MASPCS   SUBROUTINE TO CALCULATE THE HASH CODE FROM THE
         NR 6,9    NAME OF THE ROUTINE TO BE LOADED OR CALLED
         NR 7,9
         MR 6,6  THE HASH ADDRESS IS USED TO FIND THE ENTRY IN THE
         NR 7,9   INSURE NO MINUS SIGNS
         AR 6,7    MAIN ADCON TABLE   ENTPT IS USED TO RETURN TO
         SRDA 6,32(0)   DIFFERENT SECTIONS CORRESPONDING TO THE
         D 6,HSHDIV    ENTRY POINT CALLED
         LR 7,6   PUT REMAINDER IN REG 7
         SR 8,8    INDICATE MAIN ADCON TABLE LAST USED
         ST 8,HSHIN3
         LR 9,7   7 CONTAINS THE BASIC HASH INDEX NOW
         M 8,SIXF    NOW COMPUTE INDEX FOR ADCON TABLES
         ST 7,HSHIN1      SAVE  HASH INDICES
         ST 9,HSHIN2   EACH ENTRY IN THE ADCON TABLES IS SIX LONG
         L 2,BASADT  SET UP BASE REG FOR MAIN ADCON TABLE  ADTABL
         USING ADTABL,2
         L 8,ENTPT  NOW RETURN TO APPROPRIATE SECTION OF CODING
         C 8,ONE   CORRESPONDING TO ENTRY POINT CALLED
         BL RETLOC   LOAD WAS ENTRY POINT
         BE RETLOD  LOADED WAS ENTRY POINT
         B RETCAL   RUNIT WAS THE ENTRY POINT
RETLOD   LR 10,2
         AR 10,9   SETUP BASE FOR ADCON DSECT
         USING CHAADC,10
         LM 6,7,ADCPNAM    PICK UP NAME FIELD IN ADCON
         C 6,BLANKS   CHECK IF THE ADCON HAS NOT BEEN ARMED OR USED
         BNE CHKNAM
         C 7,BLANKS
         BE LODE  ADCON AVAILABLE FOR USE GO LOAD
CHKNAM   CR 4,6  CHECK ADCON NAME  AGAINST ENTRY NAME FOR MATCH
```

```
         BNE CHEOVF   RETURN IF MATCH IS FOUND OTHERWISE CHECK ADCON
         CR 5,7   OVERFLOW TABLE FOR MATCH
         BE RET3   MATCH FOUND  UPDATE LASTCL AND RETURN
CHEOVF   L 6,20(0,10)  PICK UP OVFTAB POINTER
         L 11,BASCAD   ESTABLISH BASE REG FOR OVERFLOW ADCON TABLE
         USING OVADTB,11
         SR 7,7  IF OVERFLOW TABLE POINTER IS ZERO, THERE ISNT ANY
         CR 6,7  OVERFLOW CHAIN YET. THIS WILL BE FIRST ENTRY FOR
         BE LODE1  OVERFLOW ADCON CHAIN
SRCVTB   LR 10,11 SUBROUTINE TO SEARCH OVERFLOW ADCON TABLE FOR MATCH
         AR 10,6   RECALCULATE BASE FOR ADCON GROUP
         ST 6,HSHIN2  SAVE INDEX POINTER
         ST 10,HSHIN3  INDICATE OVERFLOW ADCON TABLE LAST USED
         LM 6,7,ADCPNAM CHECK NAME OF ADCON FOR MATCH WITH INPUT
         CR 4,6
         BNE LP1
         CR 5,7
         BE RET3  MATCH FOUND RETURN
LP1      L 6,20(0,10)   PICK UP NEXT ADCON  ON CHAIN
         SR 7,7   IF THE CHAIN POINTER IS ZERO, WE HAVE SEARCHED THRU
         CR 6,7 THE OVERFLOW ADCON CHAIN
         BNE SRCVTB    NOT AT END OF CHAIN  CONTINUE SEARCH
         L 8,ENTPT  MATCH NOT FOUND IN CHAIN  RETURN
         C 8,ONE   ENTPT IS USED TO FIGURE POINT OF RETURN
         BL LODE1   ENTRY POINT WAS LOAD
         BE SRRET1   ENTRY POINT WAS LOADED
         B SRRET2     ENTRY POINT WAS RUNIT
LODE1    L 6,OVFCPN   PICK UP POINTER FOR NEXT AVAILABLE ADCON
         ST 6,20(0,10)  SET ADCON POINTER TO UPDATE CHAIN
         ST 6,HSHIN2
         LR 10,11  SWITCH TO OVERFLOW ADCON TABLE
         AR 10,6   UPDATE ADCON POINTER
         ST 10,HSHIN3  INDICATE OVERFLOW ADCON TABLE LAST USED
         LR 7,10
         A 6,SIXF
         ST 6,OVFCPN
         C 6,TABFUL  CHECK TO INSURE WE HAVENT EXHAUSTED OVERFLOW
         BH CATAST  ADCON TABLE   GO TO CATAST IF WE HAVE
LODE2    L 9,ADREND  SETUP BASE TO GET SVC FOR ARM MACRO
         USING ENDPST,9
         ARM (7),RTCHAR
         DROP 9
         LR 10,7    THE  ADCON  IS ARMED
         MVI ADCC1L,X'00'      SET ADCON FOR LOAD AND ERRCD=CODE OPTS
         MVI ADCC2L,X'01'
         LR 8,1   SAVE REG 1
         LR 1,10   SET R1 TO ADCON LOCATION
         LOAD EPLOC=(1)
         LR 1,8   THE REQUESTED FUNC IS LOADED   RESTORE REG 1
         TM ADCC2L,X'06'   CHECK FOR ERROR IN LOADING
         BNE ERROR   OOPS WE BRAGGED TO SCON ROUTINE NOT LOADED
         B RET4
RET3     L 8,ENTPT
         C 8,TWO
```

```
         BE CALL   DONT RETURN YET IF ENTRY WAS RUNIT
RET4     ST 4,LASTCL   UPDATE THE NAME AND ADCON INDEX FOR LAST
         ST 5,LASTC1   CALLED OR LOADED ROUTINE
         L 8,HSHIN2
         ST 8,LADPT
         L 8,HSHIN3
         ST 8,LADPT1
RET1     L 8,ONE   SET ERRCOD TO INDICATE ALL  IS OKAY
         ST 8,0(0,3)
RET5     L 14,4(0,13)  PICK UP CALLERS PSECT AND RETURN
         LR 13,14    RESTORE REG 13 TO CALLERS PSECT
         SR 15,15  SET RETURN CODE FOR FORTRAN
         RETURN (14,12),RC=15
ADJNAM   L 7,EIGHT TAKE THE 8 CHARACTER INPUT NAME WHICH IS RIGHT
         L 11,BLANK    ADJUSTED AND MAKE IT LEFT ADJUSTED
         SRL 11,24(0)   THIS IS AN INTERNAL SUBROUTINE LIKE HASH
         L 10,BLANK    AND SRCVTB  IT ALSO USES ENTPT TO RETURN
         L 8,CHRMSK
NDONE    LR 9,4    BASIC LOOP TO CHECK FOR BLANKS AND SHIFT
         NR 9,8    INPUT NAME IS IN REG 4 AND 5 OUTPUT NAME WILL
         CLR 9,10    WIND UP IN REG 4 AND 5
         BNE DONE   GO TO DONE IF NON BLANK CHARACTER IS FOUND
         SLDL 4,8(0)
         OR 5,11    ADD A BLANK TO THE RIGHT END
         BCT 7,NDONE   CHECK TO INSURE AGAINST AN INDEFINITE LOOP
         B RET2    NAME WAS ALL BLANKS  RETURN ERROR CODE
DONE     L 8,ENTPT
         C 8,ONE   CHECK FOR WHICH RETURN TO TAKE
         BL CHRLOD
         BE CHRLOD
         B CHRRUN
RET2     L 8,TWO    SET ERCOD INDICATER FOR TROUBLE AND RETURN
         ST 8,0(0,3)
         B RET5
LODE     LR 7,10
         B LODE2
         USING LOADED,15
LOADED   SAVE (14,12)
         L 14,72(0,13) LOAD R14 WITH  OUR PSECT
         ST 14,8(0,13)    BACKWARD PSECT POINTER
         ST 13,4(0,14)  FORWARD PSECT POINTER
         LR 13,14
         USING GNCL#P,13
         L 12,BASADR   SWITCH TO R12 FOR OUR CSECT BASE REG
         DROP 15
         USING LOAD,12
         L 10,ONE  SET ENTPT TO INDICATE LOADED WAS ENTRY POINT
         ST 10,ENTPT   LOADED CHECKS ADCON TABLES TO FIND IF A FUNC
         L 2,0(0,1)   HAS BEEN PREVIOUSLY LOADED   PICK UP PARAM LIST
         ST 2,BCDNAM  SAVE ADDRESS OF FIRST ARG
         L 3,4(0,1)
         ST 3,ERRCOD    SAVE ADDRESS OF SECOND ARGUMENT
         LM 4,5,0(2)   PICK UP FUNCTION NAME
         B ADJNAM  MAKE RIGHT ADJUSTED NAME LEFT ADJUSTED
```

```
CHRLDD   STM 4,5,RTCHAR   SAVE LEFT ADJUSTED NAME
         C 4,LASTCL   USED
         BNE CONCK1
         C 5,LASTC1
         BE RET1     MATCH FOUND  RETURN
CONCK1   B HASH  MATCH NOT FOUND HASH NAME   REG 9 HAS ADCON  INDEX
RETLDD   LR 10,2  SET REG 2 AND 10 AS BASE FOR MAIN ADCON TABLE
         AR 10,9  SET UP BASE FOR ADCON DSECT
         USING CHAADC,10
         LM 6,7,ADCPNAM  CHECK FOR MATCH OF ADCON NAME  WITH INPUT
         CR 4,6
         BNE CONCK2
         CR 5,7
         BE RET4
CONCK2   L 6,20(0,10)   MATCH NOT FOUND  SEARCH ADCON CHAIN
         SR 7,7
         CR 6,7  IF POINTER TO OVERFLOW TABLE IS ZERO,THERE IS NO CHAIN
         BE RET2 END OF CHAIN, ROUTINE NOT FOUND OR LOADED
         L 11,BASOAD  SET UP BASE REG FOR OVERFLOW ADCON TABLE
         USING OVADTB,11
         B SROVTB  SEARCH OVERFLOW ADCON TABLE FOR MATCH
SRRET1   B RET2  MATCH NOT FOUND  ROUTINE  NOT LOADED
         USING RUNIT,15
RUNIT    SAVE (14,12)
         L 14,72(0,13) LOAD REG 14 WITH OUR PSECT
         ST 14,8(0,13)  STORE BACKWARD POINTER
         ST 13,4(0,14)     STORE FORWARD POINTER
         LR 13,14   LOAD REG 13 WITH OUR PSECT AND ESTABLISH BASE REG
         USING GNCL#P,13
         L 12,BASADR   SWITCH TO R12 FOR OUR CSECT BASE REG  .
         DROP 15
         USING LOAD,12
         L 10,TWO  SET ENTPT TO INDICATE RUNIT WAS ENTRY POINT
         ST 10,ENTPT
         L 2,0(0,1)   PICK UP ARGUMENTS
         ST 2,BCDNAM  SAVE ADDRESS OF FIRST ARGUMENT
         L 3,16(0,1)  PICK UP FIFTH ARG ADDRESS
         ST 3,ERRCOD  SAVE ADDRESS OF FIFTH ARG
         L 6,12(0,1)
         ST 6,ANS  SAVE ADDRESS OF FOURTH ARG
         L 4,8(0,1)
         L 5,4(0,1)
         ST 4,ARAPAR  SAVE ADDRESSES OF SECOND AND THIRD ARGS
         ST 5,NPARM   BUILD A CALLING PARAM LIST BASED ON THE ADDRESS
         SR 6,6   OF THE ARGUMENT ARRAY WHICH IS THE SECOND ARG
         SR 10,10  INITIALIZE  FOR LOOP
         L 7,FOURF  INCREMENT FOR FULL WORD
         L 8,ONE   INCREMENT  TO COUNT ARGUMENTS
         L 9,0(0,5)    LOAD 9 WITH THE TOTAL NUMBER OF ARGS
         C 9,MAXARG   CHECK TO SEE THERE ARE NOT MORE THAN ARE
         BH RET2   PROVIDED  FOR
PARLOP   AR 6,8   LOOP TO BUILD PARAM LIST
         ST 4,PARLST(10)
         AR 10,7  ADD FULL WORD INCR ONTO PARLST INDEX
```

```
          AR 4,7   ADD FULL WORD INCR ONTO ADDRESS OF ARAARG
          CR 6,9  CHECK FOR COMPLETED PARAMETER ADDRESS LIST
          BL PARLOP     NOT COMPLETED
          LM 4,5,0(2) COMPLETE,PICKUP NAME OF FUNC TO BE CALLED
          B ADJNAM  MAKE RIGHT ADJUSTED NAME LEFT ADJUSTED
CHRRUN    STM 4,5,RTCHAR   SAVE LEFT ADJUSTED NAME
          C 4,LASTCL  CHECK INPUT NAME AGAINST NAME OF FUNC LAST CALLED
          BNE CONCK3
          C 5,LASTC1
          BE CAL   GO TO CALL THE FUNC
CONCK3    B HASH  HASH THE NAME AND GET PTS TO MAIN ADCON TABLE ENTRY
RETCAL    LR 10,2  SET UP BASE FOR ADCON DSECT
          AR 10,9
          USING CHAADC,10
          LM 6,7,ADCPNAM   CHECK TO SEE IF ADCON NAME MATCHES INPUT
          CR 4,6
          BNE CONCK4
          CR 5,7
          BE CAL1   YES, MATCH FOUND
CONCK4    L 6,20(0,10)  NO,PICK UP OVERFLOW ADCON TABLE POINTER
          SR 7,7
          CR 6,7
          BE RET2  END OF ADCON CHAIN AND NO MATCH  ERRCOD =2
          L 11,BASCAD  SETUP BASE FOR OVERFLOW ADCON TABLE
          USING OVADTB,11
          B SROVTB  SEARCH OVERFLOW ADCON TABLE FOR MATCH
SRRET2    B RET2   MATCH NOT FOUND SET ERRCOD = 2
CAL       L 10,LADPT
          ST 10,HSHIN2
          SR 6,6
        C 6,LADPT1  CHECK TO SEE WHICH ADCON TABLE TO USE
          BNE OVABAS
          A 10,BASADT   CALCULATE ADDRESS OF ADCON
          B CAL1   GO CALL  FUNC
OVABAS    A 10,BASCAD   CALCULATE ADDRESS OF ADCON
CAL1      LR 8,1   SAVE REG1 IN REG8
          LA 1,PARLST   SETUP ADDRES OF PARAM LIST IN REG 1
          LR 15,10  SET REG 15 TO LOC OF ADCON FOR CALL
          USING CHAADC,10
          MVI ADCC1C,X'01'  SET ADCON FOR CALL AND ERRCD=CODE OPTS
          MVI ADCC2C,X'01'
          CALL (15),,,E
          LR 1,8 CALL COMPLETE RESTORE REG 1
          TM ADCC2C,X'06'   CHECK FOR ERROR IN CALL
          BNE TRYLOD
CALOK     L 6,ANS   SAVE POSSIBLE FUNCTION RESULTS
          STE 0,0(0,6)
          B RET4     RETURN
TRYLOD    LR 7,10
          L 9,ADREND  SETUP BASE TO PICKUP SVC FOR ARM
          USING ENDPST,9
          ARM (7),RTCHAR
          DROP 9
          LR 8,1
```

```
         LA 1,PARLST  PICK UP ADDRESS OF PARAM LIST FOR CALL
         LR 15,10   POINT TO ADCON FOR CALL NOW FULLY ARMED
         USING CHAADC,10
         MVI ADCC1C,X'01'  SET ADCON FOR CALL AND ERRCD=CODE OPTS
         MVI ADCC2C,X'01'
         CALL (15),,,E
         LR 1,8   RESTORE REG 1
         TM ADCC2C,X'06'  CHECK FOR ERROR IN CALL
         BNE ERROR   NOTHING MORE WE CAN DO   PMD DOESNT EXIST
         B CALOK
CATAST   L 10,BASCAD   PT TO OVERFLOW ADCON TABLE  SET UP TO CLEAR
         L 6,FOURF   SET UP FULL WORD INCREMENT
         SR 7,7   CLEAR INDEX
         SR 8,8
CLEAR1   ST 8,0(7,10)   CLEAR OVERFLOW ADCON TABLE
         AR 7,6    INCREMENT INDEX BY A FULL WORD
         C 7,TABFUL   SEE IF TABLE  ALL CLEARED
         BL CLEAR1  NOT CLEARED YET
         ST 8,OVFCPN   CLEAR AVAILABLE OVERFLOW ENTRY POINTER
         L 10,PTOVF   SET BASE TO CLEAR OVERFLOW TABLE PTS IN MAIN
         SR 7,7       ADCON TABLE
         L 11,ONE  INCREMENT  TO COUNT ADCON PTS CLEARED
         SR 9,9
         L 6,SIXF  INCREMENT BETWEEN PTS TO OVERFLOW ADCON TABLE
CLEAR2   ST 8,0(7,10)  CLEAR PT TO OVERFLOW ADCON TABLE
         AR 7,6
         AR 9,11
         C 9,HSHDIV   ARE WE ALL DONE CLEARING
         BL CLEAR2  NO WE ARE NOT DONE YET
         B HASH
ERROR    SR 8,8 SET ERRCOD = 2 AND RETURN
         ST 8,0(0,10)       CLEAR ADCONS
         ST 8,4(0,10)
         ST 8,8(0,10)
         ST 8,12(0,10)
         ST 8,16(0,10)
         ST 8,20(0,10)
         B RET2
         END
.
.                                                          ISRL, ISLL, etc.
.
.
         TITLE '       THE SHIFT FUNCTIONS FOR TIME SHARING  '
*
PSECT    PSECT
SAVE     DS    F
SAVE2    DS    F
MASK     DC    X'FFFFFFFF'
         ENTRY SRA,ISRA,ARS,IARS
         ENTRY SLA,ISLA,ALS,IALS
         ENTRY SRL,ISRL
         ENTRY SLL,ISLL
         ENTRY SRDA,ISRDA,LRS,ALRS
```

```
         ENTRY SLDA,ISLDA,LLS,ALLS
         ENTRY SRDL,ISRDL,LGR,ALGR
         ENTRY SLDL,ISLDL,LGL,ALGL
         ENTRY AND,LAND,OR,LOR,EXOR,LEXOR,LCOMP,COMPL
         ENTRY HSRA,IHSRA,HSLA,IHSLA
SHIFTY   CSECT READONLY,PUBLIC
ARS      EQU   *
IARS     EQU   *
ISRA     EQU   *
SRA      SAVE  (4,8)
         USING PSECT,8
         L     8,72(0,13)     LOAD REGISTER 8.
         LM    4,5,0(1)       4 HAS THE LOCATION OF THE SHIFT COUNT
*                             5 HAS THE LOCATION OF THE DATA
         L     7,0(4)         7 IS LOADED WITH THE SHIFT COUNT.
         L     0,0(5)         0 IS LOADED WITH THE DATA
         SRA   0,0(7)         SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         LE    0,SAVE         PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         SPACE 6
IALS     EQU   *
ALS      EQU   *
ISLA     EQU   *
SLA      SAVE  (4,8)
         L     8,72(0,13)     LOAD REGISTER 8.
         LM    4,5,0(1)       4 HAS THE LOCATION OF THE SHIFT COUNT
*                             5 HAS THE LOCATION OF THE DATA
         L     7,0(4)         7 IS LOADED WITH THE SHIFT COUNT.
         L     0,0(5)         0 IS LOADED WITH THE DATA
         SLA   0,0(7)         SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         LE    0,SAVE         PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         EJECT
ISRL     EQU   *
SRL      SAVE  (4,8)
         L     8,72(0,13)     LOAD REGISTER 8.
         LM    4,5,0(1)       4 HAS THE LOCATION OF THE SHIFT COUNT
*                             5 HAS THE LOCATION OF THE DATA
         L     7,0(4)         7 IS LOADED WITH THE SHIFT COUNT
         L     0,0(5)         0 IS LOADED WITH THE DATA
         SRL   0,0(7)         SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         LE    0,SAVE         PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         SPACE 6
ISLL     EQU   *
SLL      SAVE  (4,8)
         L     8,72(0,13)     LOAD REGISTER 8.
         LM    4,5,0(1)       4 HAS THE LOCATION OF THE SHIFT COUNT
*                             5 HAS THE LOCATION OF THE DATA
         L     7,0(4)         7 IS LOADED WITH THE SHIFT COUNT
         L     0,0(5)         0 IS LOADED WITH THE DATA
```

```
         SLL   0,0(7)        SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         LE    0,SAVE       PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         EJECT
ALRS     EQU   *
LRS      EQU   *
ISRDA    EQU   *
SRDA     SAVE  (4,8)
         L     8,72(0,13)   LOAD REGISTER 8.
         LM    4,5,0(1)     4 HAS THE LOCATION OF THE SHIFT COUNT
*                           5 HAS THE LOCATION OF THE DATA
         L     7,0(4)       7 IS LOADED WITH THE SHIFT COUNT.
         L     0,0(5)       0 IS LOADED WITH THE DATA
         L     1,SAVE2
         SRDA  0,0(7)       SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         ST    1,SAVE2
         LE    0,SAVE       PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         SPACE 6
ALLS     EQU   *
LLS      EQU   *
ISLDA    EQU   *
SLDA     SAVE  (4,8)
         L     8,72(0,13)   LOAD REGISTER 8.
         LM    4,5,0(1)     4 HAS THE LOCATION OF THE SHIFT COUNT
*                           5 HAS THE LOCATION OF THE DATA
         L     7,0(4)       7 IS LOADED WITH THE SHIFT COUNT
         L     0,0(5)       0 IS LOADED WITH THE DATA
         L     1,SAVE2
         SLDA  0,0(7)       SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         ST    1,SAVE2
         LE    0,SAVE       PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         EJECT
ALGR     EQU   *
LGR      EQU   *
ISRDL    EQU   *
SRDL     SAVE  (4,8)
         L     8,72(0,13)   LOAD REGISTER 8.
         LM    4,5,0(1)     4 HAS THE LOCATION OF THE SHIFT COUNT
*                           5 HAS THE LOCATION OF THE DATA
         L     7,0(4)       7 IS LOADED WITH THE SHIFT COUNT
         L     0,0(5)       0 IS LOADED WITH THE DATA
         L     1,SAVE2
         SRDL  0,0(7)       SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,SAVE
         ST    1,SAVE2
         LE    0,SAVE       PUT IN THE FLOATING POINT REGISTER.
         RETURN (4,8)
         SPACE 6
ALGL     EQU   *
```

```
LGL     EQU   *
ISLDL   EQU   *
SLDL    SAVE  (4,8)
        L     8,72(0,13)     LOAD REGISTER 8.
        LM    4,5,0(1)       4 HAS THE LOCATION OF THE SHIFT COUNT
*                            5 HAS THE LOCATION OF THE DATA
        L     7,0(4)         7 IS LOADED WITH THE SHIFT COUNT
        L     0,0(5)         0 IS LOADED WITH THE DATA
        L     1,SAVE2
        SLDL  0,0(7)         SHIFT     SHIFT     SHIFT     SHIFT
        ST    0,SAVE
        ST    1,SAVE2
        LE    0,SAVE         PUT IN THE FLOATING POINT REGISTER.
        RETURN (4,8)
        TITLE '      AND, OR,EXCLUSIVE OR AND COMPLEMENT '
AND     EQU   *
LAND    SAVE  (4,8)
        L     8,72(0,13)     LOAD REG 8 WITH PSECT LOCATION
        LM    4,5,0(1)
        L     0,0(4)         LOAD FIRST ARG INTO REG 0.
        N     0,0(5)         AND WITH 2ND ARG
        ST    0,SAVE
        LE    0,SAVE         PUT IN FLOATING PT. REGISTERS
        RETURN (4,8)
        SPACE 6
OR      EQU   *
LOR     SAVE  (4,8)
        L     8,72(0,13)     LOAD REG 8 WITH PSECT LOCATION
        LM    4,5,0(1)
        L     0,0(4)         LOAD FIRST ARG INTO REG 0.
        O     0,0(5)         OR  WITH 2ND ARG
        ST    0,SAVE
        LE    0,SAVE         PUT IN FLOATING PT. REGISTERS
        RETURN (4,8)
        EJECT
EXOR    EQU   *
LEXOR   SAVE  (4,8)
        L     8,72(0,13)     LOAD REG 8 WITH PSECT LOCATION
        LM    4,5,0(1)
        L     0,0(4)         LOAD FIRST ARG INTO REG 0.
        X     0,0(5)         EXCLUSIVE OR WITH 2ND ARG
        ST    0,SAVE
        LE    0,SAVE         PUT IN FLOATING PT. REGISTERS
        RETURN (4,8)
        SPACE 6
COMPL   EQU   *
LCOMP   SAVE  (4,8)
        L     8,72(0,13)     LOAD REG 8 WITH PSECT LOCATION
        LM    4,5,0(1)
        L     0,0(4)         LOAD FIRST ARG INTO REG 0.
        X     0,MASK         COMPLEMENT ALL BITS
        ST    0,SAVE
        LE    0,SAVE         PUT IN FLOATING PT. REGISTERS
        RETURN (4,8)
```

```
HSRA     EQU   *
IHSRA    SAVE  (4,7)
         LM    4,5,0(1)
         L     7,0(4)          SHIFT COUNT
         LH    0,0(5)          DATA
         SRA   0,0(7)
         ST    0,52(0,13)      R8 SAVE AREA
         ST    1,56(0,13)      R9 SAVE AREA
         LE    0,52(0,13)
D1       RETURN (4,7)
HSLA     EQU   *
IHSLA    SAVE  (4,7)
         LM    4,5,0(1)
         L     7,0(4)          SHIFT COUNT
         LH    0,0(5)          DATA
         SLA   0,0(7)          SHIFT     SHIFT     SHIFT     SHIFT
         ST    0,52(0,13)      R8 SAVE AREA
         ST    1,56(0,13)      R9 SAVE AREA
         LE    0,52(0,13)
D2       RETURN (4,7)
         END
```

REFERENCES

1. Swigert, Paul: COMPUTE - A Time-Sharing Desk Calculator Program. NASA TN D-4917, 1968.
2. Anon.: IBM System/360 Time Sharing System - Concepts and Facilities. IBM Publication C28-2003-2, Oct. 1967.
3. Anon.: IBM System/360 Time Sharing System - FORTRAN Programmer's Guide. IBM Publication C28-2025-1, Mar. 1968.
4. Anon.: IBM System/360 Time Sharing System - Command System User's Guide. IBM Publication C28-2001-3, Nov. 1968.
5. Anon.: IBM System/360 Time Sharing System/360 - IBM FORTRAN IV. IBM Publication C23-2007-1, Oct. 1967.
6. Anon.: IBM System/360 Time Sharing System - Assembler Language. IBM Publication C28-2000-0, 1966.
7. Canright, R. Bruce, Jr.: A Conversational Approach to Matrix Calculations - Concepts and Implementation. NASA TM X-1749, 1969.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546
OFFICIAL BUSINESS

FIRST CLASS MAIL

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AI
SPACE ADMINISTRATION

POSTMASTER: If Undeliverable (Section 158
Postal Manual) Do Not Retu

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

— NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Notes, and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546