# GENPRO-2 MNGR MODULE REVIEW OUTLINE

NAME

MNGR

PURPOSE

MNGR reads data card packets containing control information necessary for overall processor control and for individual operation control. MNGR also generates files containing control information for these individual operations and initializes and tests arrays and pointers for the processor.

ACCESS CARDS

```
*VOLUME,NUNIT,VSN=P04290,STAGEIN=RT,CONV=TB,DS=600,
         STAGEOUT=ZT

FETCH,S=NUNIT,SN=MNGR
```

(NOTE: NUNIT is the logical unit number assigned to the volume. Also, IFTRAN control cards are required.)

USAGE

CALL MNGR

This module requires that certain .REPL cards be pre-specified. (See DRIVER Module review outline.) In addition: .REPL/$NPAR/---/, the total number of parameters for this flight. The dimension of MLIST.

.REPL/$EDIT1/---/, the keyword used for insert-type editing done on the order list in the data packet for a given operation.

.REPL/$EDIT2/---/, the keyword used for replace-type editing...

.REPL/$EDIT3/---/. the keyword used for Delete-type editing...

COMMON BLOCK LINKAGES

This module requires that certain .SAVE Blocks be specified. (see DRIVER Module Review Outline.) In addition: .SAVE OPFLWT, .SAVE OPFLRD, .SAVE WRFIL.

## DESCRIPTION

INTRODUCTION

As mentioned previously, MNGR reads data card packets containing control information necessary for overall processor control and for individual operation control. All data cards are read with a free-form input routine.

OPERATIONS

Operations are of two types: Transformation and snapshot (T-OP and S-OP). Transformation Operations actually change and store the data processed by them. Examples of Transformation-type Operations: Input, Calibration, Filtering,... Snapshot Operations however, leave all data unchanged. Examples of Snapshot Operations: Plotting, Printing, Tape Writing, Statistics,...

CONTROLS

Control Information is categorized in two types: General controls and parameter-linked controls. Parameter-linked controls are of two types: Standard and Non-Standard.

Both the DRIVER and the OPERATION modules have general control data packets. General controls specify how processing is to be done overall. For example, General Controls might tell a module: when to begin processing data, how many cycles of data to process at a time, how many cycles of overlap are required, whether or not this is a transformation operation or a snap-shot operation, how many parameters to process,...

Only operation modules require parameter-linked control data packets. P-L controls tell the module how to process each parameter individually.

All parameters being processed by a given operation module have the same number of standard parameter-linked controls. For example, in an INPUT operation module some standard P-L controls might be: where in the input frame to pick up samples for a parameter, how to decode this parameter,..... However, some modules require more information about some parameters than others. For example, in a CALIBRATION Operation, some parameters are source parameters (i.e. parameters with input rates to this operation). And some parameters are referred to as derived (i.e. mathematical operations are applied to one or more source parameters. The result of this transformation is a derived parameter.) It is necessary to specify which source parameters are needed to derive this new parameter. Thus, additional information is needed for derived parameters. This additional information is referred to as non-standard P-L CONTROLS.

The DRIVER module and all OPERATION modules require that certain general controls be specified. All OPERATION modules require that certain parameter-linked controls be specified. For a description of the data specification requirements for a module see the review outline for that module.

DATA DECK FORMAT

The DATA DECK is made up of several sets of data cards. Each set of cards has a TERMINATOR CARD. There are four such END-CARDS: ENDGEN, ENDOP, ENDFLT, ENDPROC. Each end-card has a specific function.

ENDGEN - General control information for the DRIVER and for OPERATION modules is terminated by the ENDGEN card.

ENDOP - Parameter-Linked Control information for OPERATION modules is terminated by the ENDOP card.

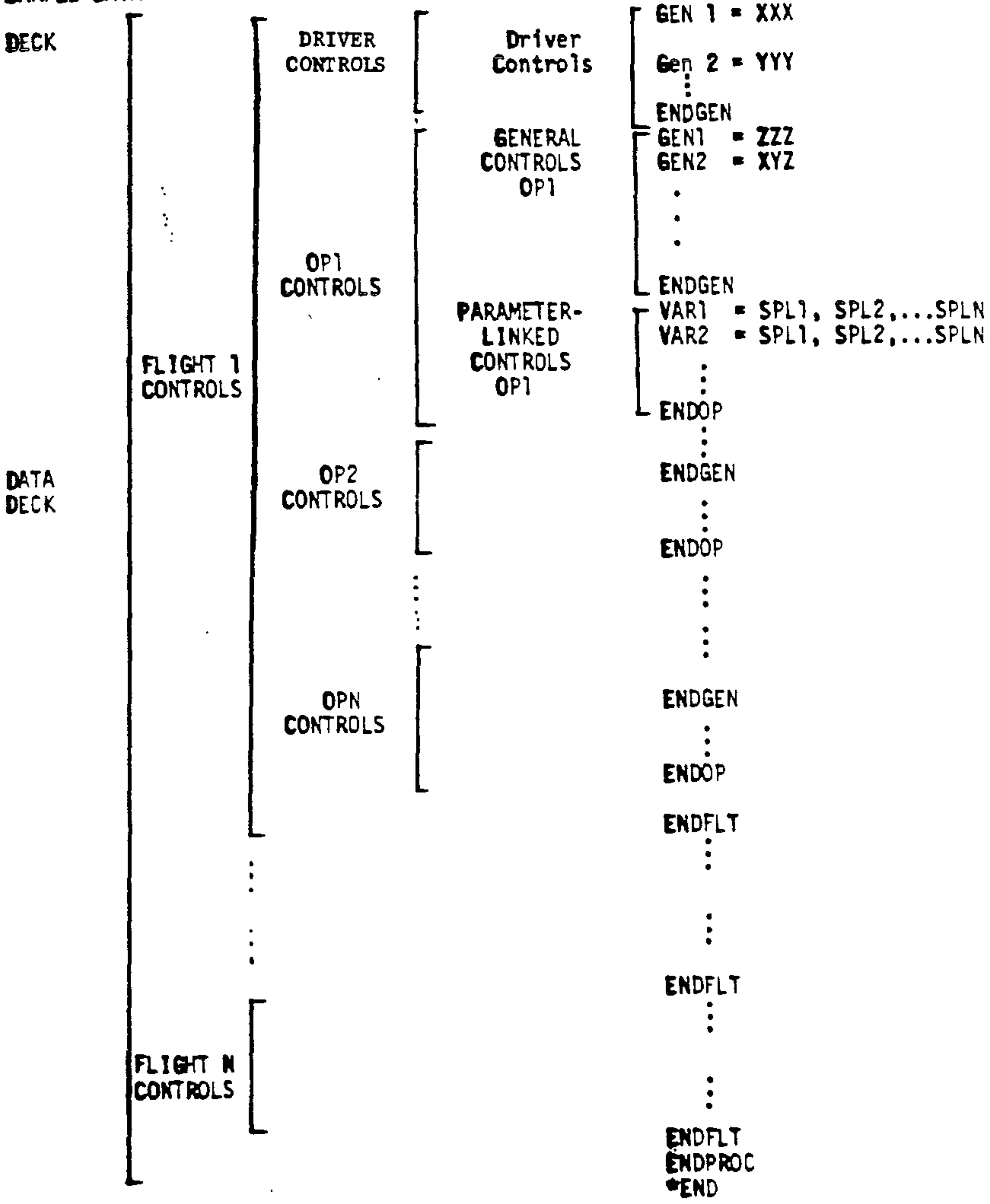
ENDFLT - The ENDFLT card follows the ENDOP card of the last operation of flight. There may be several ENDFLT cards in a data deck if several flights are to be processed. Each flight requires its own DRIVER and OPERATION data packets.

ENDPROC - The ENDPROC card follows the ENDFLT card of the last flight to be processed with this data deck. The ENDPROC card terminates the processor.

Some Rules:

1. The general controls for the DRIVER must be the first data packet in a data set for every flight.
2. The OPERATION data packets must be in the order in which the operations are to be performed on the data.
3. OPERATION data packets are made up of two sets of control cards: General control cards and PARAMETER-LINKED control cards.

## SAMPLE DATA DECK



```
*Run
GEN 1 = XXX

Gen 2 = YYY
   :
ENDGEN
GEN1   = ZZZ
GEN2   = XYZ
   .
   .
   .
ENDGEN
VAR1   = SPL1, SPL2,...SPLN
VAR2   = SPL1, SPL2,...SPLN
      :
ENDOP
      :
ENDGEN
      :
ENDOP
      :
      :
ENDGEN
      :
ENDOP
ENDFLT
      :
      :
ENDFLT
      :
      :
ENDFLT
ENDPROC
*END
```

**NITIALIZATION**
**RE-INITIALIZATION**

Several items in the OP COMMON BLOCK (see DRIVER Module outline) need initialization for each flight to be processed. However, once the first operation set has been encountered, the OP COMMON BLOCK is saved on a file and any or all of it may be used on subsequent flights. ( See flow diagram)

**EDITING**

Editing is possible in the data package on the two types of order lists: MLIST found in the DRIVER controls, and VORDER, found in the general control section of an OPERATION data packet. All edit cards must appear before the ENDGEN card. All edit cards apply to the order list for the current data packet being processed. An order list, when not specified for a particular operation, defaults back to the order list of the previous T-operation. The edit cards encountered apply to the appropriate order list. There are three types of edit cards: INS, REP, DEL.

USAGE:

INS = A, LIST

Insert LIST after item A.

REP = A, B, LIST

Replace A through B, inclusively, with LIST.

REP = A, A, LIST

Replace A with LIST.

DEL = A, B

Delete A through B, inclusively.

DEL = A, A

Delete A.

NOTE: The Edit Keywords: INS,REP,DEL may be changed by modifying .REPL statements in the MNGR module. For a complete description of the edit cards, see the sample data package.

DEFAULTING

Defaulting can be used within and between the operation data packets.

(a) As was mentioned above, when an order list is not specified in an operation data packet, the order list from the previous T-operation is accessed. If edit cards are found in the data packet, the edit cards apply to the order list that exists for this operation, whether it is actually specified in the packet, or has been arrived at through the default procedure.

(b) In Transformation operation data packets, output RATE information for each parameter must be specified. In data packets of this type, RATE is specified (RATE=X). All parameters following this RATE card and preceding the next rate cards have RATE X. Sometimes it may be desirable to use RATE information from the previous T-OP for some or all of the parameters for this operation. If this is desirable, let RATE=DFALT. Then, all parameters following this card and all un-specified parameters (i.e. parameters found in the order list for this operation and not specifically referred to in the parameter-linked control section are called un-specified parameters) will take on their rate from the previous T-OP. (For an example of how the RATE card is used, see the sample data package). The RATE card may be used an unlimited number of times and can only be used between the ENDGEN and the ENDOP card.

(c) In every operation packet in the parameter-linked control section, all parameters have one or more controls specified for each of them. The DFALT=LIST card is used in a similar manner as the RATE card. However, there are slight differences.

### DEFAULTING

The DFALT card also may be used an unlimited number of times between the ENDGEN card and the ENDOP card. If parameters require the DFALT information, they will use the preceding DFALT card. Any parameters in the order list and not specified at all between the ENDGEN card and the ENDOP card will take on the control list of the last DFALT card. Only standard parameter-linked controls may use the DFALT card.

```
Example:  DFALT = A, B, C
          VAR1  = D, D, 1.0
          VAR2  = 3, D, D
          VAR3  = 5, 7.3, D
          DFALT = Q, R, S
          VAR4  = 1, D, 9.2
                 .
                 .
                 .
```

```
The controls linked to VAR1 are:  A, B, 1.0
                       VAR2       3, B, C
                       VAR3       5, 7.3, C
                       VAR4       1, R, 9.2
```

(NOTE: Any parameters contained in the order list, VORDER, and not found in the Parameter-Linked control section will use the last RATE specified for their rate and the last DFALT specification for their controls. For a more complete example, see the sample data packet.)

### DIX=LIST

Many operations require additional control information for some of their parameters. This additional information is referred to as non-standard parameter-linked control information.

```
EXAMPLE:  DFALT = A,B,C
          DIX   = NSC1, NSC2, NSC3
          VAR1  = D, D, 1.0
```

```
                    VAR2 = 3, D, D
                    NSC2 = 'DEG C'
                    VAR3 = 5, 9, 3.0
                    VAR4 = 2, d, 1.0
DIX=LIST            NCS1 = 4.3, 9.2
                    NSC2 = 'SPEED IN M/S'
                    VAR5 = D,D,D
                    NSC3 = F3.2
```

The controls linked to each variables are:

```
                    VAR1 - A, B, 1.0
                    VAR2 - 3, B, C , NSC2, Speed in M,/s
                    VAR3 = 5, 9, 3.0
                    VAR4 = 2, B, 1.0 NSC1, 4.3, 9.2, NSC2, Speed in M,/s
                    VAR5 = A, B, C, NSC3, F3.2
```

Non-Standard parameter-linked control names must be specified in a DIX=LIST statement, where list is the list of non-standard parameter-linked control names. The DIX=LIST card may be found any where between the ENDGEN card and the ENDOP card.

MGRNC

MGRNC is an internal flag set by MNGR. MGRNC is initialized to zero when MNGR is entered the first time. When the first ENDFLT card is encountered, all initialization in the OP COMMON BLOCK can be completed. The original OP COMMON BLOCK initialization is accomplished partially by the user through data cards, and partially by the MNGR module. Once the OP COMMON BLOCK has been initialized, the entire common block is written out to a file and saved. MGRNC is then set to 1. For fights 2 through N it is up to the user whether or not to use any or all of the previous OP COMMON BLOCK for the next flight. (See flow charts and logic diagrams)

MGRNC

If the user specified elements of COMMON OP have not been defined in the DATA DECK, then the pre-existing values in the OP COMMON BLOCK, as found on the file, will be used. Thus, any or all of the OP COMMON BLOCK of the previous flight may be used by the next flight. Each time an ENDFLT card is encountered the OP COMMON BLOCK is written out to a file.

ERRORS & DIAGNOSTICS

Consistency checking of control information will be accompanied by diagnostic messages and/or error messages.

COMMON BLOCKs

The MNGR module uses .SAVE blocks to specify common blocks. See the DRIVER Module Review Outline for the common blocks needed by MNGR.

SUBROUTINES WITHIN THIS MODULE

BOOKKPR - Sets up some bookkeeping arrays for the OPERATION control files.

CYCSET - Sets negative numbers found in the arrays: NCYCSV, NCYCST, NCYEND to numbers which will cycle the data through the operation set in an optimum manner.

DMTST - Determines under and over dimensioning conditions based on the .REPL cards. Determines array size requirements.

DLT2 - Deletes information from a list.

FETCH1 - Locates an item in a list by testing every word in the list.

FETCH2 - Locates an item in a list by testing only the array header names within that list. (Note: FETCH1 and FETCH2 are utility routines available to the Operation modules to find specific items on their respective control files.)

GETNUM - A routine in the READLX package. Decodes hollerith strings into numeric information.

INITIO - An initialization routine.

INSRT2 - Inserts information into an existing list.

LEXCARD - Part of the READLX package. Reads and prints each card.

LEXCON - Part of the READLX package. Sets up the tables for READLX so that when data is encountered, READLX will know where to put it, bumping the appropriate counters and pointers.

MNGR - Reads in the data card packets. Sets up preliminary file information. Some initialization.

RANFIL - Generates the control random files for each operation.

READLX - A free-format input routine.

RP2 - Replaces existing information in a list with new information.

SEARCH - Sets up old and new rate and order information, and old index information for the control random files.

TESTLX - Part of the READLX package. Tests whether or not to read a new card.

SUBROUTINES NOT CONTAINED IN THIS MODULE

BRANRD, BRANWT, BRANCK - System library routines implemented through .SAVE blocks. (See DRIVER Module Review Outline).

OUTPUTC, EXIT - System Resident Routines.

MOVEC(AR1,AR2,NUM) - An ASCENT routine which moves memory contents from one space to another.

AR1 - Data is moved from this location.

AR2 - Data is moved to this location.

NUM - The number of points to move.

Access *ASCENT, S=ULIB, N=MOVE

## INPUT/OUTPUT

INPUT DATA PACKAGE

See attached sample data package listing and description.

OUTPUT

All input data is printed out as it is encountered on unit KCHECK.

DIAGNOSTIC & ERROR MESSAGES - When there are missing controls or control incompatibilities, diagnostic and/or error messages will be printed out on unit KERR.

RANDOM FILES - (see attached random file description and listing). All operations will have an associated random file containing all control information for that operation. These files will be accessed using the .SAVE blocks, WRFIL and RDFIL. The control file for each operation is printed out on unit KCHECK.

MULTIPLE ENTRY CAPABILITY & RAMIFICATIONS

The MNGR module may be re-entered an unlimited number of times. The end card ENDPROC terminates the processor. Until the ENDPROC card is encountered, an unlimited number of flights may be processed.

METHOD

See the attached flow diagram.

TIMING

TEST REQUIREMENTS

Test for control consistency and completeness. Test initialization/re-initialization procedures.

SPECIAL CONDITIONS OR RESTRICTIONS

Machine Dependence
O.S. Dependence > MOVEC, BRANRD, BRANWT
Portability yes

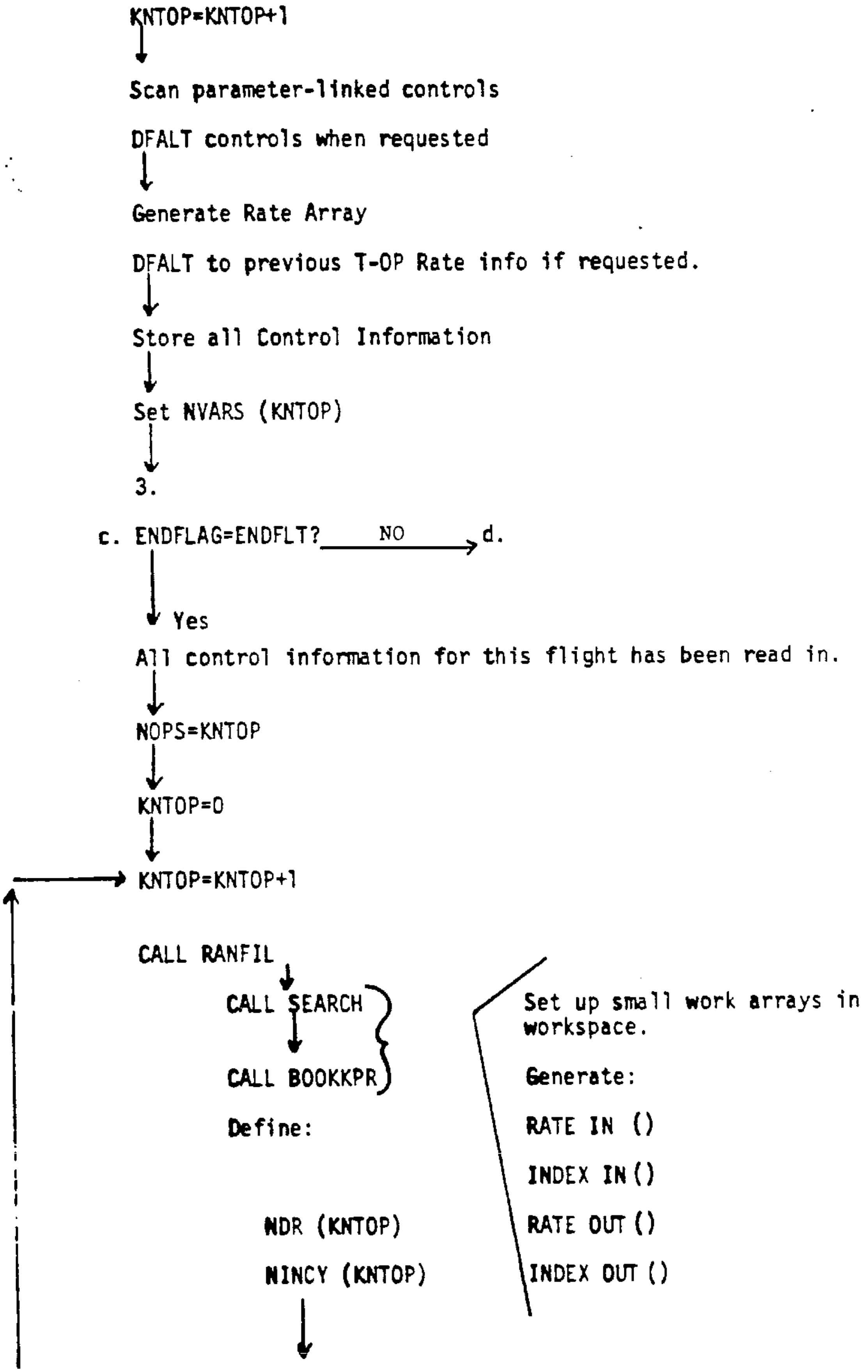
RECOMMENDATIONS

Any defaulting other than RATE defaulting on parameter-linked control defaulting (DFALT=) be implemented in the operation module itself.

Tabular list of controls be output by each operation module.

LOGIC FLOW DIAGRAM IN GRAMMATIC FORM

```
1. Some Initialization

   MGRNC=0  (Data Statement)

   KNTOP=-1
        :|
        :↓

2. MGRNC=1? __YES_________→  Restore COMMON/OP/→3.
        ↓
        |   NO

3. CALL READLX - Read a Data Packet
        ↓
4. Test on ENDFLAG

   a. ENDFLAG=ENDGEN? NO ──→ b.
     |
     ↓Yes

      Test the flags on the LEXCON variables to make sure

      they have been defined.
      |
      ↓

      Locate VORDER in CATCHALL.  If not found, use VORDER

      from the previous T-OP.
      |
      ↓

      Edit VORDER if requested.
      ↓
      Save CATCHALL

      3.

   b. ENDFLAG=ENDOP?______NO_____________→      c.
        |
        ↓ Yes
```



KNTOP=KNTOP+1

Scan parameter-linked controls

DFALT controls when requested

Generate Rate Array

DFALT to previous T-OP Rate info if requested.

Store all Control Information

Set NVARS (KNTOP)

3.

c. ENDFLAG=ENDFLT?

NO

d.

Yes

All control information for this flight has been read in.

NOPS=KNTOP

KNTOP=0

KNTOP=KNTOP+1

CALL RANFIL

CALL SEARCH

CALL BOOKKPR

Define:

NDR (KNTOP)

NINCY (KNTOP)

Set up small work arrays in workspace.

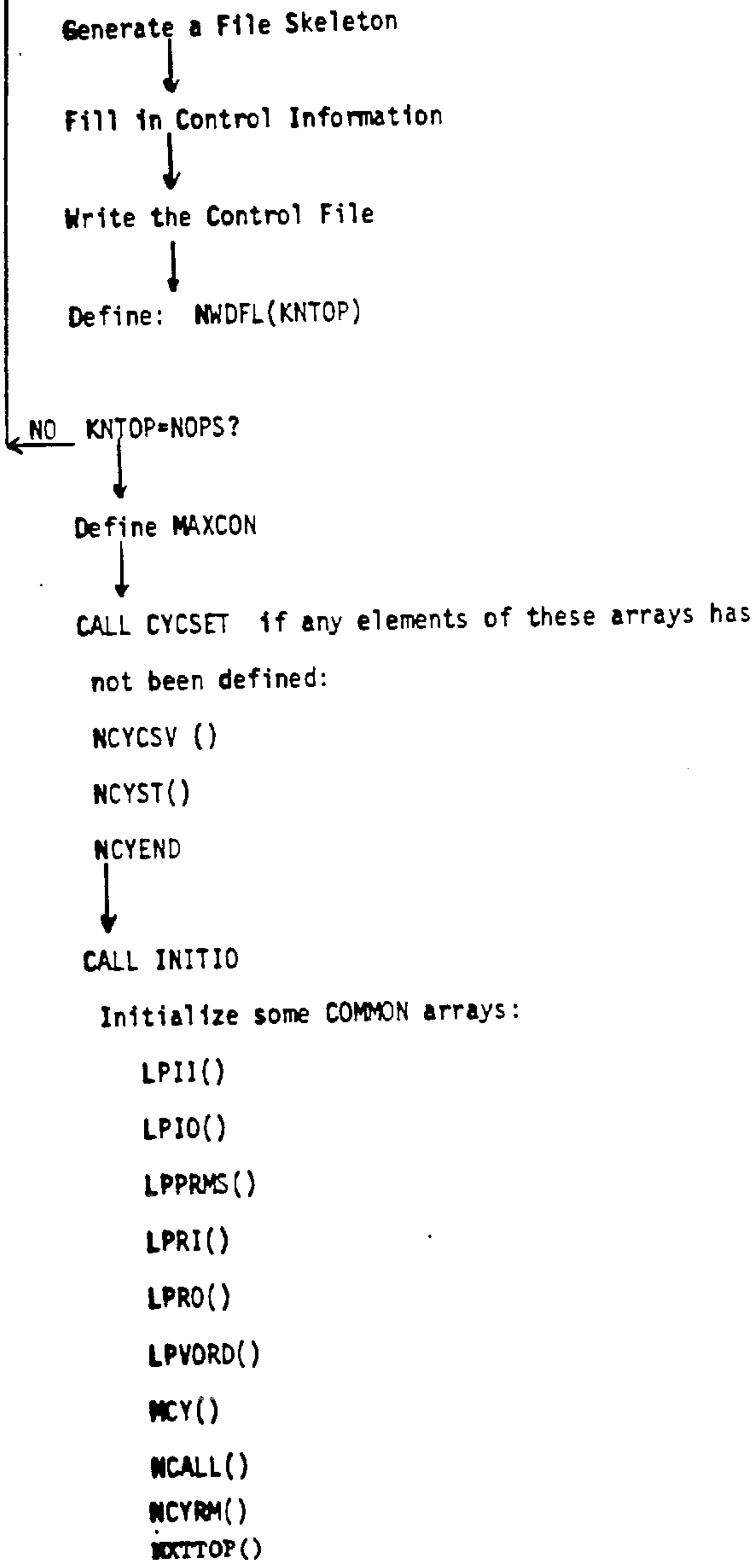
Generate:

RATE IN ()

INDEX IN ()

RATE OUT ()

INDEX OUT ()



Generate a File Skeleton

Fill in Control Information

Write the Control File

Define: NWDFL(KNTOP)

NO KNTOP=NOPS?

Define MAXCON

CALL CYCSET if any elements of these arrays has not been defined:

NCYCSV ()

NCYST()

NCYEND

CALL INITIO

Initialize some COMMON arrays:

LPII()

LPIO()

LPPRMS()

LPRI()

LPRO()

LPVORD()

NCY()

NCALL()

NCYRM()

NXTTOP()

```
          TIM()

          VARIND()
        ↓
      CALL DMTST

        Check for Dimension Integrity
        ↓
        RETURN

D.    ENDFLAG=ENDPROC?
        ↓ Yes
      STOP
```

A DESCRIPTION OF HOW TO USE DATA PACKAGE FOR GENPRO2

BRIEF DESCRIPTION OF DATA PACKAGE

With the exception of the DRIVER, all modules are divided into (1) a GENERAL CONTROL SECTION and (2) a PARAMETER-LINKED CONTROL SECTION. Each of these modules is called a CONTROL PACKET. Examples of CONTROL PACKETS are INPUT CONTROLS, CALIB CONTROLS and PLOT CONTROLS, etc.

HOW CONTROL PACKETS ARE FLAGGED

Four end flags are used.

-ENDGEN- indicates the end of general controls

The DRIVER is flagged by ENDGEN.

The GENERAL CONTROL SECTION in each CONTROL PACKET is flagged by ENDGEN.

-ENDOP- indicates the end of the PARAMETER LINKED CONTROL SECTION in each CONTROL PACKET.

-ENDFLT- indicates the end of control data card packets for a particular flight.

-ENDPROC- indicates end of run.

EDITING CAPABILITIES

Order list may be edited. The editing capabilities are insert, replace and delete. (Keywords: INS, REP,DEL) These editing keywords may be changed by modifying the Hollerith fields on the .REPL cards for $EDIT1, $EDIT2, and $EDIT3 in MNGR.

To insert into a list:

INS= A,LIST implies insert the item or items in LIST after item A in the order list.

To replace in a list:

REP= FWA,LWA,LIST implies replace FWA through LWA inclusive in the order list with the item or items in LIST.

To delete in a list:

DEL= FWA,LWA implies delete inclusively the items from FWA to LWA in the order list.

OTHER FEATURES IN THE DATA PACKAGE

The following controls NCYCSV(), NCYCST(), AND NCYEND() may be specified by the user, or they may be without specification, in which case CYCSET will determine optimum values for these arrays.

If TSTRT, TEND, and VORDER() are not specified then the program uses the times and the order list from the previous T-operation.

RATE is used by the MANAGER.

RATE is set equal to the number of samples per DELVI unit. After each RATE (There may be more than one rate) are the parameters which are to be processed at that rate.

There is a default parameter, DFALT.

The last DFALT applies to parameters not listed in the PARAMETER LINKED CONTROL SECTION but found in the order list.

All DFALT's apply to any parameters which follow that particular default parameter.

If a PARAMETER-LINKED control is set equal to D, then the D implies default and the local default value replaces the D.

On the data cards, all array names appearing on the left side of the equal sign with a set of empty parenthesis just after the array name are special general controls which need to be specified in the general control section of every operation data packet. (exception arrays: NCYCSV(), NCYST(), NCYEND() ).

The empty brackets tell READLX to put the value just to the right of the equal sign into the next available location in this general control array:

Sample data card: NAMEOP() =CALIB

If CALIB is the second operation encountered in the data deck, then NAMEOP(2)=CALIB

READLX will automatically bump the index pointer in the arrays defined using the empty set of parenthesis each time the array name is encountered.

To determine which arrays are to be specified in this manner, consult the comment cards which accompany each data packet. Currently, the arrays to be defined in this way are:

ITYPOP()

NAMEOP()

NCYCST()

NCYCSV()

NCYEND()

## SOME GENERAL INFORMATION ABOUT THE DATA PACKET

- A slash (/) indicates a comment. A slash may appear anywhere on the card.
- Each control to be specified in the data packet should be commented.
- Some controls are specified using () parentheses. Comment cards will indicate which controls are to be defined in this manner.
- A comment card of the form: CONTROL/COMMON BLOCK NAME/ , indicates this control resides in a COMMON BLOCK.
- Each control is followed by an equal sign (=), followed by a list. A list may contain more than one item. A list may be made up of Integers, Floating Point Numbers, or Hollerith information. A list may use more than one card.
- Blanks are ignored on all data cards, unless they make up a Hollerith string.
- Hollerith strings containing more than 10 (8) characters must be enclosed within quotes. Hollerith strings containing 10 (8) characters or less need not be enclosed within quotes unless these strings contain special characters, i.e., $, ' , :, blank, (, ), /.
- Commas and colons are used as delimiters
- The dollar sign ($) implies a new card.

## CONTROLS WHICH MUST BE SPECIFIED

DRIVER:

AFILE - file name for the DRIVER module (PLIB).
CNVIND - UP TO 3 CONVERSION FACTORS MAY BE DEFINED FOR THE INDEPENDENT VARIABLE
DELVI - independent variable increment.

IFLUSH - if IFLUSH=1, flush DR() storage areas of all operations before ending.
LBVIND - UP TO 3 LABELS MAY BE DEFINED FOR THE INDEPENDENT VARIABLE
MLIST - the master variable list.

TEND - time to end processing (HR,MIN,SEC).

TSTRT - time to begin processing (HR,MIN,SEC).

CONTROLS WHICH MUST BE SPECIFIED (cont'd)

OPERATIONS:

ITYPOP() - Operation type. ITYPOP() = 0 Snapshot Operation

ITYPOP() = 1 Transformation Operation

NAMEOP() - Operation name.

NCYCSV() - the number of cycles to save for this Operation.

NCYEND() - the number of cycles of overlap for the following T-Op.

NCYST() - the number of the cycle on which to start processing.

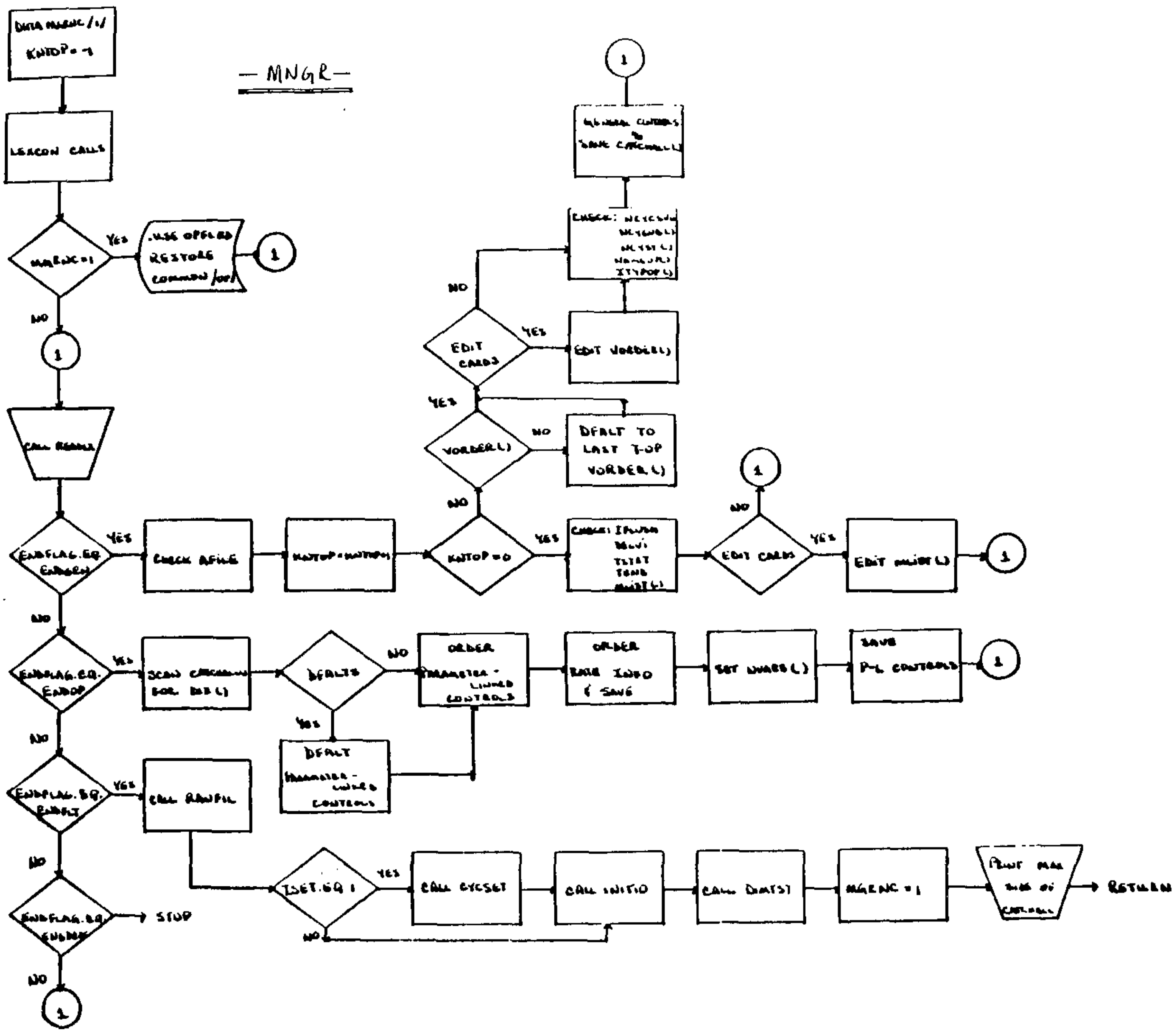
PFILE - the file name for this Operation module (PLIB).

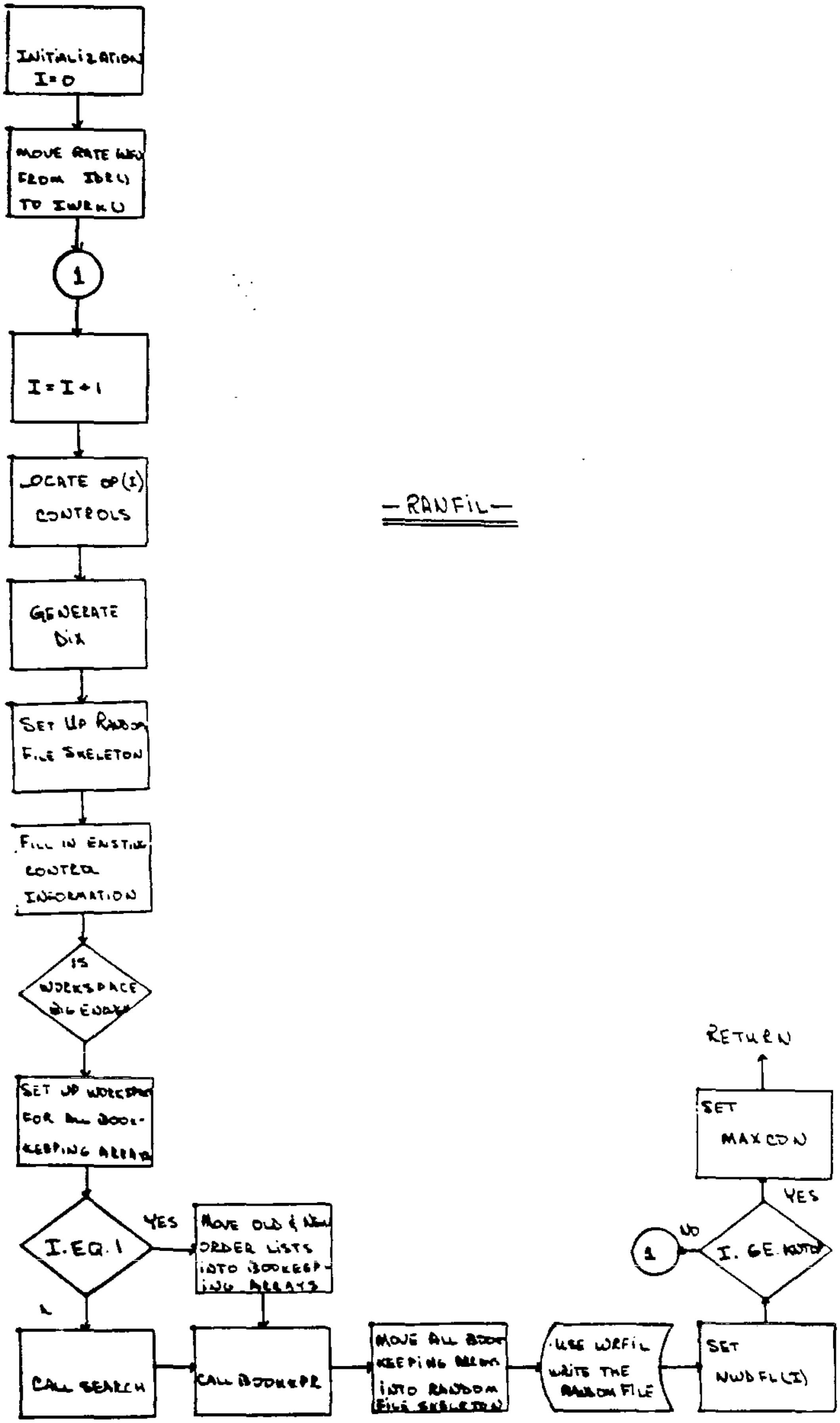
VORDER - the variable order list for this Operation.

SOME GENERAL INFORMATION ABOUT THE CONTROLS

- Transformation Operations (T-Op) require that rate information be specified in the Parameter-Linked (P-L) Control section. RATE = X. If RATE is not specified, rate information from the previous T-Op will be used.
- If P-L Controls are to be defaulted, a default must be specified. DFALT = LIST.
- Any parameters found in the order list (VORDER), but not specified in the P-L control section, take on the P-L controls of the last DFALT to be specified, and the last RATE to have been specified for that operation.
- If it is desirable for an unspecified parameter in the P-L control section to take on rate information from the previous T-Op, then specify RATE = DFALT.
- If a parameter is INPUT twice, it must be INPUT with two different names. All names in the order list (VORDER) must be unique.
- All order lists (VORDER) must be subsets of the previous T-Op's order list, unless the order list is that of a CREATE Operation.
- All control names found in the P-L control section must have been specified

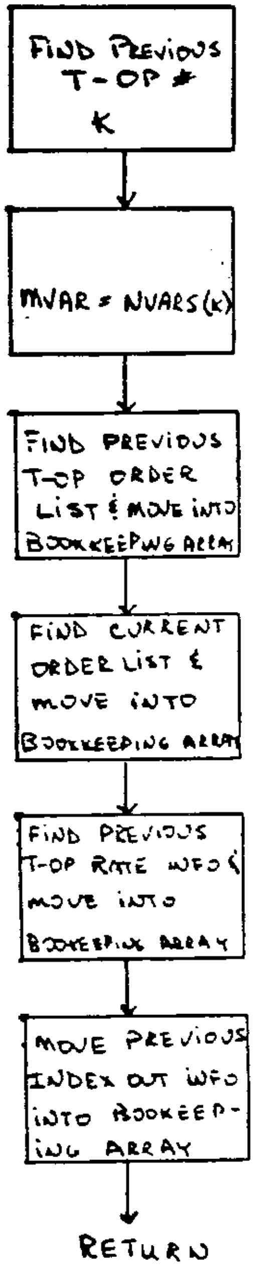
# FLOW DIAGRAMS FOR THE MAJOR ROUTINES OF THE MNGR MODULE


— MNGR —
DATA MGENC /1/
KNTOP = 1
LEXCON CALLS
MGENC = 1
YES
NO
USE OFFLRD
RESTORE
COMMON /OP/
1
CALL READL
ENDFLAG.EQ.ENDGEN
YES
CHECK AFILE
KNTOP = KNTOPM
KNTOP = 0
YES
CHECK: IFUNN
NLVI
TSTAT
TEND
MNIT( )
EDIT CARDS
YES
EDIT MNIT( )
NO
VORDER( )
NO
DFALT TO
LAST T-OP
VORDER( )
YES
EDIT CARDS
YES
EDIT VORDER( )
NO
CHECK: NCYCSW
NCYCNB( )
NCYST( )
NAMEOP( )
ITYPOP( )
GENERAL CONTROLS
TO
SAVE CARCHAR( )
ENDFLAG.EQ.ENDOP
YES
SCAN CARCHAR
FOR DEF( )
DFALTS
NO
ORDER
PARAMETER-LINKED
CONTROLS
YES
DFALT
PARAMETER-LINKED
CONTROLS
ORDER
RATE INFO
& SAVE
SET NVARS( )
SAVE
P-L CONTROLS
ENDFLAG.EQ.ENDFLT
YES
CALL RAWFIL
ISET.EQ.1
YES
CALL CYCSET
NO
CALL INITIO
CALL DMTST
MGENC = 1
PRINT MAX
SIZE OF
CARCHAR
RETURN
ENDFLAG.EQ.ENDRUN
STOP
NO

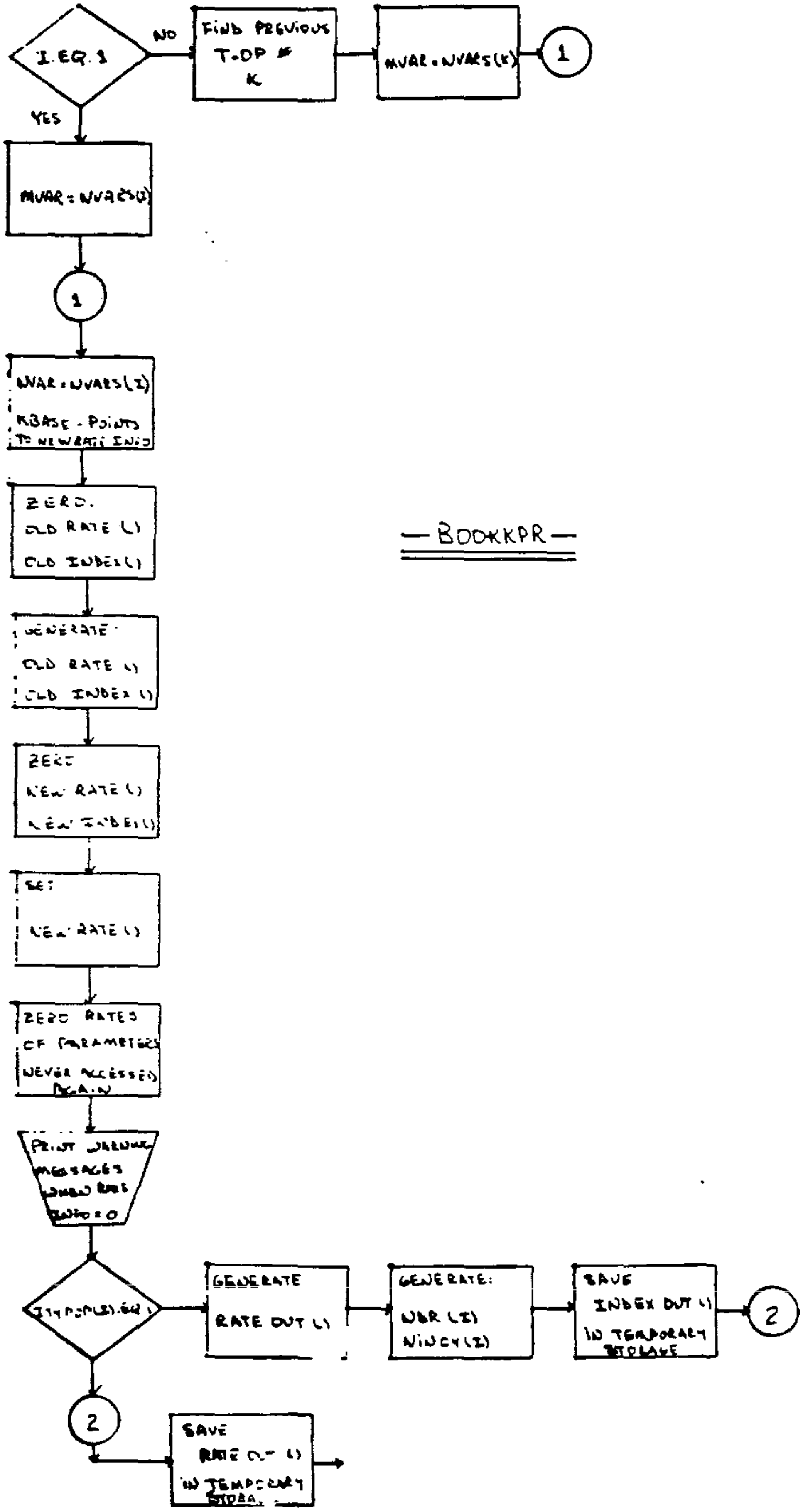


—RANFIL—

INITIALIZATION
I=0
MOVE RATE WEU FROM IDR(I) TO IWRK(I)
1
I=I+1
LOCATE OP(I) CONTROLS
GENERATE DIX
SET UP RANDOM FILE SKELETON
FILL IN EXISTING CONTROL INFORMATION
IS WORKSPACE BIG ENOUGH
SET UP WORKSPACE FOR ALL BOOK-KEEPING AREAS
I.EQ.1
YES
MOVE OLD & NEW ORDER LISTS INTO BOOKKEEP-ING ARRAYS
CALL SEARCH
CALL BOOKKPR
MOVE ALL BOOK KEEPING AREAS INTO RANDOM FILE SKELETON
USE WRFIL WRITE THE RANDOM FILE
SET NWDFL(I)
I.GE.NTO
NO
1
YES
SET MAXCON
RETURN


FIND PREVIOUS T-OP # K
MVAR = NVARS(K)
FIND PREVIOUS T-OP ORDER LIST & MOVE INTO BOOKKEEPING ARRAY
FIND CURRENT ORDER LIST & MOVE INTO BOOKKEEPING ARRAY
FIND PREVIOUS T-OP RATE INFO & MOVE INTO BOOKKEEPING ARRAY
MOVE PREVIOUS INDEX OUT INFO INTO BOOKKEEPING ARRAY
RETURN

—SEARCH—



— BOOKKPR —

I.EQ.1
NO
YES
FIND PREVIOUS T-OP # K
MVAR=NVARS(K)
1
MVAR=NVARS(I)
1
NVAR=NVARS(I)
KBASE - POINTS TO NEW RATE IND
ZERO.
OLD RATE ()
OLD INDEX()
GENERATE:
OLD RATE ()
OLD INDEX ()
ZERO
NEW RATE ()
NEW INDEX()
SET
NEW RATE ()
ZERO RATES OF PARAMETERS NEVER ACCESSED AGAIN
PRINT WARNING MESSAGES WHEN RATE INFO=0
ITYPOP(I).EQ.1
GENERATE
RATE OUT ()
GENERATE:
NBR (I)
NINCY(I)
SAVE
INDEX OUT ()
IN TEMPORARY STORAGE
2
2
SAVE
RATE OUT ()
IN TEMPORARY STORA.

# SAMPLE DATA PACKAGE

```
/
/
/
/       ORIVER CONTRCLS
/
/
/ AFILE - DRIVEF PLIB FILES
/ CELVI/CYTIM/ - INCEPENDENT VARIABLE INCREMENT
/ IFLUSH/OP/ - FLUSH FLAG, IF IFLUSH=1, FLUSH DR STORAGE AREAS OF ALL
/                OPERATICNS BFFORE ENDING
/ MLIST - MASTER VARIABLE LIST
/ TSTRT/CYCTIM/ - TIFE TO START PROCESSING (HR,MIN,SEC)
/ TEND/CYCTIM/ - TIME TO ENO PRCCESSING (HR,MIN,SEC)
/
MLIST = PSF,PSFC,TAS,TEMP,TIME
AFILE = ≠GNPR20R..0≠
IFLUSH = 0
DELVI = 1.
TSTRT = 17.,11.,40.
TEND = 17.,11.,50.
CNVIND = 3600.,60.,1.
LBVIND = HR,MIN,SEC
IVFLAG = 0
 ENDGEN
 /
 /
 /         INPUT CONTRCLS
 / GENERAL CONTROL SECTICN FOR INPUT
 /
 /
 / ITYPOP()/OP/ - OPERATION TYPE
 / KIN - LOGICAL UNIT NUMBER FOR INPUT
 / KODE - DECODING FLAGS
 /          IF KODE=0, DIGITAL WORD
 /          IF KCDE=1, ANALOG WORD
 / KPRNT - PRINT OPTICN FOR INPUT CATA
 /         IF KPRNT=1, PRINT FIRST RECORD ONLY
 /         IF KPRNT=2, PRINT EVERY DECODED RECORD
 /         IF KPRINT=3, PRINT EVERY ORIGINAL RECORD
 / MODEIN - MODE OF INPUT TAPE
 /          IF MODEIN=0, BCD INPUT
 /          IF MODEIN=1, BINARY INPUT
 / NAMEOP()/OP/ - OPERATION NAME
 / NCYCST()/OP/ - NUMBER OF CYCLE TO START PROCESSING
 / NCYCSV()/OP/ - NUMEER OF CYCLES OF DATA TO SAVE FOR THIS OPERATION
 / NCYEND()/OP/ - NUMBER OF CYCLES OF CVERLAP NEEDED FOR FOLLOWING T OP
 / NTYPIN - READ CONTROL
 /          IF NTYPIN=2, ATTEMPT PARITY RE-READ
 /          IF NTYPIN=6, SKIP RECORDS
 / PFILE - OPERATION PLIE FILE
 / VORCER - VARIABLE ORDER LIST FOR THIS OPERATION
 /
 NAMEOP()=INPUT
 ITYPOP() = 1
 PFILE = ARISN
 KPRNT = 2
 MODEIN = 1
 NTYPIN = 2
 KIN = 3
 KODE = 1,1,1,1,0,1,0,1,1,1,1,1,0,1,0,1,1,1,1,1,0,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,
 0,1,0,1,1,1,1,1,0,1,0,1,1,1,1,1,0,1,0,1,1,1,1,1,0,1,0,1
```

```
VORDER = TIME,PSF,TEMP
NCYCSV()=2
NCYEND()=0
NCYST()=1
ENDGEN
/
/
/
/         VARIABLE CONTROL SECTICN FCR INPUT
/
/    FORM OF PARAMETER-LINKED CONTROL CARDS  -  NAME = LOC
/
/ NAME - VARIABLE NAME
/ LOC - LOCATION IN DATA RECORD OF FIRST SAMPLE OF THIS VARIABLE
/
/
/  RATE - THE OUTPUT RATE FOR EACH PARAMETER (SAMPLES/DELVI)
/
/
RATE = 8
DFALT = -9999
TIME = D— letter D (u.c. DATA)
TEMP = 16
RATE = 32
PSF = 22
ENDOP
/
/      CALIB CONTROLS
/
/
/      GENERAL CONTROL SECTION FOR CALIB
/
/
/ ITYPOP()/OP/ - OPERATION TYPE
/ NAMEOP()/OP/ - OPERATION NAME
/ NCYCST()/OP/ - NUMBER OF CYCLF TO START PROCESSING
/ NCYCSV()/OP/ - NUMBER OF CYCLES OF DATA TO SAVE FOR THIS OPERATION
/ NCYEND()/OP/ - NUMBER OF CYCLES OF CVERLAP NEEDED FOR FOLLOWING T OP
/ .PFILE  -_ OPERATICN FLIB FILE
/ VORCER - VARIABLE CRDER LIST FOR THIS OPERATION
/
/
 NAMEOP( ) = CALIB
 ITYPOP() = 1
 PFILE = CAL4DRT
 INS = PSF,TAS,PSFC
 NCYCSV( ) = 4
 NCYEND( ) = 0
 NCYST( ) = 1
 ENDGEN
 /
 /
 /      VARIABLE CONTRCL SECTION FOR CALIE
 /
 /
 /   FORM OF PARAMETER-LINKED CONTRCL CARD  -  NAME = ICALC
 /
```

```
/  NAME  -  VARIABLE NAME
/  ICALC  -  FLAG TO INDICATE HOW VARIABLE IS CALCULATED
/            IF ICALC = 1, THEN VARIABLE IS UNCHANGED FROM INPUT
/            IF ICALC = 2, THEN VARIABLE IS DERIVED FROM SOURCE
/            IF ICALC .LT. 0, THEN ABS(ICALC) = NUMBER OF COEFFICIENTS
/                 IN CALIBRATION EQUATION
/
/  RATE - THE OUTPUT RATE FOR EACH PARAMETER (SAMPLES/DELVI)
/
/  A DIX LIST IS NEEDED FOR THIS OPERATION
/  DIX = COEFF, SOURCE
/
/  COEFF - THE CALIBRATION EQUATION COEFFICIENTS
/  COEFF = X1,X2,X3
/                X1 - CONSTANT
/                X2 - COEFFICIENT OF THE X TERM
/                X3 - COEFFICIENT OF THE X**2 TERM
/
/  SOURCE - VARIABLES NEEDED TO DERIVE THE PRECEDING PARAMETER
/
/
DIX = COEFF,SOURCE
RATE = DFALT
DFALT = 1
PSF = -3
COEFF = 663.2352, .368981, 2.24E-6
TEMP = -3
COEFF = -21.682, 6.0571E-2, 2.2E-6
RATE = 16
PSFC = 2
SOURCE = PSF
TAS = 2
SOURCE = PSF,TEMP
ENDOP
/
/
/     PRINTER CONTROLS
/
/     GENERAL CONTROL SECTION FOR PRINTER
/
/
/ ITYPOP()/OP/ - OPERATION TYPE
/  KUNIT  -  LOGICAL UNIT NUMBER FOR PRINTER
/ NAMEOP()/OP/ - OPERATION NAME
/ NCYCSV()/OP/ - NUMBER OF CYCLES OF DATA TO SAVE FOR THIS OPERATION
/ NCYEND()/OP/ - NUMBER OF CYCLES OF OVERLAP NEEDED FOR FOLLOWING T OP
/ NCYCST()/OP/ - NUMBER OF CYCLE TO START PROCESSING
/  PFILE  -  OPERATION FLIB FILE
/ VORDER - VARIABLE ORDER LIST FOR THIS OPERATION
/
NAMEOP( ) = PRINTER
ITYPOP() = 0
PFILE = QD-PRNTR
KUNIT = 4
VORDER = TIME,TEMP,PSF,TAS,PSFC,XXX
DEL = XXX,XXX
NCYCSV() = 4
NCYEND() = 0
NCYST() = 1
ENDGEN
```

```
/
/
/     VARIABLE CONTROL SECTION FOR  PRINTER
/
/  FORM OF THE PARAMETER-LINKED CONTROL CARD  -  NAME = UNIT,FORMAT
/
/  NAME   -  VARIABLE NAME
/  UNIT   -  PHYSICAL UNITS IN WHICH VARIABLE IS OUTPUT
/  FORMAT  -  FORMAT DESIGNATOR FOR VARIABLE
/
DFALT = DEG,F7.2
TIME = SEC,D
PSFC = MB,D
TAS=≠M/S≠,D
ENDOP
ENDFLT
ENDPROC
```

## RANDOM FILE INFORMATION

Note· The SAMPLE RANDOM FILES listed have been generated from the SAMPLE DATA PACKAGE. All General Control information that is stored in COMMON is not stored on the Random File.

## RANDOM FILE DESCRIPTION

All random files have a basic structure. An array called DIX will head up each file. DIX will contain all control array names found on the file and the location of the beginning of each of these arrays. DIX is followed by the control arrays that it points to. The general form of each file is: an array name, followed by the length of this array, followed by the contents of the array, the next array name, followed by the length of this array, followed by the contents of the array...

| WORD NUMBER | CONTENTS |
|---|---|
| 1 | ~~DIX~~ HEADER - The header word for the DIX array. DIX is a pointer array for the control file. |
| 2 | 12 - The length of the array DIX. |
| 3 | VORDER - The order list for this operation. This example assumes three parameters to be processed by this operation. |
| 4 | 15 - The location of the header word for the VORDER control array. |
| . | . |
| . | . |
|  | Any other special operation general controls would go here. This example assumes none. For a more detailed example, see the SAMPLE RANDOM FILES. |
| . | . |
| . | . |
| 5 | PARAMS - The Parameter-Linked control section of the file. This example assumes two P-L controls for each parameter and no non-standard P-L controls. |
| 6 | 20 - The location of the header word for the PARAMS control array. |
| 7 | RATEIN - The input rate information for this operation. |
| 8 | 34 - The location of the header word for the RATEIN control array. |
| 9 | INDEXIN - The input index information for this operation. This array tells the processor where to locate each input parameter to be processed. |
| 10 | 39 - The location of the header word for the INDEXIN control array. |
| 11 | RATEOUT - The output rate information for this operation. |
| 12 | 44 - The location of the header word for this RATEOUT control array. |
| 13 | INDEXOUT - The output index information for this operation. This array tells the processor where to put each parameter after it has been processed. |
| 14 | 49 - The location of the header word for the INDEXOUT control array. |
| 15 | VORDER |
| 16 | 3 |
| 17 | PAR1 |
| 18 | PAR2 |
| 19 | PAR3 |
| 20 | PARAMS |

RANDOM FILE
DESCRIPTION
(cont'd)

| WORD NUMBER | CONTENTS |
|---|---|
| 21 | 12 |
| 22 | PAR1 |
| 23 | 2 |
| 24 | P-L1 |
| 25 | P-L2 |
| 26 | PAR2 |
| 27 | 2 |
| 28 | P-L1 |
| 29 | P-L2 |
| 30 | PAR3 |
| 31 | 2 |
| 32 | P-L1 |
| 33 | P-L2 |
| 34 | RATEIN |
| 35 | 3 |
| 36 | Input rate for PAR1 |
| 37 | Input rate for PAR2 |
| 38 | Input rate for PAR3 |
| 39 | INDEXIN |
| 40 | 3 |
| 41 | Input index for PAR1 |
| 42 | Input index for PAR2 |
| 43 | Input index for PAR3 |
| 44 | RATEOUT |
| 45 | 3 |
| 46 | Output rate info for PAR1 |
| 47 | Output rate for PAR2 |
| 48 | Output rate for PAR3 |
| 49 | INDEXOUT |
| 50 | 3 |
| 51 | Output index for PAR1 |
| 52 | Output index for PAR2 |
| 53 | Output index for PAR3 |

The arrays RATEIN, INDEXIN, RATEOUT, INDEXOUT, and PARAMS are ordered as VORDER is defined.

When a parameter appears in an order list (VORDER) for the first time in any operation, information about this parameter in the RATEIN and INDEXIN arrays is set to zero.

When a parameter appears in an order list (VORDER) in one or more operations, but is not found in the order lists of subsequent operations, RATEOUT and INDEXOUT information for this parameter in the last operation to access this parameter is set to zero.

Rate information specified in an operation data packet is defined as output rate information (RATEOUT) for that operation.

The information in the INDEXIN and INDEXOUT arrays refers to locations in the DR array in BLANK COMMON.

## OP 1 RANDOM FILE

| WORD NUMBER | WORD |
|---|---|
| 1 | ~~DIX~~ HEADER |
| 2 | ~~26~~ 21 |
| 3 | VORDER |
| 4 | ~~29~~ 25 |
| 5 | KPRNT |
| 6 | ~~34~~ 30 |
| 7 | MODEIN |
| 8 | ~~37~~ 33 |
| 9 | NTYPIN |
| 10 | ~~40~~ 36 |
| 11 | KIN |
| 12 | ~~43~~ 39 |
| 13 | KODE |
| 14 | ~~46~~ 42 |
| ~~15~~ | ~~TIME1~~ |
| 16 | ~~112~~ |
| ~~17~~ | ~~TI~~ME2 |
| ~~18~~ | ~~123~~ |
| ~~19~~ 15 | PARAMS |
| ~~20~~ 16 | ~~134~~ ~~108~~ 108 |
| ~~21~~ 17 | RATEIN |
| ~~22~~ 18 | ~~145~~ ~~123~~ 119 |
| ~~23~~ 19 | INDEXIN |
| ~~24~~ 20 | ~~150~~ ~~128~~ 124 |
| ~~25~~ 21 | RATEOUT |
| ~~26~~ 22 | ~~155~~ ~~133~~ 129 |
| ~~27~~ 23 | INDEXOUT |
| ~~28~~ 24 | ~~160~~ ~~138~~ 134 |
| ~~29~~ 25 | VORDER |
| ~~30~~ 26 | 3 |
| ~~31~~ 27 | TIME |
| ~~32~~ 28 | PSF |
| ~~33~~ 29 | TEMP |
| ~~34~~ 30 | KPRNT |
| ~~35~~ 31 | 1 |
| ~~36~~ 32 | 2 |
| ~~37~~ 33 | MODEIN |
| ~~38~~ 34 | 1 |
| ~~39~~ 35 | 1 |
| ~~40~~ 36 | NTYPIN |
| ~~41~~ 37 | 1 |
| ~~42~~ 38 | 2 |
| ~~43~~ 39 | KIN |
| ~~44~~ 40 | 1 |
| ~~45~~ 41 | 3 |
| ~~46~~ 42 | KODE |
| ~~47~~ 43 | 64 |
| ~~48~~ 44 | 1 |
| ~~49~~ 45 | 1 |
| . | . |
| . | . |
| . | . |
| ~~109~~ 105 | 1 |
| ~~110~~ 106 | 0 |
| ~~111~~ 107 | 1 |
| ~~112 108~~ | ~~TIME1~~ |
| ~~113 109~~ | ~~9~~ |

**SAMPLE RANDOM FILES (cont'd)**

| WORD NUMBER | WORD |
|---|---|
| ~~114~~ | ~~17~~ |
| ~~115~~ | ~~11~~ |
| ~~116~~ | ~~40~~ |
| ~~117~~ | ~~17~~ |
| ~~118~~ | ~~13~~ |
| . | . |
| . | . |
| . | . |
| ~~123~~ | ~~TIME~~2 |
| ~~124~~ | ~~9~~ |
| ~~125~~ | ~~17~~ |
| ~~126~~ | ~~11~~ |
| ~~127~~ | ~~50~~ |
| . | . |
| . | . |
| . | . |
| ~~134~~ 108 | PARAMS |
| ~~135~~ 109 | 9 |
| ~~136~~ 110 | TIME |
| ~~137~~ . | 1 |
| ~~138~~ : | -9999 |
| ~~139~~ | PSF |
| ~~140~~ | 1 |
| ~~141~~ | 22 |
| ~~142~~ | TEMP |
| ~~143~~ : | 1 |
| ~~144~~ . | 16 |
| ~~145~~ 119 | RATEIN |
| ~~146~~ | 3 |
| ~~147~~ . | 0 |
| ~~148~~ : | 0 |
| ~~149~~ | 0 |
| ~~150~~ 124 | INDEXIN |
| ~~151~~ | 3 |
| ~~152~~ . | 0 |
| ~~153~~ : | 0 |
| ~~154~~ | 0 |
| ~~155~~ 129 | RATEOUT |
| ~~156~~ | 3 |
| ~~157~~ . | 8 |
| ~~158~~ : | 32 |
| ~~159~~ | 8 |
| ~~160~~ 134 | INDEXOUT |
| ~~161~~ | 3 |
| ~~162~~ . | 1 |
| ~~163~~ . | 17 |
| ~~164~~ 138 | 81 |

## OP 2 RANDOM FILE

| WORD NUMBER | WORD |
|---|---|
| 1 | ~~DEX~~ HEADER |
| 2 | 12 |
| 3 | VORDER |
| 4 | 15 |
| 5 | PARAMS |
| 6 | 22 |
| 7 | RATEIN |
| 8 | 56 |
| 9 | INDEXIN |
| 10 | 63 |
| 11 | RATEOUT |
| 12 | 70 |
| 13 | INDEXOUT |
| 14 | 77 |
| 15 | VORDER |
| 16 | 5 |
| 17 | TIME |
| 18 | PSF |
| 19 | TAS |
| 20 | PSFC |
| 21 | TEMP |
| 22 | PARAMS |
| 23 | 32 |
| 24 | TIME |
| 25 | 1 |
| 26 | 1 |
| 27 | PSF |
| 28 | 6 |
| 29 | -3 |
| 30 | COEFF |
| 31 | 3 |
| 32 | 663.24 |
| 33 | .37 |
| 34 | 2.24E-6 |
| 35 | TAS |
| 36 | 5 |
| 37 | 2 |
| 38 | SOURCE |
| 39 | 2 |
| 40 | PSF |
| 41 | TEMP |
| 42 | PSFC |
| 43 | 4 |
| 44 | 2 |
| 45 | SOURCE |
| 46 | 1 |
| 47 | PSF |
| 48 | TEMP |
| 49 | 6 |
| 50 | -3 |
| 51 | COEFF |
| 52 | 3 |
| 53 | -21.68 |
| 54 | .06 |
| 55 | 2.2E-6 |
| 56 | RATEIN |
| 57 | 5 |

SAMPLE RANDOM FILES (cont'd)

| WORD NUMBER | WORD |
|---|---|
| 58 | 8 |
| 59 | 32 |
| 60 | 0 |
| 61 | 0 |
| 62 | 8 |
| 63 | INDEXIN |
| 64 | 5 |
| 65 | 1 |
| 66 | 17 |
| 67 | 0 |
| 68 | 0 |
| 69 | 81 |
| 70 | RATEOUT |
| 71 | 5 |
| 72 | 8 |
| 73 | 0 |
| 74 | 16 |
| 75 | 16 |
| 76 | 8 |
| 77 | INDEXOUT |
| 78 | 5 |
| 79 | 98 |
| 80 | 0 |
| 81 | 130 |
| 82 | 194 |
| 83 | 258 |

## OP 3 RANDOM FILE

| | | | |
|---|---|---|---|
| 1 | ~~DIX~~ HEADER | | |
| 2 | ~~14~~ 18 | | |
| 3 | VORDER | | |
| 4 | ~~17~~ 21 → ~~[illegible]~~ → | 5 | TIME1 |
| ~~5~~ 9 | KUNIT | | 23 |
| ~~6~~ 10 | ~~23~~ ~~[illegible]~~ 37 | 6 | TIME2 |
| ~~7~~ 11 | PARAMS | 7 | 28 |
| ~~8~~ 12 | ~~26~~ ~~[illegible]~~ 40 | 8 | |
| ~~9~~ 13 | RATEIN | | |
| ~~10~~ 14 | ~~44~~ ~~44~~ 58 | | |
| ~~11~~ 15 | INDEXIN | | |
| ~~12~~ 16 | ~~50~~ ~~60~~ ~~[illegible]~~ 64 | | |
| ~~13~~ 17 | RATEOUT | | |
| ~~14~~ 18 | ~~56~~ ~~66~~ ~~[illegible]~~ 70 | | |
| ~~15~~ 19 | INDEXOUT | | |
| ~~16~~ 20 | ~~62~~ ~~72~~ ~~[illegible]~~ 76 | | |
| ~~17~~ 21 | VORDER | | |
| ~~18~~ 22 | 4 | | |
| ~~19~~ 23 | TIME | | |
| ~~20~~ 24 | TEMP | | |
| ~~21~~ 25 | TAS | | |
| ~~22~~ 26 | PSFC → | | TIME1 |
| ~~23~~ ~~34~~ 37 | KUNIT | 27 | 3<br>17. |
| ~~24~~ 38 | 1 | 28 | 11. |
| ~~25~~ 39 | 4 | 29 | 40 |
| ~~26~~ 40 | PARAMS | 30 | |
| ~~27~~ 41 | 16 | 31 | TIME2 |
| ~~28~~ 42 | TIME | 32 | 3 |
| ~~29~~ 43 | 2 | 33 | 17. |
| | | 34<br>35 | 11.<br>50. |
| | | 36 | |

**SAMPLE RANDOM FILES (cont'd)**

| WORD NUMBER | WORD |
|---|---|
| ~~30~~ 44 | SEC |
| ~~31~~ 45 | F7.2 |
| ~~32~~ 46 | TEMP |
| ~~33~~ 47 | 2 |
| ~~34~~ 48 | DEG |
| ~~35~~ | F7.2 |
| ~~36~~ | TAS |
| ~~37~~ | 2 |
| ~~38~~ | M/S |
| ~~39~~ | F7.2 |
| ~~40~~ | PSFC |
| ~~41~~ | 2 |
| ~~42~~ | MB |
| ~~43~~ | F7.2 |
| ~~44~~ 58 | RATEIN |
| ~~45~~ | 4 |
| ~~46~~ | 8 |
| ~~47~~ | 8 |
| ~~48~~ | 16 |
| ~~49~~ | 16 |
| ~~50~~ 64 | INDEXIN |
| ~~51~~ | 4 |
| ~~52~~ | 98 |
| ~~53~~ | 258 |
| ~~54~~ | 130 |
| ~~55~~ | 194 |
| ~~56~~ 70 | RATEOUT |
| ~~57~~ | 4 |
| ~~58~~ | 0 |
| ~~59~~ | 0 |
| ~~60~~ | 0 |
| ~~61~~ | 0 |
| ~~62~~ 76 | INDEXOUT |
| ~~63~~ | 4 |
| ~~64~~ | 0 |
| ~~65~~ | 0 |
| ~~66~~ | 0 |
| ~~67~~ 81 | 0 |

# READLX Documentation

Description of the file READLX, created for Bonnie Gacnik on June 3, 1977

## INTRODUCTION

The FORTRAN programs in the PLIB file READLX provide the user a free-format card-input facility. There are five routines in the file: READLX (IEND), LEXCON (DAT,KEY,NUM), LXCARD, TESTLX(ID), and GETNUM (B). Only the first two of these are of interest to the user.

## THE READLX DICTIONARY

The labeled common block LEXCOM, which appears in each of the five routines in the file READLX, contains, among other things, a dictionary of user program variables (initially empty). The pertinent variables are as follows:

MLEX — The current number of dictionary entries.

NLEX — The maximum number of entries which can be made in the dictionary, set to 50 by a data statement in the routine LEXCON.

KEYS(50) — Each KEYS(I), $1 \leq I \leq MLEX$, is a Hollerith (left-justified, blank-filled) "external name" of a user program variable into which values are to be read from cards.

LDATS(50) — Each LDATS(I), $1 \leq I \leq MLEX$, is the machine address (between 0 and $65535_{10}$)of the user program variable whose external name is specified by KEYS(I).

LNUMS(50) — Each LNUMS(I), $1 \leq I \leq MLEX$, is the machine address (between 0 and $65535_{10}$) of a "counter cell" in which READLX stores the largest subscript k for which DAT(k) has been set, where DAT is the user program variable specified by KEYS(I) and LDATS(I).

Obviously, if the user declares the labeled common block LEXCOM in his program, he may make dictionary entries at will. The routine LEXCON (DAT,KEY, NUM) is provided to perform this function, however. The FORTRAN statements

```
CALL LEXCON (A,6HARRAYA,NA)
CALL LEXCON (B,6HARRAYB,NB)
```

cause two dictionary entries to be made. MLEX is incremented by 2 (unless it would then exceed NLEX, in which case execution terminates with an error message). The names 'ARRAYA' and 'ARRAYB' are placed in the KEYS array, LOC(A) and LOC(B) are placed in the LDATS array, and LOC(NA) and LOC(NB) are placed in the LNUMS array. In addition, NA and NB are initialized to zero to say that no data have so far been entered in the arrays A and B. (The latter may have undesirable effects for some users.)

FORM OF INPUT

The routine READLX is called by the FORTRAN statement

CALL READLX (IEND)

(normally after one or more entries have been made in the dictionary). It reads cards from unit 5; these cards are assumed to contain statements of the general form

$$\text{keyword}\left[\left[(\left[\text{subscript}\right])\right]\begin{bmatrix}=\\ \text{or}\\ \leftarrow\end{bmatrix}\left[\left[\text{datum}\right]\right]\left[,\left[\text{datum}\right]\right]\left[\left[\text{datum}\right]\right]\ldots\right]^{*}$$

(The brackets denote optionally-included items.) Blanks may be used as desired to improve readability, except within a keyword, a subscript, or a datum. The keyword may be one of a set reserved by READLX and having special meaning (see "Reserved Keywords" below) or an external name from the dictionary (KEYS(I), for some I such that 1<I<MLEX). The subscript, if present, is usually required to be a decimal integer (for an exception,see the "reserved keyword" "CORELOADER", below). Each datum may be a decimal integer, an octal integer (suffixed by "B"), a real number (written in any acceptable FORTRAN form), a Hollerith string of the form $nHc_1c_2c_3\ldots c_n$, a Hollerith string of the form '$c_1c_2c_3\ldots c_n$', or a Hollerith string of the form $c_1c_2\ldots c_n$ where $c_1$ is an alphabetic character and $n\leq 10$. (Note that Hollerith strings of the form $nHc_1c_2c_3\ldots c_n$ or '$c_1c_2\ldots c_n$' may have n>10, in effect specifying (n+9)/10 single-word data.) A datum may also have the form k * d, where k is a "repetition factor" and d is a datum of one of the types defined previously; the effect is as if one had repeated d k times in the list of data. Note that k*d must be punched without internal blanks. If a datum is omitted (resulting in a statement like "A=,1" or "A=1,,2", but not "A=1,2,"), no value is assigned to it; it causes an element in the array in which the data are being placed to be skipped.

A statement may begin anywhere on a card (columns 1-80 are used) and may continue over more than one card; column 1 of a "continuation card" is considered to follow column 80 of the previous card. More than one statement may be punched on a given card; multiple statements on a single card are separated by dollar signs.

Assuming that the current dictionary contains definitions of "ARRAYA" and "ARRAYB", as shown in the examples in the previous section, the following are possible statements to be read by READLX:

*NOTE: A colon (a 2/8 punch on a card, printed as a ":") may be used in place of a comma to separate two data.

| Statement | Effect |
|---|---|
| ARRAYA 1. | Stores 1. in A(1). Equals sign may be omitted, but blank must then separate keyword and datum. Leaves NA=1. |
| ARRAYB = 1.,2. | Stores 1. in B(1) and 2. in B(2). Leaves NB=2. |
| ARRAYA(3)=-1, -2 | Stores -1 in A(3) and -2 in A(4). Leaves NA=4. |
| ARRAYB(3)=77B, ,-77B | Stores $77_8$ in B(3) and $-77_8$ in B(5). B(4) is skipped. Leaves NB=5. |
| ARRAYB( ) = 136.E14 | Stores 136 x $10^{14}$ in B(6). Note that if the subscript is omitted, NB+1 is used. Leaves NB=6. |
| ARRAYA = ABCD | Stores 10HABCD in A(1). Leaves NA=4. (A(4) was previously set.) |
| ARRAYB = 14HABCDEFGHIJKLMN | Stores 10HABCDEFGHI) in B(1) and 10HKLMN in B(2). Leaves NB=6. (B(6) was previously set.) |
| ARRAYA = 100* 'HOLLERITH' | Stores 10HHOLLERITH in A(1) through A(100). Leaves NA=100. |
| ARRAYA = 10* -1 | Stores -1 in A(1) through A(10). Leaves NA=100. |

RESERVED KEYWORDS

The following keywords are reserved for the exclusive use of READLX and may not be entered as external names in the dictionary: "COMMENT", "ENDOFREAD", "ENDOFCASE", "ENDOFDATA", "CLEARINPUT", "CORELOADER", and "MODIFYSKIP". These are described as follows:

| | |
|---|---|
| COMMENT (or "/") | The remainder of the card is ignored. Note that there must be at least one blank following "COMMENT" (or "/"). |
| ENDOFREAD<br>ENDOFCASE<br>ENDOFDATA | Cause READLX to execute a "return", with IEND=1,2, or 3 |
| CLEARINPUT | Causes READLX to clear (zero) all data entered in arrays defined in the current dictionary and the "counter cells" for those arrays. |

CORELOADER (subscript) = [datum] [,[datum]] [,[datum]] ....,

where the subscript is an octal constant, not suffixed by a "B", causes READLX to transfer the given data into core, starting at the location specified by the subscript.

MODIFYSKIP = n — where n is an integer, causes the value of n to be stored as the value of the variable NSKIP in the common block LEXCOM. (See "Special Features", below.)

SPECIAL FEATURES (MODIFYSKIP)

In all of the above discussion, it was assumed that READLX stored lists of data in consecutive core locations in the associated array. This is not strictly true. The variable NSKIP, in the labeled common block LEXCOM, which is set by a data statement in READLX to 1, actually controls the increment from one datum store to the next. One may set NSKIP to a value other than 1, either by accessing it directly in the labeled common block or by an input card statement "MODIFYSKIP=n". This is useful to input data to a doubly-dimensional array by now, rather than by column. Consider the following example:

```
DIMENSION A(3,3)
  .
  .
  .
  .
  .
CALL LEXCON (A, 6HARRAYA, NA)
CALL READLX (IEND)
```

Now, if A represents the matrix

```
 1. 2. 3.
 4. 5. 6.
7. 8. 9.
```

one could use the input statement

```
A=1.,4.,7.,2.,5.,8.,3.,6.,9.
```

but it may be desirable to use the statements

```
MODIFYSKIP=3
A(1) = 1.,2.,3.          (sets "A(1)","A(4)", and "A(7)"; NA=7)
A(2) = 4.,5.,6.          (sets "A(2)","A(5)", and "A(8)"; NA=8)
A(3) = 7.,8.,9.          (sets "A(3)","A(7)", and "A(9)"; NA=9)
```

instead. Note that the notation "A( )" really means "A(NA+NSKIP)", rather than "A(NA+1)", as was stated in a previous example.

SPECIAL FEATURES (CATCHALL)

Normally, if READLX finds a keyword that it does not recognize, it issues an error message and (eventually) terminates execution. If, however, KEYS(1) contains the name 'CATCHALL', it will instead make a dictionary entry for the unrecognized keyword and assign it space in the array associated with CATCHALL by the dictionary. An example will make the use of this feature clear, I think. The statement

```
CALL LEXCON (CATCH,  8HCATCHALL, NCATCH)
CALL LEXCON (A,  6HARRAYA, NA)
CALL READLX (IEND)
```

with the data statements

```
ARRAYA = 1.,2.
ARRAYB = 3.,4.
ARRAYC = 5.,6.
ARRAYA( )=7.,8.
```

will store A(1)=1., A(2)=2., A(3)=7., A(4)=8., and NA=4, as always. In addition, however, NCATCH will be set to 8 and (CATCH(I), I=1,8) will contain the following information:

```
CATCH(1) = 10HARRAYB    ,CATCH(2) = 2 , CATCH(3) = 3.,
                                    (no. of entries)
       and CATCH(4) = 4.

CATCH(5) = 10HARRAYC    ,CATCH(6) = 2, CATCH(7) = 5., and

           CATCH(8) = 6.
```

This feature allows one to use the READLX scanner to read arrays whose names are not known before card-read time and get them back in a useful form.

Note that, if KEYS(1)=8HCATCHALL when READLX is called, it zeroes the "counter cell" for the associated array ("NCATCH" in the example). Note also that entries made in the dictionary for arrays sent to CATCHALL are cleared before READLX returns control to the caller; the length of the dictionary is not permanently increased.

One warning: If n arrays have been sent to CATCHALL, no data statement may attempt to increase the length of any one of the first n-1 sent there; the nth may be added to at will. (Only one dictionary slot is used for such arrays; a second reference to one of the first n-1 acts as if that array were never seen before.)

ADDITIONAL NOTES

The same array may be given two different "external names" but, if two different "counter cells" are associated with these names, the notation "keyword( )" may have surprising results.

If the "counter cells" for a given set of arrays are not of interest, a common counter cell may be used for all of them. In this case, however, the notation "keyword ( )" should not be used and the statement "CLEARINPUT" should not be used.

Signed, sealed, and delivered by

D. Kennison

P.S. (Form of an "external name" and a keyword)

An "external name" in the dictionary is limited to ten or fewer characters in length. A keyword on a data card may be longer than ten characters; in this case, only the first ten are actually used in searching the dictionary. For example, the LEXCON call

```
CALL LEXCON (X, 16HDATA-FOR-ARRAY-X,NX)
```

would place the single word 10HDATA-FOR-A in the dictionary. The following data-card statements would then be equivalent:

```
(ten characters used)
DATA-FOR-ARRAY-X = 1.,2.
DATA-FOR-ARRAY-Y = 1.,2.
DATA-FOR-A       = 1.,2.
DATA-FOR-ARRANGEMENT-OF-FLOWERS = 1.,2.
```

Note that an "external name" or a keyword may contain any characters other than a blank, an equals sign, or a left parenthesis, any of which terminate the keyword scan, activating a scan for the next non-blank, the first datum, or a subscript, respectively.

P.P.S. (Comments on data cards)

There are several ways to put comments on data cards. First, if a statement begins with the keyword "COMMENT" or the keyword "1", the rest of the card is ignored and scanning resumes with a search for a keyword on the next card. Thus, one could use all of the following:

```
A=1. $ B=2. $ C=3. $ COMMENT SET A, B, AND C

A=1. $ B=2. $ C=3. $ / SET A, B, AND C

/ THIS ENTIRE CARD IS A COMMENT.
```

Another alternative is to simply punch the comment fllowing the last statement on a card (with at least one intervening blank). For example:

```
A=1. $ B=2. $ C=3.    SET A,B,C
```

READLX picks up the "S" of "SET" while looking for a comma (to signal the presence of another datum for entry in C) or a dollar sign (to signal the presence of another statement on the card). Instead of treating this as an error, it simply reads up the next card and begins scanning for the keyword of a new statement. Note that this prevents the final statement on the current card from being continued on the new card. To handle this case, a third option is provided: If a slash is encountered while scanning a statement, the rest of the card is ignored and scanning resumes on the next card. For example:

```
A=1. $ B=2. $ C=3.  /  SET A, B, and C
D = 1., 2., 3., 4., 5.,  /  SET (D(I),I=1,5)
    6., 7., 8., 9., 10.  /  SET (D(I),I=6,10)
    ,11., 12., 13., 14., 15.  /  SET (D(I),I=11,15)
E(1) =                   /  SET E ARRAY BELOW
    A1, A2, A3, A4, A5   /  END OF EXAMPLE
```

There is one restriction; the subscript and/or the equals sign (if any) of a keyword must appear on the same card as the keyword. Note that the slashes on the first and fourth data cards in th  above example could be omitted; I prefer to use them - for the sake of consistency and to prevent errors caused by mistakenly omitting a needed slash.