Technical Report No. 219

SIMCOMP VERSION 3.0

MAINTENANCE DOCUMENT

K. L. Stevens and J. D. Gustafson

Natural Resource Ecology Laboratory

Colorado State University

Fort Collins, Colorado 80523

GRASSLAND BIOME

U.S. International Biological Program

June 1973

TABLE OF CONTENTS

Page

ABSTRACT

This manual is the programmer's maintenance document for the SIMCØMP Version 3.0 simulation software. The description and usage of the language is contained in "SIMCØMP Version 3.0 User's Manual." The design and internal operation of the software is described in this document. Each logical segment of the compiler and the simulation program is described by the aid of flow charts and listings of the code.

INTRODUCTION

This document is intended to expand upon and further explain the operation of the SIMCØMP 3.0 compiler beyond the documentation which is contained in the commentary information contained in the source code. Each step of execution in both the compilation and execution stages is explained in the following format:

(1) Flow chart

(2) Overview of operation

(3) Listing of relevant code

(4) Line by line commentary information

Although this format does not lend itself to easy reading, the objective of providing a text which can be used as a reference to look up the explanation of any portion of the system is accomplished. This text must be used in conjunction with a complete listing of the simulation package and Technical Report No. 218 (Gustafson and Innis 1973).

SIMCØMP was designed using many of the features of the Control Data Corporation 6400 computer using the SCØPE 3.3 operating system. A knowledge of this operating system is essential in understanding the operation of some features of SIMCØMP.

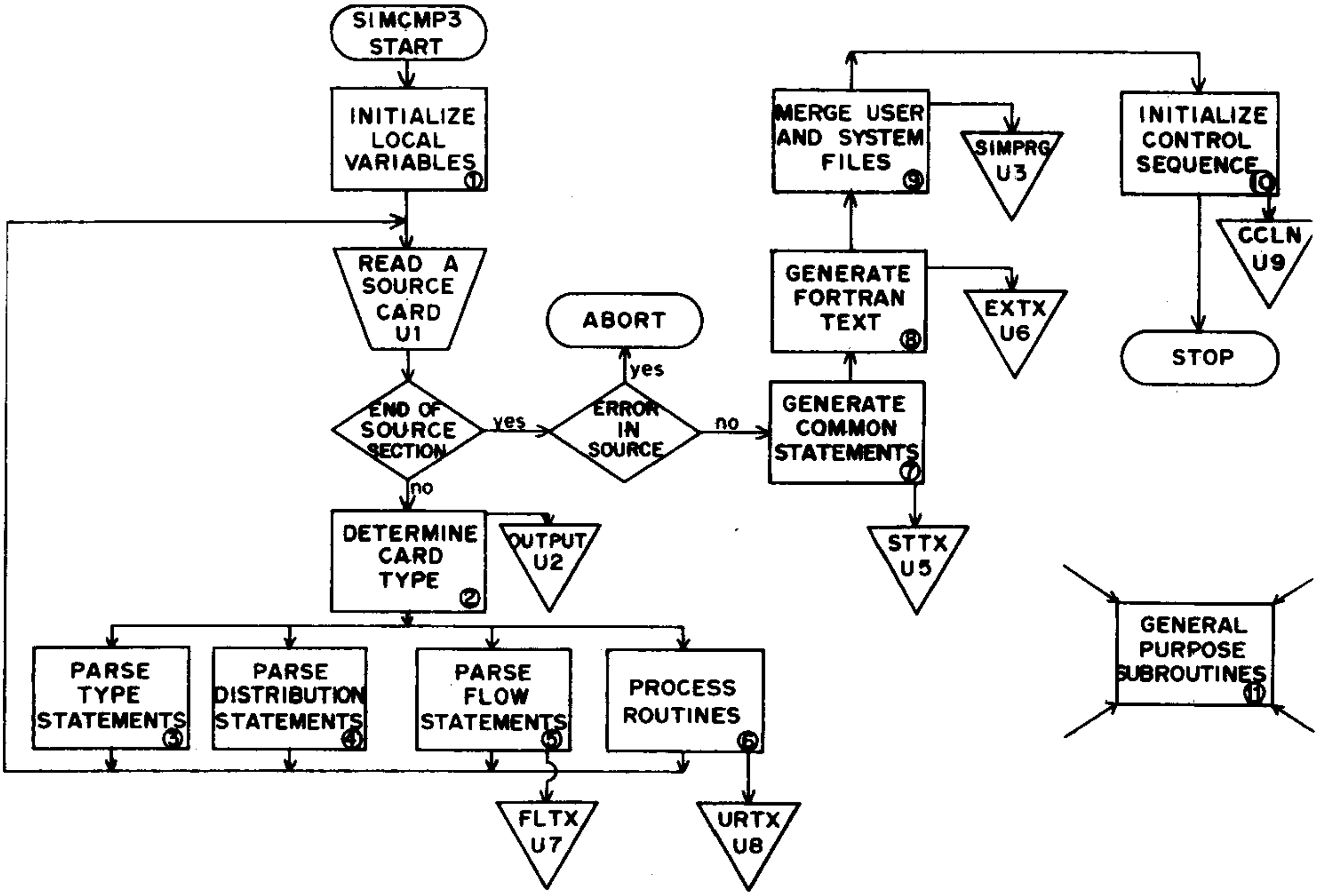
While this document is broken into four major chapters, SIMCØMP is best understood as being comprised of two major areas:

(I) Compilation of the source program (Chapter 1).

(II) Execution of the FØRTRAN object code--the simulation (Chapters 2, 3, and 4).

PART I

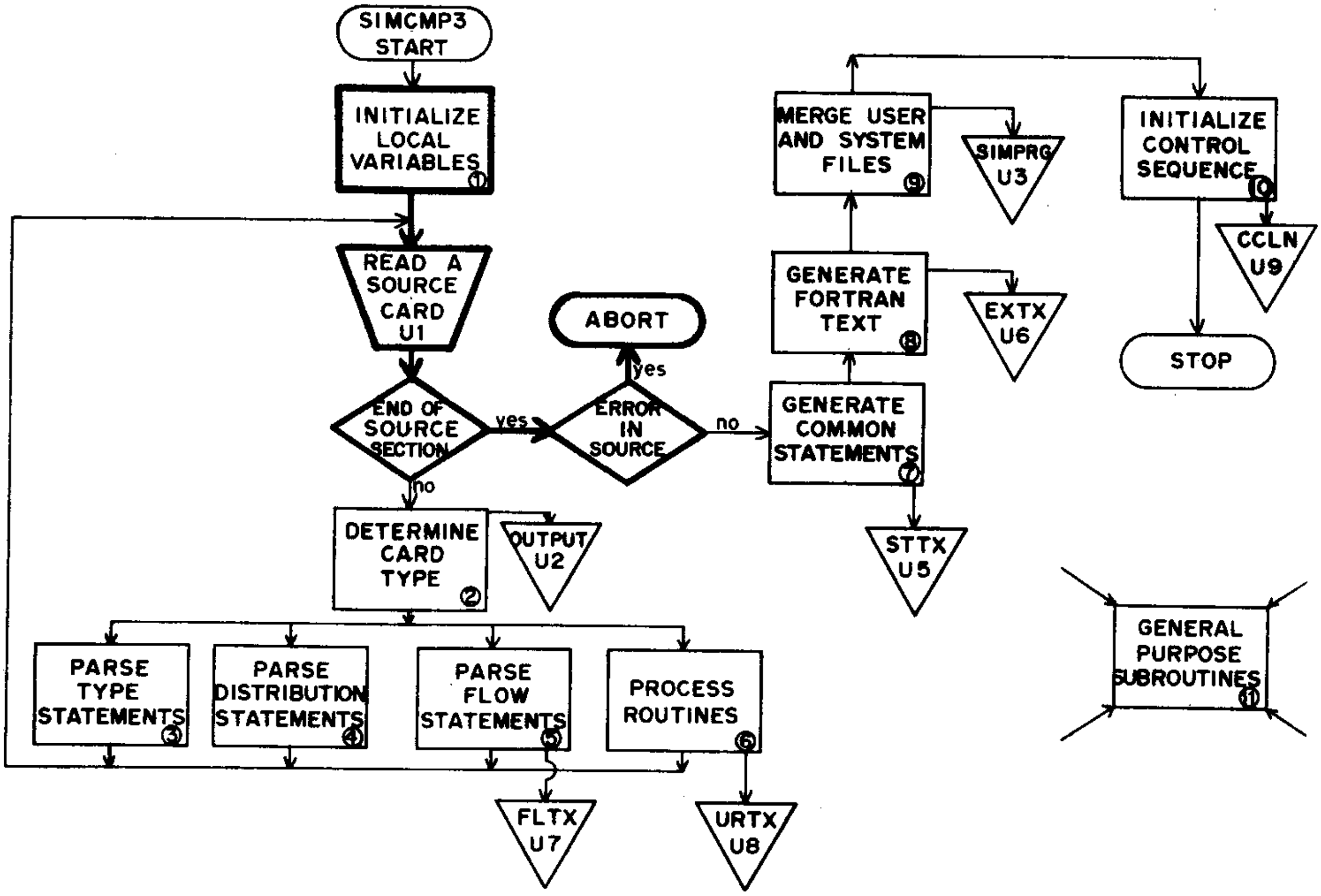
COMPILER OVERVIEW



The SIMCØMP compiler processes a SIMCØMP source deck ultimately producing a FØRTRAN program which in turn is compiled and executed. The compiler is actually a one-pass preprocessor which recognizes SIMCØMP directives which are interspersed with FØRTRAN compilable text. The various stages of compilation are diagramed in the preceding flow chart and are each subdivided and explained throughout Chapter 1.

CHAPTER 1. COMPILER OPERATION

1.1. Initialization and Card Reading



Overview

The first section of SIMCMP3 initializes compiler control variables and reads a card from the source deck. System variables are assigned values and the files developed by the compiler are assigned unit numbers.

```
      PROGRAM SIMCMP3 (INPUT=64,OUTPUT=64,SIMPRG=64.SIMCOM=64,STTX=64,EX
     1TX=64,FLTX=64,URTX=64.CCLN=64,TAPE1=INPUT.TAPE2=OUTPUT.TAPE3=SIMPR
     2G.TAPE4=SIMCOM,TAPE5=STTX.TAPE6=EXTX,TAPE7=FLTX.TAPE8=URTX.TAPE9=C
     3CLN)
      COMMON /STORAGE/ NVAR.LVR1(999),LVR2(999),NSTOR
      COMMON /ROUTINS/ NSUB.NSBL(100),SUBFLG(4),KDIST
      COMMON NFLW.NRFL.NFMAX.NFLT(1)
      COMMON /OUTP/ NLINF,NPAGE,WHEN,PRINT,NOGO,DEBUG
      COMMON /UNITS/ U1,U2.U3.U4,U5,U6,U7,U8
      DIMENSION CARD(8). COMAND(8), TEXT(8)
      INTEGER U1.U2.U3,U4.U5,U6.U7,U8
      LOGICAL FATAL.SUBFLG,PRINT,NOGO,DEBUG
      LOGICAL IFTYPE
C
C       S I M C O M P   V E R S I O N   3.0      -      FEBRUARY 1972
C         FLOW ORIENTED CONTINUOUS/EVENT SYSTEM SIMULATION COMPILER.
C.....NATURAL RESOURCES ECOLOGY LABORATORY - USIBP, COLORADO STATE UNI-
C.....VERSITY.  IMPLEMENTED ON A CONTROL DATA CORP. 6400 COMPUTER UNDER
C.....THE SCOPE 3.3 OPERATING SYSTEM.
C            DESIGN    -   JON GUSTAFSON
C            PROGRAM   -   JON GUSTAFSON
C                      -   KIM STEVENS
C
C
C.....STORAGE FILE ASSIGNMENTS...
C.....NAME     (SIZE)   UNIT  VAR.          PURPOSE
C.....INPUT     (64)     1     U1     CONTAINS USER SOURCE STATEMENTS.
C.....OUTPUT    (64)     2     U2     PRINTED OUTPUT FILE.
C.....SIMPRG    (64)     3     U3     CONTAINS GENERATED FTN PROGRAM.
C.....SIMCOM    (64)     4     U4     SYSTEM SUPPLIED TEXT FILE.
C.....STTX      (64)     5     U5     USER VARIABLE DECLARATIONS AND EXT
C.....EXTX      (64)     6     U6     USER GENERATED FTN TEXT.
C.....FLTX      (64)     7     U7     FLOW DEFINITION TEXT.
C.....URTX      (64)     8     U8     USER GENERATED EXTERNAL ROUTINES.
C.....CCLN      (64)     9     U9     RECORD OF EXECUTION CONTROL CARDS.
C
C.....INITIALIZE THE COMPILER CONTROL VARIABLES.
C
      U1=1
      U2=2
      U3=3
      U4=4
      U5=5
      U6=6
      U7=7
      U8=8
      SUBFLG(4)=.TRUE.
      SUBFLG(3)=SUBFLG(4)
      SUBFLG(2)=SUBFLG(3)
      SUBFLG(1)=SUBFLG(2)
      PRINT=.TRUE.
      NOGO=.FALSE.
      DEBUG=.FALSE.
      FATAL=.FALSE.
      NLINE=60
      NPAGE=0
      WHEN=DATE(TODAY)
      NVAR=9
      NSTOR=1008
      LVR1(1)=30010422230000012000B
      LVR2(1)=00040000000000000000B
      LVR1(2)=24111505550000022000B
      LVR2(2)=0
      LVR1(3)=24232422240000032000B
      LVR2(3)=0
      LVR1(4)=24051604550000042000B
      LVR2(4)=0
      LVR1(5)=04245555550000052000B
      LVR2(5)=0
      LVR1(6)=04242022550000062000B
      LVR2(6)=0
      LVR1(7)=04242014550000072000B
      LVR2(7)=0
      LVR1(8)=04240614550000102000B
      LVR2(8)=0
      LVR1(9)=30555555550000112000B
      LVR2(9)=76340000000000000000B
      WRITE (U5,105)
      NSUB=7
      KTYPE=0
      KDIST=0
      NSBL(1)=30032311150000000000B
      NSBL(2)=30202216240000000000B
      NSBL(3)=30201417240000000000B
      NSBL(4)=30061417200000000000B
      NSBL(5)=23240122240000000000B
      NSBL(6)=06111611230000000000B
      NSBL(7)=10011424550000000000B
      NFLW=0
      NRFL=0
      CALL FLCOR (NFMAX,NCORE)
      CALL REMARK (16H   READING INPUT)
C
C.....THE SOURCE SECTION IS READ IN AND PROCESSED CARD BY CARD.  THE TYP
C.....OF SOURCE CARD IS DETERMINED BY ROUTINE "CARDTP".  TYPE OF CURRENT
C.....SOURCE CARD IS "KTYPE", THE TYPE OF THE PREVIOUS CARD IS "JTYPE".
C
   15 READ (U1,90) CARD
      IF (EOF(U1)) 65,20,65
   20 JTYPE=KTYPE
      IFTYPE=.FALSE.
C
C.....DETERMINE CARD TYPE.
C
      CALL CARDTP (CARD,KTYPE,JTYPE,TEXT,COMAND,FATAL), RETURNS(15)
      IF (KTYPE.NE.6.A.(JTYPE.EQ.5.O.JTYPE.EQ.6)) CALL FL2DF (CARD,FATAL
     1)
      GO TO (30,25,25,35,40,45,50,55,60), KTYPE
C
C.....<INTEGER.> OR <REAL.>
C
   25 IFTYPE=.TRUE.
C
C.....<STORAGE.>...
C
   30 CALL ST1DF (TEXT,KTYPE,FATAL,IFTYPE)
      GO TO 15
C
C.....<UNIFORM.>, <NORMAL.>, <EXPONENT.>, OR <LOGNORMAL.>...
C
   35 CALL US1DF (TEXT,FATAL)
      GO TO 15
C
C.....<FLOW>...
C
   40 CALL FL1DF (COMAND,FATAL)
      WRITE (U7,90) TEXT
      GO TO 15
C
C.....<FLOW TEXT>...
C
   45 WRITE (U7,90) CARD
      GO TO 15
C
C.....<SUBROUTINE>, <FUNCTION>, OR <EVENT>...
C
   50 WRITE (U8,90) TEXT
      GO TO 15
C
C.....<CONTUATION OF KTYPE=7 OR 8>...
C
   55 WRITE (U8,90) CARD
      GO TO 15
C
C.....<ROUTINE TEXT>...
C
   60 IF (JTYPE.NE.7.A.JTYPE.NE.8) GO TO 55
      END FILE U8
      WRITE (U8,95)
      GO TO 55
```

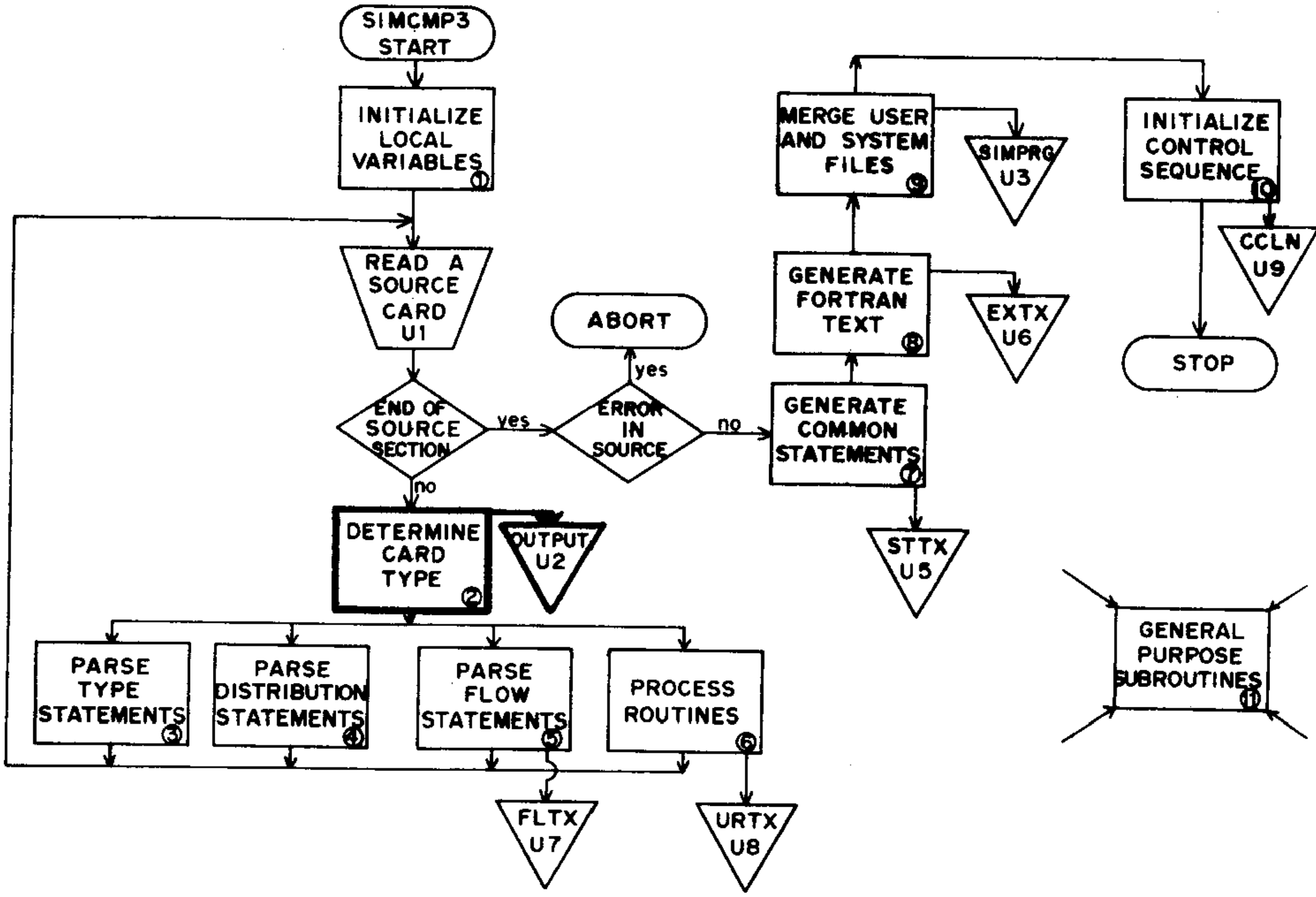
```
10150          65 IF (KTYPE.EQ.5.O.KTYPE.EQ.6) CALL FL2DF (CARD,FATAL)
10151             IF (.N.FATAL) GO TO 70
10152       C
10153       C.....FATAL ERRORS HAVE BEEN DETECTED IN THE SOURCE SECTION.
10154       C
10155             WRITE (U2,100)
10156             CALL REMARK (24H   FATAL ERROR IN SOURCE)
10157             CALL ABORT
10158          70 CALL REMARK (18H   GENERATING CODE)
10159             CALL GCOMMON
10160             CALL TX1DF
10161       C
10162       C.....GENERATE THE SIMULATION PROGRAM BY MERGING THE USER GENERATED FILE
10163       C.....WITH THE SYSTEMS TEXT FILE.
10164       C
10165             CALL POSITN (U4.2)
10166          75 CALL TRNSFR (U4,U3.NUNIT.NFILE)
10167             IF (NUNIT.LE.0) GO TO 85
10168             CALL POSITN (NUNIT.NFILE)
10169          80 CALL TRNSFR (NUNIT.U3.MUNIT,MFILE)
10170             IF (MUNIT.LE.0) GO TO 75
10171             CALL POSITN (MUNIT.MFILE)
10172             CALL TRNSFR (MUNIT.U3,LUNIT.LFILE)
10173             GO TO 80
10174          85 CONTINUE
10175             REWIND U3
10176             CALL REMARK (29H   SOURCE PROCESSING FINISHED)
10177             NCORE=NCORE+NFLW
10178             ENCODE (10,110,MESG) NCORE
10179             CALL DISPLY (MESG)
10180             IF (NOGO) STOP
10181             IF (DEBUG) WRITE (9.115)
10182             IF (.NOT.DEBUG) WRITE (9.120)
10183             REWIND 9
10184             CALL CCLTR
10185             STOP
10186       C
10187          90 FORMAT (8A10)
10188          95 FORMAT (15HC++++    5    0,65X)
10189         100 FORMAT (1H0,//,30(1H*), 12HFATAL ERRORS,30(1H*))
10190         105 FORMAT (6X, 56HCOMMON XADRS(1),TIME,TSTRT,TEND,DT,DTPR,DTPL,DTFL,X
10191            1(999),18X)
10192         110 FORMAT (O10)
10193         115 FORMAT (41HFTN,I=SIMPRG,ROUND=+-*/,S=0,LN=DEBUG,R=1.    /35HATTACH
10194            1,B,SIMCOM3,CY=2,MR=1,ID=NREL.    /37HATTACH,LIB,SIMCOM3,CY=3,MR=1,
10195            2ID=NREL.    /7HSELECT.    /13HCOPYBF,B,LGO.    /15HLOADER,PPLOADR.
10196            3    /9HMAP.PART.    /9HLOAD,LGO.    /5HNOGO.    /13HREWIND,NEWT1.
10197            4    /27HSELECT,P=PRELOAD,I=PRELOAD.    /19HPRELOAD,NEWT1,MAIN.    /
10198            55HMAIN.)
10199         120 FORMAT (34HFTN,I=SIMPRG,ROUND=+-*/,S=0,LRN=0.    /35HATTACH,B,SIMC
10200            1OM3,CY=2,MR=1,ID=NREL.    /37HATTACH,LIB,SIMCOM3,CY=3,MR=1,ID=NREL
10201            2.    /7HSELECT.    /13HCOPYBF,B,LGO.    /15HLOADER,PPLOADR.    /8H
10202            3MAP,OFF.    /9HLOAD,LGO.    /5HNOGO.    /13HREWIND,NEWT1.    /27HS
10203            4ELECT,P=PRELOAD,I=PRELOAD.    /19HPRELOAD,NEWT1,MAIN.    /5HMAIN.)
10204       C
10205             END
```

Line Number	Explanation
10024-10045	Files 1-9 are manipulated by SIMCØMP. U1 (INPUT) contains the input to the SIMCØMP compiler, the user source deck. U2 (ØUTPUT) is the output file printed by the line printer (created in Section 1.2). U3 (SIMPRG) contains the FØRTRAN executable program generated from the source deck by the SIMCØMP compiler (U3 is formed in Section 1.9).

Line Number	Explanation
	U4 (SIMCØM) contains system supplied text. The first file of U4 contains the SIMCØMP compiler, the second through the fifth files contain dummy routines which are used in the processing of distribution statements (Section 1.4), the sixth through the fourteenth files contain system supplied text which is copied onto U3 by Section 1.9. U5 (STTX) contains user variable declarations generated in Section 1.7. U6 (EXTX) contains the FØRTRAN executable text generated from the source deck (Section 1.8). U7 (FLTX) is generated by Section 1.5 and contains flow definition text. U8 (URTX) contains user generated external routines (Section 1.6). U9 (CCLN) contains execution control cards created by Section 1.10.
10046-10049	SUBFLG is manipulated by section 1.2; a value of .FALSE. indicates that the user has supplied one of the system executable routines--START, FINIS, CYCL1, CYCL2.
10050-10053	PRINT=.TRUE., then each source card is printed onto file output (Section 1.2). NØGØ=.FALSE., then execution control cards are generated by Section 1.10. (Section 1.2 is where NØGØ is altered.) DEBUG=.FALSE., then the control cards which load the SIMCØMP debugging feature are *not* generated by Section 1.10. FATAL=.TRUE., if a syntax error in the source deck is detected by any compiler section. Execution will halt after the source deck has been completely processed.
10054-10056	These variables are used by general purpose subroutine FMTPG to page format file ØUTPUT.
10057-10076	The nine system variables are defined: NVAR represents the number of declared variables. NSTØR is the relative starting location for the next variable to be declared. LVR1 and LVR2 are variable stacks containing the name, relative starting location, mode, and subscripts of each declared variable (see Section 1.3). The nine system variables are: XADRS(1), TIME, TSTRT, TEND, DT, DTPR, DTPL, DTFL, and X(999).

Line Number	Explanation
10077	U5 contains FØRTRAN executable CØMMØN card images reserving central memory for all system and user declared variables (Section 1.7).
10078-10087	The seven system events are defined. NSUB represents the number of defined events. NSBL contains the names of the system events: XCSIM, XPRNT, XPLØT, XFLØP, START, FINIS, and HALT. (Section 1.2 places each user defined event into NSBL as it is encountered.)
10078-10079	KTYPE is the current source card type (a value from 1 to 8 determined by Section 1.2). KDIST takes on a value from 1 to 4 if a distribution statement (UNIFØRM., NØRMAL., EXPØNENT., or LØGNØRMAL.) is encountered by Section 1.2.
10088-10089	NFLW is the total number of expanded flows processed from flow commands encountered in the source deck (Section 1.5). NRFL is the total number of flow commands containing DØ expressions (Section 1.5).
10090	FLCØR is a general purpose subroutine which calculates the amount of available storage for flow tables.
10092-10205	Code is explained in the appropriate subsection.

1.2. *Determine Card Type*



Overview

Each source card is parsed by this section to determine its type. Each different type of source card encountered assigns a unique value to variable KTYPE. (KTYPE later causes control to flow to the section where that particular type of card is processed by SIMCØMP.) The following values of KTYPE are assigned for each type of source card.

Source Card	Starting Column	KTYPE
STØRAGE. <var decl list>	1-5	1
INTEGER. <list>	1-5	2
REAL. <list>	1-5	3
UNIFØRM. <list>	1-5	4(KDIST=1)
NØRMAL. <list>	1-5	4(KDIST=2)
EXPØNENT. <list>	1-5	4(KDIST=3)
LØGNØRMAL. <list>	1-5	4(KDIST=4)
(<I>-<J>). <flow text>	1-5	5
<flow text>	6 or 7 --	6
SUBRØUTINE <name,args>	7--	7
FUNCTIØN <name,args>	7--	7
EVENT <name>	7--	7
<continuation of KTYPE=7 OR 8>	6	8
<routine text>	6 or 7	9
C <comment text>	1	No change
LIST.	1-5	No change
NØLIST.	1-5	No change
NØGØ.	1-5	No change
DEBUG.	1-5	No change

The input to the section is a source card.

If the source card is of KTYPE=1-5, the card is separated into CØMAND and TEXT portions. CØMAND contains the SIMCØMP recognizable command, STØRAGE., (<I>-<J>)., etc, while TEXT contains the remainder of the card, <var decl list>, <flow text>, etc. However, if KTYPE=6-9, then the entire card is filled into TEXT.

EXAMPLE. The source card STØRAGE.A,B will cause

KTYPE=1, CØMAND=10HSTØRAGE., and TEXT=10HA,B

If an EVENT source card is encountered, then the routine name is retrieved and placed into a stack, NSBL(NSUB) where NSUB is the total number of names in NSBL, containing the names of all events encountered.

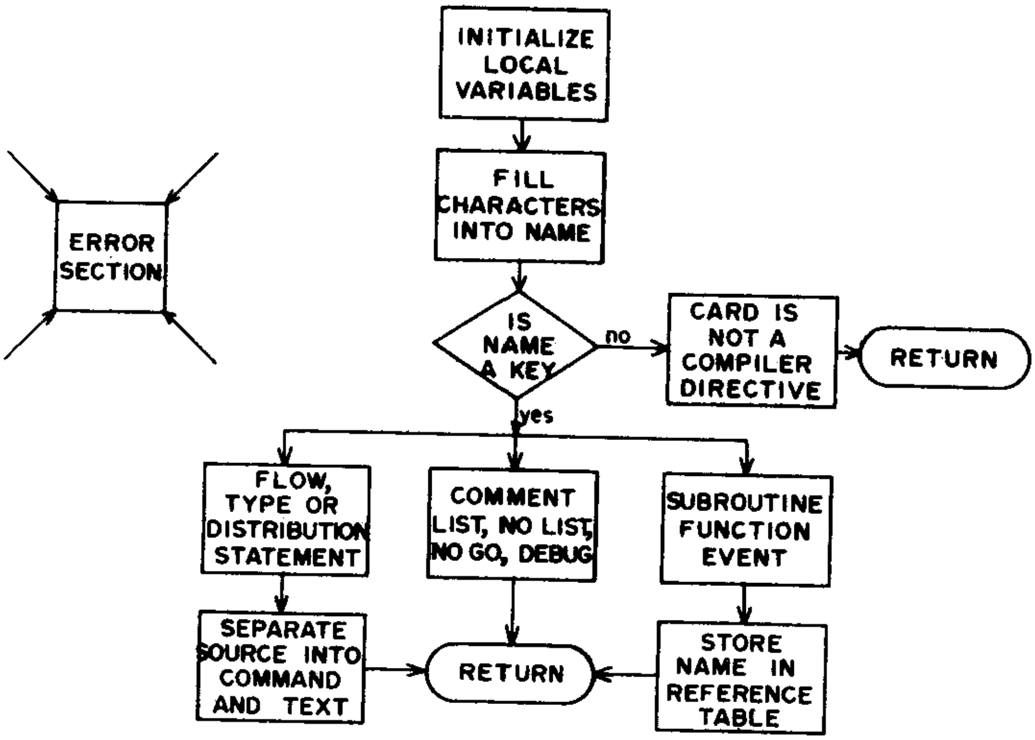
EXAMPLE. The source card EVENT ANT causes

KTYPE=7, TEXT=14HSUBRØUTINE ANT,

NSUB=NSUB+1, and NSBL(NSUB)=10HANT.

The source card is written on the output file, U7, unless a NØLIST. card is encountered. Thus, section output consists of a value for KTYPE and possibly entries in CØMAND, TEXT, NSBL, and U7. The section is broken into subsections for easier analysis.

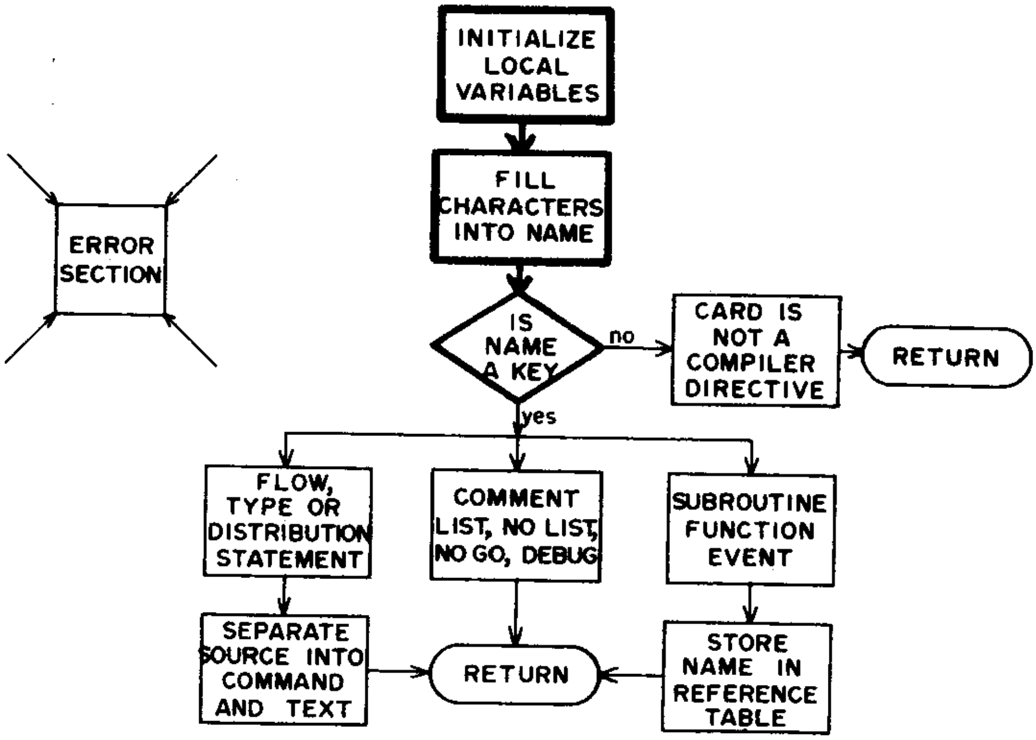
Card type flow chart


INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
yes
CARD IS NOT A COMPILER DIRECTIVE
RETURN
ERROR SECTION
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

```
      SUBROUTINE CARDTP (CARD,KTYPE,JTYPE,TEXT,COMAND,FATAL),RETURNS(M)
      COMMON /ROUTINS/ NSUB,NSBL(100),SUBFLG(4),KDIST
      COMMON /OUTP/ NLINE,NPAGE,WHEN,PRINT,NOGO,DEBUG
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), TEXT(8), COMAND(8), KEY(16), KEY1(4)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL TRIP,SUBFLG,PRINT,CONT,FATAL,NOGO,DEBUG
C
C.....THIS ROUTINE DETERMINES WHAT TYPE OF SOURCE CARD HAS BEEN ENCOUNT-
C.....ERED AND SETS "KTYPE" TO AN APPROPRIATE VALUE:
C
C         CARD FORMAT                           STARTING COLS.    KTYPE
C.....STORAGE. <VAR DECL LIST>                       1-5            1
C.....INTEGER. <LIST>                                1-5            2
C.....REAL. <LIST>                                   1-5            3
C.....UNIFORM. <LIST>                                1-5            4  (KDIST=1)
C.....NORMAL. <LIST>                                 1-5            4  (KDIST=2)
C.....EXPONENT. <LIST>                               1-5            4  (KDIST=3)
C.....LOGNORMAL. <LIST>                              1-5            4  (KDIST=4)
C.....(<I>-<J>). <FLOW TEXT>                         1-5            5
C.....<FLOW TEXT>                                    6 OR 7--       6
C.....SUBROUTINE <NAME, ARGS>                        7--            7
C.....FUNCTION <NAME, ARGS>                          7--            7
C.....EVENT <NAME>                                   7--            7
C.....<CONTINUATION OF KTYPE=7 OR 8>                 6              8
C.....<ROUTINE TEXT>                                 6 OR 7--       9
C.....C <COMMENT TEXT>                               1             NO CHANGE
C.....LIST.                                          1-5           NO CHANGE
```

```
C.....NOLIST.                                            1-5        NO CHANGE
C.....NOGO.                                              1-5        NO CHANGE
C.....DEBUG.                                             1-5        NO CHANGE
C
C.....INITIALIZE LOCAL CONTROL VARIABLES.
C
      DATA KEY/1H(,1HC,8HSTORAGE.,8HINTEGER.,5HREAL.,10HSUBROUTINE,8HFUN
     1CTION,5HEVENT,8HUNIFORM.,7HNORMAL.,9HEXPONENT.,10HLOGNORMAL.,5HLIS
     2T.,7HNOLIST.,5HNOGO.,6HDEBUG./,NK/16/
      DATA KEY1/5HSTART,5HFINIS,5HCYCL1,5HCYCL2/,NK1/4/
      DO 15 I=1,8
         COMAND(I)=10H
   15 TEXT(I)=10H
      ICOL=0
      NCOL=0
      NAME=10H
      NEVNT=0
C
C.....SCAN THE SOURCE CARD, COLUMN BY COLUMN, TO FIND A MATCH WITH A
C.....DIRECTIVE.
C
   20 ICOL=ICOL+1
      IF (ICOL.GT.72) GO TO 145
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 20
      NCOL=NCOL+1
      IF (NCOL.GT.10) GO TO 145
      CALL SCHARS (NAME,NCOL,1,ICHR)
      DO 25 I=1,NK
         IF (NAME.EQ.KEY(I)) GO TO 30
   25 CONTINUE
      GO TO 20
C
C.....A MATCH HAS BEEN FOUND.
C
   30 GO TO (35,40,45,45,45,50,50,60,70,70,70,70,75,80,85,90), I
C
C.....<FLOW>...
C
   35 IF (ICOL.GT.5) GO TO 145
      KTYPE=5
      GO TO 95
C
C.....<COMMENT>...
C
   40 IF (ICOL.GT.1) GO TO 145
      IF(.N.PRINT) RETURN M
      CALL FMTPG (1)
      WRITE (U2,170) CARD
      RETURN M
C
C.....<STORAGE.>, <INTEGER.>, OR <REAL.>...
C
   45 KTYPE=I-2
      GO TO 95
C
C.....<SUBROUTINE> OR <FUNCTION>...
C
   50 KTYPE=7
      DO 55 I=1,8
   55 TEXT(I)=CARD(I)
      GO TO 120
C
C.....<EVENT>...
C
   60 KTYPE=7
      NEVNT=1
      TEXT(1)=10H      SUBR
      TEXT(2)=10HOUTINE
      NCOL=17
      MCOL=ICOL
      ICOL=17
   65 MCOL=MCOL+1
      IF (MCOL.GT.72) GO TO 120
      CALL GCHARS (CARD,MCOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 65
      NCOL=NCOL+1
      CALL SCHARS (TEXT,NCOL,1,ICHR)
      GO TO 65
C
C.....<UNIFORM.>, <NORMAL.>, <EXPONENT.>, OR <LOGNORMAL.>...
C
   70 KTYPE=4
      KDIST=I-8
      GO TO 95
C
C.....<LIST.>...
C
   75 PRINT=.TRUE.
      RETURN M
C
C.....<NOLIST.>...
C
   80 PRINT=.FALSE.
      RETURN M
C
C.....<NOGO.>...
C
   85 NOGO=.TRUE.
      IF (.N.PRINT)RETURN M
      CALL FMTPG (1)
      WRITE (U2,210)
      RETURN M
C
C.....<DEBUG.>...
C
   90 DEBUG=.TRUE.
      IF (.N.PRINT)RETURN M
      CALL FMTPG (1)
      WRITE (U2,215)
      RETURN M
C
C.....SEPARATE THE SOURCE CARD INTO COMMAND AND TEXT PORTIONS AND OUTPUT
C.....THE RESULTS.
C
   95 ICOL=0
      TRIP=.FALSE.
      MCOL=6
      NCOL=0
  100 ICOL=ICOL+1
      IF (ICOL.GT.72) GO TO 110
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (TRIP) GO TO 105
      IF (ICHR.EQ.1H ) GO TO 100
      IF (ICHR.EQ.1H.) TRIP=.TRUE.
      NCOL=NCOL+1
      CALL SCHARS (COMAND,NCOL,1,ICHR)
      GO TO 100
  105 MCOL=MCOL+1
      CALL SCHARS (TEXT,MCOL,1,ICHR)
      GO TO 100
  110 IF (.N.PRINT) RETURN
      IF (KTYPE.EQ.5) GO TO 115
      CALL FMTPG (1)
      WRITE (U2,165) COMAND(1),TEXT
      RETURN
  115 CALL FMTPG (2)
      WRITE (U2,175) COMAND
      WRITE (U2,170) TEXT
      RETURN
C
C.....RETRIEVE THE ROUTINE NAME, STORE NAME IN ROUTINE REFERENCE TABLE,
C.....AND GENERATE FTN EXTERNAL STATEMENT.
C
  120 NCOL=0
      NAME=10H
  125 ICOL=ICOL+1
      IF (ICOL.GT.72) GO TO 130
      CALL GCHARS (TEXT,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 125
      IF (ICHR.EQ.1H() GO TO 130
      NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 150
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 125
  130 IF (NCOL.LE.0) GO TO 150
      DO 135 I=1,NK1
         IF (NAME.EQ.KEY1(I)) SUBFLG(I)=.FALSE.
  135 CONTINUE
      IF (NEVNT.EQ.0) GO TO 140
      NSUB=NSUB+1
      IF (NSUB.GT.100) GO TO 155
      NSBL(NSUB)=77777777770000000000B.AND.NAME
  140 IF (.N.PRINT) RETURN
      CALL FMTPG (2)
      WRITE (U2,180) CARD
      RETURN
C
C.....THE SOURCE CARD IS NOT A COMPILER DIRECTIVE.
C
  145 IF (NCOL.EQ.0) RETURN M
      IF (JTYPE.LT.5) GO TO 160
      DECODE (5,185,CARD) LABEL
      DECODE (6,190,CARD) COL6
      CONT=.FALSE.
      IF (LABEL.EQ.1H .A.COL6.NE.1H ) CONT=.TRUE.
      IF (JTYPE.EQ.5.O.JTYPE.EQ.6) KTYPE=6
      IF (JTYPE.GE.7.A.JTYPE.LE.9) KTYPE=9
      IF (CONT.A.(JTYPE.EQ.7.O.JTYPE.EQ.8)) KTYPE=8
      IF (.N.PRINT) RETURN
      CALL FMTPG (1)
      WRITE (U2,170) CARD
      RETURN
C
C.....IF AN ERROR OCCURED GENERATE A DIAGNOSTIC.
C
  150 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2,195)
      FATAL=.TRUE.
      KTYPE=7
      RETURN
  155 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2,200)
      FATAL=.TRUE.
      KTYPE=7
      RETURN
  160 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2,205)
      FATAL=.TRUE.
      KTYPE=0
      RETURN M
C
  165 FORMAT (1H ,14X,A10,5X,8A10)
  170 FORMAT (1H ,29X,8A10)
  175 FORMAT (1H ,14X,8A10)
  180 FORMAT (1H0,29X,8A10)
  185 FORMAT (A5)
  190 FORMAT (5X,A1)
  195 FORMAT (11H *****FE   , 47HROUTINE NAME LONGER THAN 5 CHARS OR ZER
     10 LENGTH)
  200 FORMAT (11H *****FE   , 41HNUMBER OF USER-DEFINED EVENTS EXCEEDS 1
     100)
  205 FORMAT (11H *****FE   , 32HABOVE CARD ILLEGAL AT THIS POINT)
  210 FORMAT (1H ,14X, 62HNOGO.          AUTOMATIC CONTROL CARD GENERATI
     1ON IS SUPPRESSED)
  215 FORMAT (1H ,14X, 50HDEBUG.         DEBUG CONTROL SEQUENCE IS REQUE
     1STED)
C
      END
```

Initialize and determine directive type


ERROR SECTION
INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
CARD IS NOT A COMPILER DIRECTIVE
RETURN
yes
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

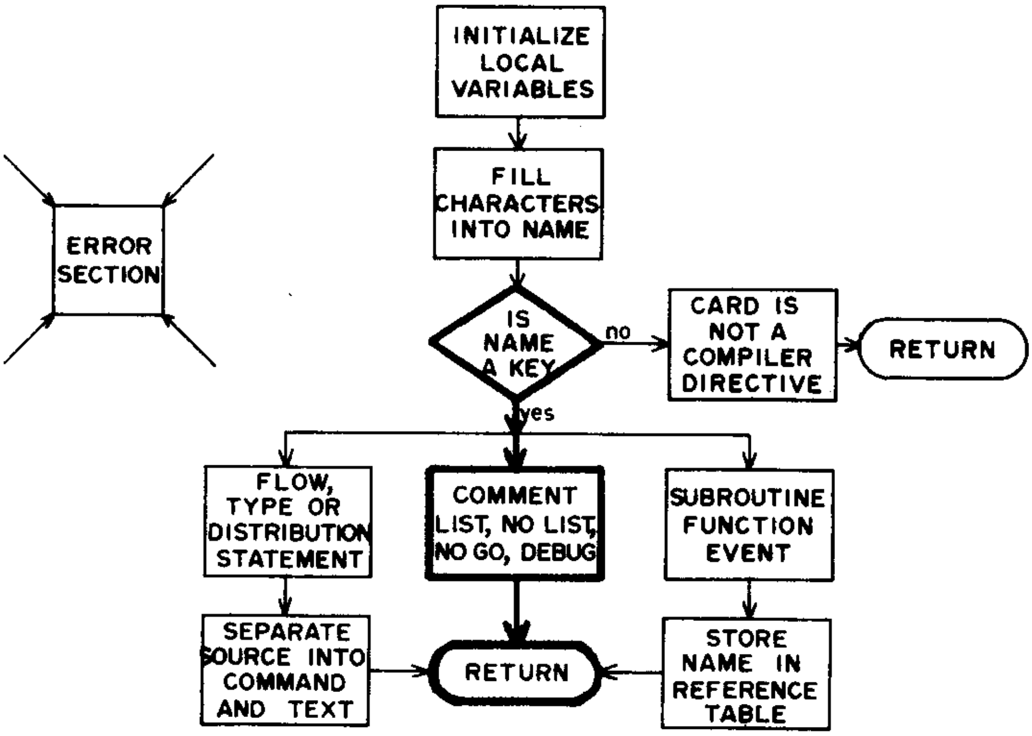
```
      SUBROUTINE CARDTP (CARD,KTYPE,JTYPE,TEXT,COMAND,FATAL),RETURNS(M)
      COMMON /ROUTINS/ NSUB,NSBL(100),SUBFLG(4),KDIST
      COMMON /OUTP/ NLINE,NPAGE,WHEN,PRINT,NOGO,DEBUG
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), TEXT(8), COMAND(8), KEY(16), KEY1(4)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL TRIP,SUBFLG,PRINT,CONT,FATAL,NOGO,DEBUG
C
C.....THIS ROUTINE DETERMINES WHAT TYPE OF SOURCE CARD HAS BEEN ENCOUNT-
C.....ERED AND SETS "KTYPE" TO AN APPROPRIATE VALUE:
C
C         CARD FORMAT                         STARTING COLS.   KTYPE
C.....STORAGE. <VAR DECL LIST>                     1-5           1
C.....INTEGER. <LIST>                              1-5           2
C.....REAL. <LIST>                                 1-5           3
C.....UNIFORM. <LIST>                              1-5           4  (KDIST=1)
C.....NORMAL. <LIST>                               1-5           4  (KDIST=2)
C.....EXPONENT. <LIST>                             1-5           4  (KDIST=3)
C.....LOGNORMAL. <LIST>                            1-5           4  (KDIST=4)
C.....(<I>-<J>). <FLOW TEXT>                       1-5           5
C.....<FLOW TEXT>                                  6 OR 7--      6
C.....SUBROUTINE <NAME, ARGS>                      7--           7
C.....FUNCTION <NAME, ARGS>                        7--           7
C.....EVENT <NAME>                                 7--           7
C.....<CONTINUATION OF KTYPE=7 OR 8>               6             8
C.....<ROUTINE TEXT>                               6 OR 7--      9
C.....C <COMMENT TEXT>                             1           NO CHANGE
C.....LIST.                                        1-5         NO CHANGE
```

```
20028          C.....NOLIST.                                            1-5         NO CHANGE
20029          C.....NOGO.                                              1-5         NO CHANGE
20030          C.....DEBUG.                                             1-5         NO CHANGE
20031          C
20032          C.....INITIALIZE LOCAL CONTROL VARIABLES.
20033          C
20034                DATA KEY/1H(,1HC,8HSTORAGE.,8HINTEGER.,5HREAL.,10HSUBROUTINE,8HFUN
20035               1CTION,5HEVENT,8HUNIFORM.,7HNORMAL.,9HEXPONENT.,10HLOGNORMAL.,5HLIS
20036               2T.,7HNOLIST.,5HNOGO.,6HDEBUG./,NK/16/
20037                DATA KEY1/5HSTART,5HFINIS,5HCYCL1,5HCYCL2/,NK1/4/
20038                DO 15 I=1,8
20039                   COMAND(I)=10H
20040             15 TEXT(I)=10H
20041                ICOL=0
20042                NCOL=0
20043                NAME=10H
20044                NEVNT=0
20045          C
20046          C.....SCAN THE SOURCE CARD, COLUMN BY COLUMN, TO FIND A MATCH WITH A
20047          C.....DIRECTIVE.
20048          C
20049             20 ICOL=ICOL+1
20050                IF (ICOL.GT.72) GO TO 145
20051                CALL GCHARS (CARD,ICOL,1,ICHR)
20052                IF (ICHR.EQ.1H ) GO TO 20
20053                NCOL=NCOL+1
20054                IF (NCOL.GT.10) GO TO 145
20055                CALL SCHARS (NAME,NCOL,1,ICHR)
20056                DO 25 I=1,NK
20057                   IF (NAME.EQ.KEY(I)) GO TO 30
20058             25 CONTINUE
20059                GO TO 20
```

Line Number	Explanation
20000	CARD contains the 80 columns of a source card. KTYPE will contain the type number of the current card. JTYPE contains the type of the previous card. TEXT is the portion of source card following a SIMCØMP command. CØMAND is the command portion of source card. FATAL is a logical flag set to .TRUE. if a syntax error is encountered. RETURNS(M) are cards which require no further processing by SIMCØMP and cause control to return to where the next source card is read in.
20031-20044	Initialize the local variables. KEY contains a list of SIMCØMP commands. NK is the number of entries in KEY. KEY1 contains a list of optional routine names that the user may specify. NK1 is the number of entries in KEY1. ICØL is the column number of source card currently being scanned. NCØL is the number of characters in NAME. NAME will contain either a SIMCØMP command or the first 10 nonblank characters of CARD. NEVNT will be set equal to 1 if an EVENT command is encountered.

Line Number	Explanation
20045-20059	The source card is scanned column by column. Each nonblank character encountered is filled into NAME, and NAME is checked against KEY. If NAME matches an entry of KEY, then the source card is a SIMCØMP command, and control flows to an appropriate segment. If there are at least 10 characters in NAME and no match with KEY, then the source card does not contain a SIMCØMP compiler directive (command).

Peripheral directives

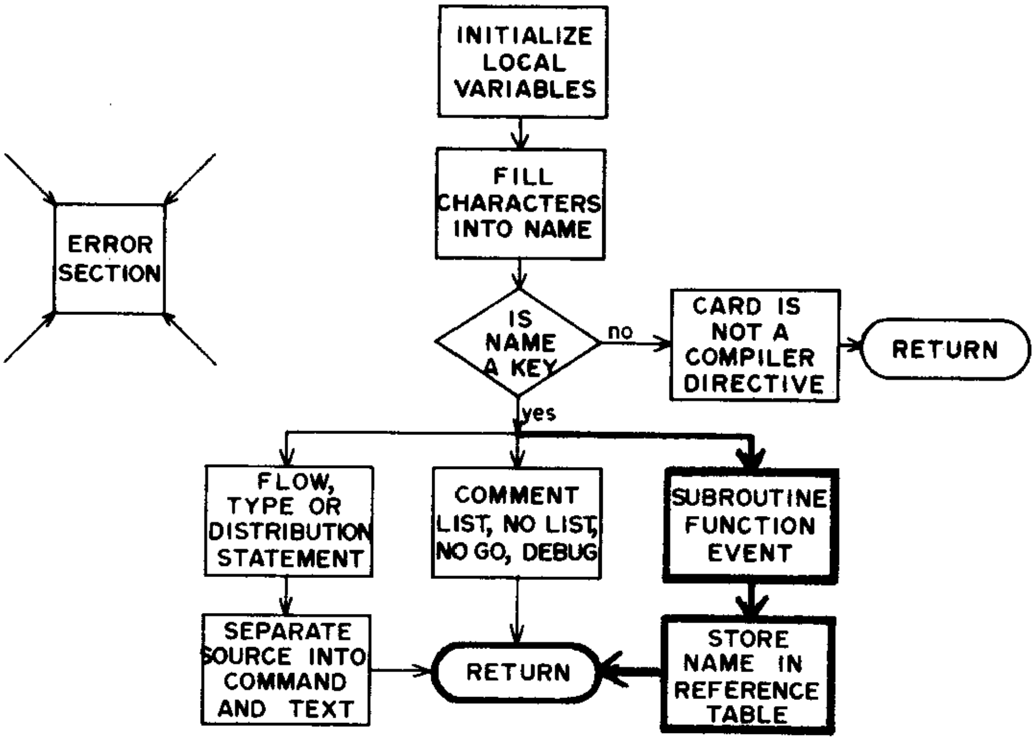

ERROR SECTION
INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
CARD IS NOT A COMPILER DIRECTIVE
RETURN
yes
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

```
      C
      C.....A MATCH HAS BEEN FOUND.
      C
         30 GO TO (35,40,45,45,45,50,50,60,70,70,70,70,75,80,85,90), I

      C
      C.....<COMMENT>...
      C
         40 IF (ICOL.GT.1) GO TO 145
            IF(.N.PRINT) RETURN M
            CALL FMTPG (1)
            WRITE (U2,170) CARD
            RETURN M

      C
      C.....<LIST.>...
      C
         75 PRINT=.TRUE.
            RETURN M
      C
      C.....<NOLIST.>...
      C
         80 PRINT=.FALSE.
            RETURN M
      C
      C.....<NOGO.>...
      C
         85 NOGO=.TRUE.
            IF (.N.PRINT)RETURN M
            CALL FMTPG (1)
            WRITE (U2,210)
            RETURN M
```

```
20131          C
20132          C.....<DEBUG.>...
20133          C
20134             90 DEBUG=.TRUE.
20135                IF (.N.PRINT)RETURN M
20136                CALL FMTPG (1)
20137                WRITE (U2,215)
20138                RETURN M
```

Line Number	Explanation
20063	A match with KEY has been found. Branch to the appropriate segment determined by index of KEY.
20070-20077	The source card is of form C <comment text>. A "C" must be in column 1 to be a comment card. PRINT is a logic flag set to .TRUE. if the user wants his source deck listed. Control returns to where next source card is read in. FMTPG is a subroutine which page-formats the source listing.
20113-20117	A LIST. card encountered indicates that user desires all cards following the LIST. card to be written onto output (PRINT set to .TRUE.).
20118-20122	A NØLIST. card inhibits following source cards from being printed onto ØUTPUT. (remains in effect until a LIST. card is encountered).
20123-20130	A NØGØ. card is encountered. Variable NØGØ is set to .TRUE. which will prohibit automatic generation of control cards.
20131-20138	A DEBUG. card is encountered; the DEBUG flag is set. This will initiate generation of debug control cards.

Subprogram directives


INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
CARD IS NOT A COMPILER DIRECTIVE
RETURN
yes
ERROR SECTION
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

```
C
C.....<SUBROUTINE> OR <FUNCTION>...
C
   50 KTYPE=7
      DO 55 I=1,8
   55 TEXT(I)=CARD(I)
      GO TO 120
C
C.....<EVENT>...
C
   60 KTYPE=7
      NEVNT=1
      TEXT(1)=10H       SUBR
      TEXT(2)=10HOUTINE
      NCOL=17
      MCOL=ICOL
      ICOL=17
   65 MCOL=MCOL+1
      IF (MCOL.GT.72) GO TO 120
      CALL GCHARS (CARD,MCOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 65
      NCOL=NCOL+1
      CALL SCHARS (TEXT,NCOL,1,ICHR)
      GO TO 65


C
C.....RETRIEVE THE ROUTINE NAME, STORE NAME IN ROUTINE REFERENCE TABLE,
C.....AND GENERATE FTN EXTERNAL STATEMENT.
C
  120 NCOL=0
      NAME=10H
```

```
20174          125 ICOL=ICOL+1
20175              IF (ICOL.GT.72) GO TO 130
20176              CALL GCHARS (TEXT,ICOL,1,ICHR)
20177              IF (ICHR.EQ.1H ) GO TO 125
20178              IF (ICHR.EQ.1H() GO TO 130
20179              NCOL=NCOL+1
20180              IF (NCOL.GT.5) GO TO 150
20181              CALL SCHARS (NAME,NCOL,1,ICHR)
20182              GO TO 125
20183          130 IF (NCOL.LE.0) GO TO 150
20184              DO 135 I=1,NK1
20185                 IF (NAME.EQ.KEY1(I)) SUBFLG(I)=.FALSE.
20186          135 CONTINUE
20187              IF (NEVNT.EQ.0) GO TO 140
20188              NSUB=NSUB+1
20189              IF (NSUB.GT.100) GO TO 155
20190              NSBL(NSUB)=77777777770000000000B.AND.NAME
20191          140 IF (.N.PRINT) RETURN
20192              CALL FMTPG (2)
20193              WRITE (U2,180) CARD
20194              RETURN
```

Line Number — Explanation

20083-20089 — A SUBRØUTINE or FUNCTIØN was encountered. TEXT contains entire source card.

20090-20106 — An EVENT command is encountered. (NEVNT, not equal to zero, indicates the presence of an EVENT.) The EVENT command is transcribed into a SUBRØUTINE command. All nonblank characters following the word EVENT are filled into TEXT following the word SUBRØUTINE.

```
EXAMPLE. CARD=15H      EVENT BUG is filled into TEXT
          as TEXT=20H      SUBRØUTINE BUG
```

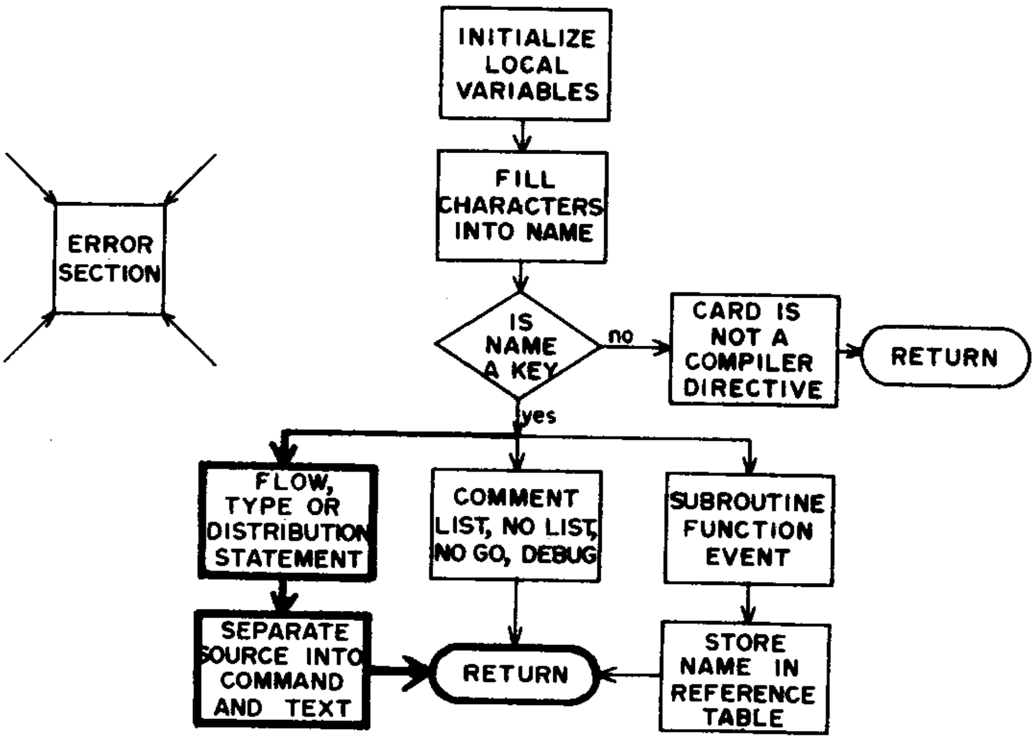
20168-20182 — The routine name is retrieved from TEXT. NAME will contain the 1-5 character code following the word FUNCTIØN or SUBRØUTINE. A left paren may delimit the routine name.

```
EXAMPLE. TEXT=20H      SUBRØUTINE BUG, then
          NAME=3HBUG.

EXAMPLE. TEXT=23H      FUNCTIØN ANT(A,B), then
          NAME=3HANT
```

20183-20186 — Once the routine name is in NAME, NAME is checked to see if it is one of the special system routine names: START, FINIS, CYCL1, or CYCL2. If it is, the SUBFLG pertaining to that routine is set to .FALSE., indicating the user has included the routine in his source deck.

Line Number	Explanation
20187-20190	If the routine is an EVENT, then the name of the event is placed in the event stack, NSBL(NSUB), where NSUB is the number of names in NSBL. NSUB=7 is the initial value. There are seven system events initially in NSBL (XCSIM,XPRNT,XPLØT,XFLØP,START, FINIS, and HALT).
20191-20194	If desired, write source card onto file output.

Process flow directive

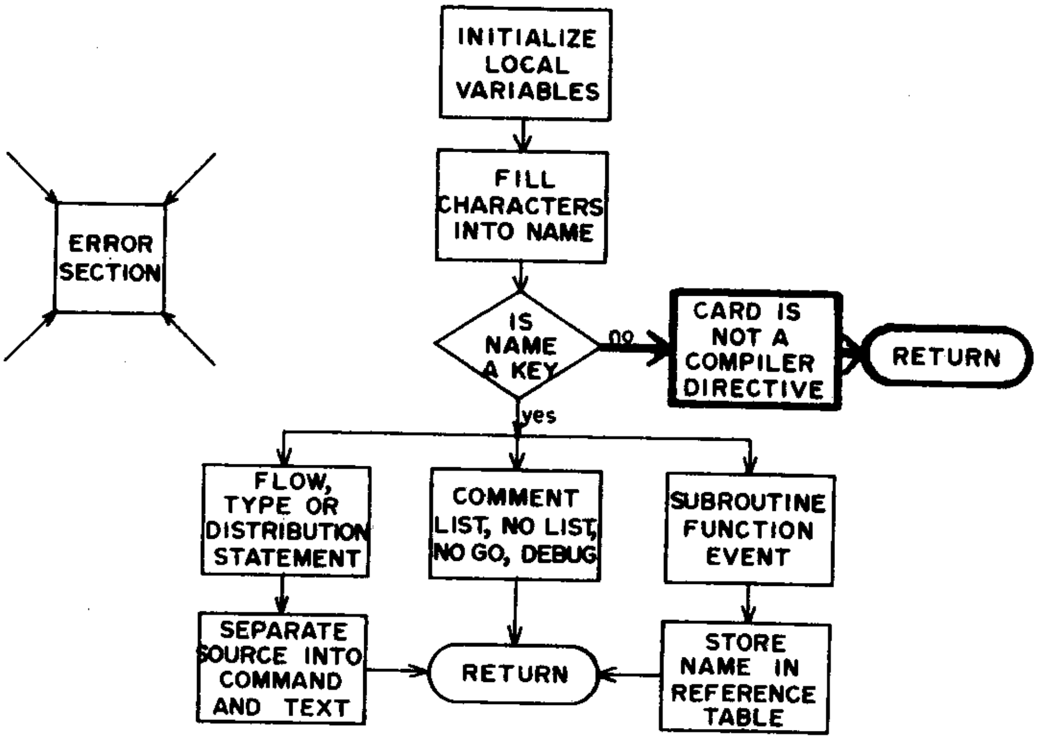

INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
CARD IS NOT A COMPILER DIRECTIVE
RETURN
yes
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE
ERROR SECTION

```
C
C.....<FLOW>...
C
   35 IF (ICOL.GT.5) GO TO 145
      KTYPE=5
      GO TO 95

C
C.....<STORAGE.>, <INTEGER.>, OR <REAL.>...
C
   45 KTYPE=I-2
      GO TO 95

C
C.....<UNIFORM.>, <NORMAL.>, <EXPONENT.>, OR <LOGNORMAL.>...
C
   70 KTYPE=4
      KDIST=I-8
      GO TO 95

C
C.....SEPARATE THE SOURCE CARD INTO COMMAND AND TEXT PORTIONS AND OUTPUT
C.....THE RESULTS.
C
   95 ICOL=0
      TRIP=.FALSE.
      MCOL=6
      NCOL=0
  100 ICOL=ICOL+1
```

```
20148             IF (ICOL.GT.72) GO TO 110
20149             CALL GCHARS (CARD,ICOL,1,ICHR)
20150             IF (TRIP) GO TO 105
20151             IF (ICHR.EQ.1H ) GO TO 100
20152             IF (ICHR.EQ.1H.) TRIP=.TRUE.
20153             NCOL=NCOL+1
20154             CALL SCHARS (COMAND,NCOL,1,ICHR)
20155             GO TO 100
20156         105 MCOL=MCOL+1
20157             CALL SCHARS (TEXT,MCOL,1,ICHR)
20158             GO TO 100
20159         110 IF (.N.PRINT) RETURN
20160             IF (KTYPE.EQ.5) GO TO 115
20161             CALL FMTPG (1)
20162             WRITE (U2,165) COMAND(1),TEXT
20163             RETURN
20164         115 CALL FMTPG (2)
20165             WRITE (U2,175) COMAND
20166             WRITE (U2,170) TEXT
20167             RETURN
```

Line Number	Explanation
20064-20069	A left paren was encountered before column 6. This indicates a flow command on source card. Proceed to separate source card into CØMAND and TEXT portions.
20078-20082	A STØRAGE., INTEGER., or REAL. command is encountered. Set KTYPE to appropriate value. (KTYPE=I-2=1 for a STØRAGE. command.)
20107-20112	A distribution command is encountered (UNIFØRM., NØRMAL., EXPØNENT., or LØGNØRMAL.). KDIST is a variable indicating which of the distribution commands was encountered. (KDIST=1 for UNIFØRM., 2 for NØRMAL., etc.)
20139-20167	The characters of the source card are packed into CØMAND until a period is encountered, indicating the end of the command portion. TRIP is set to .TRUE. and the remainder of the card (to column 73) is placed in TEXT. If a listing is desired, CØMAND and TEXT are copied to output, U2. FMTPG buffers the source listing into titled pages. (FMTPG(2) causes a blank line to be inserted between previous card and current one.)

No compiler directive encountered

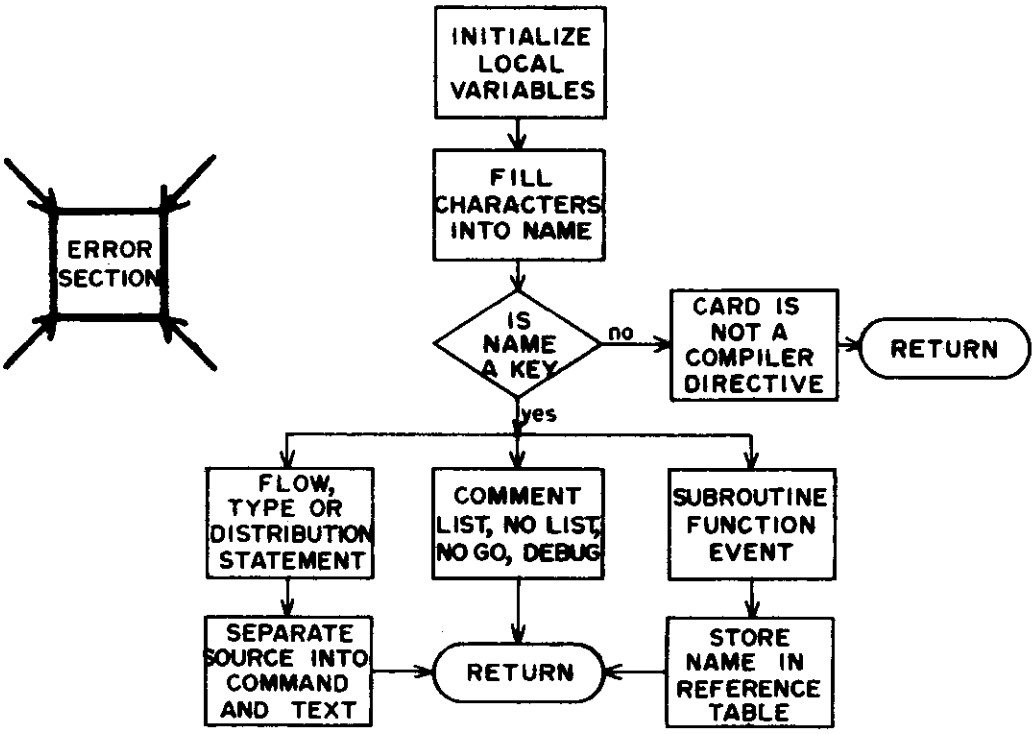

INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
yes
CARD IS NOT A COMPILER DIRECTIVE
RETURN
ERROR SECTION
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

```
20195          C
20196          C.....THE SOURCE CARD IS NOT A COMPILER DIRECTIVE.
20197          C
20198            145 IF (NCOL.EQ.0) RETURN M
20199                IF (JTYPE.LT.5) GO TO 160
20200                DECODE (5,185,CARD) LABEL
20201                DECODE (6,190,CARD) COL6
20202                CONT=.FALSE.
20203                IF (LABEL.EQ.1H .A.COL6.NE.1H ) CONT=.TRUE.
20204                IF (JTYPE.EQ.5.O.JTYPE.EQ.6) KTYPE=6
20205                IF (JTYPE.GE.7.A.JTYPE.LE.9) KTYPE=9
20206                IF (CONT.A.(JTYPE.EQ.7.O.JTYPE.EQ.8)) KTYPE=8
20207                IF (.N.PRINT) RETURN
20208                CALL FMTPG (1)
20209                WRITE (U2,170) CARD
20210                RETURN
```

Line Number	Explanation
20198-20203	If 10 characters are filled into NAME without a system command being discovered, then the source card can not be a SIMCØMP directive. If columns 1-5 on source card are blank and column 6 is not blank, then the card is a continuation (CØNT set to .TRUE.) of the previous card.

Line Number	Explanation
20204	If the previous card was a flow or flow text (KTYPE=5 or 6), then current card is flow text.
20205	If the previous card was a SUBRØUTINE, FUNCTIØN, or EVENT, then the present card is routine text (cards that follow a declared subroutine, function, or event).
20206	If the previous card was a SUBRØUTINE, FUNCTIØN, EVENT, or continuation of one of these, then current card is also a continuation.

Diagnostics

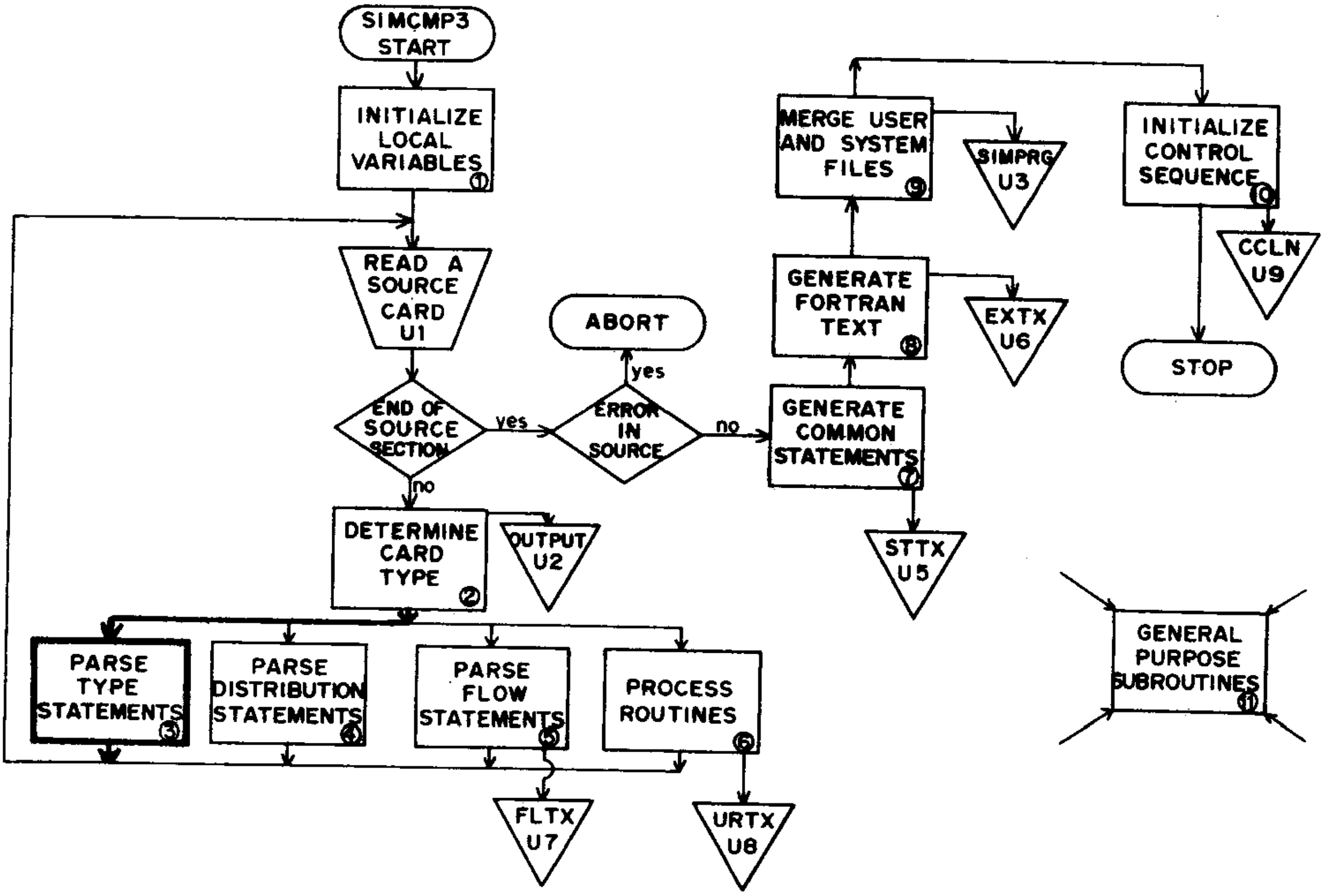

ERROR SECTION
INITIALIZE LOCAL VARIABLES
FILL CHARACTERS INTO NAME
IS NAME A KEY
no
CARD IS NOT A COMPILER DIRECTIVE
RETURN
yes
FLOW, TYPE OR DISTRIBUTION STATEMENT
COMMENT LIST, NO LIST, NO GO, DEBUG
SUBROUTINE FUNCTION EVENT
SEPARATE SOURCE INTO COMMAND AND TEXT
RETURN
STORE NAME IN REFERENCE TABLE

```
C
C.....IF AN ERROR OCCURED GENERATE A DIAGNOSTIC.
C
  150 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2,195)
      FATAL=.TRUE.
      KTYPE=7
      RETURN
  155 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2,200)
      FATAL=.TRUE.
      KTYPE=7
      RETURN
  160 CALL FMTPG (2)
      WRITE (U2,170) CARD
      WRITE (U2.205)
      FATAL=.TRUE.
      KTYPE=0
      RETURN M
C
  165 FORMAT (1H ,14X,A10,5X,8A10)
  170 FORMAT (1H ,29X,8A10)
  175 FORMAT (1H ,14X,8A10)
  180 FORMAT (1H0,29X,8A10)
  185 FORMAT (A5)
  190 FORMAT (5X,A1)
  195 FORMAT (11H *****FE    , 47HROUTINE NAME LONGER THAN 5 CHARS OR ZER
     10 LENGTH)
  200 FORMAT (11H *****FE    , 41HNUMBER OF USER-DEFINED EVENTS EXCEEDS 1
     100)
  205 FORMAT (11H *****FE    , 32HABOVE CARD ILLEGAL AT THIS POINT)
```

```
20244          210 FORMAT (1H ,14X, 62HNOGO.          AUTOMATIC CONTROL CARD GENERATI
20245             1ON IS SUPPRESSED)
20246          215 FORMAT (1H ,14X, 50HDEBUG.         DEBUG CONTROL SEQUENCE IS REQUE
20247             1STED)
20248        C
20249              END
```

Line Number	Explanation
20211-20231	A syntax error encountered while parsing a source card will cause control to arrive here. FATAL=.TRUE. and program will abort after source deck is completely read in.

1.3. *Parse Type Statements*



Overview

The three following statement forms are processed by this section:

(1) STØRAGE. <var1>,<var2>•••

(2) REAL. <var1>,<var2>•••

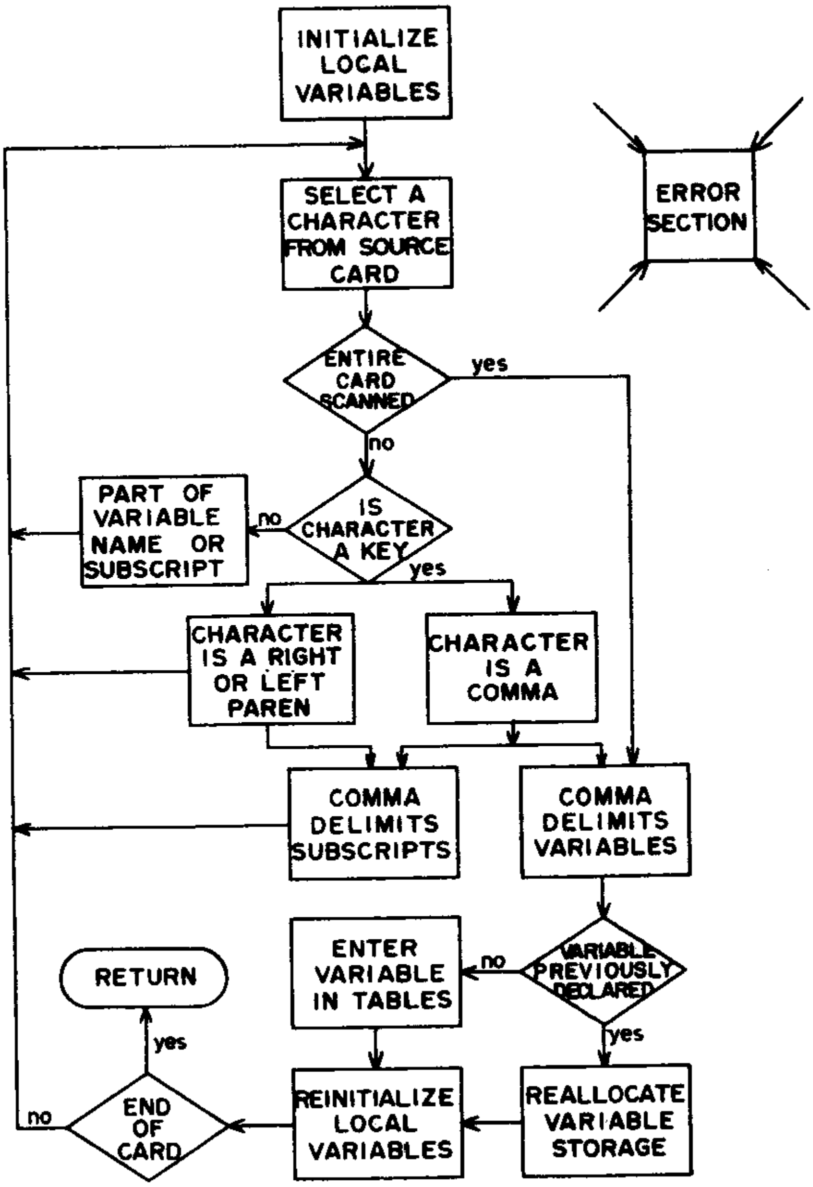
(3) INTEGER. <var1>,<var2>•••

<var1> is a variable name containing five or fewer characters and may be followed by three or fewer subscripts. Several examples of the statement forms follow:

```
STØRAGE. ANT,FØX(100,2),MØØSE(12,5,3)

REAL. MØØSE,GNAT(3)

INTEGER. FØX
```

The input to the section consists of the TEXT portion of a source card (<var1>,<var2>•••). Each variable encountered on a source card is placed in stacks LVR1 and LVR2. LVR1 contains the name of each variable, its starting location relative to the beginning of the stack, and its mode (whether the variable is real or integer). LVR2 contains the subscripts of the variable. These stacks are the output of the section and are used later to generate FØRTRAN REAL, INTEGER, and CØMMØN cards. The section is expanded for easier analysis.

Type statements flow chart

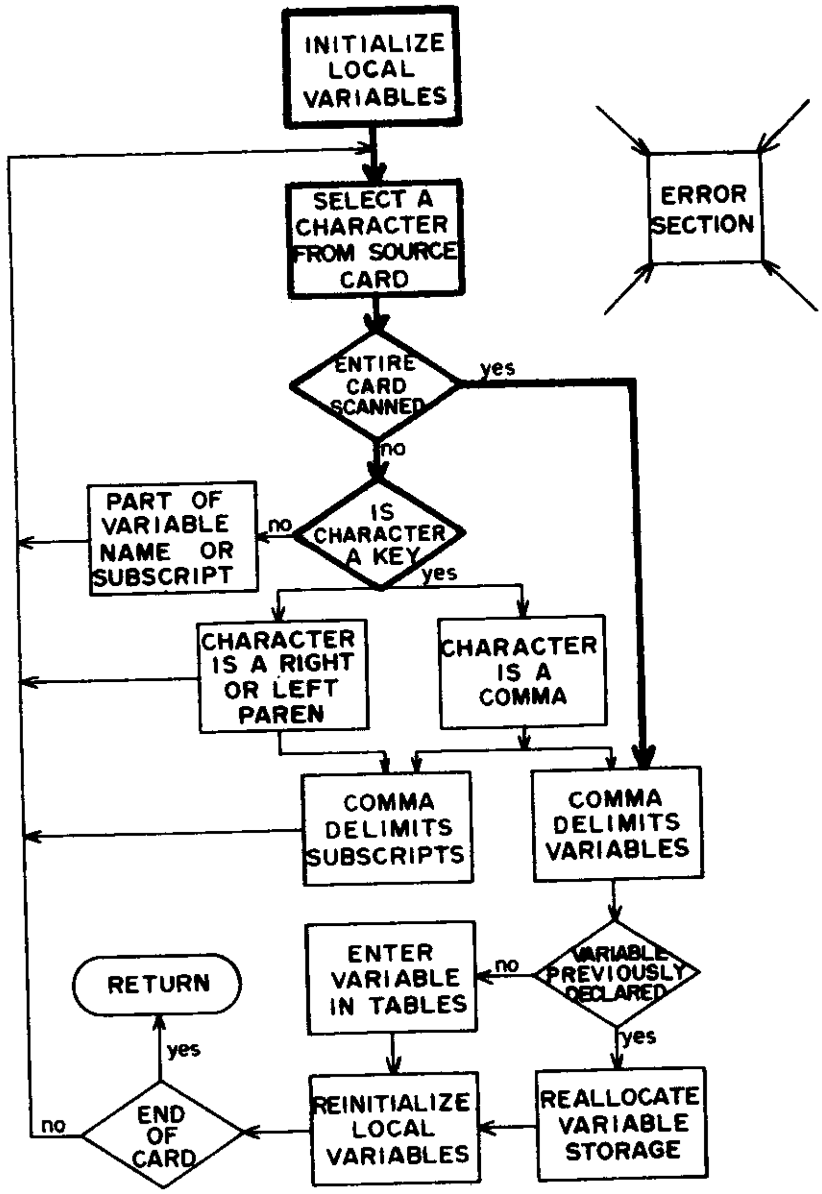

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
ERROR SECTION
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
PART OF VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
ENTER VARIABLE IN TABLES
yes
RETURN
yes
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE
no
END OF CARD

```
      SUBROUTINE STIDF (CARD,KTYPE,FATAL,IFTYPE)
      COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), KEY(3), MCOL(3), NSUB(3), NUM(3), KARD(8), MLAB
     1EL(2)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL KFLG,KSTOP,FATAL,IFTYPE
C
C.....THIS ROUTINE PARSES THE STORAGE DECLARATION DIRECTIVES GENERATING
C.....A VARIABLE REFERENCE TABLE AND FTN COMPATIBLE STORAGE DECLARATIONS
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H,,1H(,1H)/,NK/3/,NVMX/999/
      DATA MLABEL/7HINTEGER,4HREAL/
      KODE=1
      NAME=10H
      NCOL=0
      NSUB(3)=10H
      NSUB(2)=NSUB(3)
      NSUB(1)=NSUB(2)
      NUM(3)=0
      NUM(2)=NUM(3)
      NUM(1)=NUM(2)
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      KFLG=.FALSE.
      KSTOP=.FALSE.
      IDMPF=0
      ICOL=6
C
C.....RETRIEVE EACH CHARACTER COLUMN BY COLUMN SEARCHING FOR KEY CHARS.
C
   15 ICOL=ICOL+1
      IF (ICOL.GT.72) GO TO 85
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 15
      DO 20 I=1,NK
         IF (ICHR.EQ.KEY(I)) GO TO 35
   20 CONTINUE
C
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,30,30,30,95), KODE
C
C....."ICHR" IS ASSUMED PART OF A VARIABLE NAME.
C
   25 IF (NCOL.NE.0.O.ICHR.NE.1H*) GO TO 26
      IDMPF=1
      GO TO 15
   26 IF (ICHR.LT.1HA.A.ICHR.GT.1H9) GO TO 95
      NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 100
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
C
C....."ICHR" IS ASSUMED PART OF A SUBSCRIPT.
C
   30 IF (ICHR.LT.1H0.A.ICHR.GT.1H9) GO TO 95
      J=KODE-1
      MCOL(J)=MCOL(J)+1
      IF (MCOL(J).GT.4) GO TO 105
      CALL SCHARS (NSUB(J),MCOL(J),1,ICHR)
      GO TO 15
C
C....."ICHR" IS A KEY CHARACTER.
C
   35 GO TO (40,75,80), I
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 50
C
C.....THE COMMA DELIMITS SUBSCRIPTS.
C
   45 J=KODE-1
      IF (MCOL(J).LE.0) GO TO 110
      CALL GNUM (NSUB(J),1,MCOL(J),NUM(J),IERR)
      IF (IERR.NE.0) GO TO 115
      IF (NUM(J).GT.1023) GO TO 120
      IF (KFLG) KODE=4
      KODE=KODE+1
      GO TO 15
C
C.....THE COMMA DELIMITS VARIABLES.
C
   50 IF (NCOL.LE.0) GO TO 125
C
```

```
C.....CHECK TO SEE IF VARIABLE STARTS WITH "X" OR NUMERIC.
C
      CALL GCHARS (NAME,1,1,ICHR)
      IF (ICHR.EQ.1HX) GO TO 130
      IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 135
C
C.....DETERMINE VARIABLE TYPE.
C
   55 MODE=1
      IF (ICHR.GE.1HI.A.ICHR.LE.1HN) MODE=0
      IF (IFTYPE) MODE=KTYPE-2
C
C.....CHECK TO SEE IF VARIABLE HAS BEEN PREVIOUSLY DECLARED.
C
      DO 60 I=1,NVAR
         CALL GCHARS (LVR1(I),1,5,LNM)
         IF (NAME.EQ.LNM) GO TO 140
   60 CONTINUE
C
C.....THE VARIABLE IS ACCEPTABLE, STORE THE INFORMATION IN THE TABLES.
C
      NVAR=NVAR+1
      IF (NVAR.GT.NVMX) GO TO 160
      LVR1(NVAR)=0
      CALL SCHARS (LVR1(NVAR),1,5,NAME)
      CALL SBYTE (LVR1(NVAR),NSTOR,30,18)
      CALL SBYTE (LVR1(NVAR),MODE,48,2)
      CALL SBYTE (LVR1(NVAR),IDMPF,50,10)
      LVR2(NVAR)=0
      NDIM=1
      DO 65 I=1,3
         IF (NUM(I).GT.0) NDIM=NDIM*NUM(I)
   65 CALL SBYTE (LVR2(NVAR),NUM(I),10*I-10,10)
      NSTOR=NSTOR+NDIM
C
C.....REINITIALIZE THE LOCAL STORAGE VARIABLES.
C
   70 IF (KSTOP) GO TO 90
      KODE=1
      NCOL=0
      NAME=10H
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      NSUB(3)=10H
      NSUB(2)=NSUB(3)
      NSUB(1)=NSUB(2)
      NUM(3)=0
      NUM(2)=NUM(3)
      NUM(1)=NUM(2)
      KFLG=.FALSE.
      IDMPF=0
      GO TO 15
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   75 IF (KODE.NE.1) GO TO 95
      KODE=2
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   80 IF (KODE.LT.2.O.KODE.GT.4) GO TO 95
      KFLG=.TRUE.
      GO TO 45
   85 IF (KODE.GE.2.A.KODE.LE.4) GO TO 165
      KSTOP=.TRUE.
      GO TO 50
   90 RETURN
C
C.....GENERATE ERROR MESSAGES IF ERRORS ENCOUNTERED.
C
```

```
   95 WRITE (U2,170) ICHR
      FATAL=.TRUE.
      RETURN
  100 WRITE (U2,175) NAME
      FATAL=.TRUE.
      RETURN
  105 WRITE (U2,180) NSUB(J)
      FATAL=.TRUE.
      RETURN
  110 WRITE (U2,185)
      FATAL=.TRUE.
      RETURN
  115 WRITE (U2,190) NSUB(J)
      FATAL=.TRUE.
      RETURN
  120 WRITE (U2,195) NSUB(J)
      FATAL=.TRUE.
      RETURN
  125 WRITE (U2,200)
      FATAL=.TRUE.
      RETURN
  130 WRITE (U2,205) NAME
      GO TO 55
  135 WRITE (U2,210) NAME
      FATAL=.TRUE.
      RETURN
C
C.....ENTER HERE IF VARIABLE HAS OCCURRED BEFORE IN STORAGE, REAL OR
C         INTEGER STATEMENTS.
C
  140 IF (I.GT.9) GO TO 150
      IF (I.EQ.9) GO TO 145
      WRITE (U2,215) NAME
      FATAL=.TRUE.
      RETURN
  145 WRITE (U2,235)
      GO TO 70
  150 IF (NUM(1).LE.0) GO TO 155
      WRITE (U2,230) NAME
      CALL SHUFFLE (I,NUM)
  155 IF(IFTYPE) CALL SBYTE(LVR1(I),MODE,48,2)
      GO TO 70
  160 WRITE (U2,220) NVMX
      FATAL=.TRUE.
      RETURN
  165 WRITE (U2,225)
      FATAL=.TRUE.
      RETURN
C
  170 FORMAT (11H *****FE    , 11HCHARACTER ",A1, 12H" IS ILLEGAL)
  175 FORMAT (11H *****FE    , 10HVARIABLE ",A5, 27H..." IS LONGER THAN 5
     1 CHARS)
  180 FORMAT (11H *****FE    , 11HSUBSCRIPT ",A4, 27H..." IS LONGER THAN
     14 CHARS)
  185 FORMAT (11H *****FE    ,26HEXPECTED SUBSCRIPT MISSING)
  190 FORMAT (11H *****FE    , 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
  195 FORMAT (11H *****FE    , 11HSUBSCRIPT ",A4, 19H" GREATER THAN 1023)
  200 FORMAT (11H *****FE    ,30HEXPECTED VARIABLE NAME MISSING)
  205 FORMAT (11H *****NF    , 10HVARIABLE ",A5, 22H" BEGINS WITH CHAR "X
     1|)
  210 FORMAT (11H *****FE    , 10HVARIABLE ",A5, 37H" BEGINS WITH A NON-A
     1LPHABETICAL CHAR)
  215 FORMAT (11H *****FE    , 10HVARIABLE ",A5, 31H" IS A RESERVED SYSTE
     1M VARIABLE)
  220 FORMAT (11H *****FE    , 42HNUMBER OF DECLARED VARIABLES HAS EXCEED
     1ED ,I6)
  225 FORMAT (11H *****FE    , 48HA VARIABLE DECLARATION IS INCOMPLETE AT
     1 CARD END)
  230 FORMAT (11H *****NF    ,10HVARIABLE (,A5,67H) HAS BEEN PREVIOUSLY D
     1ECLARED, LAST DECLARATION IS ASSUMED CORRECT)
  235 FORMAT (1H ,T12, 60HSTATE VARIABLES SHOULD NOT BE DECLARED IN STOR
     1AGE STATEMENTS)
C
      END
```

Initialize and retrieve characters


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
ERROR SECTION
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
PART OF VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
ENTER VARIABLE IN TABLES
yes
RETURN
yes
REALLOCATE VARIABLE STORAGE
REINITIALIZE LOCAL VARIABLES
END OF CARD
no

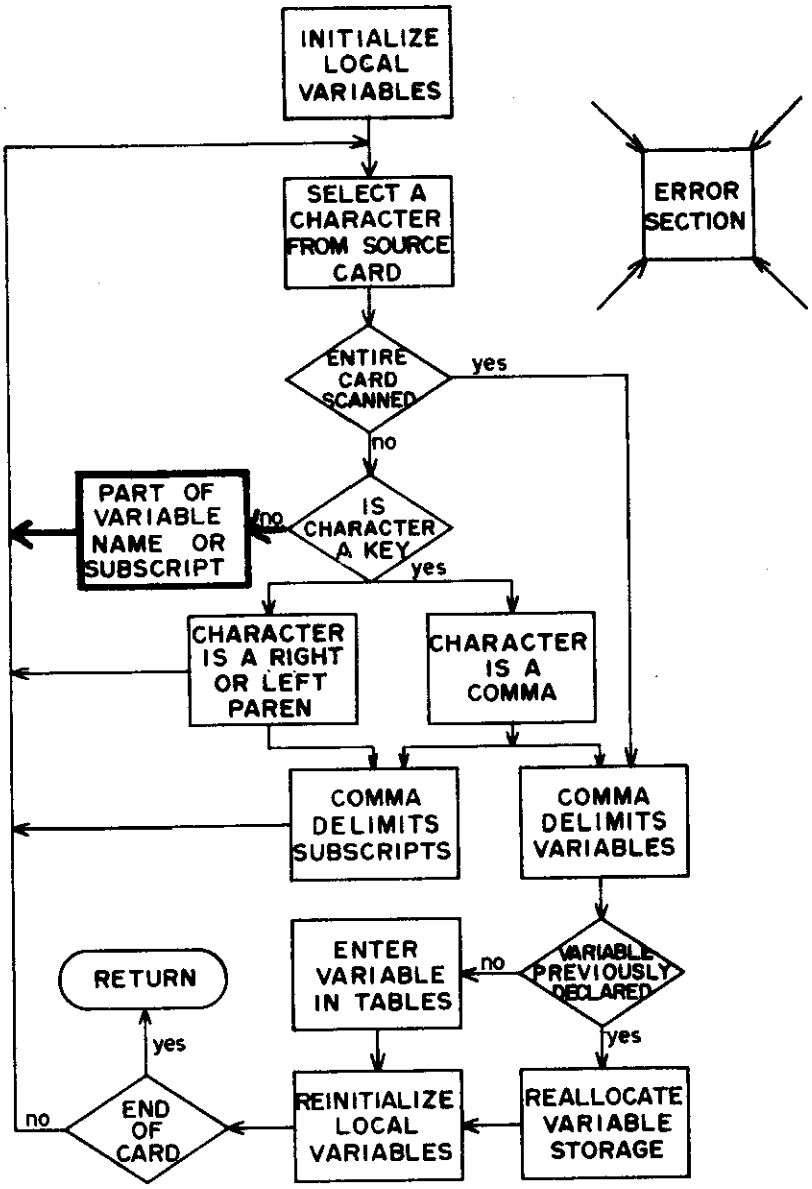
```
      SUBROUTINE STIDF (CARD,KTYPE,FATAL,IFTYPE)
      COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), KEY(3), MCOL(3), NSUB(3), NUM(3), KARD(8), MLAB
     1EL(2)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL KFLG,KSTOP,FATAL,IFTYPE
C
C.....THIS ROUTINE PARSES THE STORAGE DECLARATION DIRECTIVES GENERATING
C.....A VARIABLE REFERENCE TABLE AND FTN COMPATIBLE STORAGE DECLARATIONS
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H,,1H(,1H)/,NK/3/,NVMX/999/
      DATA MLABEL/7HINTEGER,4HREAL/
      KODE=1
```

```
21016                NAME=10H
21017                NCOL=0
21018                NSUB(3)=10H
21019                NSUB(2)=NSUB(3)
21020                NSUB(1)=NSUB(2)
21021                NUM(3)=0
21022                NUM(2)=NUM(3)
21023                NUM(1)=NUM(2)
21024                MCOL(3)=0
21025                MCOL(2)=MCOL(3)
21026                MCOL(1)=MCOL(2)
21027                KFLG=.FALSE.
21028                KSTOP=.FALSE.
21029                IDMPF=0
21030                ICOL=6
21031          C
21032          C.....RETRIEVE EACH CHARACTER COLUMN BY COLUMN SEARCHING FOR KEY CHARS.
21033          C
21034             15 ICOL=ICOL+1
21035                IF (ICOL.GT.72) GO TO 85
21036                CALL GCHARS (CARD,ICOL,1,ICHR)
21037                IF (ICHR.EQ.1H ) GO TO 15
21038                DO 20 I=1,NK
21039                   IF (ICHR.EQ.KEY(I)) GO TO 35
21040             20 CONTINUE
21041          C

21154             85 IF (KODE.GE.2.A.KODE.LE.4) GO TO 165
21155                KSTOP=.TRUE.
21156                GO TO 50
```

Line Number	Explanation
21000	IFTYPE is a logical flag that indicates the type of variable is to be set to either integer or real. CARD contains only the TEXT portion of the source card (all of card except the command -- STØRAGE., REAL., or INTEGER.).
21013-21030	Local variables are initialized. KEY contains the only possible special characters for this type of source card. NK is the length of KEY. NVMX is the maximum number of variables allowed in declaration statements. KØDE is a variable which indicates the number of special characters found within a variable name. EXAMPLE. FØX(15,4,3),BUG KØDE=1 Until left paren is encountered, then KØDE=2. KØDE=3 After first comma encountered. KØDE=4 After second comma encountered. KØDE=5 After right paren. KØDE is reset to 1 after the rightmost comma and parsing begins for BUG.

Line Number	Explanation
	NAME will contain a variable name from the source card. NCØL is the number of nonblank characters in NAME. NSUB are the characters of each subscript (BCD). NUM is the integer value of each subscript. MCØL is the number of characters in each subscript. KFLG is a flag indicating a right paren has been found. KSTØP is the end of card flag. ICØL is the source card column currently being scanned. Referring to above example, after FØX(15,4,3), has been parsed, local variables will contain: NAME=3HFØX NCØL=3 NSUB(1)=2H15 MCØL(1)=2 NUM(1)=15 NSUB(2)=1H4 MCØL(2)=1 NUM(2)=14 NSUB(3)=1H3 MCØL(3)=1 NUM(3)=3
21031-21041	A character is retrieved and checked against KEY to determine if it is a key character (a comma, right paren, or left paren). ICØL greater than 72 indicates end of source card.
21154-21156	KSTØP=.TRUE. when column 73 is reached. The last variable in a FTN. string is not followed by a comma. Process tables for last variable.

Character is part of variable name or subscripts


INITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
PART OF VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
RETURN
ENTER VARIABLE IN TABLES
no
VARIABLE PREVIOUSLY DECLARED
yes
yes
no
END OF CARD
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE

```
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,30,30,30,95), KODE
C
C....."ICHR" IS ASSUMED PART OF A VARIABLE NAME.
C
   25 IF (NCOL.NE.0.O.ICHR.NE.1H*) GO TO 26
      IDMPF=1
      GO TO 15
   26 IF (ICHR.LT.1HA.A.ICHR.GT.1H9) GO TO 95
      NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 100
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
```

```
21056          C
21057          C....."ICHR" IS ASSUMED PART OF A SUBSCRIPT.
21058          C
21059             30 IF (ICHR.LT.1H0.A.ICHR.GT.1H9) GO TO 95
21060                J=KODE-1
21061                MCOL(J)=MCOL(J)+1
21062                IF (MCOL(J).GT.4) GO TO 105
21063                CALL SCHARS (NSUB(J),MCOL(J),1,ICHR)
21064                GO TO 15
```

Line Number — Explanation

21042-21055

The character is not a comma, right paren, or left paren. Since KØDE is incremented for each key character found within a variable, KØDE=1 indicates that the selected must be part of a variable name. Therefore, this character is packed into NAME. (When the next special character is encountered, NAME will contain a complete variable name.) If the character is an asterisk, set IDMPF flag (variable name following is second class).

21056-21064

The character is assumed part of the subscript if KØDE=2, 3, or 4. Variables MCØL and NSUB are updated.

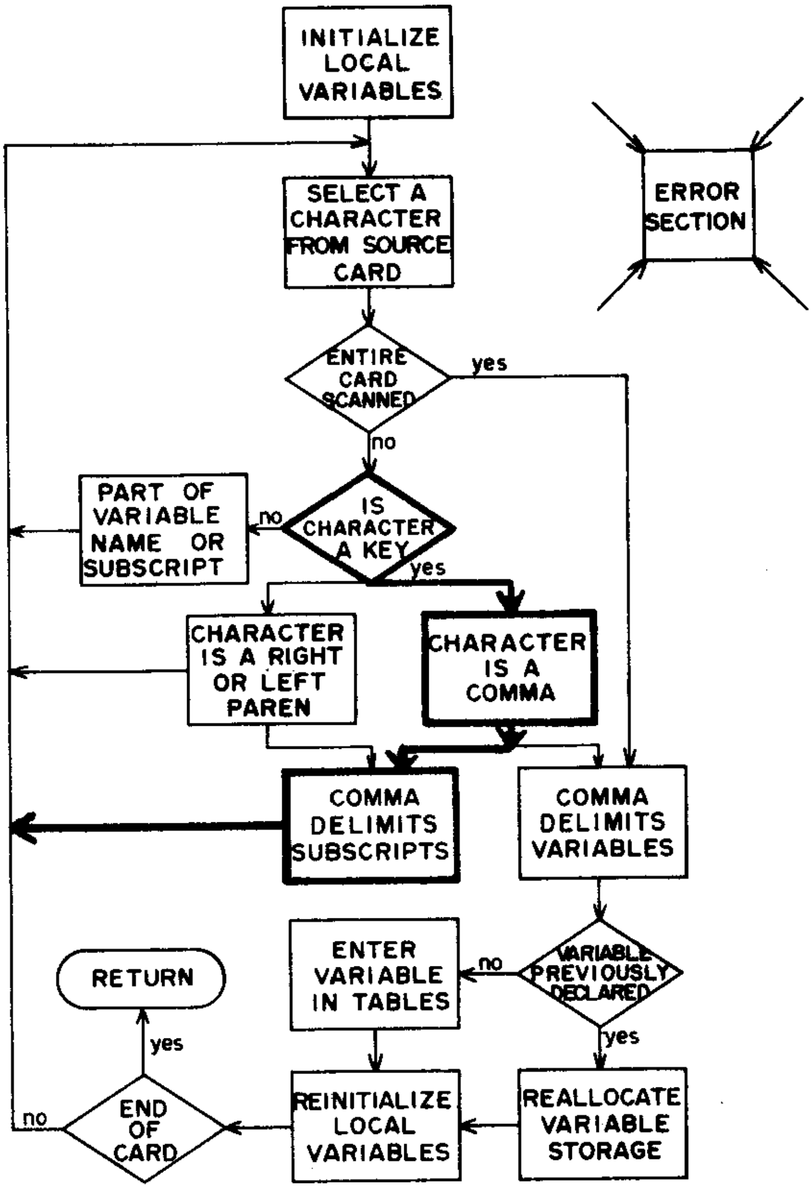
EXAMPLE: Examine BT(15,1,200) as it is parsed character by character.

B	KØDE=1	NAME=10HB	NCØL=1	NSUB=0	MCØL=0
T	KØDE=1	NAME=10HBT	NCØL=2	NSUB=0	MCØL=0
(	KØDE=2	NAME=10HBT	NCØL=2	NSUB=0	MCØL=0
1	KØDE=2	NAME=10HBT	NCØL=2	NSUB(1)=10H1	MCØL(1)=1
5	KØDE=2	NAME=10HBT	NCØL=2	NSUB(1)=10H15	MCØL(1)=2
,	KØDE=3	NAME=10HBT	NCØL=2		
1	KØDE=3	NAME=10HBT	NCØL=2	NSUB(2)=10H1	MCØL(2)=1
,	KØDE=4	NAME=10HBT	NCØL=2		
2	KØDE=4	NAME=10HBT	NCØL=2	NSUB(3)=10H2	MCØL(3)=1
0	KØDE=4	NAME=10HBT	NCØL=2	NSUB(3)=10H20	MCØL(3)=2
0	KØDE=4	NAME=10HBT	NCØL=2	10H200	MCØL(3)=3
)		KFLG=.TRUE.			

This information is used to create tables (LVR1 and LVR2) which are later used to develop FTN. CØMMØN, REAL, and INTEGER statements.

NOTE. If KØDE=5 and the retrieved character is not a key character, an error is encountered and control passes to error section.

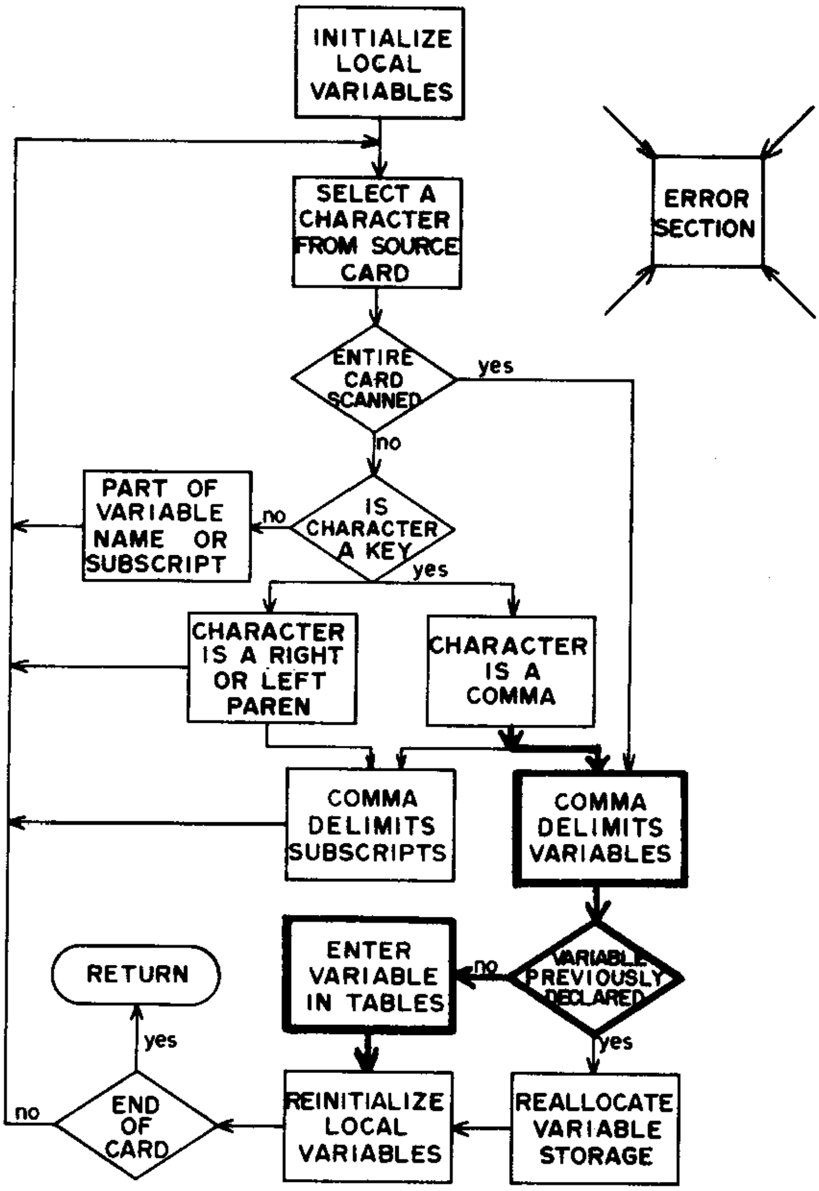
Subscript delimiter encountered


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
ERROR SECTION
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
yes
PART OF VARIABLE NAME OR SUBSCRIPT
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
yes
ENTER VARIABLE IN TABLES
RETURN
yes
END OF CARD
no
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE

```
C
C....."ICHR" IS A KEY CHARACTER.
C
   35 GO TO (40,75,80), I
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 50
C
C.....THE COMMA DELIMITS SUBSCRIPTS.
C
   45 J=KODE-1
      IF (MCOL(J).LE.0) GO TO 110
      CALL GNUM (NSUB(J),1,MCOL(J),NUM(J),IERR)
      IF (IERR.NE.0) GO TO 115
      IF (NUM(J).GT.1023) GO TO 120
      IF (KFLG) KODE=4
      KODE=KODE+1
      GO TO 15
```

Line Number	Explanation
21068	A KEY character that is a comma assigns I=1 and branches to 40.
21072	If a comma is encountered when KØDE=2,3,4, then a left paren has been encountered and the comma delimits a subscript.
21076-21083	NUM(J) contains the integer value of the Jth subscript. The BCD subscript values in NSUB are converted to integer values in NUM. KØDE is incremented and parsing continues. KFLG=.TRUE. indicates a right paren has been encountered.

Store variable information in the tables


INITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
ENTIRE CARD SCANNED
yes
no
PART OF VARIABLE NAME OR SUBSCRIPT
no
IS CHARACTER A KEY
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
RETURN
ENTER VARIABLE IN TABLES
no
VARIABLE PREVIOUSLY DECLARED
yes
yes
no
END OF CARD
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE

```
C
C.....THE COMMA DELIMITS VARIABLES.
C
   50 IF (NCOL.LE.0) GO TO 125
C
C.....CHECK TO SEE IF VARIABLE STARTS WITH "X" OR NUMERIC.
C
      CALL GCHARS (NAME,1,1,ICHR)
      IF (ICHR.EQ.1HX) GO TO 130
      IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 135
C
C.....DETERMINE VARIABLE TYPE.
C
   55 MODE=1
      IF (ICHR.GE.1HI.A.ICHR.LE.1HN) MODE=0
      IF (IFTYPE) MODE=KTYPE-2
```

```
21100          C
21101          C.....CHECK TO SEE IF VARIABLE HAS BEEN PREVIOUSLY DECLARED.
21102          C
21103                DO 60 I=1,NVAR
21104                   CALL GCHARS (LVR1(I),1,5,LNM)
21105                   IF (NAME.EQ.LNM) GO TO 140
21106             60 CONTINUE
21107          C
21108          C.....THE VARIABLE IS ACCEPTABLE, STORE THE INFORMATION IN THE TABLES.
21109          C
21110                NVAR=NVAR+1
21111                IF (NVAR.GT.NVMX) GO TO 160
21112                LVR1(NVAR)=0
21113                CALL SCHARS (LVR1(NVAR),1,5,NAME)
21114                CALL SBYTE (LVR1(NVAR),NSTOR,30,18)
21115                CALL SBYTE (LVR1(NVAR),MODE,48,2)
21116                CALL SBYTE (LVR1(NVAR),IDMPF,50,10)
21117                LVR2(NVAR)=0
21118                NDIM=1
21119                DO 65 I=1,3
21120                   IF (NUM(I).GT.0) NDIM=NDIM*NUM(I)
21121             65 CALL SBYTE (LVR2(NVAR),NUM(I),10*I-10,10)
21122                NSTOR=NSTOR+NDIM
```

Line Number	Explanation
	The comma delimits variables (enter from line 21072). KØDE=1 indicates the variable parsed contained no subscripts. KØDE=5 indicates a right paren was the previous character.
21087	The variable name is of 0 length, branch to error section.
21088-21093	If the variable starts with an "X," a nonfatal message is issued. A fatal error flag is set if the variable does not start with a letter of the alphabet.
21094-21099	IFTYPE is a flag which indicates that the statement being parsed is a REAL. or INTEGER. statement. This being the case, all variables appearing after an INTEGER. statement are assigned MØDE=0. Variables encountered after a REAL. statement are assigned MØDE=1. Variable appearing only in STORAGE. Statements are assigned MØDE values according to standard FTN conventions.
21100-21106	The variable stack is searched to determine if the variable has been previously declared.
21108-21122	Since the variable has *not* appeared before in a type statement, the variable name and characteristics are added to the variable stacks--LVR1 and LVR2. NVAR represents the number of declared variables. NVMX=999 is the maximum number of declared variables that the system allows.

Line Number	Explanation

EXAMPLE. Assume STORAGE. FØX(15,2),MØØSE appears as the first type card.

LVR1(10)=06173055550017602000B where:

NVAR=10 (NVAR=9 initially since there are nine system variables initially in LVR1).

0617305555_8=5HFØX.

$001760_8=1008_{10}$=NSTØR (NSTØR is the starting location of FØX relative to the beginning of reserved variable space.

NOTE. System variables take up the first 1007_{10} words of reserved variable space.)

$2=10_2$=MØDE indicates FØX is of type real. (Leftmost bit of the two is used as the MØDE.)

000B=class designation: a 1 in the rightmost bit would indicate that the variable is second class storage and a 0 indicates normal storage.

LVR2 contains the subscripts of the variable name in LVR1.

NDIM totals array NUM (which contains subscripts of the variable) and represents the total number of words (locations) to set aside for that variable.

Continuing with the example above.

NDIM=15*2=30 locations to allocate for FØX.

LVR2(10)= $\underline{0000001111}$ $\underline{0000000010}$ $\underline{0000}$... → zero fill to 60 total bits.

The leftmost 10 bits will contain the first subscript (0000001111=15, the first subscript of FØX).

The second group of 10 bits contains the second subscript (0000000010=2, second subscript of FØX).

The third group of bits contains the third subscript (0000000000=0, FØX has no third subscript).

21122 NSTØR=1008+30=1038 represents the starting location for the next encountered variable, in this case MØØSE. Completing the example, after MØØSE has been parsed and control again arrives here:

NVAR=11

LVR1(11)=15171723050020160000B

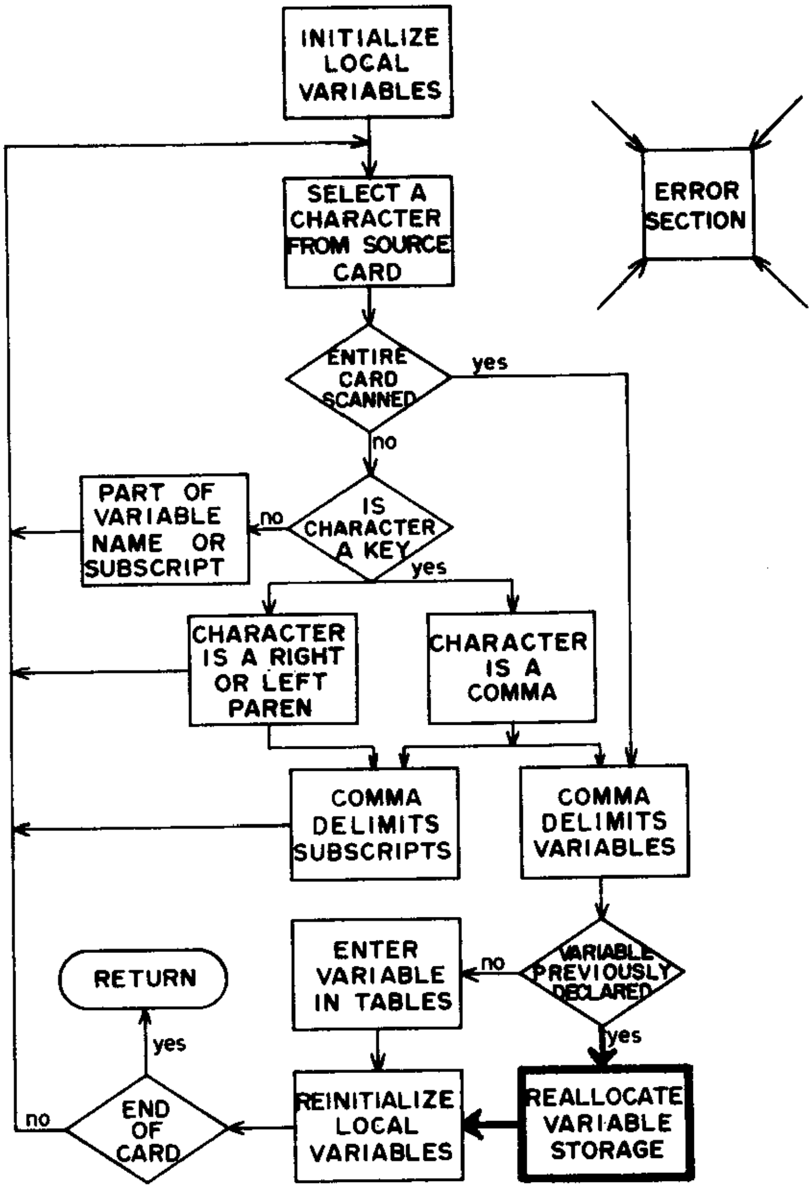
1517172305_8=5HMØØSE

002016_8 $=1038_{10}$ starting location of MØØSE

0000B =MØØSE is typed as integer

Line Number	Explanation
	LVR2(11)=000...to 60 bits. This indicates MØØSE has no subscripts, therefore NDIM=1. NSTØR=1039

Reallocate variable storage


INITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
PART OF VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
ENTER VARIABLE IN TABLES
yes
RETURN
yes
REALLOCATE VARIABLE STORAGE
REINITIALIZE LOCAL VARIABLES
END OF CARD
no

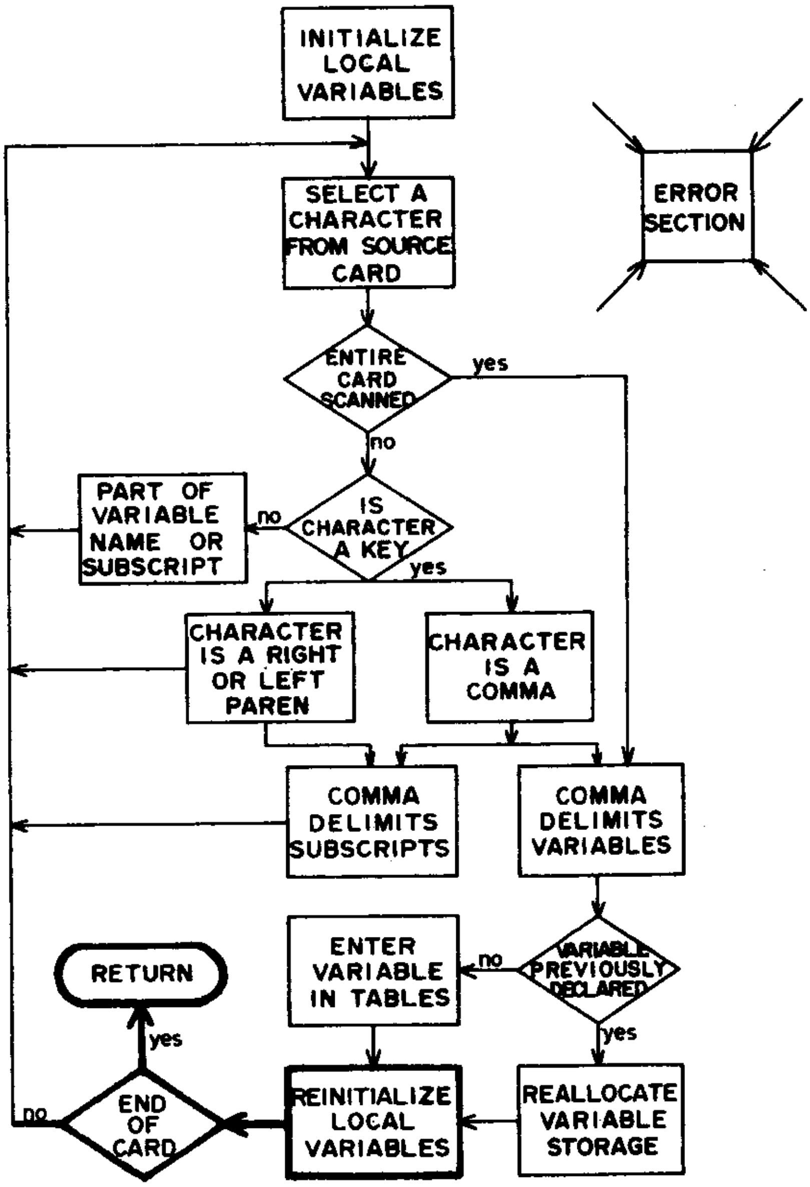
```
C
C.....ENTER HERE IF VARIABLE HAS OCCURRED BEFORE IN STORAGE, REAL OR
C        INTEGER STATEMENTS.
C
  140 IF (I.GT.9) GO TO 150
      IF (I.EQ.9) GO TO 145
      WRITE (U2,215) NAME
      FATAL=.TRUE.
      RETURN
  145 WRITE (U2,235)
      GO TO 70
  150 IF (NUM(1).LE.0) GO TO 155
      WRITE (U2,230) NAME
      CALL SHUFFLE (I,NUM)
  155 IF(IFTYPE) CALL SBYTE(LVR1(I),MODE,48,2)
      GO TO 70
```

```
22000         SUBROUTINE SHUFFLE (IBEG,NUM)
22001         COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
22002         DIMENSION NUM(3)
22003   C
22004   C.....SHUFFLE IS CALLED WHEN THE STORAGE ALLOCATIONS ARE CHANGED FOR
22005   C         A PREVIOUSLY DECLARED VARIABLE
22006   C.....STORE NEW DIMENSIONS OF PREVIOUSLY DECLARED VARIABLE
22007   C
22008         CALL GBYTE (LVR1(IBEG),NSTOR,30,18)
22009         NDIM=1
22010         LVR2(IBEG)=0
22011         DO 15 I=1,3
22012            IF (NUM(I).GT.0) NDIM=NDIM*NUM(I)
22013      15 CALL SBYTE (LVR2(IBEG),NUM(I),10*I-10,10)
22014         NSTOR=NSTOR+NDIM
22015   C
22016   C.....RECALCULATE STORAGE ADDRESSES OF ALL SUCCEEDING VARIABLES.
22017   C
22018         INC=IBEG+1
22019         IF (INC.GT.NVAR) GO TO 30
22020         DO 25 K=INC,NVAR
22021            NDIM=1
22022            NUM(3)=0
22023            NUM(2)=NUM(3)
22024            NUM(1)=NUM(2)
22025            DO 20 I=1,3
22026               CALL GBYTE (LVR2(K),NUM(I),10*I-10,10)
22027               IF (NUM(I).GT.0) NDIM=NDIM*NUM(I)
22028      20    CONTINUE
22029            CALL SBYTE (LVR1(K),NSTOR,30,18)
22030            NSTOR=NSTOR+NDIM
22031      25 CONTINUE
22032      30 RETURN
22033   C
22034         END
```

Line Number	Explanation
	The variable is compared with variable names already existing in the variable stack. If the variable has occurred previously, either the storage allocation or mode of the variable is to be changed from that in the stack (enter from line 21105).
21187-21197	The first nine variables in the stack are system variables; an attempt to change their storage allocations or mode results in a fatal error.
21198-21202	NUM contains integer values of the subscripts. If NUM=0, the variable has no subscripts and only the mode of the variable is changed. If NUM≠0, then new storage allocation must be made for this variable. Also the beginning relative location of all variables in the stack following the changed variable must be altered. This is accomplished by subroutine SHUFFLE.
21201-21202	If the statement being parsed is a REAL. or INTEGER. statement, the mode of the variable is changed to correspond with the type of statement. Control proceeds to the reinitialization section.

Line Number	Explanation
22000	IBEG is the location in LVR1 and LVR2 where the variable previously occurred.
22008-22013	The previous subscripts of the variable are deleted, and the new dimensions are inserted into LVR2.
22014-22034	The beginning relative locations for all variables following the altered variable are calculated: (1) The beginning relative location is placed in LVR1(IBEG+1). (2) The dimensions for that variable are calculated (NDIM). (3) NSTØR and NDIM are the next beginning relative location and are placed in LVR1(IBEG+2). This continues to the end of stack (NVAR).

Reinitialize

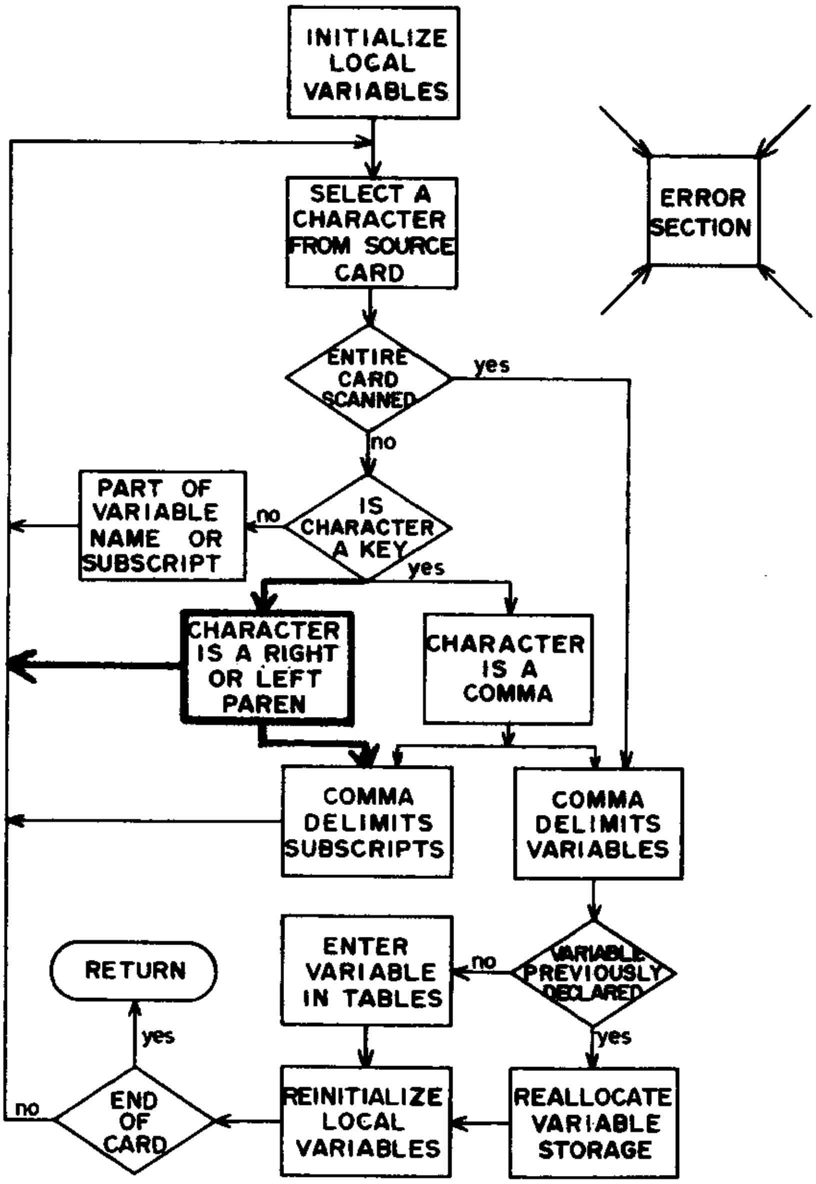

INITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
yes
PART OF VARIABLE NAME OR SUBSCRIPT
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
yes
ENTER VARIABLE IN TABLES
REALLOCATE VARIABLE STORAGE
REINITIALIZE LOCAL VARIABLES
END OF CARD
no
yes
RETURN

```
C
C.....REINITIALIZE THE LOCAL STORAGE VARIABLES.
C
   70 IF (KSTOP) GO TO 90
      KODE=1
      NCOL=0
      NAME=10H
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      NSUB(3)=10H
      NSUB(2)=NSUB(3)
      NSUB(1)=NSUB(2)
      NUM(3)=0
```

```
21137                NUM(2)=NUM(3)
21138                NUM(1)=NUM(2)
21139                KFLG=.FALSE.
21140                IDMPF=0
21141                GO TO 15

21157             90 RETURN
```

Line Number	Explanation
21126	KSTØP=.TRUE. indicates column 73 has been reached and the source card has been completely parsed.
21127-21141	All local variables are reinitialized; then parsing will continue for the next variable on the source card.

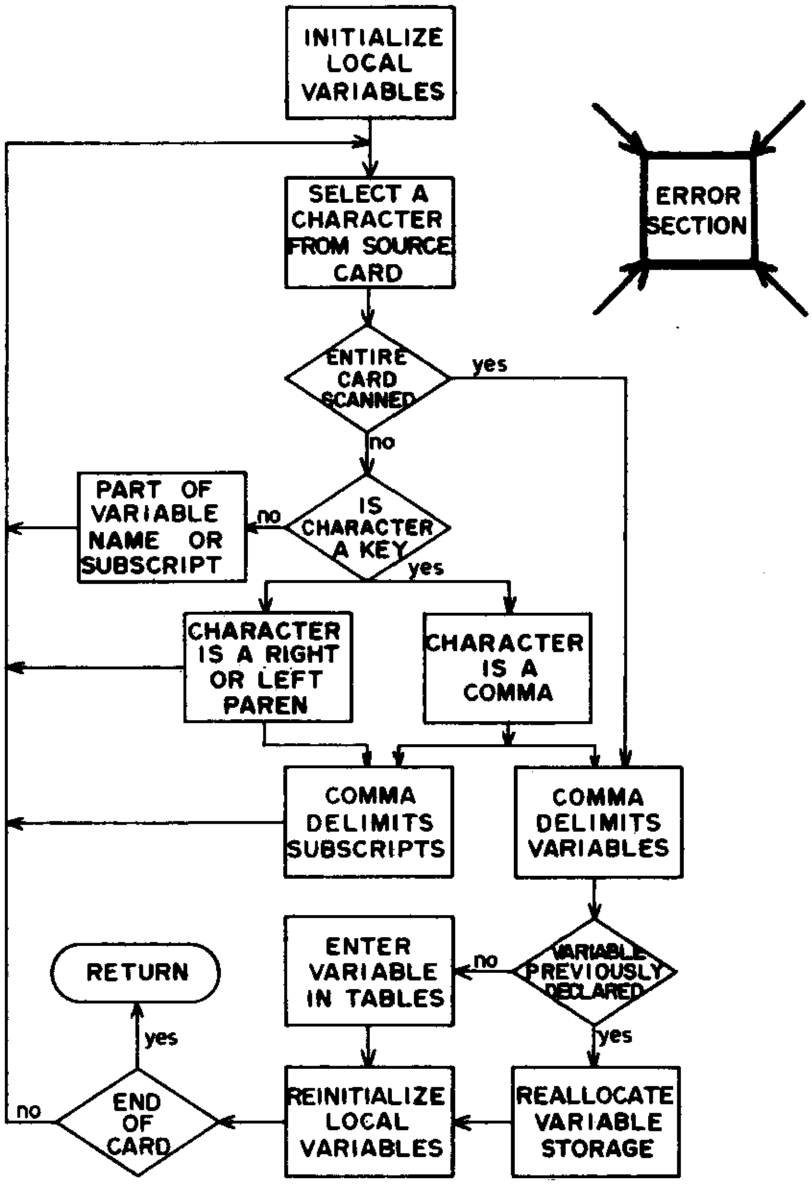
Parenthesis encountered


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
ERROR SECTION
ENTIRE CARD SCANNED
yes
no
IS CHARACTER A KEY
no
yes
PART OF VARIABLE NAME OR SUBSCRIPT
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
VARIABLE PREVIOUSLY DECLARED
no
yes
ENTER VARIABLE IN TABLES
RETURN
yes
END OF CARD
no
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE

```
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   75 IF (KODE.NE.1) GO TO 95
      KODE=2
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   80 IF (KODE.LT.2.O.KODE.GT.4) GO TO 95
      KFLG=.TRUE.
      GO TO 45
```

Line Number	Explanation
21142-21147	Left paren sets KODE=2; search begins for subscripts.
21148-21153	A right paren sets KFLG=.TRUE.. The right paren delimits the rightmost subscript of any variable.

Diagnostics

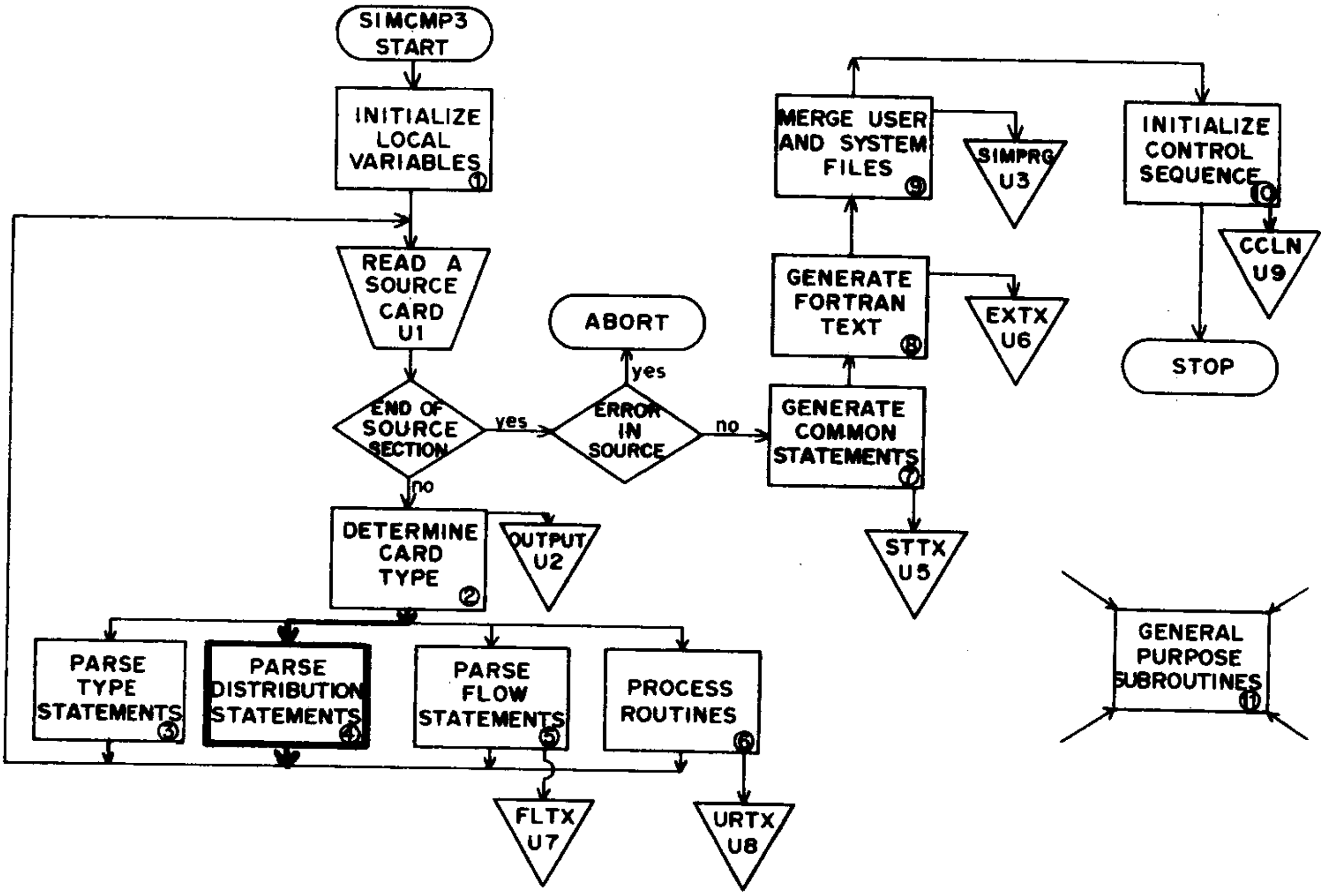

INITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
ENTIRE CARD SCANNED
yes
no
PART OF VARIABLE NAME OR SUBSCRIPT
no
IS CHARACTER A KEY
yes
CHARACTER IS A RIGHT OR LEFT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS SUBSCRIPTS
COMMA DELIMITS VARIABLES
RETURN
ENTER VARIABLE IN TABLES
no
VARIABLE PREVIOUSLY DECLARED
yes
yes
no
END OF CARD
REINITIALIZE LOCAL VARIABLES
REALLOCATE VARIABLE STORAGE

```
C
C.....GENERATE ERROR MESSAGES IF ERRORS ENCOUNTERED.
C
   95 WRITE (U2,170) ICHR
      FATAL=.TRUE.
      RETURN
  100 WRITE (U2,175) NAME
      FATAL=.TRUE.
      RETURN
  105 WRITE (U2,180) NSUB(J)
      FATAL=.TRUE.
      RETURN
  110 WRITE (U2,185)
      FATAL=.TRUE.
      RETURN
  115 WRITE (U2,190) NSUB(J)
```

```
21174              FATAL=.TRUE.
21175              RETURN
21176          120 WRITE (U2,195) NSUB(J)
21177              FATAL=.TRUE.
21178              RETURN
21179          125 WRITE (U2,200)
21180              FATAL=.TRUE.
21181              RETURN
21182          130 WRITE (U2,205) NAME
21183              GO TO 55
21184          135 WRITE (U2,210) NAME
21185              FATAL=.TRUE.
21186              RETURN


21203          160 WRITE (U2,220) NVMX
21204              FATAL=.TRUE.
21205              RETURN
21206          165 WRITE (U2,225)
21207              FATAL=.TRUE.
21208              RETURN
21209        C
21210          170 FORMAT (11H *****FE   , 11HCHARACTER ",A1, 12H" IS ILLEGAL)
21211          175 FORMAT (11H *****FE   , 10HVARIABLE ",A5, 27H..." IS LONGER THAN 5
21212            1 CHARS)
21213          180 FORMAT (11H *****FE   , 11HSUBSCRIPT ",A4, 27H..." IS LONGER THAN
21214            14 CHARS)
21215          185 FORMAT (11H *****FE   ,26HEXPECTED SUBSCRIPT MISSING)
21216          190 FORMAT (11H *****FE   , 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
21217          195 FORMAT (11H *****FE   , 11HSUBSCRIPT ",A4, 19H" GREATER THAN 1023)
21218          200 FORMAT (11H *****FE   ,30HEXPECTED VARIABLE NAME MISSING)
21219          205 FORMAT (11H *****NF   , 10HVARIABLE ",A5, 22H" BEGINS WITH CHAR "X
21220            1")
21221          210 FORMAT (11H *****FE   , 10HVARIABLE ",A5, 37H" BEGINS WITH A NON-A
21222            1LPHABETICAL CHAR)
21223          215 FORMAT (11H *****FE   , 10HVARIABLE ",A5, 31H" IS A RESERVED SYSTE
21224            1M VARIABLE)
21225          220 FORMAT (11H *****FE   , 42HNUMBER OF DECLARED VARIABLES HAS EXCEED
21226            1ED ,I6)
21227          225 FORMAT (11H *****FE   , 48HA VARIABLE DECLARATION IS INCOMPLETE AT
21228            1 CARD END)
21229          230 FORMAT (11H *****NF   ,10HVARIABLE (,A5,67H) HAS BEEN PREVIOUSLY D
21230            1ECLARED, LAST DECLARATION IS ASSUMED CORRECT)
21231          235 FORMAT (1H ,T12, 60HSTATE VARIABLES SHOULD NOT BE DECLARED IN STOR
21232            1AGE STATEMENTS)
21233        C
21234              END
```

Line Number	Explanation
21158-21186, 21203-21234	This is the error section. Control enters here after a syntax error has been encountered in a source card. There are only two nonfatal errors. A user may declare a variable beginning with the character X, or he may redefine a previously declared variable in a later declaration statement.

1.4. *Parse Distribution Statements*



Overview

The four following types of statement forms are processed by this section.

(1) UNIFØRM.<var1>,<var2>,···

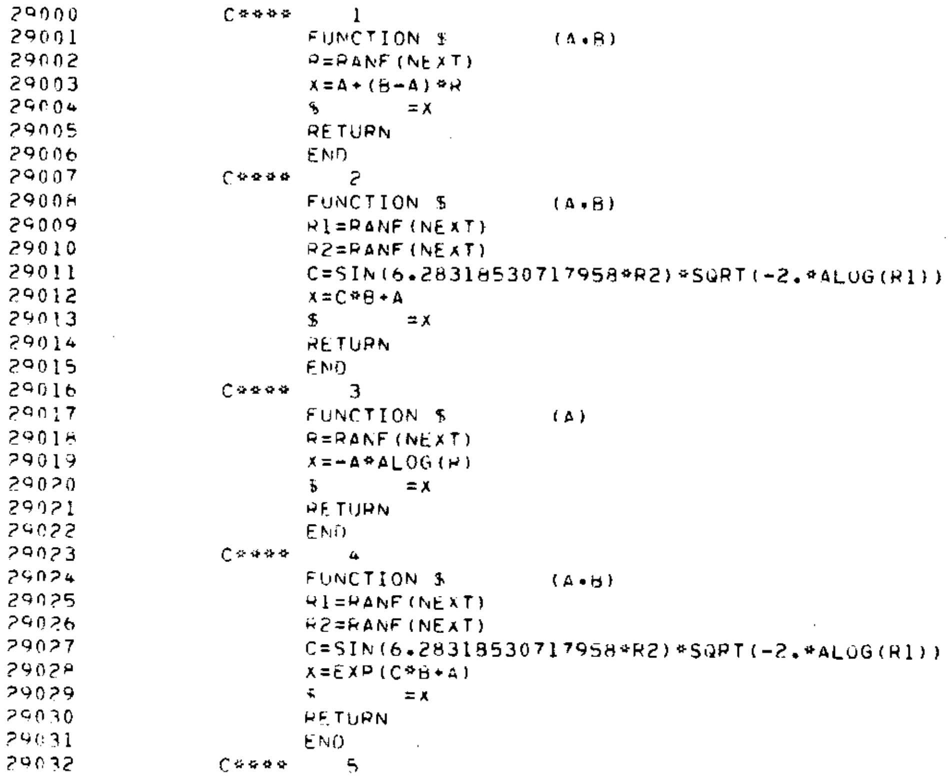
(2) NØRMAL. <var1>,<var2>,···

(3) EXPØNENT. <var1>,<var2>,···

(4) LØGNØRMAL. <var1>,<var2>,···

<var1> is a routine name containing seven or fewer characters. A FTN callable FUNCTIØN (of name <var1>) is generated which will return a random value from the distribution of which <var1> is specified as part.

The input to the section consists of the TEXT portion of a source card (<var1>,<var1>,•••) and a dummy FUNCTIØN file. The dummy file contains four routines used to calculate random variables for each distributional type. Following are the records of the dummy file: (i) The first FUNCTIØN routine will return a random value from a uniform distribution with end points A and B. (ii) The second returns a random value calculated from a normal distribution with end points A and B. (iii) The third calculates a random variable from an exponential distribution with mean = A. (iv) The fourth FUNCTIØN routine returns a random variable from a lognormal population with end points A and B.



```
C****     1
      FUNCTION $         (A,B)
      R=RANF(NEXT)
      X=A+(B-A)*R
      $       =X
      RETURN
      END
C****     2
      FUNCTION $         (A,B)
      R1=RANF(NEXT)
      R2=RANF(NEXT)
      C=SIN(6.28318530717958*R2)*SQRT(-2.*ALOG(R1))
      X=C*B+A
      $       =X
      RETURN
      END
C****     3
      FUNCTION $         (A)
      R=RANF(NEXT)
      X=-A*ALOG(R)
      $       =X
      RETURN
      END
C****     4
      FUNCTION $         (A,B)
      R1=RANF(NEXT)
      R2=RANF(NEXT)
      C=SIN(6.28318530717958*R2)*SQRT(-2.*ALOG(R1))
      X=EXP(C*B+A)
      $       =X
      RETURN
      END
C****     5
```

This section has four basic activities:

(1) Retrieves a variable name from the source TEXT.

(2) Searches dummy file for the routine matching of the distribution declared by the command portion of source card.

(3) Replaces each dollar sign ($) in the routine with the variable name.

(4) Writes the generated routine onto the user generated external routines file, URTX (the output of this section).

EXAMPLE. Examine the source card:

```
UNIFØRM. FØX
```

This card would generate the following routine written on to URTX.

```
FUNCTIØN FØX    (A,B)
R=RANF(NEXT)
X=A+(B-A)*R
FØX    =X
RETURN
END
```

The line-by-line section explanation follows.

```
25000            SUBROUTINE USIDF (CARD,FATAL)
25001            COMMON /ROUTINS/ NSUB,NSBL(100),SUBFLG(4),KDIST
25002            COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
25003            DIMENSION CARD(8), KARD(8)
25004            INTEGER U1,U2,U3,U4,U5,U6,U7,U8
25005            LOGICAL IN,KSTOP,FATAL
25006      C
25007      C.....THIS ROUTINE PROCESSES <UNIFORM.>, <NORMAL.>, <EXPONENT.>, AND
25008      C.....<LOGNORMAL.> USER VARIABLE DECLARATIONS.  FOR EACH VARIABLE DECLAR
25009      C.....ED AS A STANDARD STATISTICAL RANDOM VARIATE OF THE ABOVE TYPE AN
25010      C.....FTN CALLABLE FUNCTION OF APPROPRIATE TYPE IS GENERATED.
25011      C
25012            NCOL=0
25013            NAME=10H
25014            ICOL=6
25015            KSTOP=.FALSE.
25016      C
25017      C.....RETRIEVE THE ROUTINE NAMES.
25018      C
```

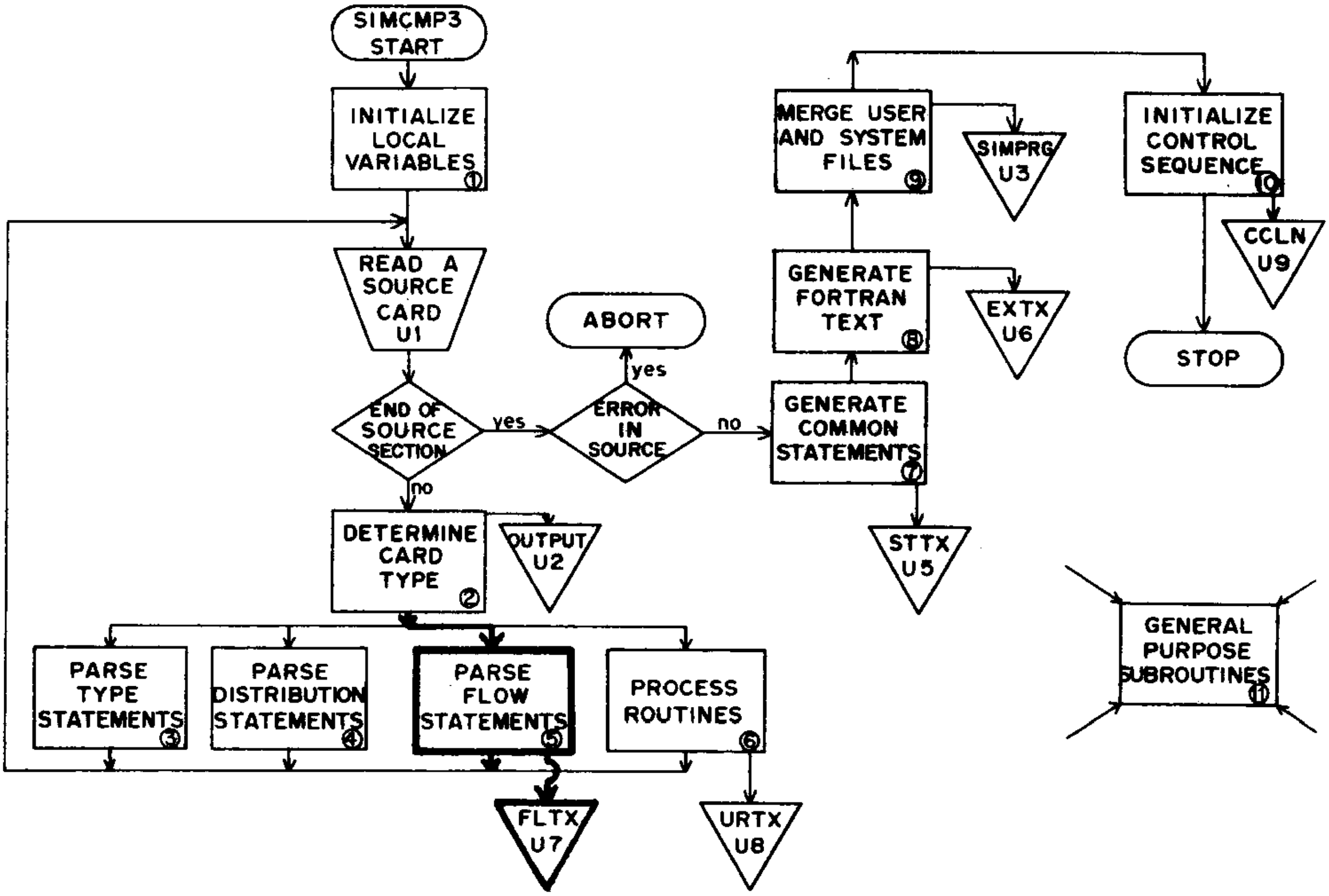
```
   15 ICOL=ICOL+1
      IF (ICOL.GT.72) GO TO 45
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 15
      IF (ICHR.EQ.1H.) GO TO 20
      IF (ICHR.LT.1HA.A.ICHR.GT.1H9) GO TO 50
      NCOL=NCOL+1
      IF (NCOL.GT.7) GO TO 55
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
C
C.....CHECK FOR A VALID ROUTINE NAME.
C
   20 IF (NCOL.LE.0) GO TO 60
      CALL GCHARS (NAME,1,1,ICHR)
      IF ((ICHR.GE.1HI.A.ICHR.LE.1HN).O.(ICHR.LT.1HA.O.ICHR.GT.1HZ)) GO
     1TO 65
C
C.....GENERATE THE ROUTINE.
C
      CALL POSITN (U4,1)
      IN=.FALSE.
   25 READ (U4,70) KARD
      IF (EOF(U4)) 40,30,40
   30 CALL GCHARS (KARD,1,5,LBL)
      IF (LBL.NE.5HC****) GO TO 35
      IF (IN) GO TO 40
      DECODE (10,75,KARD) NUM
      IF (NUM.EQ.KDIST) IN=.TRUE.
      GO TO 25
   35 IF (.N.IN) GO TO 25
      CALL GCHARS (KARD,16,1,ICHR)
      IF (ICHR.EQ.1H$) CALL SCHARS (KARD,16,7,NAME)
      CALL GCHARS (KARD,7,1,ICHR)
      IF (ICHR.EQ.1H$) CALL SCHARS (KARD,7,7,NAME)
      WRITE (U8,70) KARD
      GO TO 25
   40 IF (KSTOP) RETURN
      NCOL=0
      NAME=10H
      GO TO 15
   45 KSTOP=.TRUE.
      GO TO 20
   50 WRITE (U2,80) ICHR,ICOL
      FATAL=.TRUE.
      RETURN
   55 WRITE (U2,85) NAME
      FATAL=.TRUE.
      RETURN
   60 WRITE (U2,90)
      FATAL=.TRUE.
      RETURN
   65 WRITE (U2,95) NAME
      FATAL=.TRUE.
      RETURN
      WRITE (U2,100)
      FATAL=.TRUE.
      RETURN
C
   70 FORMAT (8A10)
   75 FORMAT (5X,I5)
   80 FORMAT (11H *****FE   , 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLU
     1MN ,I2)
   85 FORMAT (11H *****FE   , 14HROUTINE NAME ",A7, 24H..." LONGER THAN
     17 CHARS)
   90 FORMAT (11H *****FE   , 24HZERO LENGTH ROUTINE NAME)
```

```
25085          95 FORMAT (1IH *****FE    , 14HROUTINE NAME ",A7, 29H" STARTS WITH AN
25086            1ILLEGAL CHAR)
25087         100 FORMAT (11H *****FE    , 43HNUMBER OF USER-DEFINED ROUTINES EXCEEDS
25088            1 100)
25089       C
25090             END
```

Line Number	Explanation
25000	Only the TEXT portion of the distribution statement is sent to this section. FATAL=.TRUE. if a syntax error appears in the source card.
25012-25015	NAME is filled character by character with alphanumeric characters from the source TEXT. NAME will contain a variable <var1>. NCØL is the number of characters in NAME. KSTØP=.TRUE. when the end of TEXT (CARD) is reached.
25016-25028	Retrieve the variable name. A comma delimits variable (routine) names.
25029-25035	A routine name is invalid if the leading character does not implicitly type the variable as real.
25036-25040	PØSITN(I,J) is a subroutine that rewinds and skips J files on file number (name) I. In particular, PØSITN(U4,1) skips one file on file U4. The first file on U4 contains the SIMCØMP compiler and the second file, the one containing the dummy routines, is positioned for reading.
25041-25048	The dummy is scanned for cards of the form C**** I... where I represents the type of function (I=1 for uniform, 2 for normal, 3 for exponential, and 4 for lognormal). NUM contains the integer value of I. A match between KDIST and NUM indicates the proper routine has been located. (KDIST is set in CARDTP and KDIST=1 if a UNIFØRM. command is encountered, KDIST=2 upon parsing a NØRMAL. command, etc.)
25049-25055	The appropriate FUNCTIØN routine is written onto the user generated external routines file (URTX). The routine name in NAME is written over each dollar sign in the routine. Cards are written onto URTX until another C**** card is encountered, signifying the end of the routine.
25056-25059	KSTØP=.FALSE. means that a comma delimited the previous variable name. Therefore, reinitialize local variables and retrieve next variable name from CARD.

Line Number	Explanation
25060-25061	KSTØP=.TRUE. when the end of CARD is encountered. The end of CARD delimits the last variable name on the card. Go check for routine name validity.
25062-25090	Control arrives to the error section upon encountering any syntax error. FATAL=.TRUE. and the run will terminate after all of the source deck has been processed.

1.5. *Parse Flow Statements*



Overview

The three parts of a flow (flow commands, text, and end of flow) are processed by this section.

(1) Flow commands (flow declarative label) are of the form:

(<phrase>-<phrase>).

where <phrase> can be any of the following:

(a) <phrase> = <constant>

(b) <phrase> = <variable> = <constant>,<constant>

(c) <phrase> = <variable> = <constant>,<constant>,<constant>

(d) <phrase> = <variable> = <constant> * <variable> + <constant>

(e) <phrase> = <variable> = <constant> * <variable> - <constant>

Each constant or variable appearing in a phrase is called a field of that phrase.

EXAMPLE. (1-I=2,3).

The first phrase is called a constant phrase and consists of only one field, containing the integer 1.

The second phrase is called a DØ phrase and consists of three fields, I, 2, and 3. A DØ phrase is of form (b) or (c).

EXAMPLE. (I=1,11,5-J=3*I+0).

The first phrase is a DØ phrase.

The second phrase is an arithmetic phrase, form (d) or (e) above.

The fields of the second phrase are J, 3, I, and 0.

NOTE. If an arithmetic phrase is used, the other phrase must be a DØ phrase defining the variable in the third field of the arithmetic phrase. All phrases must appear exactly in one of the forms (a)-(e). For this reason the fourth field of the above arithmetic consists of 0 (SIMCØMF will not accept shortened phrases like J=3*I).

(2) Flow text is any FØRTRAN compilable statements following the flow command up to the next SIMCØMP recognizable command: another flow command, a FUNCTIØN routine, and EVENT, etc.

(3) An end of flow is processed when the next system command is encountered. The input to the section consists of any of the three parts of a flow. The section's biggest responsibilities are to:

(a) Parse flow commands--checking syntax.

(b) Expand DØ expressions. (1-I=2,3). is equivalent to (1-2). and (1-3)..

(c) Add each expanded flow to the flow stack NFLT(NFLW).

(d) Add each flow command, text, and end of flow to a flow definition text file, FLTX.

EXAMPLE. (2-I,3,5,2).

```
A=SIN(P*T)
FLØW=A
SUBRØUTINE START
   .
   .
   .
```

Assume the above flow was the first appearing in the source program. The flow stack, NFLT, would contain two flow commands and NFLW, the number of flow commands in the flow stack, would be incremented by 2.

NFLT(NFLW) = NFLT(1) = 200003_8

NFLT(2) = 200005_8

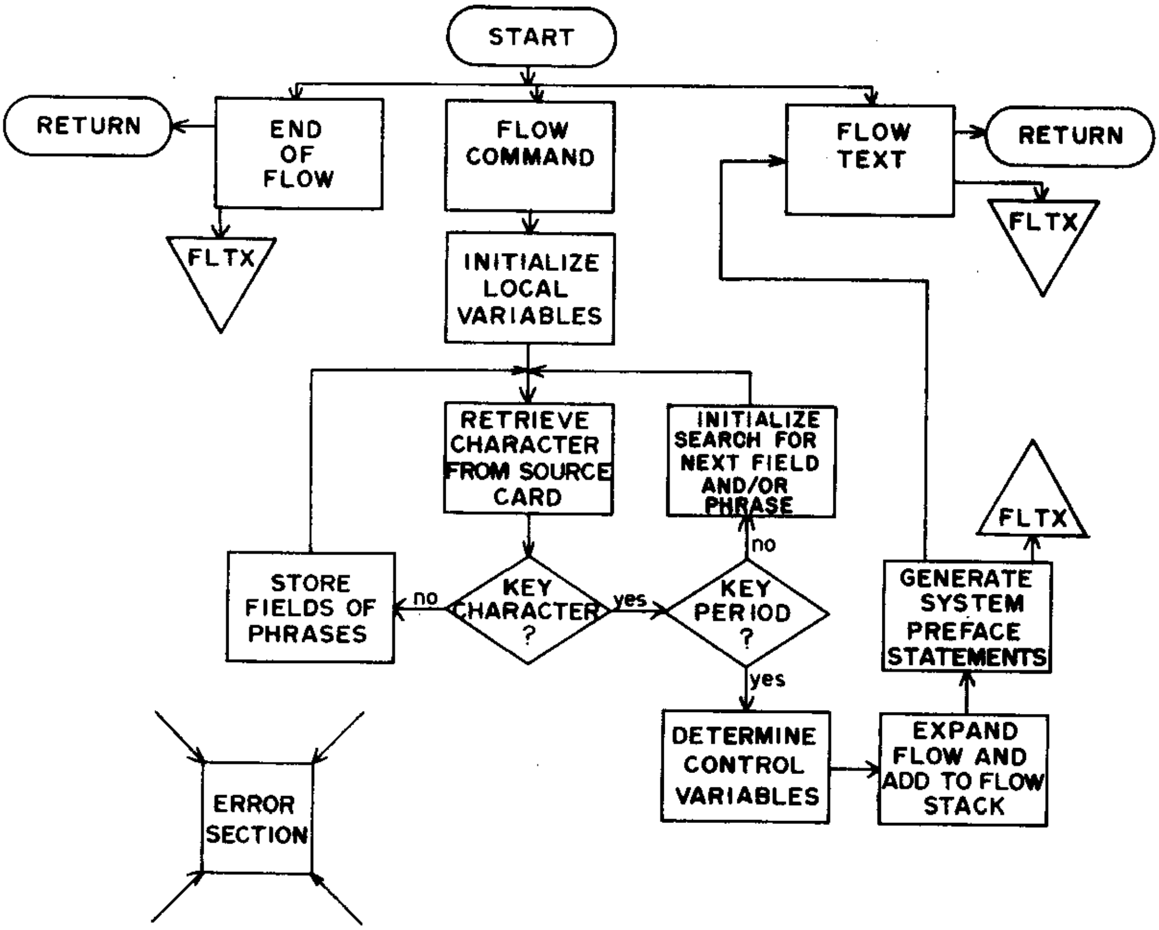
NFLW = 2

The file U7, (FLTX), would contain

```
     DØ 90001 I = 3,5,2
     FLØW = 17770 00000 00000 00000B .OR. (XMFL+1)
     A = SIN(P*T)
     FLØW = A
     XMFL = XMFL+1
     XFLW(XMFL) = FLØW
9001 CØNTINUE
```

The flow section is expanded into the following diagram to facilitate explanation.

Flow statements flow chart

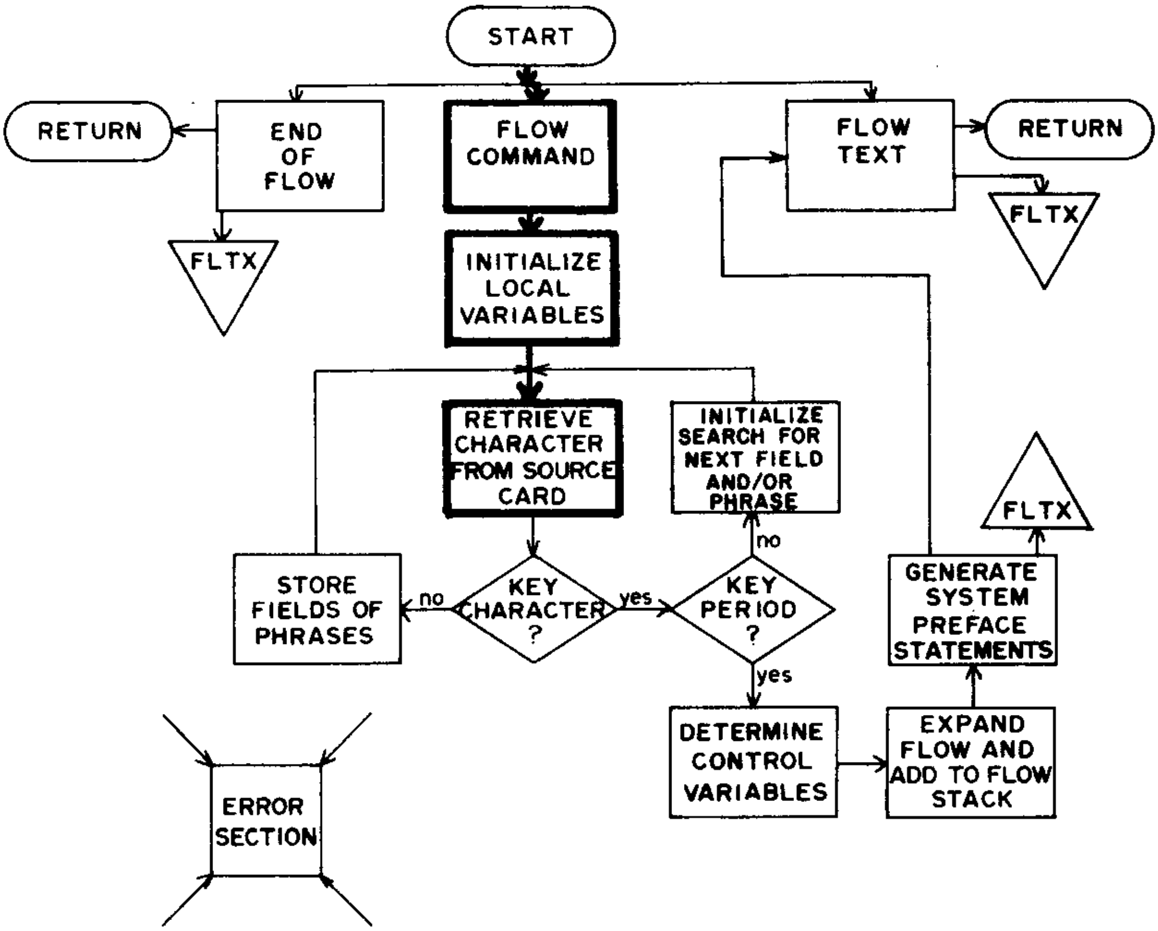

START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
no
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
ERROR SECTION
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK

```
      SUBROUTINE FL1DF (CARD,FATAL)
      COMMON NFLW,NRFL,NFMAX,NFLT(1)
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), KEY(8), NCOL(2), IPHZ(4,2), KNT(2,4), IFLD(2,4)
     1, KTYPE(2), NN(2,3), KK(2,2)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL FATAL,SWI,SWJ,S(2)
C
C.....THIS ROUTINE PARSES A FLOW DECLARATIVE LABEL OF THE FORM:
C.....(<PHRASE>-<PHRASE>).
C.....WHERE,
C.....<PHRASE>:=<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>,<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>,<CONSTANT>,<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>*<VARIABLE>+<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>*<VARIABLE>-<CONSTANT>
C.....FLOW REFERENCE TABLES ARE CREATED AND THE SYSTEM SUPPLIED PREFACE
C.....AND TERMINATION STATEMENTS ARE GENERATED.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H(,1H),1H.,1H=,1H,,1H*,1H+,1H-/
      DO 15 I=1,2
      DO 15 J=1,4
         IPHZ(J,I)=10H
         IFLD(I,J)=10H
         KNT(I,J)=0
```

```
   15 CONTINUE
      KTYPE(2)=0
      KTYPE(1)=KTYPE(2)
      NCOL(2)=0
      NCOL(1)=NCOL(2)
      KOD1=0
      KOD2=0
      ICOL=0
C
C.....RETRIEVE EACH CHARACTER IN THE FLOW LABEL, KEY CHARACTERS "KEY"
C.....CONTROL PROCESSING.
C
   20 ICOL=ICOL+1
      KSET=0
      IF (ICOL.GT.80) GO TO 150
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 20
      DO 25 I=1,8
         IF (ICHR.EQ.KEY(I)) GO TO 35
   25 CONTINUE
C
C....."ICHR" IS NOT A KEY CHARACTER, STORE THE COMPLETE SOURCE AND
C.....DESTINATION PHRASES.
C
   30 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      J=KOD1
      NCOL(J)=NCOL(J)+1
      IF (NCOL(J).GT.40) GO TO 160
      CALL SCHARS (IPHZ(1,J),NCOL(J),1,ICHR)
      IF (KSET.EQ.1) GO TO 20
C
C.....STORE THE FIELDS OF EACH PHRASE.
C
      IF (KOD2.LT.1.O.KOD2.GT.4) GO TO 155
      K=KOD2
      KNT(J,K)=KNT(J,K)+1
      IF (KNT(J,K).GT.10) GO TO 165
      CALL SCHARS (IFLD(J,K),KNT(J,K),1,ICHR)
      GO TO 20
C
C....."ICHR" IS A KEY CHARACTER.
C
   35 GO TO (40,45,50,55,60,65,70,75), I
C
C....."("...
C
   40 IF (KOD1.NE.0) GO TO 155
      KOD1=1
      KOD2=1
      GO TO 20
C
C.....")"...
C
   45 IF (KOD1.NE.2) GO TO 155
      KOD1=3
      GO TO 20
C
C....." "...
C
   50 IF (KOD1.NE.3) GO TO 155
      GO TO 95
C
C....."="...
C
   55 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      IF (KOD2.NE.1) GO TO 155
      KTYPE(KOD1)=KTYPE(KOD1)+1
      KOD2=2
      KSET=1
      GO TO 30
C
C.....","...
C
   60 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
      KOD2=KOD2+1
      KSET=1
      GO TO 30
C
C....."*"...
C
   65 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      IF (KOD2.NE.2) GO TO 155
      KTYPE(KOD1)=KTYPE(KOD1)+1
      KOD2=3
      KSET=1
      GO TO 30
C
C....."+"...
C
   70 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
      IF (KOD2.EQ.2) GO TO 30
      KOD2=4
      GO TO 30
C
C....."-"...
C
   75 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      IF (KOD1.EQ.1) GO TO 80
      IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
      IF (KOD2.EQ.2) GO TO 30
      KOD2=4
      GO TO 30
   80 IF (KOD2.LT.1.O.KOD2.GT.4) GO TO 155
      IF (KOD2.GT.1.A.KOD2.LT.4) GO TO 85
      KOD1=2
      KOD2=1
      GO TO 20
   85 IF (KOD2.EQ.2) GO TO 30
      IF (KTYPE(KOD1).EQ.2) GO TO 90
      KOD1=2
      KOD2=1
      GO TO 20
   90 KOD2=4
      GO TO 30
C
C.....PARSING OF LABEL COMPLETED.
C.....CHECK FOR LEGAL COMBINATION OF PHRASE TYPES.
C
   95 IF (IABS(KTYPE(1)-KTYPE(2)).GE.2) GO TO 170
      IF (KTYPE(1).EQ.2.A.KTYPE(2).EQ.2) GO TO 170
C
C.....INITIALIZE ITERATION CONTROL VARIABLES.
C
      NN(6)=1
      NN(5)=NN(6)
      NN(4)=NN(5)
      NN(3)=NN(4)
      NN(2)=NN(3)
      NN(1)=NN(2)
      KK(4)=0
      KK(3)=KK(4)
      KK(2)=KK(3)
      KK(1)=KK(2)
      S(2)=.FALSE.
      S(1)=S(2)
C
C.....SET CONTROL VARIABLES TO VALUES IN APPROPRIATE FIELDS AND CHECK
C.....VALUES AND VARIABLE NAMES IN FIELDS.
C
      DO 125 I=1,2
         IGO=KTYPE(I)+1
         GO TO (100,105,115), IGO
C
C.....KTYPE=0, PHRASE ASSUMED A CONSTANT.
C
  100    KNTR=KNT(I,1)
         IF (KNTR.LE.0) GO TO 185
         ENCODE (10,205,FMT) KNTR
         DECODE (KNTR,FMT,IFLD(I,1)) NN(I,1)
         NN(I,2)=NN(I,1)
         GO TO 125
C
C.....KTYPE(I)=1, PHRASE ASSUMED A DO... EXPRESSION.
C
  105    IF (KNT(I,1).LT.1.O.KNT(I,1).GT.5) GO TO 175
         CALL GCHARS (IFLD(I,1),1,1,ICHR)
         IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 175
         DO 110 J=1,3
            KNTR=KNT(I,J+1)
            IF (KNTR.LE.0) GO TO 110
            ENCODE (10,205,FMT) KNTR
            DECODE (KNTR,FMT,IFLD(I,J+1)) NN(I,J)
  110    CONTINUE
         GO TO 125
C
C.....KTYPE(I)=2, PHRASE ASSUMED AN ARITHMETIC EXPRESSION.
C
  115    IF (KNT(I,1).LT.1.O.KNT(I,1).GT.5) GO TO 175
         CALL GCHARS (IFLD(I,1),1,1,ICHR)
         IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 175
         K=3-I
         IF (IFLD(I,3).NE.IFLD(K,1)) GO TO 180
         DO 120 J=1,2
            KNTR=KNT(I,2*J)
            IF (KNTR.LE.0) GO TO 185
            ENCODE (10,205,FMT) KNTR
  120    DECODE (KNTR,FMT,IFLD(I,2*J)) KK(I,J)
         S(I)=.TRUE.
  125 CONTINUE
      DO 130 I=1,2
      DO 130 J=1,3
         IF (NN(I,J).LT.1) GO TO 185
  130 CONTINUE
      N1=NN(1,1)
      N2=NN(1,2)
      N3=NN(1,3)
      M1=NN(2,1)
      M2=NN(2,2)
      M3=NN(2,3)
      K1=KK(1,1)
      K2=KK(1,2)
      L1=KK(2,1)
      L2=KK(2,2)
      SWI=S(1)
      SWJ=S(2)
C
C.....ITERATE THROUGH ALL FLOWS DECLARED CHECKING THE RANGE OF FLOW
C.....INDICES AND CREATING A COMMENT RECORD ON THE TEMPORARY FLOW STOR
C.....FILE OF EACH EXISTING FLOW.
C
      DO 135 II=N1,N2,N3
         I=II
      DO 135 JJ=M1,M2,M3
         J=JJ
         IF (SWI) I=K1*JJ+K2
         IF (SWJ) J=L1*II+L2
         IF (I.LT.1.O.I.GT.999) GO TO 190
         IF (J.LT.1.O.J.GT.999) GO TO 190
         NFLW=NFLW+1
         IF (NFLW.GT.NFMAX) GO TO 195
         I2P15=SHIFT(I,15)
  135 NFLT(NFLW)=I2P15.O.J
C
C.....GENERATE SYSTEMS SUPPLIED PREFACE STATEMENTS.
C
      IF (KTYPE(1).EQ.0.A.KTYPE(2).EQ.0) GO TO 145
      NRFL=NRFL+1
      MLAB=I1ZR(NRFL)
      DO 140 I=1,2
         IF (KTYPE(I).EQ.0.O.KTYPE(I).EQ.2) GO TO 140
         WRITE (U7,210) MLAB,(IPHZ(J,I),J=1,4)
         K=3-I
```

```
            IF (KTYPE(K).NE.2) GO TO 140
            WRITE (U7,215) (IPHZ(J,K),J=1,4)
  140 CONTINUE
  145 WRITE (U7,220)
      RETURN
C
C.....IF ERRORS ENCOUNTERED ISSUE A DIAGNOSTIC.
C
  150 WRITE (U2,235)
      FATAL=.TRUE.
      RETURN
  155 WRITE (U2,240) ICHR,ICOL
      FATAL=.TRUE.
      RETURN
  160 WRITE (U2,245) (IPHZ(I,J),I=1,4)
      FATAL=.TRUE.
      RETURN
  165 WRITE (U2,250) IFLD(J,K)
      FATAL=.TRUE.
      RETURN
  170 WRITE (U2,255)
      FATAL=.TRUE.
      RETURN
  175 WRITE (U2,260) IFLD(I,1)
      FATAL=.TRUE.
      RETURN
  180 WRITE (U2,265)
      FATAL=.TRUE.
      RETURN
  185 WRITE (U2,270)
      FATAL=.TRUE.
      RETURN
  190 WRITE (U2,275) I,J
      FATAL=.TRUE.
      RETURN
  195 IF (NFMAX.LT.9999) GO TO 200
      WRITE (U2,280)
      FATAL=.TRUE.
      RETURN
  200 WRITE (U2,285) NFMAX
      FATAL=.TRUE.
      RETURN
C
C
      ENTRY FL2DF
C
C.....THIS ROUTINE GENERATES THE SYSTEM TERMINATION TEXT FOR A FLOW.
C
      WRITE (U7,225)
      IF (KTYPE(1).EQ.0.A.KTYPE(2).EQ.0) RETURN
      WRITE (U7,230) MLAB
      RETURN
C
  205 FORMAT (2H(I,I2,6H)      )
  210 FORMAT (6X,  4HDO 9,R4,1X,4A10,25X)
  215 FORMAT (6X,4A10,34X)
  220 FORMAT (6X, 41HFLOW=17770 00000 00000 00000B.OR.(XMFL+1),33X)
  225 FORMAT (6X, 11HXMFL=XMFL+1,63X/6X, 15HXFLW(XMFL)=FLOW,59X)
  230 FORMAT (  1H9,R4,  9H CONTINUE,66X)
  235 FORMAT (11H *****FE    , 39HFLOW DIRECTIVE UNTERMINATED AT CARD END
     1)
  240 FORMAT (11H *****FE    , 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLU
     1MN ,I2)
  245 FORMAT (11H *****FE    , 13HFLOW PHRASE ",4A10, 42H..." CONTAINS MO
     1RE THAN 40 NON-BLANK CHARS)
  250 FORMAT (11H *****FE    , 27HFLOW EXPRESSION SUB-FIELD ",A10, 42H...
     1! CONTAINS MORE THAN 10 NON-BLANK CHARS)
  255 FORMAT (11H *****FE    , 65HARITHMETIC PHRASE MUST BE USED IN CONJU
     1NCTION WITH A DO... PHRASE)
  260 FORMAT (11H *****FE    , 40HFLOW ITERATION PHRASE CONTROL VARIABLE
     1',A5, 43H| MUST BE A 5 CHAR OR LESS INTEGER VARIABLE)
  265 FORMAT (11H *****FE    , 78HTHE DO... PHRASE CONTROL VARIABLE MUST
     1BE THE OPERAND IN THE ARITHMETIC PHRASE)
  270 FORMAT (11H *****FE    , 67HA FIELD IN WHICH A CONSTANT SHOULD APPE
     1AR IS MISSING OR IS NEGATIVE)
  275 FORMAT (11H *****FE    , 14HFLOW INDICES (,I4,  1H-,I4, 59H) PRODUC
     1ED BY THE ABOVE LABEL ARE OUTSIDE THE RANGE 1 - 999)
  280 FORMAT (11H *****FE    , 28HNUMBER OF FLOWS EXCEEDS 9999)
  285 FORMAT (11H *****FE    , 57HINSUFFICIENT FIELD LENGTH, INCREASE BY
     1(NO. OF FLOWS  -   ,I4,  1H))
C
      END
      FUNCTION I1ZR(I)
      DATA ZEROS/33333333333333333333B/
      ENCODE (10,25,NUM) I
      ND=ALOG10(FLOAT(I))+1
      MASK=0
      IF (ND.LE.0) GO TO 20
      DO 15 J=1,ND
   15 MASK=MASK+63*2**(6*J-6)
   20 I1ZR=(NUM.A.MASK).O.(.N.MASK.A.ZEROS)
      RETURN
C
   25 FORMAT (I10)
C
      END
```

Initialize and retrieve characters

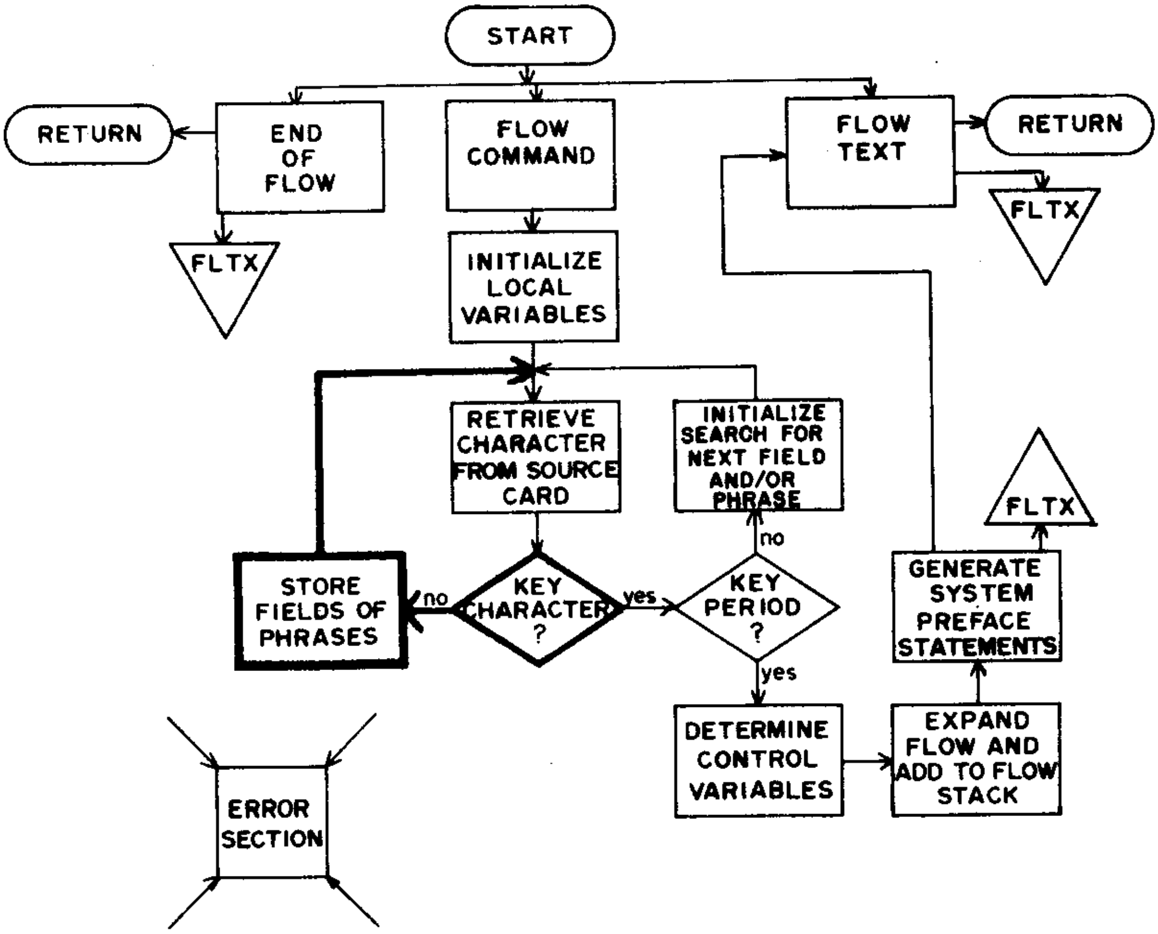

START
RETURN
END OF FLOW
FLTX
FLOW COMMAND
INITIALIZE LOCAL VARIABLES
FLOW TEXT
RETURN
FLTX
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
no
yes
GENERATE SYSTEM PREFACE STATEMENTS
ERROR SECTION
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK

```
      SUBROUTINE FLIDF (CARD,FATAL)
      COMMON NFLW,NRFL,NFMAX,NFLT(1)
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION CARD(8), KEY(8), NCOL(2), IPHZ(4,2), KNT(2,4), IFLD(2,4)
     1, KTYPE(2), NN(2,3), KK(2,2)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL FATAL,SWI,SWJ,S(2)
C
C.....THIS ROUTINE PARSES A FLOW DECLARATIVE LABEL OF THE FORM:
C.....(<PHRASE>-<PHRASE>).
C.....WHERE,
C.....<PHRASE>:=<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>,<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>,<CONSTANT>,<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>*<VARIABLE>+<CONSTANT>
C.....<PHRASE>:=<VARIABLE>=<CONSTANT>*<VARIABLE>-<CONSTANT>
C.....FLOW REFERENCE TABLES ARE CREATED AND THE SYSTEM SUPPLIED PREFACE
C.....AND TERMINATION STATEMENTS ARE GENERATED.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H(,1H),1H.,1H=,1H,,1H*,1H+,1H-/
      DO 15 I=1,2
      DO 15 J=1,4
         IPHZ(J,I)=10H
         IFLD(I,J)=10H
         KNT(I,J)=0
```

```
23027          15 CONTINUE
23028             KTYPE(2)=0
23029             KTYPE(1)=KTYPE(2)
23030             NCOL(2)=0
23031             NCOL(1)=NCOL(2)
23032             KOD1=0
23033             KOD2=0
23034             ICOL=0
23035       C
23036       C.....RETRIEVE EACH CHARACTER IN THE FLOW LABEL, KEY CHARACTERS "KEY"
23037       C.....CONTROL PROCESSING.
23038       C
23039          20 ICOL=ICOL+1
23040             KSET=0
23041             IF (ICOL.GT.80) GO TO 150
23042             CALL GCHARS (CARD,ICOL,1,ICHR)
23043             IF (ICHR.EQ.1H ) GO TO 20
```

Line Number	Explanation
23000	FLIDF processes the command portion of a FLØW. CARD contains only the command portion of the source card.
23019-23035	Initialize local variables: KEY contains all special characters that can appear in a flow command. NCØL(J) where J=1 is a count of the number of characters appearing in the first (source) phrase. NCØL(2) will contain the number of characters in the second or destination phrase. IPHZ(K,J) where K=1,4 is a list of the characters of the Jth phrase. KNT(J,K) contains the number of characters in the Kth field of the Jth phrase. IFLD(J,K) is the list of the characters in the Kth field of the Jth phrase. KTYPE(J) is the type of the Jth phrase. = 0 If the phrase is a constant. = 1 If a DØ phrase. = 2 If an arithmetic expression. KØD1=J, indicating the Jth phrase is being parsed. KØD2=K, the Kth field of the Jth phrase is being processed. KSET=1 if a special character is encountered (KEY character).
23036-23043	The flow label is scanned character by character. ICHR contains one character of the flow label (command).

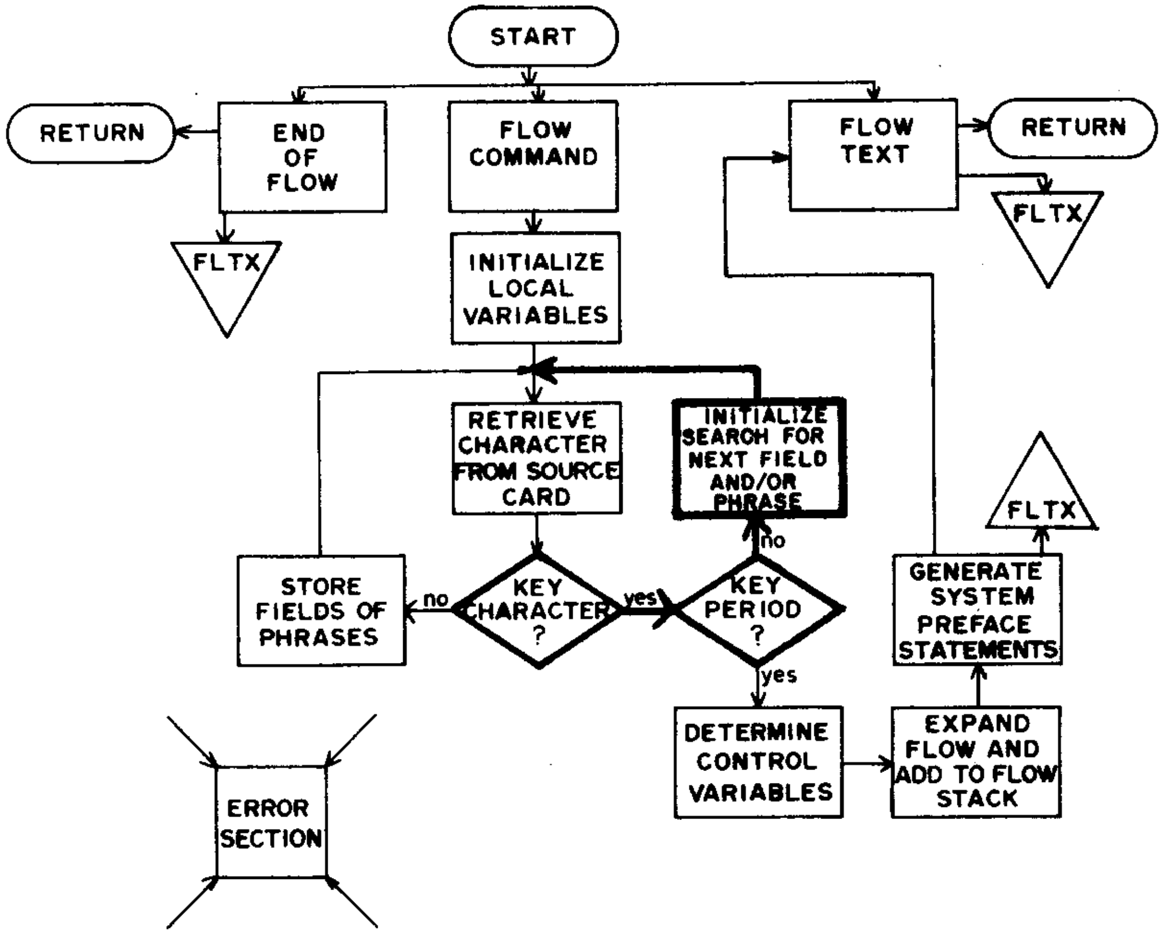
Store fields of phrases


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
no
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

```
      DO 25 I=1,8
         IF (ICHR.EQ.KEY(I)) GO TO 35
   25 CONTINUE
C
C....."ICHR" IS NOT A KEY CHARACTER. STORE THE COMPLETE SOURCE AND
C.....DESTINATION PHRASES.
C
   30 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
      J=KOD1
      NCOL(J)=NCOL(J)+1
      IF (NCOL(J).GT.40) GO TO 160
      CALL SCHARS (IPHZ(1,J),NCOL(J),1,ICHR)
      IF (KSET.EQ.1) GO TO 20
C
C.....STORE THE FIELDS OF EACH PHRASE.
C
      IF (KOD2.LT.1.O.KOD2.GT.4) GO TO 155
      K=KOD2
      KNT(J,K)=KNT(J,K)+1
      IF (KNT(J,K).GT.10) GO TO 165
      CALL SCHARS (IFLD(J,K),KNT(J,K),1,ICHR)
      GO TO 20
```

Line Number	Explanation
23044-23046	The character is checked against KEY to determine whether it is a key character (delimits phrases and/or fields).
23048-23056	If the character is not a key character, increment the phrase counter NCØL, and store the character in the phrase list, IPHZ.
23057-23065	Since the character is not a key character (KSET≠1), it must be a part of a field. Store character in the appropriate field, IFLD.

Key character begins search for next field


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
no
yes
GENERATE SYSTEM PREFACE STATEMENTS
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

```
C
C....."ICHR" IS A KEY CHARACTER.
C
  35 GO TO (40,45,50,55,60,65,70,75), I
C
C....."("...
C
  40 IF (KOD1.NE.0) GO TO 155
     KOD1=1
     KOD2=1
     GO TO 20
C
C.....")"...
C
  45 IF (KOD1.NE.2) GO TO 155
     KOD1=3
     GO TO 20
C
C....." "...
C
  50 IF (KOD1.NE.3) GO TO 155
     GO TO 95
C
C....."="...
C
  55 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
     IF (KOD2.NE.1) GO TO 155
     KTYPE(KOD1)=KTYPE(KOD1)+1
```

```
23094              KOD2=2
23095              KSET=1
23096              GO TO 30
23097        C
23098        C....."."...
23099        C
23100           60 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
23101              IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
23102              KOD2=KQD2+1
23103              KSET=1
23104              GO TO 30
23105        C
23106        C....."*"...
23107        C
23108           65 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
23109              IF (KOD2.NE.2) GO TO 155
23110              KTYPE(KOD1)=KTYPE(KOD1)+1
23111              KOD2=3
23112              KSET=1
23113              GO TO 30
23114        C
23115        C....."+"...
23116        C
23117           70 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
23118              IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
23119              IF (KOD2.EQ.2) GO TO 30
23120              KOD2=4
23121              GO TO 30
23122        C
23123        C....."-"...
23124        C
23125           75 IF (KOD1.LT.1.O.KOD1.GT.2) GO TO 155
23126              IF (KOD1.EQ.1) GO TO 80
23127              IF (KOD2.LT.2.O.KOD2.GT.3) GO TO 155
23128              IF (KOD2.EQ.2) GO TO 30
23129              KOD2=4
23130              GO TO 30
23131           80 IF (KOD2.LT.1.O.KOD2.GT.4) GO TO 155
23132              IF (KOD2.GT.1.A.KOD2.LT.4) GO TO 85
23133              KOD1=2
23134              KOD2=1
23135              GO TO 20
23136           85 IF (KOD2.EQ.2) GO TO 30
23137              IF (KTYPE(KOD1).EQ.2) GO TO 90
23138              KOD1=2
23139              KOD2=1
23140              GO TO 20
23141           90 KOD2=4
23142              GO TO 30
```

Line Number	Explanation
23069	The character is a key and delimits a phrase and/or a field.
23070-23076	The character is a left paren, signifying the beginning of the first phrase and first field. Continue parsing by retrieving next character.
23077-23082	The character is a right paren, delimiting the second phrase. The next character must be a period.
23083-23087	A period is encountered signalling that the entire flow command has been parsed. Proceed to determine control variables from local variable tables.

Line Number	Explanation
23088-23096	An equal sign (=) is the key character. The phrase type is a DØ expression (KTYPE=1). Begin search for the second field of phrase (KØD2=2). An equal (=) sign can appear only between fields 1 and 2.
23097-23104	A comma is encountered, delimiting fields. Enter character in phrase list, and look for next field. (All characters but the left and right paren and the period are entered in the phrase list, IPHZ.)
23105-23113	An asterisk is encountered. The phrase must be an arithmetic expression. Look for the third field. An * can appear only between fields 2 and 3.
23114-23121	A plus (+) sign is encountered. Look for fourth field of the phrase.
23125-23130	A minus (-) sign is encountered in the second phrase (KØD1=2). If KØD2=3, the minus sign is interpreted as delimiting the third and fourth fields of the phrase. The phrase is determined to be an arithmetic phrase. Set KØD2=4 and begin parsing fourth field of phrase. If KØD2=2, the special character (-) is interpreted as a negative of the contents of the second field, i.e., a arithmetic phrase similar to J=-3*I+6.
23131-23135	The character is interpreted as a dash delimiting phrases. Set KØD1=2 and begin parse of second phrase.
23136-23142	If KØD2=2, the special character is interpreted as a negation of the contents of the second field. If the type of phrase has already been classified as an arithmetic phrase (KTYPE(1)=2) and KØD2=3, then the character is a minus sign delimiting the third and fourth fields of phrase 1 (line 23141). Otherwise, the character is assumed a dash delimiting phrases (line 23138).

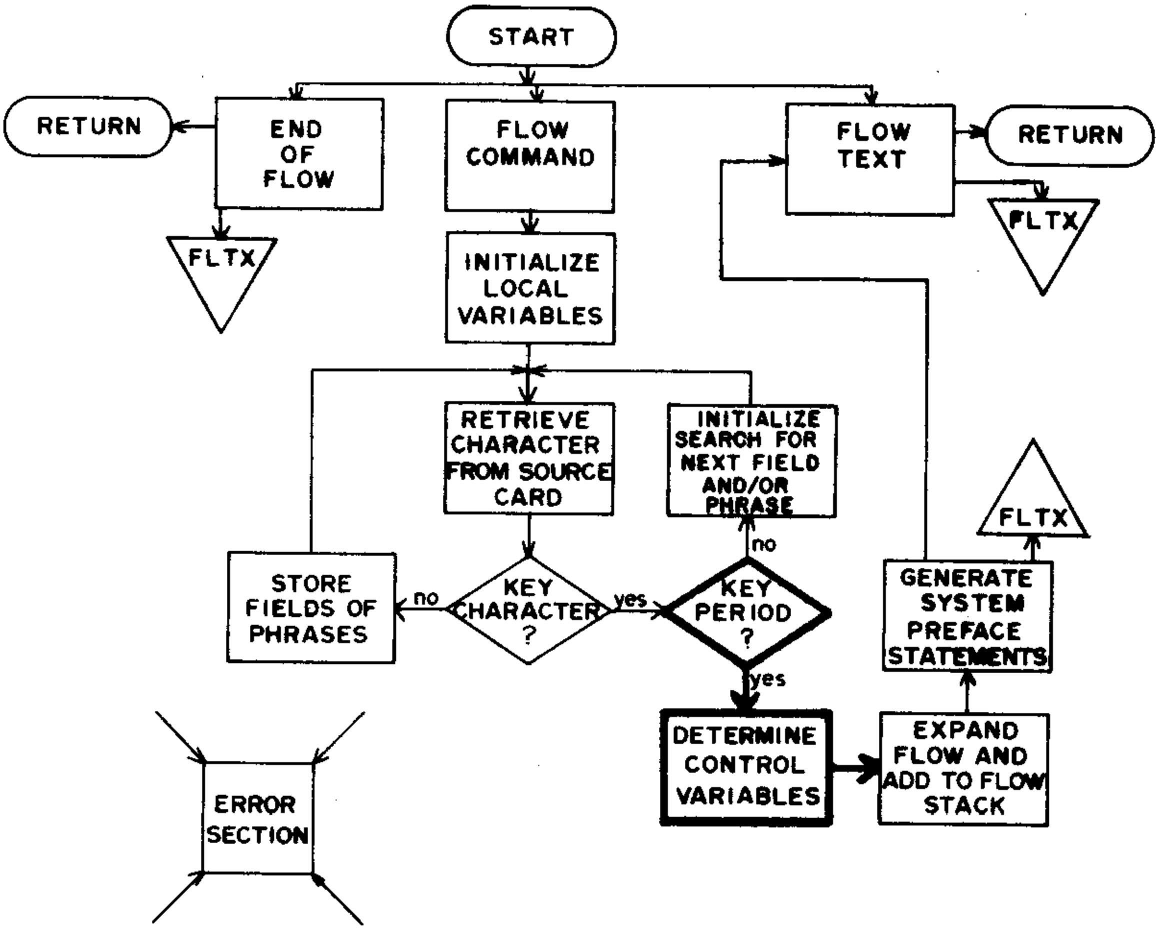
EXAMPLE. To illustrate the functions of the local variables, examine the flow as it is parsed character by character.

(I=1,11,5-J=3*I+0).

ICHR	KØD1=J	NCØL(J)	IPHZ(1,J)	KØD2=K	KNT(J,K)	IFLD(J,K)
(	1			1		
I	1	1	I	1	1	I
=	1	2	I=	2		
1	1	3	I=1	2	1	1
,	1	4	I=1,	3		
1	1	5	I=1,1	3	1	1
1	1	6	I=1,11	3	2	11
,	1	7	I=1,11,	4		
5	1	8	I=1,11,5	4	1	5
-	2			1		
J	2	1	J	1	1	J
=	2	2	J=	2		
3	2	3	J=3	2	1	3
*	2	4	J=3*	3		
I	2	5	J=3*I	3	1	I
+	2	6	J=3*I+	4		
0	2	7	J=3*I+0	4	1	0
)	3					
.	Parsing done					

The contents of NCØL, IPHZ, KNT, and IFLD are used to add flows to the flow stack and generate system preface statements.

Determine control variables


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
no
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

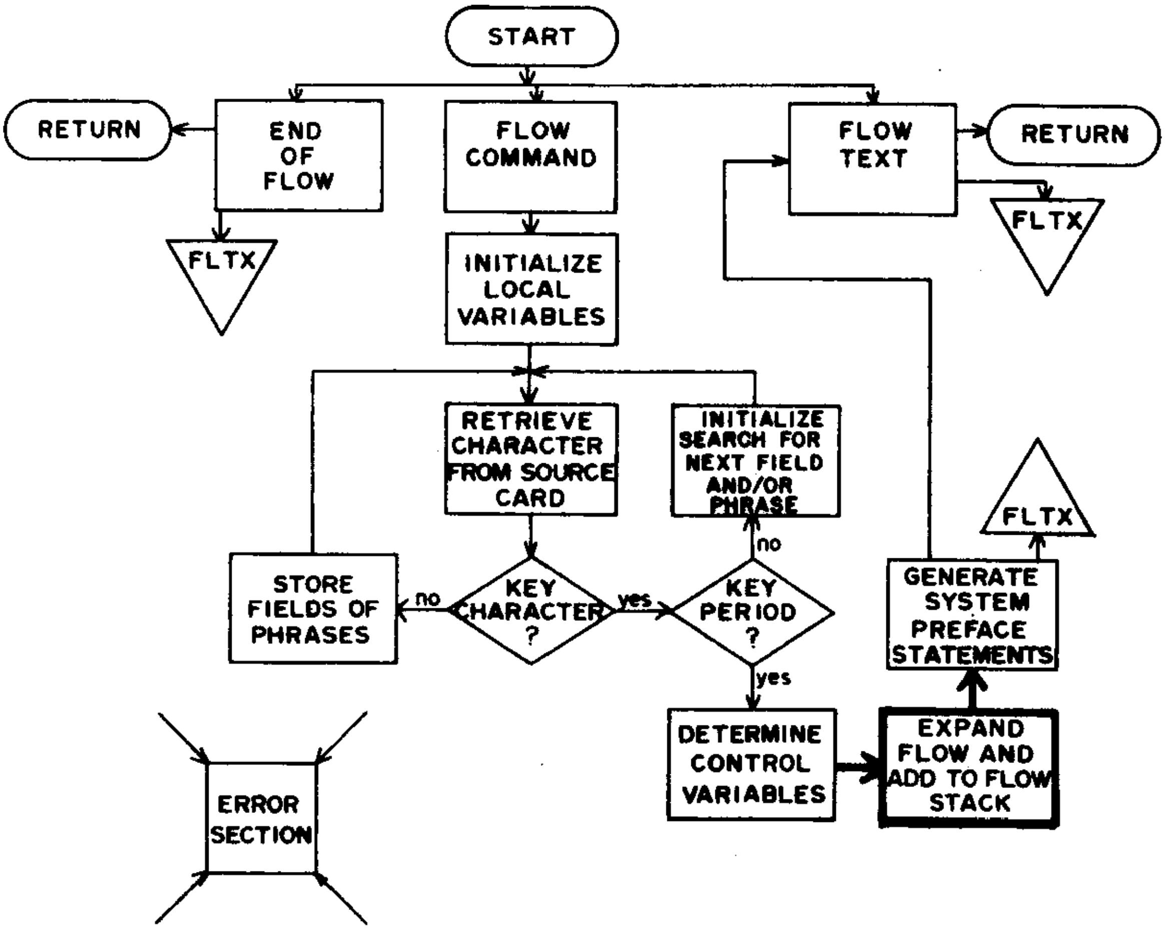
```
C
C.....PARSING OF LABEL COMPLETED.
C.....CHECK FOR LEGAL COMBINATION OF PHRASE TYPES.
C
   95 IF (IABS(KTYPE(1)-KTYPE(2)).GE.2) GO TO 170
      IF (KTYPE(1).EQ.2.A.KTYPE(2).EQ.2) GO TO 170
C
C.....INITIALIZE ITERATION CONTROL VARIABLES.
C
      NN(6)=1
      NN(5)=NN(6)
      NN(4)=NN(5)
      NN(3)=NN(4)
      NN(2)=NN(3)
      NN(1)=NN(2)
      KK(4)=0
      KK(3)=KK(4)
      KK(2)=KK(3)
      KK(1)=KK(2)
```

```
23162           S(2)=.FALSE.
23163           S(1)=S(2)
23164     C
23165     C.....SET CONTROL VARIABLES TO VALUES IN APPROPRIATE FIELDS AND CHECK
23166     C.....VALUES AND VARIABLE NAMES IN FIELDS.
23167     C
23168           DO 125 I=1,2
23169              IGO=KTYPE(I)+1
23170              GO TO (100,105,115), IGO
23171     C
23172     C.....KTYPE=0, PHRASE ASSUMED A CONSTANT.
23173     C
23174       100    KNTR=KNT(I,1)
23175              IF (KNTR.LE.0) GO TO 185
23176              ENCODE (10,205,FMT) KNTR
23177              DECODE (KNTR,FMT,IFLD(I,1)) NN(I,1)
23178              NN(I,2)=NN(I,1)
23179              GO TO 125
23180     C
23181     C.....KTYPE(I)=1, PHRASE ASSUMED A DO... EXPRESSION.
23182     C
23183       105    IF (KNT(I,1).LT.1.O.KNT(I,1).GT.5) GO TO 175
23184              CALL GCHARS (IFLD(I,1),1,1,ICHR)
23185              IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 175
23186              DO 110 J=1,3
23187                 KNTR=KNT(I,J+1)
23188                 IF (KNTR.LE.0) GO TO 110
23189                 ENCODE (10,205,FMT) KNTR
23190                 DECODE (KNTR,FMT,IFLD(I,J+1)) NN(I,J)
23191       110    CONTINUE
23192              GO TO 125
23193     C
23194     C.....KTYPE(I)=2, PHRASE ASSUMED AN ARITHMETIC EXPRESSION.
23195     C
23196       115    IF (KNT(I,1).LT.1.O.KNT(I,1).GT.5) GO TO 175
23197              CALL GCHARS (IFLD(I,1),1,1,ICHR)
23198              IF (ICHR.LT.1HA.O.ICHR.GT.1HZ) GO TO 175
23199              K=3-I
23200              IF (IFLD(I,3).NE.IFLD(K,1)) GO TO 180
23201              DO 120 J=1,2
23202                 KNTR=KNT(I,2*J)
23203                 IF (KNTR.LE.0) GO TO 185
23204                 ENCODE (10,205,FMT) KNTR
23205       120    DECODE (KNTR,FMT,IFLD(I,2*J)) KK(I,J)
23206              S(I)=.TRUE.
23207       125 CONTINUE
23208           DO 130 I=1,2
23209           DO 130 J=1,3
23210              IF (NN(I,J).LT.1) GO TO 185
23211       130 CONTINUE
```

Line Number	Explanation
23143-23148	Check for illegal combination of phrase types: A combination of a constant phrase and an arithmetic phrase is illegal. Also two arithmetic phrases are illegal.

Line Number	Explanation
23149-23163	Initialize iteration control variables. These variables will be used to expand each flow label into a series of constant flows.
23164-23170	Branch to appropriate phrase type for each phrase as determined previously. (KTYPE=0 for a constant phrase, KTYPE=1 for a DØ phrase, and KTYPE=2 for an arithmetic phrase.)
23171-23179	The phrase is assumed a constant. NN(I,1)=NN(I,2) will contain the integer value of the first (and only) field of the Ith phrase. EXAMPLE. (100-I=2,4). Implies IFLD(1,1)=100, then NN(1,1)=100=NN(1,2) and NN(1,3)=1 (initialized value).
23180-23192	The phrase is assumed a DØ expression. Variable NN(I,J) is filled and contains the J+1 field of phrase I. Examining the DØ phrase in the above example we see IFLD(2,1)=I, IFLD(2,2)=2, and IFLD(2,3)=4. Therefore NN(2,1)=2, NN(2,2)=4, NN(2,3)=1(initial value).
23193-23211	The phrase is assumed an arithmetic expression. KK(I,J) is filled containing the J/2 field of phrase I. EXAMPLE. (I=1,11,5-J=3*I+0). Preceding portions of this section would cause IFLD(2,1)=J, IFLD(2,2)=3, IFLD(3,2)=I, IFLD(4,2)=0, then KK(2,1)=3 and KK(2,2)=0. S(I)=.TRUE. indicates that the Ith phrase is an arithmetic expression.

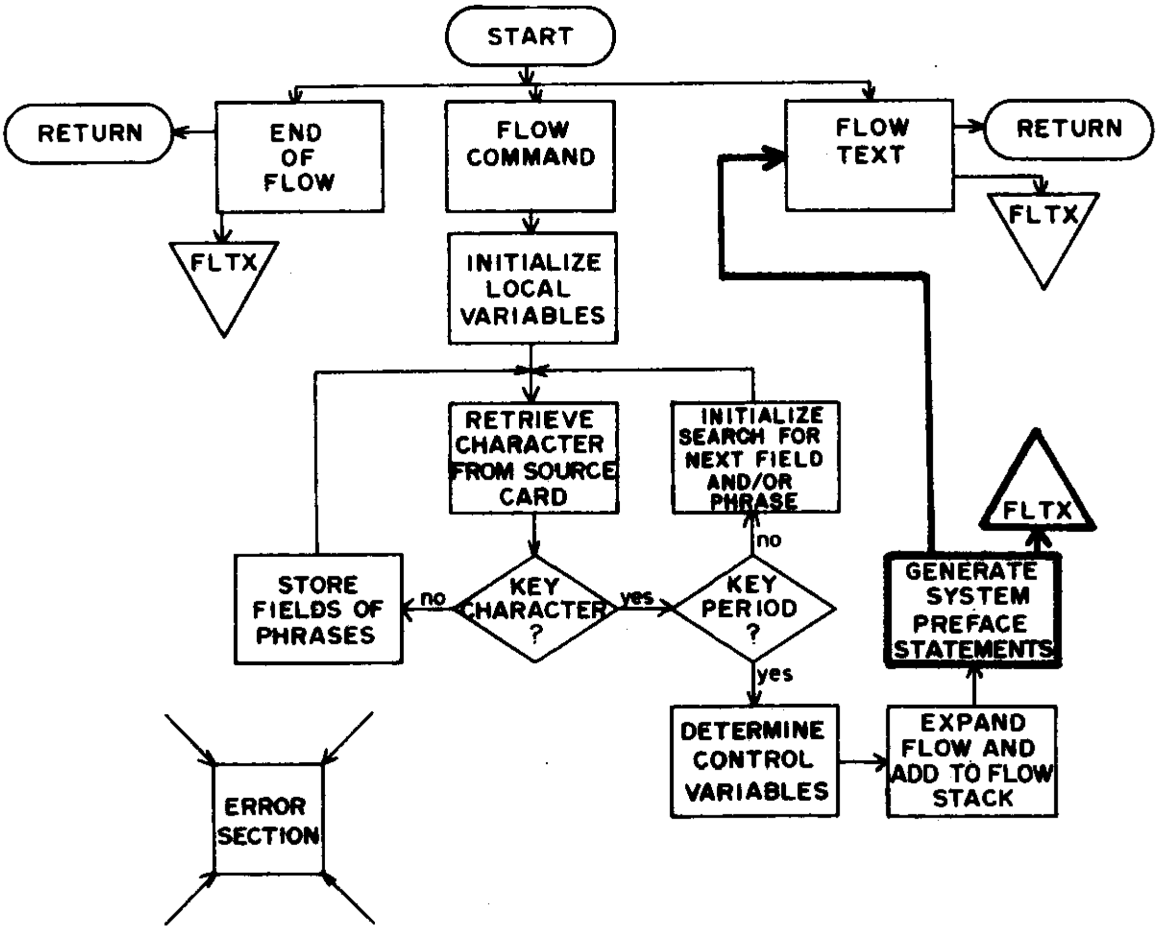
Expand flow and add to stack


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
no
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

```
      N1=NN(1,1)
      N2=NN(1,2)
      N3=NN(1,3)
      M1=NN(2,1)
      M2=NN(2,2)
      M3=NN(2,3)
      K1=KK(1,1)
      K2=KK(1,2)
      L1=KK(2,1)
      L2=KK(2,2)
      SWI=S(1)
      SWJ=S(2)
C
C.....ITERATE THROUGH ALL FLOWS DECLARED CHECKING THE RANGE OF FLOW
C.....INDICES AND CREATING A COMMENT RECORD ON THE TEMPORARY FLOW STORA
C.....FILE OF EACH EXISTING FLOW.
C
      DO 135 II=N1,N2,N3
         I=II
      DO 135 JJ=M1,M2,M3
         J=JJ
         IF (SWI) I=K1*JJ+K2
         IF (SWJ) J=L1*II+L2
         IF (I.LT.1.O.I.GT.999) GO TO 190
         IF (J.LT.1.O.J.GT.999) GO TO 190
         NFLW=NFLW+1
         IF (NFLW.GT.NFMAX) GO TO 195
         I2P15=SHIFT(I,15)
  135 NFLT(NFLW)=I2P15.O.J
```

Line Number	Explanation
23212-23223	The iteration control variables are assigned to singly dimensioned variables for use as DØ loop indices.
23225-23240	The flow label (command) is expanded into its equivalent series of constant flows and each expanded flow command is stored in the flow stack, NFLT. EXAMPLE. The flow command (1-2). is stored as NFLT(NFLW) = 100002_8 EXAMPLE. (I=1,11,5-J=3*I+0). is expanded and stored in NFLT as NFLT(NFLW) = (1-3) =100003_8 NFLT(NFLW+1) = (6-18)=600022_8 NFLT(NFLW+2) =(11-33)=1300041_8 NFLW represents the total number of expanded flows encountered from the source deck. NFMAX is the maximum number of flows that the system can handle. It is based on the amount of available core.

Generate systems preface statements


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
no
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
ERROR SECTION
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK

```
C
C.....GENERATE SYSTEMS SUPPLIED PREFACE STATEMENTS.
C
      IF (KTYPE(1).EQ.0.A.KTYPE(2).EQ.0) GO TO 145
      NRFL=NRFL+1
      MLAB=I1ZR(NRFL)
      DO 140 I=1,2
         IF (KTYPE(I).EQ.0.O.KTYPE(I).EQ.2) GO TO 140
         WRITE (U7,210) MLAB,(IPHZ(J,I),J=1,4)
         K=3-I
         IF (KTYPE(K).NE.2) GO TO 140
         WRITE (U7,215) (IPHZ(J,K),J=1,4)
  140 CONTINUE
  145 WRITE (U7,220)
      RETURN

      FUNCTION I1ZR(I)
      DATA ZEROS/33333333333333333333B/
      ENCODE (10,25,NUM) I
      ND=ALOG10(FLOAT(I))+1
      MASK=0
      IF (ND.LE.0) GO TO 20
      DO 15 J=1,ND
   15 MASK=MASK+63*2**(6*J-6)
```

```
24008          20 I1ZR=(NUM.A.MASK).O.(.N.MASK.A.ZEROS)
24009             RETURN
24010       C
24011          25 FORMAT (I10)
24012       C
24013             END
```

Line Number | Explanation

23241-23246

NRFL represents the total number of flow labels (not expanded) containing DØ expressions or arithmetic phrases.

MLAB is a zero filled right justified BCD representation of NRFL. If NRFL=25, MLAB=10H0000000025. MLAB is used as a statement label in the generation of FØRTRAN executable DØ loops for each DØ phrase encountered from the source deck.

23247-23255

The flow command is written unto the flow definition text file, U7. There are two cases:

EXAMPLE 1. The flow label consists of only constant phrases. (1-2). would cause the following card to be written on U7.

```
FLØW=17770 00000 00000 00000B.OR.(XMFL+1)
```

EXAMPLE 2. The flow label consists of a DØ phrase. (I=1,11,5-J=3*I+0). Assume NRFL=30. File U7 would contain

```
DØ 90030 I=1,11,5
J=3*I+0
FLØW=17770 00000 00000 00000B.OR.(XMFL+1)
```

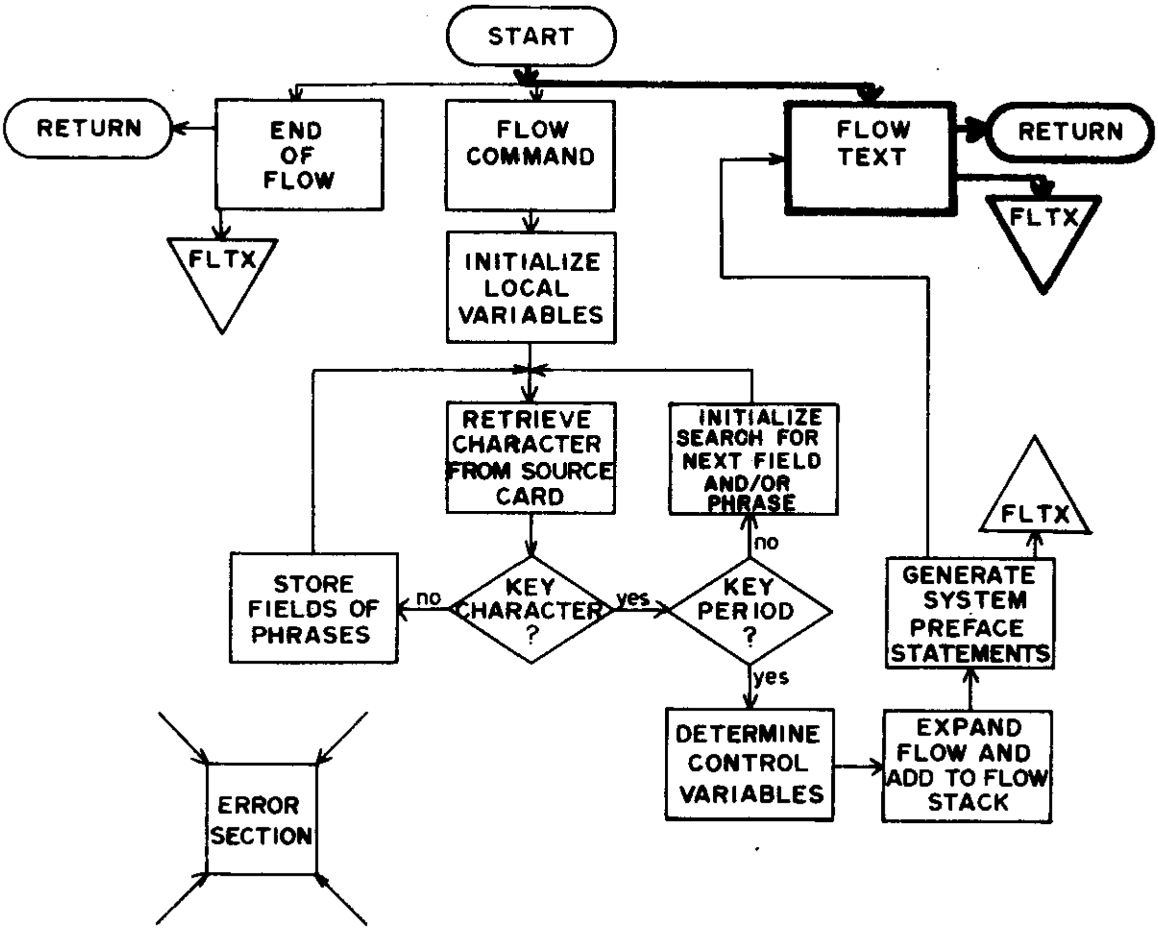
NRFL is converted to MLAB which is used to generate a unique statement label starting with a 9. For NRFL=30, 90030 is generated for a label number. In this way flow commands are converted to their equivalent FØRTRAN executable statements and are written on U7.

24000-24013

FUNCTIØN I1ZR converts an integer to its equivalent BCD representation, right justified with BCD zero fill to the left.

Line Number	Explanation
	EXAMPLE. Assume that the input to I1ZR was the integer 30. ND =2=number of digits in the integer. MASK=63+63*2**6 =•••0007777$_8$ NUM.A.MASK=•••55553633$_8$ .A. •••7777$_8$ =•••00003633$_8$ .N.MASK.A.ZEROS=•••7770000$_8$ .A. ••••••3333$_8$ =•••3330000$_8$ I1ZR=•••0003633$_8$ .0. •••3330000$_8$ =•••3333633$_8$ =10H0000000030

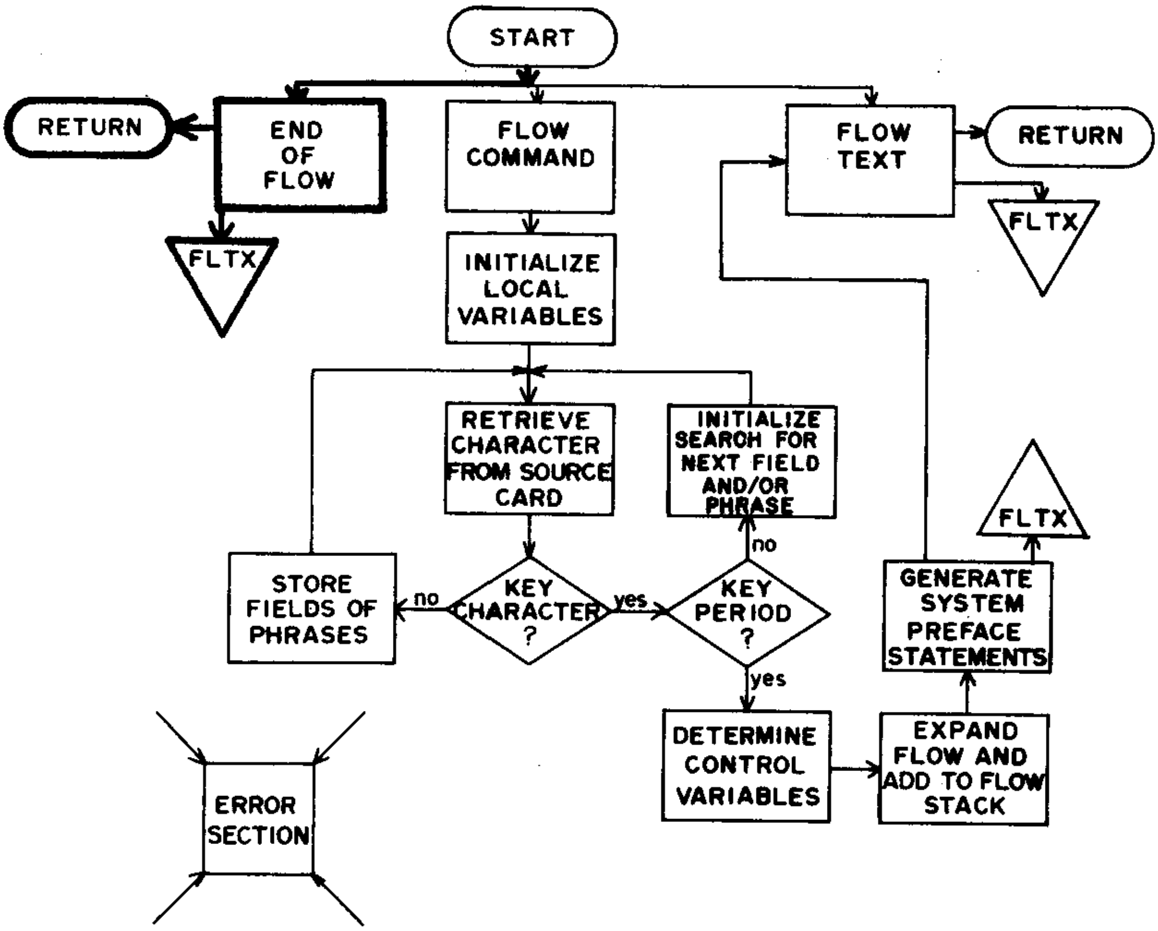
Process flow text


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
no
KEY PERIOD ?
GENERATE SYSTEM PREFACE STATEMENTS
yes
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

```
C
C.....<FLOW>...
C
   40 CALL FLIDF (COMAND,FATAL)
      WRITE (U7,90) TEXT
      GO TO 15
C
C.....<FLOW TEXT>...
C
   45 WRITE (U7,90) CARD
      GO TO 15
```

Line Number	Explanation
10126-10127	The remainder of the source card following the flow command is written on U7.
10128-10132	Any cards following a flow label (command) up to the end of flow (as determined by KTYPE is subroutine CARDTP) are considered a part of the preceding flow commands text and are written onto U7.

Generate flow termination text

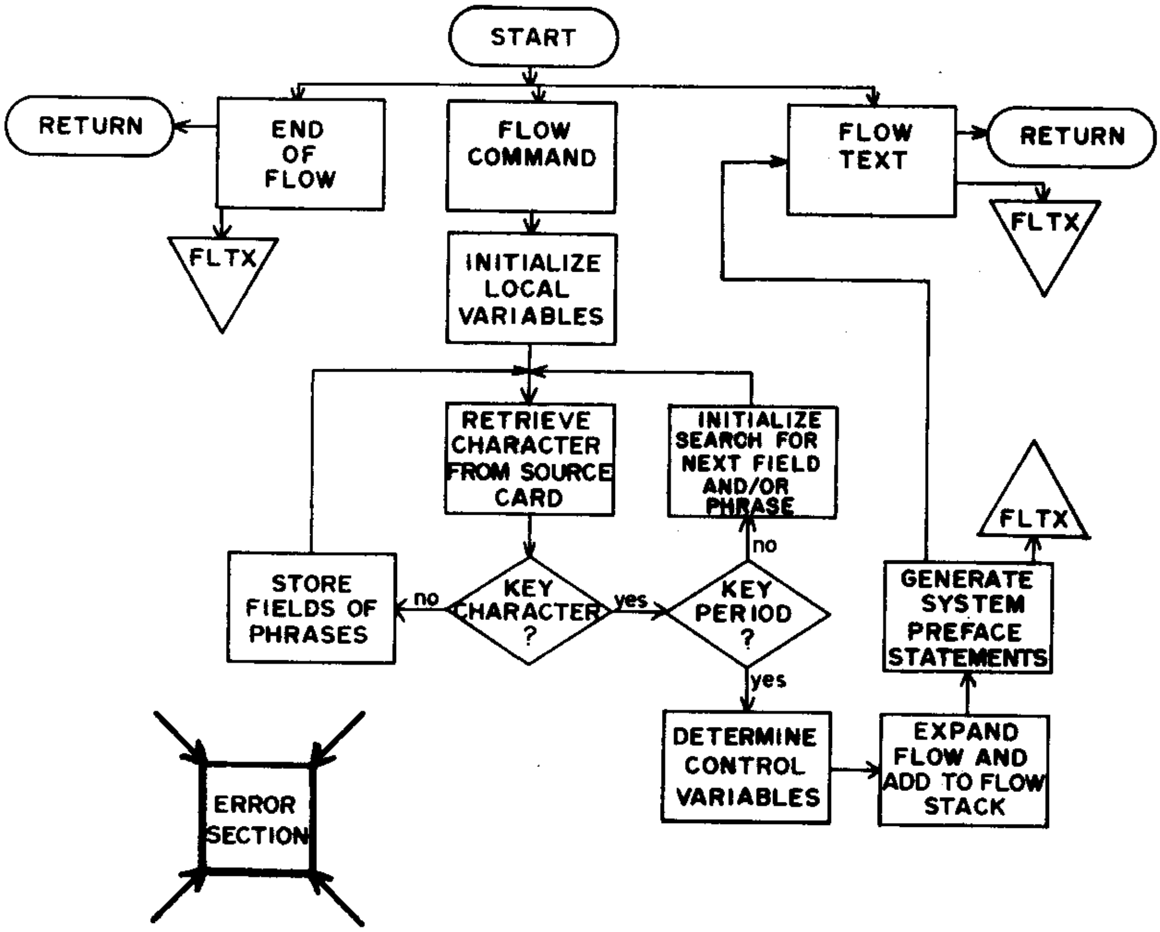

START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
KEY PERIOD ?
no
yes
GENERATE SYSTEM PREFACE STATEMENTS
ERROR SECTION
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK

```
C
C.....DETERMINE CARD TYPE.
C
      CALL CARDTP (CARD,KTYPE,JTYPE,TEXT,COMAND,FATAL), RETURNS(15)
      IF (KTYPE.NE.6.A.(JTYPE.EQ.5.O.JTYPE.EQ.6)) CALL FL2DF (CARD,FATAL
     1)
```

```
C
C
      ENTRY FL2DF
C
C.....THIS ROUTINE GENERATES THE SYSTEM TERMINATION TEXT FOR A FLOW.
C
      WRITE (U7,225)
      IF (KTYPE(1).EQ.0.A.KTYPE(2).EQ.0) RETURN
      WRITE (U7,230) MLAB
      RETURN
```

Line Number	Explanation
10105	If the previous card type was flow command or flow text and the present card is not flow text, then an end of flow must be written onto U7.
23293-23300	The following cards are written onto U7 terminating the flow XMFL=XMFL+1 XFLW(XMFL)=FLØW
23301	If the flow consists of a DØ phrase, then a continue statement is written unto U7 terminating the DØ statement.

Process errors


START
RETURN
END OF FLOW
FLOW COMMAND
FLOW TEXT
RETURN
FLTX
FLTX
INITIALIZE LOCAL VARIABLES
RETRIEVE CHARACTER FROM SOURCE CARD
INITIALIZE SEARCH FOR NEXT FIELD AND/OR PHRASE
FLTX
STORE FIELDS OF PHRASES
no
KEY CHARACTER ?
yes
no
KEY PERIOD ?
yes
GENERATE SYSTEM PREFACE STATEMENTS
DETERMINE CONTROL VARIABLES
EXPAND FLOW AND ADD TO FLOW STACK
ERROR SECTION

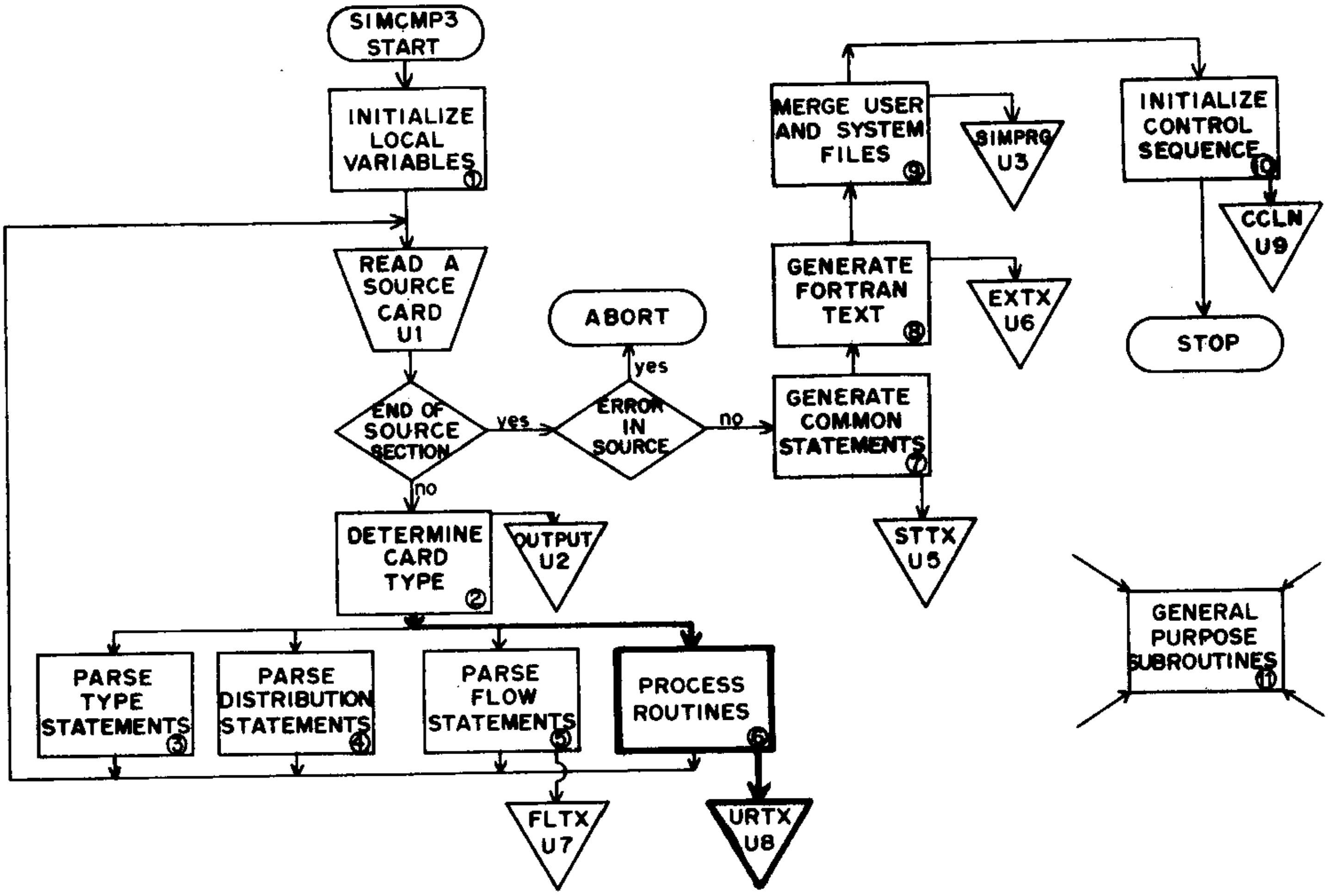
```
C
C.....IF ERRORS ENCOUNTERED ISSUE A DIAGNOSTIC.
C
  150 WRITE (U2,235)
      FATAL=.TRUE.
      RETURN
  155 WRITE (U2,240) ICHR,ICOL
      FATAL=.TRUE.
      RETURN
  160 WRITE (U2,245) (IPHZ(I,J),I=1,4)
      FATAL=.TRUE.
      RETURN
  165 WRITE (U2,250) IFLD(J,K)
      FATAL=.TRUE.
      RETURN
  170 WRITE (U2,255)
      FATAL=.TRUE.
      RETURN
  175 WRITE (U2,260) IFLD(I,1)
      FATAL=.TRUE.
      RETURN
  180 WRITE (U2,265)
      FATAL=.TRUE.
      RETURN
  185 WRITE (U2,270)
```

```
23281              FATAL=.TRUE.
23282              RETURN
23283          190 WRITE (U2,275) I,J
23284              FATAL=.TRUE.
23285              RETURN
23286          195 IF (NFMAX.LT.9999) GO TO 200
23287              WRITE (U2,280)
23288              FATAL=.TRUE.
23289              RETURN
23290          200 WRITE (U2,285) NFMAX
23291              FATAL=.TRUE.
23292              RETURN
```

```
23303        C
23304          205 FORMAT (2H(I,I2,6H)      )
23305          210 FORMAT (6X,  4HDO 9,R4,1X,4A10,25X)
23306          215 FORMAT (6X,4A10,34X)
23307          220 FORMAT (6X, 41HFLOW=17770 00000 00000 000008.OR.(XMFL+1),33X)
23308          225 FORMAT (6X, 11HXMFL=XMFL+1,63X/6X, 15HXFLW(XMFL)=FLOW,59X)
23309          230 FORMAT (  1H9,R4,  9H CONTINUE,66X)
23310          235 FORMAT (11H *****FE    , 39HFLOW DIRECTIVE UNTERMINATED AT CARD END
23311             1)
23312          240 FORMAT (11H *****FE    , 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLU
23313             1MN ,I2)
23314          245 FORMAT (11H *****FE    , 13HFLOW PHRASE ",4A10, 42H..." CONTAINS MO
23315             1RE THAN 40 NON-BLANK CHARS)
23316          250 FORMAT (11H *****FE    , 27HFLOW EXPRESSION SUB-FIELD ",A10, 42H...
23317             1" CONTAINS MORE THAN 10 NON-BLANK CHARS)
23318          255 FORMAT (11H *****FE    , 65HARITHMETIC PHRASE MUST BE USED IN CONJU
23319             1NCTION WITH A DO... PHRASE)
23320          260 FORMAT (11H *****FE    , 40HFLOW ITERATION PHRASE CONTROL VARIABLE
23321             1',A5, 43H' MUST BE A 5 CHAR OR LESS INTEGER VARIABLE)
23322          265 FORMAT (11H *****FE    , 78HTHE DO... PHRASE CONTROL VARIABLE MUST
23323             1BE THE OPERAND IN THE ARITHMETIC PHRASE)
23324          270 FORMAT (11H *****FE    , 67HA FIELD IN WHICH A CONSTANT SHOULD APPE
23325             1AR IS MISSING OR IS NEGATIVE)
23326          275 FORMAT (11H *****FE    , 14HFLOW INDICES (,I4,  1H-,I4, 59H) PRODUC
23327             1ED BY THE ABOVE LABEL ARE OUTSIDE THE RANGE 1 - 999)
23328          280 FORMAT (11H *****FE    , 28HNUMBER OF FLOWS EXCEEDS 9999)
23329          285 FORMAT (11H *****FE    , 57HINSUFFICIENT FIELD LENGTH, INCREASE BY
23330             1(NO. OF FLOWS  -  ,I4,  1H))
23331        C
23332              END
```

Line Number	Explanation
23256-23292, 23303-23332	This is the error (and format) section. Control arrives here from any of the various subsections upon encountering a syntax error in the parsing of the flow command. FATAL is set to .TRUE. and the program terminates after the entire source program is processed.

1.6. *Process Routines*



Overview

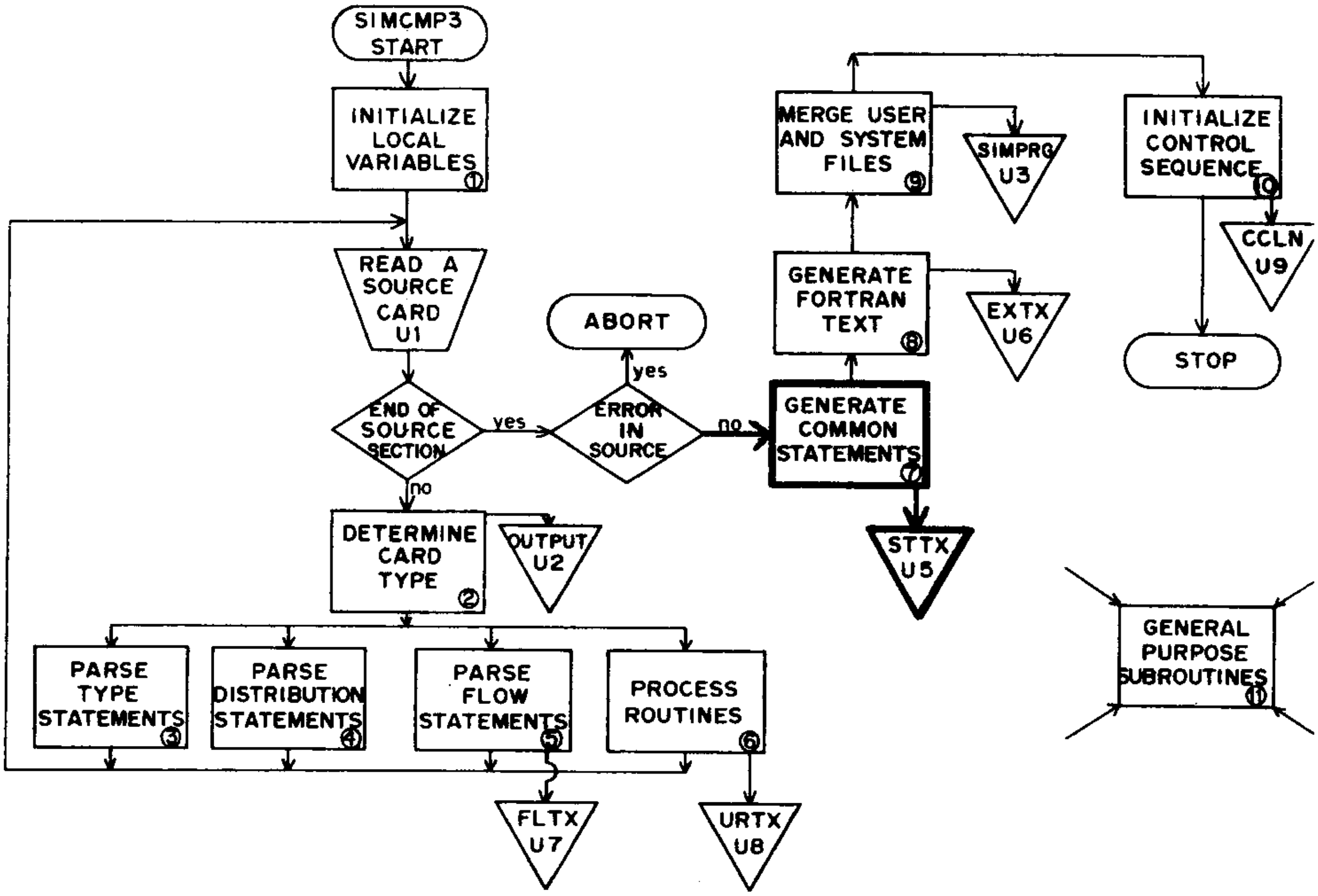
All SUBRØUTINES, FUNCTIØNS, and EVENTS are written onto a user generated external routine file, URTX (U8). Following each routine command, an end of file is written, and a flag card is inserted at the beginning of the next file. All cards belonging to the routine command are written onto U8 following the flag card. (The flag card indicates the position where CØMMØN cards will later be inserted into the external file.)

```
EXAMPLE.  Source                   File U8

          SUBRØUTINE ANT                SUBRØUTINE ANT

          RETURN                   <end file>

          END                      C++++     5     0

          FUNCTIØN BEE                  RETURN

          RETURN                        END

          END                           FUNCTIØN BEE

                                   <end file>

                                   C++++     5     0

                                        RETURN

                                        END
```

```
      C
      C.....<SUBROUTINE>, <FUNCTION>, OR <EVENT>...
      C
         50 WRITE (U8,90) TEXT
            GO TO 15
      C
      C.....<CONTUATION OF KTYPE=7 OR 8>...
      C
         55 WRITE (U8,90) CARD
            GO TO 15
      C
      C.....<ROUTINE TEXT>...
      C
         60 IF (JTYPE.NE.7.A.JTYPE.NE.8) GO TO 55
            END FILE U8
            WRITE (U8,95)
            GO TO 55


         90 FORMAT (8A10)
         95 FORMAT (15HC++++     5     0,65X)
```

Line Number	Explanation
10133-10142	A routine command (SUBRØUTINE, FUNCTIØN, or EVENT) or a continuation of a routine command is written onto U8 (KTYPE=7 or 8).
10143-10149	Routine text (KTYPE=9). If the previous card was a routine command or a continuation of a command, then (i) write an end file on U8, (ii) insert a flag card, and (iii) write the source card onto U8.

1.7. *Generate Common Statements*



Overview

All cards of the source program have been processed by SIMCØMP. During the processing, each SIMCØMP type statement (a STØRAGE., REAL., or INTEGER. card) encountered caused the name, subscripts, and type of each variable on the card to be filled into stacks LVR1 and LVR2 (see Section 1.3). This section uses these stacks as input and generates as output a user variable declarations file, STTX (U5). The file contains FØRTRAN executable CØMMØN, REAL, and INTEGER card images. The name and subscript of each variable in LVR1 and LVR2 are written onto a CØMMØN card image. If the mode of a variable disagrees with that implicitly assigned variable name by the FØRTRAN compiler, then that name is written onto either an INTEGER or REAL card image. Thus this section generates the FØRTRAN executable equivalent of SIMCØMP type statements.

```
      SUBROUTINE GCOMMON
      COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION NUM(3), IPAC(2), CCARD(8), RCARD(8), ICARD(8), IK(3)
      INTEGER CCARD,RCARD
      DATA IK/1H(,1H,,1H,/
C
C.....GCOMMON PROCESSES COMMON, INTEGER, AND REAL STATEMENTS FROM THE
C         VARIABLE STACK.
C
      ICOM=-1
      IREAL=-1
      INT=-1
      IF (NVAR.LE.9) GO TO 105
      DO 100 II=10,NVAR
         CALL GBYTE (LVR1(II),MODE,48,2)
         IP=0
         LNAM=0
         DO 15 M=1,5
            CALL GCHARS (LVR1(II),M,1,ICHR)
            IF (ICHR.EQ.1H ) GO TO 15
            IP=IP+1
      IF (IP.EQ.1) MDCHR=ICHR
            CALL SCHARS (IPAC,IP,1,ICHR)
   15    CONTINUE
         LNAM=IP
         DO 20 I=1,3
            CALL GBYTE (LVR2(II),NUM(I),10*I-10,10)
            IF (NUM(I).LE.0) GO TO 25
            ENCODE (10,110,INN) NUM(I)
            ND=ALOG10(FLOAT(NUM(I)))+1
            IP=IP+1
            CALL SCHARS (IPAC,IP,1,IK(I))
            IP=IP+1
            CALL GCHARS (INN,11-ND,ND,IOUT)
            CALL SCHARS (IPAC,IP,ND,IOUT)
            IP=IP+ND-1
   20    CONTINUE
   25    IF (I.EQ.1) GO TO 30
         IP=IP+1
         CALL SCHARS (IPAC,IP,1,1H))
C
C.....GENERATE <COMMON> CARD
C
   30    ICONT=ICOM+IP
         IF (ICOM.LE.0) GO TO 35
         IF (ICONT.LT.72) GO TO 45
         WRITE (U5,115) CCARD
   35    DO 40 I=3,8
   40    CCARD(I)=10H
         CCARD(1)=10H      COMM
         CCARD(2)=10HON
         ICOM=13
         GO TO 50
   45    CALL SCHARS (CCARD,ICOM,1,1H,)
   50    CALL SCHARS (CCARD,ICOM+1,IP,IPAC)
         ICOM=IP+ICOM+1
      MODV=0
      IF (MDCHR.LT.1HI.O.MDCHR.GT.1HN) MODV=1
```

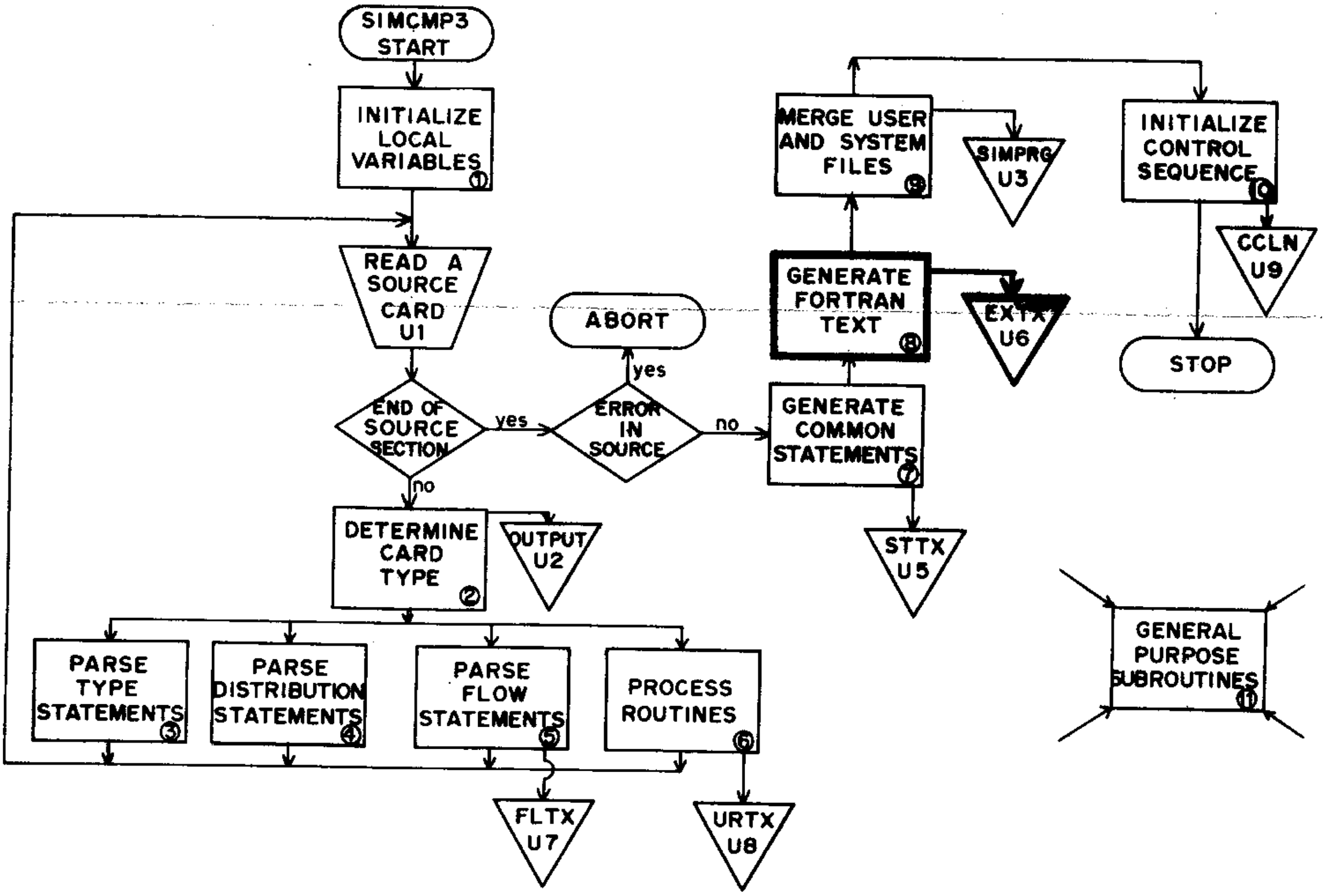
```
26060                IF (MODE.EQ.MODV) GO TO 100
26061                   IF (MODE.EQ.0) GO TO 75
26062          C
26063          C.....GENERATE TYPE <REAL> CARD
26064          C
26065                   IF (IREAL.LE.0) GO TO 55
26066                   ICONT=IREAL+LNAM
26067                   IF (ICONT.LT.72) GO TO 65
26068                   WRITE (U5,115) RCARD
26069             55    DO 60 I=2,8
26070             60    RCARD(I)=10H
26071                   RCARD(1)=10H      REAL
26072                   IREAL=11
26073                   GO TO 70
26074             65    CALL SCHARS (RCARD,IREAL,1,1H,)
26075             70    CALL SCHARS (RCARD,IREAL+1,LNAM,IPAC)
26076                   IREAL=LNAM+IREAL+1
26077                   GO TO 100
26078          C
26079          C.....GENERATE TYPE <INTEGER> CARD
26080          C
26081             75    ICONT=INT+LNAM
26082                   IF (INT.LE.0) GO TO 80
26083                   IF (ICONT.LT.72) GO TO 90
26084                   WRITE (U5,115) ICARD
26085             80    DO 85 I=3,8
26086             85    ICARD(I)=10H
26087                   ICARD(1)=10H      INTE
26088                   ICARD(2)=10HGER
26089                   INT=14
26090                   GO TO 95
26091             90    CALL SCHARS (ICARD,INT,1,1H,)
26092             95    CALL SCHARS (ICARD,INT+1,LNAM,IPAC)
26093                   INT=INT+LNAM+1
26094            100 CONTINUE
26095                IF (ICOM.GT.0) WRITE (U5,115) CCARD
26096                IF (IREAL.GT.0) WRITE (U5,115) RCARD
26097                IF (INT.GT.0) WRITE (U5,115) ICARD
26098            105 WRITE (U5,120)
26099                RETURN
26100          C
26101            110 FORMAT (I10)
26102            115 FORMAT (8A10)
26103            120 FORMAT (6X, 16HCOMMON XEVSTK(1),58X)
26104          C
26105                END
```

Line Number	Explanation
26014	NVAR is the total number of variables in LVR1. If NVAR=9, then the only variables in LVR1 are the original system variables. Therefore no FØRTRAN declaration statements need be processed.
26015-26016	One variable in LVR1 is processed with each pass until all have been processed. MØDE=1 If the variable is of type real. =0 If the variable is of type integer.
26017-26026	The variable name is taken out of LVR1 and placed left justified in IPAC. LNAM is the number of characters in the variable name (must be five or less). IP is the number of characters in IPAC.

Line Number	Explanation
26027-26041	Each subscript is retrieved from LVR2, converted in BCD format, and filled into IPAC following the variable name. The delimiters of subscripts (a left paren delimits the name and the first subscript, a comma delimits the first and second subscripts, and a right paren delimits the last subscript) are filled into IPAC in the proper position. NUM(I) is the integer value of the Ith subscript. ND is the number of characters in the subscript. IK contains subscript delimiters. INN is the right justified BCD representation of subscript. IOUT is the left justified BCD representation of subscript. EXAMPLE. The variable FØX(15,2) is stored in LVR1 and LVR2 as: LVR1(J)=0617305555001702000B LVR2(J)=00000 01111 00000 00010 00000 00000 ••• From LVR1 MØDE=1 (type real). IPAC=011730=3HFØX,IP=3,LNAM=3. From LVR2 NUM(1)=00000 01111_2=15 IØUT=2H15 NUM(2)=00000 00010_2= 2 IØUT=1H2 NUM(3)=00000 00000 = 0 IPAC is filled with delimiters and nonzero subscripts When completely filled, IPAC=10HFØX(15,2) and IP=9. Therefore the original variable is constructed in IPAC from LVR1 and LVR2.
26042-26057	The variable name and its subscripts (IPAC) are filed into CCARD which contains a CØMMØN declaration. CCARD is formatted as a FØRTRAN CØMMØN statement (CØMMØN <var1>,<var2>, etc., to column 72). ICØM is the next column number of CCARD available to place <var1>. ICØM less than zero is the initial value and signals that CCARD must be initialized. The word CØMMØN is filled onto CCARD and ICØM is set to next column (13). ICØNT estimates the end column if IPAC is filled into CCARD. If ICØNT is greater than 72, the contents of CCARD are written onto U5, CCARD is reinitialized, and IPAC is filed onto the new CØMMØN statement.

Line Number	Explanation
26058-26061	Determine whether the mode of the variable conflicts with its FØRTRAN assigned type. MDCHR contains the leading character of the variable currently being processed (line 26023). MØDV is the FØRTRAN assigned type of variable (MØDV=0 if type is integer, else MØDV=1). If the FØRTRAN assigned type (MØDV) is the same as the SIMCØMP assigned type (MØDE), no type declaration need be generated. Otherwise fill variable name into a FØRTRAN REAL or INTEGER type statement.
26062-26077	The name of the variable is placed in a REAL statement, if it is of type real (MØDE=1). IREAL is the position counter of RCARD; if less than zero, initialize RCARD. ICØNT is the end column estimate. If greater than 72, RCARD is written onto U5 and reinitialized, and the variable name in IPAC (of length LNAM) is filled into RCARD beginning in the next available column (IREAL=11).
26078-26094	If the variable is of type integer, it is filled into ICARD, containing a FØRTRAN INTEGER statement.
26094-26099	All variables in the variable stack have been filled into FØRTRAN declaration statements. Flush the remaining declarations onto U5.

1.8. *Generate FØRTRAN Text*



Overview

This section takes the stacks built by the SIMCØMP compiler, converts them into FØRTRAN text, and writes this text onto the user generated FØRTRAN text file, EXTX (U6). The structure of each file on U6 will be explained in terms of the following source deck example.

```
STØRAGE. MØØSE
(1-I=2,4)

      ⋮
      FLØW=0.
      EVENT CALC
           .
           .
           .
      RETURN
      END
      EVENT TØTL
           .
           .
           .
      RETURN
      END
```

SIMCØMP processes this source deck and generates the below values for SIMCØMP variables.

(1) Parsing the STØRAGE. statement (Section 1.3)

```
NVAR=10
LVR1(10)=15171723050017600000B
LVR2(10)=0
NSTØR=1009
```

(2) Parsing the EVENT routines (Section 1.2)

```
NSUB=9
NSBL(1)=5HXCSIM
   .
   .
   .
NSBL(8)=5H CALC
NSBL(9)=5N TØTL
```

(3) Parsing the flow command (Section 1.5)

NFLW=3
NFLT(1)=100002_8
NFLT(2)=100003_8
NFLT(3)=100004_8

EXTX, File 1

The first file contains FØRTRAN labeled common statements, reserving memory locations for the number of variables, flows, and routines declared by the user. Referring to the above source deck, one variable (NVAR=10, the first nine are system variables), three flows, and nine event routines were declared. File 1 would contain:

```
CØMMØN/XXVR1FR/XNV,XNW,XVT1(10)
CØMMØN/XXVR2FR/XVT2(10)
```

```
      CØMMØN/XXFL1WS/XNFLW,XFLWT(3)
      CØMMØN/XXFL2WS/XFLW(3)
      CØMMØN/XXEXTRN/XNEX,XEXT(9)
```

EXTX, File 2

The second file contains FØRTRAN EXTERNAL statements for each defined event name. Recall that there are seven system events in addition to any user defined events.

```
C          0
      EXTERNAL XCSIM
           .
           .
           .
      EXTERNAL CALC
      EXTERNAL TØTL
```

EXTX, File 3

This file contains generated DATA assignment statements which save the values of SIMCØMP variables. An equivalent set of system variables (starting with the character X) are generated for use during execution. The names of the system variables are defined in File 1 and their values are defined by data statements in this file:

```
C       0    0
      DATA XNV/10/,XNW/1008/
      DATA XVT1(1)/30010422230000012000B
      DATA XVT2(1)/00040000000000000000B
               .
               .
               .
      DATA XVT1(10)/15171723050017600000B
      DATA XVT2(10)/00000000000000000000B
      DATA XNEX/9/
      DATA XEXT(1)/10HXCSIM     /
               .
               .
               .
      DATA XEXT(9)/10HTØTL      /
      DATA XNFLW/3/
      DATA XFLWT(1)/00000000000000/00002B/
      DATA XFLWT(2)/00000000000000/00003B/
      DATA XFLWT(3)/00000000000000/00004B/
```

For each SIMCØMP variable, an equivalent system variable is generated possessing the values of the original SIMCØMP variable. (NOTE. The equivalent of NFLT is XFLWT, etc.)

EXTX, File 4

The fourth file contains code which computes the entry addresses to the event routines and stores these in the system routine table, XEXT.

```
C         0    0
     XN=XN+1
     XI=LØCF(XCSIM)
     XEXT(XN)=XEXT(XN).O.XI
        .
        .
        .
     XN=XN+1
     XI=LØCF(TØTL)
     XEXT(XN)=XEXT(XN).O.XI
```

Generate FØRTRAN Text Section Code

```
      SUBROUTINE TX1OF
      COMMON /ROUTINS/ NSUB,NSBL(100),SUBFLG(4),KDIST
      COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
      COMMON NFLW,NRFL,NFMAX,NFLT(1)
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      DIMENSION SNAME(4), IFLD(9)
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL SUBFLG
      DATA SNAME/5HSTART,5HFINIS,5HCYCL1,5HCYCL2/
C
C.....THIS ROUTINE GENERATES CODING ASSOCIATED WITH SYSTEM VARIABLE
C.....STORAGE AND SYSTEM VARIABLE VALUE ASSIGNMENT.
C
C.....GENERATE DUMMY ROUTINES "START", "FINIS", "CYCL1", OR "CYCL2"
C.....THEY WERE NOT GENERATED BY THE USER.
C
      DO 15 I=1,4
         IF (SUBFLG(I)) WRITE (U8,50) SNAME(I)
   15 CONTINUE
      END FILE U8
      WRITE (U8,70)
      END FILE U8
C
C.....GENERATE SYSTEM VARIABLE STORAGE DECLARATION STATEMENTS.
C
      WRITE (U6,55) NVAR,NVAR
      MFLW=NFLW
      IF (MFLW.LE.0) MFLW=1
      WRITE (U6,60) MFLW,MFLW
      WRITE (U6,65) NSUB
      END FILE U6
      WRITE (U6,70)
C
C.....GENERATE FTN EXTERNAL DECLARATIONS FOR EACH OF THE EXTERNAL EVENT
C.....ROUTINES.
C
      DO 20 I=1,NSUB
   20 WRITE (U6,75) NSBL(I)
      END FILE U6
      WRITE (U6,70)
C
C.....GENERATE THE DATA ASSIGNMENT STATEMENTS WHICH SET THE VALUES FOR
C.....THE SYSTEM VARIABLES.
C
      NST=NSTOR-1
      WRITE (U6,80) NVAR,NST
      CALL SORT
      DO 25 I=1,NVAR
   25 WRITE (U6,85) I,LVR1(I),I,LVR2(I)
      WRITE (U6,90) NSUB
      DO 30 I=1,NSUB
   30 WRITE (U6,95) I,NSBL(I)
      WRITE (U6,100) NFLW
      IF (NFLW.LE.0) GO TO 40
      DO 35 I=1,NFLW
   35 WRITE (U6,105) I,NFLT(I)
   40 END FILE U6
      WRITE (U6,70)
C
C.....GENERATE CODE WHICH COMPUTES THE ENTRY ADDRESSES TO THE EVENT
C.....ROUTINES AND STORES THESE IN THE EVENT ROUTINE TABLE.
C
      DO 45 I=1,NSUB
   45 WRITE (U6,110) NSBL(I)
      END FILE U6
      WRITE (U6,70)
      END FILE U6
      END FILE U5
      WRITE (U5,70)
      END FILE U5
      END FILE U7
      WRITE (U7,70)
      END FILE U7
      RETURN
C
   50 FORMAT (6X, 11HSUBROUTINE ,A5,58X/6X,  6HRETURN,68X/6X,  3HEND,71X
     1)
   55 FORMAT (6X, 28HCOMMON/XXVR1FR/XNV,XNW,XVT1(,I3,  1H),42X/9X, 20HCO
     1MMON/XXVR2FR/XVT2(,I3,  1H),47X)
   60 FORMAT (6X. 27HCOMMON/XXFL1WS/XNFLw,XFLWT(,I4,  1H),42X/9X, 20HCOM
     1MON/XXFL2WS/XFLW(,I4,  1H),46X)
   65 FORMAT (6X, 25HCOMMON/XXEXTRN/XNEX,XEXT(,I3,  1H),65X)
   70 FORMAT (15HC        0    0,65X)
   75 FORMAT (6X,  9HEXTERNAL ,A5,60X)
   80 FORMAT (6X,  9HDATA XNV/,I3,  7H/, XNW/,I10,  1H/,44X)
   85 FORMAT (6X, 10HDATA XVT1(,I3,  2H)/,O20,  2HB/,37X/12X, 10HDATA XV
     1T2(,I3,  2H)/,O20,  2HB/,31X)
   90 FORMAT (6X, 10HDATA XNEX/,I3,  1H/,60X)
   95 FORMAT (6X. 10HDATA XEXT(,I3,  2H)/,O20,  2HB/,37X)
  100 FORMAT (6X, 11HDATA XNFLW/,I4,  1H/,58X)
  105 FORMAT (6X, 11HDATA XFLWT(,I4,  2H)/,O20,  2HB/,35X)
  110 FORMAT (6X,  7HXN=XN+1,67X/6X,  8HXI=LOCF(,A5,  1H),60X/6X, 22HXEX
     1T(XN)=XEXT(XN).O.XI,52X)
C
      END
          IDENT SORT
          LIST  -R,-G
          ENTRY SORT
SORT      BSS   1
**********SORT ALPHABETIZES THE ARRAY CONTAINING VARIABLES DECLARED IN S
*         STORAGE STATEMENTS
          USE   /STORAGE/      .LABELED COMMON CONTAINING VARIABLES
NVAR      BSS   1              .NUMBER OF VARIABLES
LVR1      BSS   8              .SYSTEMS VARIABLES-1
K1        BSS   991            .USER VARIABLES (MAX 990)
LVR2      BSS   8              .SYSTEMS SUBSCRIPTS OF ABOVE VARIABLES
K2        BSS   991            .USER SUBSCRIPTS
NSTOR     BSS   1
          USE   *

ZERO      MACRO A
          MX1   36             .FORM MASK IN 36 HIGH ORDER BITS
          B_A   A*X1           .SAVE 30 HIGH BITS OF A
          MX1   6              .FORM 6 BIT MASK
          LX1   30             .SHIFT MASKING BITS AROUND
          BX6   X1*X2          .X6= ...005500...B DEPENDING ON X1
+         B_A   A-X6           .ZERO OUT THE BLANK IN A
          LX1   6              .LEFT SHIFT MASKING BITS
          BX3   A*X1           .X3=6 BITS OF A (ZERO FILL)
          BX6   X1*X2          .X6=SAME ORDER 6 BITS OF X2
          IX3   X3-X6          .COMPARE BITS OF A AND X2
+         ZR    X3,*-2         .IF A BITS = X2 BITS (55) LOOP BACK
          ENDM

          SB7   1              .B7=1
          SA1   NVAR           .X1=C(NVAR)
          SB2   X1-9           .B2=NUMBER OF USER DECLARED VARIABLES
          SX0   B2             .X0=NUMBER OF USER DECLARED VARIABLES
          SB1   B2
          SA2   =10H

LOOP1     AX0   1              .X0=X0/2
          SB2   X0             .B2=X0
          LE    B2,SORT        .B2 LE 0, RETURN  (SORT FINISHED)
          SB3   B1-B2
          SB4   1
```

```
28041        LOOP2    SB5    B4
28042        LOOP3    SB6    B2+B5
28043                 SA4    B5+K1          .X4=C(B5+K1) (A VARIABLE NAME)
28044                 ZERO   X4             .CONVERT BLANK FILL IN X4 TO ZERO FILL
28045                 SA5    B6+K1          .X5=C (B6+K1) (A VARIABLE NAME)
28046                 ZERO   X5             .CONVERT TO ZERO FILL
28047                 IX6    X5-X4          .IS X5 AFTER X4 ALPHABETICALLY
28048                 PL     X6,LOOP4       .YES, LOOP4    IF NO SWITCH THE TWO
28049        **********SWITCH VARIABLE NAMES
28050
28051                 SA4    B5+K1          .X4=C(B5+K1)
28052                 SA5    B6+K1          .X5=C (B6+K1)
28053                 BX6    X4             .X6=C(B5+K1)
28054                 BX7    X5             X7=C(B6+K1)
28055                 SA6    B6+K1          .C(B5+K1) IS STORED AT  B6+K1
28056                 SA7    B5+K1          .C(B6+K1) IS STORED AT B5+K1
28057
28058        **********SWITCH SUBSCRIPTS OF VARIABLE NAMES
28059                 SA4    B5+K2          .X4=C(B5+K2) A VARIABLE SUBSCRIPT
28060                 SA5    B6+K2          .X5=C(B6+K2)
28061                 BX6    X4
28062                 BX7    X5
28063                 SA6    B6+K2
28064                 SA7    B5+K2
28065                 SB5    B5-B2
28066                 GE     B5,B7,LOOP3    .B5 GE 1  THEN LOOP3
28067        LOOP4    SB4    B4+1
28068                 GT     B4,B3,LOOP1
28069                 EQ     LOOP2
28070                 END
```

A line-by-line explanation of the coding follows.

```
27000              SUBROUTINE TX1OF
27001              COMMON /ROUTINS/ NSUB,NSBL(100),SUBFLG(4),KDIST
27002              COMMON /STORAGE/ NVAR,LVR1(999),LVR2(999),NSTOR
27003              COMMON NFLW,NRFL,NFMAX,NFLT(1)
27004              COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
27005              DIMENSION SNAME(4), IFLD(9)
27006              INTEGER U1,U2,U3,U4,U5,U6,U7,U8
27007              LOGICAL SUBFLG
27008              DATA SNAME/5HSTART,5HFINIS,5HCYCL1,5HCYCL2/
27009        C
27010        C.....THIS ROUTINE GENERATES CODING ASSOCIATED WITH SYSTEM VARIABLE
27011        C.....STORAGE AND SYSTEM VARIABLE VALUE ASSIGNMENT.
27012        C
27013        C.....GENERATE DUMMY ROUTINES "START", "FINIS", "CYCL1", OR "CYCL2"
27014        C.....THEY WERE NOT GENERATED BY THE USER.
27015        C
27016              DO 15 I=1,4
27017                 IF (SUBFLG(I)) WRITE (U8,50) SNAME(I)
27018           15 CONTINUE
27019              END FILE U8
27020              WRITE (U8,70)
27021              END FILE U8


27075           50 FORMAT (6X, 11HSUBROUTINE ,A5,58X/6X,  6HRETURN,68X/6X,  3HEND,71X
27076             1)


27082           70 FORMAT (15HC         0     0,65X)
```

Line Number	Explanation
27016-27019	SUBFLAG(I)(set in Section 1.6) is .FALSE. if the user supplied the Ith routine name in SNAME. File U8 is the file containing all user supplied routines (subroutines, functions, and events). If routines named START, FINIS, CYCL1, and CYCL2 were not supplied by the user, dummy routines containing these names are written onto U8.
27020-27021	File U8 is completed. The second file begins with a flag card.

```
27022          C
27023          C.....GENERATE SYSTEM VARIABLE STORAGE DECLARATION STATEMENTS.
27024          C
27025                WRITE (U6,55) NVAR,NVAR
27026                MFLW=NFLW
27027                IF (MFLW.LE.0) MFLW=1
27028                WRITE (U6,60) MFLW,MFLW
27029                WRITE (U6,65) NSUB
27030                END FILE U6
27031                WRITE (U6,70)

27077             55 FORMAT (6X, 28HCOMMON/XXVR1FR/XNV,XNW,XVT1(,I3,  1H),42X/9X, 20HCO
27078               1MMON/XXVR2FR/XVT2(,I3,  1H),47X)
27079             60 FORMAT (6X, 27HCOMMON/XXFL1WS/XNFLW,XFLWT(,I4,  1H),42X/9X, 20HCOM
27080               1MON/XXFL2WS/XFLW(,I4,  1H),46X)
27081             65 FORMAT (6X, 25HCOMMON/XXEXTRN/XNEX,XEXT(,I3,  1H),65X)
```

Line Number	Explanation
27025	File U6 is the user general FTN. text file. A labeled common block is created reserving storage for NVAR (the number of variables declared by user on STØRAGE., REAL., or INTEGER. statements).
27026-27028	Labeled common blocks are created reserving storage for the total number of flows declared by the user.
27029-27031	The number of routines employed by the user has a common block reserved for it. These three common blocks are the contents of the first file of U6.

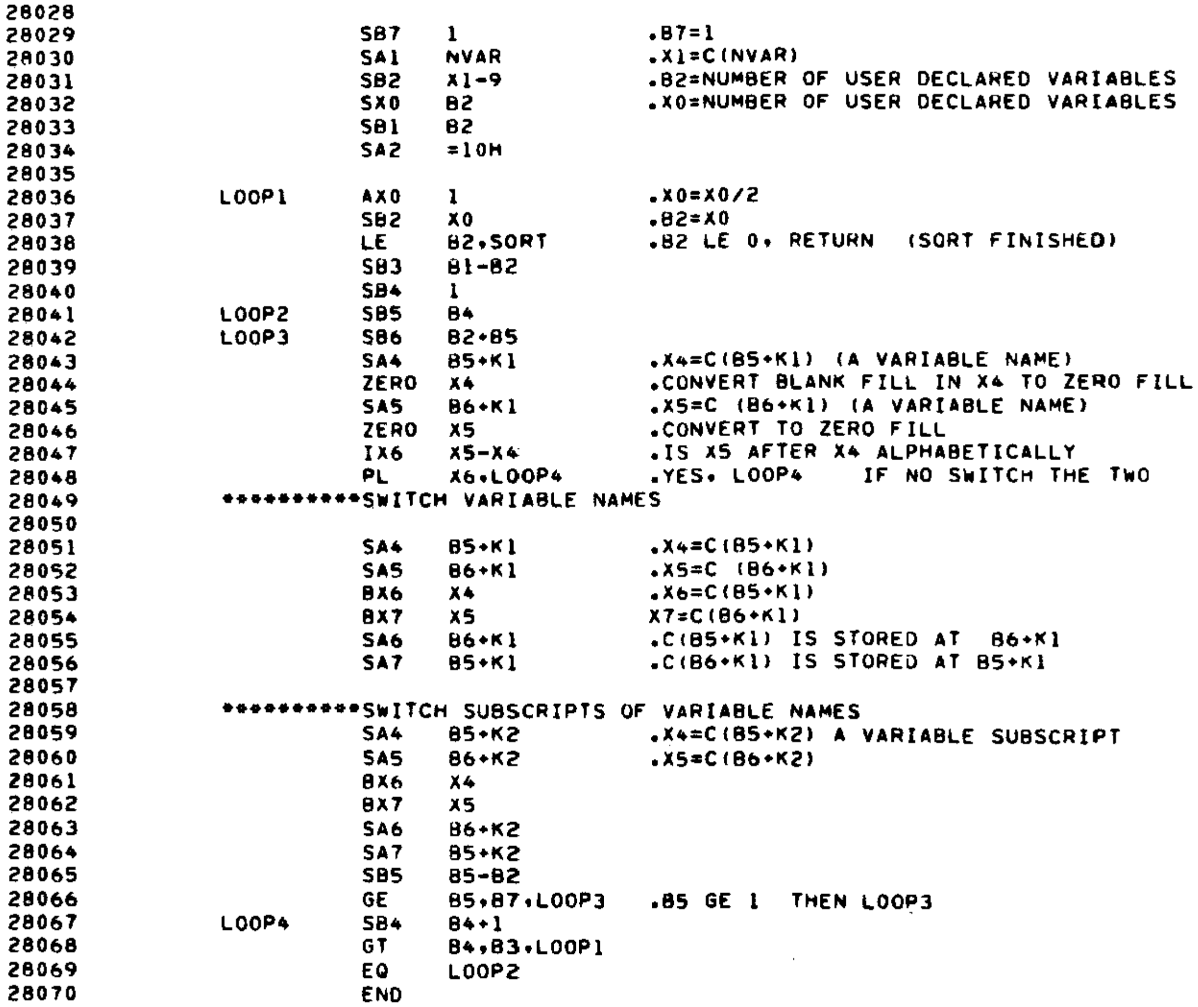
```
27032          C
27033          C.....GENERATE FTN EXTERNAL DECLARATIONS FOR EACH OF THE EXTERNAL EVENT
27034          C.....ROUTINES.
27035          C
27036                DO 20 I=1,NSUB
27037             20 WRITE (U6,75) NSBL(I)
27038                END FILE U6
27039                WRITE (U6,70)


27083             75 FORMAT (6X,  9HEXTERNAL ,A5,60X)
```

Line Number	Explanation
27036-27038	The second file of U6 contains an EXTERNAL statement for each routine declared by the user (plus the seven system defined events).
27039	The flag card signals the beginning of the next file on U6.

```
27040          C
27041          C.....GENERATE THE DATA ASSIGNMENT STATEMENTS WHICH SET THE VALUES FOR
27042          C.....THE SYSTEM VARIABLES.
27043          C
27044                NST=NSTOR-1
27045                WRITE (U6,80) NVAR,NST
27046                CALL SORT


27084             80 FORMAT (6X,  9HDATA XNV/,I3,  7H/, XNW/,I10,  1H/,44X)

28000                    IDENT SORT
28001                    LIST  -R,-G
28002                    ENTRY SORT
28003          SORT      BSS   1
28004          **********SORT ALPHABETIZES THE ARRAY CONTAINING VARIABLES DECLARED IN S
28005          *         STORAGE STATEMENTS
28006                    USE   /STORAGE/      .LABELED COMMON CONTAINING VARIABLES
28007          NVAR      BSS   1              .NUMBER OF VARIABLES
28008          LVR1      BSS   8              .SYSTEMS VARIABLES-1
28009          K1        BSS   991            .USER VARIABLES (MAX 990)
28010          LVR2      BSS   8              .SYSTEMS SUBSCRIPTS OF ABOVE VARIABLES
28011          K2        BSS   991            .USER SUBSCRIPTS
28012          NSTOR     BSS   1
28013                    USE   *
```

Line Number	Explanation
27044-27045	The contents of SIMCØMP's stacks and variables processed by the compiler are passed to the execution phase in DATA statements. The number of user declared variables will be defined by system variable XNV during execution.

Line Number	Explanation
27046	SØRT reorders the variable stacks LVR1 and LVR2 by alphabetizing the variable names in LVR1.
28005-28013	Labeled common STØRAGE contains the stacks LVR1 and LVR2. NOTE. Stack LVR1 is divided into LVR1 and K1 in SØRT. LVR1 contains the nine system variables which are not alphabetized and K1 contains the portion of the stack to alphabetize. NVAR is the total number of variables in LVR1 and K1.



```
28028
28029                SB7   1              .B7=1
28030                SA1   NVAR           .X1=C(NVAR)
28031                SB2   X1-9           .B2=NUMBER OF USER DECLARED VARIABLES
28032                SX0   B2             .X0=NUMBER OF USER DECLARED VARIABLES
28033                SB1   B2
28034                SA2   =10H
28035
28036       LOOP1    AX0   1              .X0=X0/2
28037                SB2   X0             .B2=X0
28038                LE    B2,SORT        .B2 LE 0, RETURN  (SORT FINISHED)
28039                SB3   B1-B2
28040                SB4   1
28041       LOOP2    SB5   B4
28042       LOOP3    SB6   B2+B5
28043                SA4   B5+K1          .X4=C(B5+K1) (A VARIABLE NAME)
28044                ZERO  X4             .CONVERT BLANK FILL IN X4 TO ZERO FILL
28045                SA5   B6+K1          .X5=C (B6+K1) (A VARIABLE NAME)
28046                ZERO  X5             .CONVERT TO ZERO FILL
28047                IX6   X5-X4          .IS X5 AFTER X4 ALPHABETICALLY
28048                PL    X6,LOOP4       .YES, LOOP4    IF NO SWITCH THE TWO
28049       **********SWITCH VARIABLE NAMES
28050
28051                SA4   B5+K1          .X4=C(B5+K1)
28052                SA5   B6+K1          .X5=C (B6+K1)
28053                BX6   X4             .X6=C(B5+K1)
28054                BX7   X5             X7=C(B6+K1)
28055                SA6   B6+K1          .C(B5+K1) IS STORED AT  B6+K1
28056                SA7   B5+K1          .C(B6+K1) IS STORED AT B5+K1
28057
28058       **********SWITCH SUBSCRIPTS OF VARIABLE NAMES
28059                SA4   B5+K2          .X4=C(B5+K2) A VARIABLE SUBSCRIPT
28060                SA5   B6+K2          .X5=C(B6+K2)
28061                BX6   X4
28062                BX7   X5
28063                SA6   B6+K2
28064                SA7   B5+K2
28065                SB5   B5-B2
28066                GE    B5,B7,LOOP3   .B5 GE 1  THEN LOOP3
28067       LOOP4    SB4   B4+1
28068                GT    B4,B3,LOOP1
28069                EQ    LOOP2
28070                END
```

Line Number	Explanation
28028-28033	B1 contains the number of variables in K1 to be sorted. SØRT divides the array into halves and compares corresponding items in each half, switching them if the item in the lower half is greater than the item in the upper half. The array is then divided into quarters, the items in the first quarter are compared with corresponding items in the second quarter and then those in the second quarter with those in the third quarter, etc. When an item is switched from the third quarter to the second quarter, it is then checked against the item in the first quarter to determine if it should be moved again. In general, when an item is moved from section to section it is compared with the corresponding item in the next lower section until a move is not made or the item reaches the lowest section. Continue dividing the array and following the above procedure until a pass is made comparing adjacent items.
28036-28038	The array is divided by 2. B2 contains the number of items in each section. If there are zero items in each section, then the preceding pass compared adjacent items and sorting is completed.
28039-28042	If the array were divided into halves: B4 contains the item number of each half being compared (B4=1 if the first item in the lower half is being compared with the first item in the upper half. B6 contains the index of the B4th item in the upper stack. Thus B6+K1 is the relative address of the B4th item in upper segment of K1. B5+K1 is the relative index of the B4th item in the lower part of K1. (B5 is the index of the B4th item in the lower stack.)
28043-28048	If the item in the higher section comes after the corresponding item in the lower section, compare the next item in each section (increment B4).
28049-28056	An item in the upper section should come before an item in the lower section. The items are switched.
28057-28064	K2 contains the subscripts of corresponding names in K1. Therefore, if elements of K1 are switched, then corresponding elements of K2 must be switched.

Line Number	Explanation
28065-28066	Since an item was switched, it must be compared with the corresponding item in the next lower section (LØØP3). If there is no lower section, then compare the next item in each section (LØØP4).
28067-28070	Increment B4 and compare adjacent items (LØØP2) unless all items of the two sections have been compared. Then divide sections by 2 (LØØP1).

```
28014
28015          ZERO         MACRO A
28016                       MX1    36          .FORM MASK IN 36 HIGH ORDER BITS
28017                       B_A    A*X1        .SAVE 30 HIGH BITS OF A
28018                       MX1    6           .FORM 6 BIT MASK
28019                       LX1    30          .SHIFT MASKING BITS AROUND
28020                       BX6    X1*X2       .X6= ...005500...B DEPENDING ON X1
28021           +           B_A    A-X6        .ZERO OUT THE BLANK IN A
28022                       LX1    6           .LEFT SHIFT MASKING BITS
28023                       BX3    A*X1        .X3=6 BITS OF A (ZERO FILL)
28024                       BX6    X1*X2       .X6=SAME ORDER 6 BITS OF X2
28025                       IX3    X3-X6       .COMPARE BITS OF A AND X2
28026           +           ZR     X3,*-2      .IF A BITS = X2 BITS (55) LOOP BACK
28027                       ENDM
```

Line Number	Explanation
28014-28027	This macro receives an element of K1 and returns only the 1-5 character variable name, left justified with zero fill.

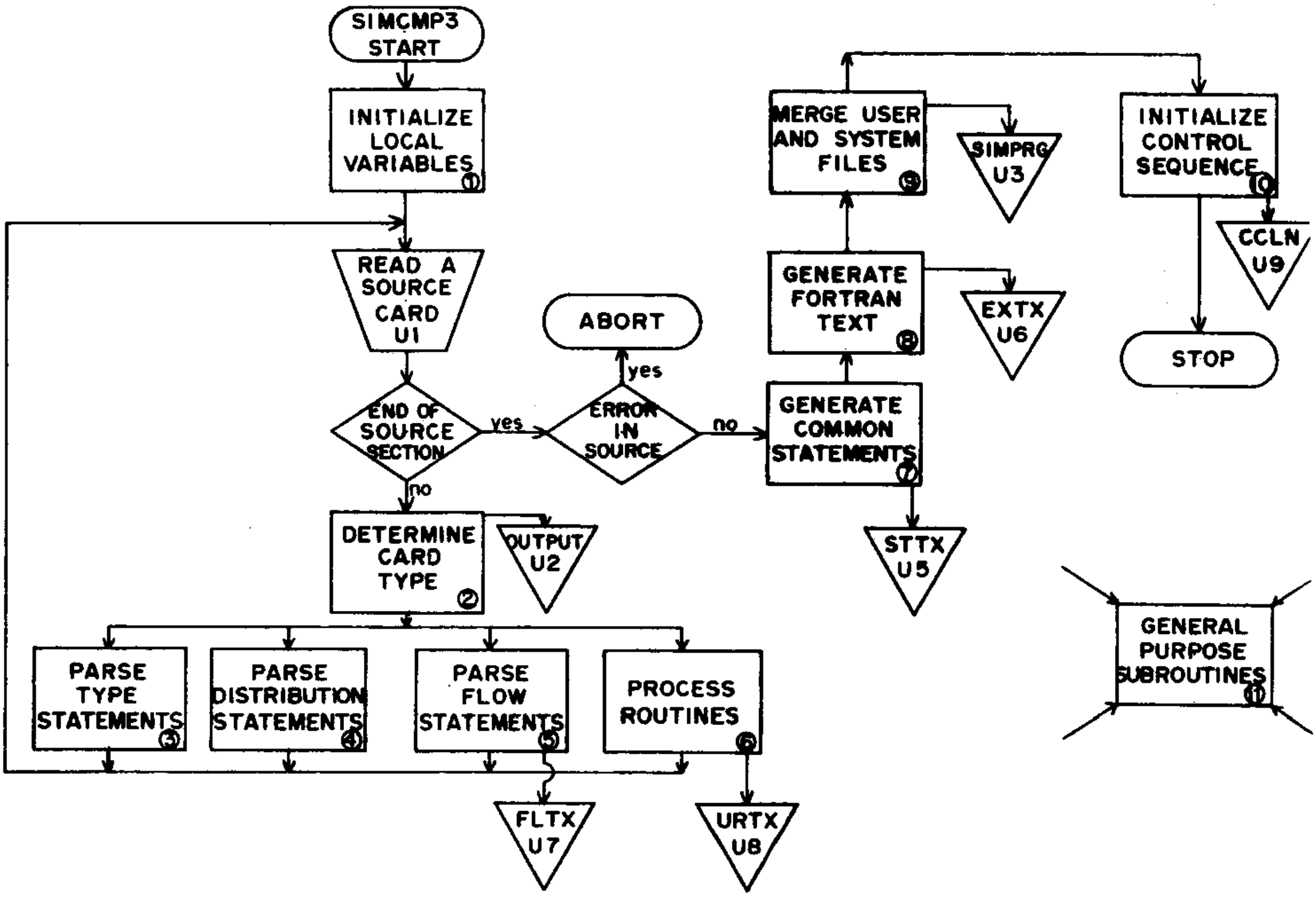
```
 27047                 DO 25 I=1,NVAR
 27048              25 WRITE (U6,85) I,LVR1(I),I,LVR2(I)
 27049                 WRITE (U6,90) NSUB
 27050                 DO 30 I=1,NSUB
 27051              30 WRITE (U6,95) I,NSRL(I)
 27052                 WRITE (U6,100) NFLW
 27053                 IF (NFLW.LE.0) GO TO 40
 27054                 DO 35 I=1,NFLW
 27055              35 WRITE (U6,105) I,NFLT(I)
 27056              40 END FILE U6
 27057                 WRITE (U6,70)


 27085              85 FORMAT (6X, 10HDATA XVTI(,I3,  2H)/,020,  2HB/,37X/12X, 10HDATA XV
 27086                1T2(,I3,  2H)/,020,  2HB/,31X)
 27087              90 FORMAT (6X, 10HDATA XNEX/,I3,  1H/,60X)
 27088              95 FORMAT (6X, 10HDATA XEXT(,I3,  2H)/,020,  2HB/,37X)
 27089             100 FORMAT (6X, 11HDATA XNFLW/,I4,  1H/,58X)
 27090             105 FORMAT (6X, 11HDATA XFLWT(,I4,  2H)/,020,  2HB/,35X)
```

Line Number	Explanation
27047-27048	For each variable in LVR1, a DATA statement is generated, containing the contents of LVR1 and LVR2. Thus the contents of LVR1(I) will be in XVT1(I) during system execution.
27049-27051	NSBL, which contains a list of all event names, is defined in the DATA section as system variable XEXT.
27052-27055	NFLT, the list of user declared flows, is defined in DATA cards as XFLWT.
27056-27057	An end of file is written, and the next file begins with the flag card.

```
27058     C
27059     C.....GENERATE CODE WHICH COMPUTES THE ENTRY ADDRESSES TO THE EVENT
27060     C.....ROUTINES AND STORES THESE IN THE EVENT ROUTINE TABLE.
27061     C
27062           DO 45 I=1,NSUB
27063        45 WRITE (U6,110) NSBL(I)
27064           END FILE U6
27065           WRITE (U6,70)
27066           END FILE U6
27067           END FILE U5
27068           WRITE (U5,70)
27069           END FILE U5
27070           END FILE U7
27071           WRITE (U7,70)
27072           END FILE U7
27073           RETURN
27074     C

27091       110 FORMAT (6X,  7HXN=XN+1,67X/6X,  8HXI=LOCF(,A5,  1H),60X/6X, 22HXEX
27092          1T(XN)=XEXT(XN).0.XI,52X)
27093     C
27094           END
```

Line Number	Explanation
27062-27063	The fifth file of U6 contains code which computes the entry address of each system and user event name.
27064-27074	Files U5, U6, and U7 are all ended with the final files of each containing only the flag card.

1.9. *Merge User Text and System Files*



Overview

The contents of files STTX (U5), EXTX (U6), FLTX (U7), and URTX (U8) are merged with file SIMCØM (U4) to create the file containing the completed FØRTRAN program SIMPRG (U3). The structure of the files is explained beginning in Section 1.8.

```
      C
      C.....GENERATE THE SIMULATION PROGRAM BY MERGING THE USER GENERATED FILE
      C.....WITH THE SYSTEMS TEXT FILE.
      C
            CALL POSITN (U4,2)
         75 CALL TRNSFH (U4,U3,NUNIT,NFILE)
            IF (NUNIT.LE.0) GO TO 85
            CALL POSITN (NUNIT,NFILE)
```

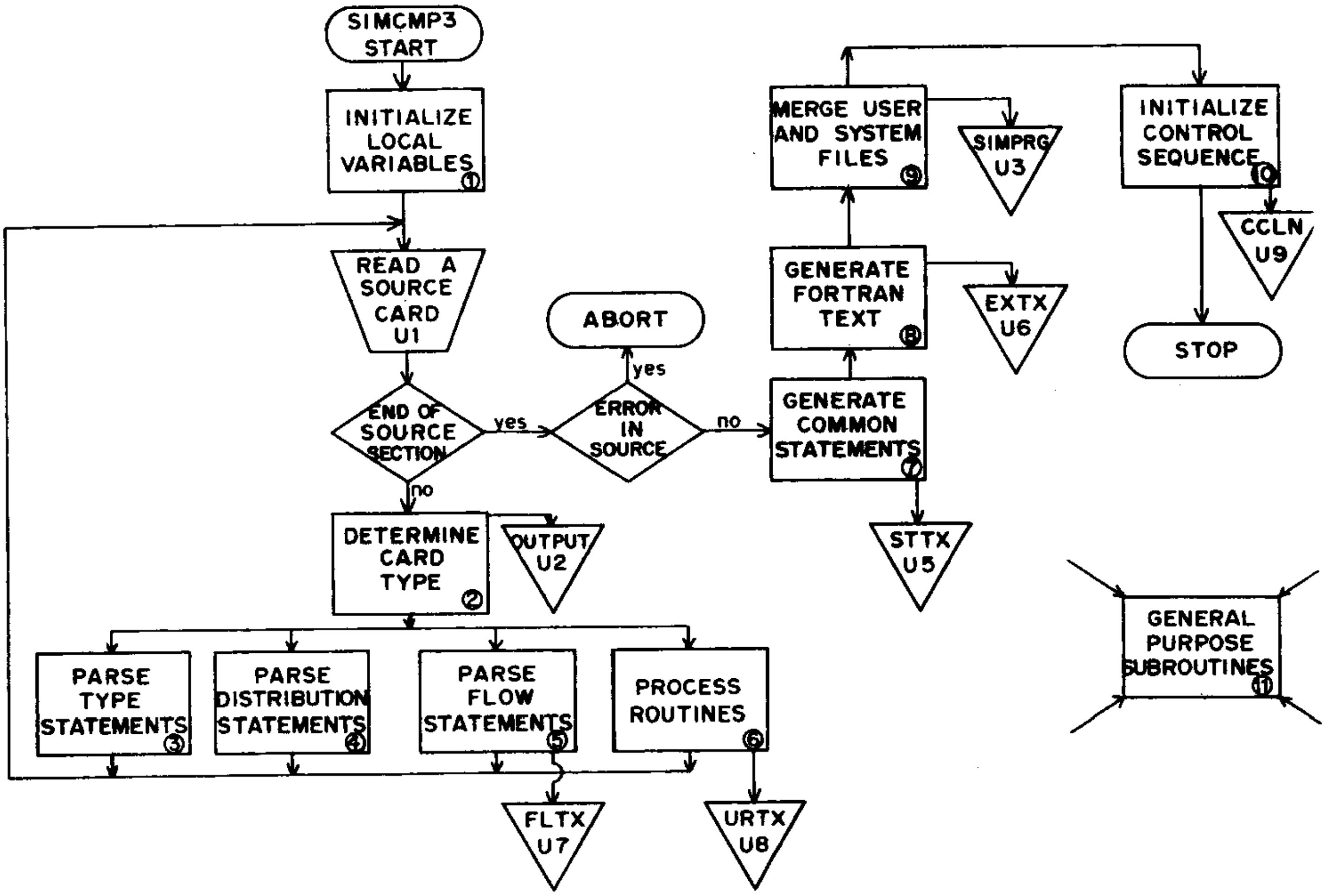
```
   80 CALL TRNSFR (NUNIT.U3.MUNIT,MFILE)
      IF (MUNIT.LE.0) GO TO 75
      CALL POSITN (MUNIT,MFILE)
      CALL TRNSFR (MUNIT.U3.LUNIT,LFILE)
      GO TO 80
   85 CONTINUE
      REWIND U3
```

```
          IDENT TRNSFR
          ENTRY TRNSFR
          TITLE  TRANSFER ONE FILE OF INFORMATION
          LIST -R,-G
          EXT SPWSA,RPWSA,IOREAD,IOWRITE,OPENS
*...THIS ROUTINE PREFORMS:
*    1)  TRANSFER OF ONE FILE OF DATA FROM LOGICAL TAPE UNIT (ARG1) T
*        LOGICAL TAPE UNIT (ARG2).
*    2)  RETURNS VALUES OF 2 DPC NUMBERS CONTAINED ON COMMENT CARD
*        ASSUMED TO FOLLOW EOF ON LOGICAL TAPE UNIT (ARG1).
*...FTN CALLING SEQUENCE:
*                CALL TRNSFR(ARG1,ARG2,ARG3,ARG4)
 GETBAS   MACRO
*...THIS MACRO RETURNS IN B2 THE FET BASE ADDRESS FOR THE FILE WITH
*    LOGICAL UNIT NUMBER GIVEN IN X1.
          SX1 X1+33B
          LX1 30
          SA2 F0S
          BX7 X1+X2
          SA7 FNS
          SB2 -FNS
          RJ =XGETBA
          NG B2,ERR
          ENDM
 TRNSFR   BSSZ 1
          SX6 A1
          SA6 BAL              .SAVE BASE ADDRESS OF ARG LIST.
          SA1 X1
          GETBAS               .GET BA OF FET FOR LOG UNIT NO IN ARG1
          SX6 B2
          SA6 LFN1             .STORE BA IN WRD 2 OF 2-WRD CALLS TO IO.
          SA6 LFN1A
          RJ OPENS             .OPEN FILE IF NECESSARY.
          SA1 BAL
          SA1 X1+1
          SA1 X1
          GETBAS               .GET BA OF FET FOR LOG UNIT NO IN ARG2.
          SX6 B2
          SA6 LFN2             .STORE BA IN WRD 2 OF 2-WRD CALL TO IO.
          RJ OPENS
          SA1 LFN1             .SET PARAMETERS IN X1, X2, X3 USED BY
          SA2 LFN2             .   SPWSA WHICH STORES ADDRESSES OF THE
          SX3 BUFR             .   WSA IN THE INPUT AND OUTPUT FETS.
          RJ SPWSA
 IOLOOP   NO                   .READ CARD INTO WSA.
          NO
          RJ IOREAD
 LFN1     BSSZ 1
          NG X1,EOF
          NZ X1,IOLOOP         .SKIP ZERO LENGTH RECORDS.
 +        NO                   .WRITE WSA.
          NO
          RJ IOWRITE
 LFN2     BSSZ 1
          EQ IOLOOP
 EOF      NO                   .EOF ENCOUNTERED, READ NEXT CARD INTO WSA.
          NO
          RJ IOREAD
 LFN1A    BSSZ 1
          MX1 54
          SA2 BUFR             .DECODE NEXT UNIT NUMBER AND STORE IN ARG3.
          BX6 -X1*X2
          SX6 X6-33B
          SA2 BAL
          SA2 X2+2
```

```
14065              SA6 X2
14066              SA2 BUFR+1          .DECODE NEXT SKIP NO AND STORE IN ARG4.
14067              AX2 30
14068              BX6 -X1*X2
14069              SX6 X6-33B
14070              SA2 BAL
14071              SA2 X2+3
14072              SA6 X2
14073              RJ RPWSA            .RESTORE WSA ADDRESSES IN FETS.
14074              EQ TRNSFR
14075     ERR      SA1 FN$             .IF IO FILES CANNOT BE FOUND PUT MESSAGE
14076              BX6 X1              .   IN JOB DAYFILE AND ABORT.
14077              SA6 ERRM+2
14078     +        NO
14079              NO
14080              RJ =XCPC
14081              VFD 18/3LMSG,6/40B,18/0,18/ERRM
14082     +        NO
14083              NO
14084              RJ =XCPC
14085              VFD 18/3LABT,6/60B,36/0
14086     FN$      BSSZ 1
14087     FO$      DATA 24012005000000000000B
14088     BAL      BSSZ 1
14089     ERRM     VFD 60/10LFILE NOT F
14090              VFD 60/10LOUND -
14091              VFD 60/0
14092     BUFR     BSSZ 9              .WORKING STORAGE AREA (WSA).
14093              END
15000              IDENT SPWSA
15001              ENTRY SPWSA,RPWSA
15002              TITLE  SET/RESTORE WORKING STORAGE AREA PARAMETERS IN FETS
15003              LIST -R,-G
15004    *...THIS ROUTINE SETS (ENTRY SPWSA) AND RESTORES (ENTRY RPWSA) THE
15005    *    WSA ADDRESSES IN THE FETS OF THE FILES USED BY TRNSFR.
15006    *    THIS ROUTINE EXPECTS:
15007    *         X1 = BA OF FET FOR FILE SPECIFIED BY ARG1 IN TRNSFR,
15008    *         X2 = BA OF FET FOR FILE SPECIFIED BY ARG2 IN TRNSFR.
15009    *         X3 = ADDRESS OF WSA.
15010     SPWSA    BSSZ 1
15011              BX7 X1
15012              SA7 FETI            .SAVE BA OF INPUT FET.
15013              BX7 X2
15014              SA7 FETO            .SAVE BA OF OUTPUT FET.
15015              BX7 X3
15016              SA7 WSA             .SAVE ADDRESS OF WSA.
15017              SA1 FETI            .SAVE INITIAL CONTENTS OF WRD 5 INPUT FET.
15018              SA2 X1+5
15019              BX7 X2
15020              SA7 WSAI
15021              SA1 WSA
15022              SX7 X1+9
15023              LX1 30
15024              BX7 X1+X7
15025              SA7 A2              .SET WSA ADDRESSES IN INPUT FET.
15026              SA1 FETO            .SAVE INITIAL CONTENTS OF WRD 5 OUTPUT FET.
15027              SA2 X1+5
15028              BX7 X2
15029              SA7 WSAO
15030              SA1 WSA
15031              SX7 X1+9
15032              LX1 30
15033              BX7 X1+X7
15034              SA7 A2              .SET WSA ADDRESSES IN OUTPUT FET.
15035              EQ SPWSA
15036     RPWSA    BSSZ 1
15037              SA1 FETI            .RESTORE WRD 5 OF IO FETS.
15038              SB2 X1
15039              SA1 WSAI
15040              BX7 X1
15041              SA7 B2+5
15042              SA1 FETO
15043              SB2 X1
15044              SA1 WSAO
15045              BX7 X1
```

```
                SA7 B2+5
                EQ RPWSA
        FETI    BSSZ 1
        FETO    BSSZ 1
        WSA     BSSZ 1
        WSAI    BSSZ 1
        WSAO    BSSZ 1
                END
                IDENT POSITN
                ENTRY POSITN
                TITLE  PREFORM REWIND AND SKIP ON SPECIFIED FILE
                LIST -R,-G
                EXT OPEN$
       *...THIS ROUTINE REWINDS AND SKIPS (ARG2) FILES FORWARD ON THE LOGICAL
       *    UNIT SPECIFIED BY (ARG1)
       *    FORTRAN CALLING SEQUENCE:
       *              CALL POSITN(ARG1,ARG2)
       *    WHERE:
       *              ARG1 = LOGICAL UNIT NO OF FILE.
       *              ARG2 = NO OF FILES TO BE SKIPPED IN FORWARD DIRECTION.
        POSITN  BSSZ 1
                SA2 A1+1            .STORE NO FILES TO BE SKIPPED IN NFS.
                SA2 X2
                BX6 X2
                SA6 NFS
                SA1 X1              .GET BA OF FET FOR LOGICAL UNIT IN ARG2.
                SX2 X1+33B
                LX2 30
                SA3 FILE0#
                BX6 X2+X3
                SA6 FILEN#
                SB2 -FILEN#
                RJ =XGETBA
                LT B2.B0.POSITN
                RJ OPEN$            .OPEN FILE IF NECESSARY.
        +       SA1 B2              .REWIND FILE.
                RJ =XCPC
        OPWRD1  DATA 00000320000000000050B
                SA1 NFS             .EXIT IF NFS .LE. ZERO.
                ZR X1.POSITN
                NG X1.POSITN
                LX1 18
                SA2 OPWRD
                BX6 X1+X2           .MASK NFS INTO SKIP INSTRUCTION WORD.
                SA6 OPWRD2
        +       SA1 B2              .PREFORM SKIP.
                RJ =XCPC
        OPWRD2  BSSZ 1
                SA1 B2              .CLEAR EOF BIT IN FET.
                SA2 =20B
                BX6 -X2*X1
                SA6 B2
                EQ POSITN
        FILE0#  DATA 24012005000000000000B
        FILEN#  BSSZ 1
        NFS     BSSZ 1
        OPWRD   DATA 00000320000000740240B
                END
                IDENT OPEN
                ENTRY OPEN$
                TITLE  PREFORM OPEN/RECALL WITH NO REWIND ON SPECIFIED FILE
                LIST -R,-G
       *...THIS ROUTINE PREFORMS AN OPEN WITH NO REWIND ON THE FILE SPECIFIED
       *    BY THE FET BA CONTAINED IN REGISTER B2 UPON ENTRY.  IF FILE IS
       *    ALREADY OPEN THEN RECALL ONLY.
        OPEN$   BSSZ 1
                SA5 B2
                SX5 X5
                NZ X5.NOPEN
        +       SA1 B2
                RJ =XCPC
        +       DATA 00000400000000000120B
        NOPEN   SA1 B2
                RJ =XCPC
        +       DATA 00000120000000777777B
                EQ OPEN$
                END
```

1.10. *Initialize Control Sequence*



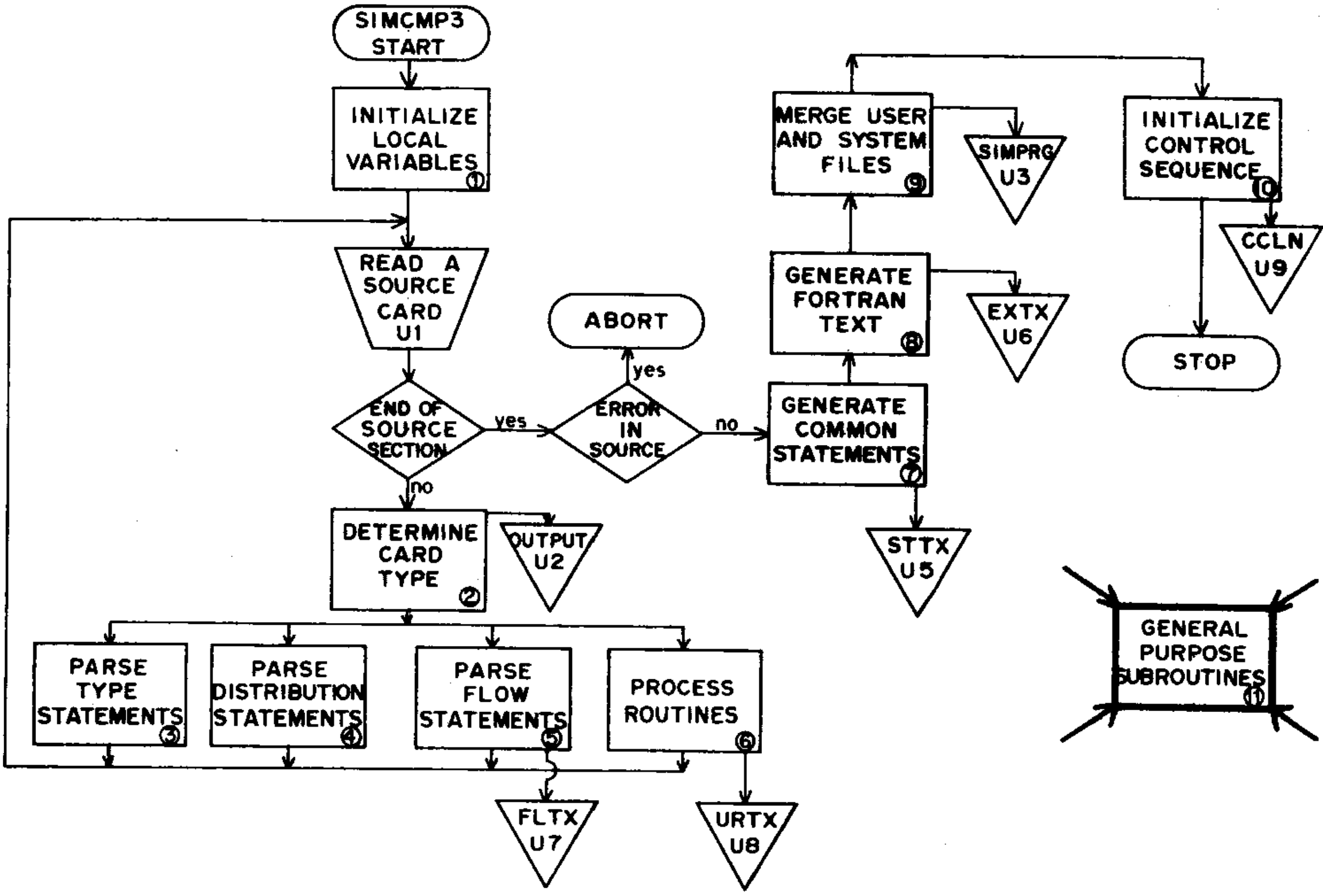
Overview

If a NØGØ. card is not encountered in the source deck, then the system control cards to be used are written to file CCLN (U9) and routine CCLTR is called. Routine CCLTR calls the peripheral processor routine CCL which causes the operating system to use file CCLN as the control card file instead of the normal first record of the INPUT file.

```
10180                 IF (NOGO) STOP
10181                 IF (DEBUG) WRITE (9,115)
10182                 IF (.NOT.DEBUG) WRITE (9,120)
10183                 REWIND 9
10184                 CALL CCLTR
10185                 STOP
```

```
                 IDENT CCLTR
                 ENTRY CCLTR
                 TITLE  TRANSFER CONTROL TO GENERATED CONTROL CARD STREAM.
                 LIST -R,-G
*...THIS ROUTINE CAUSES THE NEXT CONTROL CARDS TO BE USED BY THE SYSTEM
*    TO BE DRAWN FROM FILE (CCLN) WHICH WAS GENERATED IN THE MAIN
*    PROGRAM.
 CCLTR           BSSZ 1
                 RJ =XFLUSHO
                 SA5 64B                   .SAVE IN X5 C(RA+64B).
                 SX6 1                     .STORE PARAMETER COUNT (1) IN RA+64B.
                 SA6 64B
                 SA1 FILE                  .STORE PARAMETER (FILE NAME) IN RA+2.
                 BX6 X1
                 SA6 2
                 SX6 0                     .STORE ZERO (END OF PARAMS) IN RA+3.
                 SA6 3
 +               SA1 1                     .WAIT TILL LAST REQUEST COMPLETED.
                 NZ X1,*
                 SA1 REQU                  .POST REQUEST FOR PP PROGRAM CCL.
                 BX6 X1
                 SA6 1
 +               SA1 1                     .WAIT TILL REQUEST COMPLETED.
                 NZ X1,*
                 BX6 X5                    .RESTORE 64B.
                 SA6 64B
                 EQ CCLTR
FILE             VFD 60/4LCCLN
REQU             VFD 18/3LCCL,42/0
                 END
                 IDENT      FLUSHO
                 LIST       -R,-G
                 ENTRY      FLUSHO
FLUSHO           BSSZ       1
                 SB2        -GWORD
                 RJ         =XGETBA
                 LT         B2,B0,FLUSHO
                 SA4        B2+1
                 SA5        B2+2
                 IX6        X4-X5
                 ZR         X6,FLUSHO
                 SA5        FWORD
                 BX6        X5
                 SA6        B2
                 SA5        CIOC
                 SX6        B2
                 BX6        X6+X5
 +               SA1        1
                 NZ         X1,*
                 SA6        1
 +               SA1        1
                 NZ         X1,*
                 EQ         FLUSHO
CIOC             VFD        18/3LCIO,2/1,40/0
GWORD            VFD        60/6LOUTPUT
FWORD            VFD        36/6LOUTPUT,24/24B
                 END
```

1.11. *General Purpose Routines*



Overview

The contents of this section are small subroutines which are called to perform a generalized function from several locations or sections of SIMCMP3, the routines are:

(1) FMTPG.

(2) FLCØR.

```
      SUBROUTINE FMTPG (N)
      COMMON /OUTP/ NLINE,NPAGE,WHEN,PRINT,NOGO,DEBUG
      COMMON /UNITS/ U1,U2,U3,U4,U5,U6,U7,U8
      INTEGER U1,U2,U3,U4,U5,U6,U7,U8
      LOGICAL PRINT
C
C.....THIS ROUTINE PAGE-FORMATS THE SOURCE LISTING OUTPUT.
C
      NLINE=NLINE+N
```

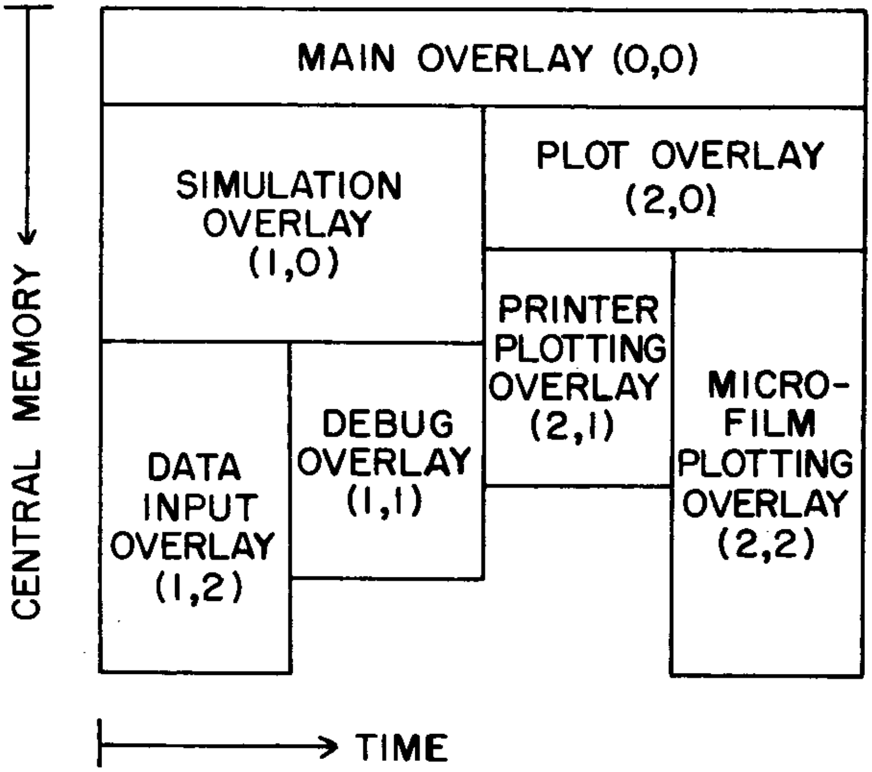
```
11009          IF (NLINE.LE.54) RETURN
11010          NPAGF=NPAGE+1
11011          WRITE (U2,15) WHEN,NPAGE
11012          NLINE=2
11013          RETURN
11014    C
11015       15 FORMAT ( 20H1SIMCOMP VERSION 3.0,10X, 14HSOURCE LISTING,26XA10,10X
11016         1,  8HPAGE NO ,I4//)
11017    C
11018          END
```

Line Number	Explanation
11000-11018	FMTPG prints a header page on the output file. N represents the number of lines that NLINE is to be incremented. There are 54 lines per page. NPAGE is the page number written onto the title line. WHEN is the date.

```
12000               IDENT FLCOR
12001               ENTRY FLCOR
12002               TITLE  CALCULATE AVAILABLE STORAGE FOR FLOW TABLES
12003               LIST -R,-G
12004    *...THIS ROUTINE RETURNS THE NUMBER OF AVAILABLE WORDS USEABLE BY THE
12005    *    FLOW TABLE AND THE CURRENT USED FIELD LENGTH NOT INCLUDING FLOW
12006    *    TABLFS.  FORTRAN CALLING SEQUENCE:
12007    *           CALL FLCORE(NFMAX,NCORE
12008     CORE      BSS 1
12009     MAX       DATA 9999
12010     FLCOR     BSS 1
12011               SB1 X1                .SAVE ADDRESSES OF ARGUMENTS
12012               SA1 A1+1              .    B1 = A(NFMAX)
12013               SB2 X1                .    B2 = A(NCORE)
12014               SX6 B0                .ACCESS CURRENT FL, RETURNED IN CORE.
12015               SA6 CORE
12016      +        NO
12017               NO
12018               RJ =XCPC
12019               VFD 18/3LMEM,24/60000000B,18/CORE
12020               SA4 CORE
12021      .        AX4 30
12022               SA5 65B               .X5 = LWA OF LOAD.
12023               IX6 X4-X5
12024               SA3 =2
12025               IX6 X6+X3
12026               SA2 MAX
12027               IX4 X6-X2
12028               NG X4,SET
12029               BX6 X2
12030     SET       SA6 B1                .X6 = MIN(9999,AMOUNT OF CORE AVAILABLE
12031               IX6 X5-X3             .    FOR FLOW TABLES)
12032               SA6 B2                .X6 = AMOUNT OF CORE USED WITHOUT FLOWS.
12033               EQ FLCOR
12034               END
```

PART II

EXECUTION OF THE FØRTRAN OBJECT CODE-- THE SIMULATION



Simulation Program Overview

The above diagram illustrates the location in core of various overlays which comprise the simulation program through time (as the programs execute it). Briefly the following is the sequence of execution. The main overlay (0,0) is in core at all times. The simulation execution overlay (1,0) is called into core by (0,0). The (1,0) overlay in turn calls the data input overlay (1,2). After the data section is processed, the execution of the simulation is done in the (1,0) overlay. If an arithmetic mode error is detected during the execution of the simulation, a subroutine in (0,0) calls in the debugging overlay. When the simulation overlay is finished executing, if PLØT's were requested, the (0,0) overlay calls in the plot overlay (2,0). Depending upon whether printer plots or microfilm plots are requested, either the printer plotting overlay (2,1) or the microfilm plotting overlay (2,2) is called in.

```
C
      OVERLAY(NEWT1,0,0)
      PROGRAM XMONITR (EXFILE=64,INPUT=64,OUTPUT=64,PLOTER=64,FILMPL=64,
     1PLTSTK=64,DEBUG=64,TAPE3=PLOTER,TAPE4=DEBUG,TAPE5=INPUT,TAPE6=OUTP
     2UT,TAPE7=EXFILE,TAPE8=PLTSTK)
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XRVC/ IRF,IMP,ICORE,IXJ(17),IER,IRA
      INTEGER XUO,XUI,XUP,XUE,XBF
      LOGICAL XPLFG,XFILM
      LOGICAL XTRACE
      CALL XDMCL
      XTRACE=.FALSE.
      XUP=3
      XUI=5
      XUO=6
      XUE=7
      XBF=8
      CALL XIFDBG
      REWIND XUP
      CALL OVERLAY (4HMAIN,1,0)
      IF (.N.XPLFG) STOP
      CALL OVERLAY (4HMAIN,2,0)
      STOP
C
      END
          IDENT XDMCL
          ENTRY XDMCL
          LIST -R,-G
          TITLE DISPLAY MINIMUM CORE REQUIRED FOR LOADING AND EXECUTION.
*...THIS ROUTINE REWINDS AND READS THE OUTPUT FILE SEARCHING FOR THE
*     OVERLAY LOADER MAPS.  IF LOADER MAPS ARE FOUND THE MINIMUM CORE RE-
*     QUIRED FOR LOADING AND EXECUTION OF THE JOB IS COMPUTED AND DIS-
*     PLAYED IN THE JOB DAYFILE.
 EQX12    MACRO
*...TESTS FOR EQUIVALENCE BETWEEN REGISTERS X1 AND X2.  IF THEY ARE
*     EQUIVALENT X3 IS SET TO ZERO.
          BX3 X1*X2
          BX3 -X3
          BX4 X1+X2
          BX3 X3*X4
          CX3 X3
          ENDM
 DCTOI    MACRO
*...CONVERTS THE DISPLAY CODED WORD IN REGISTER X1 INTO AN INTEGER IN
*     X1.
          LOCAL DC1,DC2
          BX0 X1
          MX1 0
          SX2 77B
          SX3 33B
          SB2 55B
          LX0 12
 DC1      LX0 6
          BX4 X0*X2
          SB3 X4
          EQ B2,B3,DC2
          IX4 X4-X3
          LX1 3
```

```
                BX1 X1+X4
                EQ DC1
 DC2            BSS 0
                ENDM
 ITODC          MACRO
*...CONVERTS THE INTEGER IN X1 INTO OCTAL-DPC IN X6.
                LOCAL IT1
                MX6 0
                LX1 42
                SX2 7B
                SX3 33B
                SB1 0
                SB2 6
 IT1            LX1 3
                BX0 X1*X2
                IX0 X0+X3
                LX6 6
                BX6 X6+X0
                SB1 B1+1
                LT B1.B2.IT1
                ENDM
 XDMCL          BSSZ 1
                RJ XPOSF              .READY OUTPUT FILE FOR READ.
 RDLP           RJ XRDLF              .READ A LINE OF OUTPUT.
                SA1 EOI               .CHECK FOR END-OF-INFORMATION.
                NZ X1.ERDL
                SA1 LINE              .CHECK FOR 1-ST LINE OF LOADER MAP.
                SA2 =10H1CORE MAP
                EQX12
                ZR X3.MTCH1
                SA1 CHFLG             .CHECK IF OK FOR SEARCH FOR FWA TABLES.
                ZR X1.RDLP
                SA1 LINE+3            .CHECK FOR 3-RD LINE OF LOADER MAP.
                SA2 =10HFWA TABLES
                EQX12
                ZR X3.MTCH2
                EQ RDLP
 MTCH1          SA1 LINE+2            .MAKE SURE MAP IS OVERLAY.
                SA2 =10H  OVERLAY
                EQX12
                NZ X3.RDLP
                SX6 1
                SA6 FIND              .SET MAP FOUND FLAG.
                SA6 CHFLG             .SET CHECK FOR FWA FLAG.
                SA1 LINE+10           .GET LWA LOAD.
                DCTOI                 .CONVERT FROM DPC TO INTEGER.
                SA2 LWA               .FIND MAX LWA.
                IX3 X2-X1
                PL X3.RDLP
                BX6 X1
                SA6 A2
                EQ RDLP
 MTCH2          MX6 0                 .ZERO OUT CHECK FWA FLAG.
                SA6 CHFLG
                SA1 LINE+4            .GET FWA TABLES.
                DCTOI                 .CONVERT FROM DPC TO INTEGER.
                SA2 FWA               .FIND MIN FWA.
                IX3 X1-X2
                PL X3.RDLP
                BX6 X1
                SA6 A2
                EQ RDLP
 ERDL           SA1 FIND              .CHECK FOR MAPS FOUND.
                ZR X1.XDMCL
 +              NO                    .GET CURRENT FL.
                NO
                RJ =XCPC
                VFD 18/3LMEM.3/6.21/0.18/FL
                SA1 FWA               .ROUND FWA DOWN NEAREST 100B.
                AX1 6
                LX1 6
                SA2 LWA               .ROUND LWA UP NEAREST 100B.
                SX3 X2
```

```
31106                    AX3 6
31107                    LX3 6
31108                    SB2 X2
31109                    SB3 X3
31110                    EQ B2,B3,RNDD
31111                    SX3 X3+100B
31112         RNDD       IX4 X1-X3
31113                    SA3 FL
31114                    AX3 30
31115                    IX1 X3-X4
31116                    ITODC                .CONVERT INTEGER TO DPC.
31117                    SA2 DFM+3            .DISPLAY MINIMUM FL IN DAYFILE.
31118                    BX6 X6+X2
31119                    SA6 A2
31120         +          NO
31121                    NO
31122                    RJ =XCPC
31123                    VFD 18/3LMSG,6/40B,18/0,18/DFM
31124                    EQ XDMCL
31125         XPOSF      BSSZ 1               .ENTRY FOR READYING OUTPUT FILE FOR READ.
31126                    SB2 -FNAME           .GET BA OF OUTPUT FET.
31127                    RJ =XGETBA
31128         +          SA1 B2               .OPEN FILE WITH REWIND.
31129         -          RJ =XCPC
31130                    VFD 16/4,1/0,1/1,22/0,18/160B
31131                    SX1 LINE             .SET WSA PARAMETERS IN FET+5.
31132                    SX2 X1+14
31133                    LX1 30
31134                    BX6 X1+X2
31135                    SA6 B2+5
31136                    SX6 B2               .SAVE FET BA.
31137                    SA6 FET
31138                    EQ XPOSF
31139         XRDLF      BSSZ 1               .ENTRY TO READ ONE LINE INTO WSA.
31140         READ       NO                   .EXECUTE READ.
31141                    NO
31142                    RJ =XIOREAD
31143         FET        BSS 1
31144                    ZR X1,XRDLF          .DATA READ.
31145                    PL X1,READ           .SHORT RECORD READ.
31146                    SA2 EOF              .CHECK FOR DOUBLE EOF (EOI).
31147                    SX6 1
31148                    SA6 EOF
31149                    ZR X2,READ
31150                    SA6 EOI              .SET EOI.
31151                    EQ XRDLF
31152         FNAME      VFD 60/6LOUTPUT
31153         LINE       BSSZ 14              .WORKING STORAGE AREA (WSA).
31154         FIND       BSSZ 1
31155         CHFLG      BSSZ 1
31156         EOI        BSSZ 1
31157         EOF        BSSZ 1
31158         LWA        DATA 0
31159         FWA        DATA 777777B
31160         FL         BSSZ 1
31161         DFM        DATA 10L   MINIMUM
31162                    DATA 10L FIELD LEN
31163                    DATA 10LGTH REQUIR
31164                    DATA 4LED
31165                    DATA 0
31166                    END
32000                    IDENT XCMFL
32001                    ENTRY XNCM,XCFL,XRFL
32002                    TITLE  ROUTINES FOR FIELD LENGTH REDEFINITION.
32003                    LIST -R,-G
32004         *...XNCM   -  RETURNS THE VALUE STORED IN THE LOWER 18 BITS OF RA+65.
```

```
*     .          THE NEXT AVAILABLE CM WORD.
*...XCFL  -  RETURNS THE VALUE OF THE CURRENT FIELD LENGTH.
*...XRFL  -  REDEFINES THE FIELD LENGTH.
          SPACE 1
 XNCM     BSSZ 1
          SA3 65B
          SX6 X3
          SA6 X1
          EQ XNCM
          SPACE 1
 XCFL     BSSZ 1
          SX6 X1             .SAVE ADDRESS OF ARGUMENT.
          SA6 SV.X1
          SX6 0              .OBTAIN CURRENT FIELD LENGTH.
          SA6 CORE.
 *        NO
          NO
          RJ =XCPC
          VFD 18/3LMEM,3/6B,21/0B,18/CORE.
          SA1 SV.X1          .RETURN CURRENT FIELD LENGTH IN ARGUMENT.
          SA3 CORE.
          BX6 X3
          AX6 30
          SA6 X1
          EQ XCFL
          SPACE 1
 XRFL     BSSZ 1
          SA1 X1             .REDEFINE FIELD LENGTH.
          SX6 X1
          LX6 30
          SA6 CORE.
 *        NO
          NO
          RJ =XCPC
          VFD 18/3LMEM,3/6B,21/0B,18/CORE.
          EQ XRFL
 SV.X1    BSS 1
 CORE.    BSS 1
          END
      SUBROUTINE XRECVR (NXJ,NER,NRA)
      COMMON /XRVC/ IRF,IMP,ICORE,IXJ(17),IER,IRA
      DIMENSION NXJ(17)
C
C.....CONTROL IS TRANSFERED TO THIS ROUTINE IF AN ARITHMETIC MODE ERROR
C.....OCCURS AND CROSS-REFERENCE MAPS AND/OR LOADER MAP HAVE BEEN DETER-
C.....MINED ACCESSABLE BY ROUTINE (XIFDBG).  THIS ROUTINE LOADS THE
C.....EXCHANGE JUMP PACKAGE INTERPRETATION AND DEBUGGING OVERLAY.
C
      CALL XRFL (ICORE)
      CALL REMARK (16HDEBUG,PROCESSING)
      DO 15 I=1,17
   15 IXJ(I)=NXJ(I)
      IRA=NRA
      CALL OVERLAY (4HMAIN,1,1)
      NER=1
      RETURN
C
      END
          IDENT XIFDBG
          ENTRY XIFDBG
          EXT XRECVR
          LIST -R,-G
          TITLE    ERROR RECOVERY INITIALIZATION
          SPACE 1
*...THIS ROUTINE EXAMINES THE CONTROL CARD RECORD TO DETERMINE WHAT
*    INFORMATION HAS BEEN GENERATED FOR ARITHMETIC MODE ERROR EXCHANGE
*     PACKAGE INTERPRETATION AND DEBUGGING.
*     FORTRAN CALLING SEQUENCE:
*           CALL XIFDBG
          SPACE 1
*...THE FOLLOWING ACTIONS ARE TAKEN:
*   (A)   IF A CORE MAP WAS GENERATED (I.E. MAP.OFF. NOT PRESENT) THEN
*         INITIALIZE ERROR RECOVERY ROUTINE FOR EXCHANGE JUMP PACKAGE
*         DECODING RELATIVE TO ROUTINE ENTRY POINT ADDRESSES.
*   (B)   IF CROSS REFERENCE TABLES WERE GENERATED (I.E. LP=DEBUG PARA-
*         METER PRESENT ON FTN CONTROL CARD) IN ADDITION TO (A) ABOVE,
*         THEN INITIALIZE ERROR RECOVERY ROUTINE FOR EXCHANGE PACKAGE
*         DECODING RELATIVE TO VARIABLE NAME LOCATIONS PLUS SELECTIVE
*         DUMPING OF VARIABLE STORAGE LOCATIONS CONTAINED IN ROUTINE
*         DETECTING ERROR.
*   (C)   IF NEITHER (A) NOR (B) ABOVE IS THE CASE, DO NOT INITIALIZE
*         ERROR RECOVERY ROUTINE.
          SPACE 1
  XIFDBG  BSSZ 1
          SB7 0
          SPACE 1                   .POSITION CONTROL CARD RECORD TO BEGINNING.
  BCKSP   SA1 BKSP
          BX6 X1
          SA6 FUNC
          SB7 B7+1
          NO
   +      NO
          RJ =XCPC
          VFD 18/3LACE,2/3B,22/0,18/FUNC
          SA1 FUNC
          AX1 4
          SX2 1
          BX3 X1*X2
          ZR X3,BCKSP
          SPACE 1
  RDCRD   SB7 B7-1
          NG B7,FINIS               .READ A CONTROL CARD.
          SA1 READ
          BX6 X1
          SA6 FUNC
          NO
   +      NO
          RJ =XCPC
          VFD 18/3LACE,2/3B,22/0,18/FUNC
          SPACE 1                   .WAIT UNTIL READ COMPLETED.
  WAITL   SA1 FUNC
          SX2 1
          BX3 X1*X2
          ZR X3,WAITL
          SPACE 1                   .CHECK FOR EOR.
          AX1 4
          BX3 X1*X2
          NZ X3,FINIS
          SPACE 1                   .SCAN CARD FOR MAP OR FTN.
          SA1 70B
          LX1 18
          MX2 42
          BX2 -X2
          BX2 X1*X2
          SA3 MAP
          IX3 X3-X2
          ZR X3,MAPCD
          SA3 FTN
          IX3 X3-X2
          NZ X3,RDCRD
          SPACE 1                   .FTN CARD ENCOUNTERED, BEGIN SCAN FOR
          SB1 0
          SB2 0             .  ¢=DEBUG> FIELD.
          SB3 11
          SA1 70B
          SX6 0
SCANA     SB2 B2+1
          LT B2,B3,SCANB
          SB2 0             .GET NEXT WORD ON CARD
          SA1 A1+1
          EQ SCANA
SCANB     LX1 6             .GET NEXT CHARACTER IN CURRENT WORD.
          SX2 77B
          BX2 X1*X2
          SPACE 1
          SX3 54B           .CHECK FOR EQUAL ¢=>.
          IX3 X3-X2
          NZ X3,SCANC
          SB1 1
          SX6 0
          EQ SCANA
          SPACE 1
SCANC     SX3 56B           .CHECK FOR COMMA ¢,>.
          IX3 X3-X2
          NZ X3,SCAND
          ZR B1,SCANA
          SA4 DBG
          IX4 X4-X6
          ZR X4,MATCH
          SB1 0
          SX6 0
          EQ SCANA
          SPACE 1
SCAND     SX3 52B           .CHECK FOR RIGHT PAREN ¢)>.
          IX3 X3-X2
          NZ X3,SCANE
          ZR B1,RDCRD
          SA4 DBG
          IX4 X4-X6
          ZR X4,MATCH
          EQ RDCRD
          SPACE 1
SCANE     SX3 57B           .CHECK FOR PERIOD ¢.>.
          IX3 X3-X2
          NZ X3,SCANF
          ZR B1,RDCRD
          SA4 DBG
          IX4 X4-X6
          ZR X4,MATCH
          EQ RDCRD
          SPACE 1
SCANF     ZR B1,SCANA       .STORE CHARACTER IF B1 FLAG SET.
          LX6 6
          BX6 X6+X2
          EQ SCANA
          SPACE 1
MATCH     SX6 1             .¢DEBUG> FIELD FOUND.
          SA6 XREF
          EQ RDCRD
          SPACE 1
MAPCD     LX1 24            .MAP CARD ENCOUNTERED. CHECK FOR ¢OFF>.
          MX2 42
          BX2 -X2
          BX2 X1*X2
          SA3 OFF
          IX3 X3-X2
          SX6 1
          SA6 CMAP
```

```
                 NZ X3,RDCRD
                 SX6 0
                 SA6 CMAP
                 EQ RDCRD
                 SPACE 1
        FINIS    SA1 CMAP            .ALL CONTROL CARDS SCANNED. INITIALIZE
                 ZR X1,XIFDBG        .   RECOVERY IF POSSIBLE.
                 SB3 B0
                 SB1 XRECVR
                 SX4 1
                 RJ =XSETUP.
                 EQ XIFDBG
                 SPACE 1
                 USE /XRVC/
        XREF     DATA 0
        CMAP     DATA 1
                 USE *
        BKSP     DATA 40B
        READ     DATA 10B
        FUNC     BSSZ 1
        MAP      DATA 3RMAP
        FTN      DATA 3RFTN
        DBG      DATA 5RDEBUG
        OFF      DATA 3ROFF
                 END
                 IDENT  OVERLOD
                 ENTRY  OVERLOD
                 TITLE  O V E R L O D 3 . 0
        ***
        *
        *
        *        OVERLOD3.0-     RANDOM FILE OVERLAY LOADER
        *
        *               AUTHOR-   D.C. JESSEN
        *                             NORTHWESTERN UNIVERSITY
        *
        *               DATE- 11/5/70
        *
        *               OVERLOD3.0   LOADS OVERLAY GENREATED BY THE
        *
        *               CP PROGRAM PRELOAD- WHICH TAKES OVERLAY AND PRELOADS
        *
        *               THEM ONTO A PANDOM FILE WITH AN INDEX
        *
        *               OVERLOD3.0 THEN READS THE INDEX AND LOADS
        *
        *               THE OVERLAYS FROM THE SPECIFIED FILE
        *
        *               FORMAT OF OVERLOD CALL IS:
        *
        *                                 RJ     =XLOADER
        *                            +    VFD    60/POINTER
        *
        *               WHERE POINTER IS THE FWA OF A PARAMTER LIST
        *
        *                         POINTER VFD    42/7LLFN    ,18/0
        *                                 VFD    6/L1,6/L2,6/0,1/1,41/0
        *                                 DATA   0
        *               WHERE LFN IS THE FILE NAME FROM WHICH THE OVERLAY
        *
        *               IS TO LOADED FROM
        *
        *
        *               L1 = PRIMARY OVERLAY LEVEL
        *
        *               L2 = SECONDARY OVERLAY LEVEL NUMBER
```

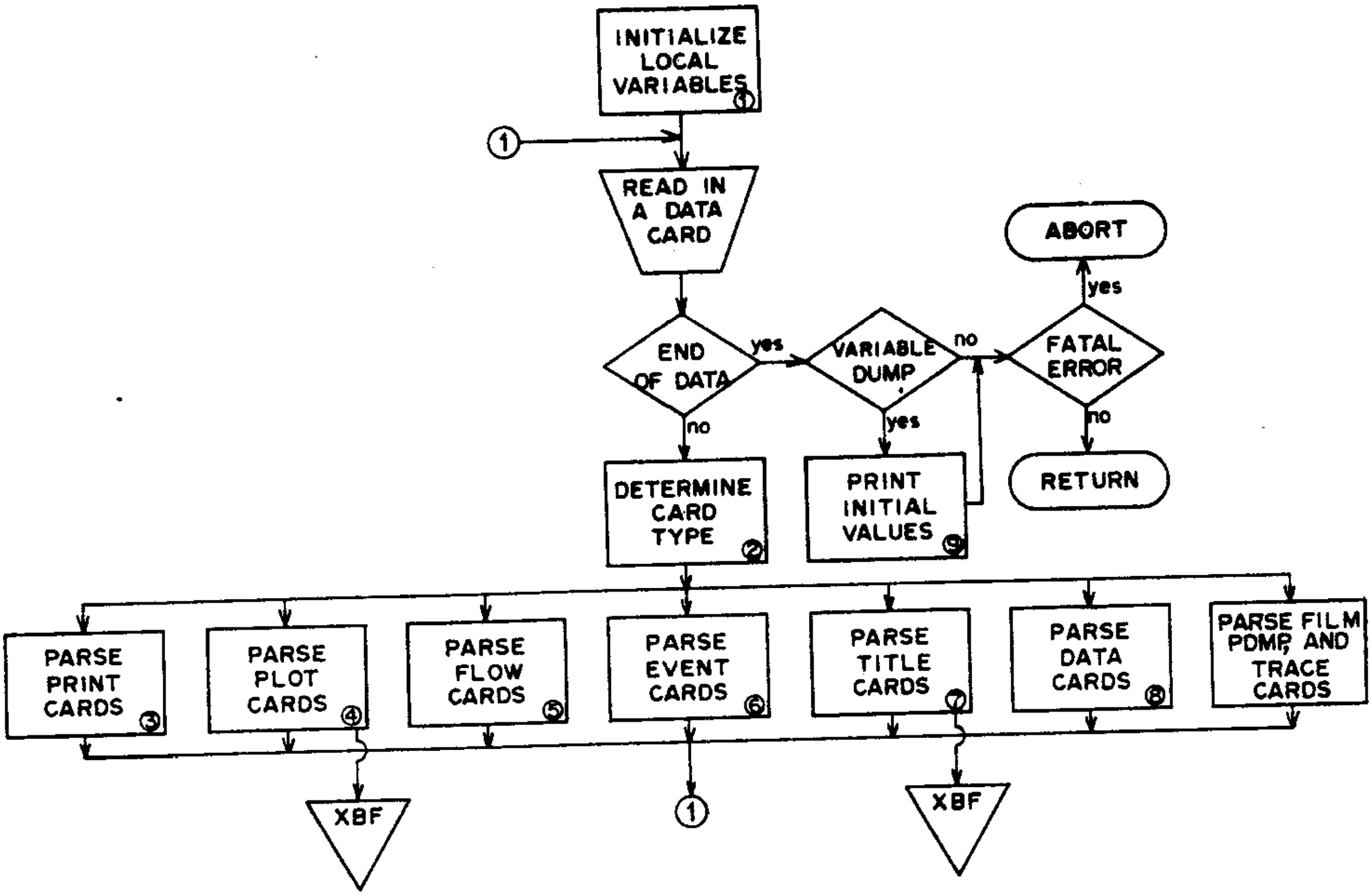
```
*
*                    THE FIRST WORD OF THE PARAMTER LIST IS ZEROED WHEN THE
*
*                    LOADING IS COMPLETE AND THE ENTRY POINT OF THE
*
*                    OVERLAY LOADED IS IN THE LOWER 18 BITS OF THE
*
*                    SECOND WORD.
*
*
*                    IF ERROR IN LOADING HAS OCCURED BIT  36 IS SET IN
*
*                    IN THE SECOND WORD OF THE PARAMTER LIST
*
*                    SEVERAL OVERLAYS MAY BE LOADED AT ONCE
*
*                    BY HAVING PAIRS OF LOADER DIRECTIVES FOLLOWING
*
*                    EACH OTHER WITH THE LAST PAIR OF LOADER DIRECTIVES
*
*                    FOLLOWED BY A ZERO WORD
*
*
***
 INDXLN   EQU    64              INDEX LENGTH
          TITLE  INITIALIZATION SECTION
          SPACE  4
**
*
*                    GET PARAMTERS FROM CALLING PROGRAM
*
*                    FORMAT OF CALL TO OVERLOD IS
*
*                              RJ     OVERLOD
*                         +    VFD    60/POINTER
*
*                    WHERE POINTER IS THE ADDRESS
*
*                    OF A TWO WORD COMMUNICATION AREA
*
*                    WORD ONE CONTAINS THE FILE NAME
*
*                    FROM WHICH THE OVERLAY IS TO BE LOADED
*
*                    WORD TWO CONTAINS THE OVERLAY LEVELS THE OVERLAY
*
*                    LOADING FLAG
*
*
**
          SPACE  4
 OVERLOD  DATA   0
          SA1    OVERLOD            GET RETURN ADDRESS
          SB1    1                INITIALIZE B1
          LX1    30               RIGHT JUSTIFY RETURN ADDRESS
          SB7    X1+B1            SET B7 TO RETURN ADDRESS +1
          SA2    X1               GET PARAMETER POINTER
 OVERAGN  MX3    42
          SA1    X2               GET FIRST PARAMETER
          SB6    X2               SAVE FWA OF PARAMETER LIST
          BX3    -X3*X1           MASK OUT FILE NAME
          SA2    A1+B1            GET SECOND PARAMETER
          ZR     X1,ERR1          ERROR NO. 1- ZERO FILE NAME
          NZ     X3,ERR1          ERROR NO. 1- NON-ZERO SL LIST POINTER
          LX2    59-OVLBIT        CHECK OVERLAY BIT
          NG     X2,LOAD          SET - GO LOAD OVERLAY
```

```
                 EQ     ERR5           NOT SET ERROR NO. 5
                 SPACE  4
      **
      *
      *                 CALL MEM TO GET THE CURRENT FIELD
      *
      *                 LENGTH ABD PLACE THIS VALUE IN THE
      *
      *                 LIMIT OF THE INPUT FET
      *
      *
      **
                 SPACE  4
       LOAD      SA4    CALLMEM
                 SX7    B0
                 BX6    X4
                 SA7    MEMSTAT
                 RJ     MTR
                 SA4    MEMSTAT
                 AX4    30
                 SX6    X4-1
                 SA6    ILIMIT         SET LIMIT IN FET AS FL-1
                 SA1    B6             GET FIRST PARAMTER AGAIN
                 SPACE  4
      **
      *
      *                 GET FILE NAME AND THE OPEN THE FILE
      *
      *                 IF NECESSARY, BRING THE INDEX INTO CORE
      *
      *
      *
      **
                 SPACE  4
                 SA3    RANFILE        GET FET FILE NAME
                 MX0    42
                 BX3    X0*X3
                 IX3    X3-X1          SEE IF SAME AS LAST OVERLAY FILE LOADED
                 ZR     X3,OVL01       ZERO- SKIP OPENING OF FILE
                 SX6    160B           OPEN,ALTER,RECALL
                 BX6    X6+X1
                 SA6    A3             PLACE IN FET WITH FILE NAME
                 SA1    CALLOPE        GET OPE CALL WORD
                 BX6    X1
                 RJ     MTR            ISSUE MTR REQUEST
                 SA3    A3+B1          GET FIRST OF FET
                 MX0    1              SEE IF RANDOM BIT STILL SET
                 LX0    48
                 BX3    X0*X3
                 ZR     X3,ERR2        ERROR NO. 2 - FILE NOT RANDOM
                 TITLE  OVERLAY SEARCHING AND LOADING SECTION.
                 SPACE  4
      **
      *
      *                 GET REQUESTED OVERLAY LEVEL NO. FROM
      *
      *                 THE COMMUNICATION AREA AND SEARCH
      *
      *                 THE INDEX FOR A MATCH
      *
      *
      **
                 SPACE  4
       OVL01     SA2    B6+B1          GET SECOND PARAMTERS
                 MX0    12
```

```
                BX2    X0*X2          MASK OUT LEVEL NUMBERS
                SB5    INDXFWA        GET FWA OF INDEX
                SB4    INDXFWA+64
       OVL02    SA4    B5             GET WORD FROM INDEX
                BX5    X0*X4          MASK OUT LEVEL
                IX6    X5-X2
                ZR     X6,OVL03       FOUND IN INDEX
                ZR     X4,ERR3        ZERO INDEX ENTRY.... ERROR
                SB5    B5+B1          ADD ONE TO WORD COUNT
                EQ     B5,B4,ERR3     ERROR...NOT IN INDEX---LIMIT
                EQ     OVL02          LOOP
                SPACE  4
       **
       *
       *               GET FWA OF OVERLAY FROM THE INDEX INFORMATION
       *
       *               THE INDEX FORMAT IS :
       *
       *               L1L2 XXXX XXYY ZZZZ ZZZZ
       *
       *                              L1 = PRIMARY OVERLAY LEVEL NO.
       *                              L2 = SECONDARY OVERLAY LEVEL NO.
       *                          XXXXXX = FWA - 1 OF OVERLAY
       *                        ZZZZZZZZ = PRU ORDINAL
       *
       *
       **
                SPACE  4
       OVL03    MX0    18             GET FWA OF OVERLAY
                LX0    48
                BX6    X0*X4
                AX6    30             RIGHT JUSTIFY
                SA6    IIN            PLACE IN *IN* OF FET
                SA6    IOUT           AND *OUT*
                MX0    20             MASK OFF PRU ORDINAL
                LX0    20
                BX6    X0*X4
                SA6    RINFO          PUT IN RETURN INFORMATION OF FET
                SX6    12B            READ,BINARY,RECALL
                SA1    RANFILE        GET FILE NMAE
                MX0    42             MASK OUT CODE AND STATUS
                BX1    X0*X1
                BX6    X6+X1          ADD FUNCTION CODE
                SA6    A1             PLACE BACK IN FET
                SA1    CALLCIO        CALL CIO TO READ
                BX6    X1
                RJ     MTR            ISSUE MTR REQUEST
                SPACE  4
       **
       *
       *               CHECK FOR EOR OR EOF STATUS
       *
       *               IF NO EOR OR EOF STATUS THEN THE ENTIRE
       *
       *               OVERLAY COULDNT BE LOADED
       *
       *               SET FATAL ERROR FLAGS
       *
       *
       **
                SPACE  4
                SA1    RANFILE        GET CODE AND STATUS
                MX0    1
                LX0    5
                BX1     X0*X1
                ZR      X1,OVL04        NO EOR OR EOF FLAG SET
                SPACE   4
      **
      *
      *
      *                 GET ENTRY POINT FOR THIS OVERLAY
      *
      *                 LEVEL AND PLACE IN LAST 18 BITS OF
      *
      *                 THE SECOND WORD OF THE COMMUNICATION AREA
      *
      *
      *
      **
                SPACE   4
                SA1     IOUT            GET FIRST WORD OF OVERLAY
                SA1     X1              GET ENTRY ADDRESS
                MX0     42              MASK OFF ENTRY ADDRESS
                BX1     -X0*X1
                SA2     A2              GET 2ND PARAMTER WORD
                BX6     X2+X1           ADD ENTRY ADDRESS
                SA6     A2              RE-WRITE
                SA1     IIN
                SX6     X1                  LWA LOADED + 1
                SA1     65B             GET THE LAST LOADED ADDRESS
                MX0     42              ADD PLACE IT IN RA+65
                BX1     X0*X1
                BX6     X6+X1
                SA6     A1              REPLACE WORD 65 WITH LWA LOADED
                EQ      EXIT
                SPACE   4
      **
      *
      *                 ON ERROR THE FATAL ERROR IN OVERLAY
      *
      *                 LOADING BIT IS SET IN THE SECOND WORD
      *
      *                 OF THE COMMUNICATION AREA
      *
      *
      *
      **
                SPACE   4
       OVL04    SX6     B1              SET ERROR FLAG IN
                LX6     FTLBIT          SECOND PARAMTER
                SA6     B6+B1           WRITE
                EQ      ERR4
       EXIT     SX7     B7-B7           CLEAR X7
                SA7     B6              CLEAR FIRST PARAMTER
                SB3     B1+B1
                SX2     B6+B3           SET TO NEXT PARAMETER
                SA1     X2
                NZ      X1,OVERAGN
                JP      B7              RETURN
       MTR      DATA    0
                SA1     B1
                NZ      X1,MTR+1        LOOP
                SA6     B1
                SA1     B1              WAIT FOR MTR TO ACCEPT
                NZ      X1,*
                EQ      MTR             RETURN
       ERR1     SX6     ER1
                EQ      MSG
       ERR2     SX6     ER2
                EQ      MSG
       ERR3     SX6     ER3
                EQ      MSG
```

```
ERR4      SX6    ER4
          EQ     MSG
MSG       SA1    CALLMSG
          BX6    X6+X1
          RJ     MTR
          SX6    B1              SET FATAL BIT
          LX6    FTLBIT
          SA6    B6+B1           WRITE
          EQ     EXIT
CALLMSG   VFD    18/3LMSG,42/0
CALLCIO   VFD    18/3LCIO,2/1,40/RANFILE
CALLOPE   VFD    18/3LOPE,2/1,40/RANFILE
ER1       DIS    ,*FATAL ERROR-ZERO FILE NAME OF NON ZERO SL*
ER2       DIS    ,*OVERLAY FILE NOT PRELOADED*
ER3       DIS    ,*OVERLAY LEVEL NOT FOUND ON FILE*
ER4       DIS    ,*FATAL ERROR- FIELD LENGTH TOO SMALL*
ERR5      SX6    ER5
          EQ     MSG
ER5       DIS    ,*OVERLOD3.0 CANT HANDLE SL LOADS*
CALLMEM   VFD 18/3LMEM,2/1,40/MEMSTAT
MEMSTAT   DATA   0
RANFILE   VFD    60/0
IFIRST    VFD    12/0,1/1,11/0,18/3,18/FIRST
IIN       VFD    42/0,18/FIRST
IOUT      VFD    42/0,18/FIRST
ILIMIT    VFD    42/0,18/FIRST+1
          DATA   0
RINFO     DATA   0
          VFD    30/0,12/INDXLN,18/INDXFWA
INDXFWA   BSS    INDXLN
FIRST     BSS    2
FTLBIT    EQU    36
OVLBIT    EQU    41
LDRDONE   EQU    29
          END
```

CHAPTER 2. PROCESS DATA



Overview

The data section supplied by the user is processed card by card.

Inputs to this area generate the following output:

(1) PRINT.<list> causes the generation of a print stack (XPRT) containing the list of variables to be printed.

(2) PLØT.<list> generates a plot stack (XPLT) and a plot file (XBF) containing the list of variables to be plotted.

(3) FLØW.<list> sets the low order bit of each entry in the flow stack (XFLWT) corresponding to each flow defined in <list>.

(4) EVENT.<name>,<time>,<priority> enters the event in the event scheduler.

(5) TITLE.<list> places <list> on file XBF.

(6) FILM. turns on a flag (XFLPR) signaling that plots are to be printed onto microfilm.

(7) <data assignments> assign values to variables. These values are stored at their proper location in blank common (XADRS).

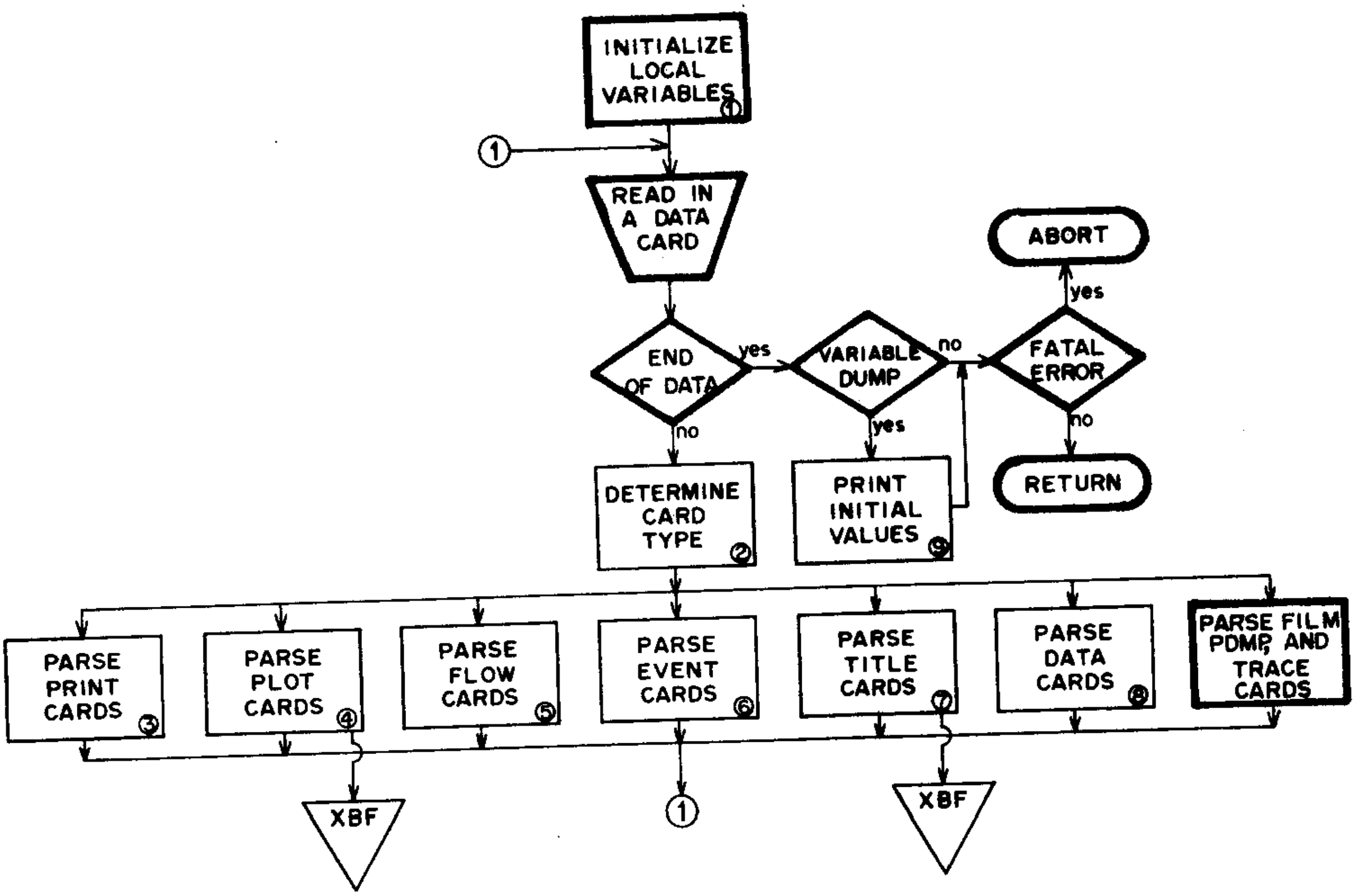
(8) NØNE. inhibits printing of initial values of user declared variables.

(9) ALL. initiates printing of initial values of first and second class variables.

(10) TRACE. turns on flag (XTRACE) signaling the periodic dump of all events in the event scheduler.

The preceding flow diagram illustrates the sections of this chapter which are explained individually.

2.1. *Initialization and Card Reading*

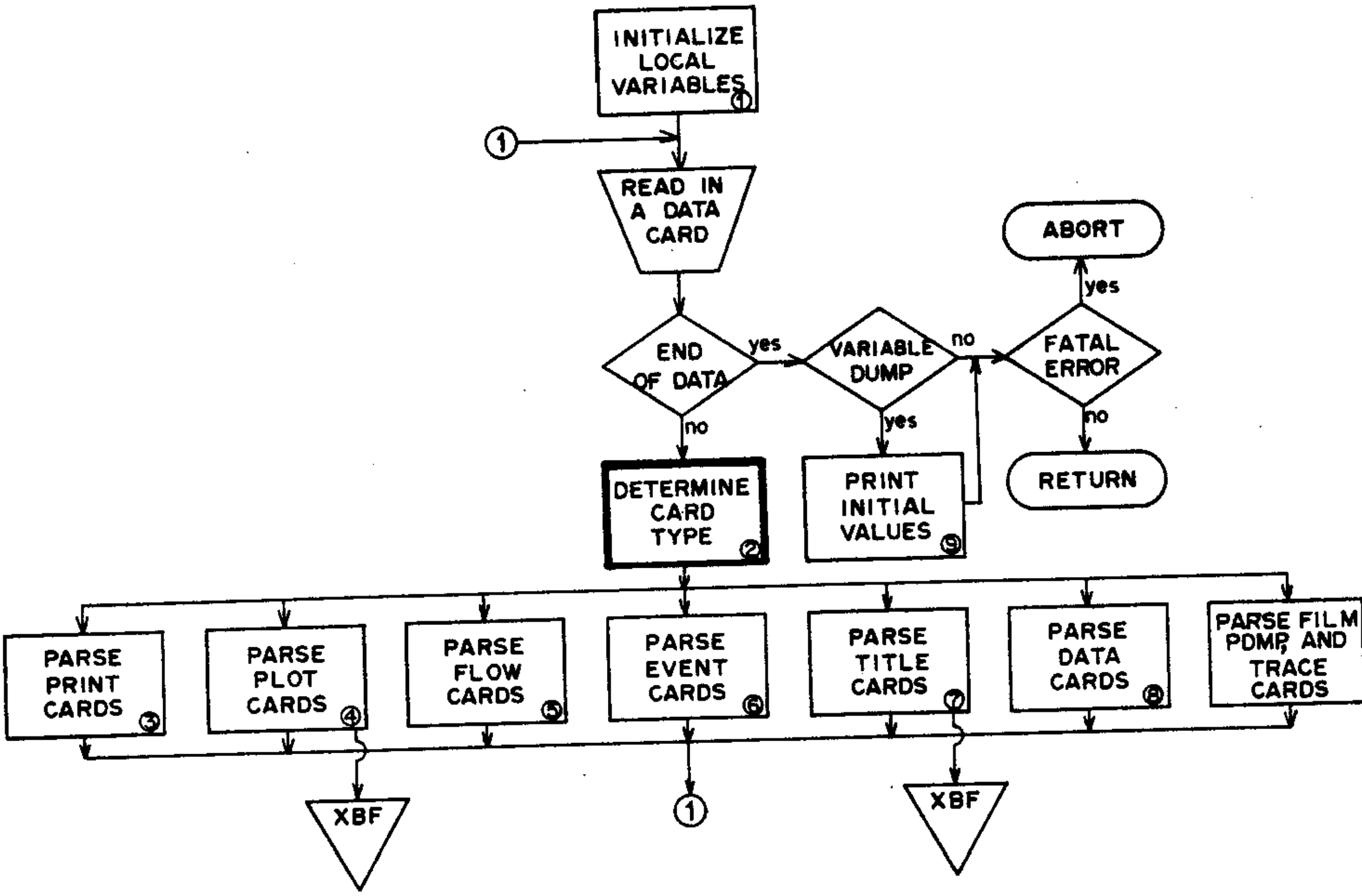


```
      OVERLAY(NEWT1,1,2)
      PROGRAM XINPUT
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON /XXPLOT/XPLT(100),XRNG(2)
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      DIMENSION XCARD(8)
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON XADRS(1)
      INTEGER XNPR,XPRT,XNPL,XPLT,XNW,XI,XJ,XUI,XTYPE,XICOL,XPDMP
      LOGICAL XFILM,XPLFG,XFLPR,XTRACE,QPLTS
C
C.....THIS ROUTINE PROCESSES THE DATA SECTION.  AFTER THE TYPE OF CARD
C.....HAS BEEN DETERMINED BY ROUTINE "XCRDTP" CONTROL IS PASSED TO THE
C.....APPROPRIATE ROUTINE.
C
C.....SET ALL USER-DECLARED VARIABLES TO INDEFINITE.
C
      CALL REMARK (18H   PROCESSING DATA)
      XPDMP=0
      XFILM=.FALSE.
      XPLFG=.FALSE.
      XFLPR=.FALSE.
      QPLTS=.FALSE.
      XNPL=1
      XPLT(1)=1000002B
      XRNG(1)=0.
      XRNG(2)=XRNG(1)
      XNPR=0
      DO 20 XI=2,XNW
   20 XADRS(XI)=17770000000000000000B.O.XI
C
C.....READ IN AND PROCESS EACH CARD OF DATA SECTION.
C
   25 CONTINUE
      READ (XUI,85) XCARD
      IF (EOF(XUI)) 80,30,80
   30 CALL XCRDTP (XCARD,XTYPE,XICOL)
      GO TO (35,40,45,50,55,60,65,70,73,75), XTYPE
C
C.....<PRINT.>...
C
   35 CALL XPRSTK (XCARD,XICOL)
      GO TO 25
C
C.....<PLOT.>...
C
   40 IF (QPLTS) GO TO 25
      CALL XPLSTK (XCARD,XICOL,QPLTS)
      GO TO 25
C
C.....<FLOW.>...
C
   45 CALL XFLSTK (XCARD,XICOL)
      GO TO 25
C
C.....<EVENT.>...
C
   50 CALL XEVSTK (XCARD,XICOL)
      GO TO 25
C
C.....+TITLE.>
C
   55 IF (QPLTS) GO TO 25
      CALL XTITLE (XCARD,XICOL)
      GO TO 25
C
C.....<FILM.>...
C
   60 XFILM=.TRUE.
      GO TO 25
C
C.....<DATA>...
C
   65 CALL XDATA (XCARD,XICOL)
      GO TO 25
C
C.....<NONE.>
C
   70 XPDMP=2
      GO TO 25
C
C.....<ALL.>
C
   73 XPDMP=1
      GO TO 25
C
C.....<TRACE.>
C
   75 XTPACE=.TRUE.
      GO TO 25
C
C.....PRINT INITIAL CONDITIONS IF REQUESTED.
C
   80 CALL XPRDMP (XPDMP)
C
   85 FORMAT (8A10)
C
      END
```

Line Number	Explanation
68018-68030	Initialize variables: XPDMP controls the printing of initial values. XFILM indicates that microfilm output is to be generated. XPLFG indicates that plots are to be generated. XFLPR indicates that flow values are to be printed. QPLTS is the logical flag set in XPLTSTK when plot variable stack exceeds 100 variables. All later plots will be ignored. XNPL is the total number of variables in the plot stack. XPLT(1) is initialized to contain the first variable in the plot stack, TIME. XNPR is the total number of variables in the print stack. XRNG is set in XPLTSTK and used in XCSTART to determine optimum DTPL. XADRS will contain the values of variables from the variable stack. It is initially set to indefinite with the index stored in the low order bits.
68034-68038	A data card is read in, its type is determined by XCRDTP, and the appropriate section processes the card.
68042-68043	A PRINT. card is encountered. XPRSTK enters variables encountered on card into the print stack.
68047-68049	A PLØT. card is encountered. Variables are entered into plot variable stack and file XBF by XPLSTK. (Unless plot variable stacks, capacity is exceeded; in this case QPLTS=.TRUE. and all following plot cards are ignored.)
68053-68054	A FLØW. card is encountered. XFLSTK sets low order bit in XFLWT of flows following FLØW. command.
68058-68059	An exogenous EVENT. card is encountered. XEVSTK enters name, time, and priority of event into exogenous event list.
68063-68065	A TITLE. card encountered. The remainder of the data card following the command will be saved and printed as a title on the first plot request following.
68069-68070	FILM. card is encountered. Set flag XFILM which will direct the plotted output onto microfilm.

Line Number	Explanation
68074-68075	Any card with a unrecognizable command in columns 1-6 is assumed to be a data assignment statement. XDATA stores the values of data assignments into their proper locations in blank common.
68079-68080	A NØNE. card is encountered. XPDMP=2 allows the listing of initial values of system control variables and inhibits printing of user declared variables (in XPRDMP)
68084-68085	An ALL. card is encountered. XPDMP=1 enables XPRDMP to print all user declared variables (first and second class storage).
68089-68090	A TRACE. command is encountered. TRACE=.TRUE. enables a dump of the event stack each time the contents of the stack change.
68094	(Enter from 68036). After data cards have been processed, XPRDMP directs the printing of initial values of user declared variables.

2.2. *Determine Card Type*



Overview

This section scans a data card and determines the type of data card encountered (assigns the card a type number, the value of variable KTYPE).

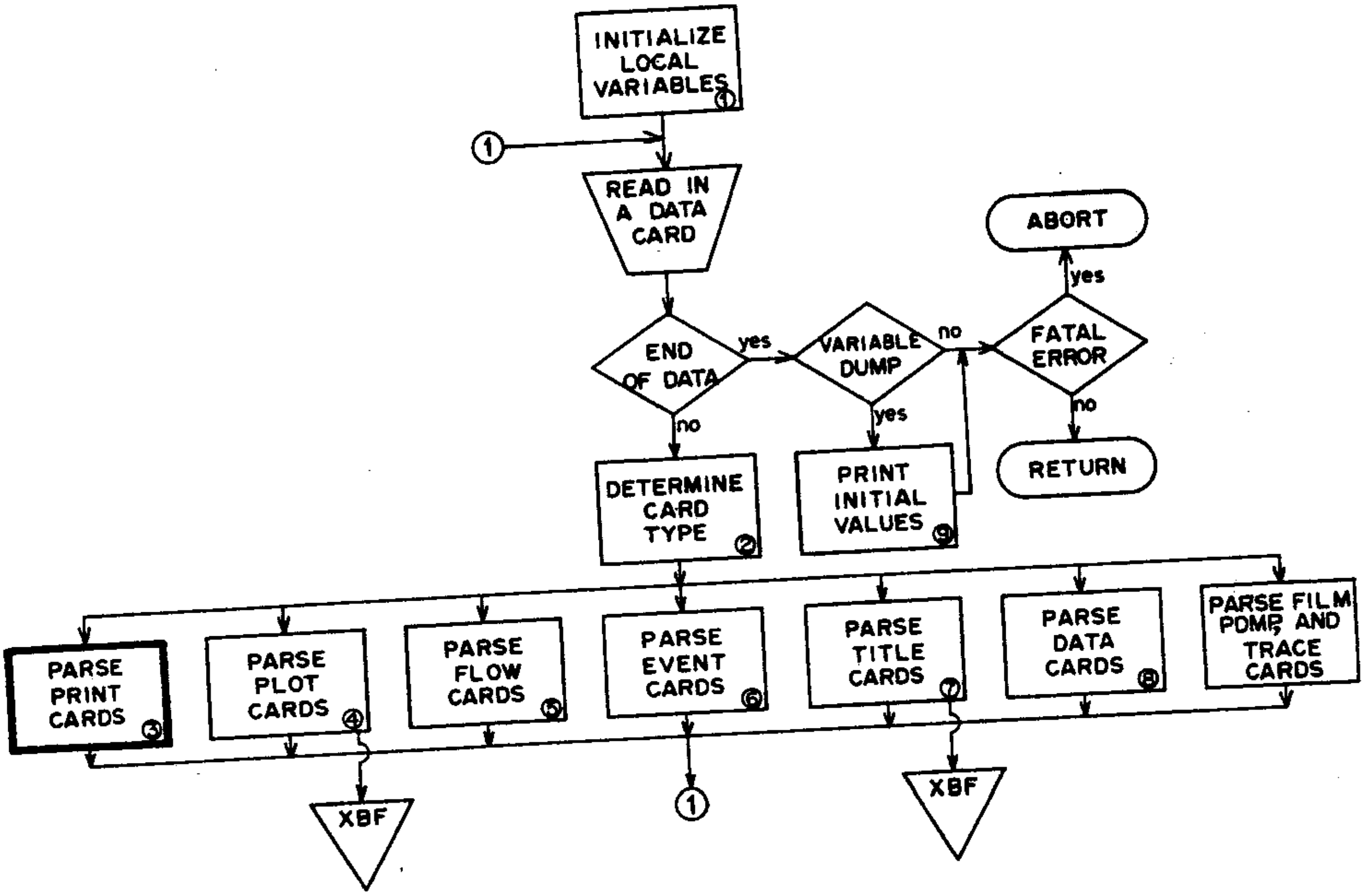
Data Section Card Format	KTYPE
PRINT.<list>	1
PLØT.<list>	2
FLØW.<list>	3
EVENT.<name>,<time>	4
TITLE.<list>	5
FILM.	6
<data assignments>	7
NØNE.	8
ALL.	9
TRACE.	10

```
69000            SUBROUTINE XCRDTP (CARD,KTYPE,JCOL)
69001            DIMENSION CARD(8), KEY(9)
69002            DATA KEY/6HPRINT.,5 PLOT.,5HFLOW.,6HEVENT.,6HTITLE.,5HFILM.,5HNONE
69003           1.,4HALL.,6HTRACE./
69004            DATA NK/9/
69005      C
69006      C.....THIS ROUTINE DETERMINES WHAT TYPE OF DATA SECTION CARD HAS BEEN
69007      C.....ENCOUNTERED AND SETS "KTYPE" TO AN APPROPRIATE VALUE:
69008      C
69009      C.....DATA SECTION CARD FORMAT                      KTYPE
69010      C       PRINT. <LIST>                                1
69011      C       PLOT. <LIST>                                 2
69012      C       FLOW. <LIST>                                 3
69013      C       EVENT. <NAME>,<TIME>                         4
69014      C       TITLE. <LIST>                                5
69015      C       FILM.                                        6
69016      C       <DATA ASSIGNMENTS>                           7
69017      C       NONE.                                        8
69018      C       ALL.                                         9
69019      C       TRACE.                                      10
69020      C
69021            ICOL=0
69022            NCOL=0
69023            NAME=10H
69024      C
69025      C.....SEARCH THE CARD, COLUMN BY COLUMN, TO FIND A MATCH WITH A KEY WORD
69026      C
69027         15 ICOL=ICOL+1
69028            IF (ICOL.GT.80) GO TO 30
69029            CALL GCHARS (CARD,ICOL,1,ICHR)
69030            IF (ICHR.EQ.1H ) GO TO 15
69031            NCOL=NCOL+1
69032            IF (NCOL.GT.6) GO TO 30
69033            CALL SCHARS (NAME,NCOL,1,ICHR)
69034            IF (NCOL.LT.5) GO TO 15
69035            DO 20 I=1,NK
69036               IF (NAME.EQ.KEY(I)) GO TO 25
69037         20 CONTINUE
69038            GO TO 15
69039      C
69040      C.....A MATCH WAS FOUND.
69041      C
69042         25 KTYPE=I
69043            IF (KTYPE.GE.7) KTYPE=KTYPE+1
69044            JCOL=ICOL
69045            RETURN
69046      C
69047      C.....NO MATCH FOUND, ASSUMED A DATA ASSIGNMENT CARD.
69048      C
69049         30 KTYPE=7
69050            JCOL=0
69051            RETURN
69052      C
69053            END
```

Line Number	Explanation
69000	XCARD=CARD contains a user data card. XTYPE=KTYPE is the card type determined by XCRDTP. XICØL=JCØL is the column number following the command portion of the data card (PRINT., PLØT., etc.).

Line Number	Explanation
69001-69023	KEY contains the list of possible data commands. NK is the number of elements in KEY. ICØL is the current column number being searched. NCØL is the number of nonblank characters in NAME. NAME contains the first five or six nonblank characters of the data card.
69027-69038	The card is search column by column. NAME is filled and checked against KEY. A match with KEY indicates that the type of data card has been determined.
69042-69045	A match is found and KTYPE is assigned a value corresponding to the matched position in KEY. JCØL is the column following the data command.
69049-69053	If a match with KEY is not found, the card is assumed to be a data assignment card.

2.3. Parse PRINT. Cards



Overview

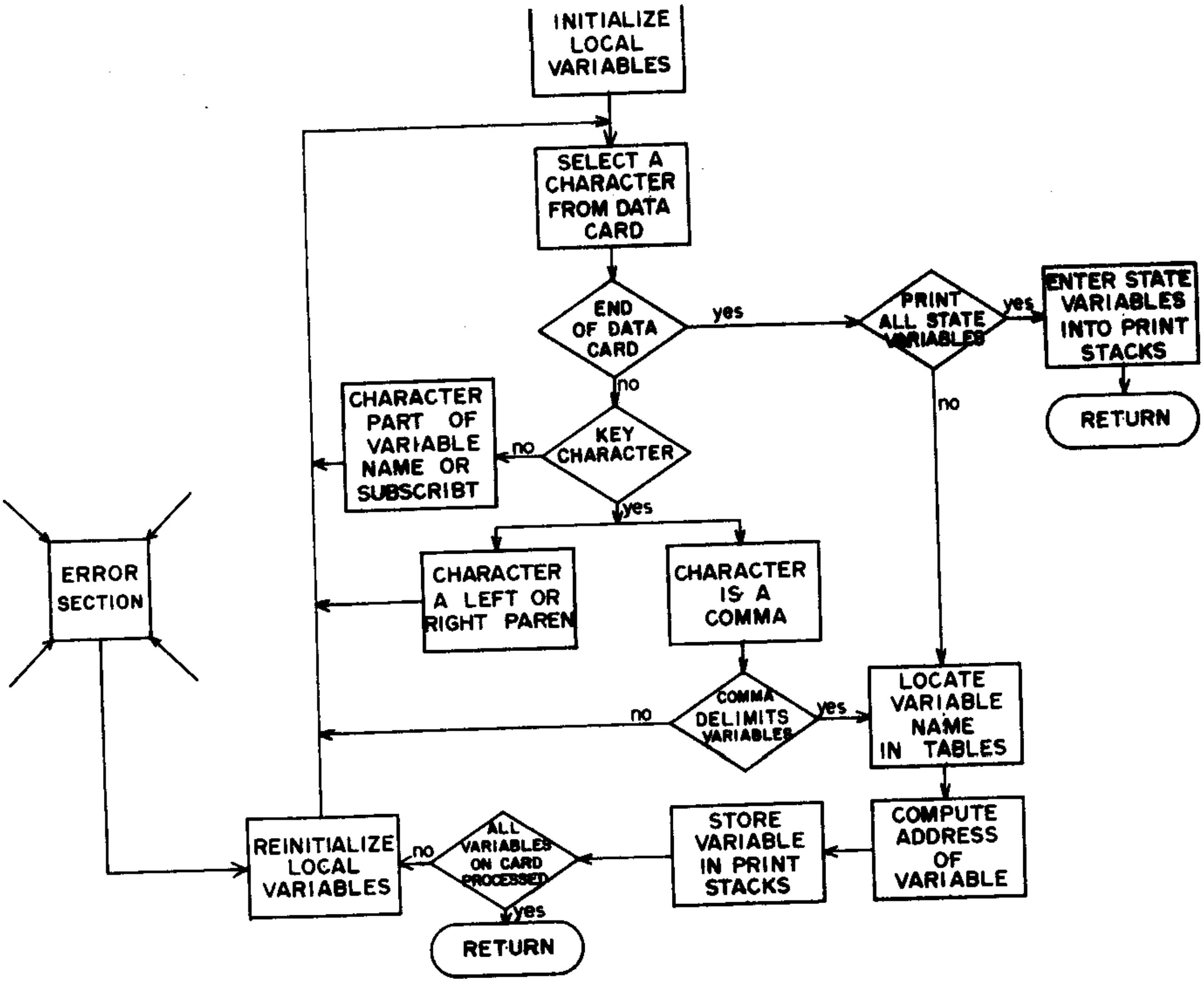
This section parses data cards of the form:

PRINT. <var1>,<var2>, •••

Each variable on the card is entered into a print stack (XPRT), which contains the list of variables to be printed onto the output file for each predetermined time increment. Thus, the input to the section is the text portion of the data card (<var1>,<var2>, •••), and the output consists of generated entries in the print stack, XPRT. XNPR is a counter of the number of variables in XPRT.

The section is subdivided for easier analysis.

PRINT. cards flow chart


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
no
PRINT ALL STATE VARIABLES
yes
no
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
KEY CHARACTER
no
yes
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
STORE VARIABLE IN PRINT STACKS
ALL VARIABLES ON CARD PROCESSED
no
yes
REINITIALIZE LOCAL VARIABLES
RETURN

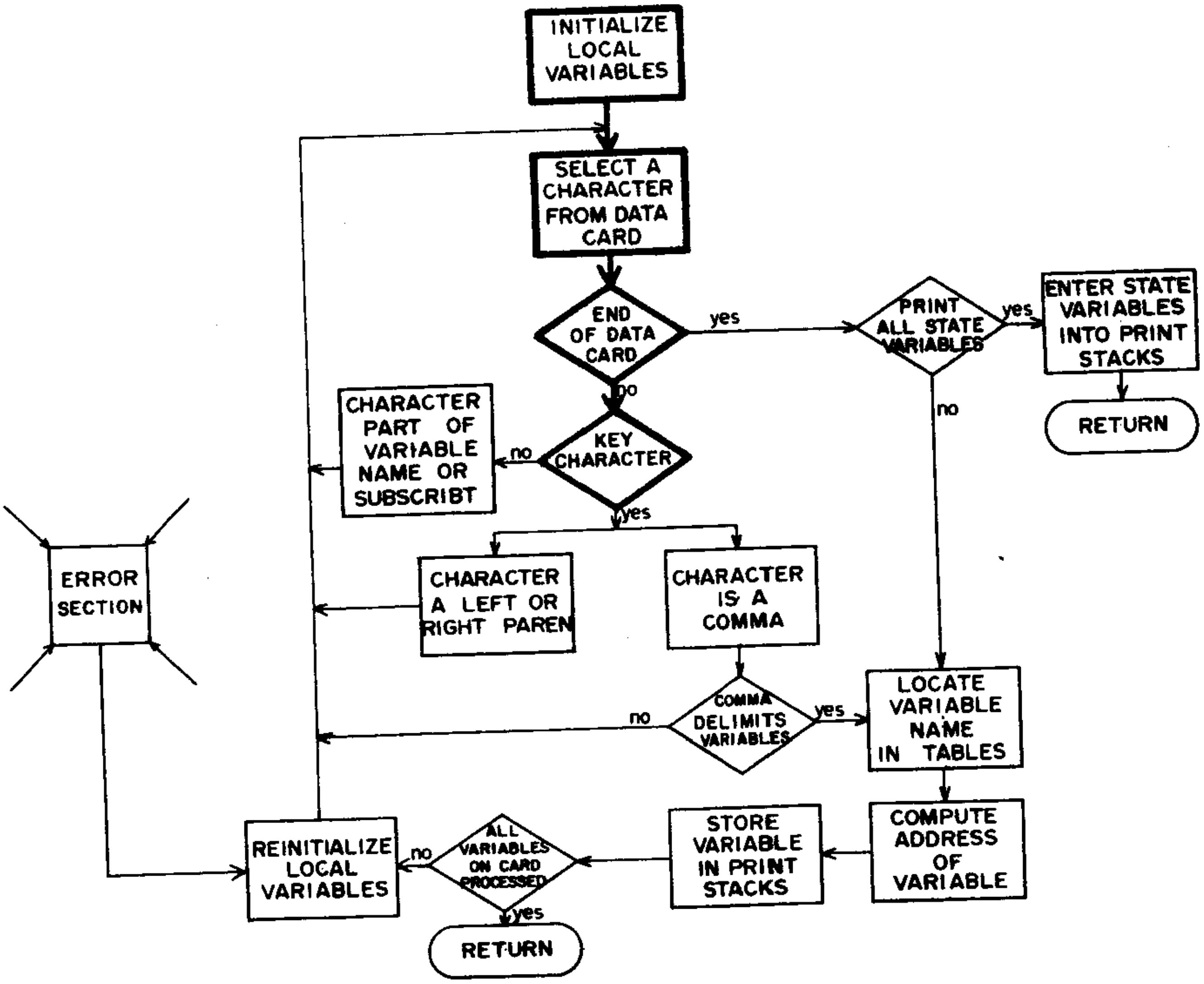
```
      SUBROUTINE XPRSTK (CARD,JCOL)
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON /XXFL1WS/ XNFLW,XFLWT(1)
      COMMON /XXFL2WS/ XFLW(1)
      DIMENSION CARD(8), MCOL(3), MSUB(3), KEY(3), N1(3), N2(3)
      INTEGER XNV,XVT1,XVT2,XNPR,XPRT,XUO,XUI,XNFLW,XFLWT
      LOGICAL STOP,ALL
C
C.....THIS ROUTINE PARSES <PRINT.> REQUEST CARDS, GENERATING AN ENTRY IN
C.....THE PRINT REQUEST STACK FOR EACH VARIABLE IN THE PRINT REQUEST.
C.....VARIABLES REQUESTED FOR PRINT MUST BE PRESENT IN THE VARIABLE
C.....REFERENCE TABLES.
C
C.....FORMAT OF PRINT REQUEST STACK INFORMATION:
C.....LOCATION  (BITS)                  INFORMATION
C     0-9      (10)     INDEX OF STORED VARIABLE NAME IN USER DECLARED
                            VARIABLE REFERNECE TABLES.
C
```

```
C.....10-11     (2)      MODE OF VARIABLE (0=INTEGER, 1=REAL).
C.....12-41    (30)      SUBSCRIPTS OF VARIABLE (3 FIELDS, 10 BITS EACH).
C.....42-59    (18)      INDEX OF VARIABLE RELATIVE TO "XADRS(1)".
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H,,1H(,1H)/,NK/3/
      ICOL=JCOL
      KODE=1
      NCOL=0
      NAME=10H
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      MSUB(3)=10H
      MSUB(2)=MSUB(3)
      MSUB(1)=MSUB(2)
      STOP=.FALSE.
      ALL=.TRUE.
C
C.....SCAN THE PRINT REQUEST, COLUMN BY COLUMN, SEARCHING FOR VARIABLE
C.....NAMES.
C
   15 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 95
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 15
      ALL=.FALSE.
      DO 20 I=1,NK
         IF (ICHR.EQ.KEY(I)) GO TO 35
   20 CONTINUE
      IF (KODE.GE.5) GO TO 120
C
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,30,30,30), KODE
C
C....."ICHR" IS ASSUMED PART OF A VARIABLE NAME.
C
   25 NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 125
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
C
C....."ICHR" IS ASSUMED PART OF A SUBSCRIPT.
C
   30 IF (ICHR.LT.1H0.O.ICHR.GT.1H9) GO TO 120
      J=KODE-1
      MCOL(J)=MCOL(J)+1
      IF (MCOL(J).GT.4) GO TO 130
      CALL SCHARS (MSUB(J),MCOL(J),1,ICHR)
      GO TO 15
C
C....."ICHR" IS A KEY CHARACTER.
C
   35 GO TO (40,85,90), I
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 45
      IF (KODE.EQ.4) GO TO 120
      KODE=KODE+1
      GO TO 15
C
C.....THE COMMA DELIMITS VARIABLES.
C
   45 IF (NCOL.LE.0) GO TO 135
C
C.....LOCATE THE VARIABLE NAME IN THE VARIABLE REFERENCE TABLES.
C
```

```
      DO 50 I=1,XNV
         CALL GCHARS (XVT1(I),1,5,LNM)
         IF (NAME.EQ.LNM) GO TO 55
   50 CONTINUE
      GO TO 140
C
C.....VARIABLE FOUND IN TABLES, RETRIEVE INFORMATION.
C
   55 INDX=I
      CALL GBYTE (XVT1(I),I1,30,18)
      CALL GBYTE (XVT1(I),MODE,48,2)
      N1(1)=1
      DO 60 J=1,2
   60 CALL GBYTE (XVT2(I),N1(J+1),J*10-10,10)
C
C.....DECODE VARIABLE SUBSCRIPTS.
C
      DO 65 J=1,3
         N2(J)=0
         IF (MCOL(J).LE.0) GO TO 65
         CALL GNUM (MSUB(J),1,MCOL(J),N2(J),IERR)
         IF (IERR.NE.0) GO TO 145
   65 CONTINUE
C
C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD
C.....ADDRESS OF THE VARIABLE.
C
      I2=0
      DO 70 J=1,3
         K=4-J
         NN=1
         IF (N2(K).GT.0) NN=N2(K)
   70 I2=N1(K)*(NN-1+I2)
C
C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD
C.....ADDRESS OF THE STORAGE BLOCK "XADRS(1)".
C
      LOC=I1+I2
C
C.....STORE INFORMATION IN PRINT STACKS.
C
      XNPR=XNPR+1
      IF (XNPR.GT.200) GO TO 150
      INFO=0
      CALL SBYTE (INFO,INDX,0,10)
      CALL SBYTE (INFO,MODE,10,2)
      DO 75 J=1,3
   75 CALL SBYTE (INFO,N2(J),10*J+2,10)
      CALL SBYTE (INFO,LOC,42,18)
      XPRT(XNPR)=INFO
C
C.....REINITIALIZE LOCAL VARIABLES.
C
   80 IF (STOP) RETURN
      NCOL=0
      NAME=10H
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      MSUB(3)=10H
      MSUB(2)=MSUB(3)
      MSUB(1)=MSUB(2)
      KODE=1
      GO TO 15
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   85 IF (KODE.NE.1) GO TO 120
      KODE=2
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   90 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 120
      KODE=5
      GO TO 15
C
C.....END OF CARD.
C
   95 IF (KODE.GE.2.A.KODE.LE.4) GO TO 155
      IF (ALL) GO TO 100
      STOP=.TRUE.
      GO TO 45
  100 IF (XNFLW.LE.0) RETURN
C
C.....ALL EXISTANT STATE VARIABLES ARE TO BE ENTERED INTO THE PRINT STAC
C
       DO 115 I=1,XNFLW
       DO 115 J=1,2
          CALL GBYTE (XFLWT(I),INDX,J*15+15,15)
          I1=INDX+8
          INFO=4577740000000777777B
          CALL SBYTE (INFO,INDX,12,10)
          CALL SBYTE (INFO,I1,42,18)
          IF (XNPR.LE.0) GO TO 110
          DO 105 K=1,XNPR
             IF (INFO.EQ.XPRT(K)) GO TO 115
  105     CONTINUE
  110     XNPR=XNPR+1
          IF (XNPR.GT.200) GO TO 150
          XPRT(XNPR)=INFO
  115 CONTINUE
      RETURN
C
C.....IF ERRORS OCCURED GENERATE DIAGNOSTIC.
C
  120 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,185) ICHR,ICOL
      GO TO 160
  125 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,190) NAME
      GO TO 160
  130 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,195) MSUB(J),ICOL
      GO TO 160
  135 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,200) ICOL
      GO TO 160
  140 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,205) NAME
      GO TO 160
  145 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,210) MSUB(J)
      GO TO 160
  150 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,215)
      XNPR=XNPR-1
      RETURN
  155 WRITE (XUO,180) (I,I=1,8),CARD
      WRITE (XUO,220) NAME
C
C.....THE FOLLOWING SEGMENT ASSUMES CONTROL WHEN AN ERROR IS ENCOUNTERED
C.....IT SCANS THE DATA CARD FOR THE BEGINNING OF A NEW VARIABLE NAME.
C
  160 MOR=0
      ICOL=ICOL-1
  165 ICOL=ICOL+1
      IF (ICOL.GT.80) RETURN
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 165
      IF (ICHR.NE.1H,) GO TO 170
      MOR=1
      GO TO 165
  170 IF (ICHR.GE.1HA.AND.ICHR.LT.1H0) GO TO 175
```

```
      MOR=0
      GO TO 165
  175 IF (MOR.NE.1) GO TO 165
      ICOL=ICOL-1
      GO TO 80
C
  180 FORMAT (6H0*****, 22HERROR IN PRINT REQUEST,//T20,8I10/T20,8( 10H1
     1234567890)/T20,8A10)
  185 FORMAT (1H0,T14, 11HCHARACTER ",A1, 20H" ILLEGAL IN COLUMN ,I2)
  190 FORMAT (1H0,T14, 10HVARIABLE ",A5, 24H..." LONGER THAN 5 CHARS)
  195 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 35H..." LONGER THAN 4 CHARS IN
     1 COLUMN ,I2)
  200 FORMAT (1H0,T14, 46HZERO LENGTH VARIABLE NAME IN OR BEFORE COLUMN
     1,I2)
  205 FORMAT (1H0,T14, 10HVARIABLE ",A5, 44H" WAS NOT DECLARED IN A <STO
     1RAGE.> STATEMENT)
  210 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
  215 FORMAT (1H0,T14, 43HMORE THAN 200 VARIABLES REQUESTED FOR PRINT)
  220 FORMAT (1H0,T14, 10HVARIABLE ",A5, 41H" WAS NOT COMPLETELY DECLARE
     1D BY CARD END)
C
      END
```

Initialize and retrieve a Character


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
no
KEY CHARACTER
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
yes
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
ERROR SECTION
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
STORE VARIABLE IN PRINT STACKS
ALL VARIABLES ON CARD PROCESSED
no
yes
REINITIALIZE LOCAL VARIABLES
RETURN

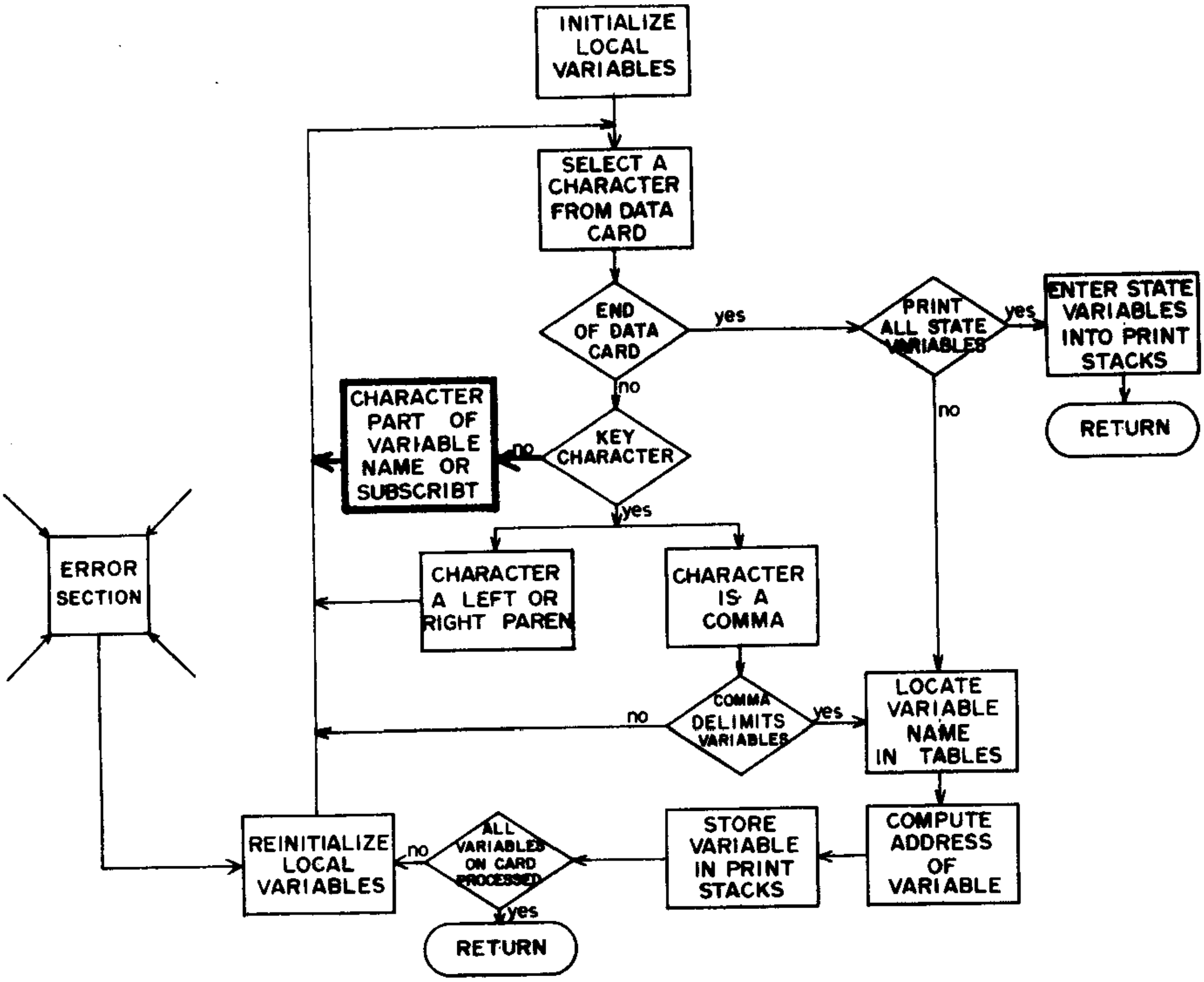
```
70000            SUBROUTINE XPRSTK (CARD,JCOL)
70001            COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
70002            COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
70003            COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
70004            COMMON /XXVR2FR/ XVT2(1)
70005            COMMON /XXFL1WS/ XNFLW,XFLWT(1)
70006            COMMON /XXFL2WS/ XFLW(1)
70007            DIMENSION CARD(8), MCOL(3), MSUB(3), KEY(3), N1(3), N2(3)
70008            INTEGER XNV,XVT1,XVT2,XNPR,XPRT,XUO,XUI,XNFLW,XFLWT
70009            LOGICAL STOP,ALL
70010      C
70011      C.....THIS ROUTINE PARSES <PRINT.> REQUEST CARDS, GENERATING AN ENTRY IN
70012      C.....THE PRINT REQUEST STACK FOR EACH VARIABLE IN THE PRINT REQUEST.
70013      C.....VARIABLES REQUESTED FOR PRINT MUST BE PRESENT IN THE VARIABLE
70014      C.....REFERENCE TABLES.
70015      C
70016      C.....FORMAT OF PRINT REGUEST STACK INFORMATION:
70017      C.....LOCATION  (BITS)                INFORMATION
70018      C      0-9      (10)     INDEX OF STORED VARIABLE NAME IN USER DECLARED
70019      C                           VARIABLE REFERNECE TABLES.
70020      C
70021      C.....10-11      (2)     MODE OF VARIABLE (0=INTEGER, 1=REAL).
70022      C.....12-41     (30)     SUBSCRIPTS OF VARIABLE (3 FIELDS, 10 BITS EACH).
           C.....42-59     (18)     INDEX OF VARIABLE RELATIVE TO "XADRS(1)".
```

```
70023              C
70024              C.....INITIALIZE LOCAL VARIABLES.
70025              C
70026                    DATA KEY/1H,,1H(,1H)/,NK/3/
70027                    ICOL=JCOL
70028                    KODE=1
70029                    NCOL=0
70030                    NAME=10H
70031                    MCOL(3)=0
70032                    MCOL(2)=MCOL(3)
70033                    MCOL(1)=MCOL(2)
70034                    MSUB(3)=10H
70035                    MSUB(2)=MSUB(3)
70036                    MSUB(1)=MSUB(2)
70037                    STOP=.FALSE.
70038                    ALL=.TRUE.
70039              C
70040              C.....SCAN THE PRINT REQUEST, COLUMN BY COLUMN, SEARCHING FOR VARIABLE
70041              C.....NAMES.
70042              C
70043                 15 ICOL=ICOL+1
70044                    IF (ICOL.GT.80) GO TO 95
70045                    CALL GCHARS (CARD,ICOL,1,ICHR)
70046                    IF (ICHR.EQ.1H ) GO TO 15
70047                    ALL=.FALSE.
70048                    DO 20 I=1,NK
70049                       IF (ICHR.EQ.KEY(I)) GO TO 35
70050                 20 CONTINUE
70051                    IF (KODE.GE.5) GO TO 120
```

Line Number	Explanation
70000-70038	CARD= XCARD is a data card containing a PRINT. COMMAND. JCØL=XICØL points to the column before the beginning of the text portion of the card (the text portion is the variable list following the PRINT. command). Initialize the local variables: KEY contains a list of the key characters that could be encountered in the text portion of the PRINT. card. NK is the number of the elements in KEY. ICØL is the column pointer as the data card is scanned. KØDE is a counter of the key characters encountered within a variable. NCØL is the number of characters in a variable name on the data card. NAME is filled with the NCØL characters of a variable name. MCØL(I) contains the number of characters in the Ith subscript of the variable. MSUB(I) will contain the characters of the Ith subscript. STØP is a logical flag indicating that the last variable on the data card has been entered in the variable stack.

Line Number	Explanation
	ALL is a logical flag indicating that all existing state variables are to be entered into the print stack.
70043-70050	The card is scanned for key characters which delimit variables, variable names, and subscripts of variables.
70051	Error has occurred. Previous character was a right paren (KØDE=5) and current character is not a key (delimiter) character.

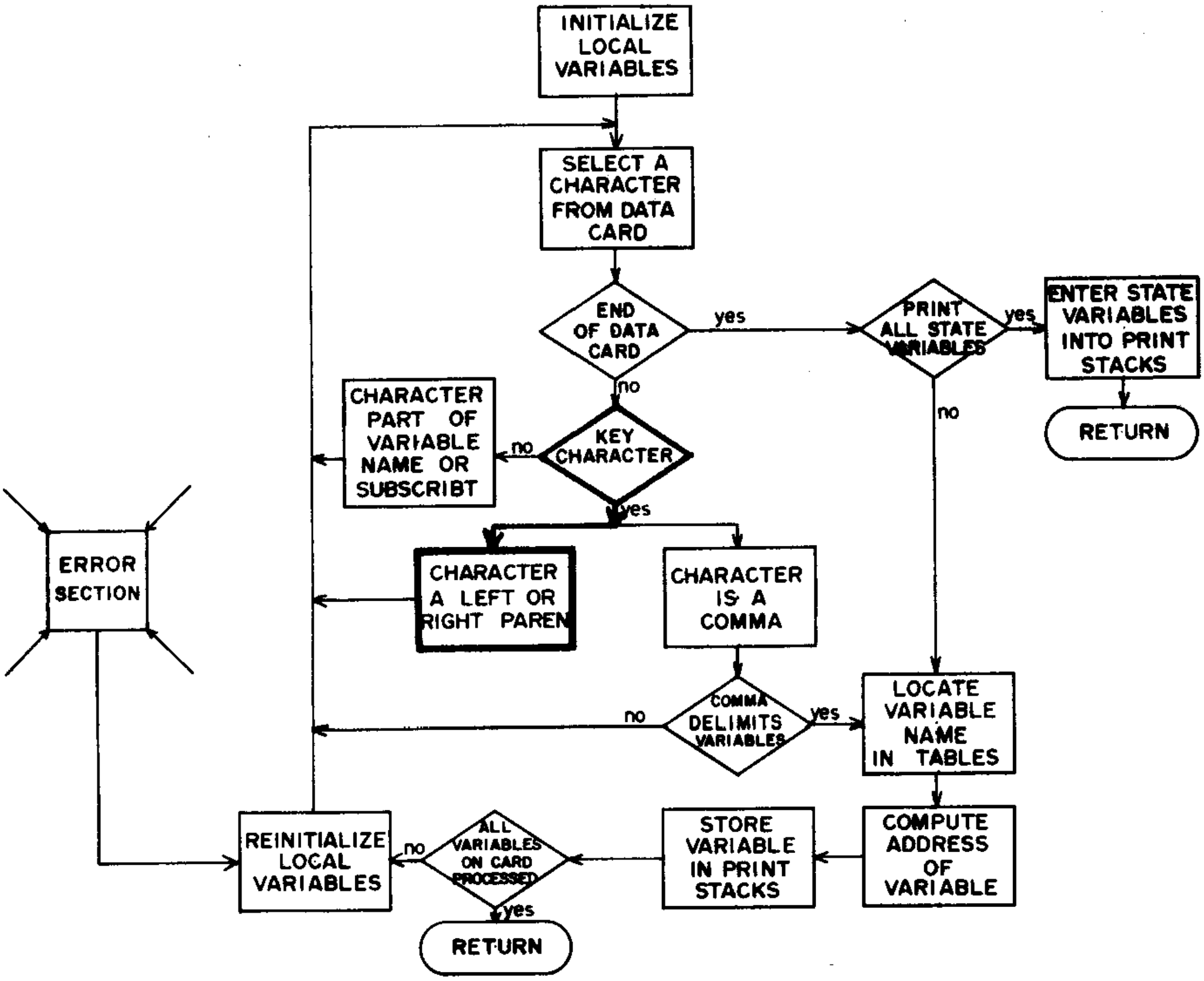
Character is part of a name or subscript


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
no
KEY CHARACTER
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
yes
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
no
COMMA DELIMITS VARIABLES
yes
LOCATE VARIABLE NAME IN TABLES
REINITIALIZE LOCAL VARIABLES
no
ALL VARIABLES ON CARD PROCESSED
STORE VARIABLE IN PRINT STACKS
COMPUTE ADDRESS OF VARIABLE
yes
RETURN

```
C
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,30,30,30), KODE
C
C....."ICHR" IS ASSUMED PART OF A VARIABLE NAME.
C
   25 NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 125
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
C
C....."ICHR" IS ASSUMED PART OF A SUBSCRIPT.
C
   30 IF (ICHR.LT.1H0.O.ICHR.GT.1H9) GO TO 120
      J=KODE-1
      MCOL(J)=MCOL(J)+1
      IF (MCOL(J).GT.4) GO TO 130
      CALL SCHARS (MSUB(J),MCOL(J),1,ICHR)
      GO TO 15
```

Line Number	Explanation
70055	KØDE is a counter of the KEY characters (left paren, right paren, and commas) encountered within a variable. KØDE=1 Initial value. KØDE=2 Left paren encountered. =3 First comma (delimits first and second subscripts) of a variable. =4 Second comma. =5 Right paren encountered. Therefore a value of KØDE greater than 1 indicates that the character must be a part of a subscript.
70059-70062	Since KODE=1, the character is assumed to be a part of a variable name. The character is filled into NAME, with NCØL equaling the number of nonblank characters in NAME. (NAME will contain the name of a variable from data card.)
70066-70071	The character is a part of the subscripts of a variable. J indicates to which subscript the character belongs. MCØL(J) is the number of characters in the Jth subscript. MSUB(J) is filled with the characters of the Jth subscript. EXAMPLE. PRINT. FØX(100,20,3) After processing this card, the variables would contain: NAME=3HFØX NCØL=3 NCØL(1)=3 MCØL(2)=2 MCØL(3)=1 MSUB(1)=3H100 MSUB(2)=2H20 MSUB(3)=1H3

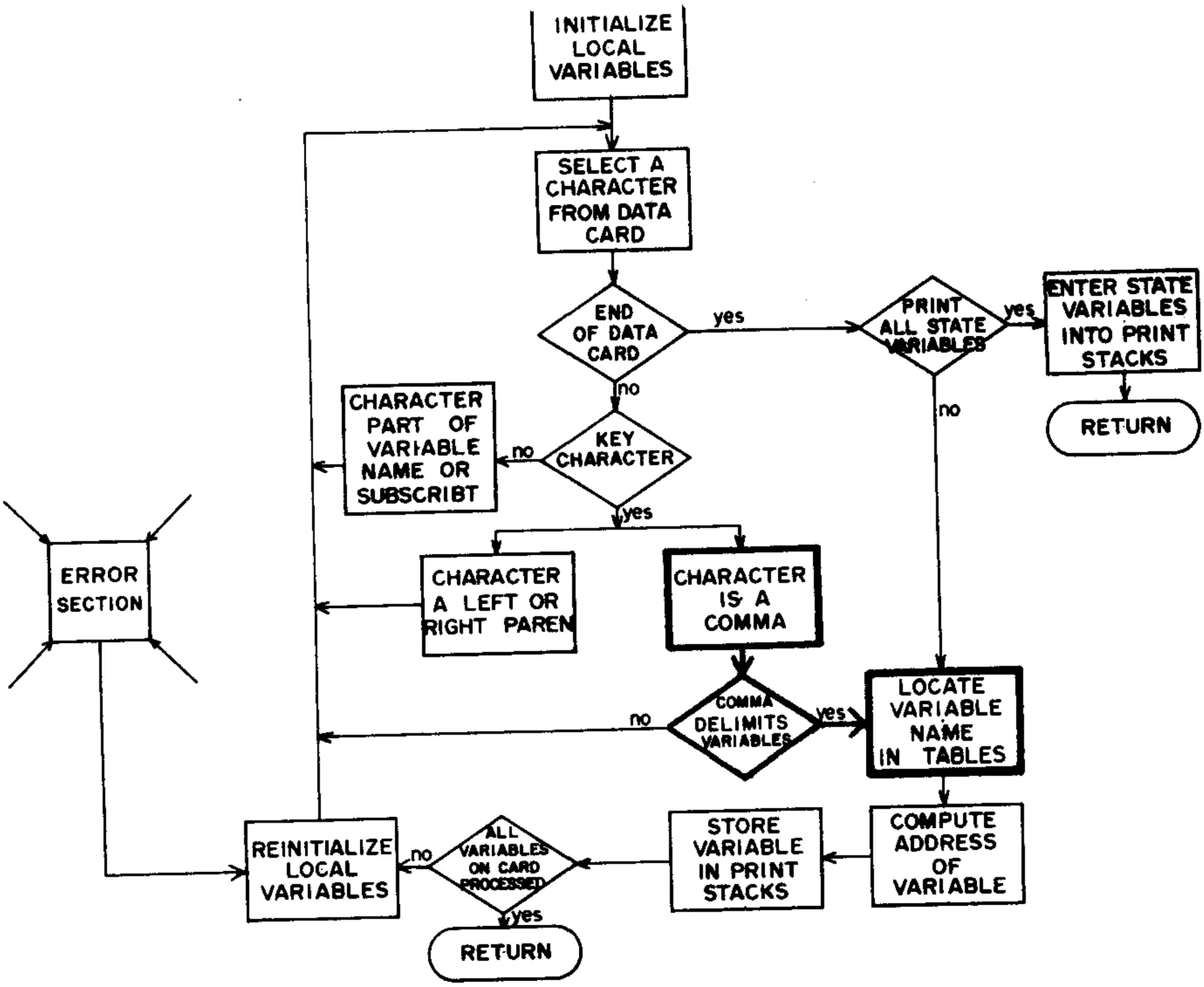
Process subscript delimiter character


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
no
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
no
KEY CHARACTER
yes
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
no
COMMA DELIMITS VARIABLES
yes
LOCATE VARIABLE NAME IN TABLES
REINITIALIZE LOCAL VARIABLES
no
ALL VARIABLES ON CARD PROCESSED
yes
RETURN
STORE VARIABLE IN PRINT STACKS
COMPUTE ADDRESS OF VARIABLE

```
70072            C
70073            C....."ICHR" IS A KEY CHARACTER.
70074            C
70075               35 GO TO (40,85,90), I


70154            C
70155            C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
70156            C
70157               85 IF (KODE.NE.1) GO TO 120
70158                  KODE=2
70159                  GO TO 15
70160            C
70161            C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
70162            C
70163               90 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 120
70164                  KODE=5
70165                  GO TO 15
70166            C
```

Line Number	Explanation
70075	The character is a comma, a left paren, or a right paren (I is determined at line 70049).
70157-70159	If a left paren is encountered, set KØDE=2 and begin looking for characters of first subscript.
70163-70165	If a right paren is encountered, set KØDE=5 and look for comma that delimits variables.

Locate name in reference tables


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
no
KEY CHARACTER
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
yes
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
STORE VARIABLE IN PRINT STACKS
ALL VARIABLES ON CARD PROCESSED
no
yes
REINITIALIZE LOCAL VARIABLES
RETURN

```
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1.O.KODE.EQ.5) GO TO 45
      IF (KODE.EQ.4) GO TO 120
      KODE=KODE+1
      GO TO 15
C
C.....THE COMMA DELIMITS VARIABLES.
C
   45 IF (NCOL.LE.0) GO TO 135
C
C.....LOCATE THE VARIABLE NAME IN THE VARIABLE REFERENCE TABLES.
C
      DO 50 I=1,XNV
         CALL GCHARS (XVT1(I),1,5,LNM)
         IF (NAME.EQ.LNM) GO TO 55
   50 CONTINUE
      GO TO 140
C
C.....VARIABLE FOUND IN TABLES, RETRIEVE INFORMATION.
```

```
70097          C
70098             55 INDX=I
70099                CALL GBYTE (XVT1(I),I1,30,18)
70100                CALL GBYTE (XVT1(I),MODE,48,2)
70101                N1(1)=1
70102                DO 60 J=1,2
70103             60 CALL GBYTE (XVT2(I),N1(J+1),J*10-10,10)
70104          C
70105          C.....DECODE VARIABLE SUBSCRIPTS.
70106          C
70107                DO 65 J=1,3
70108                   N2(J)=0
70109                   IF (MCOL(J).LE.0) GO TO 65
70110                   CALL GNUM (MSUB(J),1,MCOL(J),N2(J),IERR)
70111                   IF (IERR.NE.0) GO TO 145
70112             65 CONTINUE
```

Line Number — Explanation

70079-70082 — A comma is encountered. If KØDE=1, then the variable had no subscripts and the comma delimits variables. If KØDE=5, then the previous character was a right paren and the comma delimits variables. If KØDE=2 or 3, then the comma delimits subscripts; increment KØDE and look for next subscript.

70086-70094 — The comma delimits variables; find the variable name in the reference tables. Search through the variable stack until the variable name stored in NAME is found. (Refer to Section 1.3 for a detailed explanation of XVT1 and XVT2.)

XNV is the number of variables in XVT1.

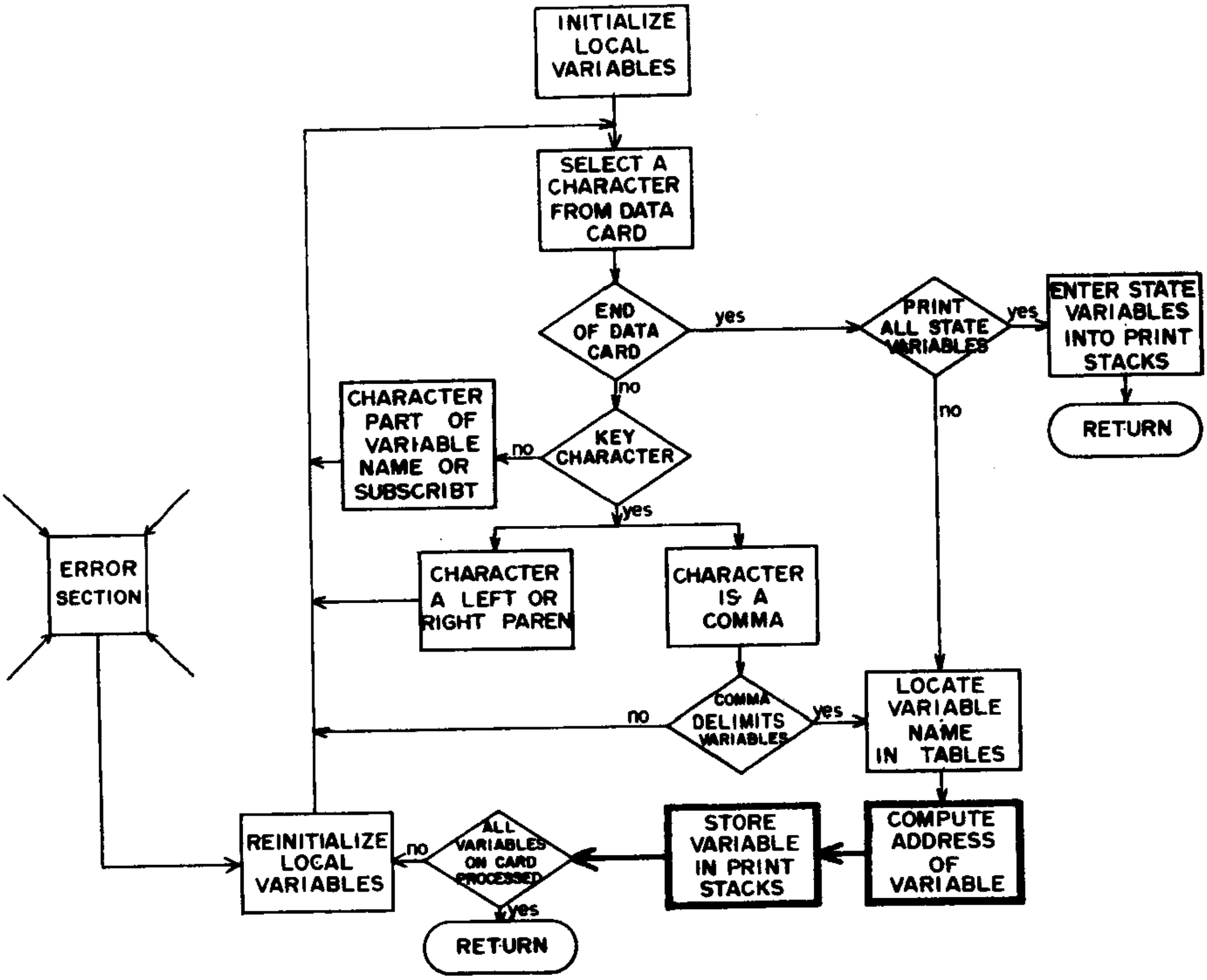
XVT1 is a list of system defined and user declared variables.

> The leftmost five characters contain the name of the variable.
>
> Columns 30-48 contain the starting location of the variable relative to the beginning of blank common (where the values of the variables are stored).
>
> Columns 48-49 contain the type of the variable; type=1 if the variable is of type real, but if type=0, then the variable is an integer.

XVT2 contains the declared subscripts of the variable name in XVT1. The first (leftmost) 10 bits of XVT2 contain the first subscript, the second 10 bits contain the second subscript, and the third subscript is contained in the third series or 10 bits.

Line Number	Explanation
70098-70112	Retrieve information related to matched variable name: INDX is the position of the variable in the stack (table). I1 is the first word address of the variable relative to the beginning of blank common (address of the value of the variable). MØDE is the type of the variable. N1(J) is the (J-1)th subscript of the variable from the tables. N2(J) is the integer value of the Jth subscript of the variable *from the PRINT. card.* NOTE. The subscripts of the variable on the PRINT. card (N2(J)) need not match the subscripts from the tables. The subscripts from the PRINT. card specify the element of the variable name to be printed while the table subscripts specify the total number of elements identified by that name. EXAMPLE. A variable could be in the tables as FØX(25) indicating FØX has 25 locations reserved for it. However, it could appear on a PRINT. card as FØX(3) indicating that the value of the third element of FØX is to be printed onto the output file periodically.

Compute address and store information


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
no
KEY CHARACTER
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
yes
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
no
COMMA DELIMITS VARIABLES
yes
LOCATE VARIABLE NAME IN TABLES
REINITIALIZE LOCAL VARIABLES
no
ALL VARIABLES ON CARD PROCESSED
STORE VARIABLE IN PRINT STACKS
COMPUTE ADDRESS OF VARIABLE
yes
RETURN

```
C
C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD
C.....ADDRESS OF THE VARIABLE.
C
      I2=0
      DO 70 J=1,3
         K=4-J
         NN=1
         IF (N2(K).GT.0) NN=N2(K)
   70 I2=N1(K)*(NN-1+I2)
C
C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD
C.....ADDRESS OF THE STORAGE BLOCK "XADRS(1)".
C
      LOC=I1+I2
C
C.....STORE INFORMATION IN PRINT STACKS.
C
      XNPR=XNPR+1
```

```
70132            IF (XNPR.GT.200) GO TO 150
70133            INFO=0
70134            CALL SBYTE (INFO,INDX,0,10)
70135            CALL SBYTE (INFO,MODE,10,2)
70136            DO 75 J=1,3
70137         75 CALL SBYTE (INFO,N2(J),10*J+2,10)
70138            CALL SBYTE (INFO,LOC,42,18)
70139            XPRT(XNPR)=INFO
```

Line Number — Explanation

70117-70122 — Compute the address of the element of the variable to be printed relative to the first word address of the variable.

EXAMPLE. STØRAGE. CØW(5) was declared by user.
If PRINT. CØW(4) appeared as a data card, then
I2=3 indicating that the address of the value
of CØW(4) is located three locations after the
first word address of CØW (i.e., address of CØW(4)
is the address of CØW(1) + 3).

EXAMPLE. STØRAGE. FØX(15,2) was declared by user.
This is processed by the compiler as:
LVR1(10)=XVT1(10)=0617305555 001760 2 000B
Breaking down LVR1 we have:

```
     0617305555=3HFØX =LNM
     001760    =1008₁₀=I1
     2         =01₂   = MØDE
                10    = INDX
```

LVR2(10)=XVT2(10)=0000001111 0000000010 000000000 ...
Breaking down LVR2 gives

$0000001111_2=15_{10}$=N1(2)
$0000000010_2=2$ =N1(3)

Assume PRINT. FØX(3,1) was the data card.
Then

N2(1)=3   N2(2)=1   N2(3)=0

Therefore,

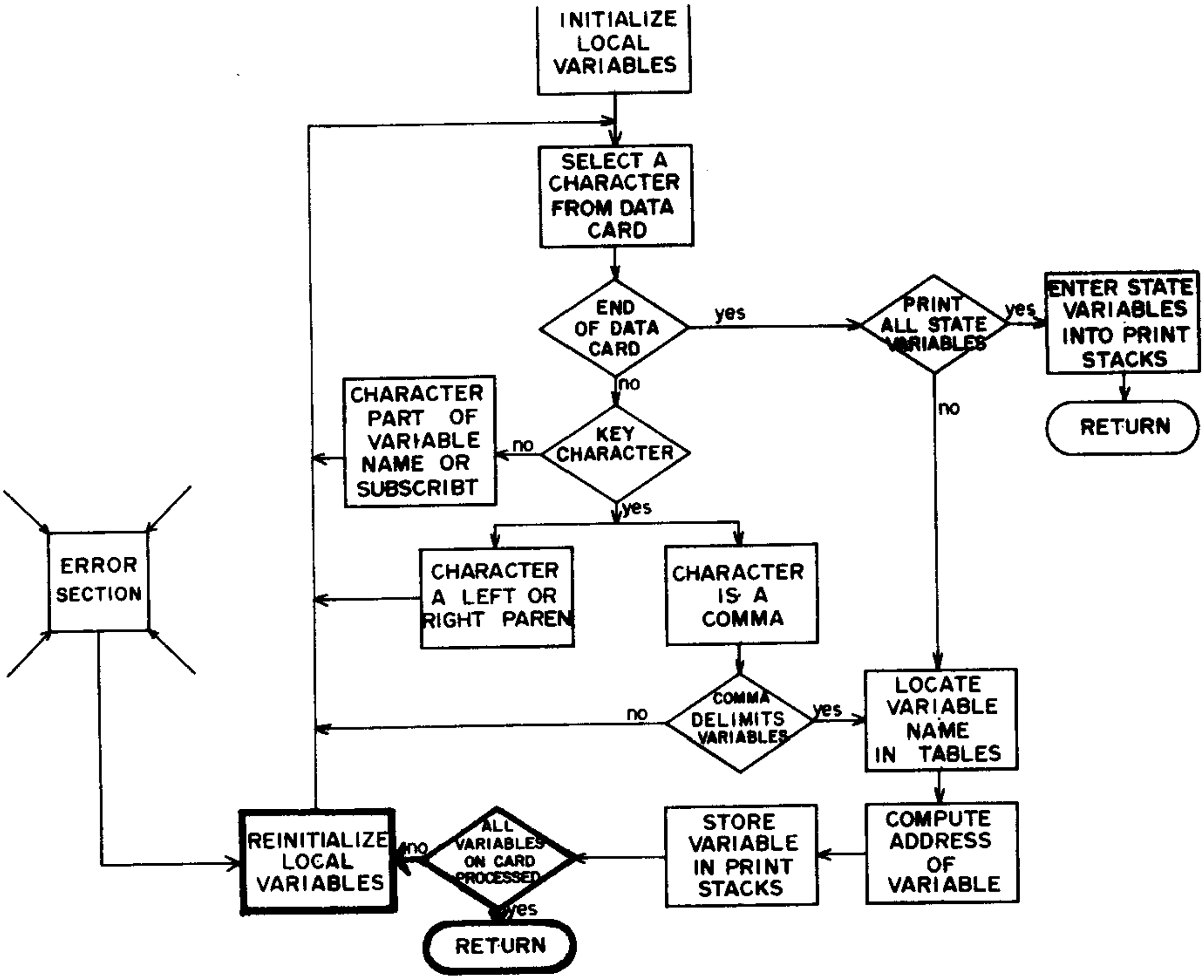
I2=2
LØC=1008+2=1010

70127 — LØC is the address of the variable to be printed, relative to the first word address of the storage block (the beginning of blank common). This address is where the value of the variable to be printed will be stored.

70131-70139 — Store the information in print stack, XPRT.
XNPR represents the total number of variables in the print stack.
Each entry in XPRT has the following format:

Line Number	Explanation
	(1) Bit numbers 0-9 contain the index of the variable in the variable tables (INDX). (2) 10-11 contain the mode of the variable (MØDE). (3) 12-21 contain the first subscript from the PRINT. card (N2(1)). (4) 22-31 contain the second subscript of the variable (N2(2)). (5) 32-41 contain third subscript. (6) 42-59 contain the address of the value of the variable relative to the beginning of blank common (LØC). Therefore PRINT. FØX(3,1) would create the following entry in the print stack. XPRT(XNPR)=0000001010 01 0000000011 0000000001 0 ••• •••0 001111110010_2 The 60 bits of XPRT broken down contain the following values: 0000001010 $=12_8$ =10 =INDX 01 =1 =MØDE 0000000011 =3 =N2(1) (1st subscript) 0000000001 =1 =N2(2) 0000000000 =0 =N2(3) ••• 001111110010=17628=1010_{10}=LØC

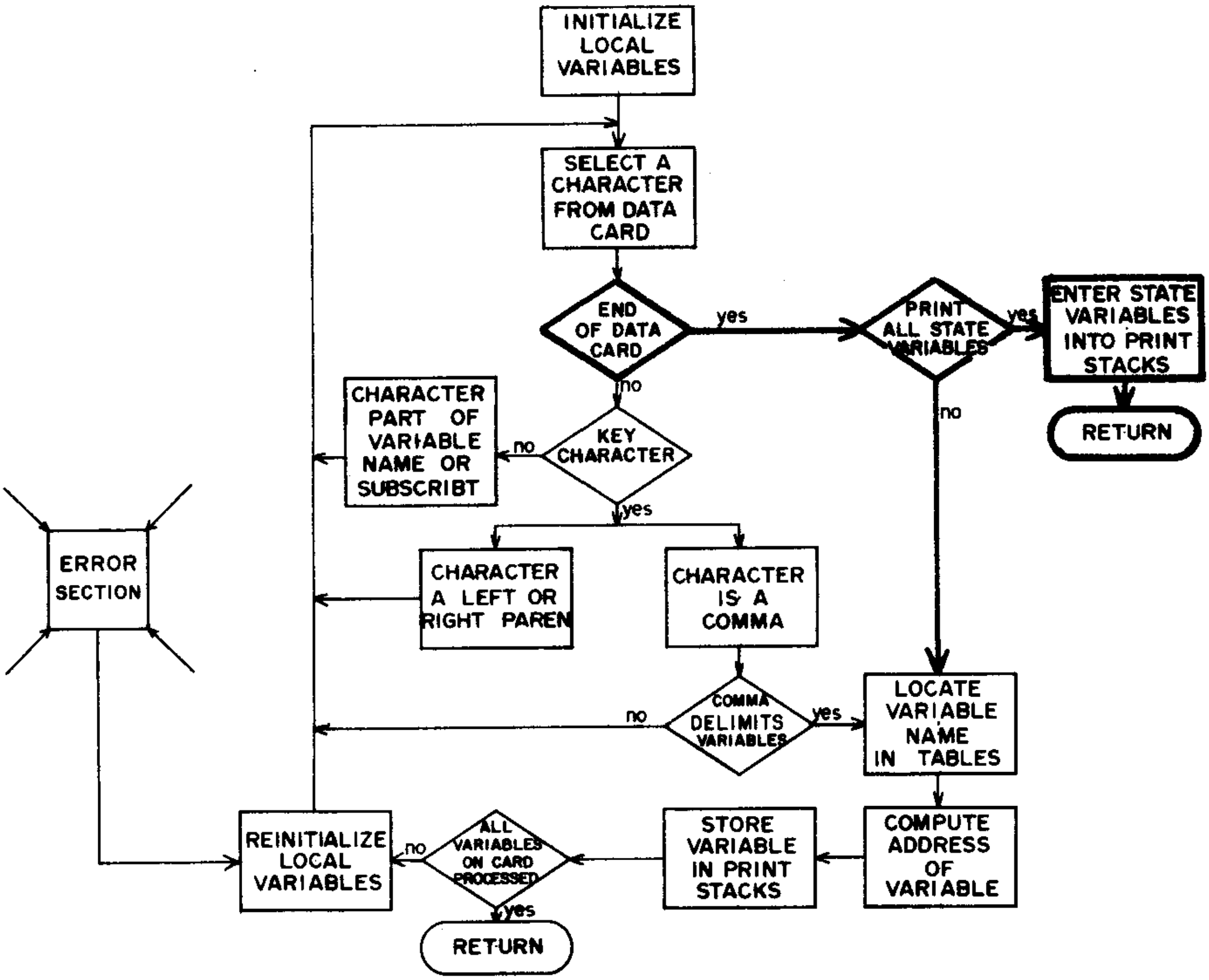
Reinitialize local variables


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
no
PRINT ALL STATE VARIABLES
yes
no
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
no
KEY CHARACTER
yes
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
REINITIALIZE LOCAL VARIABLES
no
ALL VARIABLES ON CARD PROCESSED
yes
RETURN
STORE VARIABLE IN PRINT STACKS
COMPUTE ADDRESS OF VARIABLE

```
C
C.....REINITIALIZE LOCAL VARIABLES.
C
   80 IF (STOP) RETURN
      NCOL=0
      NAME=10H
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      MSUB(3)=10H
      MSUB(2)=MSUB(3)
      MSUB(1)=MSUB(2)
      KODE=1
      GO TO 15
```

Line Number	Explanation
70143-70153	If STØP=.TRUE., then the last variable on the data card has been processed. Otherwise reinitialize local variables and begin to process next variable on data card.

Store state variables in stacks


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
no
PRINT ALL STATE VARIABLES
yes
no
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
KEY CHARACTER
no
yes
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
ERROR SECTION
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
STORE VARIABLE IN PRINT STACKS
ALL VARIABLES ON CARD PROCESSED
no
yes
REINITIALIZE LOCAL VARIABLES
RETURN

```
C.....END OF CARD.
C
   95 IF (KODE.GE.2.A.KODE.LE.4) GO TO 155
      IF (ALL) GO TO 100
      STOP=.TRUE.
      GO TO 45
  100 IF (XNFLW.LE.0) RETURN
C
C.....ALL EXISTANT STATE VARIABLES ARE TO BE ENTERED INTO THE PRINT STAC
C
      DO 115 I=1,XNFLW
      DO 115 J=1,2
         CALL GBYTE (XFLWT(I),INDX,J*15+15,15)
         I1=INDX+8
         INFO=457774000000777777B
         CALL SBYTE (INFO,INDX,12,10)
         CALL SBYTE (INFO,I1,42,18)
         IF (XNPR.LE.0) GO TO 110
         DO 105 K=1,XNPR
            IF (INFO.EQ.XPRT(K)) GO TO 115
  105    CONTINUE
```

```
70188          110    XNPR=XNPR+1
70189                 IF (XNPR.GT.200) GO TO 150
70190                 XPRT(XNPR)=INFO
70191          115 CONTINUE
70192              RETURN
```

Line Number — Explanation

70169

When ICØL>80, (enter from line 70044) then all columns of the data card have been searched. The last variable on each PRINT. card is delimited by the end of the card. Therefore branch and enter the last variable into the print stack.

70170

If ALL=.TRUE., then the PRINT. card contains no variables. This is a request to enter all existing state variables into the print stack.

70173-70187

XNFLW is the number of expanded user defined flows. (Sections 1.5 and 1.8).

XFLWT is the flow table containing XNFLW user defined flows.

INDX contains the index (subscript) of the state variable.

I1 is the address of the state variable (to be printed), relative to the beginning of blank common.

INFØ contains the representation of the state variable before it enters the print stack. Bits 0-11 are a special character signifying that the entry is a state variable. Bits 12-21 are the index of the state variable (INDX). Bits 42-59 are the address, relative to beginning of blank common (I1).

EXAMPLE. Assume the source deck contained the flow command, (2-3).

The compiler would create the following entry in the flow table:

NFLT(N)=XFLWT(N)=200003_8

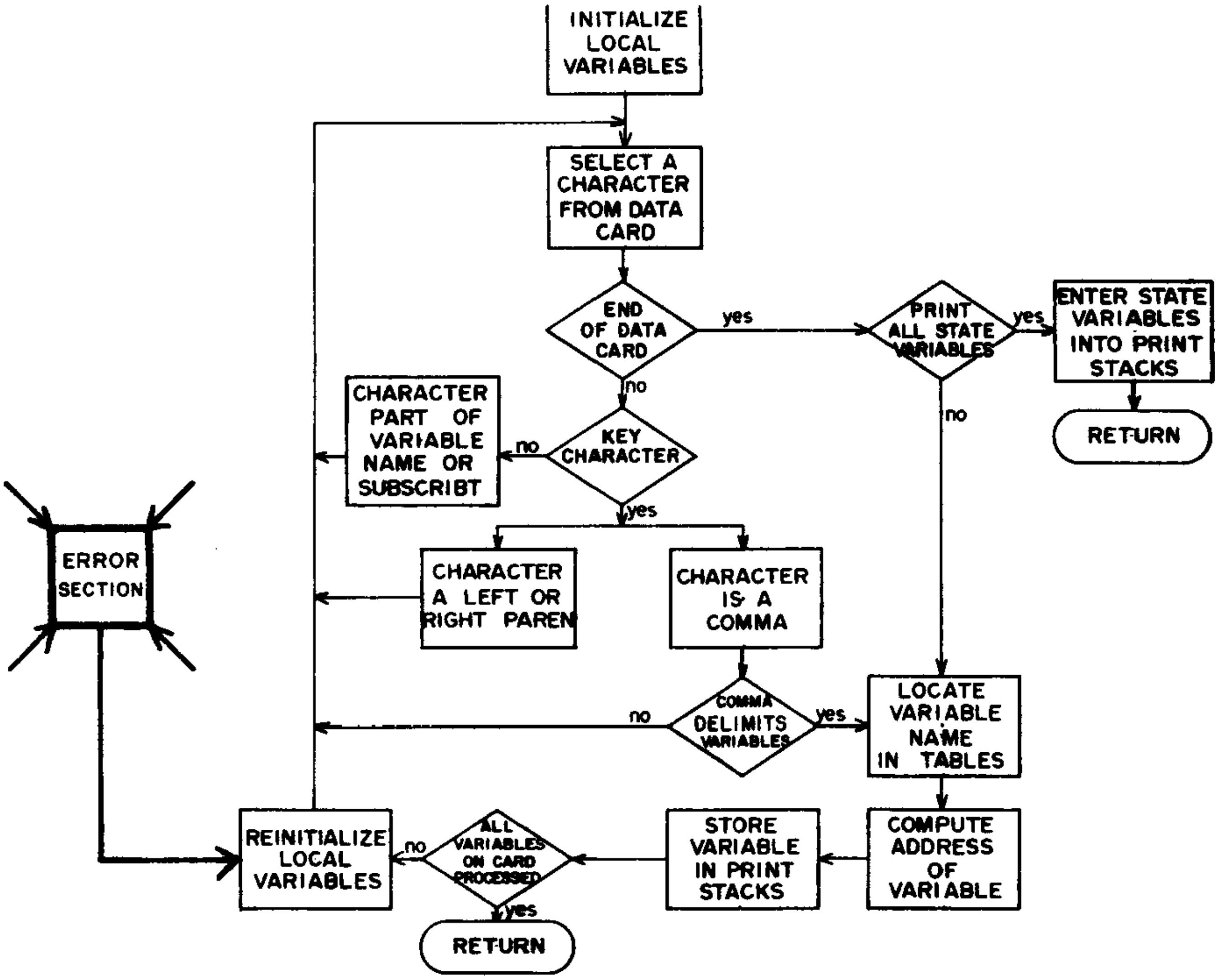
If the card PRINT. appeared in the data section, X(2) and X(3) should be entered into the print stack.

Since each flow table entry specifies two state variables, there are two passes and two entries in the print table (if that particular state variable is not already in the print stack) for each entry in the flow table.

PASS 1 processes the source state variable of each expanded flow from flow stacks.

Line Number	Explanation
	INDX=2 I1=10 (RECALL. There are eight system variables in the variable tables prior to X(1), X(2), ..., X(999)). INFØ=100101111111 0000000010 000 ••• 0001010 100101111111=4577=2HT (special character) 0000000010 =2 =INDX ••• 0001010 =10_{10} =I1
	PASS 2 processes all target state variable subscripts. INDX=3 I1=11=13_8 INFØ=100101111111 0000000011 000 ••• 001011 100101111111=24HT 0000000011 =3 =INDX ••• 0001011 =11_{10}=I1
70184-70192	Each value of INFØ is placed into the print stack unless that state variable is already in the print stack.

Diagnostics and error recovery

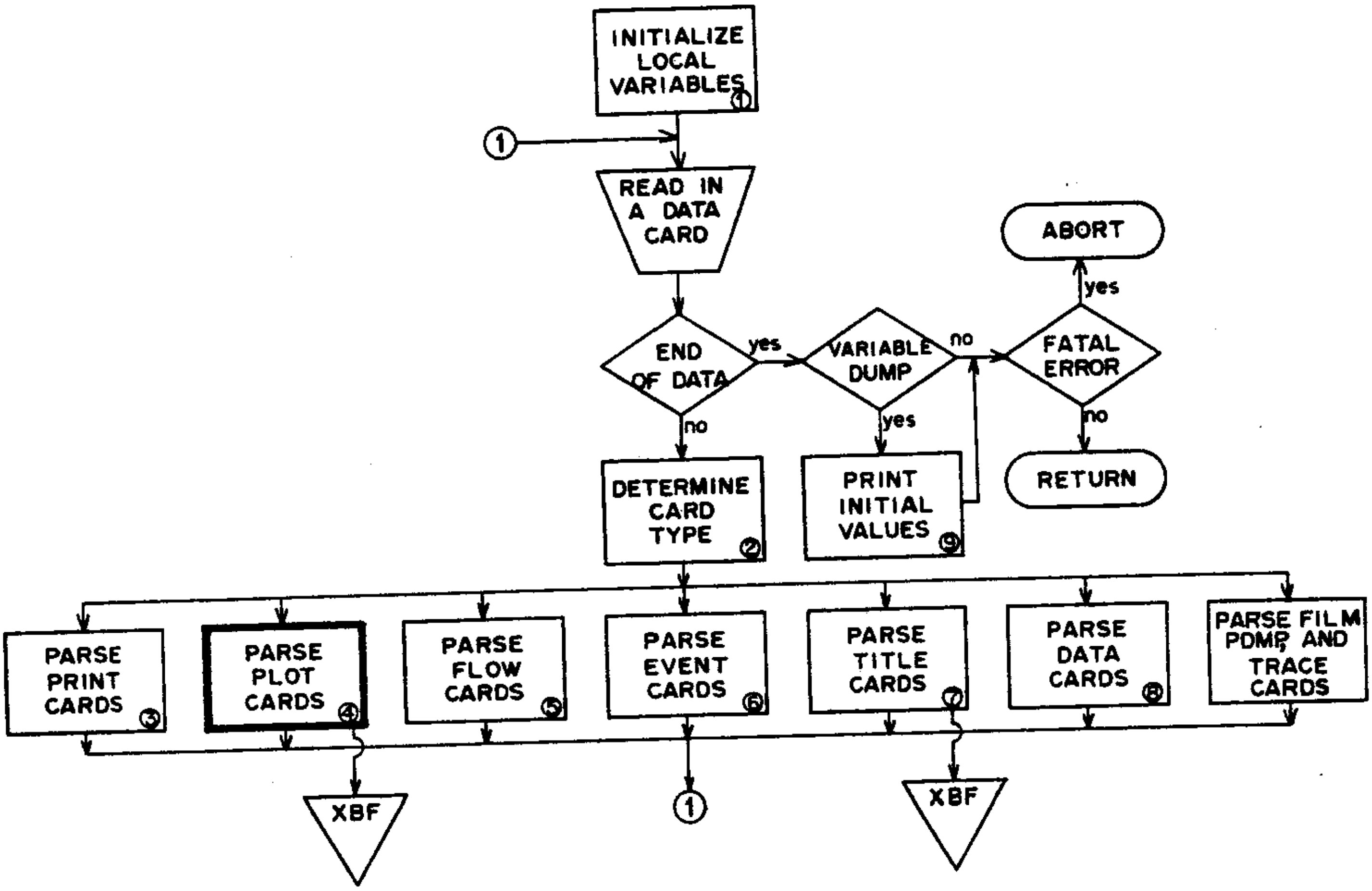

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM DATA CARD
END OF DATA CARD
yes
no
PRINT ALL STATE VARIABLES
yes
ENTER STATE VARIABLES INTO PRINT STACKS
RETURN
no
KEY CHARACTER
no
CHARACTER PART OF VARIABLE NAME OR SUBSCRIBT
yes
CHARACTER A LEFT OR RIGHT PAREN
CHARACTER IS A COMMA
ERROR SECTION
COMMA DELIMITS VARIABLES
no
yes
LOCATE VARIABLE NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
STORE VARIABLE IN PRINT STACKS
ALL VARIABLES ON CARD PROCESSED
no
yes
REINITIALIZE LOCAL VARIABLES
RETURN

```
C
C.....IF ERRORS OCCURED GENERATE DIAGNOSTIC.
C
  120 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,185) ICHR,ICOL
      GO TO 160
  125 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,190) NAME
      GO TO 160
  130 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,195) MSUB(J),ICOL
      GO TO 160
  135 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,200) ICOL
      GO TO 160
  140 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,205) NAME
      GO TO 160
  145 WRITE (XU0,180) (I,I=1,8),CARD
      WRITE (XU0,210) MSUB(J)
      GO TO 160
```

```
70214          150 WRITE (XUO,180) (I,I=1,8),CARD
70215              WRITE (XUO,215)
70216              XNPR=XNPR-1
70217              RETURN
70218          155 WRITE (XUO,180) (I,I=1,8),CARD
70219              WRITE (XUO,220) NAME
70220       C
70221       C.....THE FOLLOWING SEGMENT ASSUMES CONTROL WHEN AN ERROR IS ENCOUNTERED .
70222       C.....IT SCANS THE DATA CARD FOR THE BEGINNING OF A NEW VARIABLE NAME.
70223       C
70224          160 MOR=0
70225              ICOL=ICOL-1
70226          165 ICOL=ICOL+1
70227              IF (ICOL.GT.80) RETURN
70228              CALL GCHARS (CARD,ICOL,1,ICHR)
70229              IF (ICHR.EQ.1H ) GO TO 165
70230              IF (ICHR.NE.1H,) GO TO 170
70231              MOR=1
70232              GO TO 165
70233          170 IF (ICHR.GE.1HA.AND.ICHR.LT.1H0) GO TO 175
70234              MOR=0
70235              GO TO 165
70236          175 IF (MOR.NE.1) GO TO 165
70237              ICOL=ICOL-1
70238              GO TO 80
70239       C
70240          180 FORMAT (6H0*****, 22HERROR IN PRINT REQUEST,//T20,8I10/T20,8( 10H1
70241             1234567890)/T20,8A10)
70242          185 FORMAT (1H0,T14, 11HCHARACTER ",A1, 20H" ILLEGAL IN COLUMN ,I2)
70243          190 FORMAT (1H0,T14, 10HVARIABLE ",A5, 24H..." LONGER THAN 5 CHARS)
70244          195 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 35H..." LONGER THAN 4 CHARS IN
70245             1 COLUMN ,I2)
70246          200 FORMAT (1H0,T14, 46HZERO LENGTH VARIABLE NAME IN OR BEFORE COLUMN
70247             1,I2)
70248          205 FORMAT (1H0,T14, 10HVARIABLE ",A5, 44H" WAS NOT DECLARED IN A <STO
70249             1RAGE.> STATEMENT)
70250          210 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
70251          215 FORMAT (1H0,T14, 43HMORE THAN 200 VARIABLES REQUESTED FOR PRINT)
70252          220 FORMAT (1H0,T14, 10HVARIABLE ",A5, 41H" WAS NOT COMPLETELY DECLARE
70253             1D BY CARD END)
70254       C
70255              END
```

Line Number	Explanation
70196-70219	An error occurred on the PRINT. card and generated a diagnostic.
70224-70238	Errors on PRINT. cards are nonfatal to execution. The variable with the error is rejected and the column pointer is positioned at the beginning of the first variable following the error. Scan continues until a comma is encountered (MØR=1), followed by a numeric character. This numeric character is considered the beginning of a variable, name, the column pointer (ICØL) is positioned, and control proceeds to reinitialize the local variables.

2.4 *Parse PLØT. Cards*



Overview

Cards of the following form are processed by this section.

PLØT.(<var>,<var>,...[<val>,<val>]),(<group>),.../<var>[<val>,<val>]

where

<var>::= a 1-5 character variable name (possibly followed by subscripts).

[<val>,<val>]::= user defined range declaration, specifying the range over which variables in that group are to be plotted.

<group>: = 1-5 variable names (may also include a range declaration) to be plotted on one scale. There may be 1-5 groups per card; each group may have a different dependent axis scale.

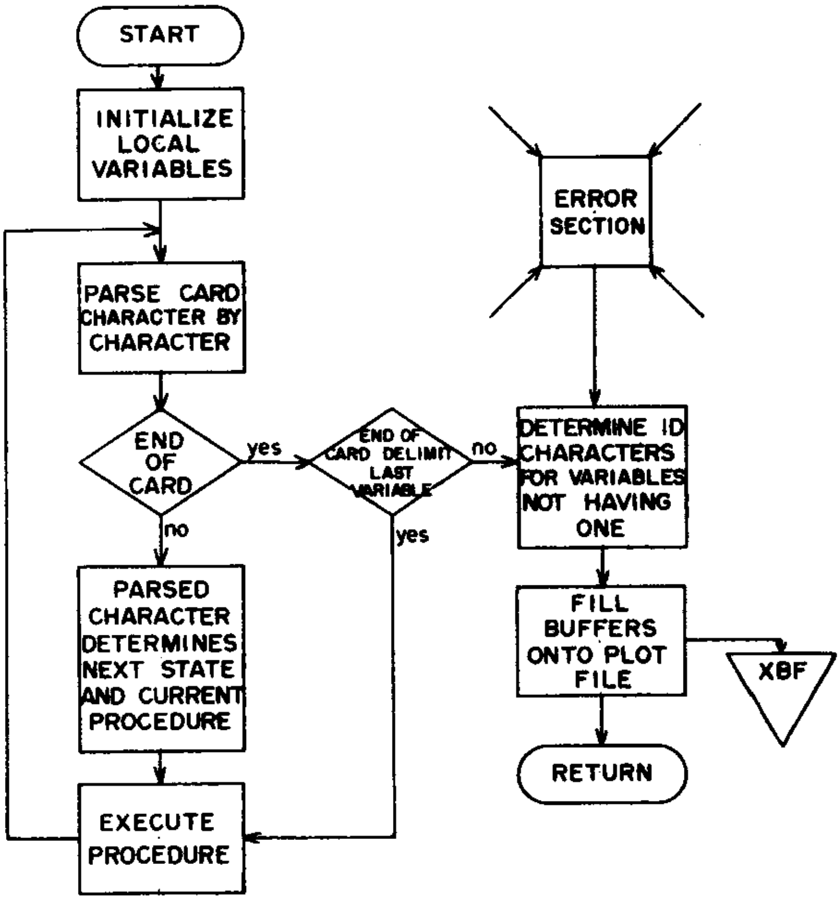
/<var>[<val>,<val>]::= the independent variable followed by a range declaration specifying the range of values for the independent variable. (The range declaration and /<var> are optional.)

A plot card image is the section input, while output consists of a plot stack (XPLT) and buffers (IBUF and RBUF) containing information about the current plot card which are flushed onto file XBF. Each different variable encountered on plot cards creates a new entry in XPLT containing information describing the mode and location of the value of the variable. (XPLT is used by routine XPLØT in the construction of a plot value file.)

Each record of XBF contains information pertaining to one plot (or title) card and is used by the plot section (PRØGRAM PLØT) to create plotted output. (See *Initialize local variables* subsection for a description of the format of XPLT, IBUF, and RBUF).

The following chart subdivides the section for easier analysis.

PLØT. cards flow chart


START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

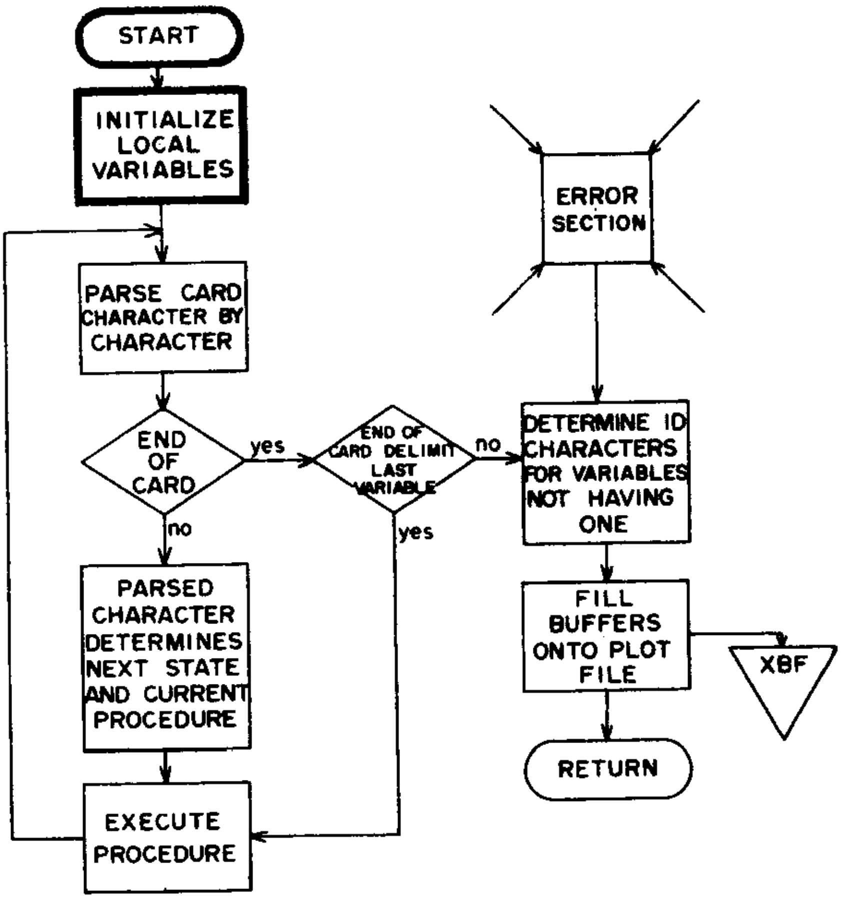
```
      SUBROUTINE XPLSTK (CARD,JCOL,QPLTS)
C
C.....THIS ROUTINE PROCESSES PLOT REQUEST CARDS OF THE FORM
C......  PLOT.(U1.LOG=A,....,UI"T1,T2"),....,(V1,....,VJ)/W"T1,T2"
C.....EACH VARIABLE NAME ENCOUNTERED IN THE PLOT REQUEST GENERATES AN
C.....ENTRY INTO THE PLOT VARIABLE STACK "XPLT(I)" IN THE FORMAT
C       LOCATION  BITS      INFORMATION
C       40-41     18        INDEX OF VARIABLE RELATIVE TO "XADRS(1)"
C       40-41     2         MODE OF VARIABLE (0=INTEGER, 1=REAL)
C.....AS THE PLOT REQUEST IS PROCESSED BUFFERS ARE FILLED CONTAINING
C.....THE INFORMATION FOR EACH PLOT. THE FORMAT FOR THE BUFFER FOLLOWS.
C       IBUF((I=1,2),J,K)---CONTAINS THE VARIABLE INFORMATION FOR THE
C                           JTH VARIABLE IN THE KTH GROUP
C                 CHARACTERS
C                 1-18      VARIABLE NAME AND SUBSCRIPTS
C                   19      A DOT IF A LOG REQUEST IS PRESENT
C                   20      VARIABLE ID CHARACTER
C       IBUF(3,J,K)---------CONTAINS THE INDEX OF VARIABLE RELATIVE TO
C                           XADRS(1)
C.....THE INDEPENDENT VARIABLE INFORMATION IS STORED AT IBUF((I=1,3),5,6
C.....RBUF((I=1,2),K)IS FILLED WITH THE RANGE SPECIFICATIONS FOR THE
C       KTH GROUP
C.....THE TWO BUFFERS ARE WRITTEN AS ONE RECORD ONTO THE PLOT STACK FILE
C
      COMMON /XXPLOT/ XPLT(100),XRNG(2)
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      DIMENSION CARD(8), KEY(8), MCOL(3), MSUB(3), NUM(3), TABLE(18), IP
     1ROC(18), LNAM(2), LOG(3), N1(3)
      INTEGER XPLT,XGRP,XNV,XNW,XVT1,XVT2
      INTEGER XUO,XUI,XUP,XBF,XNPL,XUE
      LOGICAL INDEP,XPLFG,XFILM,XTRACE,QPLTS
      DATA LOG/1HL,1HO,1HG/
      DATA KEY/1H(,1H),1H,,1H',1H|,1H.,1H=,1H//
      DATA XRNG/0.,0./, XDIF/1.E+321/
      DATA TABLE/04000000000000000000B,00000100000000300000B,1444210072
     1000306000B,00000000200002041000B,00442000000000300000B,24000B,1400
     2000B,32655326553265532655B,04016100000074000000B,00546000000000024
     3000B,00442100072000000000B,00442100002001400000B,0044210000000000
     4000B,04000000000000000000B,300000B,00021000200002041000B,000000002
     520002245100B,00000002620002245100B/
      DATA IPROC/02000000000000000000B,00000300000000200000B,1051230007
     1200306000B,00000000160001534640B,01020000000000200000B,12000B,1000
     2000B,22451224512245122451B,02021440000054000000B,01146000000000012
     3000B,00512300074200000000B,00512300004201000000B,0051230000000000
     4000B,02000000000000000000B,200000B,00017000160001534640B,160001534
     5640B,00000007560001534640B/
C
C.....INITIALIZE LOCAL VARIABLES
C
      DO 15 K=1,6
         RBUF(1,K)=0.
         RBUF(2,K)=0.
      DO 15 I=1,3
      DO 15 J=1,5
   15 IBUF(I,J,K)=0
      INDEP=.FALSE.
      XGRP=0
      NCOL=0
      ICOL=JCOL
C
C.....PARSE CARD COLUMN BY COLUMN SEARCHING FOR KEY CHARACTERS
C
      ISTATE=1
   20 JSTATE=ISTATE
   25 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 90
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 25
      DO 30 I=1,8
         IF (ICHR.EQ.KEY(I)) GO TO 35
   30 CONTINUE
      I=0
      IF (ICHR.LE.1H9.OR.ICHR.GE.1H0) I=10
      IF (ICHR.GE.1HA.AND.ICHR.LE.1HZ) I=9
      IF (I.EQ.0) I=11
   35 I=(I-1)*5
      CALL GBYTE (TABLE(JSTATE),ISTATE,I,5)
      CALL GBYTE (IPROC(JSTATE),IGO,I,5)
      IF (ISTATE.EQ.0) GO TO 295
      GO TO (40,45,50,55,60,65,75,80,85,95,150,155,160,165,170,185,20,19
     10,70), IGO
C
C.....( ENCOUNTERED--GROUP BEGINNING
C
   40 XGRP=XGRP+1
      NVAR=0
      IF (XGRP.GT.5) GO TO 245
      GO TO 20
C
C.....CHARACTER ENCOUNTERED--BEGINNING OF A VARIABLE NAME
C
   45 NCOL=1
      NUMS=0
      LNAM(2)=0
      LNAM(1)=LNAM(2)
      NAME=10H
      NVAR=NVAR+1
      IF (NVAR.GT.5) GO TO 250
      CALL SCHARS (NAME,NCOL,1,ICHR)
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      GO TO 20
C
C.....CHARACTER ENCOUNTERED--PART OF A VARIABLE NAME
C
   50 NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 255
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 20
C
C.....( ENCOUNTERED--BEGINNING OF A SUBSCRIPT
C
   55 NUMS=1
      NCOL=NCOL+1
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      MSUB(3)=0
      MSUB(2)=MSUB(3)
      MSUB(1)=MSUB(2)
      MCOL(3)=0
      MCOL(2)=MCOL(3)
      MCOL(1)=MCOL(2)
      GO TO 20
C
C.....INTEGER ENCOUNTERED--PART OF A SUBSCRIPT
C
   60 MCOL(NUMS)=MCOL(NUMS)+1
      NCOL=NCOL+1
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      IF (MCOL(NUMS).GT.3) GO TO 265
      CALL SCHARS (MSUB(NUMS),MCOL(NUMS),1,ICHR)
      GO TO 20
C
C.....COMMA ENCOUNTERED--DELIMITING SUBSCRIPTS
C
   65 NUMS=NUMS+1
      NCOL=NCOL+1
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      IF (NUMS.GT.3) GO TO 270
      GO TO 20
C
C.....) ENCOUNTERED--END OF SUBSCRIPTS
C
   70 NCOL=NCOL+1
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      GO TO 20
C
C.....PERIOD ENCOUNTERED--LOG DECLARATION
C
   75 NLOG=0
      IBUF(2,NVAR,XGRP)=IBUF(2,NVAR,XGRP).OR.5700B
      GO TO 20
C
C.....LOG CHARACTERS ENCOUNTERED
C
   80 NLOG=NLOG+1
      IF (NLOG.GT.3) GO TO 275
      IF (ICHR.NE.LOG(NLOG)) GO TO 275
      GO TO 20
C
C.....VARIABLE ID CHARACTER ENCOUNTERED
C
   85 CALL SCHARS (IBUF(2,NVAR,XGRP),10,1,ICHR)
      GO TO 20
C
C.....END OF CARD--DELIMITS INDEPENDENT VARIABLE
C
   90 IF (.NOT.INDEP) GO TO 195
      IF (NCOL.EQ.0) GO TO 195
```

```
C
C.....A VARIABLE DELIMITER ENCOUNTERED--DECODE SUBSCRIPTS
C
   95 IF (NUMS.EQ.0) GO TO 105
      DO 100 J=1,NUMS
         NUM(J)=0
         CALL GNUM (MSUB(J),1,MCOL(J),NUM(J),IERR)
  100 CONTINUE
C
C.....FIND VARIABLE ENTRY IN REFERENCE TABLES
C
  105 DO 110 I=1,XNV
         NM=(77777777770000000000B.A.XVT1(I)).O.5555555555B
         IF (NAME.EQ.NM) GO TO 115
  110 CONTINUE
      GO TO 290
C
C.....VARIABLE FOUND IN REFERENCE TABLES RETRIEVE TABLE INFORMATION
C
  115 IREF=0
      CALL GBYTE (XVT1(I),N0,30,18)
      CALL GBYTE (XVT1(I),MD,48,2)
      IF (NUMS.EQ.0) GO TO 130
      N1(1)=1
      DO 120 J=1,2
         CALL GBYTE (XVT2(I),N1(J+1),10*J-10,10)
  120 CONTINUE
C
C.....COMPUTE ADDRESS OF VARIABLE RELATIVE TO 1ST WORD ADDRESS
C.....OF VARIABLE
C
      DO 125 J=1,3
         K=4-J
         NN=1
         IF (MCOL(K).GT.0) NN=NUM(K)
  125 IREF=N1(K)*(NN-1+IREF)
C
C.....COMPUTE INDEX RELATIVE TO XADRS(1)
C
  130 IBIAS=N0+IREF
C
C.....CREATE ENTRY INFORMATION FOR PLOT VARIABLE STACK
C
      INFO=0
      CALL SBYTE (INFO,MD,40,2)
      CALL SBYTE (INFO,IBIAS,42,18)
      IF (XNPL.LE.0) GO TO 140
      DO 135 I=1,XNPL
         IF (INFO.EQ.XPLT(I)) GO TO 145
  135 CONTINUE
  140 IF (XNPL.GE.100) GO TO 297
      XNPL=XNPL+1
      XPLT(XNPL)=INFO
      I=XNPL
C
C.....ENTER INFORMATION ONTO BUFFER
C
  145 IBUF(1,NVAR,XGRP)=LNAM(1)
      CALL SCHARS (IBUF(2,NVAR,XGRP),1,8,LNAM(2))
      IBUF(3,NVAR,XGRP)=I
      NCOL=0
      IF (INDEP) GO TO 200
      GO TO 20
C
C.....SLASH ENCOUNTERED--INDEPENDENT VARIABLE SEARCH
C
  150 INDEP=.TRUE.
      NVAR=0
      XGRP=6
      GO TO 20
C
C.....LEFT BRACKET ENCOUNTERED--BEGINNING OF RANGE VALUES
```

```
C
  155 LTIM=0
      LCOL=0
      LRANGE=1
      GO TO 20
C
C.....CHARACTER ENCOUNTERED--PART OF RANGE VALUES
C
  160 IF (ICHR.EQ.1HE) GO TO 165
      IF (ICHR.EQ.1H+) GO TO 165
      IF (ICHR.EQ.1H-) GO TO 165
      GO TO 280
  165 LCOL=LCOL+1
      IF (LCOL.GT.10) GO TO 285
      CALL SCHARS (LTIM,LCOL,1,ICHR)
      GO TO 20
C
C.....COMMA OR RIGHT BRACKET ENCOUNTERED--DELIMITS RANGE DECLARATIONS
C
  170 ENCODE (10,300,FMT) LCOL
      DECODE (LCOL,FMT,LTIM) DUM
      RBUF(LRANGE,XGRP)=DUM
      LTIM=0
      LCOL=0
      LRANGE=LRANGE+1
      IF (LRANGE.LE.2) GO TO 20
      IF (RBUF(1,XGRP).LT.RBUF(2,XGRP)) GO TO 180
      IF (RBUF(1,XGRP).NE.RBUF(2,XGRP)) GO TO 175
      WRITE (XUO,305) CARD
      RBUF(2,XGRP)=0.
      RBUF(1,XGRP)=RBUF(2,XGRP)
      GO TO 20
  175 DUM=RBUF(1,XGRP)
      RBUF(1,XGRP)=RBUF(2,XGRP)
      RBUF(2,XGRP)=DUM
  180 IF (XGRP.NE.6) GO TO 20
      DUM=RBUF(2,XGRP)-RPUF(1,XGRP)
      IF (DUM.GT.XDIF) GO TO 20
      XDIF=DUM
      XRNG(1)=RBUF(1,XGRP)
      XRNG(2)=RBUF(2,XGRP)
      GO TO 20
  185 IF (NCOL.GT.0) GO TO 95
      GO TO 20
C
C.....LEFT BRACKET ENCOUNTERED--INDEPENDENT RANGE DECLARATION
C
  190 INDEP=.TRUE.
      XGRP=6
      GO TO 155
C
C.....DUMP BUFFERS ONTO PLOT FILE
C
  195 IBUF(1,1,6)=10HTIME
      IBUF(3,1,6)=1
C
C.....FILL CHARACTERS FOR ANY VARIABLES NOT SPECIFIED
C
  200 KNT=0
      DO 230 K=1,5
         IF (IBUF(1,1,K).EQ.0) GO TO 232
         DO 225 J=1,5
            IF (IBUF(1,J,K).EQ.0) GO TO 230
            ICHR=IBUF(2,J,K).AND.77B
            IF (ICHR.NE.0) GO TO 225
  205       KNT=KNT+1
            DO 215 N=1,5
               IF (IBUF(1,1,N).EQ.0) GO TO 220
               DO 210 M=1,5
                  IF (IBUF(1,M,N).EQ.0) GO TO 215
                  ICHR=IBUF(1,M,M).AND.77B
                  IF (ICHR.EQ.KNT) GO TO 205
  210          CONTINUE
  215       CONTINUE
```

```
 220        IBUF(2,J,K)=IBUF(2,J,K).OR.KNT
 225    CONTINUE
 230 CONTINUE
C
C.....IF INDEPENDENT VARIABLE IS TIME, AND NO RANGES WERE SPECIFIED SET
C.....XRNG FLAGS FOR ROUTINE XCSTART
C
 232 IF (IBUF(3,1,6).NE.1) GO TO 235
     IF (XRNG(1).NE.XRNG(2)) GO TO 235
     XRNG(2)=1.
     XRNG(1)=XRNG(2)
 235 WRITE (XBF) IBUF,RBUF
     XPLFG=.TRUE.
     RETURN
C
C.....ERROR SECTION
C
 245 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,330)
     GO TO 195
 250 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,335)
     GO TO 195
 255 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,340)
 260 IF (XGRP.GT.1) GO TO 195
     IF (NVAR.GT.1) GO TO 195
     WRITE (XUO,325)
     RETURN
 265 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,345)
     GO TO 260
 270 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,350)
     GO TO 260
 275 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,355)
     GO TO 260
 280 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,360)
     GO TO 260
 285 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,365)
     GO TO 260
 290 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,370) NAME
     GO TO 260
 295 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,375) ICHR
     GO TO 260
 297 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
     WRITE (XUO,380)
     QPLTS=.TRUE.
     RETURN
C
 300 FORMAT (2H(E,I2,6H.0)    )
 305 FORMAT (6H0*****, 23HERROR IN PLOT REQUEST--,8A10,/,T14, 43HA RANG
    1E DECLARATION HAS IDENTICAL ENDPOINTS)
 320 FORMAT (6H0*****, 51HERROR IN PLOT REQUEST--PROCESSING HALTED AT C
    1OLUMN ,I2//,T20,8I10/T20,8(10H1234567890),/,T20,8A10)
 325 FORMAT (1H ,5X, 26HPLOT REQUEST NOT PROCESSED)
 330 FORMAT (1H0,T14, 32HNO. OF GROUPS PER PLOT IS .GT. 5)
 335 FORMAT (1H0,T14, 36HNO. OF VARIABLES PER GROUP IS .GT. 5)
 340 FORMAT (1H0,T14, 34HVARIABLE NAME IS .GT. 5 CHARACTERS)
 345 FORMAT (1H0,T14, 44HVARIABLE SUBSCRIPT .GT. 999--THE UPPER LIMIT)
 350 FORMAT (1H0,T14, 30HVARIABLE HAS .GT. 3 SUBSCRIPTS)
 355 FORMAT (1H0,T14, 32HIMPROPERLY FORMATTED LOG REQUEST)
 360 FORMAT (1H0,T14, 38HILLEGAL CHARACTER IN RANGE DECLARATION)
 365 FORMAT (1H0,T14, 53HRANGE DECLARATION .GT. 10 CHARACTERS--THE UPPE
    1R LIMIT)
 370 FORMAT (1H0,T14,  9HVARIABLE ,A5, 24H NOT DECLARED IN STORAGE)
 375 FORMAT (1H0,T14, 28HILLEGAL CHARACTER DETECTED ",A1,  1H")
 380 FORMAT (1H0,T14,89HMORE THAN 100 VARIABLES NAMED IN PLOT REQUESTS,
    1 THIS AND SUBSEQUENT PLOT REQUESTS IGNORED)
C
     END
```

Initialize local variables


START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

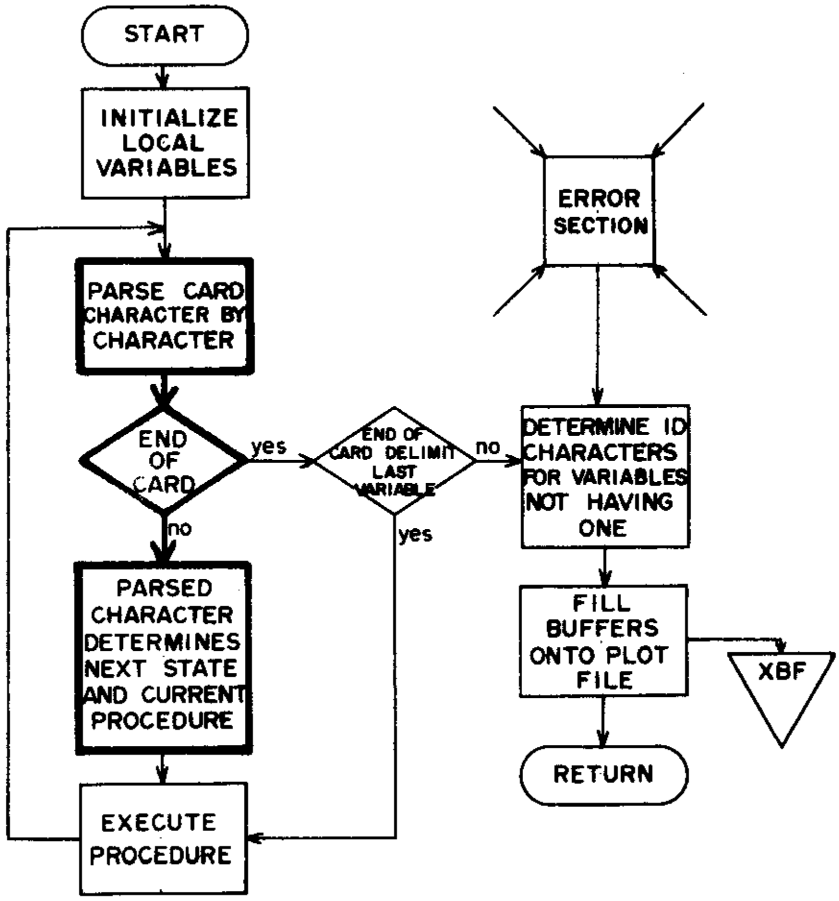
```
      SUBROUTINE XPLSTK (CARD,JCOL,QPLTS)
C
C.....THIS ROUTINE PROCESSES PLOT REQUEST CARDS OF THE FORM
C......   PLOT.(U1.LOG=A,....,UI"T1,T2"),....,(V1,...,VJ)/W"T1,T2"
C.....EACH VARIABLE NAME ENCOUNTERED IN THE PLOT REQUEST GENERATES AN
C.....ENTRY INTO THE PLOT VARIABLE STACK "XPLT(I)" IN THE FORMAT
C         LOCATION  BITS       INFORMATION
C         40-41     18         INDEX OF VARIABLE RELATIVE TO "XADRS(1)"
C         42-59     2          MODE OF VARIABLE (0=INTEGER, 1=REAL)
C.....AS THE PLOT REQUEST IS PROCESSED BUFFERS ARE FILLED CONTAINING
C.....THE INFORMATION FOP EACH PLOT. THE FORMAT FOR THE RUFFER FOLLOWS.
C         IBUF((I=1,2),J,K)---CONTAINS THE VARIABLE INFORMATION FOR THE
C                             JTH VARIABLE IN THE KTH GROUP
C                   CHARACTERS
C                   1-18      VARIABLE NAME AND SUBSCRIPTS
C                     19      A DOT IF A LOG REQUEST IS PRESENT
C                     20      VARIABLE ID CHARACTER
C         IBUF(3,J,K)---------CONTAINS THE INDEX OF VARIABLE RELATIVE TO
```

```
74018          C                                 XADRS(1)
74019          C.....THE INDEPENDENT VARIABLE INFORMATION IS STORED AT IBUF((I=1,3),5,6
74020          C.....RBUF((I=1,2),K)IS FILLED WITH THE RANGE SPECIFICATIONS FOR THE
74021          C         KTH GROUP
74022          C.....THE TWO BUFFERS ARE WRITTEN AS ONE RECORD ONTO THE PLOT STACK FILE
74023          C
74024                COMMON /XXPLOT/ XPLT(100),XRNG(2)
74025                COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
74026                COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
74027                COMMON /XXVR2FR/ XVT2(1)
74028                COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
74029                DIMENSION CARD(8), KEY(8), MCOL(3), MSUB(3), NUM(3), TABLE(18), IP
74030               1ROC(18), LNAM(2), LOG(3), N1(3)
74031                INTEGER XPLT,XGRP,XNV,XNW,XVT1,XVT2
74032                INTEGER XUO,XUI,XUP,XBF,XNPL,XUE
74033                LOGICAL INDEP,XPLFG,XFILM,XTRACE,QPLTS
74034                DATA LOG/1HL,1HO,1HG/
74035                DATA KEY/1H(,1H),1H,,1H',1H|,1H.,1H=,1H//
74036                DATA XRNG/0.,0./, XDIF/1.E+321/
74037                DATA TABLE/04000000000000000000B,00000100000000300000B,1444210007
74038               21000306000B,00000000200002041000B,00442000000000300000B,24000B,1400
74039               2000B,32655326553265532655B,04016100000074000000B,0054600000000024
74040               3000B,00442100072000000000B,00442100002001400000B,0044210000000000
74041               4000B,04000000000000000000B,300000B,00021000200002041000B,000000002
74042               520002245100B,00000002620002245100B/
74043                DATA IPROC/02000000000000000000B,00000300000000200000B,1051230007
74044               41200306000B,00000000160001534640B,01020000000000200000B,12000B,1000
74045               2000B,22451224512245122451B,02021440000054000000B,0114600000000012
74046               3000B,00512300074200000000B,00512300004201000000B,0051230000000000
74047               4000B,02000000000000000000B,200000B,00017000160001534640B,160001534
74048               5640B,00000007560001534640B/
74049          C
74050          C.....INITIALIZE LOCAL VARIABLES
74051          C
74052                DO 15 K=1,6
74053                   RBUF(1,K)=0.
74054                   RBUF(2,K)=0.
74055                DO 15 I=1,3
74056                DO 15 J=1,5
74057             15 IBUF(I,J,K)=0
74058                INDEP=.FALSE.
74059                XGRP=0
74060                NCOL=0
74061                ICOL=JCOL
```

Line Number	Explanation
74000-74061	CARD is one user data card (80 columns). JCØL points at the column before the beginning of the text portion of the card. Each different variable name encountered (from a plot card) creates an entry in the plot stack, XPLT. Columns 40-41 contain the mode of the variable (0=integer, 1=real) and columns 42-59 contain the address (of the value) of the variable relative to XADRS(1). XPLT is used to retrieve the values of all variables to be plotted and to store these values on a plot value file (to be used for the construction of plots) for each plot time increment (i.e., TSTRT, TSTRT+DTPL, TSTRT+2*DTPL, etc.).

Line Number	Explanation
	XRNG contains information used to determine the optimum DTPL (used in routine XCSTART).
	KEY is a list of all legal special characters that could be encountered during a plot card parse.
	For TABLE, each word of TABLE lists the next state for each possible character encountered (packed in groups of five bits each).
	For IPRØC, each word of IPRØC defines the procedure to be taken for each encountered character.
	RBUF(I,J) will contain the user specified range declaration for the Jth group on the plot card. If left is equal to zero, no range declaration for the group was specified. RBUF(I,6) contains the specified range declaration for the independent variable.
	IBUF(I,J,K) is a buffer filled with information about the plot card currently being processed. IBUF(I=1,2),J,K) contains information about the Jth variable of the Kth group. Characters 1-18 contain the variable name and subscripts; 19 contains a dot if a log request is present; and 20 contains the variable ID character.
	IBUF(3,J,K) contains the location (of the value) of the variable relative to the beginning of blank common (XADRS(1)). The independent variable information is filled into IBUF((I=1,3),(J=1),6).
	INDEP is a logical flag set to .TRUE. when parsing encounters information relating to the independent variable.
	XGRP contains the number of the group currently being processed.
	NCØL is a count of the number of characters in the variable name and its subscripts filled into IBUF((I=1,2),J,K)
	ICØL points to the column on the data card currently being processed.

Determine state and procedure


START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

```
C
C.....PARSE CARD COLUMN BY COLUMN SEARCHING FOR KEY CHARACTERS
C
      ISTATE=1
   20 JSTATE=ISTATE
   25 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 90
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 25
      DO 30 I=1,8
         IF (ICHR.EQ.KEY(I)) GO TO 35
   30 CONTINUE
      I=0
      IF (ICHR.LE.1H9.OR.ICHR.GE.1H0) I=10
      IF (ICHR.GE.1HA.AND.ICHR.LE.1HZ) I=9
      IF (I.EQ.0) I=11
   35 I=(I-1)*5
      CALL GBYTE (TABLE(JSTATE),ISTATE,I,5)
      CALL GBYTE (IPROC(JSTATE),IGO,I,5)
      IF (ISTATE.EQ.0) GO TO 295
```

Line Number Explanation

74065-74066 ISTATE contains the next state number.
JSTATE is the current state number.

74067-74077 Select the next character from the data card:
ICHR contains the character currently being processed.
Determine whether the character is a special character by comparing it with KEY.

I = 1-8 The character is the Ith character in KEY.
= 9 An alphabetic character (α).
= 10 A numeric character (β).
= 11 Any other character.

74078-74081 Using the code for the current character (I) and the current state number (JSTATE), determine the current procedure (IGØ) and the next state number (ISTATE). If the next state number equals zero, then an illegal character has been encountered.

TABLE contains the set of state numbers:
TABLE(JSTATE) contains the list of all possible next state numbers. Each group of five bits (from left to right) in TABLE(JSTATE) defines the next state number determined by the current character.

EXAMPLE. The first five bits of TABLE(JSTATE) contain the next state number if the current character is the first character of KEY (i.e., a left paren). If JSTATE=1 and ICHR=1H(, then the first five bits of TABLE(1) contain the next state, ISTATE=2.

Table 2.1 contains the contents of TABLE with each group of five bits represented by its base 10 equivalent. Blank entries indicate illegal conditions.

IPRØC is arranged similarly to TABLE, except that IPRØC contains a code for the procedure to execute for a given current state.

The state and procedure flow chart (Fig. 2.1) illustrates which process is executed when a given character is encountered.

○ indicates a state.
□ indicates a process.

Line Number	Explanation
	For a given current state I, all legal characters that may be encountered are shown as paths away from I. Any other character encountered while the current state is I is illegal. The procedure to execute, J is shown by each character. At the end of the path that the encountered character is on is the next state. For example, assume the current state number is 2, the only possible legal characters that may be encountered while in this state are an α or [. Assume the character is [, then the procedure executed is 12 and the next state will be 4.

Table 2.1. Expanded view of TABLE. The intersection of an encountered character with the current state number determines the next state number. All of the blank (zero) slots are illegal conditions, errors.

Current State Number	Encountered Characters											
	(	)	,	[	]	•	=	/	α	β	Else	
1	2											TABLE(1)
2				4					3			TABLE(2)
3	6	10	2	4		7	8		3	3		
4						16			16	16	16	
5		9	2						3			
6										10		
7									12			
8	13	13	13	13	13	13	13	13	13	13	13	⋮
9	2		14	4				15				
10		11	6						10			
11		9	2	4		7	8					
12		9	2	4			8		12			
13		9	2	4								
14	2											
15									3			
16			17			16			16	16	16	
17						18			18	18	18	
18					5	18			18	18	18	TABLE(18)

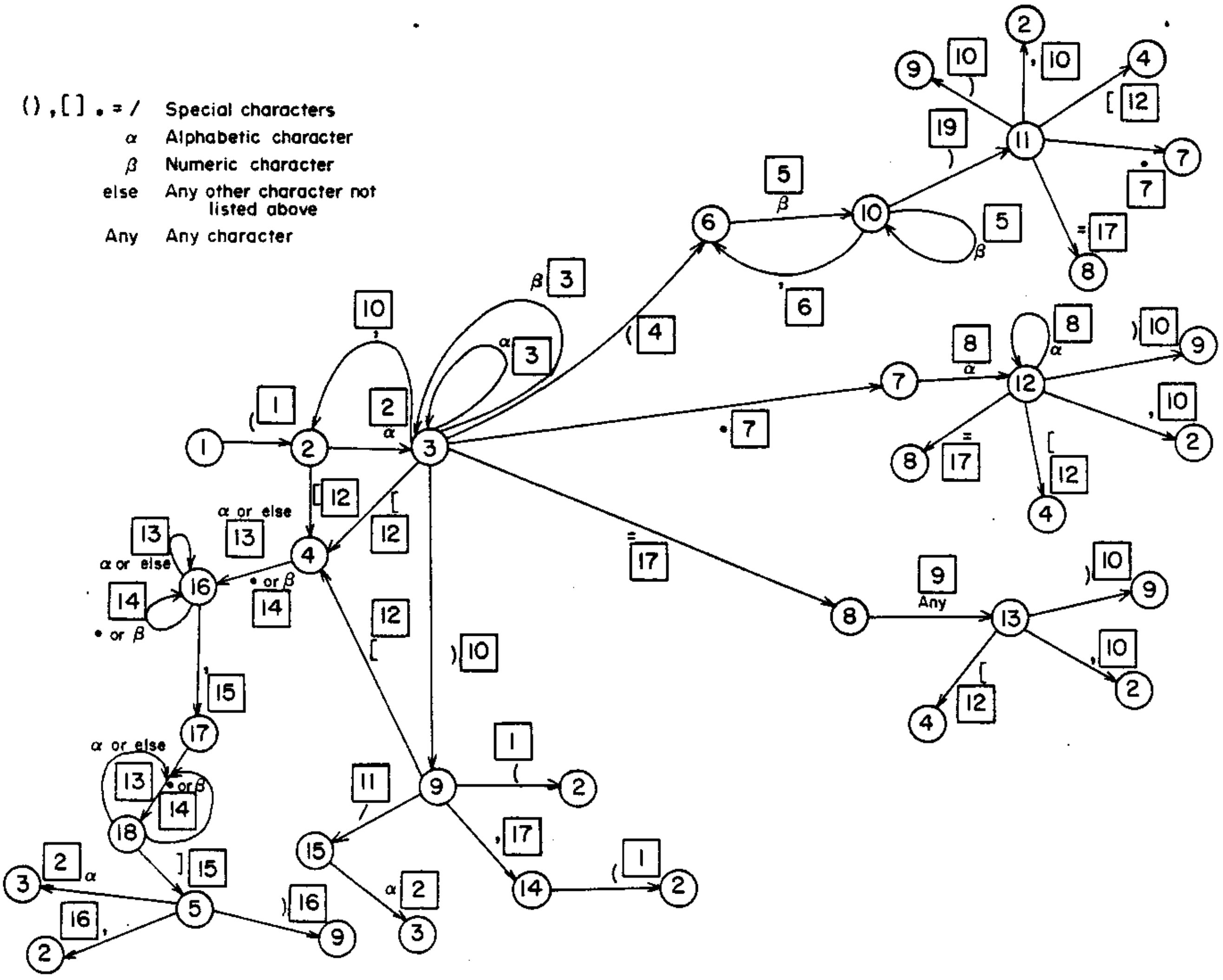
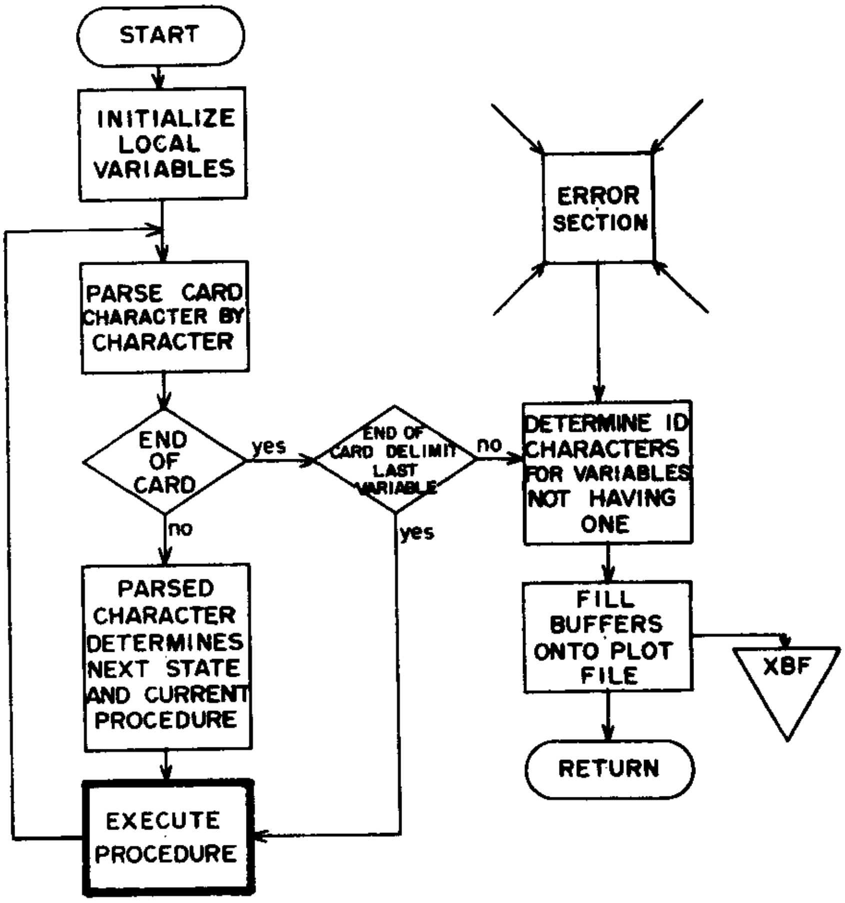


Fig. 2.1. State and procedure flow chart.

Execute procedure


START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

```
      GO TO (40,45,50,55,60,65,75,80,85,95,150,155,160,165,170,185,20,19
     10,70), IGO
C
C.....( ENCOUNTERED--GROUP BEGINNING
C
   40 XGRP=XGRP+1
      NVAR=0
      IF (XGRP.GT.5) GO TO 245
      GO TO 20
C
C.....CHARACTER ENCOUNTERED--BEGINNING OF A VARIABLE NAME
C
   45 NCOL=1
      NUMS=0
      LNAM(2)=0
      LNAM(1)=LNAM(2)
      NAME=10H
      NVAR=NVAR+1
      IF (NVAR.GT.5) GO TO 250
      CALL SCHARS (NAME,NCOL,1,ICHR)
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      GO TO 20
C
C.....CHARACTER ENCOUNTERED--PART OF A VARIABLE NAME
C
   50 NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 255
      CALL SCHARS (LNAM,NCOL,1,ICHR)
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 20
```

Line Number	Explanation
74082	Branch to the procedure indicated by IGØ, the current procedure number. NOTE. Statement number 40 (corresponds with IGØ=1) is procedure number 1 on the state and procedure flow chart, statement number 45 is procedure 2, etc.
74084-74090	A left paren is encountered, signaling the beginning of a new group. XGRP, which counts the number or groups, is incremented. NVAR, which counts the number of characters in the variable name is initialized. The maximum number of groups allowed per card is 5.
74091-74103	An alphabetic character is encountered signaling the beginning of a variable name. The character is filled into NAME which will contain the 1-5 character variable name and LNAM which will contain the variable name and subscripts. NCØL is a count of the number of characters in LNAM.
74104-74111	The character encountered is part of a variable name store the character in LNAM and NAME.

```
74112      C
74113      C.....( ENCOUNTERED--BEGINNING OF A SUBSCRIPT
74114      C
74115         55 NUMS=1
74116            NCOL=NCOL+1
74117            CALL SCHARS (LNAM,NCOL,1,ICHR)
74118            MSUB(3)=0
74119            MSUB(2)=MSUB(3)
74120            MSUB(1)=MSUB(2)
74121            MCOL(3)=0
74122            MCOL(2)=MCOL(3)
74123            MCOL(1)=MCOL(2)
74124            GO TO 20
74125      C
74126      C.....INTEGER ENCOUNTERED--PART OF A SUBSCRIPT
74127      C
74128         60 MCOL(NUMS)=MCOL(NUMS)+1
74129            NCOL=NCOL+1
74130            CALL SCHARS (LNAM,NCOL,1,ICHR)
74131            IF (MCOL(NUMS).GT.3) GO TO 265
74132            CALL SCHARS (MSUB(NUMS),MCOL(NUMS),1,ICHR)
74133            GO TO 20
74134      C
74135      C.....COMMA ENCOUNTERED--DELIMITING SUBSCRIPTS
74136      C
74137         65 NUMS=NUMS+1
74138            NCOL=NCOL+1
74139            CALL SCHARS (LNAM,NCOL,1,ICHR)
74140            IF (NUMS.GT.3) GO TO 270
74141            GO TO 20
74142          C
74143          C.....) ENCOUNTERED--END OF SUBSCRIPTS
74144          C
74145             70 NCOL=NCOL+1
74146                CALL SCHARS (LNAM,NCOL,1,ICHR)
74147                GO TO 20
```

Line Number	Explanation
74112-74123	A left paren indicates the beginning of the subscript of a variable. Initialize the variable which will contain these subscripts. NUMS indicates to which subscript of the variable the character belongs (i.e., NUMS=2, then the character belongs to the second subscript). MCØL(I) counts the number of characters in the Ith subscript. MSUB(I) is filled with the characters of the Ith subscript.
74125-74133	The numeric character encountered is filled into MSUB, and MCØL is incremented. NOTE. Three characters per subscript are the maximum number allowed; thus a subscript of 999 is the largest possible single subscript value. The character is also filled into LNAM containing the variable name and subscripts.
74134-74141	The comma encountered delimits subscripts. NUMS is incremented (the indicator of which subscript is being processed). A variable may have not more than three subscripts.
74142-74147	The right paren encountered signals the end of the subscripts.

```
74148          C
74149          C.....PERIOD ENCOUNTERED--LOG DECLARATION
74150          C
74151             75 NLOG=0
74152                IBUF(2,NVAR,XGRP)=IBUF(2,NVAR,XGRP).OR.5700B
74153                GO TO 20
74154          C
74155          C.....LOG CHARACTERS ENCOUNTERED
74156          C
74157             80 NLOG=NLOG+1
74158                IF (NLOG.GT.3) GO TO 275
74159                IF (ICHR.NE.LOG(NLOG)) GO TO 275
74160                GO TO 20
74161          C
74162          C.....VARIABLE ID CHARACTER ENCOUNTERED
74163          C
74164             85 CALL SCHARS (IBUF(2,NVAR,XGRP),10,1,ICHR)
74165                GO TO 20
```

Line Number	Explanation
74148-74153	A period encountered denotes a log declaration. Fill the period into the 19th character position of the buffer, IBUF((I=1,2),NVAR,XGRP).
74153-74160	Following the period must be the letters LØG. The current character is compared with local variable LØG(NLØG) to insure that the syntax following the period is correct. LØG(1)=1HL, LØG(2)=1HØ, LØG(3)=1HG NLØG counts the number of characters following the period (i.e., the first character following must be equivalent to LØG(NLØG)). Improper syntax results in an error message being printed.
74161-74165	Control arrives here when an equals (=) sign followed by any character is encountered. The character is a variable identification character and is stored in the 20th character position of IBUF (positioned at the current group and variable).

```
74171          C
74172          C.....A VARIABLE DELIMITER ENCOUNTERED--DECODE SUBSCRIPTS
74173          C
74174             95 IF (NUMS.EQ.0) GO TO 105
74175                DO 100 J=1,NUMS
74176                   NUM(J)=0
74177                   CALL GNUM (MSUB(J),1,MCOL(J),NUM(J),IERR)
74178            100 CONTINUE
74179          C
74180          C.....FIND VARIABLE ENTRY IN REFERENCE TABLES
74181          C
74182            105 DO 110 I=1,XNV
74183                   NM=(77777777770000000000B.A.XVT1(I)).O.5555555555B
74184                   IF (NAME.EQ.NM) GO TO 115
74185            110 CONTINUE
74186                GO TO 290
74187          C
74188          C.....VARIABLE FOUND IN REFERENCE TABLES RETRIEVE TABLE INFORMATION
74189          C
74190            115 IREF=0
74191                CALL GBYTE (XVT1(I),N0,30,18)
74192                CALL GBYTE (XVT1(I),MD,48,2)
74193                IF (NUMS.EQ.0) GO TO 130
74194                N1(1)=1
74195                DO 120 J=1,2
74196                   CALL GBYTE (XVT2(I),N1(J+1),10*J-10,10)
74197            120 CONTINUE
```

```
74198          C
74199          C.....COMPUTE ADDRESS OF VARIABLE RELATIVE TO 1ST WORD ADDRESS
74200          C.....OF VARIABLE
74201          C
74202                DO 125 J=1,3
74203                   K=4-J
74204                   NN=1
74205                   IF (MCOL(K).GT.0) NN=NUM(K)
74206            125 IREF=NI(K)*(NN-1+IREF)
74207          C
74208          C.....COMPUTE INDEX RELATIVE TO XADRS(1)
74209          C
74210            130 IBIAS=NO+IREF
74211          C
74212          C.....CREATE ENTRY INFORMATION FOR PLOT VARIABLE STACK
74213          C
74214                INFO=0
74215                CALL SBYTE (INFO,MD,40,2)
74216                CALL SBYTE (INFO,IBIAS,42,18)
74217                IF (XNPL.LE.0) GO TO 140
74218                DO 135 I=1,XNPL
74219                   IF (INFO.EQ.XPLT(I)) GO TO 145
74220            135 CONTINUE
74221            140 IF (XNPL.GE.100) GO TO 297
74222                XNPL=XNPL+1
74223                XPLT(XNPL)=INFO
74224                I=XNPL
74225          C
74226          C.....ENTER INFORMATION ONTO BUFFER
74227          C
74228            145 IBUF(1,NVAR,XGRP)=LNAM(1)
74229                CALL SCHARS (IBUF(2,NVAR,XGRP),1,8,LNAM(2))
74230                IBUF(3,NVAR,XGRP)=I
74231                NCOL=0
74232                IF (INDEP) GO TO 200
74233                GO TO 20
```

Line Number	Explanation
	A comma or a right paren delimits variables (flow through the state table determines whether the character delimits subscripts or variables).
74174-74178	The subscript characters stored in MSUB are decoded and their integer equivalents are placed in NUM. RECALL. NUMS is a counter of the number of subscripts encountered. (If NUMS=0 the variable is not a subscripted variable.) MCØL(J) is the number of characters in Jth subscript. MSUB(J) are characters of Jth subscript, then NUM(J) contains the integer value of the Jth subscript.
74179-74186	The variable reference tables are searched for a match with the currently parsed variable name (NAME).

Line Number	Explanation
74187-74197	The location of the variable in the reference tables is found. Retrieve information stored there. I is the index of the variable's location in stack. NO is the location of the first word address of the variable relative to the beginning of blank common. MD is the mode of the variable. N1(J) is the (J-1)th subscript of variable from tables (see PRINT. card section for more detailed analysis).
74198-74206	IREF contains the address of the variable (from the plot card) relative to the first word address of the variable (EXAMPLE. F(5) relative to F(1) assigns IREF=4).
74210	IBIAS contains the index of the variable relative to the beginning of blank common. (XADRS(IBIAS) is where the value of F(5) will be stored.)
74212-74216	INFØ contains information about the variable which is to be stored in the plot variable stack. Bits 40-41 are the mode of the variable; bits 42-59 are the index of variable relative to XADRS(1).
74217-74224	The plot stack is searched to determine if the variable already exists in the plot stack. If the information is not in the plot stack, enter it. XNPL is the number of variables in stack.
74225-74231	Enter information into buffer. Information describing the current variable in the current group is filled into IBUF((I=1,3), NVAR, XGRP) (see *Initialize local variable* subsection for IBUF format).
74232	If variable is independent variable, the parsing of card is done.

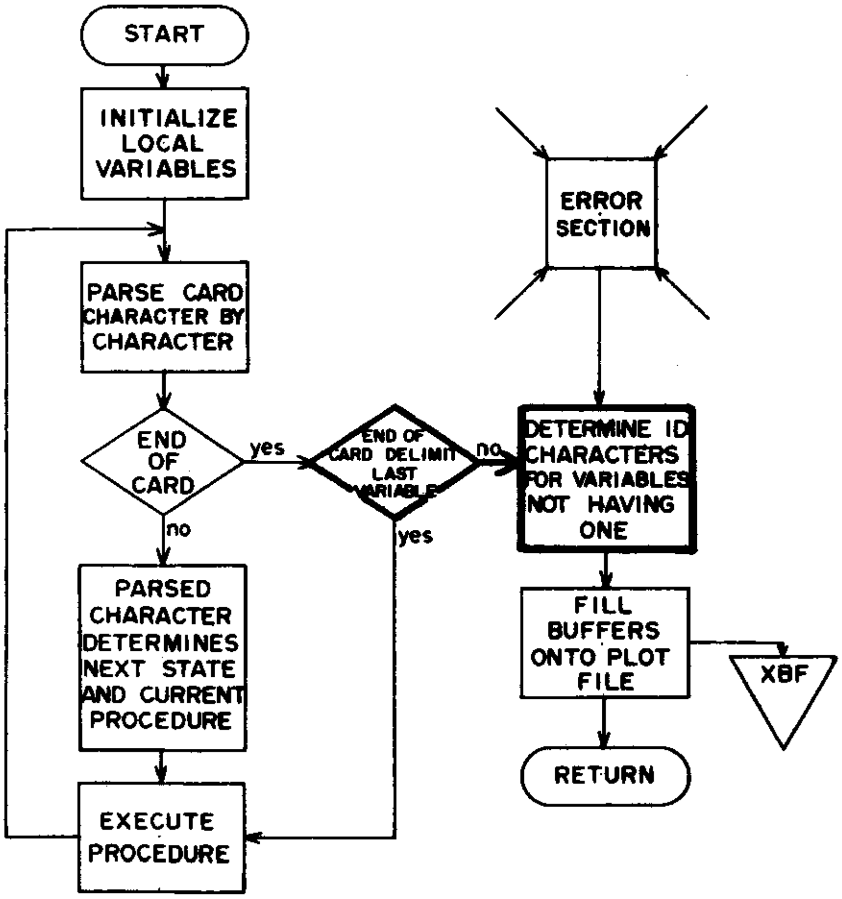
```
74234         C
74235         C.....SLASH ENCOUNTERED--INDEPENDENT VARIABLE SEARCH
74236         C
74237           150 INDEP=.TRUE.
74238               NVAR=0
74239               XGRP=6
74240               GO TO 20
74241         C
74242         C.....LEFT BRACKET ENCOUNTERED--BEGINNING OF RANGE VALUES
74243         C
74244           155 LTIM=0
74245               LCOL=0
74246               LRANGE=1
74247               GO TO 20
```

```
74248          C
74249          C.....CHARACTER ENCOUNTERED--PART OF RANGE VALUES
74250          C
74251            160 IF (ICHR.EQ.1HE) GO TO 165
74252                IF (ICHR.EQ.1H+) GO TO 165
74253                IF (ICHR.EQ.1H-) GO TO 165
74254                GO TO 280
74255            165 LCOL=LCOL+1
74256                IF (LCOL.GT.10) GO TO 285
74257                CALL SCHARS (LTIM,LCOL,1,ICHR)
74258                GO TO 20
74259          C
74260          C.....COMMA OR RIGHT BRACKET ENCOUNTERED--DELIMITS RANGE DECLARATIONS
74261          C
74262            170 ENCODE (10,300,FMT) LCOL
74263                DECODE (LCOL,FMT,LTIM) DUM
74264                RBUF(LRANGE,XGRP)=DUM
74265                LTIM=0
74266                LCOL=0
74267                LRANGE=LRANGE+1
74268                IF (LRANGE.LE.2) GO TO 20
74269                IF (RBUF(1,XGRP).LT.RBUF(2,XGRP)) GO TO 180
74270                IF (RBUF(1,XGRP).NE.RBUF(2,XGRP)) GO TO 175
74271                WRITE (XUO,305) CARD
74272                RBUF(2,XGRP)=0.
74273                RBUF(1,XGRP)=RBUF(2,XGRP)
74274                GO TO 20
74275            175 DUM=RBUF(1,XGRP)
74276                RBUF(1,XGRP)=RBUF(2,XGRP)
74277                RBUF(2,XGRP)=DUM
74278            180 IF (XGRP.NE.6) GO TO 20
74279                DUM=RBUF(2,XGRP)-RBUF(1,XGRP)
74280                IF (DUM.GT.XDIF) GO TO 20
74281                XDIF=DUM
74282                XRNG(1)=RBUF(1,XGRP)
74283                XRNG(2)=RBUF(2,XGRP)
74284                GO TO 20
74285            185 IF (NCOL.GT.0) GO TO 95
74286                GO TO 20
74287          C
74288          C.....LEFT BRACKET ENCOUNTERED--INDEPENDENT RANGE DECLARATION
74289          C
74290            190 INDEP=.TRUE.
74291                XGRP=6
74292                GO TO 155
```

Line Number	Explanation
74234-74240	A slash encountered indicates that an independent variable (probably not TIME) was specified. Begin parsing of independent variable. Independent variable information is stored at IBUF(I,J,XGRP) where XGRP=6. INDEP = .TRUE. indicates that information being parsed pertains to independent variable.
74244-74247	A left bracket is encountered within a dependent group. Initialize variables which will contain the parsed range values. LTIM will contain the characters of a range value. LCØL is a count of the characters in LTIM. LRANGE indicates which range value is currently being parsed (LRANGE=1, then first value is being parsed).

Line Number	Explanation
74248-74254	The character encountered is a part of range values and is not a numeric character or a dot. The only legal characters in this category are 1HE, 1H+, or 1H-. Any other character signals a parse error.
74255-74258	The character encountered is a numeric character or a 1H., 1HE, 1H+, or 1H-. Store character in LTIM. (A range value must consist of 10 or fewer values, LCØL≤10--or else an error exists.)
74259-74267	The comma or right bracket encountered delimits range values. The characters in LTIM are encoded into a floating point value and this value is stored in RBUF. (RBUF(I,J) contains the Ith range value for the Jth group if nonzero). Reinitialize variables and look for next range value.
74268-74277	LRANGE>2 indicates at parsing of range values is complete. If both values in RBUF are equal, an error condition exists; if RBUF(1,XGRP)>RBUF(2,XGRP), switch the two.
74278-74284	XGRP=6 indicates that the range specification is for the independent variable. The difference between range values is calculated (DUM); if this difference is less than any other difference between range values (for the independent variable), store the values in XRNG and the difference in XDIF.
74285-74286	A right paren or a comma is encountered following a right bracket. If NCØL>0, the character delimits a variable name.
74287-74292	A left bracket is encountered indicating that range values are specified for the independent variable.

Determine ID characters

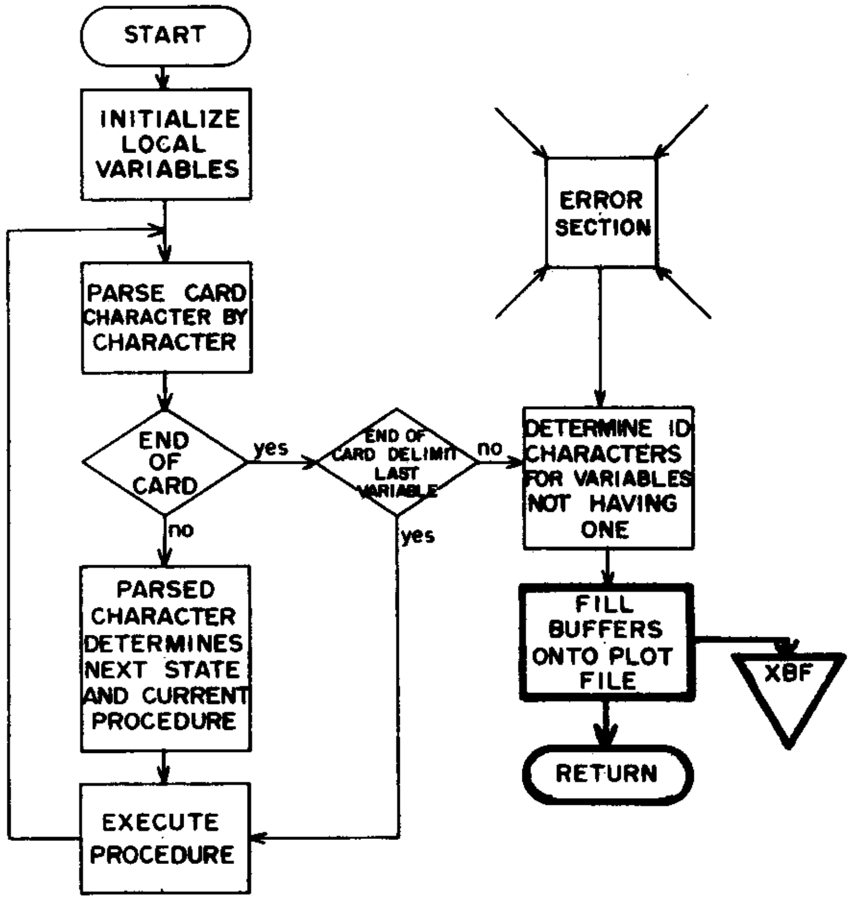

START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
END OF CARD DELIMIT LAST VARIABLE
no
yes
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

```
C
C.....END OF CARD--DELIMITS INDEPENDENT VARIABLE
C
   90 IF (.NOT.INDEP) GO TO 195
      IF (NCOL.EQ.0) GO TO 195
```

```
  195 IBUF(1,1,6)=10HTIME
      IBUF(3,1,6)=1
C
C.....FILL CHARACTERS FOR ANY VARIABLES NOT SPECIFIED
C
  200 KNT=0
      DO 230 K=1,5
         IF (IBUF(1,1,K).EQ.0) GO TO 232
         DO 225 J=1,5
            IF (IBUF(1,J,K).EQ.0) GO TO 230
            ICHR=IBUF(2,J,K).AND.77B
            IF (ICHR.NE.0) GO TO 225
  205       KNT=KNT+1
            DO 215 N=1,5
               IF (IBUF(1,1,N).EQ.0) GO TO 220
               DO 210 M=1,5
                  IF (IBUF(1,M,N).EQ.0) GO TO 215
                  ICHR=IBUF(1,M,M).AND.77B
                  IF (ICHR.EQ.KNT) GO TO 205
  210          CONTINUE
  215       CONTINUE
  220       IBUF(2,J,K)=IBUF(2,J,K).OR.KNT
  225    CONTINUE
  230 CONTINUE
```

Line Number	Explanation
74166-74170; 74296-74297	End of card is encountered (delimits the independent variable). If INDEP=.FALSE. (or NCØL=0), then an independent variable was not specified and parsing is finished. (TIME is filled into IBUF as independent variable.) Otherwise, proceed to line 74171 and create entries in plot stack for independent variable.
74298-74319	Assign ID characters for any variables not assigned a character by the user. IBUF is searched for a nonzero entry in IBUF(I,J,K) having a zero where the character should be stored. Set KNT=1=IRA, and all characters in IBUF are checked against KNT; if none are equal to KNT, KNT is unique and is assigned as the plot character for the first variable name not having a character specified. Otherwise, increment KNT (KNT=2=IRB) and compare KNT with all existing ID characters, etc., until a unique KNT is determined. Continue process for all remaining variables not having ID characters.

Flush buffer

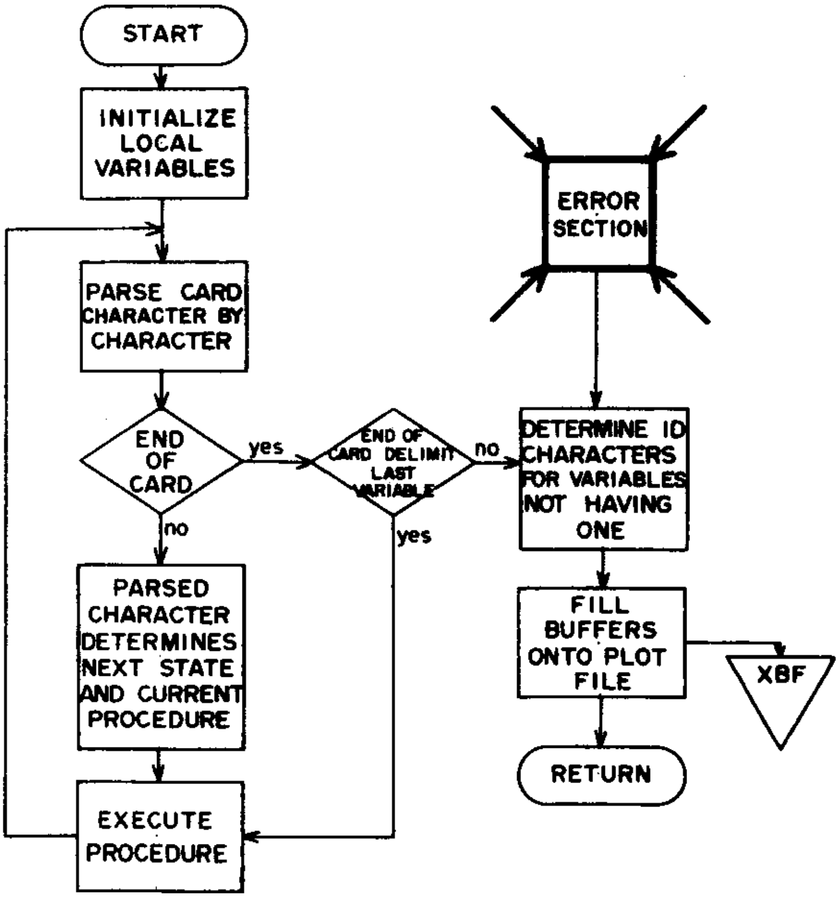

START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

```
74320          C
74321          C.....IF INDEPENDENT VARIABLE IS TIME, AND NO RANGES WERE SPECIFIED SET
74322          C.....XRNG FLAGS FOR ROUTINE XCSTART
74323          C
74324            232 IF (IBUF(3,1,6).NE.1) GO TO 235
74325                IF (XRNG(1).NE.XRNG(2)) GO TO 235
74326                XRNG(2)=1.
74327                XRNG(1)=XRNG(2)
74328            235 WRITE (XBF) IBUF,RBUF
74329                XPLFG=.TRUE.
74330                RETURN
```

Line Number	Explanation
74324-74327	XRNG is a flag to routine XCSTART used to determine the optimum DTPL. XRNG=0 (initial value) if no range declarations for the independent variable were specified and TIME is never the independent variable. XRNG=1 if no range declarations for the independent variable were specified and TIME is the independent variable of at least one plot.

Line Number	Explanation
	XRNG(1)≠XRNG(2); XRNG contains the values of the independent range declaration having the smallest difference between values.
74328-74330	Write IBUF and RBUF onto plot file XBF which will be used in the construction of plotted output. XPLFG=.TRUE. indicates that plot requests are present.

Diagnostics

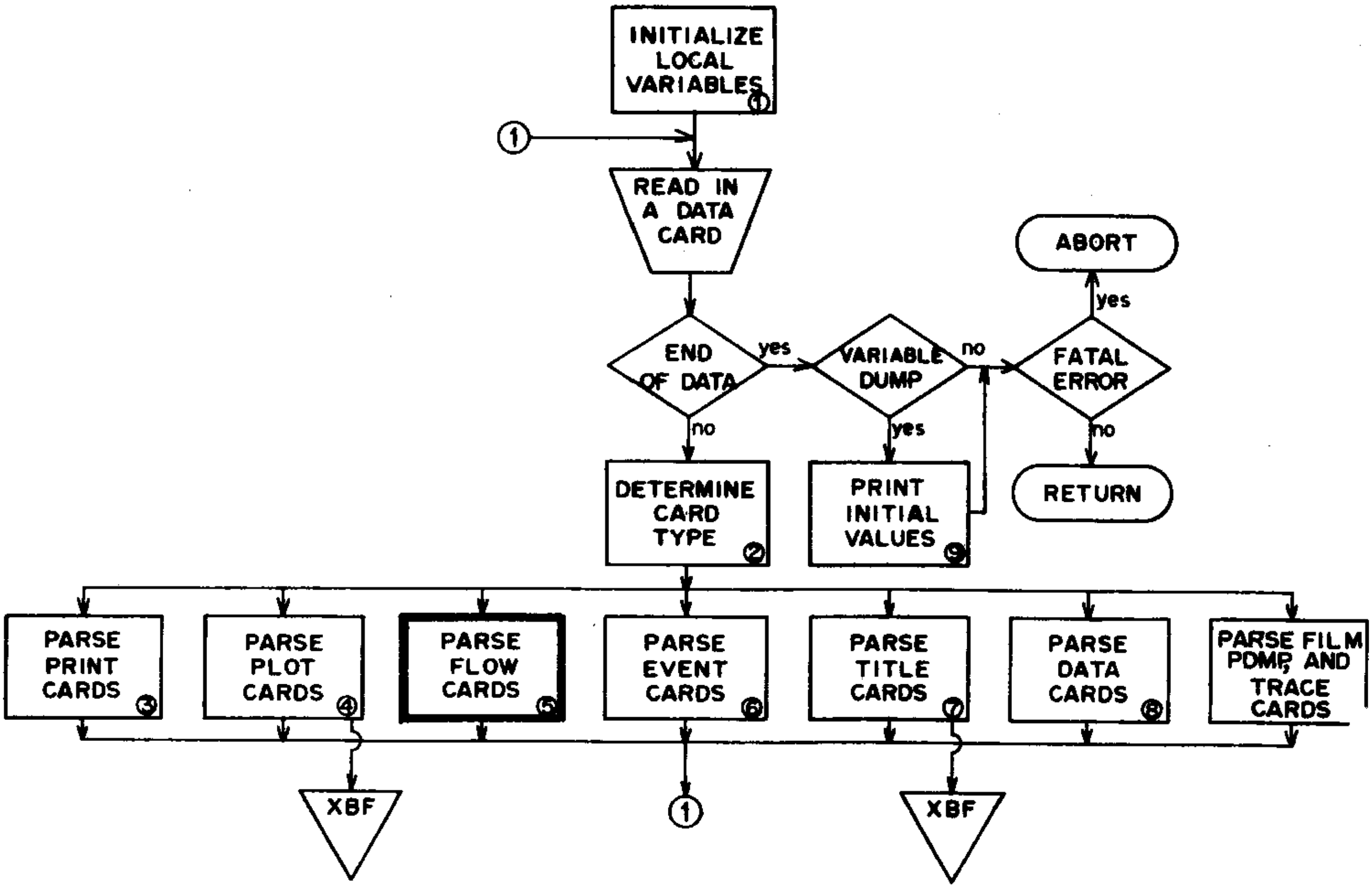

START
INITIALIZE LOCAL VARIABLES
PARSE CARD CHARACTER BY CHARACTER
END OF CARD
yes
no
END OF CARD DELIMIT LAST VARIABLE
no
yes
PARSED CHARACTER DETERMINES NEXT STATE AND CURRENT PROCEDURE
EXECUTE PROCEDURE
ERROR SECTION
DETERMINE ID CHARACTERS FOR VARIABLES NOT HAVING ONE
FILL BUFFERS ONTO PLOT FILE
XBF
RETURN

```
C
C.....ERROR SECTION
C
  245 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,330)
      GO TO 195
  250 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,335)
      GO TO 195
  255 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,340)
  260 IF (XGRP.GT.1) GO TO 195
      IF (NVAR.GT.1) GO TO 195
      WRITE (XUO,325)
      RETURN
  265 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,345)
      GO TO 260
  270 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,350)
      GO TO 260
  275 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,355)
      GO TO 260
  280 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,360)
      GO TO 260
  285 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
      WRITE (XUO,365)
      GO TO 260
```

```
74361          290 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
74362              WRITE (XUO,370) NAME
74363              GO TO 260
74364          295 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
74365              WRITE (XUO,375) ICHR
74366              GO TO 260
74367          297 WRITE (XUO,320) ICOL,(I,I=1,8),CARD
74368              WRITE (XUO,380)
74369              QPLTS=.TRUE.
74370              RETURN
74371        C
74372          300 FORMAT (2H(E,I2,6H.0)    )
74373          305 FORMAT (6H0*****, 23HERROR IN PLOT REQUEST--,8A10,/,T14, 43HA RANG
74374             1E DECLARATION HAS IDENTICAL ENDPOINTS)
74375          320 FORMAT (6H0*****, 51HERROR IN PLOT REQUEST--PROCESSING HALTED AT C
74376             1OLUMN ,I2//,T20,8I10/T20,8(10H1234567890),/,T20,8A10)
74377          325 FORMAT (1H ,5X, 26HPLOT REQUEST NOT PROCESSED)
74378          330 FORMAT (1H0,T14, 32HNO. OF GROUPS PER PLOT IS .GT. 5)
74379          335 FORMAT (1H0,T14, 34HNO. OF VARIABLES PER GROUP IS .GT. 5)
74380          340 FORMAT (1H0,T14, 34HVARIABLE NAME IS .GT. 5 CHARACTERS)
74381          345 FORMAT (1H0,T14, 44HVARIABLE SUBSCRIPT .GT. 999--THE UPPER LIMIT)
74382          350 FORMAT (1H0,T14, 30HVARIABLE HAS .GT. 3 SUBSCRIPTS)
74383          355 FORMAT (1H0,T14, 32HIMPROPERLY FORMATTED LOG REQUEST)
74384          360 FORMAT (1H0,T14, 38HILLEGAL CHARACTER IN RANGE DECLARATION)
74385          365 FORMAT (1H0,T14, 53HRANGE DECLARATION .GT. 10 CHARACTERS--THE UPPE
74386             1R LIMIT)
74387          370 FORMAT (1H0,T14,  9HVARIABLE ,A5, 24H NOT DECLARED IN STORAGE)
74388          375 FORMAT (1H0,T14, 28HILLEGAL CHARACTER DETECTED ",A1,  1H")
74389          380 FORMAT (1H0,T14,89HMORE THAN 100 VARIABLES NAMED IN PLOT REQUESTS,
74390             1 THIS AND SUBSEQUENT PLOT REQUESTS IGNORED)
74391        C
74392              END
```

Line Number	Explanation
74331-74392	When an error occurs, parsing stops and a diagnostic is issued. The contents of the plot card (up to the error) is loaded onto the plot file and will be plotted.

2.5 Parse FLOW. Cards



Overview

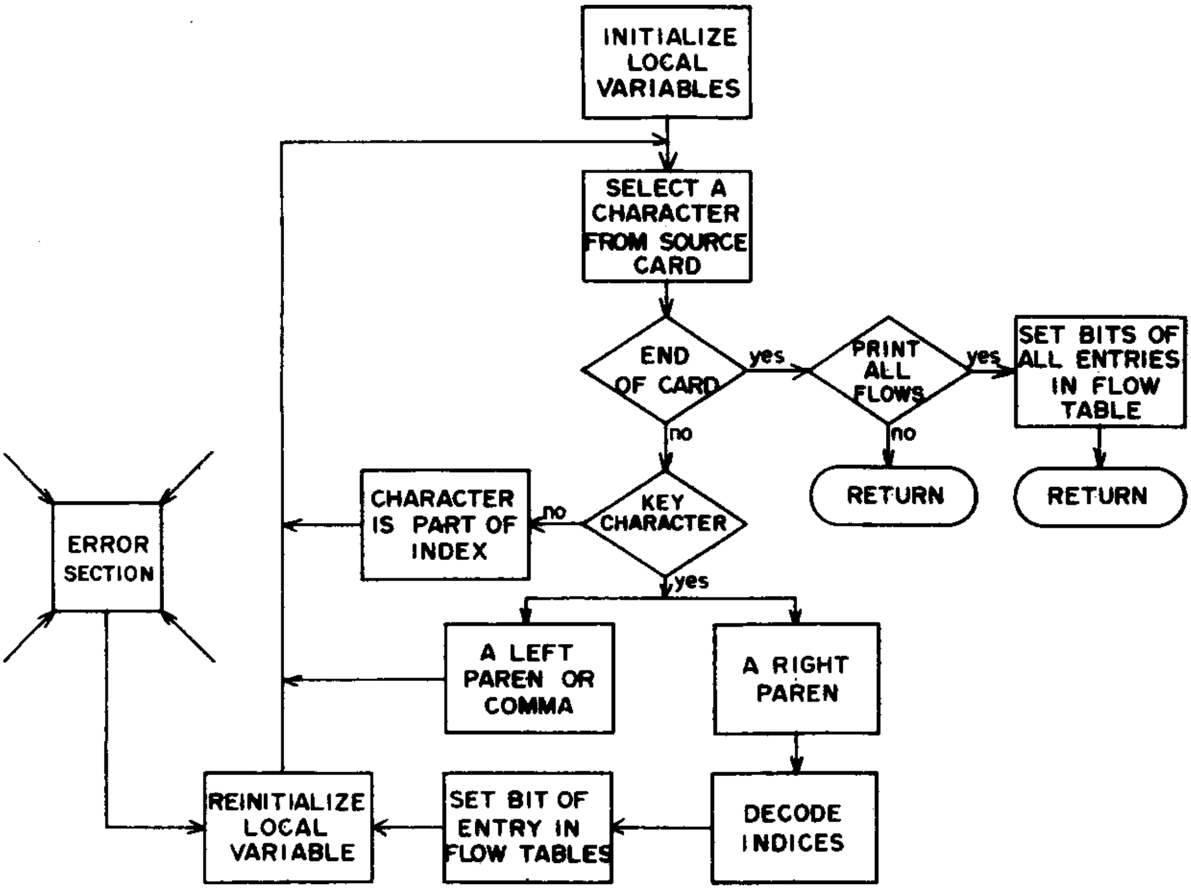
This section parses data cards of the form

FLØW. <flow1>,<flow2>···

where <flow1>: = (A,B). . A and B are indices of state variables, defining a flow from state variable A to B. <flow1> must have been defined in the source deck section by a flow command of the form (A-B). which created an entry for that flow in the flow table, XFLWT. For each <flow1> encountered on the data card, the entry corresponding to that flow is located in the flow stack and the low order bit of that entry is set to 1 (used by a later section to determine if the value of that flow is to be written onto output). XFLPR is set to .TRUE. indicating that there are flow values to be printed.

The section is subdivided for easier analysis.

FLØW. cards flow chart

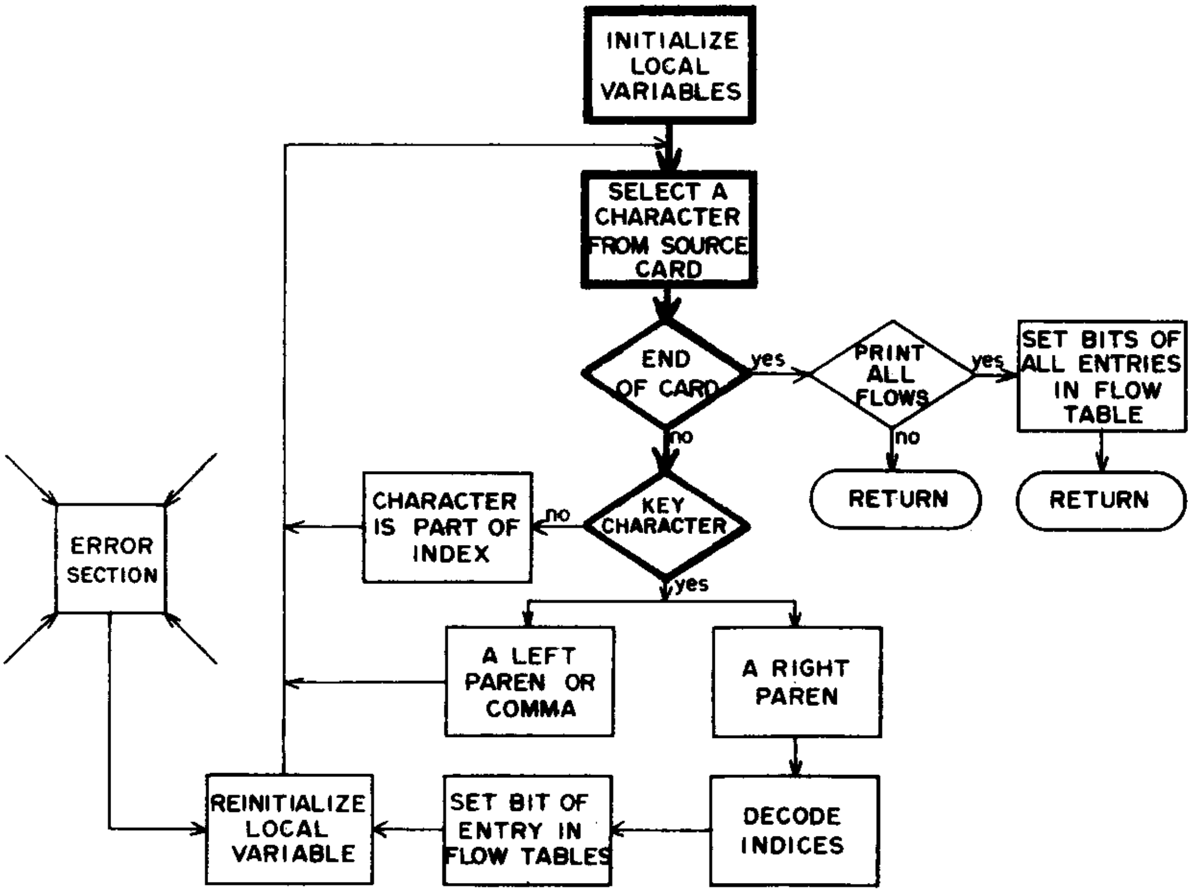

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
no
RETURN
RETURN
no
KEY CHARACTER
no
CHARACTER IS PART OF INDEX
ERROR SECTION
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
DECODE INDICES
SET BIT OF ENTRY IN FLOW TABLES
REINITIALIZE LOCAL VARIABLE

```
      SUBROUTINE XFLSTK (CARD,JCOL)
      COMMON /XXFL1WS/ XNFLW,XFLWT(1)
      COMMON /XXFL2WS/ XFLW(1)
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      DIMENSION CARD(8), KEY(3), NCOL(2), INDX(2), N1(2)
      INTEGER XNFLW,XFLWT,XUO,XUI
      LOGICAL XFLPR
      LOGICAL ALL
C
C.....THIS ROUTINE PARSES FLOW PRINT REQUEST CARDS, SETTING THE LOW ORDE
C.....BIT IN THE ENTRY IN THE FLOW REFERENCE TABLE CONTAINING THE INDICE
C.....OF THE FLOW TO BE PRINTED.  REQUESTS FOR THE PRINTING OF NONEXIST-
C.....ING FLOWS ISSUE A DIAGNOSTIC BUT ARE OTHERWISE IGNORED.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H(,1H,,1H)/,NK/3/
      NCOL(2)=0
      NCOL(1)=NCOL(2)
      INDX(2)=10H
      INDX(1)=INDX(2)
      N1(2)=0
      N1(1)=N1(2)
      KODE=3
      ALL=.TRUE.
      ICOL=JCOL
C
C.....SCAN THE CARD FOR KEY CHARACTERS.
C
   15 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 75
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 15
      ALL=.FALSE.
      DO 20 I=1,NK
         IF (ICHR.EQ.KEY(I)) GO TO 30
   20 CONTINUE
C
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,25,85), KODE
C
C....."ICHR" IS ASSUMED PART OF AN INDEX.
C
   25 IF (ICHR.LT.1H0.O.ICHR.GT.1H9) GO TO 85
      J=KODE
      NCOL(J)=NCOL(J)+1
      IF (NCOL(J).GT.3) GO TO 90
      CALL SCHARS (INDX(J),NCOL(J),1,ICHR)
      GO TO 15
C
C....."ICHR" IS A KEY CHARACTER.
C
   30 GO TO (35,40,50), I
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   35 IF (KODE.NE.3) GO TO 85
      KODE=1
      GO TO 15
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1) GO TO 45
      IF (KODE.EQ.3) GO TO 15
      GO TO 85
   45 KODE=2
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   50 IF (KODE.NE.2) GO TO 85
C
C.....DECODE THE INDICES.
C
      DO 55 I=1,2
         IF (NCOL(I).LE.0) GO TO 95
         CALL GNUM (INDX(I),1,NCOL(I),N1(I),IERR)
         IF (IERR.NE.0) GO TO 100
         IF (N1(I).LT.1.O.N1(I).GT.999) GO TO 100
   55 CONTINUE
C
C.....FIND ENTRY OF FLOW REQUESTED FOR PRINT IN THE FLOW REFERENCE TABLE
C
      IF (XNFLW.LE.0) GO TO 105
      DO 60 I=1,XNFLW
         CALL GBYTE (XFLWT(I),M1,30,15)
         CALL GBYTE (XFLWT(I),M2,45,15)
         IF (M1.EQ.N1(1).A.M2.EQ.N1(2)) GO TO 65
   60 CONTINUE
      GO TO 110
C
C.....REQUESTED FLOW FOUND IN TABLE, SET LOW ORDER BIT FOR THIS ENTRY TO
C.....A ONE (1).
C
```

```
   65 CALL SBYTE (XFLWT(I),1,0,1)
      XFLPR=.TRUE.
C
C.....REINITIALIZE LOCAL VARIABLES.
C
   70 KODE=3
      NCOL(2)=0
      NCOL(1)=NCOL(2)
      INDX(2)=10H
      INDX(1)=INDX(2)
      N1(2)=0
      N1(1)=N1(2)
      GO TO 15
C
C.....END OF CARD, CHECK FOR REQUEST FOR PRINTING OF ALL EXISTING FLOWS.
C
   75 IF (KODE.NE.3) GO TO 115
      IF (.N.ALL) RETURN
      IF (XNFLW.LE.0) GO TO 105
      DO 80 I=1,XNFLW
   80 CALL SBYTE (XFLWT(I),1,0,1)
      XFLPR=.TRUE.
      RETURN
C
C.....IF ERRORS OCCURED GENERATE A DIAGNOSTIC.
C
   85 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,145) ICHR,ICOL
      GO TO 120
   90 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,150) INDX(J)
      GO TO 120
   95 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,155) ICOL
      GO TO 120
  100 WRITE (XUO,140) (J,J=1,8),CARD
      WRITE (XUO,160) INDX(I)
      GO TO 120
  105 WRITE (XUO,140) (J,J=1,8),CARD
      WRITE (XUO,165)
      RETURN
  110 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,170) (N1(I),I=1,2)
      GO TO 70
  115 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,175)
      RETURN
C
C.....FLOW BRANCHES HERE AFTER AN ERROR IS ENCOUNTERED.
C.....THE SEGMENT DISCARDS A FLOW CONTAINING AN ERROR AND SEARCHES THE
C.....CARD FOR THE NEXT FLOW.
C
  120 ICOL=ICOL-1
      MOR=1
  125 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 75
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICOL.EQ.1H ) GO TO 125
      IF (MOR.EQ.1) GO TO 130
      IF (ICHR.NE.1H,) GO TO 125
      MOR=1
      GO TO 125
  130 IF (ICHR.EQ.1H() GO TO 135
      MOR=0
      GO TO 125
  135 ICOL=ICOL-1
      GO TO 70
C
  140 FORMAT (6H0*****, 27HERROR IN FLOW PRINT REQUEST,//T20,8I10/T20,8(
     1 10H1234567890)/T20,8A10)
```

```
      145 FORMAT (1H0,T14, 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLUMN ,I2)
      150 FORMAT (1H0,T14, 11HFLOW INDX ",A3, 24H..." LONGER THAN 3 CHARS)
      155 FORMAT (1H0,T14, 43HZERO LENGTH FLOW INDEX IN OR BEFORE COLUMN ,I2
     1)
      160 FORMAT (1H0,T14, 12HFLOW INDEX ",A3, 31H" NOT DECODABLE OR OUT OF
     1RANGE)
      165 FORMAT (1H0,T14, 42HFLOW PRINTING REQUESTED - NO FLOWS DEFINED)
      170 FORMAT (1H0,T14,  6HFLOW (,I2,  1H,,I2, 16H) DOES NOT EXIST)
      175 FORMAT (1H0,T14, 37HFLOW INDICES UNTERMINATED AT CARD END)
C
          END
```

Initialize and retrieve a character

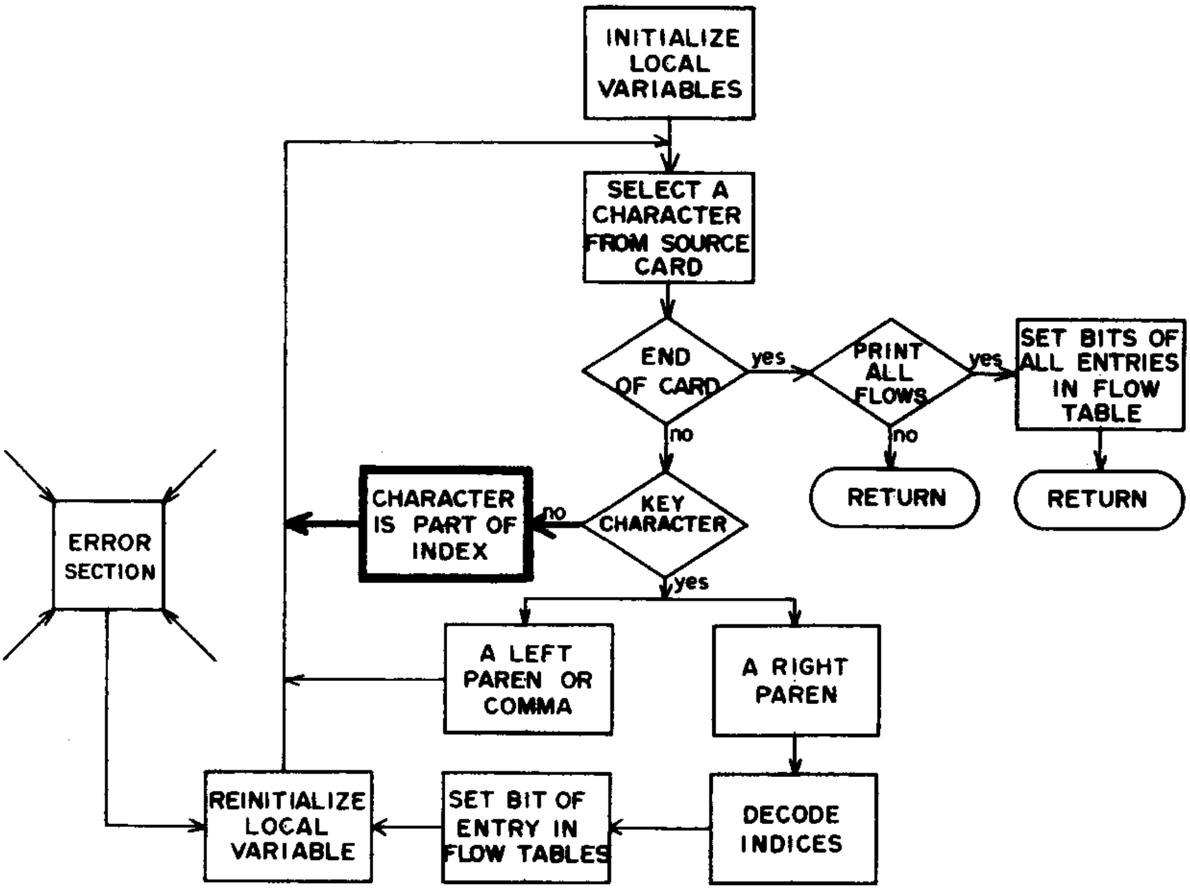

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
no
RETURN
RETURN
no
KEY CHARACTER
no
CHARACTER IS PART OF INDEX
ERROR SECTION
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
DECODE INDICES
SET BIT OF ENTRY IN FLOW TABLES
REINITIALIZE LOCAL VARIABLE

```
      SUBROUTINE XFLSTK (CARD,JCOL)
      COMMON /XXFL1WS/ XNFLW,XFLWT(1)
      COMMON /XXFL2WS/ XFLW(1)
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      DIMENSION CARD(8), KEY(3), NCOL(2), INDX(2), N1(2)
      INTEGER XNFLW,XFLWT,XUO,XUI
      LOGICAL XFLPR
      LOGICAL ALL
C
C.....THIS ROUTINE PARSES FLOW PRINT REQUEST CARDS, SETTING THE LOW ORDE
C.....BIT IN THE ENTRY IN THE FLOW REFERENCE TABLE CONTAINING THE INDICE
C.....OF THE FLOW TO BE PRINTED.  REQUESTS FOR THE PRINTING OF NONEXIST-
C.....ING FLOWS ISSUE A DIAGNOSTIC BUT ARE OTHERWISE IGNORED.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA KEY/1H(,1H,,1H)/,NK/3/
      NCOL(2)=0
      NCOL(1)=NCOL(2)
```

```
71020              INDX(2)=10H
71021              INDX(1)=INDX(2)
71022              N1(2)=0
71023              N1(1)=N1(2)
71024              KODE=3
71025              ALL=.TRUE.
71026              ICOL=JCOL
71027        C
71028        C.....SCAN THE CARD FOR KEY CHARACTERS.
71029        C
71030           15 ICOL=ICOL+1
71031              IF (ICOL.GT.80) GO TO 75
71032              CALL GCHARS (CARD,ICOL,1,ICHR)
71033              IF (ICHR.EQ.1H ) GO TO 15
71034              ALL=.FALSE.
71035              DO 20 I=1,NK
71036                 IF (ICHR.EQ.KEY(I)) GO TO 30
71037           20 CONTINUE
```

Line Number	Explanation
71000-71026	Initialize the local variables. CARD contains the data card image. JCØL is the number of the column preceding the text portion of the data card. (FLØW.(12,3) JCØL would point at the period following FLØW and the text portion is (12,3). KEY contains a list of the delimiters encountered. NCØL(J) will contain the number of characters in the Jth index. INDX(J) contains the characters of the Jth index. N1(J) contains the integer value of the characters in INDX(J) KØDE is an indicator of which KEY character has been discovered. ALL is a logical flag, indicating (if .TRUE.) that all existing flows are to be printed. This occurs if the FLØW. command appears with no text following. ICØL is the column number currently being scanned.
71027-71037	Scan the card for key characters. A character is retrieved from CARD and checked to see if it is a KEY character.

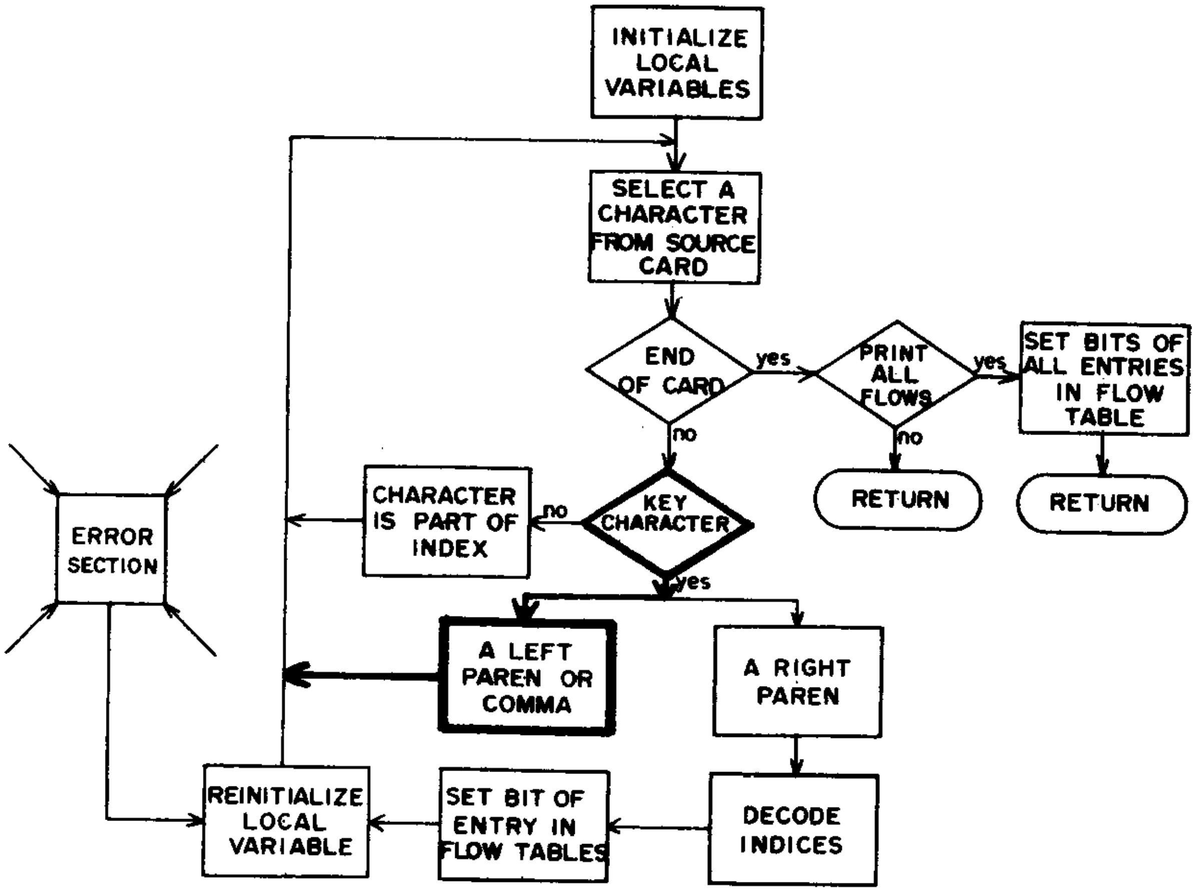
Character is part of an index


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
no
no
RETURN
RETURN
ERROR SECTION
CHARACTER IS PART OF INDEX
no
KEY CHARACTER
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
REINITIALIZE LOCAL VARIABLE
SET BIT OF ENTRY IN FLOW TABLES
DECODE INDICES

```
C
C....."ICHR" IS NOT A KEY CHARACTER.
C
      GO TO (25,25,85), KODE
C
C....."ICHR" IS ASSUMED PART OF AN INDEX.
C
   25 IF (ICHR.LT.1H0.O.ICHR.GT.1H9) GO TO 85
      J=KODE
      NCOL(J)=NCOL(J)+1
      IF (NCOL(J).GT.3) GO TO 90
      CALL SCHARS (INDX(J),NCOL(J),1,ICHR)
      GO TO 15
```

Line Number	Explanation
71038-71050	The character is not a key character. KØDE is a counter of the key characters encountered. KØDE=3 Initial value. KØDE=1 If a left paren was encountered. =2 If a comma was encountered. Therefore a value of 1 or 2 indicates that the character must be a part of an index. NCØL(J) is the number of characters in the Jth index. INDX(J) is the characters of the Jth index. EXAMPLE. FLØW.(99,2) causes NCØL(1)=2 NCØL(2)=1 INDX(1)=2H99 INDX(2)=1H2

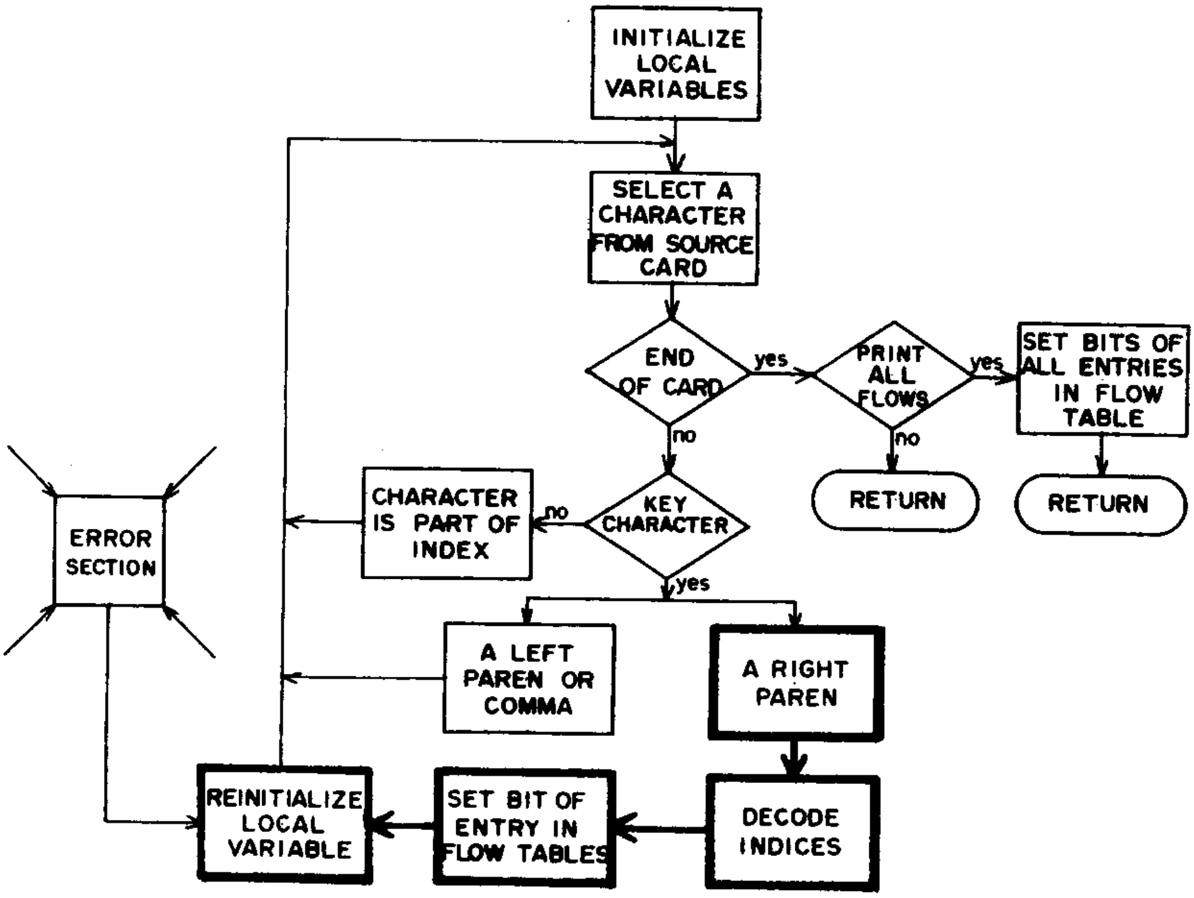
Process subscript delimiter


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
RETURN
no
RETURN
no
ERROR SECTION
CHARACTER IS PART OF INDEX
no
KEY CHARACTER
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
REINITIALIZE LOCAL VARIABLE
SET BIT OF ENTRY IN FLOW TABLES
DECODE INDICES

```
C
C....."ICHR" IS A KEY CHARACTER.
C
   30 GO TO (35,40,50), I
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   35 IF (KODE.NE.3) GO TO 85
      KODE=1
      GO TO 15
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   40 IF (KODE.EQ.1) GO TO 45
      IF (KODE.EQ.3) GO TO 15
      GO TO 85
   45 KODE=2
      GO TO 15
```

Line Number	Explanation
71051-71068	(Enter from line 71036) KEY characters are the delimiters of indices. A left paren sets KØDE=1 and looks for first index. A comma delimits the first and second index; therefore set KØDE=2 and look for second index. A value of KØDE=3 indicates that the comma encountered delimits flows; therefore, search for left paren beginning next flow (i.e., FLØW.(1,2),(3,4)).

Locate flow in table

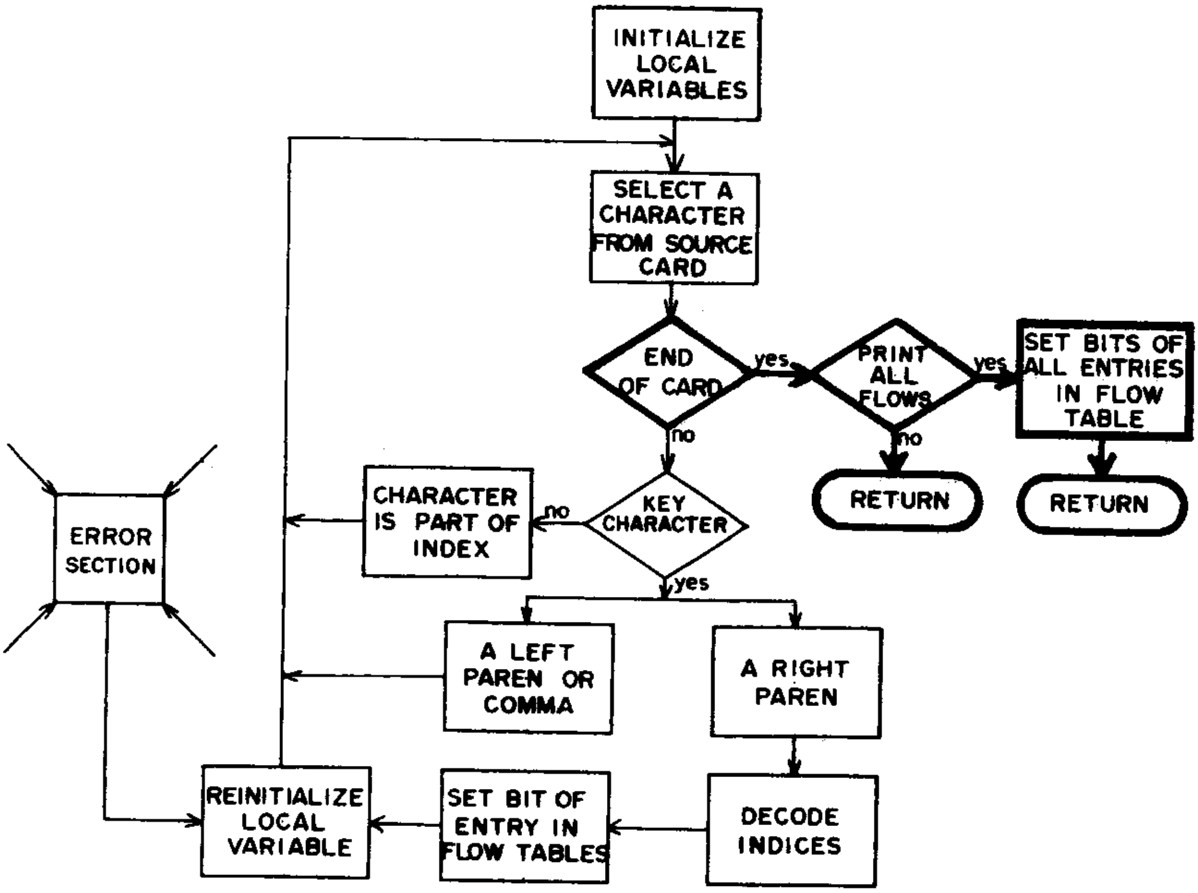

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
no
no
RETURN
RETURN
ERROR SECTION
CHARACTER IS PART OF INDEX
no
KEY CHARACTER
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
REINITIALIZE LOCAL VARIABLE
SET BIT OF ENTRY IN FLOW TABLES
DECODE INDICES

```
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   50 IF (KODE.NE.2) GO TO 85
C
C.....DECODE THE INDICES.
C
      DO 55 I=1,2
         IF (NCOL(I).LE.0) GO TO 95
         CALL GNUM (INDX(I),1,NCOL(I),N1(I),IERR)
         IF (IERR.NE.0) GO TO 100
         IF (N1(I).LT.1.O.N1(I).GT.999) GO TO 100
   55 CONTINUE
C
C.....FIND ENTRY OF FLOW REQUESTED FOR PRINT IN THE FLOW REFERENCE TABLE
C
      IF (XNFLW.LE.0) GO TO 105
      DO 60 I=1,XNFLW
         CALL GBYTE (XFLWT(I),M1,30,15)
         CALL GBYTE (XFLWT(I),M2,45,15)
         IF (M1.EQ.N1(1).A.M2.EQ.N1(2)) GO TO 65
   60 CONTINUE
      GO TO 110
C
C.....REQUESTED FLOW FOUND IN TABLE, SET LOW ORDER BIT FOR THIS ENTRY TO
```

```
71094          C.....A ONE (1).
71095          C
71096             65 CALL SBYTE (XFLWT(I),1,0,1)
71097                XFLPR=.TRUE.
71098          C
71099          C.....REINITIALIZE LOCAL VARIABLES.
71100          C
71101             70 KODE=3
71102                NCOL(2)=0
71103                NCOL(1)=NCOL(2)
71104                INDX(2)=10H
71105                INDX(1)=INDX(2)
71106                N1(2)=0
71107                N1(1)=N1(2)
71108                GO TO 15
71109          C
```

Line Number	Explanation
71072	Right paren encountered (KØDE≠2 indicates error condition).
71076-71081	N1(J) is filled with the integer values of the characters in INDX(J)
71085-71091	The flow table, XFLWT, is searched for a match with the indices in N1.
71096-71097	A match is found; set the low order bit to a 1 and set XFLPR=.TRUE., indicating that there are flows to be printed.
	EXAMPLE. Assume that the flow command (1-3). appeared in the source deck section. This would create an entry in the flow stack, XFLWT. NFLT(I)=XFLWT(I)=000...100003$_8$ The command FLØW.(1,3) appears in the data section, indicating that the value of the flow from X(1)-X(3) is to be written onto the output file for each predetermined time increment. This data card would assign the following values to local variables. N1(1)=1 N1(2)=3 A match occurs with the values of N1 and XFLWT(I) setting a sign bit in XFLWT(I): XFLWT(I)=4000...100003$_8$ The sign bit will alert other sections to print out the value of the flow associated with that entry in the flow tables.
71101-71108	Reinitialize local variables and continue to search for other flows on the data card.

Set flag bits in flow table

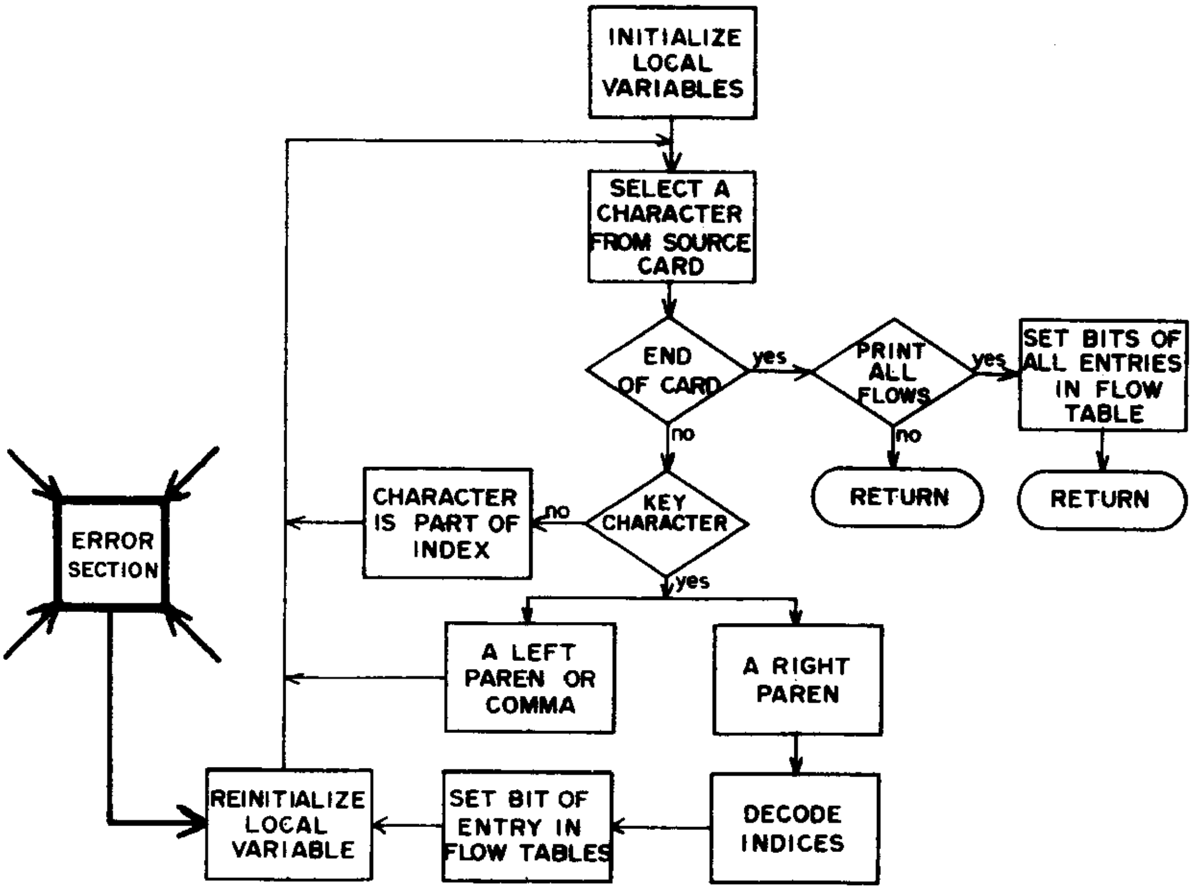

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
RETURN
no
no
RETURN
ERROR SECTION
CHARACTER IS PART OF INDEX
no
KEY CHARACTER
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
REINITIALIZE LOCAL VARIABLE
SET BIT OF ENTRY IN FLOW TABLES
DECODE INDICES

```
71110          C.....END OF CARD. CHECK FOR REQUEST FOR PRINTING OF ALL EXISTING FLOWS.
71111          C
71112             75 IF (KODE.NE.3) GO TO 115
71113                IF (.N.ALL) RETURN
71114                IF (XNFLW.LE.0) GO TO 105
71115                DO 80 I=1,XNFLW
71116             80 CALL SBYTE (XFLWT(I),1,0,1)
71117                XFLPR=.TRUE.
71118                RETURN
```

Line Number	Explanation
71112-71118	Enter from line 71031. If ALL=.TRUE., then a FLØW. command appeared on a card without any text (no specific flows specified). This is a request to print all existing flows. The sign bit is set for every flow entry (XNFLW entries) in the flow table, XFLWT.

Line Number	Explanation
	XFLPR=.TRUE. alerting a later section that flow values are to be printed.

Diagnostics

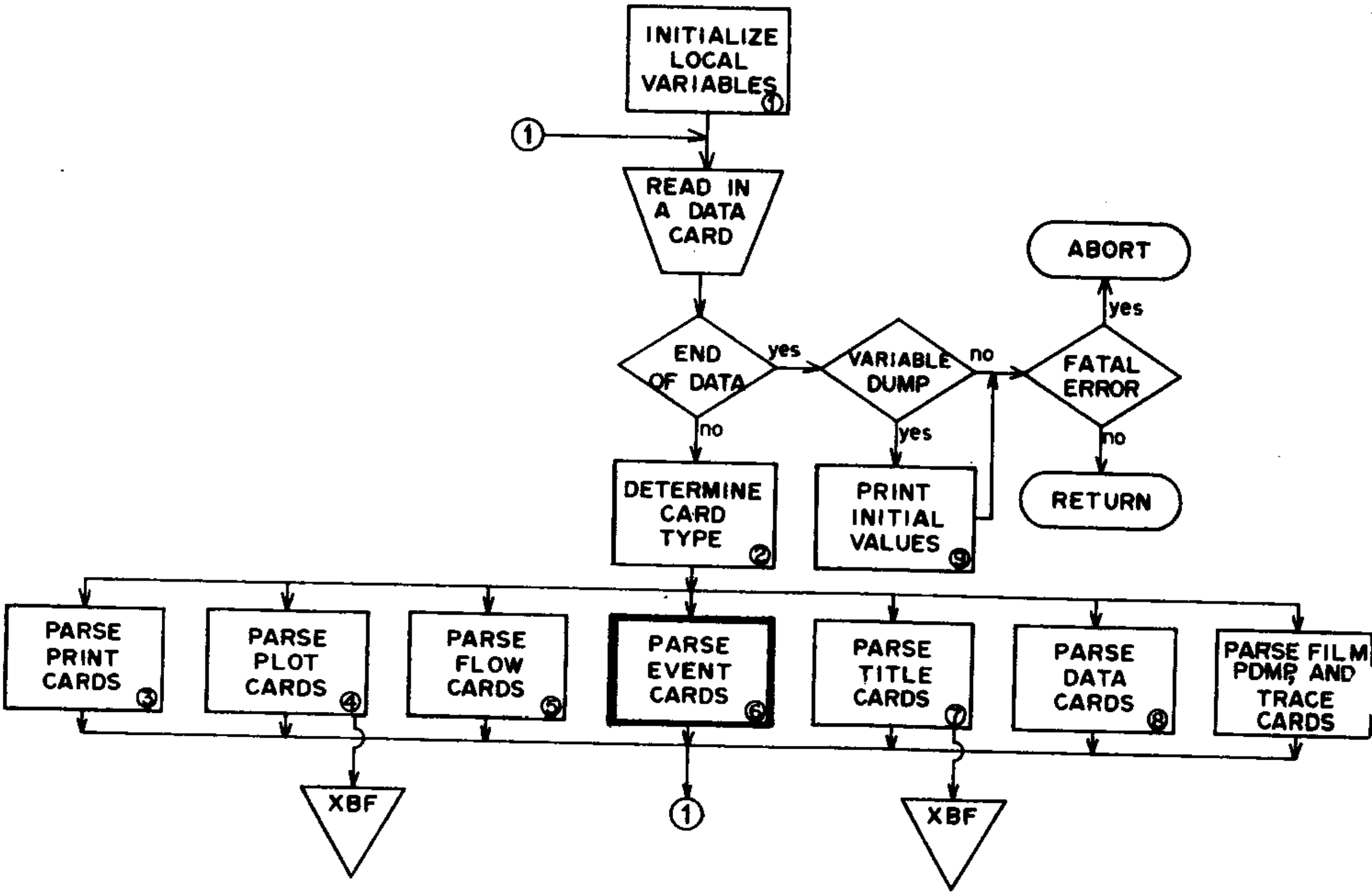

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
PRINT ALL FLOWS
yes
SET BITS OF ALL ENTRIES IN FLOW TABLE
RETURN
no
RETURN
no
KEY CHARACTER
no
CHARACTER IS PART OF INDEX
ERROR SECTION
yes
A LEFT PAREN OR COMMA
A RIGHT PAREN
REINITIALIZE LOCAL VARIABLE
SET BIT OF ENTRY IN FLOW TABLES
DECODE INDICES

```
C
C.....IF ERRORS OCCURED GENERATE A DIAGNOSTIC.
C
   85 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,145) ICHR,ICOL
      GO TO 120
   90 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,150) INDX(J)
      GO TO 120
   95 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,155) ICOL
      GO TO 120
  100 WRITE (XUO,140) (J,J=1,8),CARD
      WRITE (XUO,160) INDX(I)
      GO TO 120
  105 WRITE (XUO,140) (J,J=1,8),CARD
      WRITE (XUO,165)
      RETURN
  110 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,170) (N1(I),I=1,2)
      GO TO 70
  115 WRITE (XUO,140) (I,I=1,8),CARD
      WRITE (XUO,175)
      RETURN
C
C.....FLOW BRANCHES HERE AFTER AN ERROR IS ENCOUNTERED.
C.....THE SEGMENT DISCARDS A FLOW CONTAINING AN ERROR AND SEARCHES THE
C.....CARD FOR THE NEXT FLOW.
```

```
71147          C
71148            120 ICOL=ICOL-1
71149                MOR=1
71150            125 ICOL=ICOL+1
71151                IF (ICOL.GT.80) GO TO 75
71152                CALL GCHARS (CARD,ICOL+1,ICHR)
71153                IF (ICOL.EQ.1H ) GO TO 125
71154                IF (MOR.EQ.1) GO TO 130
71155                IF (ICHR.NE.1H,) GO TO 125
71156                MOR=1
71157                GO TO 125
71158            130 IF (ICHR.EQ.1H() GO TO 135
71159                MOR=0
71160                GO TO 125
71161            135 ICOL=ICOL-1
71162                GO TO 70
71163          C
71164            140 FORMAT (6H0*****, 27HERROR IN FLOW PRINT REQUEST,//T20,8I10/T20,8(
71165               1 10H1234567890)/T20,8A10)
71166            145 FORMAT (1H0,T14, 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLUMN ,I2)
71167            150 FORMAT (1H0,T14, 11HFLOW INDX ",A3, 24H..." LONGER THAN 3 CHARS)
71168            155 FORMAT (1H0,T14, 43HZERO LENGTH FLOW INDEX IN OR BEFORE COLUMN ,I2
71169               1)
71170            160 FORMAT (1H0,T14, 12HFLOW INDEX ",A3, 31H" NOT DECODABLE OR OUT OF
71171               1RANGE)
71172            165 FORMAT (1H0,T14, 42HFLOW PRINTING REQUESTED - NO FLOWS DEFINED)
71173            170 FORMAT (1H0,T14,  6HFLOW (,I2,  1H,,I2, 16H) DOES NOT EXIST)
71174            175 FORMAT (1H0,T14, 37HFLOW INDICES UNTERMINATED AT CARD END)
71175          C
```

Line Number	Explanation
71119-71142	If a syntax error occurs on a data card, a nonfatal diagnostic is issued.
71148-71162	An error in a flow causes processing of that flow to be rejected (it will not be printed). A search is made for the beginning of the first flow following the error. The column pointer is positioned, and processing for that flow begins. A comma followed by a left paren indicates the beginning of a new flow. ICØL is positioned at the column preceding the flow beginning (at the comma) and processing begins for the new flow.

2.6 Parse EVENT. Statements



Overview

This section processes data statements of the form:

EVENT.<name>,<time>,<priority>

This statement defines a call to an EVENT subroutine specified by the user in the deck section (or defines a call to a system event, HALT, XCSIM, etc.). <name> is the 1-5 character designation of the event name, <time> defines the time at which the event is to occur, and <priority> is a number used as a "tie breaker" if two or more events happen to occur at the same time.

The output of the section is the list of exogenously scheduled events, XEVIL. Each entry in XEVIL contains a name, time, and priority of an event request from a data card. Routine XCSTART defines a call to each event listed in XEVIL.

```
      SUBROUTINE XEVSTK (CARD,JCOL)
      COMMON /XXEXTRN/ XNEX.XEXT(1)
      COMMON /XXEVENT/ XEVADR,XFLMAX.XNEV.XDONE,XNEVIL,XEVIL(20,3)
      COMMON/XXUNITS/XUO.XUI,XUP,XUE.XBF,XTRACE,XPLFG,XFILM,XNPL
      DIMENSION NTIME(2).NPRIOR(2),CARD(8)
      INTEGER XNEVIL,XEVIL.XNEX,XEXT,XUO
C
C.....THIS ROUTINE PROCESSES EXOGENOUS EVENT REQUESTS OF THE FORM:
C           EVENT.  <NAME>,<TIME>.<PRIORITY>
C.....THE MACHINE ADDRESS OF THE EVENT IS ENTERED INTO THE EVENT STACK
C.....ALONG WITH THE SCHEDULED TIME OF OCCURANCE.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      LCOL=0
      MCOL=0
      NCOL=0
      NAME=10H
      NTIME(1)=NTIME(2)=10H
      NPRIOR(1)=NPRIOR(2)=10H
      KODE=1
      IERR=0
      ICOL=JCOL
C
C.....THE EVENT REQUEST CARD IS PARSED.
C
   15 ICOL=ICOL+1
      IF (ICOL.GT.80) GO TO 50
      CALL GCHARS (CARD,ICOL,1,ICHR)
      IF (ICHR.EQ.1H ) GO TO 15
      IF (ICHR.EQ.1H,) GO TO 45
      GO TO (20,25,35), KODE
C
C.....ICHR IS ASSUMED PART OF AN EVENT NAME.
C
   20 NCOL=NCOL+1
      IF (NCOL.GT.5) GO TO 75
      CALL SCHARS (NAME,NCOL,1,ICHR)
      GO TO 15
C
C.....ICHR IS ASSUMED PART OF AN EVENT TIME.
C
   25 IF (ICHR.GE.1H0.O.ICHR.LE.1H-) GO TO 30
      IF (ICHR.EQ.1H.) GO TO 30
      IF (ICHR.EQ.1HE) GO TO 30
      GO TO 80
   30 MCOL=MCOL+1
      IF (MCOL.GT.20) GO TO 85
      CALL SCHARS (NTIME,MCOL,1,ICHR)
      GO TO 15
C
C.....ICHR IS ASSUMED PART OF A PRIORITY.
C
   35 IF (ICHR.GE.1H0.O.ICHR.LE.1H-) GO TO 40
      IF (ICHR.EQ.1H.) GO TO 40
      IF (ICHR.EQ.1HE) GO TO 40
      GO TO 80
   40 LCOL=LCOL+1
      IF (LCOL.GT.20) GO TO 85
```

```
          CALL SCHARS (NPRIOR,LCOL,1,ICHR)
          GO TO 15
C
C.....A COMMA HAS BEEN ENCOUNTERED.
C
   45 KODE=KODE+1
      IF (KODE.GT.3) GO TO 80
      GO TO 15
C
C.....PROCESS PARSED ITEMS.
C
   50 IF (KODE.LT.1) RETURN
      IF (XNEX.LE.0) GO TO 90
      DO 55 I=1,XNEX
         NNM=(77777777770000000000B.A.XEXT(I)).O.5555555555B
         IF (NAME.EQ.NNM) GO TO 60
   55 CONTINUE
      GO TO 90
   60 XNEVIL=XNEVIL+1
      IF (XNEVIL.GT.20) GO TO 95
      XEVIL(XNEVIL,1)=NAME
      TIME=0
      XEVIL(XNEVIL,2)=TIME
      IPRIOR=1
      XEVIL(XNEVIL,3)=IPRIOR
      IF (KODE.GE.2) GO TO 65
      IF (IERR) 105,100
   65 ENCODE (10,110,FMT) MCOL
      DECODE (MCOL,FMT,NTIME) TIME
      XEVIL(XNEVIL,2)=TIME.A.77777777777777777777B
      IF (KODE.GE.3) GO TO 70
      IF (IERR) 105,100
   70 ENCODE (10,110,FMT) LCOL
      DECODE (LCOL,FMT,NPRIOR) PRIOR
      IPRIOR=PRIOR
      XEVIL(XNEVIL,3)=IPRIOR
      RETURN
C
C.....IF ERROR ENCOUNTERED, PRINT DIAGNOSTIC.
C
   75 WRITE (XUO,115) (I,I=1,8),CARD
      WRITE (XUO,120) NAME
      RETURN
   80 WRITE (XUO,115) (I,I=1,8),CARD
      WRITE (XUO,125) ICHR,ICOL
      IERR=1
      KODE=KODE-1
      GO TO 50
   85 WRITE (XUO,115) (I,I=1,8),CARD
      WRITE (XUO,130) ICOL
      IERR=1
      KODE=KODE-1
      GO TO 50
   90 WRITE (XUO,115) (I,I=1,8),CARD
      WRITE (XUO,135) NAME
      RETURN
   95 WRITE (XUO,115) (I,I=1,8),CARD
      WRITE (XUO,140)
      XNEVIL=XNEVIL-1
      RETURN
  100 WRITE (XUO,115) (I,I=1,8),CARD
  105 WRITE (XUO,145) NAME,TIME,IPRIOR
      RETURN
C
  110 FORMAT (2H(E,I2,6H.0    ))
  115 FORMAT (6H0*****,32HERROR IN EXOGENOUS EVENT REQUEST,//T20,8I10/T2
     10,8(10H1234567890)/T20,8A10)
  120 FORMAT (1H0,T14,12HEVENT NAME (,A5,24H...) LONGER THAN 5 CHARS)
  125 FORMAT (1H0,T14,11HCHARACTER (,A1,20H) ILLEGAL IN COLUMN ,I2)
  130 FORMAT (1H0,T14,48HTIME OR PRIORITY LONGER THAN 20 CHARS AT COLUMN
     1 ,I2)
```

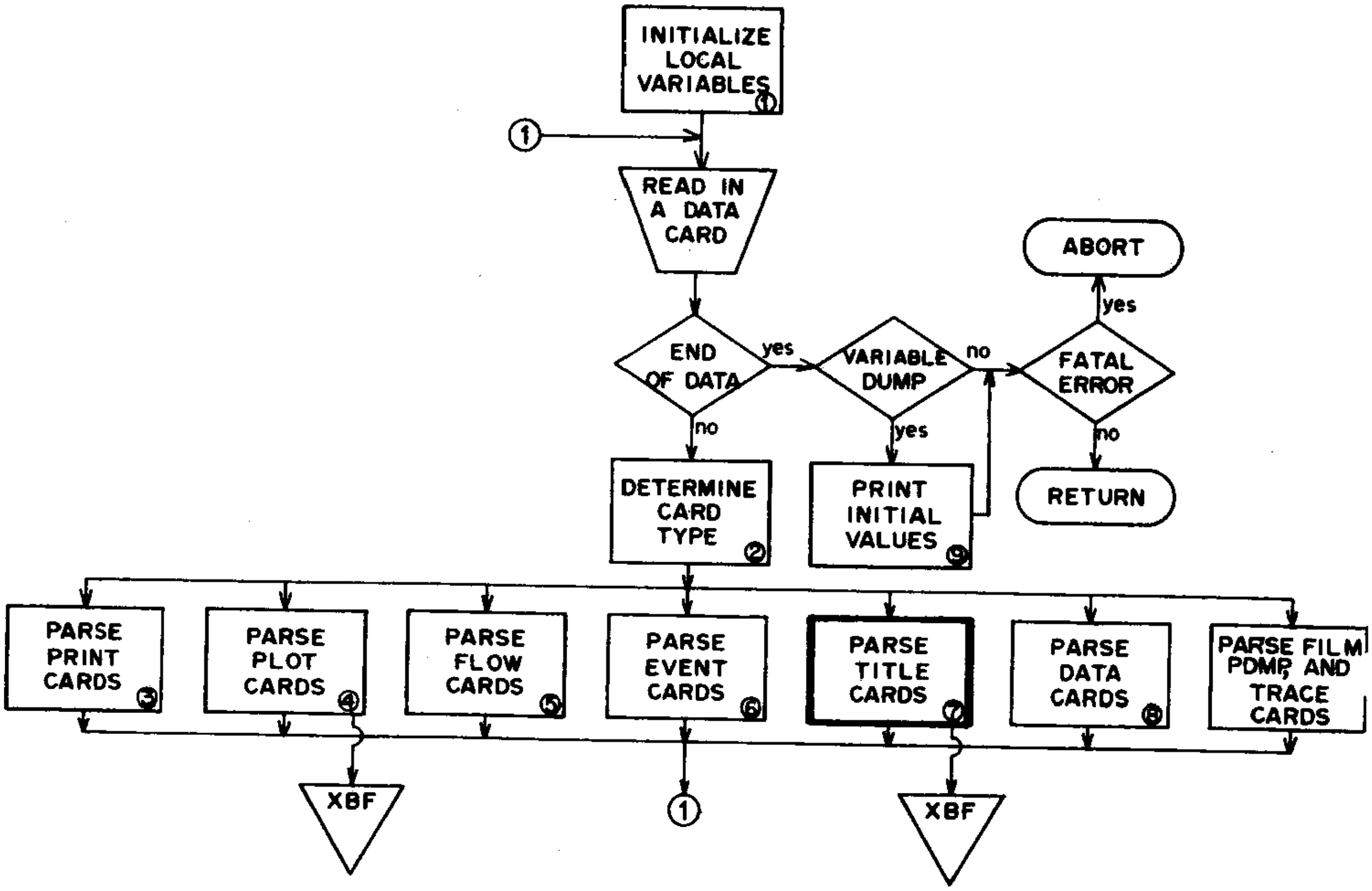
```
72130          135 FORMAT (1H0,T14,7HEVENT (,A5,17H) IS NON-EXISTANT)
72131          140 FORMAT (1H0,T14,69HNO. OF EXOGENOUSLY SCHEDULED EVENTS EXCEEDS 20,
72132             1 ABOVE REQUEST IGNORED)
72133          145 FORMAT (1H0,T14,6HEVENT ,A5,19H SCHEDULED AT TIME ,E16.8,16H AT PR
72134             1IORITY OF ,I3)
72135       C
72136              END
```

Line Number	Explanation
72000-72022	CARD contains a data card image. JCØL points at the column before the beginning of the text section. LCØL is a count of the number of characters in the priority portion of an event. NPRIØR is the priority number assigned to each event. If two events are scheduled for the same execution time, then the one with the lowest priority number is executed first. MCØL is the number of characters in the time portion of an event request. NTIME contains the MCØL characters of the event time. NCØL is the number of characters in the name portion of an event. NAME is the NCØL characters of the event name. KØDE is a counter of the delimiters encountered. KØDE=1 Initial value, search for event name. =2 Comma encountered, recover event time. =3 Comma encountered, recover event priority. ICØL points at the column currently being scanned.
72026-72031	A character is retrieved from CARD, and depending on the value of KØDE, is assumed either part of an event name, time, or priority.
72035-72038	The character is assumed part of an event name; fill the character into NAME. NCØL counts the number of characters in NAME (NCØL$\le$ 5).
72042-72049	The character is assumed part of an event time. NTIME is filled with MCØL characters of the event time.
72053-72060	The character is assumed part of an event priority. NPRIØR is filled with LCØL characters of priority from data card. NOTE. The only legal characters that may occur in a priority definition are: numeric characters, 1H+, 1H-, 1H., or 1HE. All other characters cause an error message to be printed.

Line Number	Explanation
72064-72066	A comma delimits either event name and time or time and priority. Increment KØDE and begin parse for event time or priority.
72070-72076	End of card is encountered; parsing is assumed to be complete. Search event stack (XEXT) for a match with event name (NAME) parsed from data card. RECALL. (from Sections 1.2 and 1.8) XEXT contains the list of user (and system) defined event names. XNEX is the number of entries in XEXT. A match with NAME and an entry of XEXT indicates that an event of that name exists in user's source section (or, for system events, named event exists in system text).
72077-72083	A match was found; enter event information into a stack of exogenous event requests. XEVIL(I,J) contains the Ith encountered exogenous event request where J=1 Contains the event name. =2 Contains the event time. =3 Contains the event priority. XNEVIL is the number of entries in XEVIL
72084-72085	If KØDE<2, then an error has occurred while parsing the time; print a diagnostic and return. NOTE. With an error in the time parse, the named event is still entered in the exogenous event stack, but with default value of time=0 and priority=1. IERR≠0 indicates that a diagnostic has already been printed (lines 72102-72111); therefore print only secondary message (line 72120).
72086-72090	Encode the characters in NTIME into their equivalent floating point value. Store the complement of this number in XEVIL. KØDE<3 indicates an error in parse of priority, XEVIL contains a default priority=1. print appropriate diagnostic.
72091-72095	Encode the characters in NPRIØR into their equivalent floating point value and store in XEVIL.

Line Number	Explanation
72097-72136	Error section. Nonfatal diagnostics are issued for errors encountered while parsing the card.
72102-72111	An error occurred in the parse of either time or priority; event may still be entered into exogenous event stack.

2.7 Parse TITLE. Cards



Overview

This section processes cards of the form:

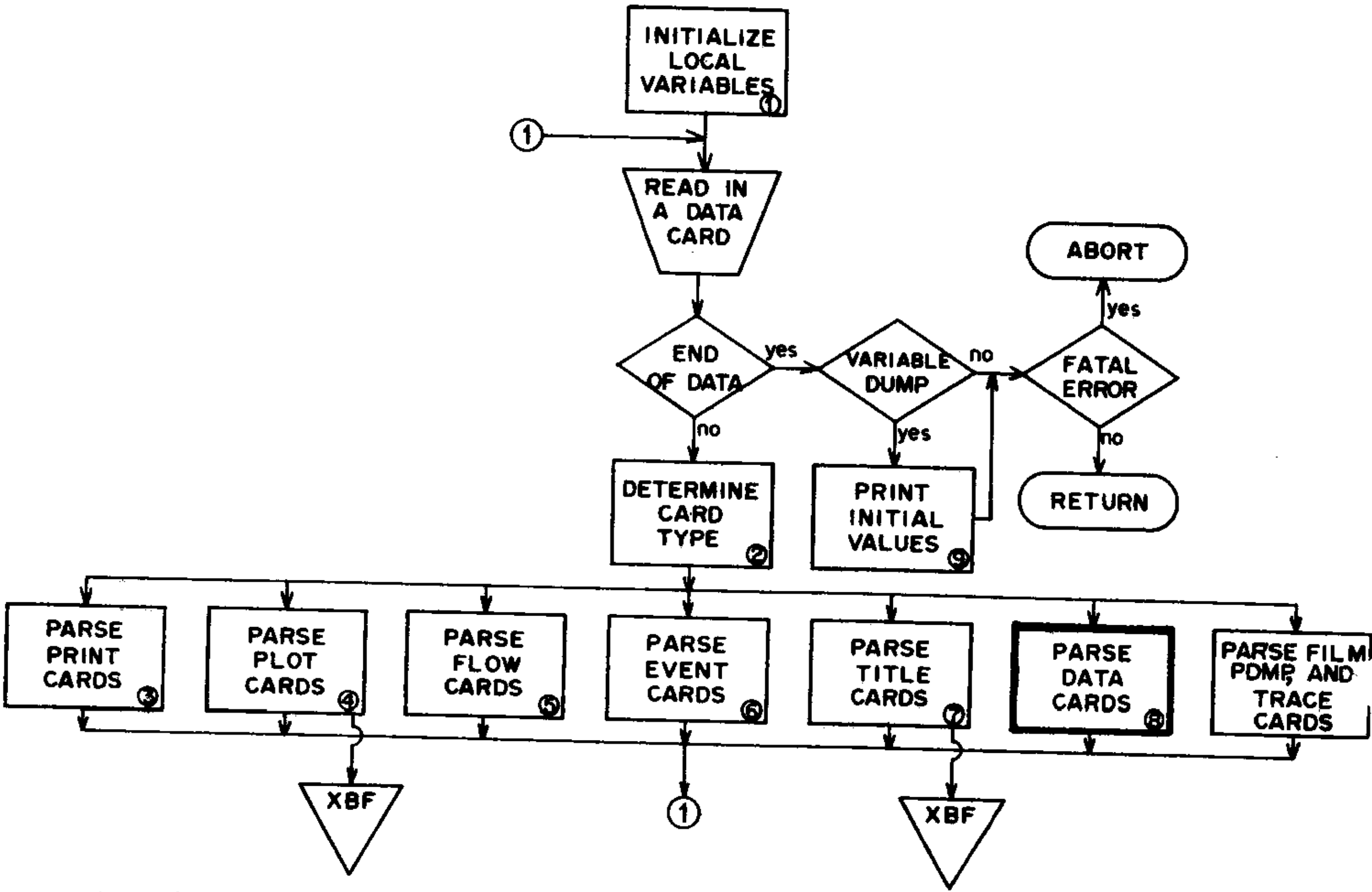
TITLE. <plot title>

where <plot title> is the desired label for the graph which will be generated by the first PLØT. card following the TITLE. card. The contents of each TITLE. card is written onto the plot variable file, XBF, to be used (by PROGRAM PLØT) in the construction of plotted output.

```
75000              SUBROUTINE XTITLE (CARD,JCOL)
75001        C
75002        C.....XTITLE PROCESSES A TITLE REQUEST FOR THE SUCCEEDING PLOT CARD.
75003        C          IBUF((I=1,3),(J=1,3),1) CONTAINS THE REMAINING 74 COLUMNS OF
75004        C          THE  TITLE. CARD
75005        C          IBUF(3,5,6) CONTAINS THE FLAG WORD   TITLE.
75006        C
75007              COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
75008              COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
75009              DIMENSION CARD(8)
75010              INTEGER XUO,XUI,XUP,XUE,XBF,XNPL
75011              DO 15 I=1,3
75012              DO 15 J=1,5
75013              DO 15 K=1,6
75014           15 IBUF(I,J,K)=0
75015              IBUF(3,5,6)=10HTITLE.
75016              ICOL=JCOL
75017              KCOL=0
75018           20 ICOL=ICOL+1
75019              KCOL=KCOL+1
75020              IF (ICOL.GT.80) GO TO 25
75021              CALL GCHARS (CARD,ICOL,1,ICHR)
75022              CALL SCHARS (IBUF,KCOL,1,ICHR)
75023              GO TO 20
75024        C
75025        C.....ONE RECORD IS WRITTEN ONTO THE PLOT STACK FILE XBF
75026        C
75027           25 WRITE (XBF) IBUF,RBUF
75028              RETURN
75029        C
75030              END
```

Line Number	Explanation
75011-75017	Initialize variables: IBUF is the working buffer which is flushed onto a record of plot file XBF. IBUF(3,5,6) is filled with the flag word TITLE. to distinguish records on XBF containing plot title information from records containing plot variable information. ICØL is the number of the column following the command TITLE. CARD contains a TITLE. card from user's source deck.
75018-75023	Fill the characters on CARD (starting in column ICØL) into IBUF (starting in column 1).
75024-75030	Flush IBUF onto XBF.

2.8 Parse Data Cards



Overview

This section parses cards of the form:

<data statement>$<data statement>$•••

where <data statement>: = <variable name>=<value>

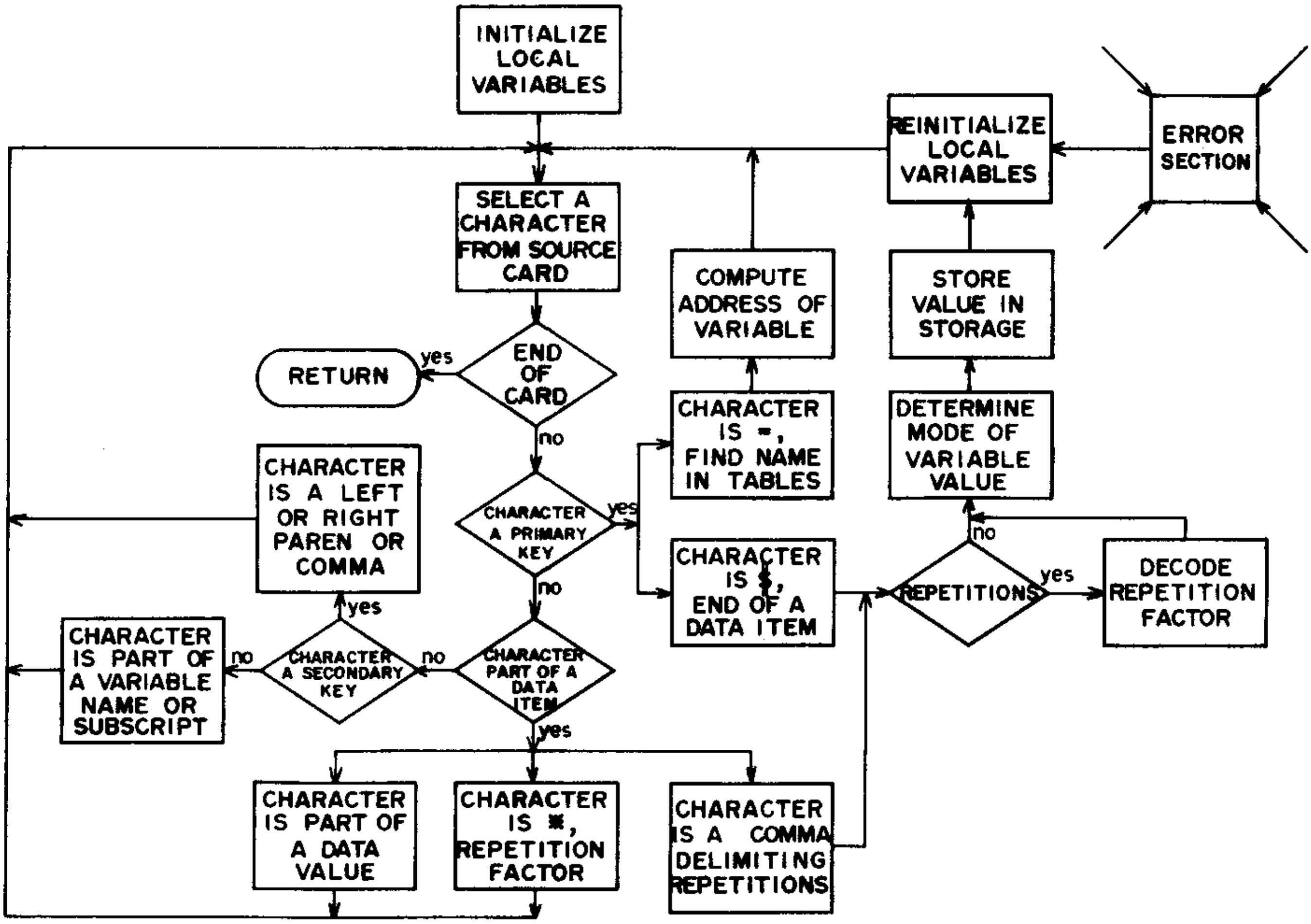
<variable name>=<repetition factor>*<value>

<variable name>=<value>,<value>,•••

The name of the variable is matched with an entry in the variable tables, XVT1 and XVT2. The location of the variable (on the data card) relative to the beginning of blank common is calculated (relative to XADRS(1)), and the <value> is stored at that location.

The section is subdivided for easier analysis.

Data cards flow chart

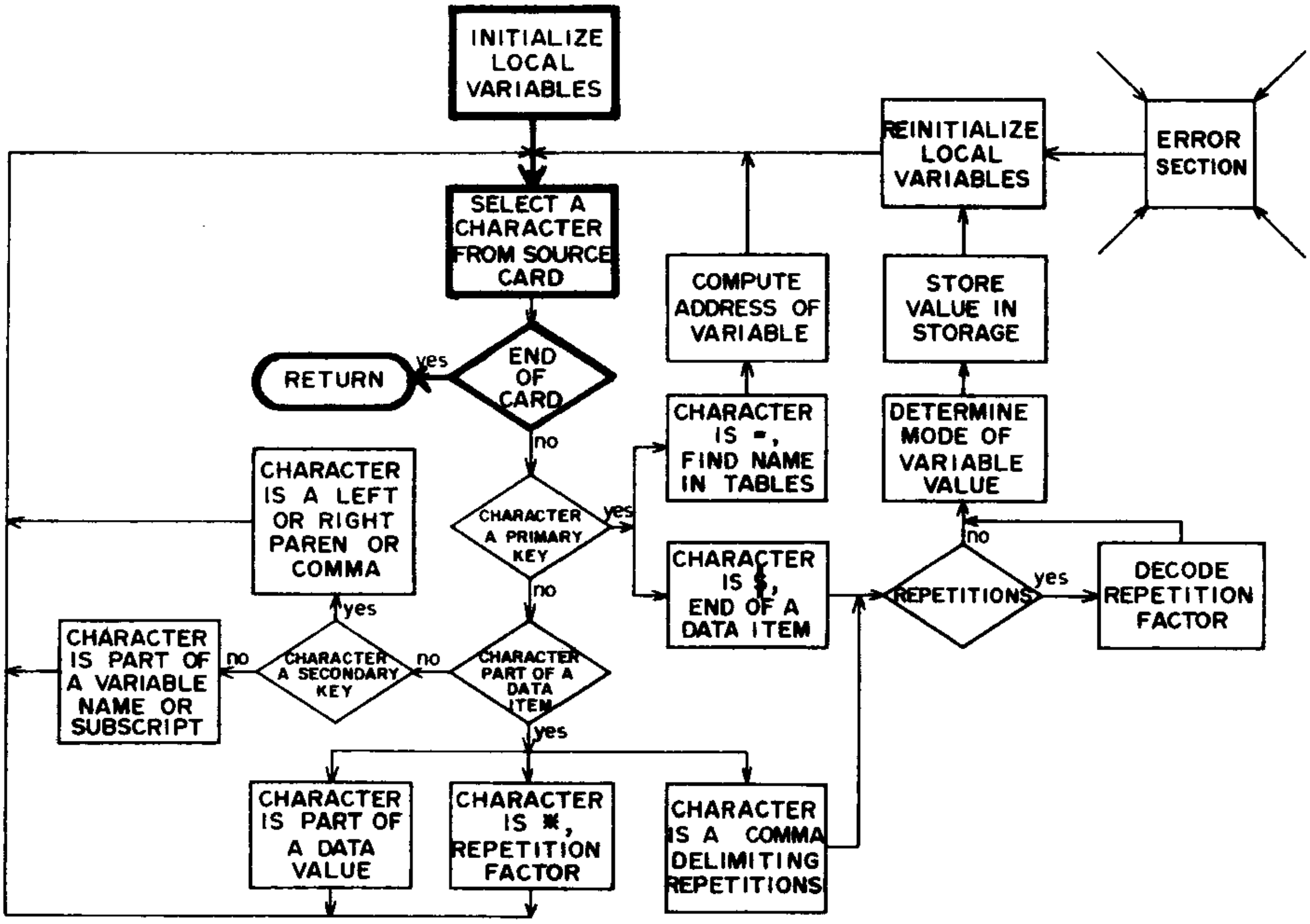

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
RETURN
no
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS =, FIND NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
no
DECODE REPETITION FACTOR
DETERMINE MODE OF VARIABLE VALUE
STORE VALUE IN STORAGE
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
CHARACTER PART OF A DATA ITEM
no
CHARACTER A SECONDARY KEY
yes
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
no
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
      SUBROUTINE XDATA (XCARD,XICOL)
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON XADRS(1)
      DIMENSION XCARD(8)
      INTEGER XICOL,XICHR,XI,XNK,XKEY(3),XKOD1,XKOD2,XNCOL,XMCOL(3),XSUB
     1(3),XLCOL,XNUM(2),XNM,XREPT,XNREPT,XNV,XVT1,XVT2,XI1,XMD,XN1(3),XN
     22(3),XIERR,XJ,XI2,XLOC1,XLOC2,XLOC,XUO,XUI,XTIMES,XNAME,XWRITE
      LOGICAL XSTOP,XFLOAT,XFNI
C
C.....THIS ROUTINE PARSES DATA VALUE ASSIGNMENT STATEMENTS.  THE VARIABL
C.....NAME AND VALUE ARE RETRIEVED FROM THE DATA CARD AND THE VALUE IS
C.....DECODED AND STORED IN THE VARIABLE STORAGE BLOCK.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA XKEY/1H(,1H,,1H)/,XNK/3/
      DATA XNCOL/0/,XNAME/1H /,XMCOL/3*0/,XSUB/3*1H /
      DATA XKOD1/1/,XKOD2/1/
      DATA XLCOL/0/,XNUM/2*1H /,XNREPT/0/
      DATA XSTOP/.FALSE./,XWRITE/0/
      DATA XFLOAT/.FALSE./, XFNI/.FALSE./
C
C.....RETRIEVE EACH CHARACTER, COLUMN BY COLUMN, STORING THE PARSED
C.....CHARACTERS IN APPROPRIATE LOCATIONS.
```

```
C
      IF (XFNI) GO TO 210
   15 XICOL=XICOL+1
      IF (XICOL.GT.80) RETURN
      CALL GCHARS (XCARD,XICOL,1,XICHR)
      IF (XICHR.EQ.1H ) GO TO 15
      IF (XICHR.EQ.1H=) GO TO 75
      IF (XICHR.EQ.1H$) GO TO 150
C
C.....CHARACTER IS NOT A PRIMARY KEY CHARACTER.
C
      GO TO (20,60), XKOD1
C
C.....CHARACTER IS ASSUMED PART OF VARIABLE NAME OR SUBSCRIPT, SEARCH FO
C.....SECONDARY KEY CHARACTERS.
C
   20 DO 25 XI=1,XNK
         IF (XICHR.EQ.XKEY(XI)) GO TO 40
   25 CONTINUE
C
C.....CHAR IS NOT A SECONDARY KEY CHARACTER.
C
      IF (XKOD2.GT.4) GO TO 155
      GO TO (30,35,35,35), XKOD2
C
C.....CHAR ASSUMED PART OF VARIABLE NAME.
C
   30 XNCOL=XNCOL+1
      IF (XNCOL.GT.5) GO TO 160
      CALL SCHARS (XNAME,XNCOL,1,XICHR)
      GO TO 15
C
C.....CHAR ASSUMED PART OF A SUBSCRIPT.
C
   35 XI=XKOD2-1
      IF (XICHR.LT.1H0.O.XICHR.GT.1H9) GO TO 155
      XMCOL(XI)=XMCOL(XI)+1
      IF (XMCOL(XI).GT.4) GO TO 165
      CALL SCHARS (XSUB(XI),XMCOL(XI),1,XICHR)
      GO TO 15
C
C.....CHAR IS A SECONDARY KEY CHARACTER.
C
   40 GO TO (45,50,55), XI
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   45 IF (XKOD2.NE.1) GO TO 155
      XKOD2=2
      GO TO 15
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   50 IF (XKOD2.LT.2.O.XKOD2.GT.3) GO TO 155
      XKOD2=XKOD2+1
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   55 IF (XKOD2.LT.2.O.XKOD2.GT.4) GO TO 155
      XKOD2=5
      GO TO 15
C
C.....CHARACTER IS ASSUMED PART OF A DATA ITEM.
C
   60 IF (XICHR.EQ.1H,) GO TO 105
      IF (XICHR.EQ.1H*) GO TO 70
      IF (XICHR.GE.1H0.A.XICHR.LE.1H-) GO TO 65
      IF (XICHR.EQ.1H.) GO TO 65
      IF (XICHR.EQ.1HE) GO TO 65
      GO TO 155
   65 XLCOL=XLCOL+1
      IF (XICHR.EQ.1H.) XFLOAT=.TRUE.
      IF (XICHR.EQ.1HE) XFLOAT=.TRUE.
      IF (XLCOL.GT.20) GO TO 170
      CALL SCHARS (XNUM,XLCOL,1,XICHR)
      GO TO 15
C
C.....DATA ITEM CONTAINS REPETITION FACTOR.
C
   70 IF (XLCOL.GT.10) GO TO 175
      XREPT=XNUM(1)
      XNREPT=XLCOL
      XLCOL=0
      XNUM(2)=10H
      XNUM(1)=XNUM(2)
      GO TO 15
C
C.....AN EQUAL "=" HAS BEEN ENCOUNTERED. FIND VARIABLE NAME IN THE
C.....REFERENCE TABLES AND RETRIEVE INDICES FOR VALUE STORAGE.
C
   75 IF (XKOD1.NE.1) GO TO 155
      IF (XNCOL.LE.0) GO TO 180
      DO 80 XI=1,XNV
         CALL GCHARS (XVT1(XI),1,5,XNM)
         IF (XNAME.EQ.XNM) GO TO 85
   80 CONTINUE
      GO TO 185
C
C.....VARIABLE FOUND IN REFERENCE TABLES. RETRIEVE TABLE INFORMATION.
C
   85 CALL GBYTE (XVT1(XI),XI1,30,18)
      CALL GBYTE (XVT1(XI),XMD,48,2)
      XN1(1)=1
      DO 90 XJ=1,2
   90 CALL GBYTE (XVT2(XI),XN1(XJ+1),XJ*10-10,10)
C
C.....DECODE VARIABLE SUBSCRIPTS.
C
      DO 95 XJ=1,3
         XN2(XJ)=1
         IF (XMCOL(XJ).LE.0) GO TO 95
         CALL GNUM (XSUB(XJ),1,XMCOL(XJ),XN2(XJ),XIERR)
         IF (XIERR.NE.0) GO TO 190
   95 CONTINUE
C
C.....COMPUTE ADDRESS OF THE VARIABLE RELATIVE TO FIRST WORD ADDRESS OF
C.....VARIABLE.
C
      XI2=0
      DO 100 XI=1,3
         XJ=4-XI
  100 XI2=XN1(XJ)*(XN2(XJ)-1+XI2)
C
C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD OF
C.....THE STORAGE BLOCK "XADRS(1)".
C
      XLOC1=XI1+XI2
      XLOC2=-1
      XKOD1=2
      GO TO 15
C
C.....A COMPLETE DATA ITEM HAS BEEN RETRIEVED, DECODE ACCORDING TO
C.....VARIABLE MODE AND STORE VALUE.
C
  105 XTIMES=1
      IF (XNREPT.LE.0) GO TO 110
C
C.....DECODE REPETITION FACTOR.
C
      CALL GNUM (XREPT,1,XNREPT,XTIMES,XIERR)
      IF (XIERR.NE.0) GO TO 195
      IF (XTIMES.LE.0) GO TO 200
C
C.....DECODE DATA ITEM.
```

```
C
  110 IF (XLCOL.LE.0) GO TO 180
      XI=XMD+1
      GO TO (115,125), XI
C
C.....FIXED POINT MODE
C
  115 IF (.NOT.XFLOAT) GO TO 120
      IF (XWRITE.EQ.XNAME) GO TO 130
      XWRITE=XNAME
      WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,280) XNAME
      GO TO 130
  120 ENCODE (10,215,XFMT) XLCOL
      DECODE (XLCOL,XFMT,XNUM) XVAL
      GO TO 135
C
C.....FLOATING POINT MODE
C
  125 IF (XFLOAT) GO TO 130
      IF (XWRITE.EQ.XNAME) GO TO 130
      XWRITE=XNAME
      WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,285) XNAME
  130 ENCODE (10,220,XFMT) XLCOL
      DECODE (XLCOL,XFMT,XNUM) XVAL
C
C.....STORE VALUE IN STORAGE BLOCK.
C
  135 DO 140 XI=1,XTIMES
          XLOC2=XLOC2+1
          XLOC=XLOC1+XLOC2
  140 XADRS(XLOC)=77777777777777777777B.A.XVAL
C
C.....REINITIALIZE LOCAL VARIABLES.
C
  145 XLCOL=0
      XNUM(2)=10H
      XNUM(1)=XNUM(2)
      XNREPT=0
      IF (.N.XSTOP) GO TO 15
      XNCOL=0
      XNAME=10H
      XFLOAT=.FALSE.
      XMCOL(3)=0
      XMCOL(2)=XMCOL(3)
      XMCOL(1)=XMCOL(2)
      XSUB(3)=10H
      XSUB(2)=XSUB(3)
      XSUB(1)=XSUB(2)
      XKOD1=1
      XKOD2=1
      XSTOP=.FALSE.
      GO TO 15
C
C.....A DOLLAR "$" HAS BEEN ENCOUNTERED.
C
  150 IF (XKOD1.NE.2) GO TO 155
      XSTOP=.TRUE.
      GO TO 105
C
C.....IF ERRORS FOUND GENERATE DIAGNOSTIC.
C
  155 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,230) XICHR,XICOL
      GO TO 205
  160 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,235) XNAME
      GO TO 205
  165 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,240) XSUB(XI),XICOL
      GO TO 205
  170 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,245) XNUM
      GO TO 205
  175 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,250) XNUM(1)
      GO TO 205
  180 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,255) XICOL
      GO TO 205
  185 WRITE (XUO,290) (XJ,XJ=1,8),XCARD
      WRITE (XUO,260) XNAME
      GO TO 205
  190 WRITE (XUO,225) (XI,XI=1,8),XCARD
      WRITE (XUO,265) XSUB(XJ)
      GO TO 205
  195 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,270) XREPT
      GO TO 205
  200 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,275) XREPT
C
C.....BELOW SEGMENT IS EXECUTED WHEN AN ERROR IS ENCOUNTERED. IT SEARCHE
C.....REMAINDER OF CARD FOR A $ SIGN AND LOOPS BACK TO MAIN PROGRAM.
C
  205 XFNI=.TRUE.
      XICOL=XICOL-1
  210 XICOL=XICOL+1
      IF (XICOL.GT.80) RETURN
      CALL GCHARS (XCARD,XICOL,1,XICHR)
      IF (XICHR.NE.1H$) GO TO 210
      XFNI=.FALSE.
      XSTOP=.TRUE.
      GO TO 145
C
  215 FORMAT (2H(I,I2,6H)      )
  220 FORMAT (2H(E,I2,6H.0)    )
  225 FORMAT (6H0*****, 24HERROR IN DATA ASSIGNMENT,//T20,8I10/T20,8( 10
     1H1234567890)/T20,8A10)
  230 FORMAT (1H0,T14, 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLUMN ,I2)
  235 FORMAT (1H0,T14, 10HVARIABLE ",A5, 24H..." LONGER THAN 5 CHARS)
  240 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 35H..." LONGER THAN 4 CHARS IN
     1 COLUMN ,I2)
  245 FORMAT (1H0,T14, 11HDATA ITEM ",2A10, 25H..." LONGER THAN 20 CHARS
     1)
  250 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 25H..." LONGER T
     1HAN 10 CHARS)
  255 FORMAT (1H0,T14, 59HZERO LENGTH VARIABLE NAME OR DATA ITEM IN OR B
     1EFORE COLUMN ,I2)
  260 FORMAT (1H0,T14, 10HVARIABLE ",A5, 44H" WAS NOT DECLARED IN A <STO
     1RAGE.> STATEMENT)
  265 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
  270 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 15H" NOT DECODAB
     1LE)
  275 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 28H" LESS THAN O
     1R EQUAL TO ZERO)
  280 FORMAT (1H0,T14, 17HINTEGER VARIABLE ,A5, 46H WAS ASSIGNED A REAL
     1VALUE IN THE DATA SECTION)
  285 FORMAT (1H0,T14, 14HREAL VARIABLE ,A5, 50H WAS ASSIGNED AN INTEGER
     1 VALUE IN THE DATA SECTION)
  290 FORMAT (7H0**NF**,24HERROR IN DATA ASSIGNMENT,//T20,8I10/T20,8( 10
     1H1234567890)/T20,8A10)
C
      END
```

Initialize and retrieve a character

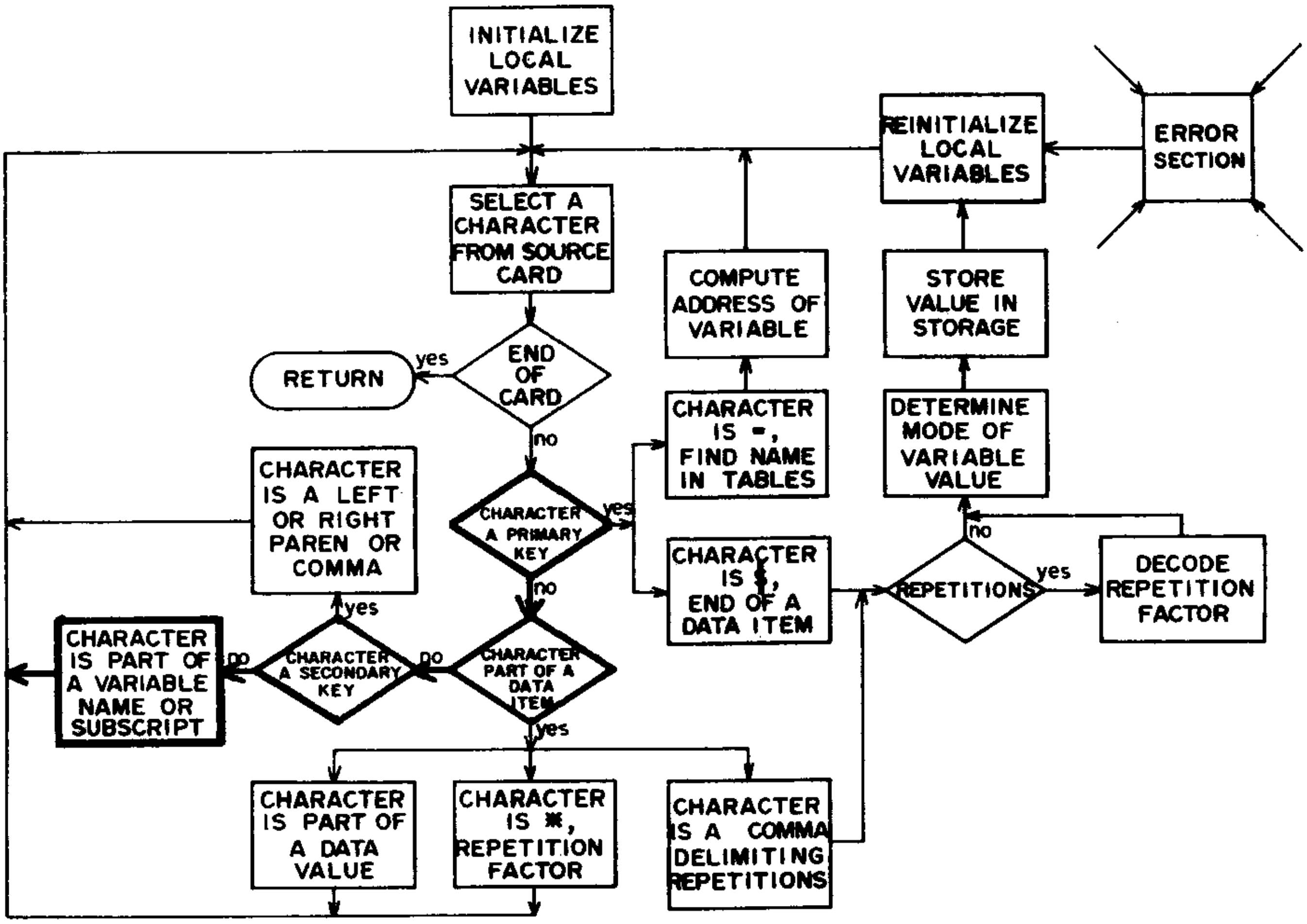

INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
      SUBROUTINE XDATA (XCARD,XICOL)
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON XADRS(1)
      DIMENSION XCARD(8)
      INTEGER XICOL,XICHR,XI,XNK,XKEY(3),XKOD1,XKOD2,XNCOL,XMCOL(3),XSUB
     1(3),XLCOL,XNUM(2),XNM,XREPT,XNREPT,XNV,XVT1,XVT2,XI1,XMD,XN1(3),XN
     22(3),XIERR,XJ,XI2,XLOC1,XLOC2,XLOC,XUO,XUI,XTIMES,XNAME,XWRITE
      LOGICAL XSTOP,XFLOAT,XFNI
C
C.....THIS ROUTINE PARSES DATA VALUE ASSIGNMENT STATEMENTS.  THE VARIABL
C.....NAME AND VALUE ARE RETRIEVED FROM THE DATA CARD AND THE VALUE IS
C.....DECODED AND STORED IN THE VARIABLE STORAGE BLOCK.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA XKEY/1H(,1H,,1H)/,XNK/3/
      DATA XNCOL/0/,XNAME/1H /,XMCOL/3*0/,XSUB/3*1H /
      DATA XKOD1/1/,XKOD2/1/
      DATA XLCOL/0/,XNUM/2*1H /,XNREPT/0/
      DATA XSTOP/.FALSE./,XWRITE/0/
      DATA XFLOAT/.FALSE./, XFNI/.FALSE./
C
```

```
73024         C.....RETRIEVE EACH CHARACTER, COLUMN BY COLUMN, STORING THE PARSED
73025         C.....CHARACTERS IN APPROPRIATE LOCATIONS.
73026         C
73027               IF (XFNI) GO TO 210
73028            15 XICOL=XICOL+1
73029               IF (XICOL.GT.80) RETURN
73030               CALL GCHARS (XCARD,XICOL,1,XICHR)
73031               IF (XICHR.EQ.1H ) GO TO 15
73032               IF (XICHR.EQ.1H=) GO TO 75
73033               IF (XICHR.EQ.1H$) GO TO 150
```

Line Number	Explanation
73000-73022	XCARD contains one card (80 columns) from data section. XICØL points at the column currently being scanned (initialized to zero in XCRDTP for this section). XKEY contains a list of characters which delimit the subscripts of a variable. XNK is the number of elements in XKEY. XNCØL is the number of characters in XNAME. XNAME is filled with the characters of a variable name from a data card. XMCØL(I) is the number of characters in the Ith subscript of a variable from a data card currently being parsed. XSUB(I) is filled with the characters of the Ith subscript. XKØD1=1, then the retrieved character is part of a variable name or subscript. =2, the character is part of a data value. XKØD2 is a count of the XKEY characters encountered while parsing subscripts. XLCØL is the number of characters in XNUM. XNUM is filled with the characters of a value of a variable. XNREPT is the number of characters in the repetition factor. XSTØP is a logical flag set to .TRUE. when the end of a data statement is encountered ($). XWRITE contains the variable name if a mode error occurs. XFLØAT is set to .TRUE. if the value portion of a data statement is of type real. Since data statements can be continued from card to card, the local variables are initially set by DATA cards so that they will not be reinitialized at the beginning of each card.
73027-73033	The card is scanned column by column.

Character is part of a name or subscript

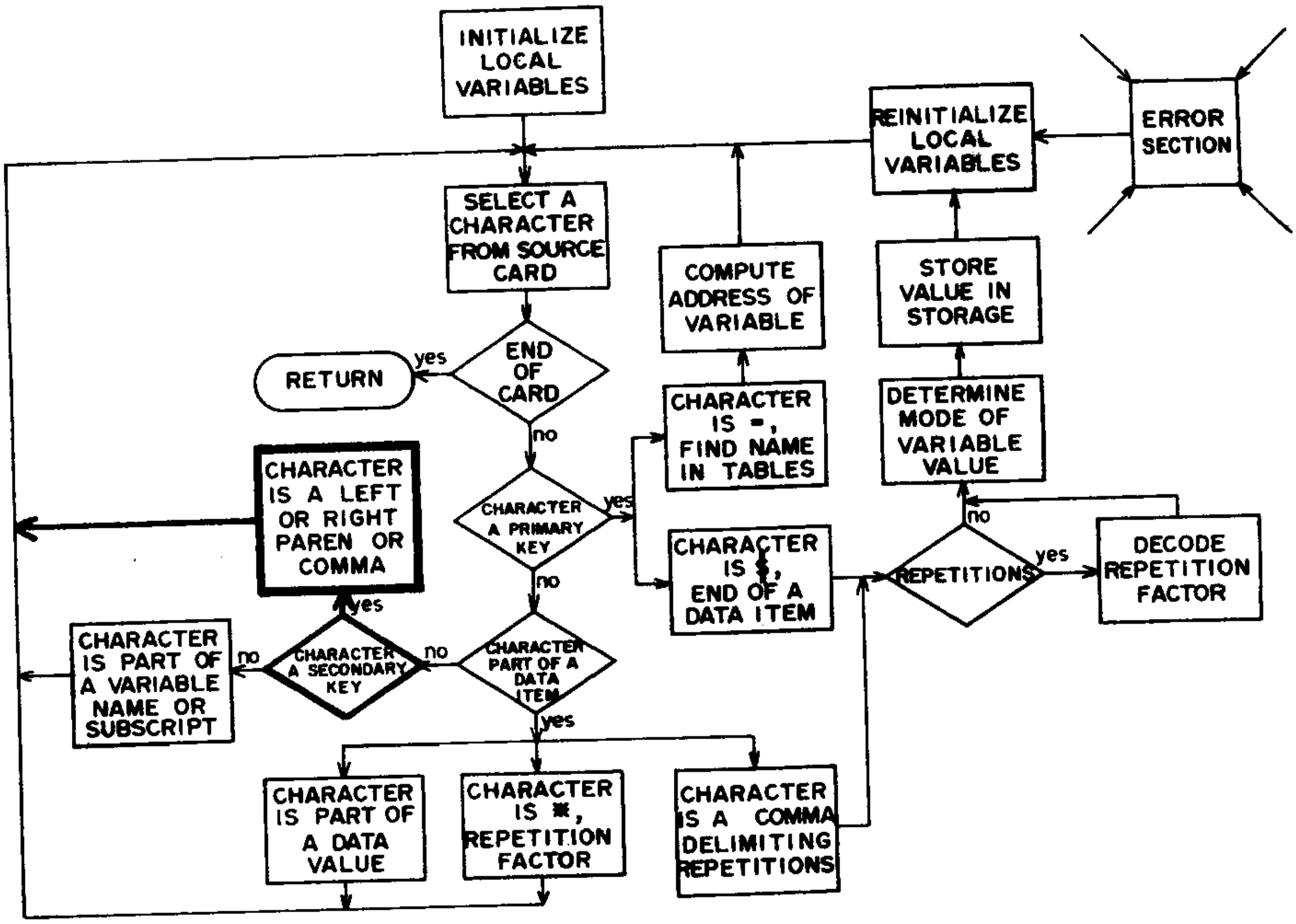

INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
RETURN
no
CHARACTER A PRIMARY KEY
yes
CHARACTER IS =, FIND NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
no
DECODE REPETITION FACTOR
DETERMINE MODE OF VARIABLE VALUE
STORE VALUE IN STORAGE
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
no
CHARACTER PART OF A DATA ITEM
no
CHARACTER A SECONDARY KEY
yes
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
no
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
C
C.....CHARACTER IS NOT A PRIMARY KEY CHARACTER.
C
      GO TO (20,60), XKOD1
C
C.....CHARACTER IS ASSUMED PART OF VARIABLE NAME OR SUBSCRIPT, SEARCH FO
C.....SECONDARY KEY CHARACTERS.
C
   20 DO 25 XI=1,XNK
         IF (XICHR.EQ.XKEY(XI)) GO TO 40
   25 CONTINUE
C
C.....CHAR IS NOT A SECONDARY KEY CHARACTER.
C
      IF (XKOD2.GT.4) GO TO 155
      GO TO (30,35,35,35), XKOD2
C
C.....CHAR ASSUMED PART OF VARIABLE NAME.
C
   30 XNCOL=XNCOL+1
      IF (XNCOL.GT.5) GO TO 160
      CALL SCHARS (XNAME,XNCOL,1,XICHR)
      GO TO 15
```

```
73057          C
73058          C.....CHAR ASSUMED PART OF A SUBSCRIPT.
73059          C
73060             35 XI=XKOD2-1
73061                IF (XICHR.LT.1H0.O.XICHR.GT.1H9) GO TO 155
73062                XMCOL(XI)=XMCOL(XI)+1
73063                IF (XMCOL(XI).GT.4) GO TO 165
73064                CALL SCHARS (XSUB(XI),XMCOL(XI),1,XICHR)
73065                GO TO 15
```

Line Number	Explanation
73037	XKØD1 indicates whether the retrieved character is part of a variable name (or subscript), or part of a data item. XKØD1=1 Initial value. XKØD1=2 When an equal sign is encountered, characters to the right of an equal sign are assumed data values (items).
73042-73044	Check to see if the retrieved character is a secondary key character (delimits variable name and subscripts); if not, then the character must be a part of the variable name or one of the subscripts.
73048-73049	XKØD2 counts how many secondary key characters have been encountered XKØD2=1 Initial value. =2 When a left paren is encountered. =3 When comma separating first and second subscript is encountered. =4 For comma between second and third subscript. =5 When a right paren is encountered.
73053-73056	The character is part of a variable name (XKØD2=1). XNCØL is the number of characters in XNAME. XNAME is filled with the characters of the variable name.
73060-73065	The character is part of a subscript (XKØD2>1). XI is the subscript to which the character belongs. XMCØL(XI) is the number of characters in the XIth subscript. XSUB(XI) is filled with the characters of the XIth subscript. EXAMPLE. FØX(100,13,2) XNCØL=3 XNAME=10HFØX XMCØL(1)=3 XSUB(1)=10H100 XMCØL(2)=2 XSUB(2)=10H13 XMCØL(3)=1 XSUB(3)=10H2

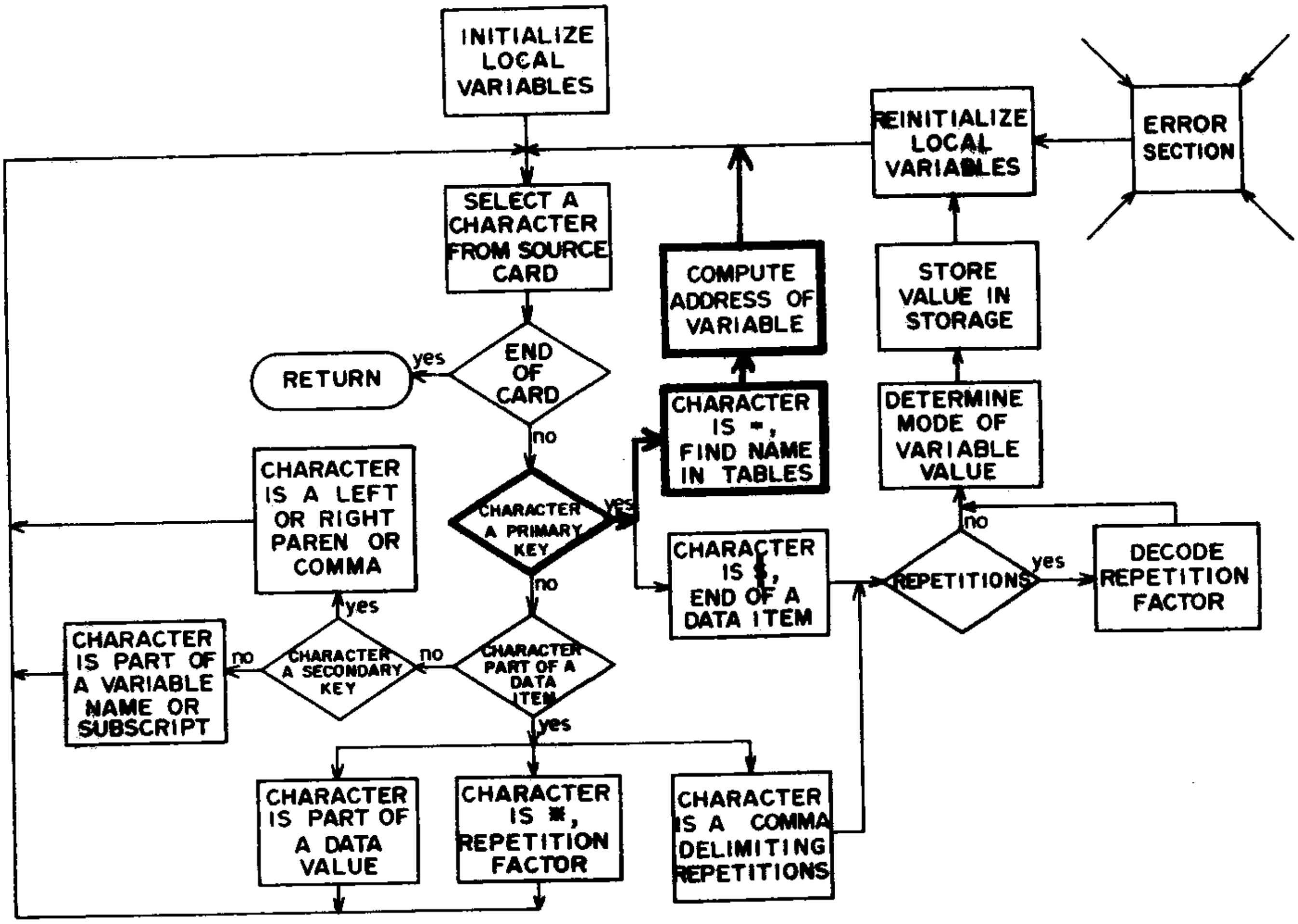
Process subscript delimiter character


INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
no
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
C
C.....CHAR IS A SECONDARY KEY CHARACTER.
C
   40 GO TO (45,50,55), XI
C
C.....A LEFT PAREN. "(" HAS BEEN ENCOUNTERED.
C
   45 IF (XKOD2.NE.1) GO TO 155
      XKOD2=2
      GO TO 15
C
C.....A COMMA "," HAS BEEN ENCOUNTERED.
C
   50 IF (XKOD2.LT.2.O.XKOD2.GT.3) GO TO 155
      XKOD2=XKOD2+1
      GO TO 15
C
C.....A RIGHT PAREN. ")" HAS BEEN ENCOUNTERED.
C
   55 IF (XKOD2.LT.2.O.XKOD2.GT.4) GO TO 155
      XKOD2=5
      GO TO 15
```

Line Number	Explanation
73069	Enter here (from line 73043) if the current character is a secondary key character (a comma, left paren, or right paren). XI=1 implies character matched XKEY(1) containing a left paren.
73073-73075	A left paren is encountered; set to XKOD2=2, and look for first subscript.
73079-73082	A comma is encountered (delimits subscripts); look for next subscript.
73085-73087	A right paren is encountered delimiting last subscript; look for an equal sign.

Locate parsed variable


INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
no
REPETITIONS
yes
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

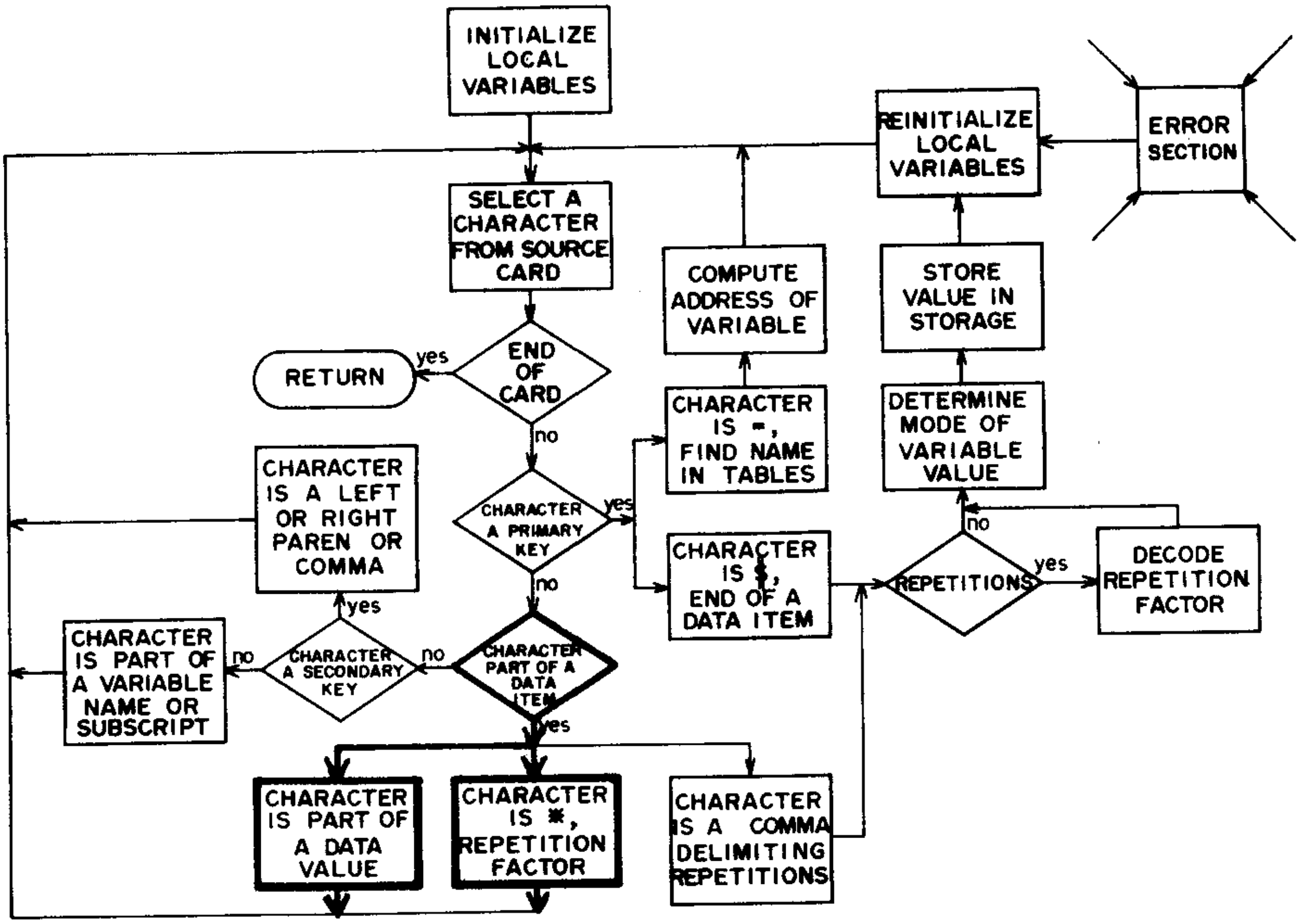
```
      C
      C.....AN EQUAL "=" HAS BEEN ENCOUNTERED, FIND VARIABLE NAME IN THE
      C.....REFERENCE TABLES AND RETRIEVE INDICES FOR VALUE STORAGE.
      C
         75 IF (XKOD1.NE.1) GO TO 155
            IF (XNCOL.LE.0) GO TO 180
            DO 80 XI=1,XNV
               CALL GCHARS (XVT1(XI),1,5,XNM)
               IF (XNAME.EQ.XNM) GO TO 85
         80 CONTINUE
            GO TO 185
      C
      C.....VARIABLE FOUND IN REFERENCE TABLES. RETRIEVE TABLE INFORMATION.
      C
         85 CALL GBYTE (XVT1(XI),XI1,30,18)
            CALL GBYTE (XVT1(XI),XMD,48,2)
            XN1(1)=1
            DO 90 XJ=1,2
         90 CALL GBYTE (XVT2(XI),XN1(XJ+1),XJ*10-10,10)
      C
      C.....DECODE VARIABLE SUBSCRIPTS.
      C
            DO 95 XJ=1,3
               XN2(XJ)=1
```

```
73137              IF (XMCOL(XJ).LE.0) GO TO 95
73138              CALL GNUM (XSUB(XJ),1,XMCOL(XJ),XN2(XJ),XIERR)
73139              IF (XIERR.NE.0) GO TO 190
73140       95 CONTINUE
73141      C
73142      C.....COMPUTE ADDRESS OF THE VARIABLE RELATIVE TO FIRST WORD ADDRESS OF
73143      C.....VARIABLE.
73144      C
73145            XI2=0
73146            DO 100 XI=1,3
73147               XJ=4-XI
73148      100   XI2=XN1(XJ)*(XN2(XJ)-1+XI2)
73149      C
73150      C.....COMPUTE THE ADDRESS OF THE VARIABLE RELATIVE TO THE FIRST WORD OF
73151      C.....THE STORAGE BLOCK "XADRS(1)".
73152      C
73153            XLOC1=XI1+XI2
73154            XLOC2=-1
73155            XKOD1=2
73156            GO TO 15
```

Line Number	Explanation
	(Enter from line 73032)
73117-73123	An equal sign is encountered (a primary key character). Therefore the variable and its subscripts are completely parsed (filled in XNAME and XSUB). Search the variable table for variable name.
73127-73131	The variable name is located in the tables. Extract the mode and relative address. XI1 is the beginning of the variable name relative to the beginning of blank common (where the values of the variables are located). XMD is the type of the variable (declared by the user in STORAGE., REAL., or INTEGER. declaration statements. XN1(J) contains the value of the (J-1)th subscript of the variable in the tables.
73135-73140	Decode variable subscripts from data card. XN2(J) contains the integer value of the Jth subscript of the variable from the data card.
73145-73148	XI2 is the address of the element of the variable to be printed relative to the first word address of the variable.
73153-73156	XLØC1 is the address of the element of the variable to be printed relative to the first word address of the storage block (the beginning of blank common). XKØD1=2 indicating next characters are part of data item.

Line Number	Explanation
	EXAMPLE. REAL. FØX(15,2) was declared by user in the deck. This card would create an entry in the variable tables: LVR1(10)=XVT1(10)=0617305555 001760 2 000B 0617305555=3HFØX =XNM 1760 =1008_{10}=XI1 2 =SMD XVT2(10)=0000001111 0000000010 0000000000...0 0000001111=15=XN1(2) 0000000010=2 =XN1(3) FØX(3,1)=10.$ appeared in the data section creating values for local variables: XN2(1)=3 XN2(2)=1 XN2(3)=1 XI2=2 XLØC1=1008+2=1010 Therefore, the value 10. will be stored at XAPRS(1010).

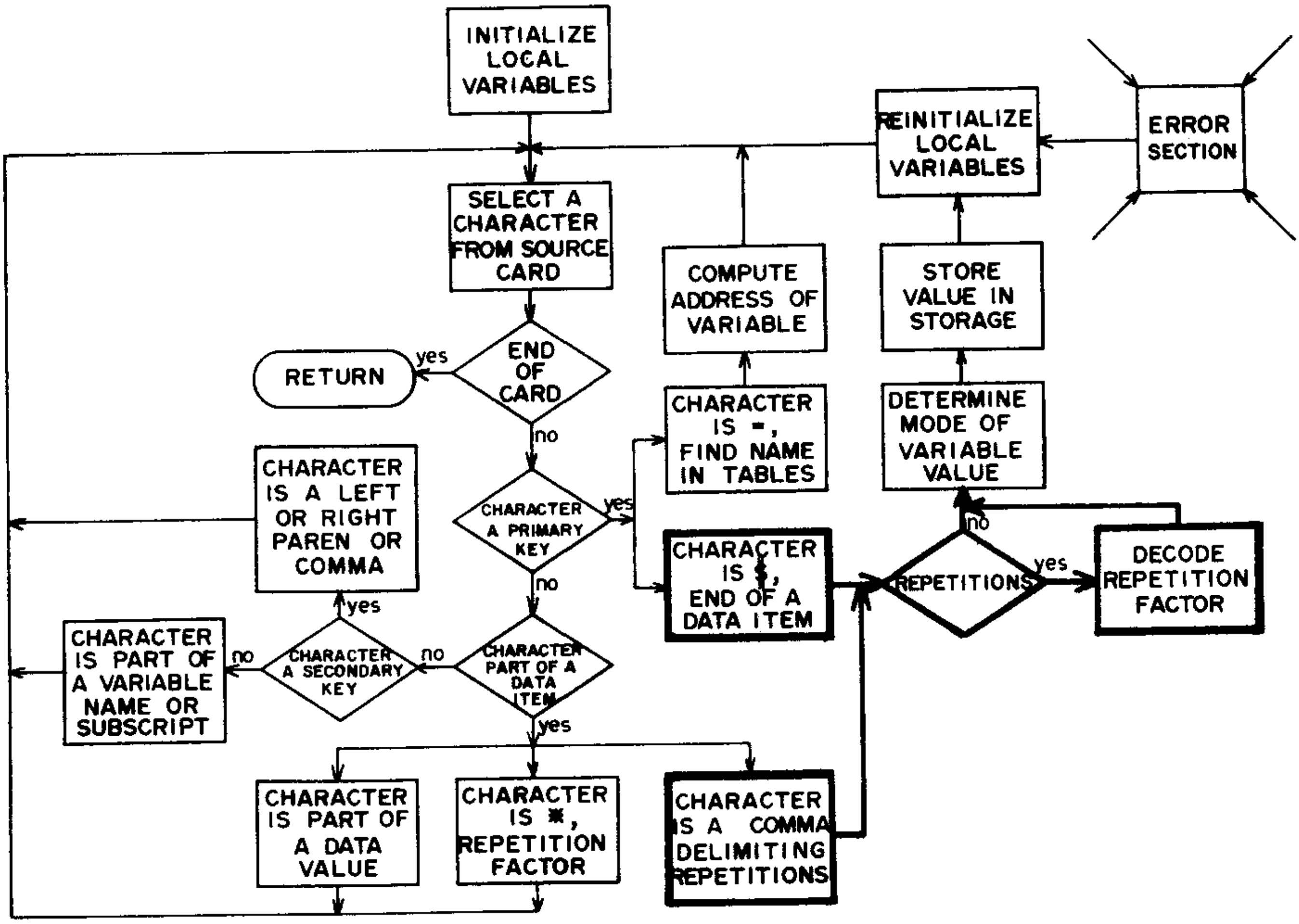
Character is part of a data item


INITIALIZE LOCAL VARIABLES
SELECT A CHARACTER FROM SOURCE CARD
END OF CARD
yes
RETURN
no
CHARACTER A PRIMARY KEY
yes
CHARACTER IS =, FIND NAME IN TABLES
COMPUTE ADDRESS OF VARIABLE
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
DECODE REPETITION FACTOR
no
DETERMINE MODE OF VARIABLE VALUE
STORE VALUE IN STORAGE
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
no
CHARACTER PART OF A DATA ITEM
no
CHARACTER A SECONDARY KEY
yes
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
no
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
C
C.....CHARACTER IS ASSUMED PART OF A DATA ITEM.
C
   60 IF (XICHR.EQ.1H,) GO TO 105
      IF (XICHR.EQ.1H*) GO TO 70
      IF (XICHR.GE.1H0.A.XICHR.LE.1H-) GO TO 65
      IF (XICHR.EQ.1H.) GO TO 65
      IF (XICHR.EQ.1HE) GO TO 65
      GO TO 155
   65 XLCOL=XLCOL+1
      IF (XICHR.EQ.1H.) XFLOAT=.TRUE.
      IF (XICHR.EQ.1HE) XFLOAT=.TRUE.
      IF (XLCOL.GT.20) GO TO 170
      CALL SCHARS (XNUM,XLCOL,1,XICHR)
      GO TO 15
C
C.....DATA ITEM CONTAINS REPETITION FACTOR.
C
   70 IF (XLCOL.GT.10) GO TO 175
      XREPT=XNUM(1)
      XNREPT=XLCOL
      XLCOL=0
      XNUM(2)=10H
      XNUM(1)=XNUM(2)
      GO TO 15
```

Line Number	Explanation
	Enter from line 73036 An equal sign has set XKØD1=2 indicating that following characters are part of a data item.
73091-73102	The character is part of a data value. XFLØAT is a logical flag set to .TRUE. if a character peculiar to real values is encountered (an E or a period). XNUM is filled with the characters of the data value. XLCØL is the number of characters in XNUM.
73106-73112	An asterisk is encountered indicating that the preceding characters were a repetition factor. XREPT contains the characters of the repetition factor. XNREPT is the number of characters in XREPT. XNUM and XLCØL are reinitialized to contain the data value which must follow the asterisk. EXAMPLE. NEWT(1,1)=15*0.0$ XREPT=10H15 XNREPT=2 XNUM=10H0.0 XLCØL=3 XFLØAT=.TRUE.

Process data delimiter value

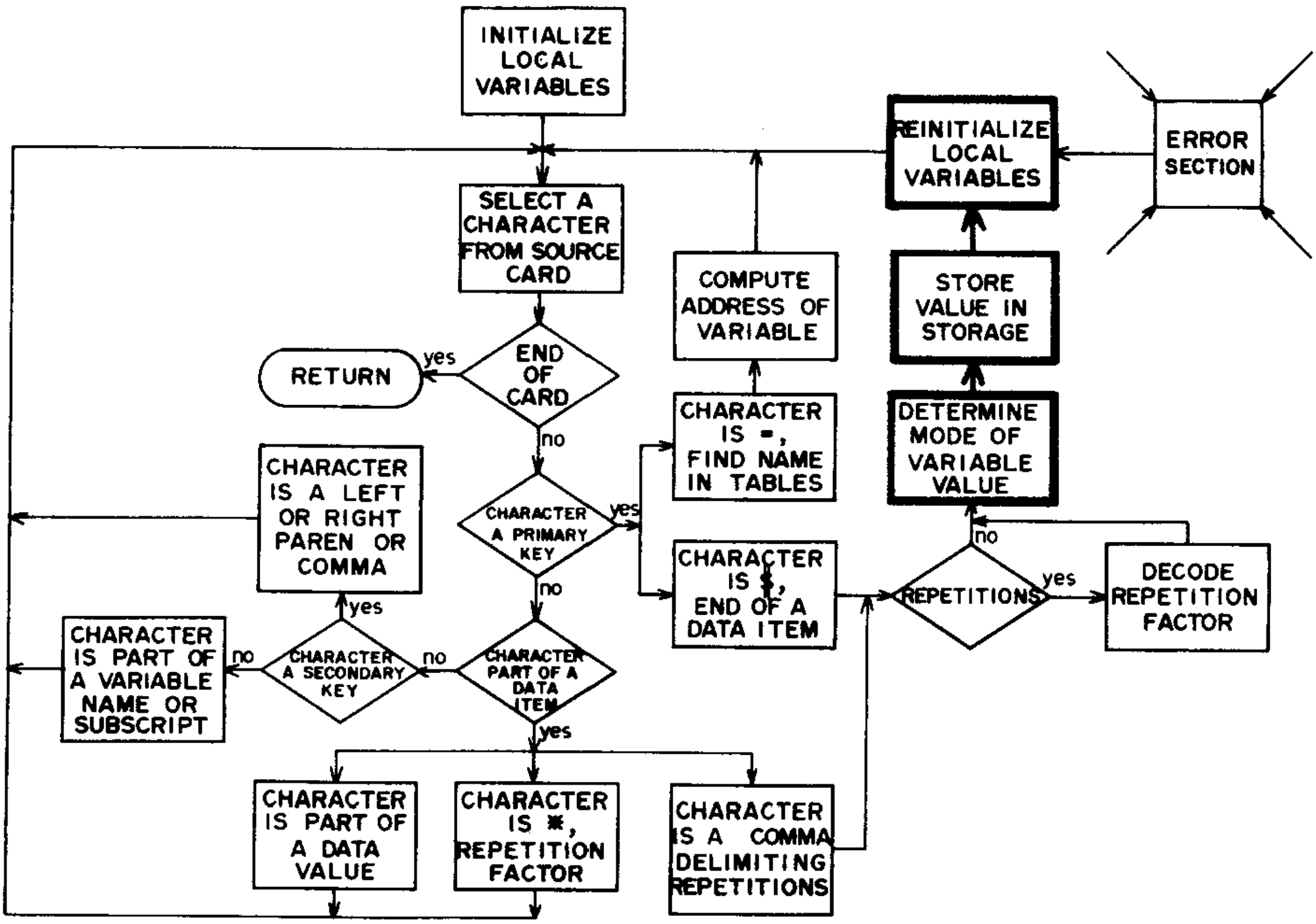

INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
      C
      C.....A DOLLAR "$" HAS BEEN ENCOUNTERED.
      C
        150 IF (XKOD1.NE.2) GO TO 155
            XSTOP=.TRUE.
            GO TO 105
```

```
      C
      C.....A COMPLETE DATA ITEM HAS BEEN RETRIEVED, DECODE ACCORDING TO
      C.....VARIABLE MODE AND STORE VALUE.
      C
        105 XTIMES=1
            IF (XNREPT.LE.0) GO TO 110
      C
      C.....DECODE REPETITION FACTOR.
      C
            CALL GNUM (XREPT,1,XNREPT,XTIMES,XIERR)
            IF (XIERR.NE.0) GO TO 195
            IF (XTIMES.LE.0) GO TO 200
```

Line Number	Explanation
	(Enter from line 73033).
73225-73230	A dollar sign delimits data statements, XSTØP=.TRUE. and indicates that a complete data statement has been scanned (variable name, subscripts, data value). Proceed to store the value in the tables (alternate entry from line 73091).
73157-73168	A comma delimits sequential data values; proceed to store the next value in the sequence. EXAMPLE. NEWT(1,1)=12,24,0,14$ XNREPT denotes the number of characters in the repetition factor (characters before an asterisk). If there was a repetition factor, then XTIMES contains the integer value of the repetition factor (if no repetition factor, XTIMES=1).

Store data value


INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
no
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

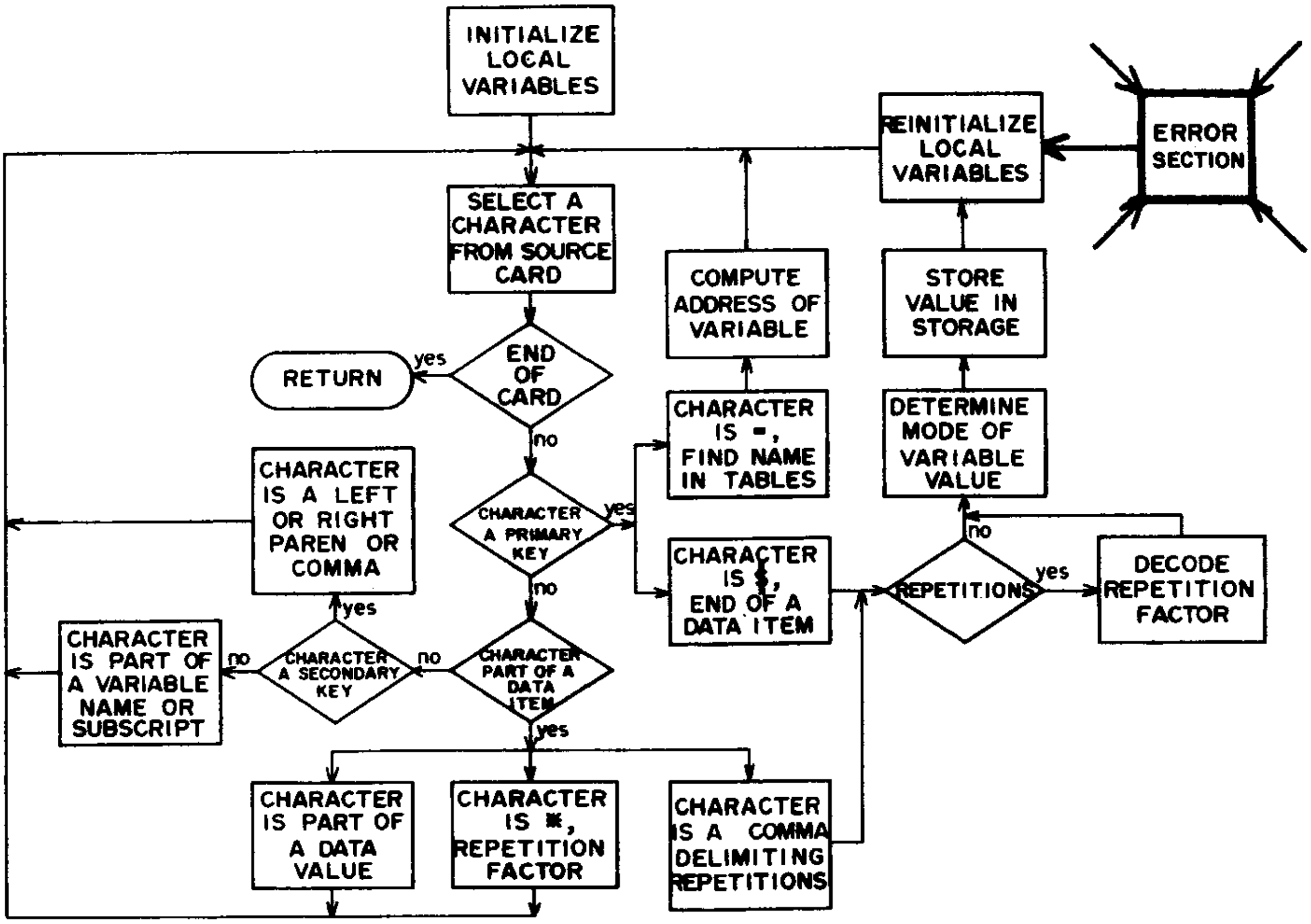
```
C
C.....DECODE DATA ITEM.
C
  110 IF (XLCOL.LE.0) GO TO 180
      XI=XMD+1
      GO TO (115,125), XI
C
C.....FIXED POINT MODE
C
  115 IF (.NOT.XFLOAT) GO TO 120
      IF (XWRITE.EQ.XNAME) GO TO 130
      XWRITE=XNAME
      WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,280) XNAME
      GO TO 130
  120 ENCODE (10,215,XFMT) XLCOL
      DECODE (XLCOL,XFMT,XNUM) XVAL
      GO TO 135
C
C.....FLOATING POINT MODE
C
  125 IF (XFLOAT) GO TO 130
      IF (XWRITE.EQ.XNAME) GO TO 130
      XWRITE=XNAME
      WRITE (XUO,225) (XJ,XJ=1,8),XCARD
      WRITE (XUO,285) XNAME
```

```
73195          130 ENCODE (10,220,XFMT) XLCOL
73196              DECODE (XLCOL,XFMT,XNUM) XVAL
73197         C
73198         C.....STORE VALUE IN STORAGE BLOCK.
73199         C
73200          135 DO 140 XI=1,XTIMES
73201                  XLOC2=XLOC2+1
73202                  XLOC=XLOC1+XLOC2
73203          140 XADRS(XLOC)=77777777777777777777B.A.XVAL
73204         C
73205         C.....REINITIALIZE LOCAL VARIABLES.
73206         C
73207          145 XLCOL=0
73208              XNUM(2)=10H
73209              XNUM(1)=XNUM(2)
73210              XNREPT=0
73211              IF (.N.XSTOP) GO TO 15
73212              XNCOL=0
73213              XNAME=10H
73214              XFLOAT=.FALSE.
73215              XMCOL(3)=0
73216              XMCOL(2)=XMCOL(3)
73217              XMCOL(1)=XMCOL(2)
73218              XSUB(3)=10H
73219              XSUB(2)=XSUB(3)
73220              XSUB(1)=XSUB(2)
73221              XKOD1=1
73222              XKOD2=1
73223              XSTOP=.FALSE.
73224              GO TO 15
```

Line Number	Explanation
73171-73174	The data value is decoded according to the type of the variable declared by a STØRAGE., REAL., or INTEGER. statement in the source deck. XMD=0 if the variable was declared integer. =1 if the declaration was real.
73178-73183	The variable was declared integer and the data value is of type real. A nonfatal diagnostic is issued. The value is decoded using real format. XFLØAT=.TRUE. if a period or an E was encountered while parsing the value. XWRITE prohibits issuing more than one diagnostic for one variable, if sequential data values are encountered. EXAMPLE. Declared integer variable, NEWT(1,1)=1.,0.,12.,14.,69.$ initiates the processing of one diagnostic instead of five, since XNAME=XWRITE for the last four values of NEWT.
73184-73186	The data value is type integer. XVAL contains the integer value of the data value characters on the data card (XNUM).
73190-73196	The data value of is type real. XVAL contains a floating point value of the XLCØL characters from XNUM.

Line Number	Explanation
73199-73203	The value of the variable is stored at its proper location relative to the beginning of blank common (XADRS(1)). XLØC2, initially set to -1, is used to store XTIMES repetitions in sequential locations in blank common. EXAMPLE. MØØSE(1)=3*2$ would set XTIMES=3. XADRS(XLØC)=XADRS(XLØC+1)=XADRS(XLØC+2)=2 thus storing three values in XADRS (blank common can be considered one array, with XADRS(1) the first location in blank common), beginning at the address of MØØSE(1) relative to XADRS(1).
73207-73211	Reinitialize local variables used for parsing a data value. XSTØP=.FALSE. indicates that there are sequential data values present associated with the same variable name. Therefore, proceed to parse next value and store it in XADRS immediately after the preceding value.
73212-73224	A complete data statement has been processed. Initialize all local variables, and proceed to scan for the beginning of the next data statement (a variable name).

Diagnostics

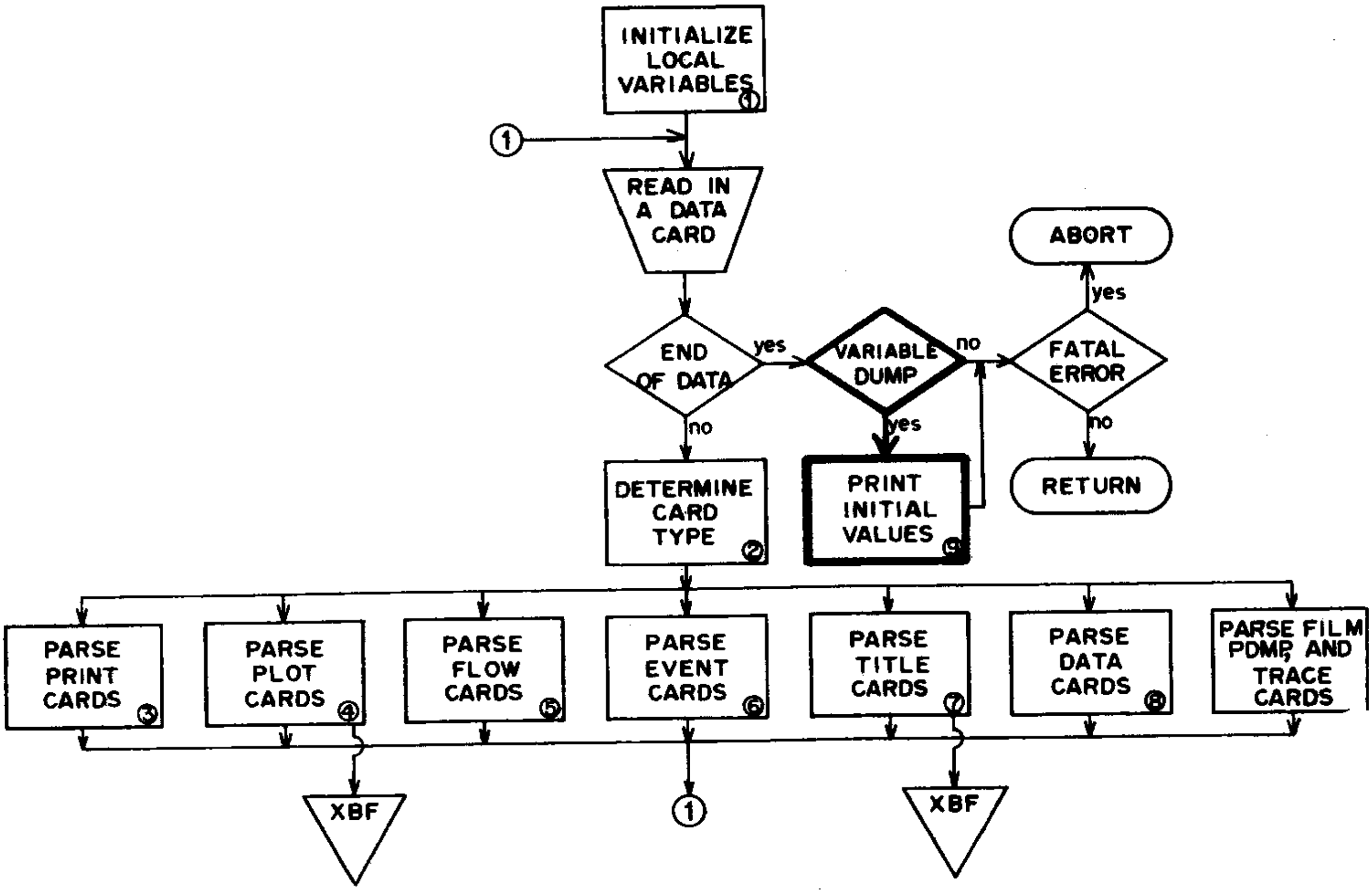

INITIALIZE LOCAL VARIABLES
REINITIALIZE LOCAL VARIABLES
ERROR SECTION
SELECT A CHARACTER FROM SOURCE CARD
COMPUTE ADDRESS OF VARIABLE
STORE VALUE IN STORAGE
RETURN
yes
END OF CARD
no
CHARACTER IS =, FIND NAME IN TABLES
DETERMINE MODE OF VARIABLE VALUE
CHARACTER IS A LEFT OR RIGHT PAREN OR COMMA
CHARACTER A PRIMARY KEY
yes
no
CHARACTER IS $, END OF A DATA ITEM
REPETITIONS
yes
DECODE REPETITION FACTOR
yes
CHARACTER IS PART OF A VARIABLE NAME OR SUBSCRIPT
no
CHARACTER A SECONDARY KEY
no
CHARACTER PART OF A DATA ITEM
yes
CHARACTER IS PART OF A DATA VALUE
CHARACTER IS *, REPETITION FACTOR
CHARACTER IS A COMMA DELIMITING REPETITIONS

```
      C
      C.....IF ERRORS FOUND GENERATE DIAGNOSTIC.
      C
        155 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,230) XICHR,XICOL
            GO TO 205
        160 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,235) XNAME
            GO TO 205
        165 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,240) XSUB(XI),XICOL
            GO TO 205
        170 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,245) XNUM
            GO TO 205
        175 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,250) XNUM(1)
            GO TO 205
        180 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
            WRITE (XUO,255) XICOL
            GO TO 205
        185 WRITE (XUO,290) (XJ,XJ=1,8),XCARD
            WRITE (XUO,260) XNAME
            GO TO 205
        190 WRITE (XUO,225) (XI,XI=1,8),XCARD
```

```
73256               WRITE (XUO,265) XSUB(XJ)
73257               GO TO 205
73258           195 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
73259               WRITE (XUO,270) XREPT
73260               GO TO 205
73261           200 WRITE (XUO,225) (XJ,XJ=1,8),XCARD
73262               WRITE (XUO,275) XREPT
73263         C
73264         C.....BELOW SEGMENT IS EXECUTED WHEN AN ERROR IS ENCOUNTERED. IT SEARCHE
73265         C.....REMAINDER OF CARD FOR A $ SIGN AND LOOPS BACK TO MAIN PROGRAM.
73266         C
73267           205 XFNI=.TRUE.
73268               XICOL=XICOL-1
73269           210 XICOL=XICOL+1
73270               IF (XICOL.GT.80) RETURN
73271               CALL GCHARS (XCARD,XICOL,1,XICHR)
73272               IF (XICHR.NE.1H$) GO TO 210
73273               XFNI=.FALSE.
73274               XSTOP=.TRUE.
73275               GO TO 145
73276         C
73277           215 FORMAT (2H(I,I2,6H)      )
73278           220 FORMAT (2H(E,I2,6H.0)    )
73279           225 FORMAT (6H0*****, 24HERROR IN DATA ASSIGNMENT,//T20,8I10/T20,8( 10
73280              1H1234567890)/T20,8A10)
73281           230 FORMAT (1H0,T14, 11HCHARACTER ",A1, 23H" IS ILLEGAL IN COLUMN ,I2)
73282           235 FORMAT (1H0,T14, 10HVARIABLE ",A5, 24H..." LONGER THAN 5 CHARS)
73283           240 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 35H..." LONGER THAN 4 CHARS IN
73284              1 COLUMN ,I2)
73285           245 FORMAT (1H0,T14, 11HDATA ITEM ",2A10, 25H..." LONGER THAN 20 CHARS
73286              1)
73287           250 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 25H..." LONGER T
73288              1HAN 10 CHARS)
73289           255 FORMAT (1H0,T14, 59HZERO LENGTH VARIABLE NAME OR DATA ITEM IN OR B
73290              1EFORE COLUMN ,I2)
73291           260 FORMAT (1H0,T14, 10HVARIABLE ",A5, 44H" WAS NOT DECLARED IN A <STO
73292              1RAGE.> STATEMENT)
73293           265 FORMAT (1H0,T14, 11HSUBSCRIPT ",A4, 15H" NOT DECODABLE)
73294           270 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 15H" NOT DECODAB
73295              1LE)
73296           275 FORMAT (1H0,T14, 24HDATA REPETITION FACTOR ",A10, 28H" LESS THAN O
73297              1R EQUAL TO ZERO)
73298           280 FORMAT (1H0,T14, 17HINTEGER VARIABLE ,A5, 46H WAS ASSIGNED A REAL
73299              1VALUE IN THE DATA SECTION)
73300           285 FORMAT (1H0,T14, 14HREAL VARIABLE ,A5, 50H WAS ASSIGNED AN INTEGER
73301              1 VALUE IN THE DATA SECTION)
73302           290 FORMAT (7H0**NF**,24HERROR IN DATA ASSIGNMENT,//T20,8I10/T20,8( 10
73303              1H1234567890)/T20,8A10)
73304         C
73305               END
```

Line Number	Explanation
73234-73262	Syntax errors are encountered while parsing a data card result in the generation of a diagnostic.
73267-73275	Search for the beginning of the first data statement following the error (set column counter, XICØL, to column number of the next dollar sign). Reinitialize local variables and continue scanning.

2.9 Print Initial Values



Overview

This section proceeds through the variable stack and writes all of the values of each variable in the stack onto the output file. The values of the variables are located in XADRS. If two or more sequential values in XADRS are equivalent and belong to the same variable name, then only one print entry is generated for these values.

The appearance of a NØNE. card in the data section prohibits the execution of this section. Appearance of an ALL. card prints first (normal) and second class storage variables in two separate tables. Default allows printing of one table containing first class variables only.

```
      SUBROUTINE XPRDMP (IPDMP)
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON /XXFL1WS/ XNFLW,XFLWT(1)
      COMMON /XXFL2WS/ XFLW(1)
      COMMON XADRS(1)
      INTEGER XLN,XNV,XNW,XVT1,XVT2,XN(3),XNDIM,XM(3),XUO,XUI,XNFLW,XI,X
     1VAL(2),XJ,XNAME,XK,XICOL,XJCOL,XICHR,XL,XCH(3),XA,XSTR(4)
      INTEGER XINC,XREP,XPRIM(2),XMBEG,XMO,XIO,XKO
      LOGICAL XIFREP
      LOGICAL XFLG,XINDF
C
C.....THIS ROUTINE PRINTS THE VALUES FOR THE SIMULATION CONTROL VARIABLE
C.....AND STATE VARIABLES, AND USER DEFINED VARIABLES.
C
C.....INITIALIZE LOCAL VARIABLES.
C
      DATA XCH/1H(,1H,,1H,/
      XIFREP=.FALSE.
      WRITE (XUO,215)
C
C.....PRINT SIMULATION CONTROL VARIABLES.
C
      IF (XNFLW.LE.0) GO TO 20
      WRITE (XUO,220)
      DO 15 XI=2,8
         CALL XHOL1V (XADRS(XI),XVAL)
   15 WRITE (XUO,225) XVT1(XI),XVAL
      GO TO 30
   20 XFLG=.TRUE.
      DO 25 XI=2,8
         IF (XINDF(XADRS(XI))) GO TO 25
         CALL XHOL1V (XADRS(XI),XVAL)
         IF (XFLG) WRITE (XUO,220)
         XFLG=.FALSE.
         WRITE (XUO,225) XVT1(XI),XVAL
   25 CONTINUE
C
C.....PRINT STATE VARIABLES.
C
   30 IF (XNFLW.LE.0) GO TO 95
      XFLG=.TRUE.
      XLN=0
      XMBFG=9
   35 DO 85 XI=XMBEG,1007
         IF (XINDF(XADRS(XI))) GO TO 85
         IF (XFLG) WRITE (XUO,230)
         XFLG=.FALSE.
         CALL XHOL1V (XADRS(XI),XPRIM)
         IF (XI.EQ.1007) GO TO 50
         XINC=-1
         XMO=XI+1
C
C.....CHECK TO SEE IF NEXT VARIABLE HAS SAME VALUE AS LAST.
C
         DO 40 XKO=XMO,1007
            CALL XHOL1V (XADRS(XKO),XVAL)
            IF (XVAL(1).NE.XPRIM(1)) GO TO 45
            IF (XVAL(2).NE.XPRIM(2)) GO TO 45
            XINC=XKO
   40    CONTINUE
   45    IF (XINC.LE.0) GO TO 50
         XJ=XI-8
         ENCODE (10,270,XREP) XJ
         XJ=XINC-8
         ENCODE (10,275,XNAME) XJ
         XIFREP=.TRUE.
         GO TO 55
   50    XJ=XI-8
```

```
         ENCODE (10,235,XNAME) XJ
         XREP=10H    X
         XIFREP=.FALSE.
C
C.....PRINT OUT THE VARIABLES, FOUR TO A LINE.
C
   55    XLN=XLN+1
         IF (XLN.GT.4) XLN=1
         GO TO (60,65,70,75), XLN
   60    WRITE (XUO,240) XREP,XNAME,XPRIM
         GO TO 80
   65    WRITE (XUO,245) XREP,XNAME,XPRIM
         GO TO 80
   70    WRITE (XUO,250) XREP,XNAME,XPRIM
         GO TO 80
   75    WRITE (XUO,255) XREP,XNAME,XPRIM
   80    IF (XIFREP) GO TO 90
   85 CONTINUE
      GO TO 95
   90 XMBEG=XINC+1
      IF (XMBEG.GT.1007) GO TO 95
      XIFREP=.FALSE.
      GO TO 35
C
C.....PRINT USER DECLARED VARIABLES.
C
   95 CONTINUE
      IF (IPDMP.GE.2) RETURN
      IF (XNV.LE.9) RETURN
      IPDMP=IPDMP+1
      DO 210 ICLS=1,IPDMF
      XFLG=.TRUE.
      CALL XPRB2F (XSTR,XDUM)
      DO 205 XI=10,XNV
C
C.....RETRIEVE VARIABLE CHARACTERISTICS.
C
         CALL GBYTE (XVT1(XI),JCLS,50,10)
         IF (ICLS-1.NE.JCLS) GO TO 205
         IF (.N.XFLG) GO TO 97
         XFLG=.FALSE.
         IF (ICLS.EQ.1) WRITE (XUO,260)
         IF (ICLS.EQ.2) WRITE (XUO,280)
   97    CONTINUE
         XNAME=(77777777770000000000B.A.XVT1(XI)).O.5555555555B
         CALL GBYTE (XVT1(XI),XK,30,18)
         XNDIM=1
         DO 100 XJ=1,3
            CALL GBYTE (XVT2(XI),XN(XJ),XJ*10-10,10)
            IF (XN(XJ).GT.0) XNDIM=XNDIM*XN(XJ)
            XM(XJ)=0
            IF (XN(XJ).GT.0) XM(XJ)=1
  100    CONTINUE
         XMBEG=1
         XIFREP=.FALSE.
  105    DO 195 XJ=XMBEG,XNDIM
C
C.....STORE VARIABLE NAME IN OUTPUT STRING.
C
            XICOL=0
            XJCOL=0
            XSTR(4)=10H
            XSTR(3)=XSTR(4)
            XSTR(2)=XSTR(3)
            XSTR(1)=XSTR(2)
  110       XICOL=XICOL+1
            CALL GCHARS (XNAME,XICOL,1,XICHR)
            IF (XICHR.EQ.1H ) GO TO 115
            XJCOL=XJCOL+1
            CALL SCHARS (XSTR,XJCOL,1,XICHR)
            GO TO 110
C
```

```
C.....STORE SUBSCRIPTS OF VARIABLE.
C
  115       DO 125 XL=1,3
               IF (XM(XL).LE.0) GO TO 130
               XJCOL=XJCOL+1
               CALL SCHARS (XSTR,XJCOL,1,XCH(XL))
               ENCODE (10,265,XA) XM(XL)
               XICOL=0
  120          XICOL=XICOL+1
               IF (XICOL.GT.10) GO TO 125
               CALL GCHARS (XA,XICOL,1,XICHR)
               IF (XICHR.EQ.1H ) GO TO 120
               XJCOL=XJCOL+1
               CALL SCHARS (XSTR,XJCOL,1,XICHR)
               GO TO 120
  125       CONTINUE
  130       CONTINUE
            XL=XK+XJ-1
            XINC=-1
            CALL XHOL1V (XADRS(XL),XVAL)
            IF (XNDIM.EQ.1) GO TO 170
            IF (XJ.GE.XNDIM) GO TO 165
C
C.....CHECK TO SEE IF SUCCEEDING SUBSCRIPTS HAVE THE SAME VALUE.
C
            XMO=XJ+1
            DO 135 XIO=XMO,XNDIM
               XKO=XK+XIO-1
               CALL XHOL1V (XADRS(XKO),XPRIM)
               IF (XVAL(1).NE.XPRIM(1)) GO TO 140
               IF (XVAL(2).NE.XPRIM(2)) GO TO 140
               XINC=XIO
  135       CONTINUE
  140       IF (XINC.LE.0) GO TO 165
            XIFREP=.TRUE.
C
C.....UPDATE VARIABLE SUBSCRIPTS TO LAST REPEATED VALUE.
C
            DO 150 XIO=XMO,XINC
               DO 145 XKO=1,3
                  IF (XM(XKO).LE.0) GO TO 150
                  XM(XKO)=XM(XKO)+1
                  IF (XM(XKO).LE.XN(XKO)) GO TO 150
                  XM(XKO)=1
  145          CONTINUE
  150       CONTINUE
            XJCOL=XJCOL+1
            CALL SCHARS (XSTR,XJCOL,1,1H-)
            XJCOL=XJCOL-1
            DO 160 XL=1,3
               IF (XM(XL).LE.0) GO TO 165
               XJCOL=XJCOL+1
               IF (XL.GT.1) CALL SCHARS (XSTR,XJCOL,1,1H,)
               ENCODE (10,265,XA) XM(XL)
               XICOL=0
  155          XICOL=XICOL+1
               IF (XICOL.GT.10) GO TO 160
               CALL GCHARS (XA,XICOL,1,XICHR)
               IF (XICHR.EQ.1H ) GO TO 155
               XJCOL=XJCOL+1
               CALL SCHARS (XSTR,XJCOL,1,XICHR)
               GO TO 155
  160       CONTINUE
  165       XJCOL=XJCOL+1
            CALL SCHARS (XSTR,XJCOL,1,1H))
  170       XJCOL=XJCOL+2
            CALL SCHARS (XSTR,XJCOL,1,1H=)
            XJCOL=XJCOL+1
C
C.....OUTPUT THE VARIABLE NAME AND VALUE.
C
```

```
76214                    XICOL=0
76215       175          XICOL=XICOL+1
76216                    IF (XICOL.GT.15) GO TO 180
76217                    CALL GCHARS (XVAL,XICOL,1,XICHR)
76218                    IF (XICOL.GT.11.A.XICHR.EQ.1H ) GO TO 180
76219                    XJCOL=XJCOL+1
76220                    CALL SCHARS (XSTR,XJCOL,1,XICHR)
76221                    GO TO 175
76222       180          CALL XPRBUF (XSTR,XJCOL)
76223      C
76224      C.....INCREMENT THE VARIABLE SUBSCRIPTS.
76225      C
76226                    XL=0
76227       185          XL=XL+1
76228                    IF (XL.GT.3) GO TO 190
76229                    IF (XM(XL).LE.0) GO TO 190
76230                    XM(XL)=XM(XL)+1
76231                    IF (XM(XL).LE.XN(XL)) GO TO 190
76232                    XM(XL)=1
76233                    GO TO 185
76234       190          IF (XIFREP) GO TO 200
76235       195       CONTINUE
76236                 GO TO 205
76237       200       XMBEG=XINC+1
76238                 XIFREP=.FALSE.
76239                 IF (XMBEG.GT.XNDIM) GO TO 205
76240                 GO TO 105
76241       205 CONTINUE
76242           CALL XPRBIF (XSTR,XDUM)
76243       210 CONTINUE
76244           RETURN
76245      C
76246       215 FORMAT ( 20H1SIMCOMP VERSION 3.0,10X, 16HPARAMETER VALUES,/)
76247       220 FORMAT (1H0,T38, 33H- SIMULATION CONTROL PARAMETERS -,/)
76248       225 FORMAT (1H ,T41,A5,3H = ,2A10)
76249       230 FORMAT (1H0,T45, 19H- STATE VARIABLES -,/)
76250       235 FORMAT (1H(,I3,6H) =    )
76251       240 FORMAT (1H ,T5,A5,A9,A10,A5)
76252       245 FORMAT (1H+,T36,A5,A9,A10,A5)
76253       250 FORMAT (1H+,T67,A5,A9,A10,A5)
76254       255 FORMAT (1H+,T98,A5,A9,A10,A5)
76255       260 FORMAT (1H0,T37,34H- PRIMARY USER DEFINED VARIABLES -,/)
76256       265 FORMAT (I10)
76257       270 FORMAT (2HX(,I3,5H      )
76258       275 FORMAT (1H-,I3,6H) =    )
76259       280 FORMAT (1H0,T36,36H- SECONDARY USER DEFINED VARIABLES -,/)
76260      C
76261           END
```

Line Number	Explanation
76000	IPDMP is a flag set in XINPUT to control printing of initial values. =0 List only first class storage variables. =1 List both first and second class variables. =2 NØNE. card is encountered. Suppress listing of user defined variables.
76021	Print the parameter dump page title.
76025-76030	Print the control variable values. If there were flows defined by user (XNFLW>0), then all simulation control variables and their values are printed.

Line Number	Explanation
	XVT1(2-8) contains the names of the control variables. XADRS(2-8) contains their values. XHØL1V is a general purpose routine that converts a floating point value into its equivalent BCD representation. XVAL contains the BCD representation of the value of XADRS(XI).
76031-76038	If no flows are defined by the user, then only control variables defined by user are printed. XINDF(X) returns a value of .TRUE. if X is indefinite. Print user defined state variables.
76042-76053	XLN is the number of variables written on a given line (four variables printed per line). XMBEG is the beginning index of XADRS from which sequential defined values are to be printed The values of the state values (X(1-999)) reside in XADRS(9-1007). All defined values in XADRS are to be printed. XPRIM contains the BCD value of the current location in XADRS. XI is the current index of XADRS. XMØ is the next index (location) in XADRS. XINC is greater than zero if two or more sequential locations in XADRS have the same value.
76057-76062	Check to determine if the next location in XADRS has the same value as the current value (XPRIM). XVAL is the BCD value of next location. Continue until a later value does not match the current value (then XINC will contain the address of the last matched value).
76063-76069	A series of locations from XI (current location) to (and including) XINC have identical values. XJ is the index of the current state variable. (XJ=1 for X(1) while the value of X(1) is in location XADRS(XI), XI=9). XREP is the encoded index of the current state variable (for X(1), XREP=10HX(1)). XNAME is the encoded index of last state variable having the same value as current state variable (if X(1)=X(2)=...X(10), then XNAME=10H= 10) =). XREP and XNAME are combined to generate sequential variables having the same value as a single print entry. XIFREP is turned "on" if there are sequential values equivalent to value of current index (XI).

Line Number	Explanation
76070-76073	The next location in XADRS is not equivalent to current location. XNAME is the encoded current state variable index (if XI=17, then XNAME=10H(17) =).
76077-76089	The state variables and their values are printed out, four print entries (columns) per line. XLN controls which of the four print columns that the current variable indices and value are printed on. EXAMPLE 1. Assume X(1)=X(2)...X(10)=0. and XI=9, the entry printed would be X(1- 10) = 0. EXAMPLE 2. Assume X(17)=4., XI=25 (that is, XADRS(17+8)=4.) and X(16)≠X(17)≠X(18). Then the print column would contain: X(17) = 4.
76090-76093	If identical repeated values existed, then the next "current" location is the location immediately following that of the last repeated value.
76097-76113	Print user declared variables (if IPDMP<2). If first and second class variable printing is desired, two passes through the variable stacks are made (line 76101). Otherwise, only one pass is made for first class variable printout. XPRBUF, XPRF1F, and XPRB2F are entries to the general purpose routine that print variables and their values in four columns onto output. XNV is the total number of variables in the variable tables (the first nine are system variables). JCLS is the class of the variable from variable stack (JCLS=0 for first class variable, JCLS=1 for second class). Values (XADRS(1008-...)) associated with user variables (XVT1(10-XNV)) are printed, four entries per line. XFLG controls printing of the first or second class variable title line.
76114-76125	XI is the current index of the variable stack. XNAME is the retrieved variable name. XK is the location of the value of first word address of variable name (the value of FØX(1,1) is stored at XADRS(XK)). XNDIM contains the total number of storage allocations for the current variable name. (If FØX was declared as STORAGE. FØX(15,2), then XNDIM=15×2=30 representing the number of sequential locations reserved for values of FØX in XADRS beginning at XADRS(XK).)

Line Number	Explanation
	XN(I) contains the Ith subscript of variable name (from above example, XN(1)=15, XN(2)=2). XM(I) contains the current subscripts (the beginning subscripts of FØX are FØX(1,1), therefore XM(1)=XM(2)=1).
76126-76141	All values associated with the current variable name will be printed. (There are XNDIM values.) XJ is the current index of the variable. XSTR is a buffer which is filled with the variable name, one of its subscripts, and the value of the variable associated with that subscript. XJCØL counts the number of characters filled into XSTR. The name of the variable is first filled into XSTR.
76144-76159	The current subscripts of the variable name are filled into the output string, together with appropriate delimiters. XCH contains a list of subscript delimiters. Therefore for first pass associated with variable name FØX, XSTR=10HFØX(1,1).
76160-76164	XL is the location (index) of the value of the current index of the variable name. XVAL is filled with the BCD characters of the value of the current index of the variable.
76168-76177	A check is made to determine if the next index (location) of the variable has the same value as the current index. XINC is the index of the last matched value.
76181-76210	Sequential values of the variable are equivalent to the current value (XIFREP=.TRUE.). The subscripts of the variable are updated to the last repeated value. XMØ contains the current updated subscripts. EXAMPLE. Assume the first value of FØX (i.e., FØX(1,1)) is stored at XADRS(XK) and XADRS(XK)=XADRS(XK+1)= XADRS(XK+2)=3. that is the first three values of FØX are equivalent. RECALL. XM(I) contains the current subscripts (initialized to the beginning subscripts of FØX, i.e., XM(1)=1, XM(2)=1). XM is updated, XM(1)=3, XM(2)=1 (which indicates that FØX(1,1)=FØX(2,1)=FØX(3,1). The updated subscripts are filled into the output string, thus XSTR=FØX(1,1-3,1).

Line Number	Explanation
76214-76222	The output string is completed with the addition of the value of the variable. XSTR=FØX(1,1-3,1) = 3. XSTR and the number of characters of XSTR (XJCØL) are sent to XPRBUF where they are printed as a column entry onto file output.
76226-76233	The current subscripts are updated to the next subscript value (thus XM(1)=4, XM(2)=1).
76234-76240	If there were repeated values, then the current index of the variable (XJ) must be the index following the last matched value (XINC+1). In this manner all values of each variable in the variable stack are printed onto file output. NOTE. Do not confuse the current index (XJ) of a variable with its current subscripts (XM). The index is just a linear count of the locations. Thus (assume STØRAGE. ANT (2,2,2)) XM(1)=1,XM(2)=1,XM(3)=1 XJ=1 XM(2)=2,XM(2)=1,XM(3)=1 XJ=2 XM(1)=1,XM(2)=2,XM(3)=1 XJ=3 XM(2)=2,XM(2)=2,XM(3)=1 XJ=4

CHAPTER 3. SIMULATION EXECUTION

Overview

This chapter is presented by a listing of the skeleton execution code with line-referenced commentary information. A flow chart is not included since the entire flow of control is by direction of the event schedular and event stack.

3.1. *Main Execution Loop*

```
      OVERLAY(NEWT1,1,0)
      PROGRAM XEXECTV
      COMMON /XXEVENT/ XEVADR,XFLMAX,XNEV,XDONE,XNEVIL,XEVIL(20,3)
      COMMON /XRVC/ XRF,XMP,XCORE,XXJ(17),XER,XRA
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XRF,XTRACE,XPLFG,XFILM,XNPL
C
C%%%%     5     0
C%%%%     6     0
C
      INTEGER XNW,XCNXT,XCORE,XEVADR,XFLMAX,XNEV,XNEVIL
      LOGICAL XDONE
      LOGICAL XTRACE
C
C.....ROUTINE "XEXECTV" IS THE SIMULATION EXECUTIVE.  CONTROL OF INITIAL
C.....ZATION, EVENT SEQUENCING, AND TERMINATION PROCESSING RESIDES HERE.
C
C.....READ IN AND PROCESS THE DATA SECTION.
C
      CALL XNCM (XCNXT)
      CALL XCFL (XCORE)
      XNEV=0
      XNEVIL=0
      CALL OVERLAY (4HMAIN,1,2)
      XCNXT=XCNXT+XNW+101
      CALL XRFL (XCNXT)
      CALL REMARK (23H   EXECUTING SIMULATION)
      XDONE=.FALSE.
C
C.....INITIALIZE THE EVENT SCHEDULER.
C
      XEVADR=LOCF(XEVSTK)
      XFLMAX=XCORE
      CALL XCSTART
C
C.....ENTER THE EVENT LOOP AND EXECUTE SIMULATION.
C
   15 CONTINUE
      IF (XNEV.LE.0) GO TO 20
      IF (XDONE) GO TO 20
C
C.....CHECK FOR INTEGRITY OF CURRENT ENTRY IN EVENT STACK.
C
      CALL XCHKSM(XEVSTK(XNEV-1),XEVSTK(XNEV))
C
C.....UPDATE "TIME" TO THE TIME OF THE NEXT EVENT.
C
      TIME=XEVSTK(XNEV)
C
C.....TRANSFER CONTROL TO THE NEXT EVENT ROUTINE.
C
      XTRNS=77777B.A.XEVSTK(XNEV-1)
      IF (XTRACE) CALL XEVTRC
      XNEV=XNEV-2
      CALL XTRNSF (XTRNS)
      GO TO 15
C
C.....IF THE EVENT STACK IS EXHAUSTED, EXECUTE SIMULATION TERMINATION
C.....PROCESSING AND RETURN CONTROL TO THE EXECUTION MONITOR.
C
   20 CONTINUE
      CALL XRFL (XCORE)
C
      END
```

```
              SUBROUTINE XEVTRC
              COMMON/XXUNITS/XUO.XUI,XUP.XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
              COMMON /XXEVENT/ XEVADR,XFLMAX.XNEV,XDONE,XNEVIL,XEVIL(20.3)
              COMMON /XXEXTRN/ XNEX.XEXT(1)
              COMMON XADRS(1)
              EQUIVALENCE (TIME,XADRS(2))
              INTEGER XNEV,XNEX,XEXT,XUO,XEVADR
              WRITE (XUO.25) TIME
              K=LOCF(XADRS)
              K=XEVADR-K
              DO 20 I=1,XNEV,2
                 IPR=1777000000B.A.XADRS(I+K)
                 IPR=SHIFT(IPR,-18)
                 IDRS=777777B.A.XADRS(I+K)
                 NAME=10H
                 TEVNT=XADRS(I+K+1)
                 DO 15 J=1,XNEX
                    JDRS=777777B.A.XEXT(J)
                    IF (IDRS.NE.JDRS) GO TO 15
                    NAME=XEXT(J)
                    GO TO 20
           15    CONTINUE
           20 WRITE (XUO,30) NAME.IDRS,IPR,TEVNT
              WRITE (XUO.35)
              RETURN
        C
           25 FORMAT (  7H0TRACE.,5X, 15HEVENT SCHEDULER,6X,  7HTIME = ,G15.9/)
           30 FORMAT (1H ,T10,A5,5X,O6,4X.O4,6X,G15.9)
           35 FORMAT (1H )
        C
              END
                  IDENT XTRNSF
                  ENTRY XTRNSF
        *....ROUTINE "XTRNXF" EXECUTES A RETURN JUMP TO THE ADDRESS PASSED IN
        *.     THE ARGUMENT OF THE FORTRAN CALLING SEQUENCE - CALL XTRNSF(ARG).
         XTRNSF   BSS    1                .
                  SX6    A0               .SAVE CONTENTS OF A0 TEMPA0
                  SA6    TEMPA0           .
                  SA2    X1               .X1=ADDRESS OF ARG, X2=CONTENTS OF ARG
                  LX2    30               .LEFT SHIFT JUMP ADDRESS 30 BITS
                  SA3    INST             .PUT EMPTY RJ INSTR. IN X3
                  BX6    X2+X3            .MASK JUMP ADDRESS INTO RJ INSTR, RESULT IN
         +        SA6 JUMPER
         JUMPER   BSSZ   1                .EXECUTE RJ
                  SA4    TEMPA0           .RESTORE A0
                  SA0    X4               .
                  EQ     XTRNSF           .RETURN CONTROL TO CALLING ROUTINE
         TEMPA0   BSS    1
         INST    DATA   0100000000460004600OB
             END
             SUBROUTINE HALT
             COMMON /XXEVENT/ XEVADR.XFLMAX,XNEV,XDONE,XNEVIL,XEVIL(20,3)
             LOGICAL XDONE
        C
        C.....THIS ROUTINE, WHEN SCHEDULED AS AN EVENT, WILL SET THE HALTING FLA
        C.....WHICH WILL IN TURN TERMINATE THE EVENT LOOP.
        C
             XDONE=.TRUE.
             RETURN
        C
             END



                 IDENT XEVSCH
                 ENTRY EVENT,CANCEL
                 LIST -R
                 TITLE EVENT SCHEDULER.
        *...THIS ROUTINE ENTERS OR REMOVES ENTRIES FROM THE EVENT STACK.  THE
        *     ROUTINE ACCEPTS THE FOLLOWING TWO CALLING SEQUENCES:
        *
        *     CALL EVENT(A1,A2,A3)
        *         THE MACHINE ADDRESS OF THE ROUTINE NAMED A1 SCHEDULED TO OCCURE
        *            AT TIME A2 AT PRIORITY A3 IS ENTERED.
        *
```

```
*   CALL CANCEL(A1,A2,A3)
*       THE FIRST TO OCCURE ENTRY OF ROUTINE NAMED A1 IS REMOVED FROM
*       THE EVENT STACK.  A2 IS IGNORED.  IN A3 IS RETURNED THE STATUS
*       OF THE CANCELLATION OPERATION:
*                0  -  ROUTINE FOUND IN STACK AND CANCELLED.
*                1  -  ROUTINE NOT FOUND IN STACK, NO ACTION TAKEN.
*                2  -  EVENT STACK EMPTY, NO ACTION TAKEN.
*
*...THE EVENT STACK IS THE LAST ARRAY IN BLANK COMMON (LOCATED AFTER THE
*   USER-DECLARED VARIABLES), NAMED XEVSTK.  THE STACK IS A PUSH-DOWN
*   STACK CONTAINING CONSECUTIVE PAIRS OF ENTRIES.  FOR THE I-TH EVENT
*   IN THE STACK THE FOLLOWING INFORMATION IS STORED:
*
*      LOCATION          BITS            CONTENTS
*    XEVSTK(2*I-1)       0-9      CHECKSUM OF REMAINING BITS IN PAIR.
*    XEVSTK(2*I-1)      32-41     PRIORITY LEVEL.
*    XEVSTK(2*I-1)      42-59     ENTRY ADDRESS OF EVENT ROUTINE.
*    XEVSTK(2*I)         0-59     TIME OF OCCURANCE (REAL VALUE).
*
*
 RTV.ADR  MACRO
*...SEARCH FOR A MATCH TO THE EVENT NAME IN THE EVENT CROSS-REFERENCE
*   TABLE.  IF FOUND, X0 IS SET THE THE ENTRY ADDRESS OF THE EVENT.
         LOCAL RT1,RT2
         MX3 30
         BX2 X1*X3
         SB2 XLST
         SA4 B2
         SB4 X4
         SB3 1
 RT1     SA4 B2+B3
         BX7 X4*X3
         BX5 X2*X7
         BX6 X2+X7
         BX6 -X6
         BX6 X5+X6
         BX6 -X6
         CX6 X6
         ZR X6,RT2
         SB3 B3+1
         GE B4,B3,RT1
         EQ ERR.2
 RT2     BX0 -X3*X4
         ENDM
*
 EVENT   BSSZ 1                .EVENT SCHEDULING.
         SX6 A0
         SA6 SV.A0
         SA0 A1
         SA1 XNEV              .CHECK FOR EVENT STACK OVERFLOW.
         SX1 X1+2
         SA2 EV.MAX
         IX3 X2-X1
         PL X3,FL.2
         ZR X3,FL.2
         SA3 EFL.LG
         NZ X3,ERR.1
         SA1 CFL.ADR
         RJ =XXCFL
         SA1 XFLMAX
         SA2 CFL.VAL
         IX3 X1-X2
         SA4 =1000B
         IX5 X4-X3
         NG X5,FL.1
         SX6 1
         SA6 EFL.LG
         BX4 X3
 FL.1    IX6 X2+X4
         SA1 EV.MAX
         IX7 X1+X4
         SA6 NFL.VAL
         SA7 EV.MAX
         SA1 NFL.ADR
         RJ =XXRFL
```

```
FL.2      SA1 A0                .GET ADDRESS OF EVENT, STORE IN X0.
          SA1 X1
          RTV.ADR
          SA1 A0+1              .LOAD TIME OF OCCURANCE AND CHECK FOR
          SA1 X1                .   FLOATING POINT VALUE.
          BX2 X1
          AX2 48
          NZ X2,EV.1
          PX1 X1
          NX1 X1
EV.1      SA2 A0+2              .LOAD PRIORITY AND CHECK FOR LEGAL VALUE
          SA2 X2
          BX3 X2
          AX3 48
          ZR X3,EV.2
          UX2 B1,X2
          LX2 B1,X2
EV.2      SA3 =1000B
          ZR X2,EV.3
          NG X2,EV.3
          IX4 X3-X2
          PL X4,EV.4
EV.3      BX2 X3
EV.4      SA3 XNEV              .INITIALIZE STACK SEARCH.
          SB7 X3
          SA5 XEVADR
          SB6 X5+B7
          SB5 0
EV.5      SB5 B5+2              .ENTER SEARCH LOOP.
          LT B7,B5,EV.7         .CHECK FOR SEARCH EXHAUSTED.
          SB4 B5-1
          SA3 B6-B4             .CHECK SCHEDULED TIMES.
          SA4 B6-B5
          RX6 X3-X1
          NX6 X6
          NG X6,EV.6
          NZ X6,EV.7
          BX6 X4                .TIMES EQUAL CHECK PRIORITIES.
          LX6 10
          AX6 28
          IX7 X6-X2
          ZR X7,EV.6
          PL X7,EV.7
EV.6      BX6 X3                .PUSH DOWN CURRENT STACK ENTRIES.
          SA6 A3+2
          BX6 X4
          SA6 A4+2
          EQ EV.5
EV.7      LX2 18                .BUILD NEW ENTRIES INCLUDING CHECKSUM.
          BX0 X0+X2
          CX6 X0
          CX7 X1
          IX2 X6+X7
          LX2 50
          BX6 X0+X2
          SB5 B5-2              .PLACE ENTRIES IN STACK AT CURRENT LOCATION
          SA6 B6-B5
          SB5 B5-1
          BX6 X1
          SA6 B6-B5
          SX6 B7+2
          SA6 XNEV
          SA1 SV.A0
          SA0 X1
          EQ EVENT
CANCEL    BSSZ 1                .EVENT CANCELLATION.
          SX6 A0
          SA6 SV.A0
          SA0 A1
          SX6 2                 .CHECK FOR ENPTY EVENT STACK, SET RETURN
          SA1 A0+2              .   STATUS OF 2.
          SA6 X1
          SA1 XNEV
          ZR X1,CN.RET
```

```
                SB7 X1
                SA1 A0                    .GET ADDRESS OF ROUTINE TO BE CANCELLED.
                SA1 X1
                RTV.ADR
                SX6 1                     .SET RETURN STATUS TO 1.
                SA1 A0+2
                SA6 X1
                SA1 XEVADR                .INITIALIZE STACK SEARCH.
                SB6 X1+B7
                MX5 42
                BX5 -X5
                SB5 2
       CN.1     SA1 B6-B5                 .ENTER SEARCH LOOP.
                BX1 X5*X1
                IX2 X0-X1
                ZR X2.CN.2                .CHECK FOR MATCH.
                SB5 B5+2
                LT B7.B5.CN.RET
                EQ CN.1
       CN.2     SX6 0                     .SET RETURN STATUS TO 0.
                SA1 A0+2
                SA6 X1
                SX6 B7-2                  .MATCH FOUND. PUSH DOWN ENTRIES. ELIMINA-
                SA6 XNEV                  .   TING MATCHED ENTRY.
       CN.3     SB5 B5-2
                ZR B5.CN.RET
                SB4 B5+2
                SA5 B6-B5
                BX6 X5
                SA6 B6-B4
                SB5 B5-1
                SB4 B4-1
                SA5 B6-B5
                BX6 X5
                SA6 B6-B4
                SB5 B5+1
                EQ CN.3
       CN.RET   SA1 SV.A0
                SA0 X1
                EQ CANCEL
       ERR.1    SB2 -OUT.F                .FIELD LENGTH EXHAUSTED.
                SB3 -FMT.1
                RJ =XOPUTCI.
                SA1 XNEV
                PX1 X1
                SA2 =2.
                FX6 X1/X2
                UX6 B1.X6
                LX6 B1.X6
                SA6 OP.VAL
                SB1 OP.VAL
                SB2 1
                RJ =XOUTPTC.
                SB1 -1
                RJ =XOUTPTC.
                RJ =XEXIT$
       ERR.2    SB2 -OUT.F                .EVENT DOES NOT EXIST.
                SB3 -FMT.2
                RJ =XOPUTCI.
                SA1 A0
                SA1 X1
                BX6 X1
                SA6 OP.VAL
                SB1 OP.VAL
                SB2 1
                RJ =XOUTPTC.
                SB1 -1
                RJ =XOUTPTC.
                RJ =XEXIT$
       SV.A0    BSS 1
       EV.MAX   DATA 100
       EFL.LG   BSSZ 1
       CFL.VAL  BSS 1
       CFL.ADR  VFD 60/CFL.VAL
```

```
 NFL.VAL  BSS 1
 NFL.ADR  VFD 60/NFL.VAL
 OUT.F    VFD 60/6LOUTPUT
 OP.VAL   BSS 1
 FMT.1    DATA C'(6H0*****,33HERROR DETECTED BY EVENT SCHEDULER,//T14,I5
,,65H EVENTS CURRENTLY SCHEDULED; EVENT STACK FULL; MORE CORE REQUIRED)'
 FMT.2    DATA C'(6H0*****,33HERROR DETECTED BY EVENT SCHEDULER,//T14
,,8HROUTINE ,A10,38H IS NOT A USER OR SYSTEM DEFINED EVENT)'
          USE /XXEVENT/
 XEVADR   BSS 1                .INITIALIZED IN (1,0) OVERLAY.
 XFLMAX   BSS 1                .INITIALIZED IN (1,0) OVERLAY.
 XNEV     BSSZ 1
 XDONE    BSS 1
 XNEVIL   BSSZ 1
 XEVIL    BSS 60
          USE /XXEXTPN/
 XLST     BSS 1
          USE 0
          END
          IDENT XCHKSM
          ENTRY XCHKSM
          LIST -R
          TITLE  CHECK CURRENT EVENT ENTRY CHECKSUM.
*...THIS ROUTINE COMPARES THE CHECKSUM BITS IN THE ENTRY FOR THE EVENT
*    ABOUT TO BE EXECUTED WITH THE COUNT OF BITS IN THE EVENT NOTICE.
*    IF THE CHECK SUM FAILS A DIAGNOSTIC IS ISSUED AND THE SIMULATION IS
*    TERMINATED.
 XCHKSM   BSSZ 1
          SA2 A1+1             .LOAD ENTRY TO BE CHECKSUMMED.
          SA1 X1
          SA2 X2
          LX1 10               .GET CHECK SUM VALUE.
          SX3 1777B
          BX4 X3*X1
          SB2 X4
          BX4 -X3*X1
          CX4 X4
          CX5 X2
          IX6 X4+X5
          SB3 X6
          EQ B2,B3,XCHKSM
          SB2 -OUT.F
          SB3 -ER.FMT
          RJ =XOPUTCI.
          SB1 -1
          RJ =XOUTPTC.
          SA1 200000B
          RJ =XEXIT$
 OUT.F    VFD 60/6LOUTPUT
 ER.FMT   DATA C'(6H0*****,33HERROR DETECTED BY EVENT SCHEDULER,//T14,69
,HEVENT STACK DESTROYED: CHECK FOR SUBSCRIPT OF USER VARIABLE TOO LARGE)
,'
          END
```

3.2. *System Initialization*

```
37000              SUBROUTINE XCSTART
37001        C
37002        C%%%%    5    0
37003        C%%%%    6    0
37004        C
37005              COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
37006              COMMON /XXPLOT/ XPLT(100),XRNG(2)
37007              COMMON /XXEVENT/ XEVADR,XFLMAX,XNEV,XDONE,XNEVIL,XEVIL(20,3)
37008              COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XRF,XTRACE,XPLFG,XFILM,XNPL
37009              INTEGER XNPR,XPRT,XNPL,XPLT,XUO,XNFLW,XFLWT,XNEX,XEXT,XNV,XNW,
37010             1   XVT1,XVT2,XN,XI,XNEV,XBIAS,XMODE,XI1,XI2,XI3,XMODN(2),XEVIL,
37011             2   XNEVIL
37012              LOGICAL XDONE,XINDF,XPLFG,XFILM,XFLPR,XTRACE,XSTART,XFINIS,XHALT
37013        C
37014        C%%%%    6    1
37015        C.....INITIALIZE THE SYSTEM DEFINED CRUSS-REFERENCE TAHLES.
37016        C
37017              DATA XMODN/7HINTEGER,4HREAL/
37018        C
37019        C%%%%    6    2
37020        C.....COMPUTE THE MACHINE ADDRESS FOR EACH OF THE EVENT ROUTINES IN THE
37021        C.....EXTERNAL EVENT REFFRENCE TABLE AND STORE IN THE HIGH ORDER 18 BITS
37022        C.....OF EACH REFERENCE LOCATION.
37023        C
37024              XN=0
37025        C
37026        C%%%%    6    3
```

Line Number	Explanation
37002-37003	The comment cards are instructions to the compiler to fill the contents of a given file into this location (Section 1.9).. Thus C%%%% 5 0 is a command interpreted by the compiler to print the contents of the first file of file name 5 into the text stream here. Common statements generated by the compiler; declared storage for variables, flows, and event names are entered here.
37014, 37019, 37026	EXTERNAL statements, DATA statements, and CARD sequences which calculate the machine address of each event routine are generated by the compiler and entered after the signal comment locations.

```
37027        C.....IF EXOGENOUSLY SCHEDULED EVENTS EXIST IN THE EVENT STACK THEN
37028        C.....INSERT THE MACHINE ADDRESS OF THE EVENT WHERE THE INDX FOR THE
37029        C.....EVENT WAS STORED.
37030        C
37031              XINC=100.
37032              IF (XFILM) XINC=800.
37033              XSTART=.TRUE.
```

```
37034              XFINIS=.TRUE.
37035              XHALT=.TRUE.
37036              IF (XNEVIL.LE.0) GO TO 20
37037              DO 15 XI=1,XNEVIL
37038                 CALL EVENT (XEVIL(XI,1),XEVIL(XI,2),XEVIL(XI,3))
37039                 IF (XEVIL(XI,1).EQ.5HSTART) XSTART=.FALSE.
37040                 IF (XEVIL(XI,1).EQ.5HFINIS) XFINIS=.FALSE.
37041                 IF (XEVIL(XI,1).EQ.4HHALT) XHALT=.FALSE.
37042           15 CONTINUE
37043           20 CONTINUE
```

Line Number   Explanation

37031-37035

XINC defines the number of plot locations available for paper or film plotting. It is later used to determine the optimum DTPL increment.

XSTART, XFINIS and XHALT are logical flags which, if set to .TRUE., cause the system to schedule routines START, HALT, and FINIS.

37036-37043

XNEVIL is the number of exogenously declared events (events declared by the user in the data section.)

XEVIL is a stack containing the exogenously declared events. XEVIL(XI,J) where J=1 is the XIth declared event name, J=2 is the time of occurrence, and J=3 is the priority.

Schedule all exogenously declared events.

Subroutine EVENT controls the scheduling of all events.

If the user scheduled START, FINIS, or HALT, turn the appropriate flags off so that the system will not also schedule them.

```
37044         C
37045         C.....IF TRACE IS REQUESTED WRITE OUT THE VARIABLE REFERENCE TABLE.
37046         C
37047               IF (.NOT.XTRACE) GO TO 30
37048               WRITE (XUO,135)
37049               DO 25 XI=1,XNV
37050                  XBIAS=SHIFT(7777770000B.A.XVT1(XI),-12)
37051                  XMODE=SHIFT(6000B.A.XVT1(XI),-10)+1
37052                  XI1=SHIFT(77740000000000000000B.A.XVT2(XI),-50).A.1777B
37053                  XI2=SHIFT(37760000000000000B.A.XVT2(XI),-40)
37054                  XI3=SHIFT(17770000000000B.A.XVT2(XI),-30)
37055            25 WRITE (XUO,140) XVT1(XI),XMODN(XMODE),XBIAS,XI1,XI2,XI3
37056               WRITE (XUO,145)
37057            30 CONTINUE
```

Line Number   Explanation

37044-37057

If a trace is requested by user (XTRACE=.TRUE. caused by a TRACE. card in the data deck), print out a variable reference table.

Line Number	Explanation
	XNV is the number of variables in the variable stacks (Section 1.3). XBIAS is the location of the value of the variable relative to the beginning of blank common. XMØDE is the variable type + 1. XI1, XI2, and XI3 are the variable subscripts. XVT1(XI) contains the name of the XIth variable in the stack. XMØDN(1)=10HINTEGER. (2)=10HREAL.

```
37058          C
37059          C.....INITIALIZE THE EVENT STACK, SCHEDULING THOSE EVENTS WHICH ARE
37060          C.....CONTROLLED BY SIMULATION CONTROL VARIABLES.
37061          C
37062                IF (XNFLW.LE.0) GO TO 60
37063          C
37064          C.....A CONTINUEOUS SIMULATION HAS BEEN DEFINED, INITIALIZE THE EVENT
37065          C.....SCHEDULAR FOR THIS CASE.
37066          C
37067                CALL XSETIME (TSTRT,TEND,DT,DTPR,DTPL,DTFL,XNPR,XFLPR)
37068                IF (XSTART) CALL EVENT (5HSTART,TSTRT,100)
37069                CALL EVENT (5HXCSIM,TSTRT,300)
37070                IF (XFINIS) CALL EVENT (5HFINIS,TEND,500)
37071                IF (XHALT) CALL EVENT (4HHALT,TEND,512)
37072          C
37073          C.....IF SYSTEM GENERATED OUTPUT IS DESIRED, <PRINT.> REQUESTS MUST
37074          C.....HAVE BEEN PRESENT.
37075          C
37076                IF (XNPR.LE.0) GO TO 35
37077                CALL EVENT (5HXPRNT,TSTRT,200)
37078          C
37079          C.....IF SYSTEM GENERATED PLOTTING IS DESIRED, <PLOT.> REQUESTS MUST
37080          C.....HAVE BEEN PRESENT.
37081          C
37082             35 IF (.N.XPLFG) GO TO 55
37083                CALL EVENT (5HXPLOT,TSTRT,200)
37084                IF (DTPL.GT.0.) GO TO 55
```

Line Number	Explanation
37062	XNFLW is the number of flows. If there are flow commands in the source deck, the simulation is continuous. Lines 37067-37084 initialize the event scheduler for this case.
37067	XSETIME is a utility program which defines values for system variables (TSTRT,TEND,DT, ...) not initialized by the user.

Line Number	Explanation
37068-37071	The system schedules routines START, FINIS, and HALT if they were not exogenously scheduled by the user; schedule XCSIM, which controls calculations of the flow commands at time TSTRT.
37072-37077	If PRINT. commands were encountered (XNPR is the number of variables in the print stack), XPRNT is scheduled by routine EVENT. (XPRNT directs the printing of desired variables, variables following PRINT. commands, and is scheduled at TIME=TSTRT, TSTRT+DTPR, TSTRT+2*DTPR, etc.)
27082-37084	If PLØT. commands were encountered (XPLFG set to .TRUE. in routine XPLTSTK), then event XPLØT is scheduled by the system.

```
37085         C
37086         C.....TIME INTERVAL FOR PLOT GENERATION HAS NOT BEEN EXOGENOUSLY DEFINED
37087         C.....COMPUTE RESOLUTION OPTIMUM FOR PLOT.
37088         C
37089               IF (XRNG(1).NE.XRNG(2)) GO TO 40
37090               IF (XRNG(1).EQ.1.) DTPL=(TEND-TSTRT)/XINC
37091               IF (XRNG(1).NE.1.) DTPL=DT
37092               GO TO 45
37093            40 XDT1=(XRNG(2)-XRNG(1))/XINC
37094               XDT2=(TEND-TSTRT)/XINC
37095               DTPL=AMIN1(XDT1,XDT2)
37096            45 DTPL=AMAX1(DTPL,DT)
37097               WRITE (XUO,170) DTPL
```

Line Number	Explanation
37085-37097	If DTPL was not set by user (then DTPL=0, line 37084), then system computes the optimum DTPL. There are three possible cases: (1) XRNG(1)≠XRNG(2). (Variable XRNG is filled in routine XPLTSTK.) If independent range declarations are specified with variable TIME as the independent variable, XRNG will contain the range values having the minimum difference. This difference (divided by the number of plot positions, XINC) is compared to the difference TEND-TSTRT. DTPL is assigned the smallest difference or DT whichever is the larger.

Line Number	Explanation
	(2) XRNG(1)=XRNG(2)=1. This signals that TIME is the independent variable of at least one plot and no range declarations are assigned with TIME being the independent variable. Therefore, DTPL is assigned the larger of TEND-TSTRT (divided by XINC) and DT. (3) XRNG(1)=XRNG(2)=0. This signals that TIME is never the independent variable, set DTPL=DT.

```
37098           C
37099           C.....IF FLOW PRINTING REQUESTED AT REGULAR INTERVALS, SCHEDULE THEIR
37100           C.....OCCURANCE.
37101           C
37102              55 IF (.N.XFLPR) RETURN
37103                 CALL EVENT (5HXFLOP,TSTRT,400)
37104                 RETURN
```

Line Number	Explanation
37102-37104	If flow print requests were present (XFLPR set to .TRUE. upon encountering FLØW. commands in user's data section), event XFLØP is scheduled to initially occur at TIME=TSTRT. No flow commands were present; the user specified events only.

```
37105           C
37106           C.....THE SIMULATION CONTAINS EVENTS ONLY. SCHEDULE SYSTEM DEFINED EVENT
37107           C.....IF REQUESTED.
37108           C
37109              60 IF (.N.XINDF(TSTRT)) GO TO 70
37110                 IF (XNEV.GT.0) GO TO 65
37111                 WRITE (XUO,150)
37112                 RETURN
37113              65 TSTRT=XEVSTK(XNEV)
37114                 WRITE (XUO,155) TSTRT
37115              70 IF (XSTART) CALL EVENT (5HSTART,TSTRT,100)
37116                 IF (XINDF(TEND)) GO TO 80
37117                 IF (TEND.GE.TSTRT) GO TO 75
37118                 WRITE (XUO,160)
37119                 TEND=1777 0000 0000 0000 0004B
37120                 GO TO 80
37121              75 IF (XFINIS) CALL EVENT (5HFINIS,TEND,500)
37122                 IF (XHALT) CALL EVENT (4HHALT,TEND,512)
```

Line Number	Explanation
37109-37114	If TSTRT is not defined by user, TSTRT is set to the scheduled time of the first event to occur.
37115-37122	START is scheduled if it was not exogenously scheduled by user. If TEND is defined, FINIS and HALT are scheduled if they were not exogenously defined by user. (If TEND is less than TSTRT, TEND is assumed undefined.)

```
37123          C
37124          C.....IF PRINT. REQUESTS PRESENT. CHECK FOR LEGAL DTPR.
37125          C
37126             80 IF (XNPR.LE.0) GO TO 100
37127                IF (XINDF(DTPR)) GO TO 85
37128                IF (DTPR.LE.0.) GO TO 85
37129                GO TO 95
37130             85 IF (XINDF(TEND)) GO TO 90
37131                DTPR=(TEND-TSTRT)/10.
37132                WRITE (XUO,165) DTPR
37133                GO TO 95
37134             90 DTPR=1.
37135                WRITE (XUO,165) DTPR
37136             95 CALL EVENT (5HXPRNT,TSTRT,200)
```

Line Number	Explanation
37125-37136	If print requests were present (XNPR = number of variables in the print stack) and DTPR is undefined (or <0), DTPR is calculated so that 10 prints through the simulation will be outputted. Schedule the first print to occur at TIME=TSTRT (event XPRNT controls the printing of user variable values).

```
37137          C
37138          C.....IF PLOT. REQUESTS PRESENT. CHECK FOR LEGAL DTPL.
37139            100 IF (.N.XPLFG) RETURN
37140                IF (XINDF(DTPL)) GO TO 105
37141                IF (DTPL.LE.0.) GO TO 105
37142                GO TO 120
37143            105 IF (XRNG(1).NE.XRNG(2)) GO TO 115
37144                IF (XRNG(1).NE.1.) GO TO 110
37145                IF (XINDF(TEND)) GO TO 110
37146                DTPL=(TEND-TSTRT)/XINC
37147                WRITE (XUO,170) DTPL
37148                GO TO 120
37149            110 DTPL=1.
37150                WRITE (XUO,170) DTPL
37151                GO TO 120
37152            115 DTPL=(XRNG(2)-XRNG(1))/XINC
37153                WRITE (XUO,170) DTPL
```

```
37154          120 CALL EVENT (5HXPLOT,TSTRT,200)
37155              RETURN
37156        C
37157          135 FORMAT (  7HOTRACE.,5X, 20HVARIABLE DEFINITIONS,/T11,  4HNAME,5X,
37158             1 4HMODE,9X,  4HBIAS,10X, 10HDIMENSIONS,/)
37159          140 FORMAT (1H ,9X,A5,5X,A7,6X,I6,6X,3(I4,2X))
37160          145 FORMAT (1H )
37161          150 FORMAT (66H0*****NO FIRST EVENT IS SCHEDULED, THE SIMULATION WILL
37162             1NOT EXECUTE)
37163          155 FORMAT (81H0*****TSTRT IS UNDEFINED AND WILL BE GIVEN THE VALUE OF
37164             1 THE FIRST EVENT, TSTRT = ,G15.9)
37165          160 FORMAT (53H0*****TEND .LE. TSTRT, TEND WILL BE ASSUMED UNDEFINED)
37166          165 FORMAT (67H0*****DTPR .LE. 0 OR UNDEFINED AND WILL BE GIVEN THE VA
37167             1LUE, DTPR = ,G15.9)
37168          170 FORMAT (67H0*****DTPL .LE. 0 OR UNDEFINED AND WILL BE GIVEN THE VA
37169             1LUE, DTPL = ,G15.9)
37170        C
37171              END
```

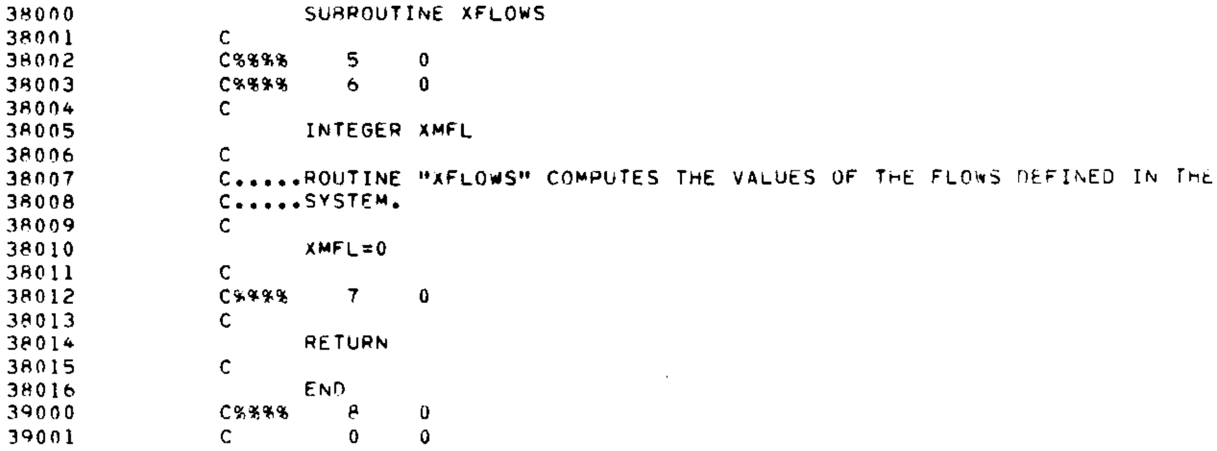
Line Number	Explanation
37139-37153	If plot requests were present and DTPL was not set by the user, set DTPL to the optimum value. (1) XRNG(1)≠XRNG(2). XRNG contains the independent variable range declaration having the minimum difference (with TIME the independent variable). DTPL is this difference divided by the number of plot positions (XINC). (2) XRNG(1)=XRNG(2)=1. This signals that TIME is the independent variable of at least one plot and that no range declarations were specified with TIME as the independent variable. DTPL=(TEND-TSTRT)/XINC. (3) XRNG(1)=XRNG(2)=0. TIME is never the independent variable; set DTPL=1.
37154-37155	Schedule event XPLØT which stores plot values to first occur at TIME=TSTRT.

```
40000              SUBROUTINE XSETIME (S,E,DT,DTPR,DTPL,DTFL,XNPR,XFLPR)
40001              COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
40002              INTEGER XUO,XNPR
40003        C
40004        C.....XSETIME DETERMINES VALUES FOR TSTRT, TEND, DT, DTPL, DTPR, AND
40005        C        DTFL IF ANY OR ALL OF THESE VARIABLES ARE NOT INITIALIZED BY
40006        C        THE USER.
40007        C
40008              LOGICAL XINDF,XPLFG,XFLPR
40009              K=1
40010              IF (XINDF(S)) K=K+4
40011              IF (XINDF(E)) K=K+2
40012              IF (XINDF(DT)) K=K+1
40013              GO TO (50,45,40,35,30,25,20,15), K
40014           15 S=0.
40015              E=10.
```

```
      DT=1.
      GO TO 50
   20 S=0.
      E=10.*DT
      GO TO 50
   25 S=E-10.
      DT=1.
      GO TO 50
   30 S=E-10.*DT
      GO TO 50
   35 E=S+10.
      DT=1.
      GO TO 50
   40 E=S+10.*DT
      GO TO 50
   45 DT=(E-S)/10.
   50 IF (DT.GT.0..A.S.LE.E) GO TO 53
      K=1
      S=0.
      E=10.
      DT=1.
      WRITE (XUO,105) S,E,DT
   53 IF (K.NE.1) WRITE (XUO,85) S,E,DT
      IF (XNPR.LE.0) GO TO 60
      IF (XINDF(DTPR)) GO TO 55
      IF (DTPR.LE.0.) GO TO 55
      IF (DTPR.GE.DT) GO TO 60
      WRITE (XUO,90) DT,DTPR
      DTPR=DT
      GO TO 60
   55 XDT=(E-S)/10.
      DTPR=AMAX1(XDT,DT)
      WRITE (XUO,95) DTPR
   60 IF (.NOT.XINDF(DTPL)) GO TO 65
      IF (.NOT.XPLFG) GO TO 70
      DTPL=0.
   65 IF (DTPL.EQ.0.) GO TO 70
      IF(DTPL.GE.DT) GO TO 70
      WRITE (XUO,100) DT,DTPL
      DTPL=DT
   70 IF (.NOT.XFLPR) GO TO 80
      IF (XINDF(DTFL)) GO TO 75
      IF (DTFL.LE.0.) GO TO 75
      IF(DTFL.GE.DT) GO TO 80
      WRITE (XUO,110) DTFL,DT
      DTFL=DT
      GO TO 80
   75 XDT=(E-S)/10.
      DTFL=AMAX1(XDT,DT)
      WRITE (XUO,115) DTFL
   80 RETURN
C
   85 FORMAT (6H0*****, 58HTSTRT, TEND, AND/OR DT WERE UNDEFINED, VALUES
     1 SELECTED ARE,/,T14,  8HTSTRT = ,G15.9,/,T14,  8HTEND   = ,G15.9,/,
     2T14,  7HDT    = ,G15.9)
   90 FORMAT (6H0*****,.57HSINCE DTPR IS < DT, DTPR WILL BE ASSIGNED THE
     1 VALUE OF DT,/,T14.  7HDTPR = ,G15.9,/,T14,  7HDT    = ,G15.9)
   95 FORMAT (6H0*****,81HDTPR .LE. 0 OR UNDEFINED, DTPR IS ASSIGNED THE
     1 MAXIMUM OF (TEND-TSTRT)/10. AND DT/T14,7HDTPR = ,G15.9)
  100 FORMAT (6H0*****,54HSINCE DTPL < DT, DTPL WILL BE ASSIGNED THE VAL
     1UE OF DT,/,T14,  7HDTPL = ,G15.9,/,T14,  8HDT     = ,G15.9)
  105 FORMAT (6H0*****,58HTSTRT .GT. TEND OR DT .LE. ZERO, DEFAULT VALUE
     1S CHOSEN ARE/T14,8HTSTRT = ,G15.9/T14,8HTEND   = ,G15.9/T14,8HDT
     2 = ,G15.9)
  110 FORMAT (6H0*****, 54HSINCE DTFL < DT, DTFL WILL BE ASSIGNED THE VA
     1LUE OF DT,/,T14.  7HDTFL = ,G15.9,/,T14,  7HDT    = ,G15.9)
  115 FORMAT (6H0*****,81HDTFL .LE. 0 OR UNDEFINED, DTFL IS ASSIGNED THE
     1 MAXIMUM OF (TEND-TSTRT)/10. AND DT/T14,7HDTFL = ,G15.9)
C
      END
```

Line Number	Explanation
	XSETIME determines values for any system variables not defined by the user.
40009-40013	Branch to the appropriate combination of undefined variables.
40014-40017	TSTRT(S), TEND(E), and DT are all undefined. Set default values for each of them.
40018-40020	TSTRT and TEND are indefinite.
40021-40023	TSTRT and DT are indefinite.
40024-40025	Calculate a value for TSTRT.
40026-40028	Determine values for TEND and DT.
40029-40030	TEND is undefined.
40031	DT is undefined.
40032-40037	If DT<0 or TEND<TSTRT, set default values.
40042-40045	DTPR<DT; a nonfatal message is issued and sets DTPR=DT. Variable prints should not occur more often than the simulation step size.
40046-40048	DTPR is undefined. Calculate a value of DTPR and issue a nonfatal diagnostic.
40049-40051	DTPL is undefined. Set DTPL=0. to cause optimum setting of value for DTPL (in XCSTART).
40052-40055	DTPL<DT; set DTPL=DT. The plot interval should not be less than the simulation step size.
40056-40062	If DTFL<DT (and flow requests are present), DTFL=DT.
40063-40066	DTFL is undefined (or ≤ 0). Determine a value for DTFL.

3.3. *Flow Calculations*



```
38000                SUBROUTINE XFLOWS
38001          C
38002          C%%%%     5     0
38003          C%%%%     6     0
38004          C
38005                INTEGER XMFL
38006          C
38007          C.....ROUTINE "XFLOWS" COMPUTES THE VALUES OF THE FLOWS DEFINED IN THE
38008          C.....SYSTEM.
38009          C
38010                XMFL=0
38011          C
38012          C%%%%     7     0
38013          C
38014                RETURN
38015          C
38016                END
39000          C%%%%     8     0
39001          C         0     0
```

Line Number	Explanation
38002-38003	User variable declarations are inserted at the flagged comment C%%%% 5 0. The system variable common declarations containing the variables XNFLW, XFLWT, and XFLW (the flow tables) which are computed in XFLØWS are inserted at C%%%% 6 0.
38012	The FØRTRAN text of the user's flow calculations is inserted here. After compilation this routine is called to calculate the flows declared by the user. The values of the flows are stored in array XFLW by the generated FØRTRAN text (refer to Section 1.8 for the details of what text is generated and inserted here).

3.4. *Updating of User's State Variables*

```
46000                 SUBROUTINE XCSIM
46001                 COMMON /XXFL1WS/ XNFLW,XFLWT(1)
46002                 COMMON /XXFL2WS/ XFLW(1)
46003                 COMMON XADRS(1)
46004                 DIMENSION X(999)
46005                 EQUIVALENCE (TIME,XADRS(2)), (DT,XADRS(5)), (X(1),XADRS(9))
46006                 INTEGER XI,XJ,XK,XNFLW,XFLT,XMSK1,XMSK2
46007                 DATA XMSK1/7777700000B/
46008                 DATA XMSK2/77777B/
46009           C
46010           C.....XCSIM IS THE SYSTEM DEFINED EVENT WHICH EXECUTES THE CONTINUOUS
46011           C.....PORTION OF THE SIMULATION.
46012           C
46013           C.....CALCULATE THE FLOWS.
46014           C
46015                 CALL CYCL1
46016                 CALL XFLOWS
46017           C
46018           C.....UPDATE THE STATE VARIABLES.
46019           C
46020                 DO 15 XK=1,XNFLW
46021                    XI=SHIFT(XFLWT(XK).A.XMSK1,-15)
46022                    XJ=XFLWT(XK).A.XMSK2
46023                    X(XI)=X(XI)-XFLW(XK)*DT
46024                    X(XJ)=X(XJ)+XFLW(XK)*DT
46025              15 CONTINUE
46026                 CALL CYCL2
46027           C
46028           C.....RESHEDULE THIS EVENT AT (TIME+DT).
46029           C
46030                 CALL EVENT (5HXCSIM,TIME+DT,300)
46031                 RETURN
46032           C
46033                 END
```

Line Number	Explanation
	XCSIM is a system event initiated by the appearance of flow commands in the user's program. Each occurrence of XCSIM executes the flows defined by the user and updates the values of the state variables associated with each flow. XCSIM reschedules itself to next occur at time = TIME+DT.
46015	CYCL1 is a dummy routine. However, if the user writes a CYCL1 routine, it replaces the dummy routine and is called by the system immediately before the flows are calculated.
46016	XFLØWS contains the text which calculates each flow. (The information placed on file FLTX is filled into XFLØWS by the compiler, see Section 1.5.)

Line Number	Explanation
46020-46025	Update the state variables associated with each flow. The *source* state variable is decremented by the calculated FLØW value times DT and the target (sink) state variable is incremented by this amount. (This section updates the state variables by the integral of the flow.) XNFLW is the number of (expanded) flow commands. XFLW(I) is the calculated value (via XFLØWS) of the Ith flow in the stack. XFLWT(I) contains the indices of the two state variables that define the Ith (expanded) flow. The values of these variables change with the integral of the Ith calculated flow value. XI is the index of the source state variable.
46026-46030	CYCL2 is a dummy routine which may be replaced by a user written CYCL2. The routine is executed after the state variables have been updated. XCSIM reschedules itself to next appear at time = TIME+DT.

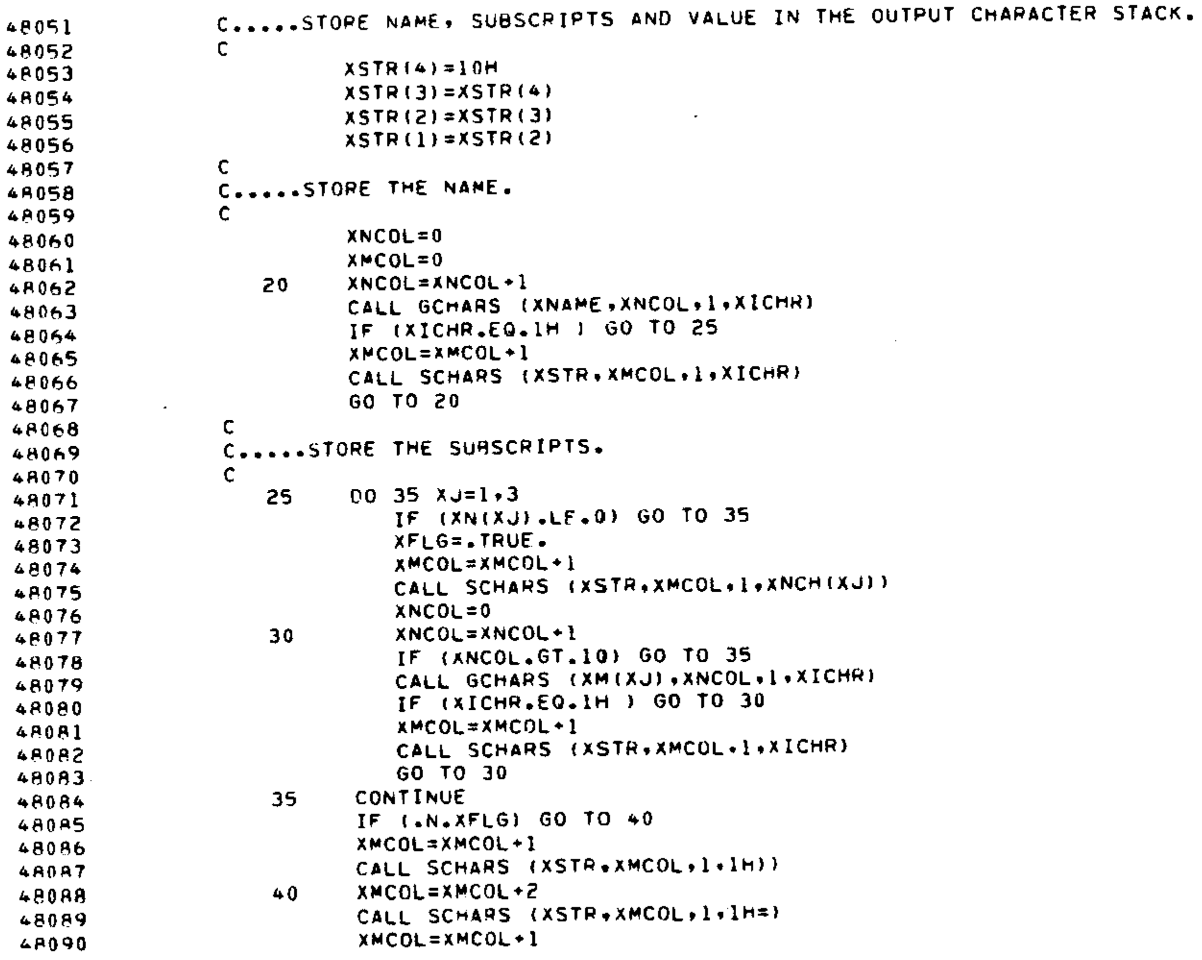
3.5. *Printing of Variables*

```
48000          SUBROUTINE XPRNT
48001          COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
48002          COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
48003          COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
48004          COMMON /XXVR2FR/ XVT2(1)
48005          COMMON XADRS(1)
48006          EQUIVALENCE (TIME,XADRS(2)), (DTPR,XADRS(6))
48007          DIMENSION XN(3), XM(3), XNUM(2), XSTR(4), XNCH(3)
48008          INTEGER XNV,XNW,XVT1,XVT2,XNPR,XPRT,XUO,XUI
48009          INTEGER XI1,XMD,XI2,XNAME,XJ,XNCOL,XMCOL,XICHR,XI
48010          INTEGER XN,XM,XNUM,XSTR,XNCH
48011          LOGICAL XFLG,XFRST
48012    C
48013    C.....THIS ROUTINE WILL GENERATE PRINTED OUTPUT FOR TIME AND EACH OF THE
48014    C.....VARIABLES IN THE PRINT STACK.
48015    C
48016    C.....INITIALIZE LOCAL VARIABLES.
48017    C
48018          DATA XNCH/1H(,1H,,1H,/
48019          DATA XFRST/.TRUE./
48020          IF (XNPR.LE.0) RETURN
48021          IF (DTPR.GT.0.) CALL EVENT (5HXPRNT,TIME+DTPR,200)
48022          IF (XFRST) WRITE (XUO,60)
48023          XFRST=.FALSE.
48024          WRITE (XUO,65) TIME
48025          CALL XPRB2F (XSTR,XDUM)
```

Line Number	Explanation
	XPRNT is a system event that is initially scheduled at TIME=TSTRT, if PRINT. requests were encountered in the data stream. The values of all variables on PRINT. cards are printed by XPRNT. Values are printed at TIME=TSTRT, TIME=TSTRT+DTPR, TIME=TSTRT+2*DTPR, etc. (Each occurrence of XPRNT schedules another occurrence at TIME=TIME+DTPR.)
48018-48019	XNCH contains delimiters of variable subscripts. XFRST allows the printing of an output label the first time XPRNT occurs.
48021-48025	Schedule XPRNT to next occur at TIME+DTPR. The simulation time of occurrence is printed, labeling the TIME at which values are printed. XPRB2F is an entry point in XPRBUF (utility routine) that initializes the buffer used to hold a string of variables and values to be printed.

```
48026              DO 55 XI=1.XNPR
48027                 XFLG=.FALSE.
48028        C
48029        C.....RETRIEVE INFORMATION FROM PRINT STACK.
48030        C
48031                 XI1=SHIFT(77740000000000000000B.A.XPRT(XI),-50)
48032                 XMD=SHIFT(30000000000000000B.A.XPRT(XI),-48)
48033                 XN(1)=SHIFT(7774000000000000B.A.XPRT(XI),-38)
48034                 XN(2)=SHIFT(3776000000000B.A.XPRT(XI),-28)
48035                 XN(3)=SHIFT(1777000000B.A.XPRT(XI),-18)
48036                 XI2=777777B.A.XPRT(XI)
48037        C
48038        C.....RETRIEVE VARIABLE NAME.
48039        C
48040                 XNAME=(77777777770000000000B.A.XVT1(XI1)).O.5555555555B
48041        C
48042        C.....GENERATE SUBSCRIPT CHARACTERS.
48043        C
48044                 DO 15 XJ=1.3
48045           15    ENCODE (10.70.XM(XJ) )XN(XJ)
48046        C
48047        C.....RETRIEVE VALUE FOR VARIABLE AND GENERATE HOLLARITH STRING.
48048        C
48049                 CALL XHOL1V (XADPS(XI2).XNUM)
48050        C
```

Line Number	Explanation
48026-48027	The entire list of variables to be printed is processed singularly. The name, subscripts, and current value of the variable are recovered. XNPR is the number of variables to be printed. XPRT is the print stack containing the variables to be printed. XFLG is a logical flag, set to .TRUE. if the recovered variable is subscripted. (Refer to Section 2.3 for description of XPRT.)
48031-48036	XI1 is the location (index) of the variable in the variable tables (XVT1 and XVT2). XMD is the mode of the variable. XN(I) is the Ith subscript of the variable. XI2 is the location (index) of the value of the variable relative to the beginning of blank common (where the values of variables are stored, see Section 2.8).
48040-48049	XNAME is the variable name, retrieved from the variable tables. XM(I) is the BCD representation of the Ith subscript of variable stored in XN(I) (an integer value). XHOL1V is a utility routine that converts a value to its BCD representation. XADRS(XI2) contains the value of the variable. XNUM is the BCD representation of value.



```
48051         C.....STORE NAME, SUBSCRIPTS AND VALUE IN THE OUTPUT CHARACTER STACK.
48052         C
48053                 XSTR(4)=10H
48054                 XSTR(3)=XSTR(4)
48055                 XSTR(2)=XSTR(3)
48056                 XSTR(1)=XSTR(2)
48057         C
48058         C.....STORE THE NAME.
48059         C
48060                 XNCOL=0
48061                 XMCOL=0
48062            20   XNCOL=XNCOL+1
48063                 CALL GCHARS (XNAME,XNCOL,1,XICHR)
48064                 IF (XICHR.EQ.1H ) GO TO 25
48065                 XMCOL=XMCOL+1
48066                 CALL SCHARS (XSTR,XMCOL,1,XICHR)
48067                 GO TO 20
48068         C
48069         C.....STORE THE SUBSCRIPTS.
48070         C
48071            25   DO 35 XJ=1,3
48072                    IF (XN(XJ).LF.0) GO TO 35
48073                    XFLG=.TRUE.
48074                    XMCOL=XMCOL+1
48075                    CALL SCHARS (XSTR,XMCOL,1,XNCH(XJ))
48076                    XNCOL=0
48077            30        XNCOL=XNCOL+1
48078                    IF (XNCOL.GT.10) GO TO 35
48079                    CALL GCHARS (XM(XJ),XNCOL,1,XICHR)
48080                    IF (XICHR.EQ.1H ) GO TO 30
48081                    XMCOL=XMCOL+1
48082                    CALL SCHARS (XSTR,XMCOL,1,XICHR)
48083                    GO TO 30
48084            35    CONTINUE
48085                  IF (.N.XFLG) GO TO 40
48086                  XMCOL=XMCOL+1
48087                  CALL SCHARS (XSTR,XMCOL,1,1H))
48088            40    XMCOL=XMCOL+2
48089                  CALL SCHARS (XSTR,XMCOL,1,1H=)
48090                  XMCOL=XMCOL+1
```

Line Number	Explanation
48053-48056	XSTR is a buffer that is filled with the variable name, subscripts, and value of the variable.
48060-48067	Pack the variable name in XSTR. Each character is selected from XNAME and filled into XSTR. XNCØL is a count of the characters in XNAME. XMCØL is the number of characters filled in XSTR.
48071-48084	The subscripts (BCD) of the variable are filled into XSTR following the name. Appropriate delimiters (parens and commas) are filled into XSTR between the subscripts. XFLG=.TRUE. if the variable has no subscripts. Following the last subscript, a right paren followed by an equal sign are stored in XSTR.

```
48091    C
48092    C.....STORE THE VALUE.
48093    C
48094          XNCOL=0
48095       45 XNCOL=XNCOL+1
48096          IF (XNCOL.GT.15) GO TO 50
48097          CALL GCHARS (XNUM,XNCOL,1,XICHR)
48098          IF (XNCOL.GT.11.A.XICHR.EQ.1H ) GO TO 50
48099          XMCOL=XMCOL+1
48100          CALL SCHARS (XSTR,XMCOL,1,XICHR)
48101          GO TO 45
48102    C
48103    C.....OUTPUT THE VARIABLE NAME, VALUE STRING.
48104    C
48105       50 CALL XPRBUF (XSTR,XMCOL)
48106       55 CONTINUE
48107          CALL XPRB1F (XSTR,XDUM)
48108          RETURN
48109    C
48110       60 FORMAT ( 19H1SIMULATION RESULTS,/)
48111       65 FORMAT (1H0,  7HTIME = ,G15.9)
48112       70 FORMAT (I10)
48113    C
48114          END
```

Line Number	Explanation
48094-48101	Following the equal sign, the value (BCD) for the variable is placed into XSTR.
48105-48108	XPRBUF is a utility routine which prints the contents of XSTR onto output. Proceed, then to reinitialize XSTR and begin filling it with the information of the next variable. After all variables have been processed, XPRB1F (an entry point in XPRBUF) flushes any remaining information in XPRBUF to output.

```
49000          SUBROUTINE XPRBUF (STR,KNT)
49001          DIMENSION STR(4), LINE(14)
49002          COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
49003          INTEGER XUO
49004    C
49005    C.....THIS ROUTINE BUFFERS THE OUTPUT STRINGS INTO OUTPUT LINES AND
49006    C.....OUTPUTS THE GENERATED LINES.
49007    C
49008          NCOL=0
49009          NLINE=NLINE+1
49010          IF (NLINE.LE.4) GO TO 25
49011          NLINE=1
49012       15 WRITE (XUO,60) LINE
49013          DO 20 I=1,14
49014       20 LINE(I)=10H
49015       25 IF (KNT.LE.35) GO TO 30
49016          IF (NLINE.EQ.4) GO TO 15
49017       30 ICOL=(NLINE-1)*29+9
49018       35 NCOL=NCOL+1
49019          IF (NCOL.GT.KNT) GO TO 40
49020          CALL GCHARS (STR,NCOL,1,ICHR)
49021          ICOL=ICOL+1
49022          CALL SCHARS (LINE,ICOL,1,ICHR)
49023          GO TO 35
```

```
49024             40 IF (NLINE.EQ.4) GO TO 45
49025                IF (ICOL.GE.NLINE*29+9) NLINE=NLINE+1
49026                RETURN
49027          C
49028          C.....FLUSH BUFFER.
49029          C
49030                ENTRY XPRB1F
49031                IF (NLINE.EQ.0) RETURN
49032             45 WRITE (XUO,60) LINE
49033                NLINE=0
49034                DO 50 I=1,14
49035             50 LINE(I)=10H
49036                RETURN
49037          C
49038          C.....BUFFER INITIALIZATION.
49039          C
49040                ENTRY XPRB2F
49041                NLINE=0
49042                DO 55 I=1,14
49043             55 LINE(I)=10H
49044                RETURN
49045          C
49046             60 FORMAT (13A10,A2)
49047          C
49048                END
```

Line Number — Explanation

XPRBUF flushes a string of characters (STR) into an output line buffer (LINE). When the line buffer is filled, it is flushed onto the output file as one printed line. KNT is the number of characters in the input string (STR).

49008-49014 — Generally, four input strings of characters fill the line buffer. When a new string of characters enter, the line buffer, if full, is flushed to output to accommodate the new string.
NLINE is the number of strings in the line buffer (LINE).

49015-49016 — If the input string has a large number of characters and LINE is partially filled with three previous strings, flush LINE.

49017-49023 — The line buffer is filled from STR a character at a time.
ICØL is the current position in LINE where the NCØLth character of STR is placed.

49024-49026 — If the line buffer is now full, it is flushed to output. If the input string consisted of an abnormally large number of characters, it fills two string locations in LINE.

49030-49036 — XPRB1F is an entry point which causes the line buffer to be flushed to output, if it contains any character strings. The string counter (NLINE) and the buffer are then reinitialized.

Line Number	Explanation
49040-49044	XPRB2F initialized the line buffer, readying it to accept strings of characters.

3.6. *Plotting of Variables*

```
50000              SUBROUTINE XPLOT
50001              COMMON /XXPLOT/XPLT(100),XRNG(2)
50002              COMMON XADRS(1)
50003              COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
50004              EQUIVALENCE (TIME,XADRS(2)), (DTPL,XADRS(7))
50005              INTEGER XNPL,XPLT,XI,XMD,XINDX,XVAL(100),XUP
50006        C
50007        C.....THIS ROUTINE GENERATES A RECORD OF THE PLOTTING VALUES FOR EACH OF
50008        C.....THE VARIABLES ENTERED IN THE PLOT VARIABLE STACK AT EACH CALL.
50009        C
50010              IF (XNPL.LE.0) RETURN
50011              IF (DTPL.GT.0.) CALL EVENT (5HXPLOT,TIME+DTPL,200)
50012              DO 15 XI=1,XNPL
50013                 CALL GBYTE (XPLT(XI),XMD,40,2)
50014                 CALL GBYTE (XPLT(XI),XINDX,42,18)
50015                 IF (XMD.EQ.0) XVAL(XI)=FLOAT(XADRS(XINDX)).A.777777777777777777
50016            1    77B
50017                 IF (XMD.EQ.1) XVAL(XI)=XADRS(XINDX).A.77777777777777777777B
50018           15 CONTINUE
50019              WRITE (XUP) (XVAL(XI),XI=1,XNPL)
50020              RETURN
50021        C
50022              END
```

Line Number	Explanation
	XPLØT is a system event that is initiated (first scheduled) by the appearance of PLØT. requests in the user data stream. The routine stores the values of all variables in the plot stack (present on PLØT. cards) at each DTPL time increment. The occurrence of XPLØT stores the values of plot variables at TIME= TSTRT and rescheduled itself to again occur at TIME= TIME+DTPL.
50010-50011	XNPL is the number of variables in the plot stack. EVENT schedules XPLØT to again occur at TIME+DTPL. TIME is the current simulation "clock" time.
50012-50018	The value of each variable in the plot stack is retrieved and printed onto file 3 (XUP). XMD is the mode of the variable. XINDX is the location (index) of the value of the variable, relative to the beginning of blank common (XADRS(1)). XVAL(XI) contains the current value of the XIth variable in the plot stack.

3.7. *Printing of Flow Values*

```
51000              SUBROUTINE XFLOP
51001              COMMON /XXFL1WS/ XNFLW,XFLWT(1)
51002              COMMON /XXFL2WS/ XFLW(1)
51003              COMMON XADRS(1)
51004              EQUIVALENCE (TIME,XADRS(2)), (DT,XADRS(5)), (DTFL,XADRS(8))
51005              COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
51006              INTEGER XNFLW,XFLWT,XUO,XSTR(4),XI,XCHEK,XM1,XM2,XVAL(2),XICOL
51007              INTEGER XJCOL,XICHR
51008        C
51009        C.....THIS SCHEDULABLE ROUTINE PRINTS THE VALUES OF THE FLOWS THAT WERE
51010        C.....REQUESTED FOR PRINTING BY FLOW PRINT REQUESTS.
51011        C
51012              IF (DTFL.GT.0.) CALL EVENT (5HXFLOP,TIME+DTFL,400)
51013              XTDT=TIME+DT
51014              WRITE(6,30) TIME,XTDT
51015              CALL XPRB2F (XSTR,XDUM)
51016              DO 25 XI=1,XNFLW
51017                 CALL GBYTE (XFLWT(XI),XCHEK,0,1)
51018                 IF (XCHEK.NE.1) GO TO 25
51019                 CALL GBYTE (XFLWT(XI),XM1,30,15)
51020                 CALL GBYTE (XFLWT(XI),XM2,45,15)
51021                 XSTR(4)=10H
51022                 XSTR(3)=XSTR(4)
51023                 ENCODE (20,35,XSTR(1) )XM1,XM2
51024                 CALL XHOL1V (XFLW(XI),XVAL)
51025                 XICOL=0
51026                 XJCOL=16
51027          15     XICOL=XICOL+1
51028                 IF (XICOL.GT.15) GO TO 20
51029                 CALL GCHARS (XVAL,XICOL,1,XICHR)
51030                 IF (XICOL.GT.11.A.XICHR.EQ.1H ) GO TO 20
51031                 XJCOL=XJCOL+1
51032              IF(XICHR.EQ.1HE)XJCOL=XJCOL-4
51033                 CALL SCHARS (XSTR,XJCOL,1,XICHR)
51034                 GO TO 15
51035          20     CALL XPRBUF (XSTR,XJCOL)
51036          25 CONTINUE
51037             CALL XPRB1F (XSTR,XDUM)
51038             RETURN
51039        C
51040          30 FORMAT( 25H0VALUES OF FLOWS, TIME = ,G15.9,4H TO ,G15.9)
51041          35 FORMAT (5HFLOW(,I3,1H,,I3,8H) =     )
51042        C
51043             END
```

Line Number	Explanation
51000-51012	XFLØP is initiated by the appearance of FLØW. requests in the data stream. This system routine prints the values of all flow commands appearing on the FLØW. data cards. The occurrence of XFLØP causes the printing of the values at the current time (TIME) and reschedules XFLØP to next occur at time = TIME+DTFL.
51015	XPRB2F is an entry point in XPRBUF (utility routine) that initializes the buffer used to print four variables (and their values) per line.

Line Number	Explanation
51016-51020	The entire flow stack is searched (XNFLW entries in XFLWT(I)) and any entries that have the high-order bit set are processed and printed (Section 2.5). XCHEK retrieves the high-order bit. XMI and XMI are the two state variables (indices) defined by each expanded flow.
51021-51026	XSTR contains a string of characters to be printed. (XSTR contains the encoded flow indices followed by the value of the flow at the current time.) XFLW(I) contains the current value of the flow defined in XFLWT(I). XHØL1V is a utility program that converts the value of the flow into its BCD representation. XVAL is the BCD representation of the value. XICØL is a count of the characters in XVAL. XJCØL is the number of characters in XSTR.
51027-51034	The characters of the value are filled into XSTR, character by character. Floating point numbers (E format) are truncated. The four least significant digits are cut off, producing an 11-digit code.
51035-51043	Four flows and their values are printed per line via utility routine XPRBUF. XPRB1F flushes any remaining information in XPRBUF to the print file.

```
         SUBROUTINE XHOLIV (VAL,NUM)
         DIMENSION NUM(2)
         CALL GBYTE (VAL,IBITS,0,12)
         IF (IBITS.EQ.0.O.IBITS.EQ.7777B) GO TO 15
         IF (IBITS.EQ.1777B) GO TO 20
         IF (IBITS.EQ.3777B) GO TO 25
         IF (IBITS.EQ.4000B) GO TO 30
         ENCODE (20,35,NUM) VAL
         RETURN
      15 ENCODE (20,40,NUM) VAL
         RETURN
      20 NUM(1)=10HINDEFINITE
         NUM(2)=10H
         RETURN
      25 NUM(1)=10H+ INFINITE
         NUM(2)=10H
         RETURN
      30 NUM(1)=10H- INFINITE
         NUM(2)=10H
         RETURN
C
      35 FORMAT (G15.9,5X)
      40 FORMAT (I11,9X)
C
         END
```

Line Number	Explanation
	XHOL1V converts a value (VAL) into a Binary Coded Decimal (BCD) equivalent form (NUM).
52002-52006	The high-order bits (the exponent of the input value) are examined to determine into which BCD form the value will be converted.
52007-52008	The input value is a floating point number. NUM is the E or F formatted (depending on the magnitude of the value) BCD representation of the value.
52009-52010	The input value is a fixed point number. NUM is the I format representation of the value.
52011-52019	The value is infinite or indefinite. NUM is a hollerith word which describes the value.

3.8. *Debugging Overlay*

```
      OVERLAY(NEWT1,1,1)
      PROGRAM XDMPXJ
      COMMON /XRVC/ IRF,IMP,ICORE,IXJ(17),IEH,IRA
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
      COMMON /XRDP/ IFILE,LINE(14),IEOF,IFET
      DIMENSION IBUF(12,8), MES(7,3), IEMT(3), ITYP(7)
      INTEGER CONTENT
      LOGICAL TWO,XILGV
C
C.....THIS IS THE EXCHANGE JUMP PACKAGE INTERPRETATION AND VARIABLE DUMP
C.....OVERLAY.  THIS OVERLAY IS CALLED IF AN ARITHMETIC MODE ERROR OCCUR
C.....AND LOADER MAPS AND/OR VARIABLE CROSS-REFERENCE MAPS HAVE BEEN
C.....DETERMINED ACCESSIBLE BY ROUTINE "XIFDBG" IN THE LEVEL 0.0 OVERLAY
C.....PRIOR TO SIMULATION EXECUTION.
C
      DATA MES/10HATTEMPTED ,10HTO REFEREN,10HCE CENTRAL,10H MEMORY OU,1
     10HTSIDE ESTA,10HBLISHED LI,10HMITS      ,10HFLOATING P,10HOINT ARI
     2TH,10HMETIC UNIT,10H RECIEVED ,10HAN INFINIT,10HE OPERAND ,10H
     3      ,10HFLOATING P,10HOINT ARITH,10HMETIC UNIT,10H RECIEVED ,10H
     4AN INDEFIN,10HITE OPERAN,10HD         /
      DATA ITYP/00141707110301145555B,00111624050705225555B,002205011455
     1555555B,00031715201405305555B,00041725021405555555B,000503235555
     25555555B,00161755243120055555B/
      DATA L16/177777B/
      DATA L17/377777B/
      DATA I42/77777777777777000000B/
C
C.....RETRIEVE INFORMATION FROM VARIABLE CROSS-REFERENCE MAPS AND LOADER
C.....MAP.
C
      NVAR=0
      IF (IRF.NE.0) CALL XREFMP
      CALL XLODMP
C
C.....COMPUTE ABSOLUTE ADDRESS OF VARIABLES.
C
      CALL XABSLC
C
C.....EXTRACT AND OUTPUT INFORMATION IN EXCHANGE JUMP PACKAGE.
C....."IP" IS THE ABSOLUTE ADDRESS OF THE LOCATION WHERE THE ERROR WAS
C.....DETECTED.
C
      IP=SHIFT(IRA,-30).A.777777B
C
C....."IEM" IS THE EXIT MODE, "IEMT(I),I=1,2,3" ARE THE DECOMPOSED EXIT
C.....MODES IF MORE THAN ONE ERROR WAS DETECTED.
C
      IEM=SHIFT(IRA,-48)
      DO 15 I=1,3
         J=I-1
   15 IEMT(I)=SHIFT(IEM,-J).A.1
      WHEN=DATE(TODAY)
      CTIME=TIME(CURRENT)
      WRITE (6,160) WHEN,CTIME,IEM
C
C.....OUTPUT EXIT MODE(S) AND THE EXPLANATION.
C
      TWO=.FALSE.
      DO 20 I=1,3
         IF (IEMT(I).EQ.0) GO TO 20
         WRITE (6,165) (MES(J,I),J=1,7)
         IF (TWO) WRITE (6,170)
         TWO=.TRUE.
   20 CONTINUE
      WRITE (6,175) IP
C
C.....DETERMINE THE ROUTINE IN WHICH THE ERROR OCCURED.
C
      NAME=0
      IF (NBLK.LE.1) GO TO 35
      NB1=NBLK-1
      DO 25 I=1,NB1
         J=I+1
         CALL XRLXRM (I,IBN,ICF,ILB)
         JLB=IBLK(J).A.L17
         IF (IP.LT.ILB.O.IP.GE.JLB) GO TO 25
         NAME=IBN
         LOC=IP-ILB
         GO TO 30
   25 CONTINUE
   30 IF (NAME.EQ.0) GO TO 35
      WRITE (6,180) LOC,NAME
   35 CONTINUE
      IF (IP.EQ.400000B) WRITE (6,225)
      IF (IEMT(2).NE.0.O.IEMT(3).NE.0) WRITE (6,185)
C
C.....INTERPRET AND OUTPUT THE EXCHANGE JUMP PACKAGE.
C
      WRITE (6,190)
      DO 85 I=1,8
C
C.....RETRIEVE THE CONTENTS OF THE I-TH ADDRESS (A0 THROUGH A7) REGISTER
C.....FROM THE EXCHANGE JUMP PACKAGE.
C
         LOC=SHIFT(IXJ(I),-18).A.777777B
         IBUF(1,I)=LOC
         DO 40 J=2,12
   40    IBUF(J,I)=1H
         IF (NVAR.LE.1) GO TO 55
C
C.....DETERMINE IF THE CURRENT ADDRESS REGISTER CORRESPONDS TO A USER
C.....VARIABLE.
C
         NV1=NVAR-1
         DO 45 J=1,NV1
            CALL XRVXRT (J,INM,ITP,ILR,IBL,IAY,ILA)
            IF (INM.EQ.61142701625555000000B) GO TO 45
            ILA1=IVAR(2,J+1).A.L16
            IF (LOC.LT.ILA.O.LOC.GE.ILA1) GO TO 45
            GO TO 50
   45    CONTINUE
         GO TO 55
C
C.....THE ADDRESS OF A USER VARIABLE WAS CONTAINED IN THE CURRENT ADDRES
C.....REGISTER.
C
   50    IF (IAY.EQ.1) GO TO 80
         IF (LOC.NE.ILA) GO TO 55
C
C.....THE VARIABLE IS NOT AN ARRAY, PACK THE NAME, TYPE, RELATIVE LOCA-
C.....TION, RELOCATION POINT, OCTAL VALUE, AND DECODED VALUE OF THE
C.....VARIABLE IN THE OUTPUT BUFFER.
C
         IBUF(2,I)=INM
         IBUF(3,I)=ITYP(ITP)
         ENCODE (7,240,IBUF(5,I) )ILR
         IBUF(6,I)=IBL
         IVAL=CONTENT(LOC)
         CALL OPCOC (IBUF(7,I),IVAL)
         CALL OPCDE (IBUF(10,I),LOC,ITP)
         GO TO 85
   55    IF (NBLK.LE.1) GO TO 65
C
C.....THE CURRENT ADDRESS REGISTER DOES NOT CORRESPOND TO A USER VARIABL
C.....DETERMINE THE ROUTINE NAME WHICH CONTAINS THIS ADDRESS.
C
         NB1=NBLK-1
         DO 60 K=1,NB1
            ILB=IBLK(K).A.L17
            JLB=IBLK(K+1).A.L17
            IF (LOC.GE.ILB.A.LOC.LT.JLB) GO TO 75
   60    CONTINUE
   65    CONTINUE
         ILB=IBLK(NBLK).A.L17
         IF (LOC.LT.0.O.LOC.GE.ILB) GO TO 70
```

```
            IVAL=CONTENT(LOC)
            CALL DPCOC (IBUF(7,I),IVAL)
            GO TO 85
C
C.....THE ADDRESS IS OUT OF RANGE OF THE USERS FIELD LENGTH.
C
   70       IBUF(7,I)=9H      OUT
            IBUF(8,I)=8HOF RANGE
            GO TO 85
C
C.....PACK THE ROUTINE RELOCATION INFORMATION IN THE OUTPUT BUFFER.
C
   75       ILOC=LOC-ILB
            ENCODE (7,240,IBUF(5,I) )ILOC
            IBUF(6,I)=IBLK(K).A.I42
            IVAL=CONTENT(LOC)
            CALL DPCOC (IBUF(7,I),IVAL)
            CALL DPCDV (IBUF(10,I),IVAL)
            GO TO 85
C
C.....THE VARIABLE IS AN ARRAY, PACK THE INFORMATION ABOUT THE VARIABLE
C.....AND ITS CONTENTS IN THE OUTPUT BUFFER.
C
   80       IBUF(2,I)=INM
            IBUF(3,I)=ITYP(ITP).O.47000000000000000000B
            INDX=LOC-ILA+1
            ENCODE (5,235,IBUF(4,I) )INDX
            ENCODE (7,240,IBUF(5,I) )ILR
            IBUF(6,I)=IBL
            IVAL=CONTENT(LOC)
            CALL DPCOC (IBUF(7,I),IVAL)
            CALL DPCDE (IBUF(10,I),LOC,ITP)
   85 CONTINUE
C
C.....OUTPUT THE BUFFER OF CODED ADDRESS REGISTER INFORMATION.
C
      DO 90 I=1,8
         J=I-1
   90 WRITE (6,195) J,(IBUF(K,I),K=1,12)
C
C.....EXTRACT AND PACK THE CONTENTS OF THE OPERAND (X0 THROUGH X7) AND
C.....INCREMENT (B0 THROUGH B7) REGISTERS IN THE OUTPUT BUFFER.
C
      DO 95 I=1,8
         CALL DPCOC (IBUF(1,I),IXJ(I+8))
         CALL DPCDV (IBUF(4,I),IXJ(I+8))
         IBUF(7,I)=IXJ(I).A.777777B
         IF (I.EQ.1) IBUF(7,I)=0
   95 CONTINUE
C
C.....OUTPUT THE PACKED BUFFER.
C
      WRITE (6,200)
      DO 100 I=1,8
         J=I-1
  100 WRITE (6,205) J,(IBUF(K,I),K=1,5),J,IBUF(7,I)
C
C.....DUMP ALL VARIABLES BY RELOCATION BLOCK.
C
      IF (NVAR.LE.1) GO TO 155
      IF (NBLK.LE.1) GO TO 155
      NV1=NVAR-1
      NB1=NBLK-1
      LKNT=66
      LIBN=0
C
C.....ITERATE THROUGH EACH OF THE RELOCATION BLOCKS.
C
      DO 150 I=1,NB1
         CALL XRLXRM (I,IBN,ICF,IBP)
C
C.....ITERATE THROUGH EACH OF THE VARIABLES.
C
      DO 150 J=1,NV1
         CALL XRVXRT (J,INM,ITP,ILR,IBL,IAY,ILA)
         IF (INM.EQ.61142701625555000000B) GO TO 150
         IF (IBN.NE.IBL) GO TO 150
```

```
C
C.....WRITE NEW RELOCATION NAME IF CURRENT VARIABLE IS START OF NEW BLOC
C
      IF (LIBN.EQ.IBN) GO TO 110
      LKNT=LKNT+5
      LIBN=IBN
      IF (LKNT.LE.58) GO TO 105
      WRITE (6,210) IBN
      LKNT=7
      GO TO 110
  105 WRITE (6,230) IBN
  110 CONTINUE
      L1=0
      L2=0
      L=1
      NREP=1
C
C.....CLEAR THE OUTPUT BUFFER.
C
      DO 115 K=1,10
  115 IBUF(K,1)=1H
C
C.....PACK INFORMATION IN THE OUTPUT BUFFER.
C
      IBUF(1,1)=INM
      IBUF(2,1)=ITYP(ITP)
      IF (IAY.EQ.1) IBUF(2,1)=IBUF(2,1).O.47000000000000000000B
      ENCODE (7,240,IBUF(3,1) )ILA
      ENCODE (7,240,IBUF(4,1) )ILR
      IVAL=CONTENT(ILA)
      IF (XILGV(IVAL)) IVAL=IVAL.A.77770000000000000000B
      LVAL=IVAL
      CALL DPCOC (IBUF(5,1),IVAL)
      CALL DPCDE (IBUF(8,1),ILA,ITP)
      IF (IAY.NE.1) GO TO 135
C
C.....CURRENT VARIABLE IS AN ARRAY.
C
      IF (ITP.EQ.4.O.ITP.EQ.5) L=2
      L1=ILA
      L2=IVAR(2,J+1).A.L16
      GO TO 135
C
C.....ITERATE THROUGH THE ARRAY.
C
  120 L1=L1+L
      IF (L1.GE.L2) GO TO 145
C
C.....CLEAR BUFFER FOR ITERATED VALUES.
C
      DO 125 K=1,4
  125 IBUF(K,1)=1H
C
C.....RETRIEVE VALUES AND CHECK FOR REPEATED VALUES.
C
      IVAL=CONTENT(L1)
      IF (XILGV(IVAL)) IVAL=IVAL.A.77770000000000000000B
      IF (IVAL.NE.LVAL) GO TO 130
      NREP=NREP+1
      GO TO 120
C
C.....PRINT REPETITION FACTOR IF A REPETITION IS COMPLETED.
C
  130 IF (NREP.GT.1) WRITE (6,220) NREP
      LVAL=IVAL
      NREP=1
C
C.....PACK VALUE IN OUTPUT BUFFER.
C
      CALL DPCOC (IBUF(5,1),IVAL)
      CALL DPCDE (IBUF(8,1),L1,ITP)
C
C.....WRITE CURRENT BUFFER.
C
  135 LKNT=LKNT+1
      IF (LKNT.LE.60) GO TO 140
      WRITE (6,210) IBN
      LKNT=7
```

```
   140    WRITE (6,215) (IBUF(K,1),K=1,10)
          IF (L1.NE.0) GO TO 120
C
C.....PRINT REPETITION FACTOR IF A REPETITION IS COMPLETED.
C
  145    IF (NREP.GT.1) WRITE (6,220) NREP
  150 CONTINUE
C
C.....DUMP INTERPRETATION COMPLETE, FLUSH OUTPUT BUFFER.
C
  155 IFILE=17252420252400000000B
      CALL XFLUSH
C
  160 FORMAT ( 22H1ARITHMETIC MODE ERROR,18X, 15HDIAGNOSTIC DUMP,25X,A10
     1.10X,A10///2X, 14HTYPE OF ERROR:,/10X, 13HERROR MODE = ,I1/)
  165 FORMAT (18X,7A10)
  170 FORMAT (1H+,13X,  3HAND)
  175 FORMAT (1H0,9X, 36HOCCURING (APPROXIMATELY) AT ADDRESS ,O6,  1HB)
  180 FORMAT (1H+,T55, 18HWHICH IS LOCATION ,O6, 13HB IN ROUTINE ,A10)
  185 FORMAT (1H0/9X, 83HNON-STANDARD FLOATING POINT ARITHMETIC - TABLES
     1 OF NON-STANDARD RESULTS BY DIVISION,///28X, 12HDIVIDE (A/B),20X,
     2 5HWHERE,//36X,  1HB,27X, 21H+0    =    0000 X...X B,/26X,  2H+N,4X,
     3  2H-N,4X,  2H+0,4X,  2H-0,18X, 21H-0    =    7777 X...X B,/62X, 37H

     4+INF    =    3777 X...X B   (+ INFINITY),/20X, 27H+N      --     --     +I
     5NF  -INF,15X, 37H-INF    =    4000 X...X B  (- INFINITY),/16X, 31HA
     6  -N      --      --     -INF   +INF,15X, 39H+IND    =    1777 X...X B   (+
     7INDEFINITE),/20X, 27H+0       0       0     +IND   +IND,15X, 39H-IND    =
     8  6000 X...X B   (- INDEFINITE),/20X, 27H-0       0       0     +IND   +IN
     9D,18X, 57HN    =    ANY WORD EXCEPT +INF, -INF, +IND, -IND, +0, OR -
     *0)
  190 FORMAT (1H0/2X, 22HEXCHANGE JUMP PACKAGE:,//6X,  7HADDRESS,14X, 10
     1HREFERENCED,5X,  4HTYPE,4X,  5HINDEX,3X,  5HLOCAL,3X,  9HCONTAINED
     2,/5X,  9HREGISTERS,2X,  8HCONTENTS,2X, 13HVARIABLE NAME,2X,1H*,  6
     3H=ARRAY,2X,  5H(DEC),2X,  7HADDRESS,5X,  2HIN,12X, 13HVALUE (OCTAL
     4),15X, 13HDECODED VALUE,/)
  195 FORMAT (8X,  1HA,I1,7X,O6,  1HB,5X,A7,4X,A8,2X,A5,2X,A7,2X,A9,2X,2
     1A10,A5,2X,2A10,A8)
  200 FORMAT (1H0/8X,  7HOPERAND,81X,  9HINCREMENT,/7X,  9HREGISTERS,12X
     1,  8HCONTENTS,16X, 13HDECODED VALUE,31X,  9HREGISTERS,4X, 18HCONTE
     2NTS           ,/)
  205 FORMAT (10X,  1HX,I1,8X,2A10,A5,4X,2A10,30X,  1HB,I1,9X,O6,  1HB)
  210 FORMAT ( 19H1VARIABLE DUMP   -   ,A9//5X,  8HVARIABLE,7X,  4HTYPE,20
     1X,  5HLOCAL,/7X,  4HNAME,7X,1H*,  6H=ARRAY,5X,  8HLOCATION,5X,  7H
     2ADDRESS,4X,  8HREPEATED,11X, 13HVALUE (OCTAL),17X, 13HDECODED VALU
     3E,/)
  215 FORMAT (6X,A7,5X,A8,5X,A7,5X,A7,17X,2A10,A5,5X,2A10,A8)
  220 FORMAT (1H+,T55,I5,2H _)
  225 FORMAT (1H0,17X, 71HCHECK FOR A REFERENCE TO AN NON-EXISTANT ROUTI
     1NE (UNSATISFIED EXTERNAL))
  230 FORMAT (1H ,//, 19H VARIABLE DUMP   -   ,A9,//)
  235 FORMAT (I5)
  240 FORMAT (O6,1HB)
C
      END
      SUBROUTINE XLODMP
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
      COMMON /XRDP/ IFILE,LINE(14),IEOF,IFET
C
C.....THIS ROUTINE LOCATES, READS AND INTERPRETS THE LOADER MAP FOR THE
C.....1.0 OVERLAY.  THE NAME AND ADDRESS OF EACH PROGRAM AND COMMON BLOC
C.....IS ENTERED INTO THE BLOCK REFERENCE STACK (IBLK(I),I=1,...,NBLK)
C.....IN THE FOLLOWING FORMAT:
C.....LOCATION   (BITS)           INFORMATION
C.....0-41       (42)       NAME OF PROGRAM OR COMMON BLOCK.
C        42       (1)       COMMON BLOCK FLAG (1 IF COMMON, 0 OTHERWISE).
C.....43-59      (17)        ADDRESS OF BLOCK OR PROGRAM.
C
      INTEGER XROC1
      LOGICAL DONE,FIND,MAX
      DATA NBLKMX/100/
      DATA MSK/77777777777777000000B/
      DATA ICBL/400000B/
      DATA L17/377777B/
      IFILE=17252420252400000000B
      DONE=.FALSE.
      FIND=.FALSE.
      MAX=.FALSE.
      NLINE=0
      NBLK=0
      CALL XFLUSH
      REWIND 6
      CALL XROPI
   15 CALL XREADL
      IF (IEOF) 55,20,15
   20 IF (DONE) GO TO 15
      IF (FIND) GO TO 30
C
C.....CHECK FOR THE STARTING LINE OF THE 1.0 OVERLAY.
C
      IF (LINE(1).EQ.10H1CORE MAP .A.LINE(4).EQ.10H   01.00  ) GO TO 25
      GO TO 15
   25 FIND=.TRUE.
   30 CONTINUE
      NLINE=NLINE+1
      IF (NLINE.NE.1) GO TO 35
C
C.....RETRIEVE THE LAST WORD ADDRESS OF BLANK COMMON.
C
      NAME=61142701625555400000B
      LOC=XROC1(LINE,11,3)
      NBLK=NBLK+1
      IF (NBLK.GT.NBLKMX) GO TO 80
      IBLK(NBLK)=NAME.O.LOC
      LOC=XROC1(LINE,12,3)
      IF (LOC.LE.0) GO TO 15
C
C.....RETRIEVE THE STARTING ADDRESS OF BLANK COMMON.
C
      NAME=50555055555555400000B
      NBLK=NBLK+1
      IF (NBLK.GT.NBLKMX) GO TO 80
      IBLK(NBLK)=NAME.O.LOC
      GO TO 15
   35 IF (NLINE.LE.4) GO TO 15
C
C.....CHECK FOR LAST LINE OF LOADER MAP.
C
      IF (LINE(2).EQ.10H----UNSATI.OR.LINE(2).EQ.10H--ENTRY---) GO TO 50
      IF (LINE(2).EQ.1H ) GO TO 40
C
C.....RETRIEVE ROUTINE NAME AND LOCATION.
C
      NAME=LINE(2).A.MSK
      LOC=XROC1(LINE,3,3)
      NBLK=NBLK+1
      IF (NBLK.GT.NBLKMX) GO TO 80
      IBLK(NBLK)=NAME.O.LOC
   40 IF (LINE(7).EQ.1H ) GO TO 15
C
C.....RETRIEVE COMMON BLOCK NAME AND LOCATION.
C
      LOC=XROC1(LINE,8,3)
      NAME=(LINE(7).A.MSK).O.ICBL.O.LOC
      DO 45 I=1,NBLK
         IF (NAME.EQ.IBLK(I)) GO TO 15
   45 CONTINUE
      NBLK=NBLK+1
      IF (NBLK.GT.NBLKMX) GO TO 80
      IBLK(NBLK)=NAME
      GO TO 15
   50 DONE=.TRUE.
      GO TO 15
   55 IF (.NOT.FIND) GO TO 70
      IF (.NOT.DONE) GO TO 75
      IF (MAX) GO TO 85
   60 IF (NBLK.LE.1) RETURN
```

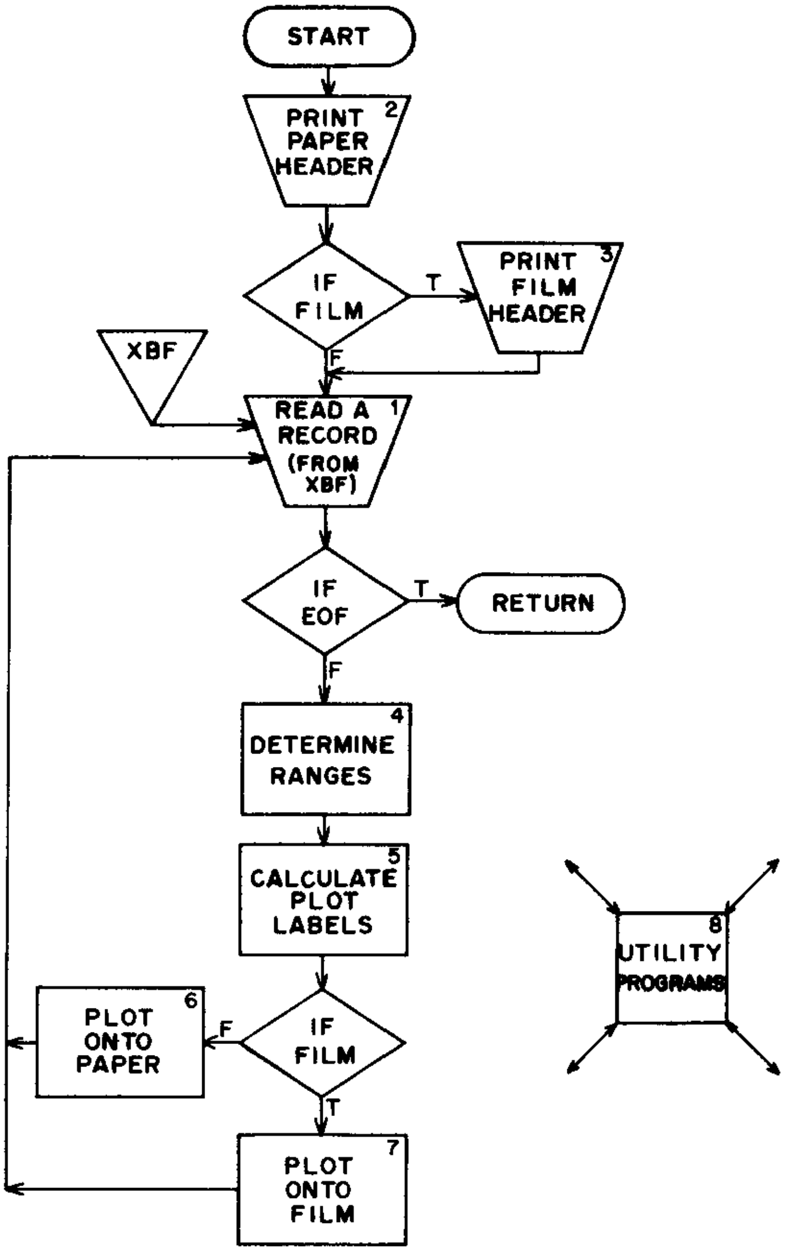
```
C
C.....SORT THE RELOCATION BLOCK STACK ACCORDING TO ABSOLUTE LOCATION.
C
      NI=NBLK-1
      DO 65 I=1,NI
         IJ=I+1
      DO 65 J=IJ,NBLK
         IBL=IBLK(I).A.L17
         JBL=IBLK(J).A.L17
         IF (IBL.LE.JBL) GO TO 65
         KEEP=IBLK(I)
         IBLK(I)=IBLK(J)
         IBLK(J)=KEEP
   65 CONTINUE
      RETURN
   70 WRITE (6,90)
      GO TO 60
   75 WRITE (6,95)
      GO TO 60
   80 MAX=.TRUE.
      DONE=.TRUE.
      NBLK=NBLK-1
      GO TO 15
   85 WRITE (6,100) NBLKMX
      GO TO 60
C
   90 FORMAT ( 20H0NO LOADER MAP FOUND)
   95 FORMAT ( 38H0TERMINAL LINE OF LOADER MAP NOT FOUND)
  100 FORMAT ( 36H0NUMBER OF BLOCK REFERENCES EXCEEDS ,I3)
C
      END
      SUBROUTINE XREFMP
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
      COMMON /XRDP/ IFILE,LINE(14),IEOF,IFET
      DIMENSION KEY1(4), KEY2(6), KEY3(6)
C
C.....THIS ROUTINE READS AND INTERPRETS THE VARIABLE CROSS-REFERENCE MAP
C.....CONTAINED ON FILE (DEBUG).  THE INFORMATION IS STORED IN TWO CONSE
C.....UTIVE WORDS OF THE VARIABLE CROSS REFERENCE STACK (IVAR(1-2,I),
C.....I=1,...,NVAR) IN THE FOLLOWING FORMAT:
C.....LOCATION   (BITS)              INFORMATION
C.....WORD 1
C       0-41      (42)     VARIABLE NAME.
C.....42-44        (3)     ARITHMETIC MODE OF VARIABLE.
C                              1 = LOGICAL
C                              2 = INTEGER
C                              3 = REAL
C                              4 = COMPLEX
C                              5 = DOUBLE PRECISION
C                              6 = EXTENDED CORE STORAGE
C                              7 = UNDETERMINED (NO TYPE)
C.....45-59       (15)     RELATIVE ADDRESS OF VARIABLE.
C.....WORD 2
C       0-41      (42)     RELOCATION (PROGRAM OR COMMON BLOCK NAME).
C        42        (1)     COMMON BLOCK FLAG (1 IF COMMON, 0 OTHERWISE).
C        43        (1)     ARRAY FLAG (1 IF ARRAY, 0 OTHERWISE).
C.....44-59       (16)     ABSOLUTE ADDRESS OF VARIABLE.
C
      INTEGER XROC1,XROC2
      LOGICAL IN,COM
      DATA NVARMX/500/
      DATA KEY1/7HPROGRAM,10HSUBROUTINE,8HFUNCTION,10HBLOCK DATA/
      DATA KEY2/9H   INLINE,10H   EXTERNA,9H   COMMON,10H   STATIST,10H
     1  STATEME,10H   FILE NA/
      DATA KEY3/7HLOGICAL,7HINTEGER,4HREAL,7HCOMPLEX,6HDOUBLE,3HECS/
      DATA MSK/77777777777777000000B/
      IFILE=04050225070000000000B
      IN=.FALSE.
      COM=.FALSE.
      NVAR=0
      REWIND 4
      CALL XROPI
   15 CALL XREADL
      IF (IEOF) 120,20,15
   20 CONTINUE
```

```
      IF (LINE(2).EQ.1H ) GO TO 15
C
C.....CHECK FOR AND RETRIEVE PROGRAM LAST WORD ADDRESS.
C
      IF (LINE(1).NE.10H     PROGR) GO TO 30
C
C.....CHECK FOR ENTRY OF A NON-USER LAST WORD ADDRESS.
C
      IF (ISUBR.EQ.6HXFLOWS) GO TO 25
      ICHK=SHIFT(ISUBR,-54)
      IF (ICHK.EQ.1RX) GO TO 15
   25 CONTINUE
      NAME=61142701625555700000B
      IREL=ISUBR.A.MSK
      LOC=XROC1(LINE,3,1)
      NAME=NAME.O.LOC
      GO TO 95
C
C.....CHECK FOR A NEW SUBPROGRAM NAME.
C
   30 IF (LINE(1).NE.1H1) GO TO 40
      IN=.FALSE.
      COM=.FALSE.
      DO 35 I=1,4
         IF (LINE(2).NE.KEY1(I)) GO TO 35
         ISUBR=SHIFT(LINE(3),12)
         IF (I.EQ.4) ISUBR=KEY1(4)
         GO TO 15
   35 CONTINUE
      GO TO 15
   40 IF (IN.O.COM) GO TO 50
C
C.....CHECK FOR A NEW VARIABLE DEFINITION TABLE.
C
      IF (LINE(1).NE.10H   VARIABL) GO TO 45
      IN=.TRUE.
      GO TO 15
C
C.....CHECK FOR A NEW COMMON BLOCK LENGTH TABLE.
C
   45 IF (LINE(1).NE.9H   COMMON) GO TO 15
      COM=.TRUE.
      GO TO 15
C
C.....CHECK FOR TERMINATION OF A VARIABLE OR COMMON TABLE.
C
   50 DO 55 I=1,6
         IF (LINE(1).NE.KEY2(I)) GO TO 55
         IN=.FALSE.
         COM=.FALSE.
         IF (I.EQ.3) COM=.TRUE.
         GO TO 15
   55 CONTINUE
      IF (IN) GO TO 60
C
C.....RETRIEVE COMMON BLOCK LAST WORD ADDRESS.
C.....CHECK FOR ENTRY OF A NON-USER COMMON BLOCK.
C
      ICHK=SHIFT(LINE(2),-54)
      IF (ICHK.EQ.1RX) GO TO 15
      NAME=61142701625555700000B
      IREL=(LINE(2).A.MSK).O.400000B
      LOC=XROC2(LINE,2,10)
      NAME=NAME.O.LOC
      GO TO 95
C
C.....RETRIEVE VARIABLE CROSS REFERENCE INFORMATION.
C
   60 DO 90 I=1,2
         J=(I-1)*5+1
         IF (LINE(J).EQ.1H ) GO TO 90
         IF (LINE(J+4).EQ.5H F.P.) GO TO 90
         IREL=ISUBR
         ICFL=0
         IF (LINE(J+4).EQ.1H ) GO TO 65
         IREL=LINE(J+4)
         ICFL=1
C
C.....CHECK FOR ENTRY OF A NON-USER VARIABLE.
C
   65    IF (IREL.EQ.6HXFLOWS) GO TO 70
         ICHK=SHIFT(IREL,-54)
         IF (ICHK.EQ.1RX) GO TO 90
C
C.....RETRIEVE REMAINING INFORMATION.
C
   70    NAME=LINE(J+1).A.MSK
         IREL=IREL.A.MSK
         ITYP=LINE(J+2)
         DO 75 K=1,6
            IF (ITYP.EQ.KEY3(K)) GO TO 80
   75    CONTINUE
         K=7
   80    ITYP=SHIFT(K,15)
         LOC=XROC1(LINE,J,3)
         ICFL=SHIFT(ICFL,17)
         IARY=0
         IF (LINE(J+3).EQ.6H ARRAY) IARY=1
         IARY=SHIFT(IARY,16)
         NAME=NAME.O.ITYP.O.LOC
         IREL=IREL.O.ICFL.O.IARY
         GO TO 95
   85    CONTINUE
   90 CONTINUE
      GO TO 15
C
C.....PLACE INFORMATION IN STACK.
C
   95 IF (NVAR.LE.0) GO TO 105
      DO 100 K=1,NVAR
         IF (NAME.EQ.IVAR(1,K).A.IREL.EQ.IVAR(2,K)) GO TO 110
  100 CONTINUE
  105 NVAR=NVAR+1
      IF (NVAR.GT.NVARMX) GO TO 115
      IVAR(1,NVAR)=NAME
      IVAR(2,NVAR)=IREL
  110 IF (IN) GO TO 85
      GO TO 15
  115 WRITE (6,125)
      NVAR=NVAR-1
  120 RETURN
C
  125 FORMAT ( 49H0VARIABLE CROSS REFERENCE TABLE OVERFLOW IN DEBUG,/5X,
     1 30HDEBUG OUTPUT MAY BE INCOMPLETE)
C
      END
      SUBROUTINE XABSLC
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
C
C.....THIS ROUTINE COMPUTES AND INSERTS THE ABSOLUTE LOCATIONS OF THE
C.....VARIABLES INTO THE RIGHTMOST 16 BITS OF WORD 2 OF EACH ENTRY IN
C.....THE VARIABLE CROSS REFERENCE STACK.  THE STACK IS THEN SORTED INTO
C.....ASCENDING ORDER ACCORDING TO ABSOLUTE LOCATION.
C
      INTEGER CONTENT
      DATA MSK/77777777777777000000B/
      IF (NBLK.LE.0.O.NVAR.LE.0) RETURN
      DO 25 I=1,NVAR
         IREL=IVAR(2,I).A.MSK
         DO 15 J=1,NBLK
            IBL=IBLK(J).A.MSK
            IF (IREL.EQ.IBL) GO TO 20
   15    CONTINUE
         IVAR(2,I)=IVAR(2,I).O.177777B
         GO TO 25
   20    LOC=(IVAR(1,I).A.77777B)+(IBLK(J).A.377777B)
         IVAR(2,I)=IVAR(2,I).O.LOC
   25 CONTINUE
      IF (NVAR.LE.1) RETURN
      NV=NVAR-1
      DO 40 I=1,NV
```

```
         IJ=I+1
      DO 40 J=IJ,NVAR
         ILOC=IVAR(2,I).A.177777B
         JLOC=IVAR(2,J).A.177777B
         IF (ILOC.LT.JLOC) GO TO 40
         IF (ILOC.NE.JLOC) GO TO 30
         INM=IVAR(1,I).A.MSK
         IF (INM.EQ.61142701625555000000B) GO TO 40
   30    CONTINUE
         DO 35 K=1,2
            KEEP=IVAR(K,I)
            IVAR(K,I)=IVAR(K,J)
   35    IVAR(K,J)=KEEP
   40 CONTINUE
      RETURN
C
      END
      SUHROUTINE XRVXRT (N,INM,ITP,ILR,IBL,IAY,ILA)
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
C
C.....THIS ROUTINE UNPACKS AND RETURNS THE INFORMATION STORED IN THE N-T
C.....ENTRY IN THE VARIAPLE CROSS-REFERENCE STACK.
C
      IVR=IVAR(1,N)
      INM=IVR.A.77777777777777000000B
      ITP=IVR.A.700000B
      ITP=SHIFT(ITP,-15)
      ILR=IVR.A.77777B
      IVR=IVAR(2,N)
      IBL=IVR.A.77777777777777000000B
      ICF=IVR.A.400000B
      ICF=SHIFT(ICF,-17)
      IF (ICF.EQ.0) GO TO 25
      JBL=50B
      KBL=IBL
      KS=54
   15 JBL=SHIFT(JBL,6)
      KBL=SHIFT(KBL,6)
      KS=KS-6
      ICH=KBL.A.77B
      IF (ICH.EQ.50B) GO TO 25
      IF (ICH.EQ.55B.O.ICH.EQ.0) GO TO 20
      JBL=JBL.O.ICH
      GO TO 15
   20 JBL=JBL.O.50B
      IBL=SHIFT(JBL,KS)
   25 IAY=IVR.A.200000B
      IAY=SHIFT(IAY,-16)
      ILA=IVR.A.177777B
      RETURN
C
      END
      SUBROUTINE XRLXRM (N,INM,ICF,IHL)
      COMMON /XLCB/ NBLK,IBLK(100),NVAR,IVAR(2,500)
C
C.....THIS ROUTINE UNPACKS AND RETURNS THE INFORMATION STORED IN THE N-T
C.....ENTRY IN THE BLOCK REFERENCE STACK.
C
      IBK=IBLK(N)
      INM=IBK.A.77777777777777000000B
      ICF=IBK.A.400000B
      ICF=SHIFT(ICF,-17)
      IBL=IBK.A.377777B
      IF (ICF.EQ.0) RETURN
      JNM=INM
      JBL=50B
      KS=54
   15 JBL=SHIFT(JBL,6)
      JNM=SHIFT(JNM,6)
      KS=KS-6
      ICH=JNM.A.77B
      IF (ICH.EQ.50B) RETURN
      IF (ICH.EQ.55B.O.ICH.EQ.0) GO TO 20
      JBL=JBL.O.ICH
      GO TO 15
   20 JBL=JBL.O.50B
      INM=SHIFT(JBL,KS)
      RETURN
C
      END
      INTEGER FUNCTION XROC1(L,N,M)
      DIMENSION L(14)
      INTEGER BASE
C
C.....THIS ROUTINE DECODES THE 6 CHARACTERS IN "LINE" STARTING AT CHAR-
C.....ACTER "M" IN WORD "N".  ENTRY "XROC1" DECODES OCTAL DPC REPRESEN-
C.....TATIONS, ENTRY "XROC2" DECODES DECIMAL DPC REPRESENTATIONS.
C
      DATA M0/77B/
      BASE=8
      GO TO 15
      ENTRY XROC2
      BASE=10
   15 CONTINUE
      IW=N
      IP=M-1
      NDIG=6
      XROC1=0
   20 NDIG=NDIG-1
      IF (NDIG.LT.0) RETURN
      IP=IP+1
      IF (IP.LE.10) GO TO 25
      IP=IP-10
      IW=IW+1
   25 KSH=(10-IP)*6
      M1=SHIFT(M0,KSH)
      ICHR=L(IW).A.M1
      ICHR=SHIFT(ICHR,-KSH).A.M0
      IF (ICHR.EQ.1R ) GO TO 20
      IDIG=ICHR-33B
      IF (IDIG.LT.0.O.IDIG.GT.BASE-1) GO TO 30
      XROC1=XROC1+IDIG*(BASE**NDIG)
      GO TO 20
   30 WRITE (6,35) L(IW)
      CALL EXIT
C
   35 FORMAT ( 62H0SYSTEM ERROR IN DEBUG, ATTEMPT TO DECODE AND ILLEGAL
     1VALUE - ,A10)
C
      END
      SUBROUTINE DPCOC (HOLLR,OCTAL)
      INTEGER OCTAL,HOLLR(1)
C
C.....THIS ROUTINE ENCODES THE VALUE "OCTAL" INTO THE ARRAY "HOLLR".  TH
C.....ENCODED INFORMATION IS THE OCTAL REPRESENTATION OF THE VALUE IN TH
C.....FORMAT:
C                XXXX XXXX XXXX XXXX XXXXB
C.....WHERE X IS A DPC OCTAL DIGIT.
C
      HOLLR(1)=0
      HOLLR(2)=0
      HOLLR(3)=0
      K=0
      J=1
      DO 20 I=1,20
         K=K+1
         N=I*3
         IO=SHIFT(OCTAL,N).A.7B
         IC=IO+33B
         IF (K.LE.10) GO TO 15
         K=1
         J=J+1
   15    HOLLR(J)=SHIFT(HOLLR(J),6)
         HOLLR(J)=HOLLR(J).O.IC
         IF (MOD(I,4).NE.0) GO TO 20
         K=K+1
         IC=55B
         IF (I.EQ.20) GO TO 20
         HOLLR(J)=SHIFT(HOLLR(J),6)
         HOLLR(J)=HOLLR(J).O.IC
   20 CONTINUE
      HOLLR(J)=SHIFT(HOLLR(J),36)
      HOLLR(J)=HOLLR(J).O.0255555555555B
      RETURN
```

```
C
      END
      SUBROUTINE DPCDE (HOLLR,LOC,TYPE)
      INTEGER HOLLR(3),TYPE,CONTENT,VAL(2)
C
C.....THIS ROUTINE ENCODES THE CONTENTS OF THE LOCATION "LOC" ACCORDING
C.....TO THE MODE "TYPE".  OUT-OF-RANGE AND INDEFINITE VALUES ARE
C.....TREATED SEPARATELY.
C
      VAL(1)=CONTENT(LOC)
      VAL(2)=CONTENT(LOC+1)
      HOLLR(1)=1H
      HOLLR(2)=1H
      HOLLR(3)=1H
      IT=TYPE
      IB=SHIFT(VAL(1),-48).A.7777B
      IF (IB.NE.3777B) GC TO 15
      HOLLP(1)=10H       +
      HOLLR(2)=8HINFINITF
      RETURN
   15 IF (IB.NE.4000B) GO TO 20
      HOLLR(1)=10H       -
      HOLLR(2)=8HINFINITE
      RETURN
   20 IF (IB.NE.1777B) GO TO 25
      HOLLR(1)=10H       + I
      HOLLR(2)=9HNDEFINITE
      RETURN
   25 IF (IB.NE.6000B) GO TO 30
      HOLLR(1)=10H       - I
      HOLLR(2)=9HNDEFINITE
      RETURN
   30 IF (IT.NE.1) GO TO 35
      ENCODE (10,55,L) VAL(1)
      HOLLR(2)=6H FALSE
      IF (L.EQ.10H         T) HOLLR(2)=5H TRUE
      RETURN
   35 IF (IT.NE.2) GO TO 40
      ENCODE (22,60,HOLLR) VAL(1)
      RETURN
   40 IF (IT.NE.3) GO TO 45
      ENCODE (24,65,HOLLP) VAL(1)
      RETURN
   45 IF (IT.NE.4) GO TO 50
      ENCODE (28,70,HOLLR) VAL(1),VAL(2)
      RETURN
   50 IF (IT.NE.5) RETURN
      ENCODE (28,75,HOLLR) VAL
      RETURN
C
   55 FORMAT (L10)
   60 FORMAT (6X,I16)
   65 FORMAT (4X,1PG20.13)
   70 FORMAT (1H(,G13.6,1H,,G12.5,1H))
   75 FORMAT (1PD28.21)
C
      END
      SUBROUTINE DPCDV (HOLLR,OCTAL)
      INTEGER HOLLR(2),OCTAL
C
C.....THIS ROUTINE ENCODES THE VALUE "OCTAL" WHERE THE TYPE OF THE VALUE
C.....IS NOT PREDETERMINED.
C
      IB=SHIFT(OCTAL,-48).A.7777B
      IF (IB.EQ.0.O.IB.EQ.7777B) GO TO 15
      IF (IB.EQ.3777B) GO TO 20
      IF (IB.EQ.4000B) GO TO 25
      IF (IB.EQ.1777B) GO TO 30
      IF (IB.EQ.6000B) GO TO 35
      ENCODE (20,40,HOLLR) OCTAL
      RETURN
   15 ENCODE (20,45,HOLLR) OCTAL
      RETURN
   20 HOLLR(1)=10H     + INF
      HOLLR(2)=5HINITE
      RETURN
   25 HOLLR(1)=10H      - INF
      HOLLR(2)=5HINITE
      RETURN
   30 HOLLR(1)=10H     + INDE
      HOLLR(2)=6HFINITE
      RETURN
   35 HOLLR(1)=10H     - INDE
      HOLLR(2)=6HFINITE
      RETURN
C
   40 FORMAT (1PG20.13)
   45 FORMAT (2X,I16,2X)
C
      END
          IDENT CONTENT
          ENTRY CONTENT
          TITLE  RETURN THE CONTENTS OF THE ARGUMENT.
          LIST -R,-G
*...THIS FUNCTION RETURNS THE CONTENTS OF THE ADDRESS PASSED IN THE
*    ARGUMENT.
 CONTENT  BSS 1
          SA1 X1
          SA1 X1
          BX6 X1
          EQ CONTENT
          END
          IDENT XILGV
          ENTRY XILGV
          TITLE  CHECK FOR ILLEGAL VALUE.
          LIST -P,-G
*...THIS FUNCTION RETURNS THE VALUE .TRUE. IF THE ARGUMENT IS OUT OF
*    RANGE OR INDEFENITE, OTHERWISE .FALSE. IS RETURNED.
 XILGV    BSSZ 1
          SA1 X1
          SX6 -1
          OR X1,XILGV
          ID X1,XILGV
          SX6 0
          EQ XILGV
          END
          IDENT XFLUSH
          ENTRY XFLUSH
          TITLE  FLUSH BUFFER OF SPECIFIED FILE.
          LIST -R,-G
*...THIS ROUTINE FLUSHES THE BUFFER OF THE FILE NAMED IN THE FIRST WORD
*    OF THE COMMON BLOCK /XRDP/.
 XFLUSH   BSSZ 1
          SA1 FILE             .GET BA OF FET FOR FILE.
          SB2 A1
          SB2 -B2
          RJ =XGETBA
          NG B2,XFLUSH
          SA4 B2+1             .CHECK FOR IN = OUT.
          SA5 B2+2
          IX6 X4-X5
          ZR X6,XFLUSH
          SA5 FUNC             .SET FUNCTION CODE OF FET+0.
          SA4 FILE
          BX6 X5+X4
          SA6 B2
          SA5 CIOC             .BUILD REQUEST FOR CIO.
          SX6 B2
          BX6 X6+X5
 +        SA1 1                .WAIT TILL LAST REQUEST COMPLETED.
          NZ X1,*
          SA6 1                .POST REQUEST FOR CIO.
 +        SA1 1                .WAIT TILL REQUEST COMPLETED.
          NZ X1,*
          EQ XFLUSH
 CIOC     VFD 18/3LCIO,2/1,40/0
 FUNC     VFD 36/0,24/24B
          USE /XRDP/
 FILE     BSS 1
 WSA      BSS 14
 EOF      BSS 1
 FET      BSS 1
          USE *
          END
          IDENT XROPI
          ENTRY XROPI
          TITLE  OPEN AND SET WSA FOR FILE.
          LIST L
          LIST -R,-G
*...THIS ROUTINE OPENS THE FILE NAMED IN THE FIRST WORD OF COMMON BLOCK
*    /XRDP/.  THE WORKING STORAGE AREA PARAMETERS IN FET+5 ARE ALSO SET.
 XROPI    BSSZ 1
          SA1 FILE            .GET BA OF FET FOR FILE.
          SB2 A1
          SB2 -B2
          RJ =XGETBA
          NG B2,XROPI
          SA5 B2              .CHECK FOR FILE ALREADY OPEN.
          SX5 X5
          NZ X5,NOPEN
 +        SA1 B2              .OPEN FILE WITH NO REWIND.
          RJ =XCPC
 +        VFD 18/4,42/120B
 NOPEN    SA1 B2              .RECALL FILE.
          RJ =XCPC
 +        VFD 18/1,2/1,40/777777B
          SX1 WSA             .SET WSA PARAMETERS IN FET+5.
          SX2 X1+14
          LX1 30
          BX6 X1+X2
          SA6 B2+5
          SX6 B2              .SAVE FET BA.
          SA6 FET
          EQ XROPI
          USE /XRDP/
 FILE     BSS 1
 WSA      BSS 14
 EOF      BSS 1
 FET      BSS 1
          USE *
          END
          IDENT XREADL
          ENTRY XREADL
          TITLE  READ ONE LINE WITH BLANK FILL.
          LIST -R,-G
*...EACH CALL TO THIS ROUTINE READS ONE 136 CHARACTER LINE INTO THE
*    WORKING STORAGE AREA (WSA).  THE READ STATUS FLAG (EOF) IS SET:
*        -1           END OF FILE
*         0           READY WITH DATA
*        +1           SHORT RECORD READ
 XREADL   BSSZ 1
          SA1 FET             .STORE BA OF FET IN CALL TO IO.
          BX6 X1
          SA6 LFN
 +        NO                  .EXECUTE READ.
          NO
          RJ =XIOREAD
 LFN      BSS 1
          SX6 0
          SA6 EOF
          ZR X1,SET           .READY WITH DATA.
          SX6 1
          SA6 EOF
          PL X1,SET           .SHORT RECORD READ.
          SA2 DEOF
          SX7 1
          SA7 DEOF
          ZR X2,XREADL
          BX6 -X6
          SA6 EOF
          EQ XREADL           .END FO FILE.
 SET      SX7 0
          SA7 DEOF
          EQ XREADL
          USE /XROP/
FILE      BSS 1
LINE      BSS 14
EOF       BSS 1
FET       BSS 1
          USE *
DEOF      BSSZ 1
          END
      OVERLAY(NEWT1,1,2)
      PROGRAM XINPUT
      COMMON /XXPRNT/ XNPR,XPRT(200),XFLPR
      COMMON /XXPLOT/XPLT(100),XRNG(2)
      COMMON/XXUNITS/XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      DIMENSION XCARD(8)
      COMMON /XXVR1FR/ XNV,XNW,XVT1(1)
      COMMON /XXVR2FR/ XVT2(1)
      COMMON XADRS(1)
      INTEGER XNPR,XPRT,XNPL,XPLT,XNW,XI,XJ,XUI,XTYPE,XICOL,XPOMP
      LOGICAL XFILM,XPLFG,XFLPR,XTRACE,QPLTS
C
C.....THIS ROUTINE PROCESSES THE DATA SECTION.  AFTER THE TYPE OF CARD
C.....HAS BEEN DETERMINED BY ROUTINE "XCRDTP" CONTROL IS PASSED TO THE
C.....APPROPRIATE ROUTINE.
C
C.....SET ALL USER-DECLARED VARIABLES TO INDEFINITE.
C
      CALL REMARK (18H   PROCESSING DATA)
      XPOMP=0
      XFILM=.FALSE.
      XPLFG=.FALSE.
      XFLPR=.FALSE.
      QPLTS=.FALSE.
      XNPL=1
      XPLT(1)=1000002B
      XRNG(1)=0.
      XRNG(2)=XRNG(1)
      XNPR=0
      DO 20 XI=2,XNW
   20 XADRS(XI)=17770000000000000000B.O.XI
C
C.....READ IN AND PROCESS EACH CARD OF DATA SECTION.
C
   25 CONTINUE
      READ (XUI,85) XCARD
      IF (EOF(XUI)) 80,30,80
   30 CALL XCRDTP (XCARD,XTYPE,XICOL)
      GO TO (35,40,45,50,55,60,65,70,73,75), XTYPE
C
C.....<PRINT.>...
C
   35 CALL XPRSTK (XCARD,XICOL)
      GO TO 25
C
C.....<PLOT.>...
C
   40 IF (QPLTS) GO TO 25
      CALL XPLSTK (XCARD,XICOL,QPLTS)
      GO TO 25
C
C.....<FLOW.>...
C
   45 CALL XFLSTK (XCARD,XICOL)
      GO TO 25
C
C.....<EVENT.>...
```

CHAPTER 4. PLOT DOCUMENTATION

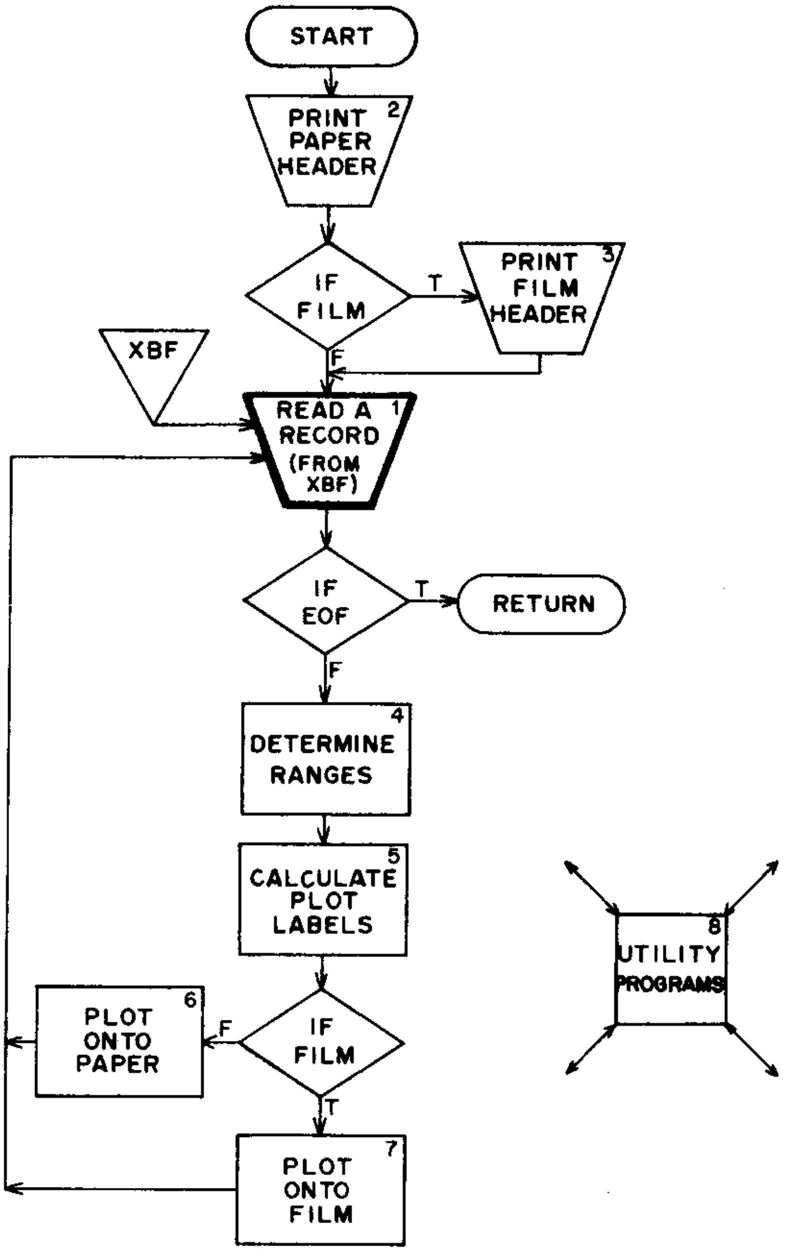


Overview

Plots requested by user (via PLØT. data cards) are generated on paper or microfilm, depending on the absence (or presence) of a FILM. card. The plot file, XBF, which directs the plotting procedures is generated in routine XPLSTK. The output (whether on film or paper) consists of (i) a header page listing the names of the dependent variables (and their identification symbols) and the independent variable name for each group of each plot and (ii) a plotted page or frame of film for each requested plot containing the graphic representation of that plot. Each plotted page contains a plot title, labels indicating the values at horizontal and vertical grid lines, and the grid area containing a vector representation of each dependent variable vs. the selected independent variable (each vector representation is labeled with the identification symbol of the dependent variable).

The chapter is expanded into blocks for easy analysis.

4.1. *Plot Supervisor*



Overview

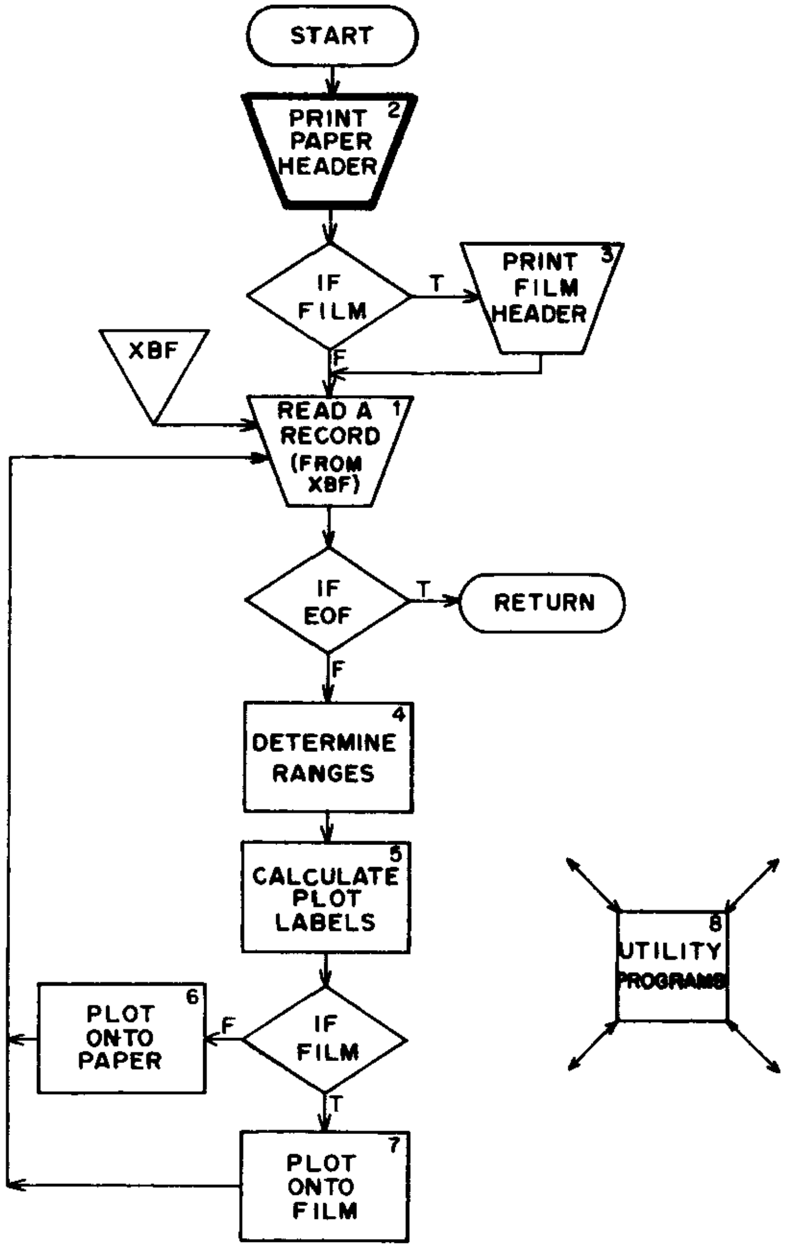
Program PLØT directs the generation of plots onto either the printer or the microfilm plotter. Each record on file XBF contains information for a single plot.

```
77000            OVERLAY(NEWT1,2,0)
77001            PROGRAM PLOT
77002      C
77003      C.....PLOT CONTROLS THE PRINTING OF PLOTS EITHER ONTO PAPER OR FILM.
77004      C
77005            COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
77006            COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
77007            COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
77008           1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
77009            INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
77010            LOGICAL FILM,LOGRP,RANGE
77011            CALL REMARK (19H   GENERATING PLOTS)
77012            KNTPLT=0
77013      C
77014      C.....PRINT A HEADER PAGE ON FILM AND/OR ON PAPER.
77015      C
77016            CALL PRTHEAD
77017            IF (FILM) CALL OVERLAY (4HMAIN,2,2,6HRECALL)
77018            DO 15 I=1,8
77019         15 ITITL(I)=10H
77020      C
77021      C.....XBF IS PROCESSED A RECORD AT A TIME--EACH RECORD DIRECTING THE
77022      C.....PRINTING OF ONE PLOT (OR A TITLE FOR A PLOT).
77023      C
77024            REWIND XBF
77025         20 READ (XBF) IBUF,RBUF
77026            IF (EOF(XBF)) 45,25,45
77027         25 IF (IBUF(3,5,6).NE.6HTITLE.) GO TO 35
77028            DO 30 J=1,3
77029            DO 30 I=1,3
77030            K=I+3*(J-1)
77031            IF (K.GT.8) GO TO 20
77032            ITITL(K)=IBUF(I,J,1)
77033         30 CONTINUE
77034            GO TO 20
77035         35 KNTPLT=KNTPLT+1
77036            CALL DETRAN, RETURNS(37)
77037            CALL LABELS
77038            REWIND XUP
77039            IF (FILM) CALL OVERLAY (4HMAIN,2,2,6HRECALL)
77040            IF (.NOT.FILM) CALL OVERLAY (4HMAIN,2,1,6HRECALL)
77041         37 DO 40 I=1,8
77042         40 ITITL(I)=10H
77043            GO TO 20
77044         45 CONTINUE
77045      C
77046            END
```

Line Number	Explanation
77011-77012	A message is printed in the dayfile stating that the plotting process has begun. KNTPLT contains the plot number currently being processed.
77016-77019	A header page describing the plots requested is printed onto paper. FILM is a logical flag set if plotting is to be done on microfilm. If plotting is desired on microfilm, a header page is printed onto a frame of microfilm. ITITL will contain the title (if specified) of the plot being processed.
77024-77026	A record of XBF is read. Each record either contains information describing a plot, or it contains a title for a plot. The format for each record is as follows: IBUF ([I=1,3],J,K) contains information pertaining to the Jth variable in the Kth group.

Line Number	Explanation
	IBUF (1-2,J,K) is the first 18 characters containing the variable name. Character 19 contains a dot if a log declaration exists for the variable, and character 20 contains the identification symbol for the variable name. IBUF([I=1,3],1,6) contains information (formatted as above describing the independent variable. RBUF(1-2,J) contains the range declarations for the Jth group. RBUF(1-2,6) contains the range declaration for the independent variable.
777027-77033	If IBUF(3,5,6) contains the flag word TITLE., then the record contains a plot title (the title labels the following record). The title is filled into ITITL and control returns to process the next record.
77035-77037	The current record contains information describing one plot. KNTPLT is the current plot number. DETRAN determines the maximum and minimum ranges for all groups and the independent variable. An abnormal return by DETRAN is made if an infinite of an indefinite value is detected in the stream of values for the independent variable. The plot is ignored and processing begins for the next record (lines 77041-77044). LABELS calculates scale labels for each dependent variable and for the independent variable.
77039-77040	The plot is printed on either paper or film.
77041-77044	Reinitialize the buffer to hold a plot title and loop back to process the next record of XBF.

4.2. *Paper Directory*



Overview

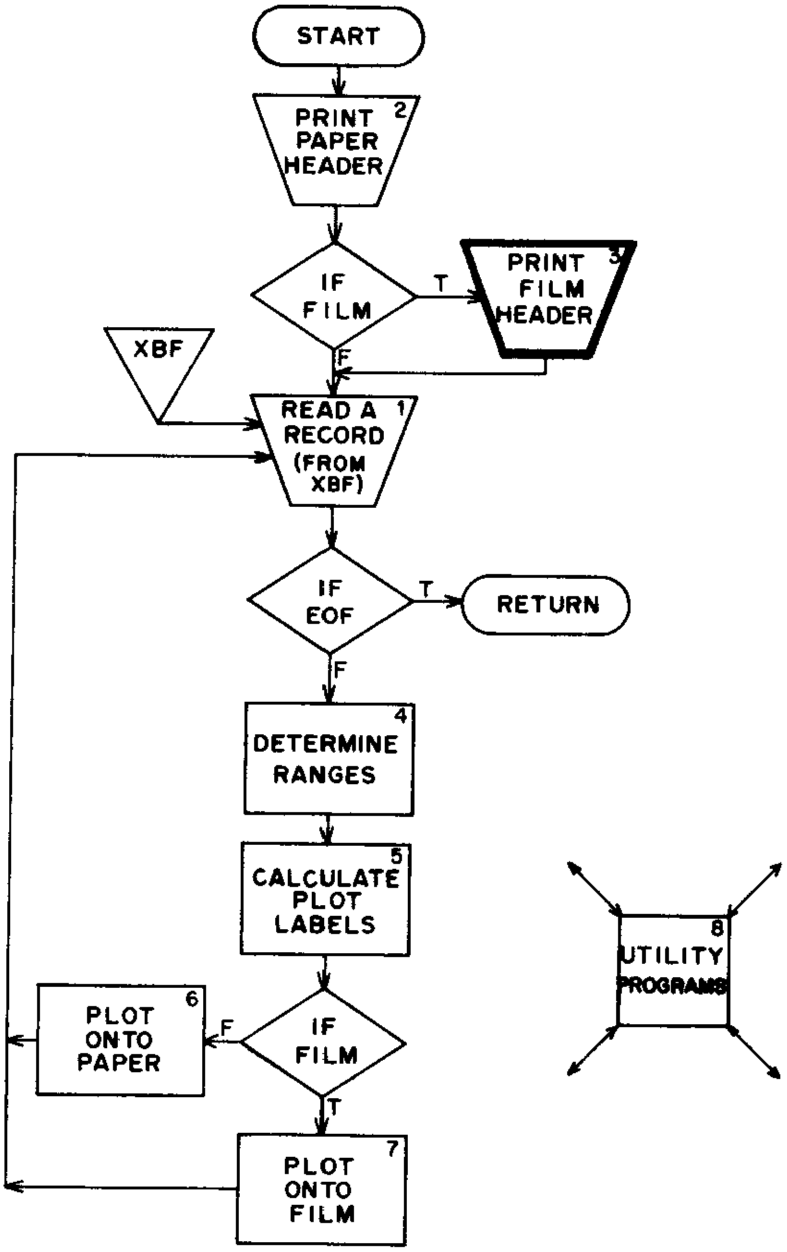
PRTHEAD prints a header page onto paper describing the independent and dependent variables for each plot requested by the user. The range limits (if defined by user) log requests, and identification characters for each variable name are listed, plot by plot.

```
78000          SUBROUTINE PRTHEAD
78001          COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
78002          COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
78003          COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
78004         1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
78005          INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
78006          LOGICAL FILM,LOGRP,RANGE
78007    C
78008    C.....PRINTS OUT A HEADER PAGE LISTING
78009    C
78010          REWIND XBF
78011          WHEN=DATE(TODAY)
78012          TIMES=TIME(NOW)
78013          WRITE (XUO,50) WHEN,TIMES
78014          I=0
78015       15 READ (XBF) IBUF,RBUF
78016          IF (EOF(XBF)) 45,20,45
78017       20 IF (IBUF(3,5,6).EQ.6HTITLE.) GO TO 15
78018          I=I+1
78019          WRITE (XUO,55)
78020          LOG=10H
78021          ICHR=IBUF(2,1,6).AND.7700B
78022          IF (ICHR.NE.0) LOG=10HLOG
78023          WRITE (XUO,60) I,IBUF(1,1,6),IBUF(2,1,6),LOG
78024          IF (RBUF(1,6).NE.0..OR.RBUF(2,6).NE.0.) WRITE (XUO,65) RBUF(1,6),R
78025         1BUF(2,6)
78026          DO 40 K=1,5
78027             IF (IBUF(1,1,K).EQ.0) GO TO 15
78028             WRITE (XUO,70) K
78029             IF (RBUF(1,K).NE.0..OR.RBUF(2,K).NE.0.) WRITE (XUO,75) RBUF(1,K
78030         1   ),RBUF(2,K)
78031             LOG=10H
78032             DO 25 J=1,5
78033                ICHR=IBUF(2,J,K).AND.7700B
78034                IF (ICHR.NE.0) LOG=10HLOG
78035       25    CONTINUE
78036             DO 30 J=1,5
78037                IF (IBUF(1,J,K).EQ.0) GO TO 35
78038                WRITE (XUO,80) IBUF(1,J,K),IBUF(2,J,K),LOG,IBUF(2,J,K)
78039       30    CONTINUE
78040       35    WRITE (XUO,85)
78041       40 CONTINUE
78042          GO TO 15
78043       45 CONTINUE
78044          RETURN
78045    C
78046       50 FORMAT (1H1,T10, 28HGRAPHICAL SIMULATION RESULTS,T60,A10,T90,A10,/
78047         1///T3,122HGRAPH   GROUP   GROUP RANGE DECLARATION    DEPENDENT VAR
78048         2IABLE(S)   PLOTTED      INDEPENDENT VARIABLE   INDEPENDENT VARIABLE
78049         3,/T3,  3HNO.,T70,  9HCHARACTER,T105, 17HRANGE DECLARATION,/)
78050       55 FORMAT (1H ,T3,125(1H-))
78051       60 FORMAT (1H0,T3,I2,T82,A10,A7,A3)
78052       65 FORMAT (1H+,T105,G10.3,  4H TO ,G10.3)
78053       70 FORMAT (1H+,T11,I2)
78054       75 FORMAT (1H+,T19,G10.3,  4H TO ,G10.3)
78055       80 FORMAT (1H+,T46,A10,A7,A3,T70,R1,/)
78056       85 FORMAT (1H )
78057    C
78058          END
```

Line Number	Explanation
78010-78013	Print a title for the header page containing the current time and date.
78014-78017	A record from XBF is read. If the contents of the record is a plot title, another record is read. (RECALL. XBF is created in XPLSTK and each record contains information pertaining to a plot or plot title.)

Line Number	Explanation
78018-78025	Write the plot number and independent variable name for the current plot. I is the current plot number. ICHR contains a dot if a log declaration was present for the independent variable. LØG contains 3HLØG if a log declaration was present. IBUF(1-2,1,6) contains the independent variable name (first 18 characters). RBUF (1-2,6) (if nonzero contains the user specified range declaration for the independent variable).
78026-78027	Print the variables of each group. K is the group counter. A zero entry in IBUF(1,1,K) indicates that there were only K-1 groups defined for present plot.
78028-78031	Print the group number and range declaration (if specified by user) for the current group.
78032-78035	Determine whether a log declaration was present for any variable in the current group. (A log declaration within a group causes the log$_{10}$ of all variables within a group to be plotted.)
78036-78039	Print the variable name, log designator, and identification character for each variable in the current group. A zero entry in IBUF(1,J,K) indicates that there were J-1 variables in the Kth group of current plot.
78040-78041	Continue printing process until each variable of each group in current plot is printed. Proceed to read in and print the contents of the next record, etc., until all plots are printed.

4.3. *Film Directory*



Overview

FLMHEAD prints a header page onto microfilm describing the independent and dependent variables for each plot requested by the user. The range limits (if defined by user), log requests, and identification characters for each variable name are listed, plot by plot.

```
      SUBROUTINE FLMHEAD
C
C.....PRINTS A HEADER PAGE ON MICROFILM
C
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,XFILM,XNPL
      INTEGER XBF
      DIMENSION H(20), NVAR(5), LOG(6)
      DATA H/10HGRAPHICAL ,10HSIMULATION,10H RESULTS  ,10HGRAPH   GR,10H
     1OUP   GROU,10HP RANGE DE,10HCLARATION ,10H   DEPENDE,10HNT VARIABL
     2,10HE(S)   PLO,10HTTED     I,10HNDEPENDENT,10H VARIABLE ,10H  INDE
     3PEND,10HENT VARIAB,10HLE        ,10HNO.       ,10HCHARACTER ,10HRA
     4NGE DECL,10HARATION   /
      WHEN=DATE(DUM)
      TIMES=TIME(DUM)
      CALL SET (0.,1.,0.,1.,1.,132.,1.,66.,1)
      CALL OPTION(0,1,0,0,0)
      I=0
      Y=1.
      REWIND XBF
   15 READ (XBF) IBUF,RBUF
      IF (EOF(XBF)) 65,20,65
   20 IF (IBUF(3,5,6).EQ.6HTITLE.) GO TO 15
      X=2.
      I=I+1
      NGRP=0
      LOG(6)=10H
      LOG(5)=LOG(6)
      LOG(4)=LOG(5)
      LOG(3)=LOG(4)
      LOG(2)=LOG(3)
      LOG(1)=LOG(2)
      NVAR(5)=0
      NVAR(4)=NVAR(5)
      NVAR(3)=NVAR(4)
      NVAR(2)=NVAR(3)
      NVAR(1)=NVAR(2)
      ICHR=IBUF(2,1,6).AND.7700B
      IF (ICHR.NE.0) LOG(6)=10HLOG
      DO 30 K=1,5
         IF (IBUF(1,1,K).EQ.0) GO TO 35
         X=X+1.
         NGRP=NGRP+1
         DO 25 J=1,5
            IF (IBUF(1,J,K).EQ.0) GO TO 30
            X=X+1.
            ICHR=IBUF(2,J,K).AND.7700B
            IF (ICHR.NE.0) LOG(K)=10HLOG
            NVAR(K)=NVAR(K)+1
   25    CONTINUE
   30 CONTINUE
   35 IF (Y.GE.X) GO TO 40
      CALL FRAME
      Y=66.
      CALL PWRT (10.,Y,H(1),30,0,0)
      CALL PWRT (60.,Y,WHEN,10,0,0)
      CALL PWRT (90.,Y,TIMES,10,0,0)
      Y=Y-3.
      CALL PWRT (3.,Y,H(4),123,0,0)
      Y=Y-1.
      CALL PWRT (3.,Y,H(17),10,0,0)
      CALL PWRT(71.,Y,H(18),10,0,0)
      CALL PWRT (107.,Y,H(19),20,0,0)
      Y=Y-2.
   40 CALL LINEP (3.,Y,126.,Y,10)
      Y=Y-2.
      CALL FRSTPT (3.,Y)
      CALL NUMBR (I,2HI2)
      CALL PWRT (82.,Y,IBUF(1,1,6),17,0,0)
      CALL PWRT (99.,Y,LOG(6),3,0,0)
      IF (RBUF(1,6).EQ.0..AND.RBUF(2,6).EQ.0.) GO TO 45
      CALL FRSTPT (105.,Y)
      CALL NUMBR (RBUF(1,6),5HG10.3)
      CALL PWRT (115.,Y,4H TO ,4,0,0)
```

```
85074              CALL FRSTPT (119.,Y)
85075              CALL NUMBR (RBUF(2,6),5HG10.3)
85076           45 DO 60 K=1,NGRP
85077                 NV=NVAR(K)
85078                 CALL FRSTPT (11.,Y)
85079                 CALL NUMBR (K,2HI2)
85080                 IF (RBUF(1,K).EQ.0..AND.RBUF(2,K).EQ.0.) GO TO 50
85081                 CALL FRSTPT (19.,Y)
85082                 CALL NUMBR (RBUF(1,K),5HG10.3)
85083                 CALL PWRT (29.,Y,4H TO ,4,0,0)
85084                 CALL FRSTPT (33.,Y)
85085                 CALL NUMBR (RBUF(2,K),5HG10.3)
85086           50    DO 55 J=1,NV
85087                    CALL PWRT (46.,Y,IBUF(1,J,K),17,0,0)
85088                    CALL PWRT (63.,Y,LOG(K),3,0,0)
85089                    CALL GCHARS (IBUF(2,J,K),10,1,ICHR)
85090                    IF (ICHR.EQ.1H$) CALL SCHARS (ICHR,2,1,1H$)
85091                    CALL PWRT (70.,Y,ICHR,1,0,0)
85092                    Y=Y-1.
85093           55    CONTINUE
85094                 Y=Y-1.
85095           60 CONTINUE
85096              GO TO 15
85097           65 CALL REMARK (23H   MICROFILM GENERATION)
85098              RETURN
85099       C
85100              END
```

Line Number	Explanation
	NOTE. SET, ØPTIØN, FRAME, PWRT, LINEP, FRSTPT, and NUMBR are CSU library routines for the film plotter and are explained in the Colorado State University Computer Center User's Manual.
85008-85019	Initialize local variables, define grid area (on microfilm plotter by calling subroutine SET), and set character size and print intensity (ØPTIØN). H contains the title for the header page. WHEN is today's date. TIMES is the current clock time.
85020-85022	Read a record from XBF. If the record consists of a plot title, read another record.
85023-85036	Initialize control variables for current plot. X is a count of the number of lines that will be taken by the current plot heading. I is the current plot number. NGRP is the number of groups in current plot. If LØG(I) contains 3HLØG, then the log of variable in the Ith group is to be plotted. NVAR(I) is the number of variables in the Ith group.
85037-85038	Determine if the log of the independent variable is to be plotted. If so, LØG(6) contains 3HLØG.

Line Number — Explanation

85039-85050

Search IBUF to determine the number of groups in the current plot.

If IBUF(1,1,K)=0, then there are K-1 groups packed into IBUF for the current plot.

IBUF(1,J,K)=0, then there are J-1 variables assigned to the Kth group.

ICHR is nonzero if a log request was present for the Jth variable of the Kth group.

LØG(K) contains 3HLØG if a log request was present for any variable in the Kth group.

85051-85063

Print the title of the header page onto film. Y counts the number of available lines. (There are 66 print lines available per page.) If the number of lines taken by the current plot (X) exceeds the number of lines available, a new frame is advanced, and the title is printed at the top of the page.

NOTE. The number of available lines (Y) is decremented each time a line is printed.

85064-85075

Print the information pertaining to the independent variable for the current plot and the current plot number.

Line Number	Item printed
85067	Current plot number.
85068	Independent variable name.
85069	A log flag, if log requests were present for the independent variable.
85072	Independent variable range declaration (if specified by user).

85076-85085

Print information pertaining to each group. Print range declaration for current group.

85086-85093

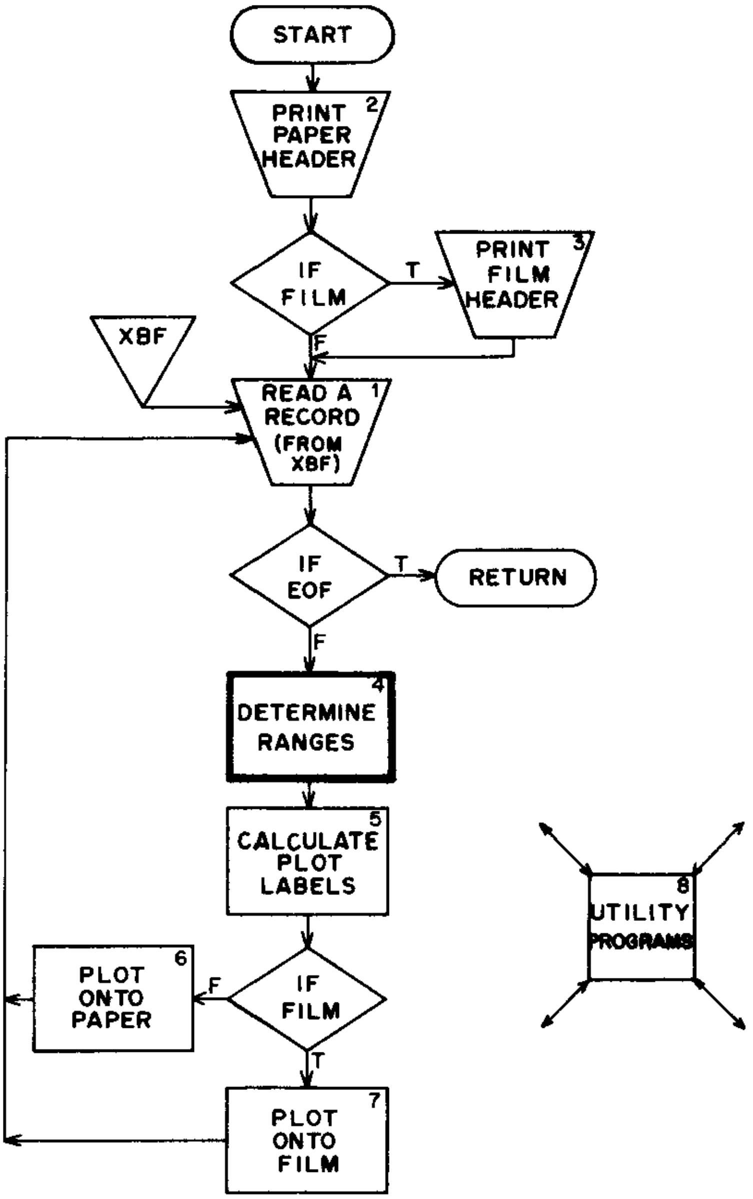
Print information describing each variable in the current group.

Line number	Item printed
85087	Variable name.
85088	Log flag (if requested).
85091	Variable ID character.

85097

After the header page is completely printed, a message is printed onto the day file, indicating microfilm has been generated.

4.4. *Determine Ranges*



Overview

DETRAN determines (i) the number of groups in the current plot, (ii) the number of variables per group, and (iii) the range over which each group (and the independent variable) will be plotted.

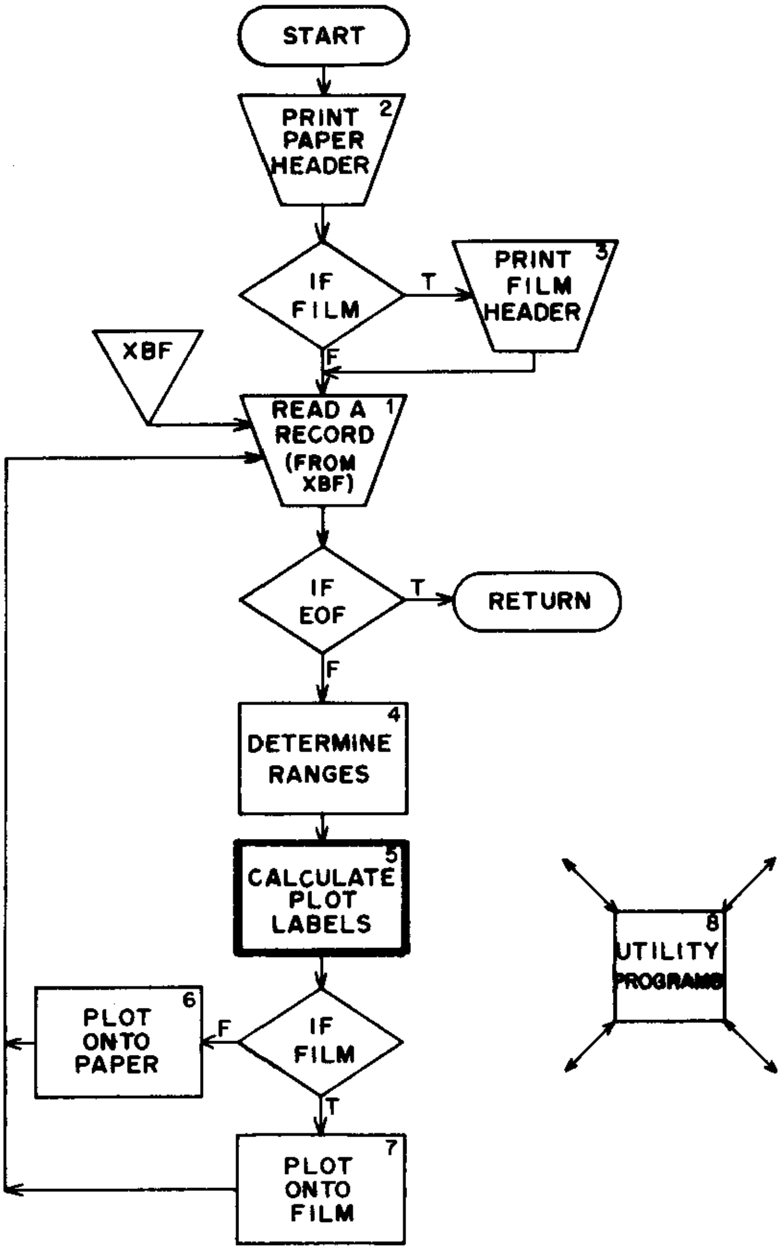
```
      SUHROUTINE DETRAN, RETURNS(M)
C
C.....DETERMINES THE MAXIMUM AND MINIMUM VALUES FOR EACH GROUP AND FOR
C.....THE INDEPENDENT VARIABLE.
C
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XRF,XTRACE,XPLFG,FILM,NVARS
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
     1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
      INTEGER XUO,XUI,XUP,XUE,XRF,XTRACE
      LOGICAL FILM,LOGRP,RANGE
      XMIN=1.E+321
      XMAX=-XMIN
      RANGE(6)=.FALSE.
      DO 15 K=1,5
         RANGE(K)=.FALSE.
         YMIN(K)=XMIN
   15 YMAX(K)=-XMIN
C
C.....DETERMINE NGRP--NUMBER OF GROUPS IN CURRENT PLOT AND NVAR(K)--
C.....NUMBER OF VARIABLES IN KTH GROUP
C
      DO 35 K=1,5
         IF (IBUF(1,1,K).EQ.0) GO TO 40
C
C.....IF RBUF(1,K).NE.RBUF(2,K) THEN A RANGE OF VALUES WAS SPECIFIED FOR
C.....THE KTH PLOT BY THE USER.
C.....MIN AND MAX VALUES ARE SET TO RBUF VALUES
C
         IF (RBUF(1,K).EQ.0..AND.RBUF(2,K).EQ.0.) GO TO 20
         RANGE(K)=.TRUE.
         YMIN(K)=RBUF(1,K)
         YMAX(K)=RBUF(2,K)
   20    DO 25 J=2,5
            IF (IBUF(1,J,K).EQ.0) GO TO 30
   25    CONTINUE
         NVAR(K)=5
         GO TO 35
   30    NVAR(K)=J-1
   35 CONTINUE
      NGRP=5
      GO TO 45
   40 NGRP=K-1
   45 IF (RBUF(1,6).GE.RBUF(2,6)) GO TO 55
      XMIN=RBUF(1,6)
      XMAX=RBUF(2,6)
      RANGE(6)=.TRUE.
      DO 50 K=1,NGRP
         IF (.NOT.RANGE(K)) GO TO 55
   50 CONTINUE
      RETURN
C
C.....SEARCH FILE CONTAINING PLOT VARIABLE VALUES TO DETERMINE
C.....MINIMUM AND MAXIMUM VALUES FOR ALL GROUPS NOT HAVING
C.....RANGE SPECIFICATIONS.
C
   55 REWIND XUP
   60 READ (XUP) (Z(II),II=1,NVARS)
      IF (EOF(XUP)) 80,65,80
   65 DO 75 K=1,NGRP
         IF (RANGE(K)) GO TO 75
         NV=NVAR(K)
         DO 70 J=1,NV
            ZZ=Z(IBUF(3,J,K))
            CALL CHECKZ(ZZ,IERR), RETURNS(70)
            YMIN(K)=AMIN1(YMIN(K),ZZ)
            YMAX(K)=AMAX1(YMAX(K),ZZ)
   70    CONTINUE
   75 CONTINUE
      IF (RANGE(6)) GO TO 60
      ZZ=Z(IBUF(3,1,6))
      CALL CHECKZ(ZZ,IERR), RETURNS(77)
      XMIN=AMIN1(XMIN,ZZ)
```

```
79073                XMAX=AMAX1(XMAX,ZZ)
79074                GO TO 60
79075             77 WRITE (XUO,85) IBUF(1,1,6),IBUF(2,1,6),KNTPLT
79076                RETURN M
79077             80 CONTINUE
79078                RETURN
79079         C
79080             85 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST,//T14, 40HATTEMPTED TO
79081               1PLOT INDEPENDENT VARIABLE ',A10,A8, 38H1 WITH AN INFINITE OR INDEF
79082               2INITE VALUE,/T14, 13HPLOT NUMBER ',I2,  9H1 IGNORED)
79083         C
79084                END
```

Line Number	Explanation
79011-79017	Initialize range control variables. XMIN will contain the minimum value of the independent variable. XMAX will contain the maximum value of the independent variable. RANGE(I) is a logical flag set to .TRUE. if a user declared range specification exists for the Ith group (I=6 corresponds to the independent variable.) YMIN(I) will contain the minimum value of all variables in the Ith group. YMAX(I) will contain the maximum value for the Ith group.
79018-79042	Scan IBUF to determine the number of groups and the number of variables per group for the current plot.
79029-79032	If the user specified a range of values for the Kth group, YMIN(K) and YMAX(K) contain the values. RANGE(K) flag is set, indicating range declarations were found for the Kth group.
79033-79038	Determine NVAR(K), the number of variables in the Kth group, for each group in the current plot.
79040-79042	NGRP contains the number of groups in the current plot.
79043-79046	If a range of values were specified for the independent variable, XMIN and XMAX will contain these values. RANGE(6) is the flag corresponding to the independent variable.
79047-79050	Search through the range flags. If all flags are set, then MAX and MIN values have been determined for each group, return.
79051-79084	Search the file containing the plot values and determine the maximum and minimum for each group (and the independent variable) not having its corresponding RANGE flag set.

Line Number	Explanation
79056-79058	Rewind the range value file, XUP. Read a record of values. RECALL. NVARS is the total number of variables present on plot cards. Each record of XUP contains the values that each of the NVARS variables had at a given time period.
79059-79068	Search each group to determine if the RANGE flag is set for that group. If not, obtain the value of each variable in that group and compute YMIN, the minimum of these values, and YMAX, the maximum of the values for the entire XUP file. IBUF(3,J,K) contains the location (index) of the value (in XUP) of the Jth variable in the Kth group. ZZ contains the value of the Jth variable in the Kth group. CHECKZ is a utility subroutine that determines whether ZZ has an infinite or indefinite value and returns control to a different part of the calling routine if this is the case.
79069-79074	Determine the minimum and maximum values for the independent variable if the RANGE flag is not set.
79075-79076	If an infinite or indefinite value is encountered while searching for minimums and maximums for the independent variable, a diagnostic is printed and the current plot will not be plotted.

4.5. *Calculate Plot Labels*



Overview

LABELS fills arrays YLINE and XLINE which are printed as dependent and independent scales labeling divisions along each axis.

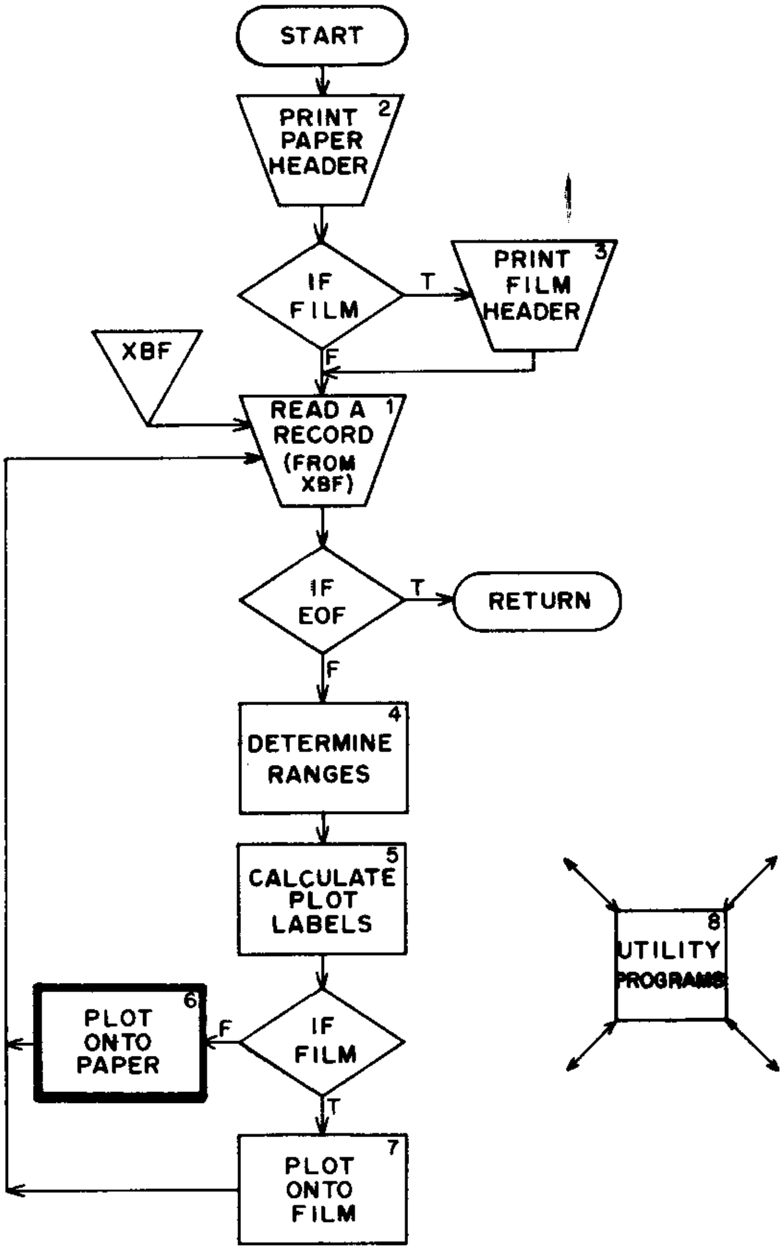
```
      SUBROUTINE LABELS
C
C.....CALCULATES X AND Y AXIS LABELS FOR CURRENT PLOT.
C
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
     1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
      INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
      LOGICAL FILM,LOGRP,RANGE
      DO 15 I=1,6
   15 LOGRP(I)=.FALSE.
C
C.....DETERMINE LABEL VALUES FOR EACH GROUP
C
      DO 55 K=1,NGRP
         NV=NVAR(K)
         DO 20 J=1,NV
            ICHR=IBUF(2,J,K).AND.7700B
            IF (ICHR.NE.0) LOGRP(K)=.TRUE.
   20    CONTINUE
         IF (LOGRP(K)) GO TO 35
C
C.....THE SCALE FOR CURRENT GROUP IS LINEAR
C
         IF (RANGE(K)) GO TO 25
         CALL ROUND (YMIN(K),YMAX(K),YMIN(K),YMAX(K),MJDY)
   25    YLINE(K,1)=YMIN(K)
         YLINE(K,6)=YMAX(K)
         ZINC=(YMAX(K)-YMIN(K))/5.
         STEP=YMIN(K)
         DO 30 J=2,5
            STEP=STEP+ZINC
   30    YLINE(K,J)=STEP
         GO TO 55
C
C.....THE SCALE FOR THE CURRENT GROUP IS LOG
C
   35    IF (YMIN(K).GT.0.) GO TO 40
         WRITE (XUO,85) YMIN(K),KNTPLT
         YMIN(K)=1.
   40    IF (YMAX(K).GT.YMIN(K)) GO TO 45
         WRITE (XUO,90) YMAX(K),KNTPLT,YMIN(K)
         YMAX(K)=10.*YMIN(K)
   45    YMIN(K)=ALOG10(YMIN(K))
         YMAX(K)=ALOG10(YMAX(K))
         YLINE(K,1)=10.**YMIN(K)
         YLINE(K,6)=10.**YMAX(K)
         ZINC=(YMAX(K)-YMIN(K))/5.
         STEP=YMIN(K)
         DO 50 J=2,5
            STEP=STEP+ZINC
   50    YLINE(K,J)=10.**STEP
   55 CONTINUE
      ICHR=IBUF(2,1,6).AND.7700B
      IF (ICHR.NE.0) LOGRP(6)=.TRUE.
      IF (LOGRP(6)) GO TO 65
C
C.....THE SCALE FOR INDEPENDENT VARIABLE IS LINEAR
C
      XLINE(1)=XMIN
      XLINE(6)=XMAX
      ZINC=(XMAX-XMIN)/5.
      STEP=XMIN
      DO 60 J=2,5
         STEP=STEP+ZINC
   60 XLINE(J)=STEP
      RETURN
C
C.....THE SCALE FOR THE INDEPENDENT VARIABLE IS LOG
C
   65 IF (XMIN.GT.0.) GO TO 70
      WRITE (XUO,85) XMIN,KNTPLT
      XMIN=1.
   70 IF (XMAX.GT.XMIN) GO TO 75
      WRITE (XUO,90) XMAX,KNTPLT,XMIN
      XMAX=10.*XMIN
```

```
80077          75 XMIN=ALOG10(XMIN)
80078             XMAX=ALOG10(XMAX)
80079             XLINE(1)=10.**XMIN
80080             XLINE(6)=10.**XMAX
80081             ZINC=(XMAX-XMIN)/5.
80082             STEP=XMIN
80083             DO 80 J=2,5
80084                STEP=STEP+ZINC
80085          80 XLINE(J)=10.**STEP
80086             RETURN
80087       C
80088          85 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST,/T14, 37HATTEMPT TO TAK
80089            1E LOG OF NUMBER .LE. 0.,/T14, 24HTHE LOWER RANGE VALUE = ,G10.3, 1
80090            20H  IN PLOT ,I2, 25H  WILL BE SET EQUAL TO 1.)
80091          90 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST,/T14, 43HUPPER RANGE VA
80092            1LUE IS .LE. LOWER RANGE VALUE,/T14, 14HUPPER VALUE = ,G10.3, 10H
80093            2IN PLOT ,I2, 47H  WILL BE SET TO 10 TIMES THE LOWER RANGE VALUE,/T
80094            314, 20HLOWER RANGE VALUE = ,G10.3)
80095       C
80096             END
```

Line Number	Explanation
80000-80011	LØGRP(I) is set to .TRUE. if a log request was present for the Ith group. XMIN,XMAX is the range of values over which the independent variable is to be plotted. YMIN(I),YMAX(I) is the range of the Ith group. XLINE(1-6) will contain the six labels for values along the independent axis. YLINE(I,1-6) will contain six labels for values of the Ith group used in labeling the dependent axis. NGRP is the number of groups in the current plot. NVAR(K) is the number of variables in the Kth group. RANGE(K) is .T. if the user specified ranges for the Kth group.
80015-80053	Determine the six label values for each group.
80017-80020	Search through all variables of the current group and set the LØGRP flag if a log request appears with any variable. RECALL. Routine XPLSTK that if the ninth character of IBUF(2,J,K) is nonzero, then a log request was present for the K variable of the Jth group.
80021-80034	A log request was *not* present for any variable in the current group; therefore, the scale for this group is linear. If the user did not specify the ranges in YMIN and YMAX (RANGE=.F.), then routine ROUND rounds off these values to present easier read labels. YLINE(K,1-6) is filled with six equally spaced numbers from YMIN(K) to YMAX(K).

Line Number	Explanation
80035-80053	A log request was present for at least one variable in the current group; therefore, the log of all variables in the group will be plotted and the labels will contain linear equivalents of the log values plotted.
80038-80040	Since logs are defined only for numbers greater than zero, if YMIN(K) is less than or equal to zero, a diagnostic is printed and YMIN(K) is set to 1.
80041-80043	If YMAX(K)<YMIN(K), a diagnostic is printed and YMAX(K) is set to 10 times the value of YMIN(K).
80044-80052	The log of the minimum and maximum range values are stored in YMIN(K) and YMAX(K). YLINE(K,1-6) is filled with the antilog of equally spaced values from YMIN(K) to YMAX(K).
80054-80056	Determine whether the log of independent variable values was requested by the user.
80060-80066	The scale for the independent variable is linear. XLINE(1-8) is filled with six equally spaced values from XMIN to XMAX.
80071-80086	The log of independent values is to be printed. The log of XMIN and XMAX is stored in XMIN and XMAX. XLINE(1-6) is filled with the antilogs of six equally spaced points from XMIN to XMAX.

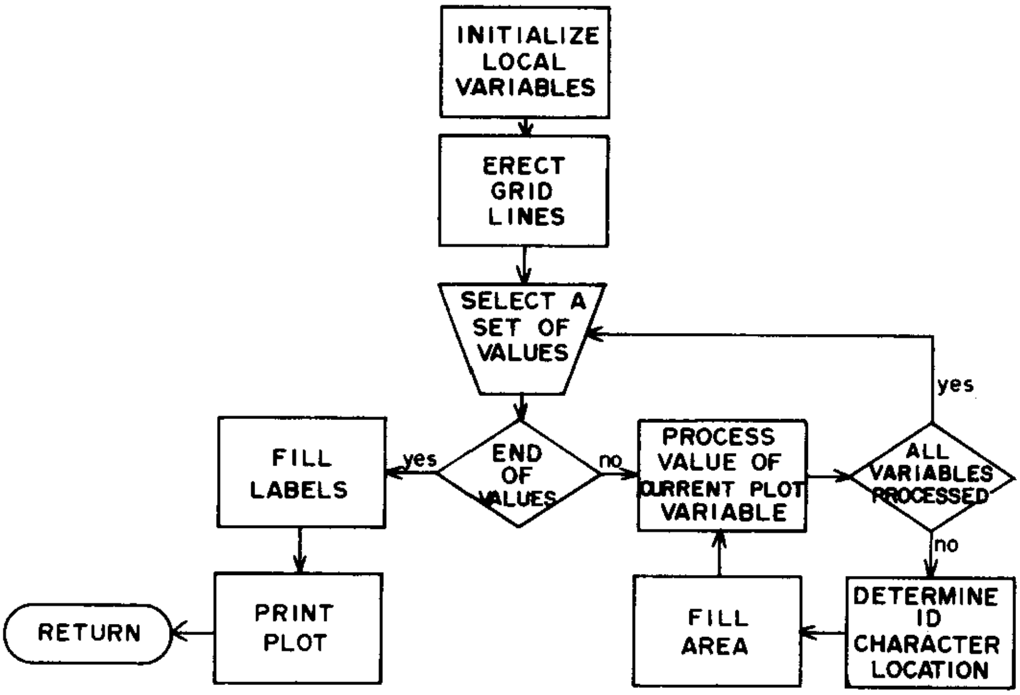
4.6. *Paper Plotter*



Overview

A requested plot is printed onto a paper page. The technique consists of mapping retrieved values for the independent and dependent variables (X,Y) into a 100 × 50 location area. At the grid location, calculated by the mapping function, a character identifying the dependent variable is placed. The process continues until all values of all variables (requested by the current plot) have been filled into the grid area. The area is then printed onto paper with appropriate title and axis labels. This section is subdivided for easier analysis.

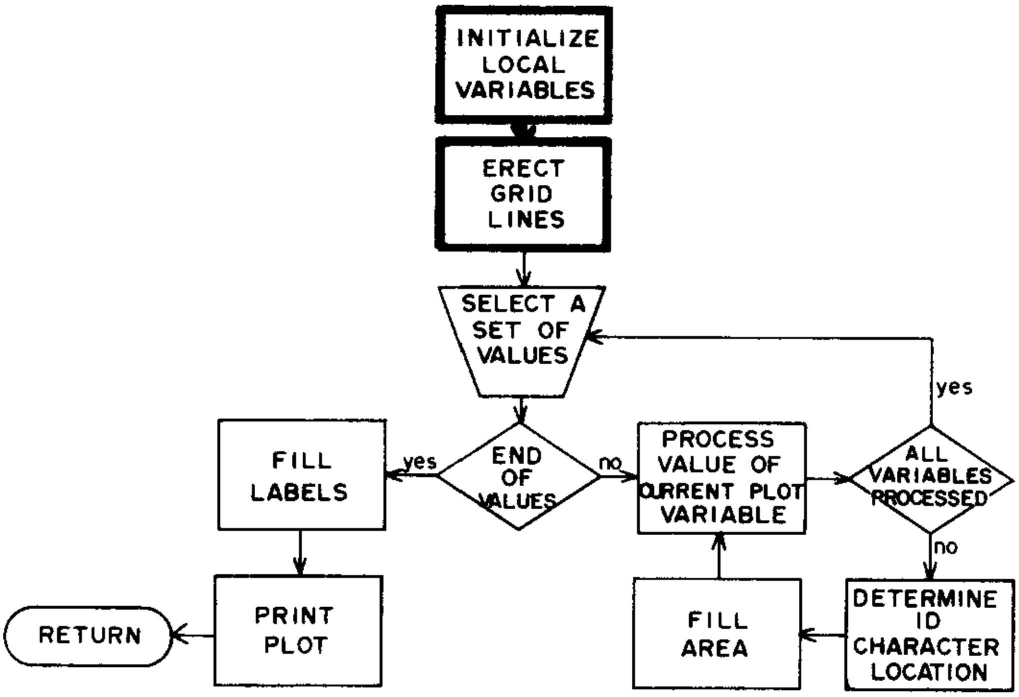
Paper plotter flow chart



```
      OVERLAY(NEWT1,2,1)
      PROGRAM PRINTPL
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
     1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
      COMMON AREA(50,100)
      DIMENSION INDF(3)
      DIMENSION LABEL(2,6,5), KEY(6), MASK(10)
      INTEGER AREA,DASH,BLANK,LINE,EQUIV
      INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
      LOGICAL FILM,LOGRP,RANGE
      LOGICAL LDIF
C
C.....GRAPH PRINTS A PLOT ON PAPER.
C
      DATA INDF/10H??????????,10H>>>>>>>>>>,10H<<<<<<<<<</
      DATA DASH/10H----------/
      DATA LINE/10HIIIIIIIIII/
      DATA BLANK/10H          /
      DATA EQUIV/10H==========/
      DATA MASK(1)/77000000000000000000B/
      DATA MASK(2)/00770000000000000000B/
      DATA MASK(3)/00007700000000000000B/
      DATA MASK(4)/00000077000000000000B/
      DATA MASK(5)/00000000770000000000B/
      DATA MASK(6)/00000000007700000000B/
      DATA MASK(7)/00000000000077000000B/
      DATA MASK(8)/00000000000000770000B/
      DATA MASK(9)/00000000000000007700B/
      DATA MASK(10)/00000000000000000077B/
      LDIF=.FALSE.
C
C.....INITAILIZE THE GRID AREA (POINTS WILL BE PLOTTED IN THE AREA
C.....DEFINED BY ARRAY (AREA).
C
      DO 15 I=1,50
      DO 15 J=1,10
   15 AREA(I,J)=BLANK
C
C.....ERECT HORIZONTAL GRID LINES IN AREA.
```

```
C
      DO 20 I=10,40,10
      DO 20 J=1,10
   20 AREA(I,J)=DASH
C
C.....ERECT VERTICAL GRID LINES.
C
      DO 25 I=1,50
      DO 25 J=2,8,2
   25 AREA(I,J)=LINE.AND.MASK(10).OR.AREA(I,J).AND.(.NOT.MASK(10))
      DELX=XMAX-XMIN
C
C.....FILE 3 CONTAINS ALL GENERATED VALUES OF VARIABLES.  DEPENDENT (Y)
C.....AND INDEPENDENT (X) VALUES ARE RECOVERED AND THE ID CHARACTER FOR
C.....EACH  DEPENDENT VARIABLE OF EACH GROUP IS PLACED IN AREA AT THE
C.....COORDINATES (X,Y)
C
   30 READ (XUP) (Z(II),II=1,NVARS)
      IF (EOF(XUP)) 70,35,70
   35 CONTINUE
      X=Z(IBUF(3,1,6))
      CALL CHECKZ(X,IERR), RETURNS(180)
      IF (.NOT.LOGRP(6)) GO TO 40
      IF (X.LE.0.) GO TO 30
      X=ALOG10(X)
   40 IF (X.LT.XMIN.OR.X.GT.XMAX) GO TO 30
C
C.....ICOL- POINTS TO A COLUMN IN AREA CORRESPONDING TO THE RECOVERED
C.....VALUE OF INDEPENDENT VARIABLE(X).
C.....IWRD--THE AREA WORD CONTAINING ICOL
C.....IPOS--THE POSITION WITHIN IWRD WHERE ICOL IS LOCATED.  EACH
C.....WORD HAS 10 POSITIONS.
C
      ICOL=INT((X-XMIN)*99./DELX+0.5)
      IWRD=ICOL/10+1
      IPOS=MOD(ICOL,10)+1
      DO 65 I=1,NGRP
         NV=NVAR(I)
      DO 65 J=1,NV
         Y=Z(IBUF(3,J,I))
         CALL CHECKZ(Y,IERR), RETURNS(63)
         IF (.NOT.LOGRP(I)) GO TO 45
         IF (Y.LE.0.) GO TO 65
         Y=ALOG10(Y)
   45    IF (Y.LT.YMIN(I).OR.Y.GT.YMAX(I)) GO TO 65
         DELY=YMAX(I)-YMIN(I)
C
C.....IROW--ROW POINTER FOR RECOVERED VALUE OF A DEPENDENT VARIABLE(Y).
C
         IROW=INT((Y-YMIN(I))*49./DELY+0.5)
         IROW=50-IROW
         ISYM=0
         ICHR=IBUF(2,J,I).AND.77B
         DO 50 K=1,10
C
C.....ISYM--ID CHARACTER TO BE ENTERED AT AREA(IROW,IWRD(IPOS))
C
            ISYM=ISYM.OR.ICHR
   50    ICHR=SHIFT(ICHR,6)
         ITEST=AREA(IROW,IWRD).AND.MASK(IPOS)
         JTEST=ISYM.AND.MASK(IPOS)
C
C.....BELOW CONTROLS FILLING OF AREA--ATTEMPTS TO FILL ID CHARS OVER
C.....DIFFERENT ID CHAR RESULTS IN A = SIGN BEING PUT IN AREA
C.....A VERTICAL OR HORIZONTAL GRID ELEMENT IS REPLACED BY AN ID CHAR.
C
         DO 55 IZIP=1,3
            INTEST=INDF(IZIP).AND.MASK(IPOS)
            IF (ITEST.EQ.INTEST) GO TO 65
   55    CONTINUE
         IF (ITEST.EQ.JTEST) GO TO 65
         JTEST=EQUIV.AND.MASK(IPOS)
         IF (ITEST.EQ.JTEST) GO TO 65
         JTEST=BLANK.AND.MASK(IPOS)
         IF (ITEST.EQ.JTEST) GO TO 60
         JTEST=DASH.AND.MASK(IPOS)
         IF (ITEST.EQ.JTEST) GO TO 60
         JTEST=LINE.AND.MASK(IPOS)
         IF (ITEST.EQ.JTEST) GO TO 60
         ISYM=EQUIV
   60    AREA(IROW,IWRD)=AREA(IROW,IWRD).AND.(.NOT.MASK(IPOS)).OR.ISYM.A
     1   ND.MASK(IPOS)
         GO TO 65
C
C.....FILL SPECIAL SYMBOLS INTO AREA FOR INDEFINITE AND INFINITE VALUES
C.....OF Y.
C
   63    ISYM=INOF(IERR)
         IROW=25
         IF (IERR.EQ.2) IROW=1
         IF (IERR.EQ.3) IROW=50
         LDIF=.TRUE.
         GO TO 60
   65 CONTINUE
      GO TO 30
   70 CONTINUE
C
C.....ARRAY LABEL IS FILLED--IT CONTAINS ID CHARS FOR LABELING Y AXIS
C.....FOLLOWED BY STEP SIZES.
C
      DO 75 I=1,6
      DO 75 J=1,5
      DO 75 K=1,2
   75 LABEL(K,I,J)=BLANK
      DO 80 I=1,NGRP
         NV=NVAR(I)
         KEY(I)=BLANK
      DO 80 J=1,NV
         CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
   80 CALL SCHARS (KEY(I),J,1,ICHR)
      IS=3
      IF (NGRP.EQ.2.OR.NGRP.EQ.3) IS=2
      IF (NGRP.GE.4) IS=1
      DO 85 I=1,6
         IJ=IS
         K=7-I
      DO 85 J=1,NGRP
         ENCODE (20,120,LABEL(1,I,IJ) )KEY(J),YLINE(J,K)
         IF (IJ.EQ.3) ENCODE (20,125,LABEL(1,I,IJ) )KEY(J),YLINE(J,K)
   85 IJ=IJ+1
C
C.....PRINT PLOT HEADING AND TITLE.
C
      WRITE (XUO,115) KNTPLT,ITITL
      WRITE (XUO,130) (LABEL(N,1,1),N=1,2)
      WRITE (XUO,135) (LABEL(N,1,2),N=1,2)
C
C.....PRINT Y LABELS WHERE APPROPRIATE AND FLUSH AREA LINE BY LINE.
C
      DO 90 I=1,3
         J=I+2
   90 WRITE (XUO,140) (LABEL(N,1,J),N=1,2),(AREA(I,N),N=1,10)
      DO 95 I=4,7
   95 WRITE (XUO,145) (AREA(I,N),N=1,10)
      M=7
      DO 105 I=2,5
         DO 100 J=1,5
          M=M+1
  100    WRITE (XUO,140) (LABEL(N,I,J),N=1,2),(AREA(M,N),N=1,10)
      DO 105 J=1,5
          M=M+1
  105 WRITE (XUO,145) (APEA(M,N),N=1,10)
      DO 110 I=1,3
          M=M+1
  110 WRITE (XUO,140) (LABEL(N,6,I),N=1,2),(AREA(M,N),N=1,10) .
      WRITE (XUO,150)
      WRITE (XUO,155) (LABEL(N,6,4),N=1,2)
      WRITE (XUO,130) (LABEL(N,6,5),N=1,2)
      WRITE (XUO,160) (XLINE(I),I=1,6)
      WRITE (XUO,165) IBUF(1,1,6),IBUF(2,1,6)
      IF (LDIF) WRITE (XUO,170)
      GO TO 185
C
  115 FORMAT (1H1,  9HPLOT NO. ,I2,T26,7A10,A4)
  120 FORMAT (A5,1X,G13.6,1X)
  125 FORMAT (A5,1X,G13.6,1H-)
  130 FORMAT (1H ,4X,2A10)
  135 FORMAT (1H ,4X,2A10,1X,100(1H_))
  140 FORMAT (1H ,4X,2A10,1H",10A10,1H")
  145 FORMAT (1H ,24X,1H",10A10,1H")
  150 FORMAT (1H+,T27,100(1H_))
  155 FORMAT (1H ,4X,2A10,1X,1H",18X,1H",4(19X,1H"))
  160 FORMAT (1H ,19X,G13.6,6X,G13.6,4(7X,G13.6))
  165 FORMAT (1H0,55X,A10,A8)
  170 FORMAT (6H0*****,102HNOTE--IN PRECEDING PLOT, EITHER INFINITE OR I
     1NDEFINITE VALUES WERE ENCOUNTERED FOR DEPENDENT VARIABLES./, 84H C
     2HECK PLOT FOR FOLLOWING INDICATORS--?(INDEFINITE), >(+ INFINITE),
     3OR <(- INFINITE))
  175 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST.//T14, 40HATTEMPTED TO
     1PLOT INDEPENDENT VARIABLE ',A10,A8, 38H' WITH AN INFINITE OR INDEF
     2INITE VALUE./T14, 13HPLOT NUMBER ',I2,  9H' IGNORED)
  180 WRITE(XUO,175) IBUF(1,1,6),IBUF(2,1,6),KNTPLT
  185 CONTINUE
C
      END
```

Erect grid lines

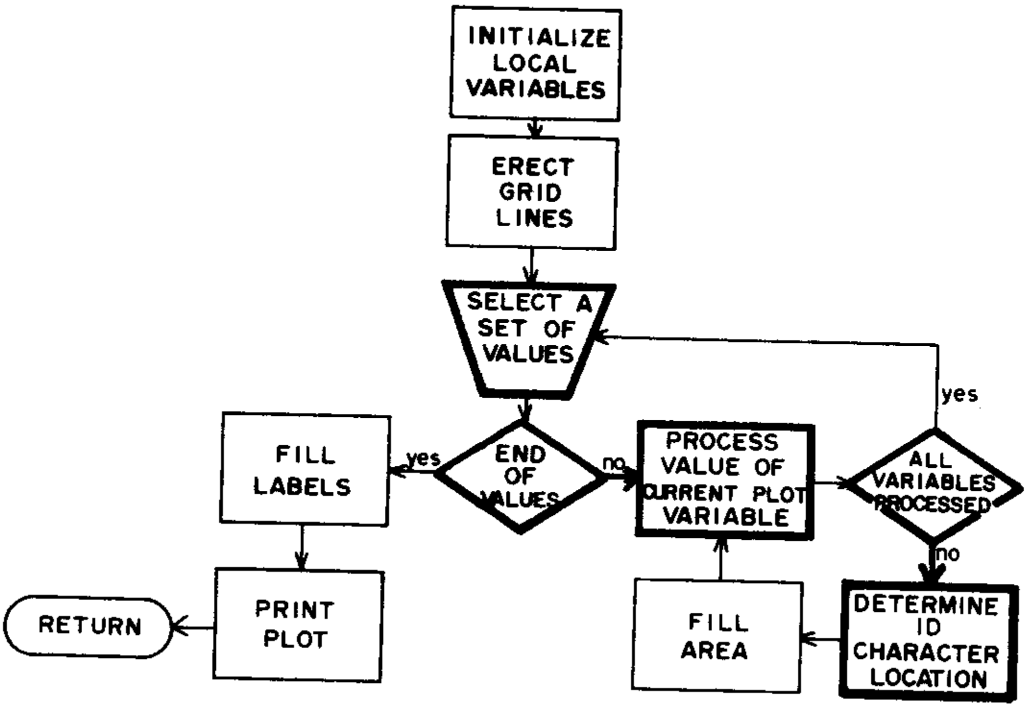


```
      OVEPLAY(NEwT1,2,1)
      PROGRAM PRINTPL
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
     1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
      COMMON AREA(50,100)
      DIMENSION INDF(3)
      DIMENSION LABEL(2,6,5), KEY(6), MASK(10)
      INTEGER AREA,DASH,BLANK,LINE,EQUIV
      INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
      LOGICAL FILM,LOGRP,RANGE
      LOGICAL LDIF
C
C.....GRAPH PRINTS A PLOT ON PAPER.
C
      DATA INDF/10H??????????,10H>>>>>>>>>>,10H<<<<<<<<<</
      DATA DASH/10H----------/
      DATA LINE/10H||||||||||/
      DATA BLANK/10H          /
      DATA EQUIV/10H==========/
      DATA MASK(1)/77000000000000000000B/
      DATA MASK(2)/00770000000000000000B/
      DATA MASK(3)/00007700000000000000B/
      DATA MASK(4)/00000077000000000000B/
      DATA MASK(5)/00000000770000000000B/
      DATA MASK(6)/00000000007700000000B/
      DATA MASK(7)/00000000000077000000B/
      DATA MASK(8)/00000000000000770000B/
      DATA MASK(9)/00000000000000007700B/
      DATA MASK(10)/00000000000000000077B/
      LDIF=.FALSE.
```

```
83032          C
83033          C.....INITAILIZE THE GRID AREA (POINTS WILL BE PLOTTED IN THE AREA
83034          C.....DEFINED BY ARRAY (AREA).
83035          C
83036                DO 15 I=1,50
83037                DO 15 J=1,10
83038             15 AREA(I,J)=BLANK
83039          C
83040          C.....ERECT HORIZONTAL GRID LINES IN AREA.
83041          C
83042                DO 20 I=10,40,10
83043                DO 20 J=1,10
83044             20 AREA(I,J)=DASH
83045          C
83046          C.....ERECT VERTICAL GRID LINES.
83047          C
83048                DO 25 I=1,50
83049                DO 25 J=2,8,2
83050             25 AREA(I,J)=LINE.AND.MASK(10).OR.AREA(I,J).AND.(.NOT.MASK(10))
83051                DELX=XMAX-XMIN
```

Line Number	Explanation
83016-83051	Initialize local variables and erect grid lines. INDF contains special characters which will be printed if an indefinite or infinite value is detected. Arrays DASH and LINE contain characters used to erect the grid lines. EQUIV is the character filled into an area if there is more than one character to be filled into a given location. MASK selects the location within a word of AREA where character is to be filled. LDIF is the flag indicating that an indefinite or infinite value has been encountered. AREA is the total grid area (50 characters high and 100 characters wide).

Determine plot location



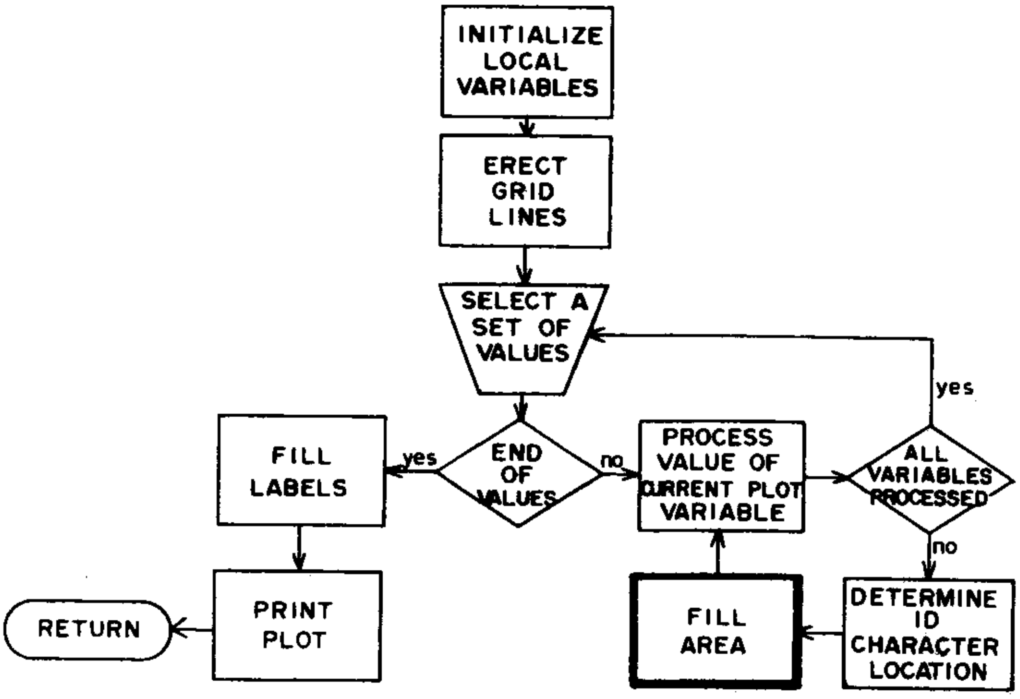
```
C
C.....FILE 3 CONTAINS ALL GENERATED VALUES OF VARIABLES.  DEPENDENT (Y)
C.....AND INDEPENDENT (X) VALUES ARE RECOVERED AND THE ID CHARACTER FOR
C.....EACH  DEPENDENT VARIABLE OF EACH GROUP IS PLACED IN AREA AT THE
C.....COORDINATES (X,Y)
C
   30 READ (XUP) (Z(II),II=1,NVARS)
      IF (EOF(XUP)) 70,35,70
   35 CONTINUE
      X=Z(IBUF(3,1,6))
      CALL CHECKZ(X,IERR), RETURNS(180)
      IF (.NOT.LOGRP(6)) GO TO 40
      IF (X.LE.0.) GO TO 30
      X=ALOG10(X)
   40 IF (X.LT.XMIN.OR.X.GT.XMAX) GO TO 30
C
C.....ICOL- POINTS TO A COLUMN IN AREA CORRESPONDING TO THE RECOVERED
C.....VALUE OF INDEPENDENT VARIABLE(X).
C.....IWRD--THE AREA WORD CONTAINING ICOL
C.....IPOS--THE POSITION WITHIN IWRD WHERE ICOL IS LOCATED.  EACH
C.....WORD HAS 10 POSITIONS.
C
      ICOL=INT((X-XMIN)*99./DELX+0.5)
      IWRD=ICOL/10+1
      IPOS=MOD(ICOL,10)+1
      DO 65 I=1,NGRP
         NV=NVAR(I)
      DO 65 J=1,NV
         Y=Z(IBUF(3,J,I))
         CALL CHECKZ(Y,IERR), RETURNS(63)
         IF (.NOT.LOGRP(I)) GO TO 45
         IF (Y.LE.0.) GO TO 65
         Y=ALOG10(Y)
   45    IF (Y.LT.YMIN(I).OR.Y.GT.YMAX(I)) GO TO 65
         DELY=YMAX(I)-YMIN(I)
C
C.....IROW--ROW POINTER FOR RECOVERED VALUE OF A DEPENDENT VARIABLE(Y).
```

```
83089          C
83090                    IROW=INT((Y-YMIN(I))*49./DELY+0.5)
83091                    IROW=50-IROW
83092                    ISYM=0
83093                    ICHR=IBUF(2,J,I).AND.77B
83094                    DO 50 K=1,10
83095          C
83096          C.....ISYM--ID CHARACTER TO BE ENTERED AT AREA(IROW,IWRD(IPOS))
83097          C
83098                       ISYM=ISYM.OR.ICHR
83099             50     ICHR=SHIFT(ICHR,6)
83100                    ITEST=AREA(IROW,IWRD).AND.MASK(IPOS)
83101                    JTEST=ISYM.AND.MASK(IPOS)
```

Line Number	Explanation
83058-83060	File XUP contains the generated values of all variables in the plot stack. The first record contains a list of the values of the variables at TIME=TSTRT, the second record at TIME=TSTRT+DTPR, etc. Each value of the variable in IBUF(1-2,J,K) is stored at Z(IBUF(3,J,K)) in each record of the file.
83061-83066	X is the current value of the independent variable. (If the log of the variable is desired to be plotted, then X contains the log of the current value.)
83074-83076	ICØL is the column of the plot area corresponding to the current value of X. (The range of the independent values are mapped into the 100 column range of AREA.) IWRD is the word containing ICØL. Ten words with 10 locations each comprise the length of the independent axis. IPØS is the location within IWRD where ICØL is located.
83077-83086	The current value of each variable in each group requested for the plot is mapped onto its proper position in AREA. NV is the number of variables in the current group. Y is the value of the current dependent variable. CHECK examines Y to determine if it is indefinite or infinite and returns to a different location if it is (RETURNS(52)). DELY is the range of the dependent variables for the current group.
83090-83094	IRØW is the row of the plot area corresponding to the current Y value. (The range of the independent variables are mapped into the 50 row range of the grid area.)

Line Number	Explanation
83097-83101	ISYM contains the identification character of the current variable. The 10 locations in ICHR are filled with the character in ISYM (SHIFT shifts the contents of ICHR six bits left). ITEST contains the character presently in the location of area determined by (X,Y). JTEST contains the character to be inserted into the (X,Y) location of AREA, e.g., AREA [IRØW,IWRD(IPØS)].

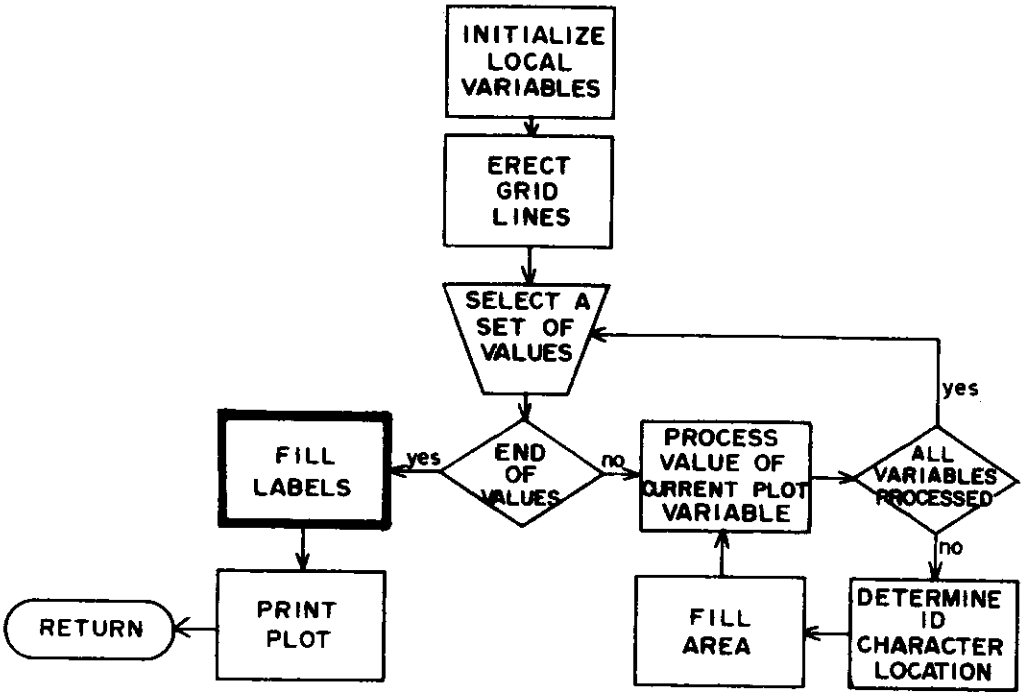
Map ID symbol into location



```
C
C.....BELOW CONTROLS FILLING OF AREA--ATTEMPTS TO FILL ID CHARS OVER
C.....DIFFERENT ID CHAR RESULTS IN A = SIGN BEING PUT IN AREA
C.....A VERTICAL OR HORIZONTAL GRID ELEMENT IS REPLACED BY AN ID CHAR.
C
          DO 55 IZIP=1,3
             INTEST=INDF(IZIP).AND.MASK(IPOS)
             IF (ITEST.EQ.INTEST) GO TO 65
   55     CONTINUE
          IF (ITEST.EQ.JTEST) GO TO 65
          JTEST=EQUIV.AND.MASK(IPOS)
          IF (ITEST.EQ.JTEST) GO TO 65
          JTEST=BLANK.AND.MASK(IPOS)
          IF (ITEST.EQ.JTEST) GO TO 60
          JTEST=DASH.AND.MASK(IPOS)
          IF (ITEST.EQ.JTEST) GO TO 60
          JTEST=LINE.AND.MASK(IPOS)
          IF (ITEST.EQ.JTEST) GO TO 60
          ISYM=EQUIV
   60     AREA(IROW,IWRD)=AREA(IROW,IWRD).AND.(.NOT.MASK(IPOS)).OR.ISYM.A
     1    ND.MASK(IPOS)
          GO TO 65
C
C.....FILL SPECIAL SYMBOLS INTO AREA FOR INDEFINITE AND INFINITE VALUES
C.....OF Y.
C
   63     ISYM=INDF(IERR)
          IROW=25
          IF (IERR.EQ.2) IROW=1
          IF (IERR.EQ.3) IROW=50
          LDIF=.TRUE.
          GO TO 60
   65 CONTINUE
      GO TO 30
   70 CONTINUE
```

Line Number	Explanation
83107-83110	A symbol identifying an infinite or indefinite value takes precedence over any character to be inserted. (Thus a character is ignored if it is to be inserted at a location containing an INDF character.)
83111-83113	A character is ignored if it is to be inserted at a location containing: (1) the identical character. (2) an equivalent sign (indicates that several different characters share the same location).
83114-83120	The character replaces (1) a blank. (2) a horizontal or vertical grid element. An equivalent sign is inserted if the current character (is not any of the above) and the symbol to be inserted are different.
83121-83123	The location of AREA is filled with the determined character.
83128-83136	If Y is indefinite or infinite, a special symbol is filled into AREA. CHECKZ detects abnormal values and passes control to this location. IERR is assigned a value by CHECKZ (line 83081). IERR = 1 If Y value is indefinite. = 2 If negative infinite. = 3 If positive infinite. = 4 If normal value.

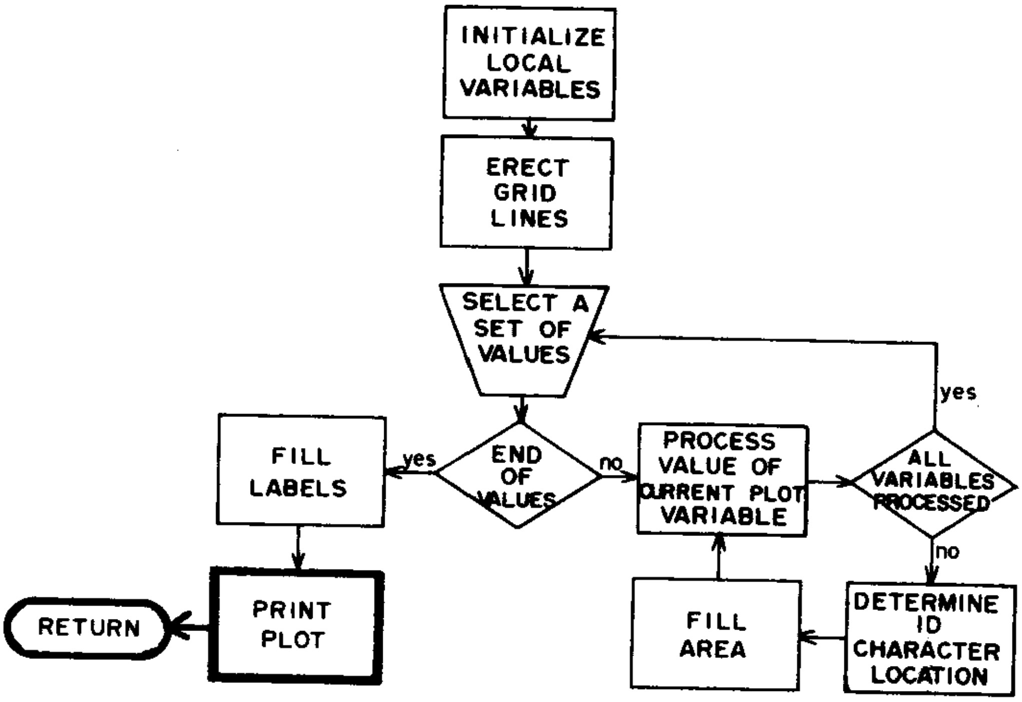
Fill Y axis labels



```
      C
      C.....ARPAY LABEL IS FILLED--IT CONTAINS ID CHARS FOR LABELING Y AXIS
      C.....FOLLOWED RY STEP SIZES.
      C
            DO 75 I=1,6
            DO 75 J=1,5
            DO 75 K=1,2
         75 LABEL(K,I,J)=BLANK
            DO 80 I=1,NGRP
               NV=NVAR(I)
               KEY(I)=BLANK
            DO 80 J=1,NV
               CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
         80 CALL SCHARS (KEY(I),J,1,ICHR)
            IS=3
            IF (NGRP.EQ.2.OR.NGRP.EQ.3) IS=2
            IF (NGRP.GE.4) IS=1
            DO 85 I=1,6
               IJ=IS
               K=7-I
            DO 85 J=1,NGRP
               ENCODE (20,120,LABEL(1,I,IJ) )KEY(J),YLINE(J,K)
               IF (IJ.EQ.3) ENCODE (20,125,LABEL(1,I,IJ) )KEY(J),YLINE(J,K)
         85 IJ=IJ+1
```

Line Number	Explanation
83140-83160	The Y axis label is filled. It contains ID symbols for each variable in a particular group and the step sizes of each group. KEY(I) contains the list of ID symbols for the Ith group. IJ controls which parts of LABEL are filled if there are fewer than five groups requested. Five labels (corresponding to a maximum of five possible groups) are printed one below another to the left of each of the six horizontal grid lines. If there are five groups, then the label of the third group would be exactly adjacent to each grid line. Thus the set of labels is centered on each grid line. To center the label set if fewer than five groups are requested requires shifting the group information into later labels. (Thus, if only one group is requested, the label information of that group is filled into the third label, with labels 1,2,4, and 5 left blank, IJ=3. The label set, when printed, would show the group centered on each grid line.) LABEL(1,I,IJ) contains the Ith step size and ID list of the Jth group.

Output plot

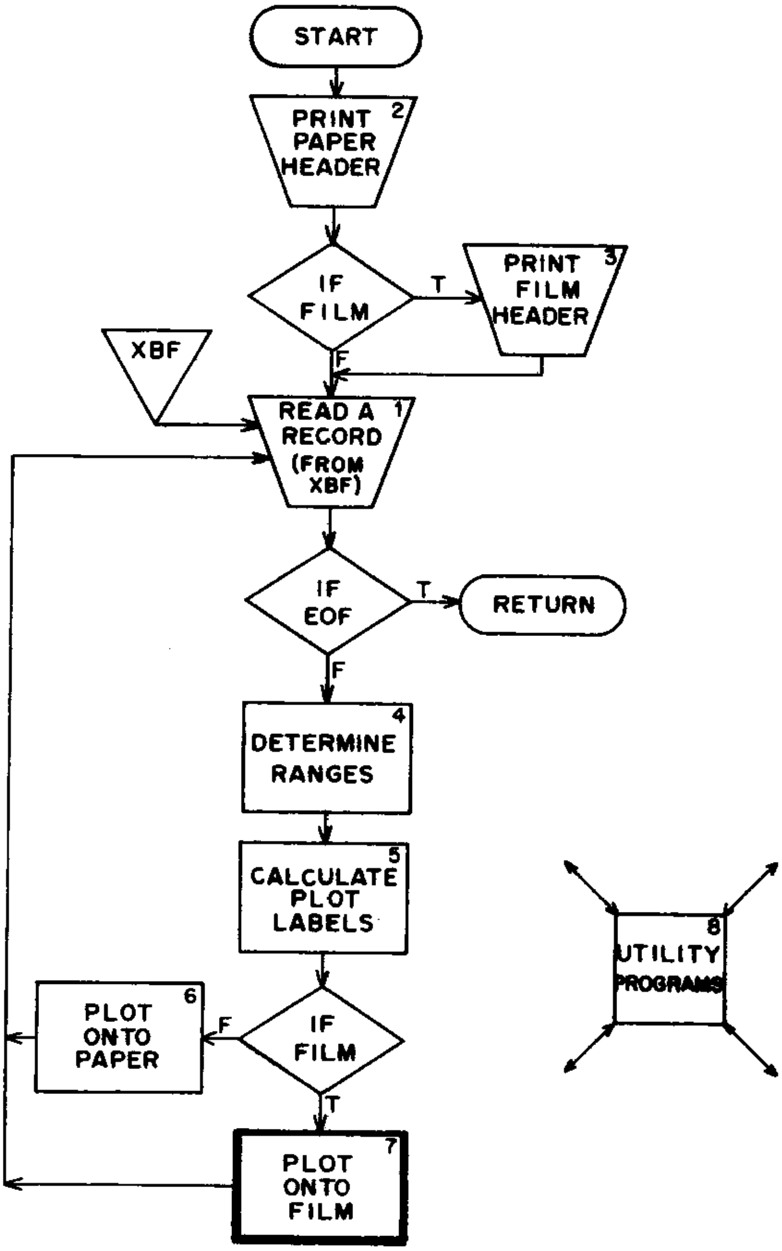


```
C
C.....PRINT PLOT HEADING AND TITLE.
C
      WRITE (XUO,115) KNTPLT,ITITL
      WRITE (XUO,130) (LABEL(N,1,1),N=1,2)
      WRITE (XUO,135) (LABEL(N,1,2),N=1,2)
C
C.....PRINT Y LABELS WHERE APPROPRIATE AND FLUSH AREA LINE BY LINE.
C
      DO 90 I=1,3
         J=I+2
   90 WRITE (XUO,140) (LABEL(N,1,J),N=1,2),(AREA(I,N),N=1,10)
      DO 95 I=4,7
   95 WRITE (XUO,145) (AREA(I,N),N=1,10)
      M=7
      DO 105 I=2,5
         DO 100 J=1,5
            M=M+1
  100    WRITE (XUO,140) (LABEL(N,I,J),N=1,2),(AREA(M,N),N=1,10)
      DO 105 J=1,5
         M=M+1
  105 WRITE (XUO,145) (AREA(M,N),N=1,10)
      DO 110 I=1,3
         M=M+1
  110 WRITE (XUO,140) (LABEL(N,6,I),N=1,2),(AREA(M,N),N=1,10)
      WRITE (XUO,150)
      WRITE (XUO,155) (LABEL(N,6,4),N=1,2)
      WRITE (XUO,130) (LABEL(N,6,5),N=1,2)
      WRITE (XUO,160) (XLINE(I),I=1,6)
      WRITE (XUO,165) IBUF(1,1,6),IBUF(2,1,6)
      IF (LDIF) WRITE (XUO,170)
      GO TO 185
C
  115 FORMAT (1H1, 9HPLOT NO. ,I2,T26,7A10,A4)
  120 FORMAT (A5,1X,G13.6,1X)
```

```
83196           125 FORMAT (A5,1X,G13.6,1H-)
83197           130 FORMAT (1H ,4X,2A10)
83198           135 FORMAT (1H ,4X,2A10,1X,100(1H_))
83199           140 FORMAT (1H ,4X,2A10,1H",10A10,1H")
83200           145 FORMAT (1H ,24X,1H",10A10,1H")
83201           150 FORMAT (1H+,T27,100(1H_))
83202           155 FORMAT (1H ,4X,2A10,1X,1H",18X,1H",4(19X,1H"))
83203           160 FORMAT (1H ,19X,G13.6,6X,G13.6,4(7X,G13.6))
83204           165 FORMAT (1H0,55X,A10,A8)
83205           170 FORMAT (6H0*****,102HNOTE--IN PRECEDING PLOT, EITHER INFINITE OR I
83206              1NDEFINITE VALUES WERE ENCOUNTERED FOR DEPENDENT VARIABLES,/, 84H C
83207              2HECK PLOT FOR FOLLOWING INDICATORS--?(INDEFINITE), >(+ INFINITE),
83208              3OR <(- INFINITE))
83209           175 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST,//T14, 40HATTEMPTED TO
83210              1PLOT INDEPENDENT VARIABLE ',A10,A8, 38H| WITH AN INFINITE OR INDEF
83211              2INITE VALUE,/T14, 13HPLOT NUMBER ',I2,  9H| IGNORED)
83212           180 WRITE(XUO,175) IBUF(1,1,6),IBUF(2,1,6),KNTPLT
83213           185 CONTINUE
83214         C
83215               END
```

Line Number	Explanation
83163-83166	The plot heading, containing the plot number and title (optional), is printed.
83170-83188	AREA is printed a horizontal row at a time. The labels are printed to align with the horizontal grid lines. (The set of five labels is printed beginning at rows 1, 8, 18, 28, ... Therefore, the third label of each set would be printed to the left of rows 1, 10, 20, 30, 40, and 50. RECALL. The horizontal grid lines were filled into AREA at rows 10, 20, 30, and 40.)
83189-83191	The step size of the independent variable is printed below AREA with each step size aligned under a vertical grid line. The name of the independent is printed, labeling the X axis. If indefinite or infinite values are encountered, a message alerting the user is printed.

4.7. *Microfilm Plotter*

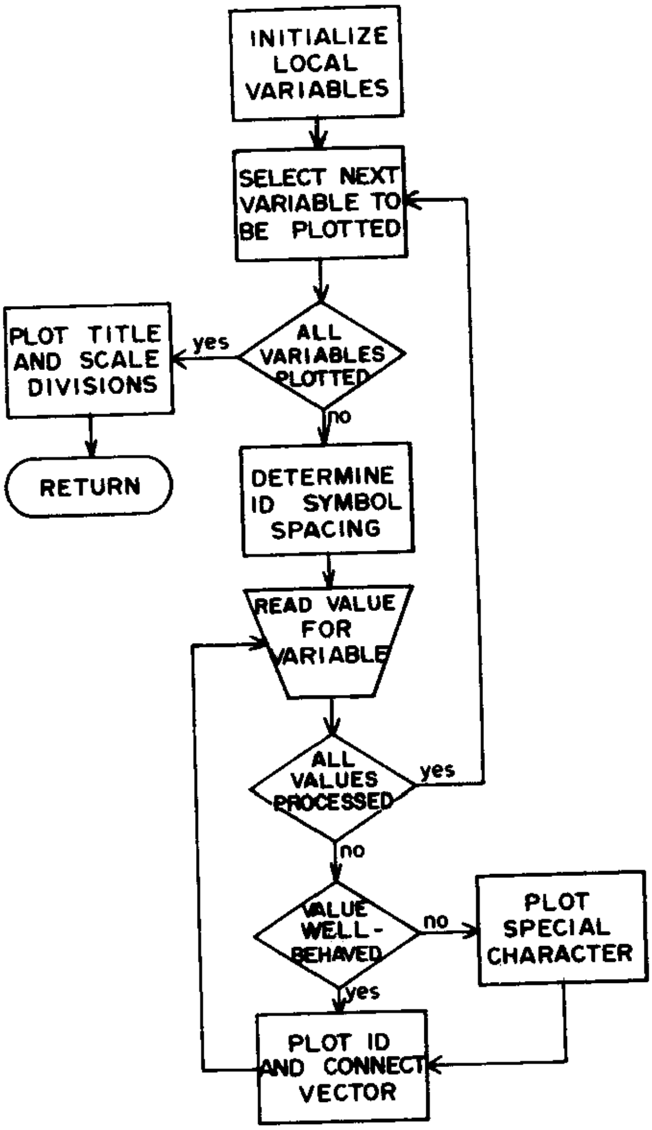


Overview

A requested plot is printed onto a frame of microfilm. The plotting technique consists of (i) retrieving a value from the value file for the current dependent variable, (ii) connecting a line segment from the previous coordinate (the preceding independent and dependent values) to the current coordinate, (iii) printing the ID symbol for the curve five times across the graph, and (iv) printing special characters if a retrieved dependent value is indefinite or infinite. This process is repeated for each variable in each group of the current plot. After all variables have been plotted, grid lines, the plot title, and scale divisions (labeling values of the grid lines) are printed onto the frame.

The section is subdivided for easier analysis.

Microfilm plotter flow chart

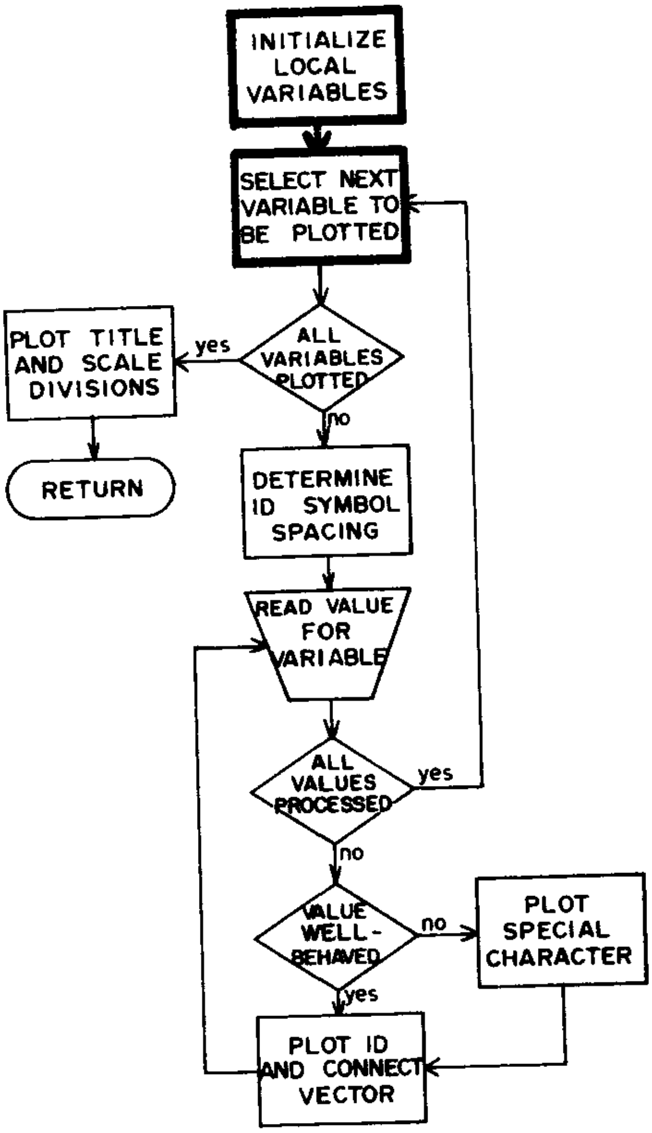

INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
PLOT TITLE AND SCALE DIVISIONS
RETURN
no
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
PLOT SPECIAL CHARACTER
yes
PLOT ID AND CONNECT VECTOR

```
       OVERLAY(NEWT1,2,2)
       PROGRAM MICRO
       COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
       COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
       COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
      1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
       INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
       LOGICAL FILM,LOGRP,RANGE
       DIMENSION KEY(5), INFIN(8)
       LOGICAL ICHAR,LDIF
       LOGICAL IFIRST
       DATA NTITL/10H PLOT NO. /
       DATA INFIN/10HPLOT VARIA,10HBLE WENT I,10HNFINITE OR,10H INDEFINIT
      1,10HE--CHECK G,10HRAPH FOR S,10HPECIAL CHA,10HRACTERS    /
```

```
C
C.....MICRO PRINTS DESIRED PLOTS ON FILM.  5 GROUPS AND A MAXIMUM OF 5
C.....VARIABLES PER GROUP IS ALLOWED.  (TOTAL OF 25 VARIABLES PER PLOT.)
C
      IF (KNTPLT.NE.0) GO TO 15
      CALL FLMHEAD
      GO TO 115
   15 CALL FRAME
      CALL OPTION (0,1,0,0,0)
      CALL SETLINE (1)
      ICHAR=.FALSE.
      LDIF=.FALSE.
      IFPOS=-11
      IFNEG=-11
      IFIND=-11
      BLANK=10H
      KNT=0
C
C.....THE FOLLOWING LOOPS COLLECT VALUES FOR EACH VARIABLE IN A PARTICU-
C.....LAR PLOT AND PRINT EACH VALUE VERSES THE INDEPENDENT VARIABLE
C.....ON A MICROFILM GRID.
C
      DO 75 I=1,NGRP
         CALL SET (.14,.94,.1,.9,XMIN,XMAX,YMIN(I),YMAX(I),1)
         NV=NVAR(I)
      DO 75 J=1,NV
C
C.....VARIABLE ID SYMBOLS ARE PRINTED NEAR EACH PLOTTED VARIABLE CURVE.
C.....FOLLOWING ROUTINE 1).  ARRANGES 5 SYMBOLS FOR A CURVE ACROSS THE
C.....GRID.  2).  OFFSETS ID SYMBOLS OF DIFFERENT VARIABLES.
C
         KNT=KNT+1
         IF (KNT.GT.5) KNT=1
         ZKNT=KNT
         X1=XMIN
         Y1=YMIN(I)
         IFIRST=.TRUE.
         PEWIND XUP
         W1=ABS(XLINE(2))
         WHER=XLINE(2)*.2*ZKNT
   20    READ (XUP) (Z(II),II=1,NVARS)
         IF (EOF(XUP)) 75,25,75
   25    X=Z(IBUF(3,1,6))
         CALL CHECKZ(X,IERR), RETURNS(113)
         IF (.NOT.LOGRP(6)) GO TO 30
         IF (X.LE.0.) GO TO 20
         X=ALOG10(X)
   30    IF (X.LT.XMIN.OR.X.GT.XMAX) GO TO 20
         IF (X.LT.WHER) GO TO 35
         WHER=WHER+W1
         ICHAR=.TRUE.
   35    Y=Z(IBUF(3,J,I))
         CALL CHECKZ(Y,IERR), RETURNS(43)
         IF (.NOT.LOGRP(I)) GO TO 40
         IF (Y.LE.0.) GO TO 75
         Y=ALOG10(Y)
   40    IF (Y.LT.YMIN(I).OR.Y.GT.YMAX(I)) GO TO 75
         GO TO 60
C
C.....ENTER BELOW WHEN A VARIABLE GOES INFINITE OR INDEFINITE.  A
C.....SPECIAL SYMBOL IS PRINTED INDICATING THE VARIABLES DEPARTURE FROM
C.....NORMAL.
C
   43    LDIF=.TRUE.
         GO TO (45,50,55), IERR
   45    Y=YLINE(I,3)
```

```
         CALL FRSTPT (X,Y)
         CALL MXMY (IX,IY)
         IX=IX-10
         IF (IX.LE.IFIND) GO TO 60
         IFIND=IX+10
         JCHR=10H;
         IY=IY+24
         CALL PSYM (IFIND,IY,JCHR,0,1,1)
         CALL FRSTPT (X1,Y1)
         GO TO 60
   50    Y=YMAX(I)
         CALL FRSTPT (X,Y)
         CALL MXMY (IX,IY)
         IX=IX-10
         IF (IX.LE.IFPOS) GO TO 60
         IFPOS=IX+10
         JCHR=76000000000000000000B
         IY=IY-32
         CALL PSYM (IFPOS,IY,JCHR,0,1,1)
         CALL FRSTPT (X1,Y1)
         GO TO 60
   55    Y=YMIN(I)
         CALL FRSTPT (X,Y)
         CALL MXMY (IX,IY)
         IX=IX-10
         IF (IX.LE.IFNEG) GO TO 60
         IFNEG=IX+10
         JCHR=76000000000000000000B
         IY=IY+24
         CALL PSYM (IFNEG,IY,JCHR,0,1,1)
         CALL FRSTPT (X1,Y1)
C
C.....PLOTS THE ID SYMBOLS--1 SYMBOL PER DEPENDENT VARIABLE FOR EACH 1/5
C.....INCREMENT OF THE INDEPENDENT VARIABLES RANGE.
C
   60    IF (.NOT.ICHAR) GO TO 65
         CALL FRSTPT (X,Y)
         CALL MXMY (IX,IY)
         IF (IERR.EQ.2) IY=IY-24
         IX=IX+8
         IY=IY+8
         CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
         IF (ICHR.EQ.1H$) CALL SCHARS (ICHR,2,1,1H$)
         CALL PWRT (IX,IY,ICHR,1,0,0)
         CALL FRSTPT (X1,Y1)
         ICHAR=.FALSE.
C
C.....CONNECTS A VECTOR FROM THE LAST VARIABLE COORDINATES TO THE
C.....PRESENT COORDINATES.
C
   65    IF (IFIRST) GO TO 70
         IF (IERR.LE.3) GO TO 70
         CALL VECTOR (X,Y)
         X1=X
         Y1=Y
         GO TO 20
   70    X1=X
         Y1=Y
         IFIRST=.FALSE.
         CALL FRSTPT (X1,Y1)
         GO TO 20
   75 CONTINUE
C
C.....PRINT OUT PLOT TITLE AND DRAW GRID
C
      CALL SETLINE (0)
      CALL GRID (5,0,5,0)
      DO 80 M=1,74
        CALL GCHARS (ITITL,M,1,ICHR)
        IF (ICHR.EQ.1H$) CALL SCHARS (ITITL,M,1,1H )
   80 CONTINUE
        CALL OPTION (0,1,0,0,1)
        CALL PWRT (480,984,NTITL,10,1,0)
        CALL FRSTPT (600,984)
        CALL NUMBR (KNTPLT,2HI2)
        CALL PWRT (104,948,ITITL,74,1,0)
C
C.....PACKS VARIABLE IDENTIFICATION CHARACTERS FOR ONE GROUP IN A
C.....SINGLE WORD.
C
        DO 90 I=1,NGRP
          NV=NVAR(I)
          KEY(I)=BLANK
          K=0
        DO 90 J=1,NV
          CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
          K=K+1
          IF (ICHR.NE.1H$) GO TO 85
          CALL SCHARS (KEY(I),K,1,ICHR)
          K=K+1
   85     CALL SCHARS (KEY(I),K,1,ICHR)
   90   CONTINUE
C
C.....FOLLOWING PLOTS SCALES DIVISIONS FOR Y AXIS
C
        CALL OPTION (0,0,0,0,0)
      DO 110 MM=1,2
        WHER=.1
        WHER=WHER-.16
        DO 100 K=1,6
          WHER=WHER+.16
          IYCOL=WHER*1024.+8.
        IF(NGRP.GE.3) IYCOL=WHER*1024.+26.
        IF (NGRP.EQ.5) IYCOL=WHER*1024.+44.
        DO 95 I=1,NGRP
            CALL PWRT (9,IYCOL,KEY(I),5,0,0)
            CALL FRSTPT (57,IYCOL)
            CALL FNDFMT (YLINE(I,K),IFMT)
            CALL NUMBR (YLINE(I,K),IFMT)
   95     IYCOL=IYCOL-18
  100   CONTINUE
C
C.....FOLLOWING PLOTS SCALE DIVISIONS FOR X AXIS
C
        WHER=.14
        WHER=WHER-.16
        DO 105 K=1,6
          WHER=WHER+.16
          IXCOL=WHER*1024.-40.
          IF (K.EQ.1) IXCOL=WHER*1024.
          CALL FRSTPT (IXCOL,85)
          CALL FNDFMT (XLINE(K),IFMT)
          CALL NUMBR (XLINE(K),IFMT)
  105   CONTINUE
      CALL PWRT (432,50,IBUF(1,1,6),19,0,0)
  110 CONTINUE
      CALL OPTION(0,1,0,0,0)
      IF (LDIF) CALL PWRT (5,5,INFIN,80,0,0)
      GO TO 115
  113 WRITE(XUO,114) IBUF(1,1,6), IRUF(2,1,6), KNTPLT
  114 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST.//T14, 40HATTEMPTED TO
     1PLOT INDEPENDENT VARIABLE ',A10,A8, 38H' WITH AN INFINITE OR INDEF
     2INITE VALUE./T14, 13HPLOT NUMBER ',I2,  9H' IGNORED)
  115 CONTINUE
C
      END
```

Initialize and step through variable list


INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
no
PLOT TITLE AND SCALE DIVISIONS
RETURN
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
yes
PLOT SPECIAL CHARACTER
PLOT ID AND CONNECT VECTOR

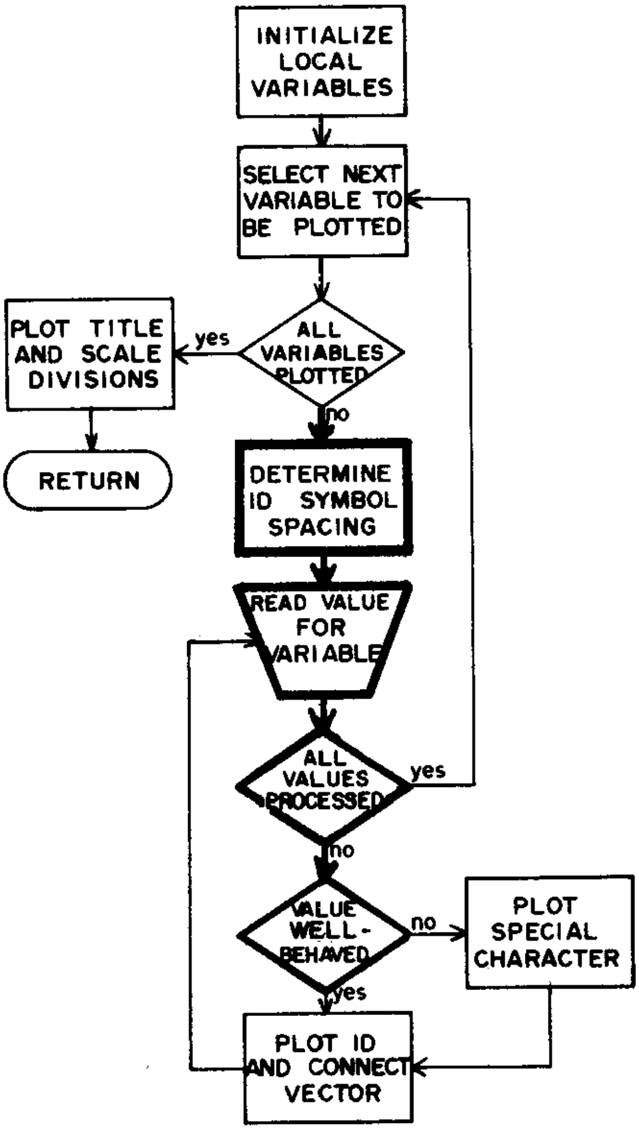
```
      OVERLAY(NEWT1,2,2)
      PROGRAM MICRO
      COMMON /XXUNITS/ XUO,XUI,XUP,XUE,XBF,XTRACE,XPLFG,FILM,NVARS
      COMMON /XTEMP/ IBUF(3,5,6),RBUF(2,6)
      COMMON /GRPHI/ KNTPLT,XLINE(6),YLINE(5,6),XMIN,XMAX,YMIN(5),YMAX(5
     1),Z(100),LOGRP(6),RANGE(6),ITITL(8),NVAR(5),NGRP
      INTEGER XUO,XUI,XUP,XUE,XBF,XTRACE
      LOGICAL FILM,LOGRP,RANGE
      DIMENSION KEY(5), INFIN(8)
      LOGICAL ICHAR,LDIF
      LOGICAL IFIRST
      DATA NTITL/10H PLOT NO. /
      DATA INFIN/10HPLOT VARIA,10HBLE WENT I,10HNFINITE OR,10H INDEFINIT
     1,10HE--CHECK G,10HRAPH FOR S,10HPECIAL CHA,10HRACTERS    /
```

```
84014         C
84015         C.....MICRO PRINTS DESIRED PLOTS ON FILM.  5 GROUPS AND A MAXIMUM OF 5
84016         C.....VARIABLES PER GROUP IS ALLOWED.  (TOTAL OF 25 VARIABLES PER PLOT.)
84017         C
84018               IF (KNTPLT.NE.0) GO TO 15
84019               CALL FLMHEAD
84020               GO TO 115
84021            15 CALL FRAME
84022               CALL OPTION (0,1,0,0,0)
84023               CALL SETLINE (1)
84024               ICHAR=.FALSE.
84025               LDIF=.FALSE.
84026               IFPOS=-11
84027               IFNEG=-11
84028               IFIND=-11
84029               BLANK=10H
84030               KNT=0
84031         C
84032         C.....THE FOLLOWING LOOPS COLLECT VALUES FOR EACH VARIABLE IN A PARTICU-
84033         C.....LAR PLOT AND PRINT EACH VALUE VERSES THE INDEPENDENT VARIABLE
84034         C.....ON A MICROFILM GRID.
84035         C
84036               DO 75 I=1,NGRP
84037                  CALL SET (.14,.94,.1,.9,XMIN,XMAX,YMIN(I),YMAX(I),1)
84038                  NV=NVAR(I)
84039               DO 75 J=1,NV
```

Line Number	Explanation
84000-84030	The overlay containing the microfilm plotting routines is called once for each plot requested (if FILM=.TRUE.). NTITL contains the plot header label. INFIN is a message issued if an indefinite or infinite value is encountered. KNTPLT is the current plot number. Before the first plot is printed, header information containing information about the plots to be printed is written onto a film frame (accomplished by general purpose subroutine FLMHEAD). FRAME advances a new frame. ØPTIØN sets the intensity and size of the characters to be printed. SETLINE sets the intensity of the lines to be plotted. ICHAR is a logical flag which controls the periodic printing of identification symbols (labeling each plotted line). LDIF controls the printing of a message if a value of a dependent variable is detected to be infinite or indefinite. IFPØS controls the printing of special characters indicating that a value or series of values are positive infinite. IFNEG controls the printing of special characters for detected negative infinite values. IFIND controls the periodic printing of characters indicating that a value is indefinite. KNT offsets the identification characters of the first five variables per plot.

Line Number	Explanation
84036-84039	Each variable is separately plotted across the entire range of the independent variable. Processing continues until all variables of each group of the current plot have been plotted. SET defines the mapping scale for the independent variable and dependent variables of the current group. (The upper, lower, left, and right bounds are defined.) NV is the number of variables in the current group of the current plot. All variables of one group are plotted; then the scale factors change and each variable of the next group is processed, etc., until all variables of the current plot have been processed.

Symbol spacing and variable values enterer


INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
PLOT TITLE AND SCALE DIVISIONS
RETURN
no
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
PLOT SPECIAL CHARACTER
yes
PLOT ID AND CONNECT VECTOR

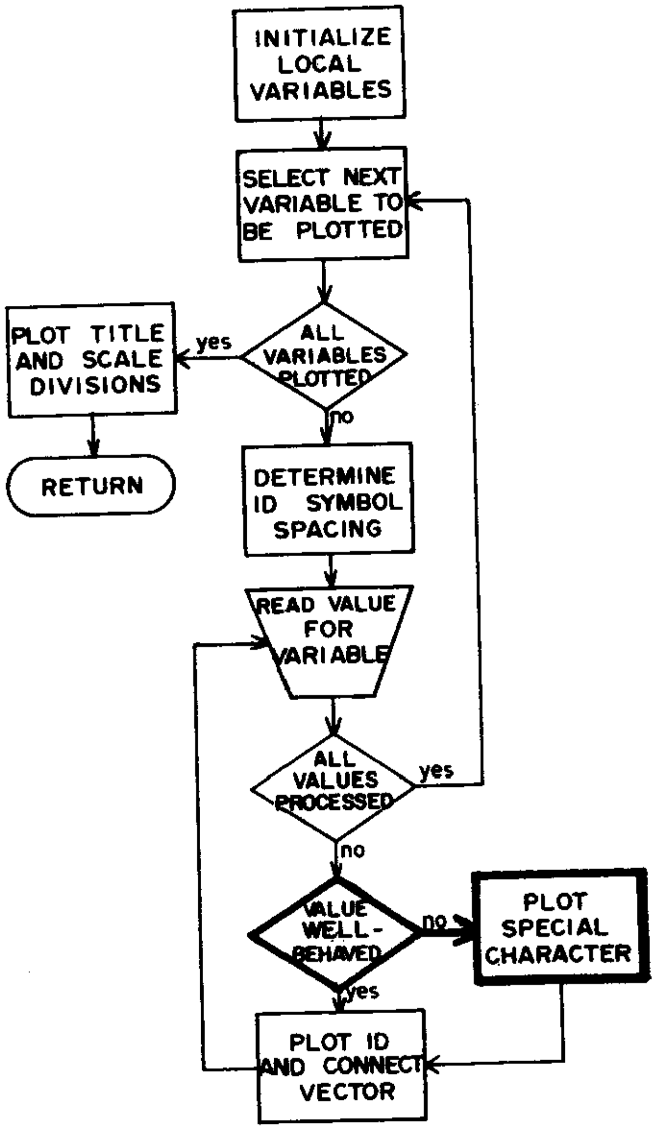
```
C
C.....VARIABLE ID SYMBOLS ARE PRINTED NEAR EACH PLOTTED VARIABLE CURVE.
C.....FOLLOWING ROUTINE 1). ARRANGES 5 SYMBOLS FOR A CURVE ACROSS THE
C.....GRID.  2). OFFSETS ID SYMBOLS OF DIFFERENT VARIABLES.
C
      KNT=KNT+1
      IF (KNT.GT.5) KNT=1
      ZKNT=KNT
      X1=XMIN
      Y1=YMIN(I)
      IFIRST=.TRUE.
      REWIND XUP
      W1=ABS(XLINE(2))
      WHER=XLINE(2)*.2*ZKNT
   20 READ (XUP) (Z(II),II=1,NVARS)
      IF (EOF(XUP)) 75,25,75
   25 X=Z(IBUF(3,1,6))
      CALL CHECKZ(X,IERR), RETURNS(113)
      IF (.NOT.LOGRP(6)) GO TO 30
```

```
84059               IF (X.LE.0.) GO TO 20
84060               X=ALOG10(X)
84061         30    IF (X.LT.XMIN.OR.X.GT.XMAX) GO TO 20
84062               IF (X.LT.WHER) GO TO 35
84063               WHER=WHER+W1
84064               ICHAR=.TRUE.
84065         35    Y=Z(IBUF(3,J,I))
84066               CALL CHECKZ(Y,IERR), RETURNS(43)
84067               IF (.NOT.LOGRP(I)) GO TO 40
84068               IF (Y.LE.0.) GO TO 75
84069               Y=ALOG10(Y)
84070         40    IF (Y.LT.YMIN(I).OR.Y.GT.YMAX(I)) GO TO 75
84071               GO TO 60
```

Line Number	Explanation
84045-84053	Each of the five equal areas of the plot grid (divided by vertical lines) is subdivided by KNT into five locations for identification symbols to be printed. One ID character for each curve is printed in each area at a given location within that area. Thus the curve associated with the first variable has its ID symbols printed in the first location of each area. The ID symbols of the second curve are printed in the second location of each area. Using this technique the ID characters are staggered across each area and are not cluttered in one spot. ZKNT is the current character location. W1 is the distance from the location in one area to the corresponding location in the next area. WHER is the distance along the independent axis from the origin of the current location. Therefore, when the value of the independent variable = WHER, the plot location has been reached and the ID character of the current curve is to be printed. IFIRST indicates that the retrieved values are from the first set of values from the value file.
84054-84055	File XUP is the value file. Each record of this file contains a value for each variable in the plot stack. The first record contains the values of the variables at time TSTRT, the second record contains values at time TSTRT+DTPR, etc., to TEND. This file is rewound and searched for each variable on the current plot. X is the retrieved value of the independent variable (take the log of the independent variable if the log is desired).
84061-84064	When the value of the independent reaches WHER, ICHAR is set (indicating that the ID character is to be plotted at the present location and WHER is incremented to point at the next area).

Line Number	Explanation
84065-84071	Y is the retrieved value of the dependent variable. CHECKZ determines whether the retrieved value is indefinite, infinite, or within range. If the value is not well behaved, control returns to a different portion of the section (RETURNS(28)). If the logs of the variables in the current group are desired (LØGRP(I)=.TRUE.), then the log of Y is assigned to variable Y. When the coordinates of a variable indicate an infinite or indefinite value, a special character is plotted.

Plot special character

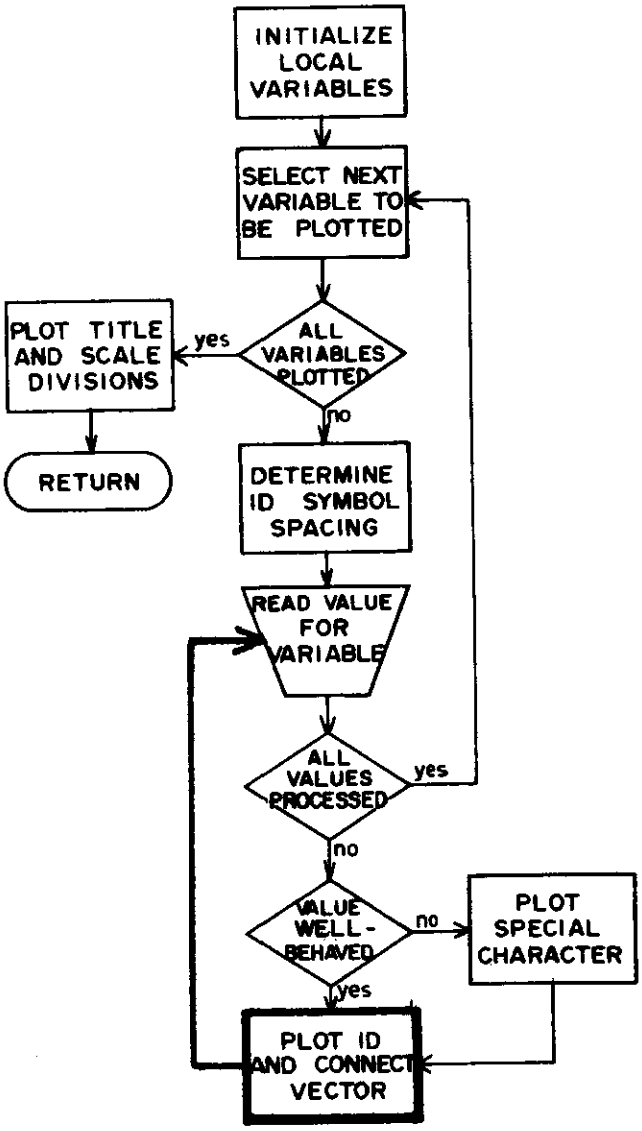

INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
PLOT TITLE AND SCALE DIVISIONS
RETURN
no
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
PLOT SPECIAL CHARACTER
yes
PLOT ID AND CONNECT VECTOR

```
C
C.....ENTER BELOW WHEN A VARIABLE GOES INFINITE OR INDEFINITE.  A
C.....SPECIAL SYMBOL IS PRINTED INDICATING THE VARIABLES DEPARTURE FROM
C.....NORMAL.
C
   43     LDIF=.TRUE.
          GO TO (45,50,55), IERR
   45     Y=YLINE(I,3)
```

```
84080                 CALL FRSTPT (X,Y)
84081                 CALL MXMY (IX,IY)
84082                 IX=IX-10
84083                 IF (IX.LE.IFIND) GO TO 60
84084                 IFIND=IX+10
84085                 JCHR=10H;
84086                 IY=IY+24
84087                 CALL PSYM (IFIND,IY,JCHR,0,1,1)
84088                 CALL FRSTPT (X1,Y1)
84089                 GO TO 60
84090           50    Y=YMAX(I)
84091                 CALL FRSTPT (X,Y)
84092                 CALL MXMY (IX,IY)
84093                 IX=IX-10
84094                 IF (IX.LE.IFPOS) GO TO 60
84095                 IFPOS=IX+10
84096                 JCHR=7600000000000000000B
84097                 IY=IY-32
84098                 CALL PSYM (IFPOS,IY,JCHR,0,1,1)
84099                 CALL FRSTPT (X1,Y1)
84100                 GO TO 60
84101           55    Y=YMIN(I)
84102                 CALL FRSTPT (X,Y)
84103                 CALL MXMY (IX,IY)
84104                 IX=IX-10
84105                 IF (IX.LE.IFNEG) GO TO 60
84106                 IFNEG=IX+10
84107                 JCHR=7600000000000000000B
84108                 IY=IY+24
84109                 CALL PSYM (IFNEG,IY,JCHR,0,1,1)
84110                 CALL FRSTPT (X1,Y1)
```

Line Number	Explanation
84077-84078	LDIF is a flag that causes a message to be printed on film if a variable value is infinite or indefinite. IERR is a value set by CHECKZ: = 1 If the value sent to CHECKZ is indefinite. = 2 If value is positive infinite. = 3 Negative infinite. = 4 Normal valued (within range and defined).
84079-84089	The current value (of dependent variable) is indefinite; Y is assigned a value and a special character is printed at that coordinate position. If the variable stays indefinite for a period of time, IFIND controls the printing of special characters spaced 10 raster points apart.
84090-84099	The current value is positive infinite; special characters are printed at the top of the plot. The current Y value is the top of the grid (YMAX).
84100-84110	The value is negative infinite; characters are plotted at the bottom of the grid area.

Plot ID symbol and connect vector

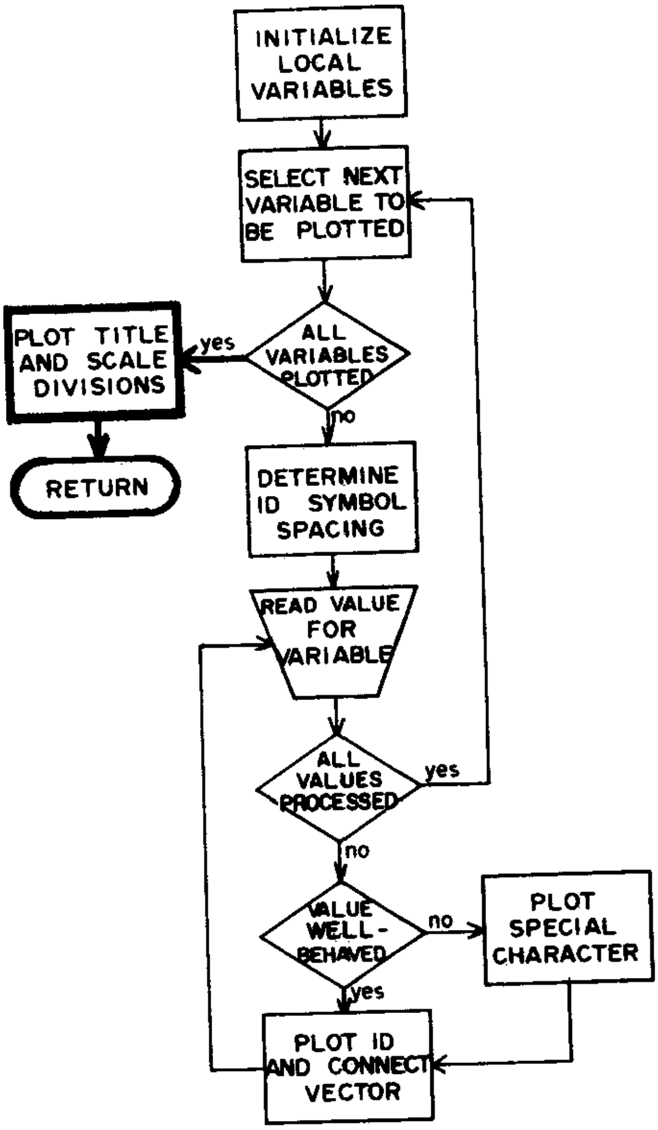

INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
PLOT TITLE AND SCALE DIVISIONS
RETURN
no
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
PLOT SPECIAL CHARACTER
yes
PLOT ID AND CONNECT VECTOR

```
C
C.....PLOTS THE ID SYMBOLS--1 SYMBOL PER DEPENDENT VARIABLE FOR EACH 1/5
C.....INCREMENT OF THE INDEPENDENT VARIABLES RANGE.
C
   60 IF (.NOT.ICHAR) GO TO 65
      CALL FRSTPT (X,Y)
      CALL MXMY (IX,IY)
      IF (IERR.EQ.2) IY=IY-24
      IX=IX+8
      IY=IY+8
      CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
      IF (ICHR.EQ.1H$) CALL SCHARS (ICHR,2,1,1H$)
      CALL PWRT (IX,IY,ICHR,1,0,0)
      CALL FRSTPT (X1,Y1)
      ICHAR=.FALSE.
C
C.....CONNECTS A VECTOR FROM THE LAST VARIABLE COORDINATES TO THE
C.....PRESENT COORDINATES.
```

```
84129        C
84130          65    IF (IFIRST) GO TO 70
84131                IF (IERR.LE.3) GO TO 70
84132                CALL VECTOR (X,Y)
84133                X1=X
84134                Y1=Y
84135                GO TO 20
84136          70    X1=X
84137                Y1=Y
84138                IFIRST=.FALSE.
84139                CALL FRSTPT (X1,Y1)
84140                GO TO 20
84141          75 CONTINUE
```

Line Number	Explanation
84114-84123	One ID character is printed between each major grid division for each plotted curve. ICHAR indicates when the proper location with respect to the independent axis is reached. FRSTPT positions the plotter head at coordinates (X,Y), the current coordinate values. MXMY returns the integer position of the plotter head in IX and IY. The integer positions are offset by eight raster points so that the character will be printed near but not on the curve it identifies PWRT prints the ID character of the current variable (in QC) at coordinates (IX,IY).
84124-84125	The plotter head is reset (positioned) at the preceding coordinates (XI,YI). The character flag is turned off until the plotting process reaches the proper location in the next vertical grid.
84130-84135	Connect a line between the previous coordinate (XI,YI) to the current coordinate (X,Y). VECTOR draws a line from the previous raster position to the position (X,Y). XI and YI are updated (the preceding coordinates) and X and Y will be filled with the next values from value file.
84135-84141	If the current coordinates are from the first set of the value file (therefore no previous coordinates) or the current coordinates are infinite or indefinite, then the plot beam is positioned to the current coordinate position and no vector is drawn.

Plot title and draw grid lines


INITIALIZE LOCAL VARIABLES
SELECT NEXT VARIABLE TO BE PLOTTED
ALL VARIABLES PLOTTED
yes
PLOT TITLE AND SCALE DIVISIONS
RETURN
no
DETERMINE ID SYMBOL SPACING
READ VALUE FOR VARIABLE
ALL VALUES PROCESSED
yes
no
VALUE WELL-BEHAVED
no
PLOT SPECIAL CHARACTER
yes
PLOT ID AND CONNECT VECTOR

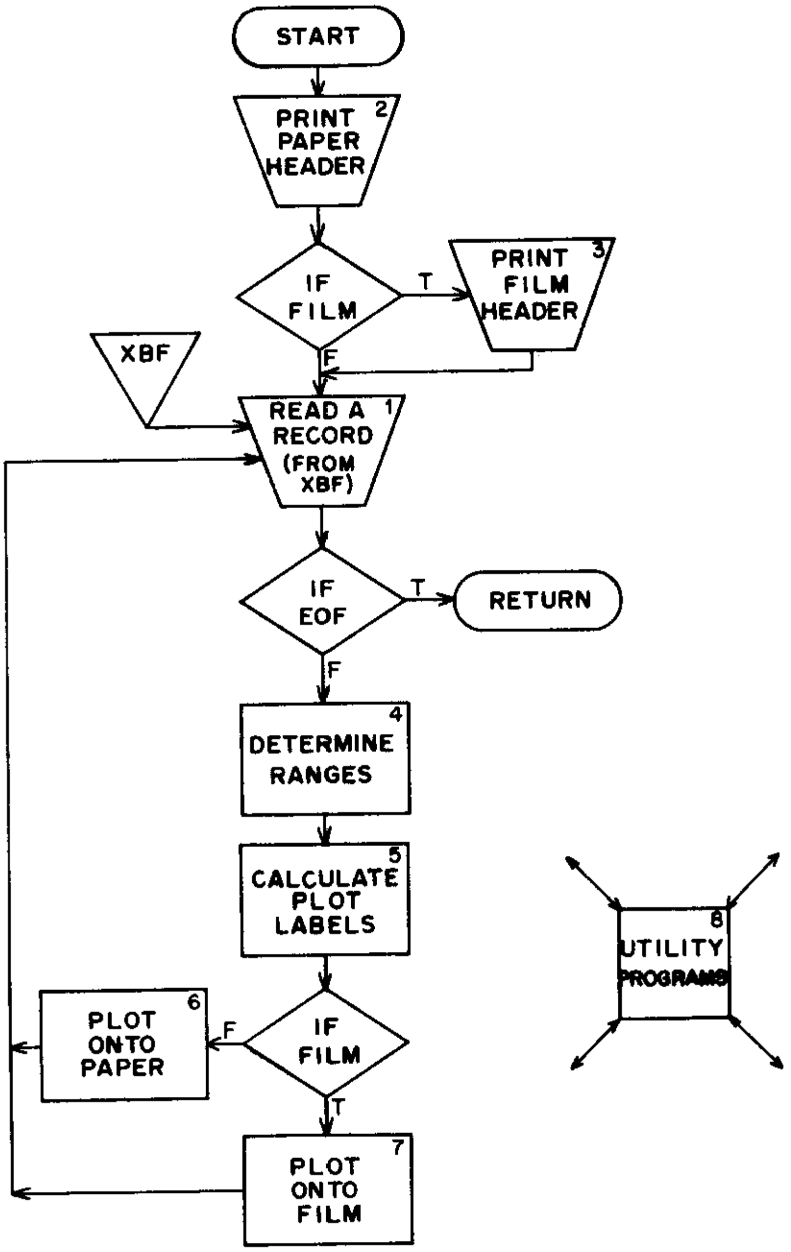
```
C
C.....PRINT OUT PLOT TITLE AND DRAW GRID
C
      CALL SETLINE (0)
      CALL GRID (5,0,5,0)
      DO 80 M=1.74
         CALL GCHARS (ITITL,M.1.ICHR)
         IF (ICHR.EQ.1H$) CALL SCHARS (ITITL,M,1,1H )
   80 CONTINUE
         CALL OPTION (0.1,0.0,1)
         CALL PWRT (480,984.NTITL,10.1.0)
         CALL FRSTPT (600,984)
         CALL NUMBR (KNTPLT.2HI2)
         CALL PWRT (104,948,ITITL,74,1,0)
C
C.....PACKS VARIABLE IDENTIFICATION CHARACTERS FOR ONE GROUP IN A
C.....SINGLE WORD.
C
```

```
84160               DO 90 I=1,NGRP
84161                  NV=NVAR(I)
84162                  KEY(I)=BLANK
84163                  K=0
84164               DO 90 J=1,NV
84165                  CALL GCHARS (IBUF(2,J,I),10,1,ICHR)
84166                  K=K+1
84167                  IF (ICHR.NE.1H$) GO TO 85
84168                  CALL SCHARS (KEY(I),K,1,ICHR)
84169                  K=K+1
84170          85      CALL SCHARS (KEY(I),K,1,ICHR)
84171          90   CONTINUE
84172       C
84173       C.....FOLLOWING PLOTS SCALES DIVISIONS FOR Y AXIS
84174       C
84175                CALL OPTION (0,0,0,0,0)
84176             DO 110 MM=1,2
84177                WHER=.1
84178                WHER=WHER-.16
84179                DO 100 K=1,6
84180                   WHER=WHER+.16
84181                   IYCOL=WHER*1024.+8.
84182                IF(NGRP.GE.3) IYCOL=WHER*1024.+26.
84183                IF (NGRP.EQ.5) IYCOL=WHER*1024.+44.
84184                DO 95 I=1,NGRP
84185                      CALL PWRT (9,IYCOL,KEY(I),5,0,0)
84186                      CALL FRSTPT (57,IYCOL)
84187                      CALL FNDFMT (YLINE(I,K),IFMT)
84188                      CALL NUMBR (YLINE(I,K),IFMT)
84189          95      IYCOL=IYCOL-18
84190         100    CONTINUE
84191       C
84192       C.....FOLLOWING PLOTS SCALE DIVISIONS FOR X AXIS
84193       C
84194                WHER=.14
84195                WHER=WHER-.16
84196                DO 105 K=1,6
84197                   WHER=WHER+.16
84198                   IXCOL=WHER*1024.-40.
84199                   IF (K.EQ.1) IXCOL=WHER*1024.
84200                   CALL FRSTPT (IXCOL,85)
84201                   CALL FNDFMT (XLINE(K),IFMT)
84202                   CALL NUMBR (XLINE(K),IFMT)
84203        105     CONTINUE
84204              CALL PWRT (432,50,IBUF(1,1,6),19,0,0)
84205        110 CONTINUE
84206              CALL OPTION(0,1,0,0,0)
84207              IF (LDIF) CALL PWRT (5,5,INFIN,80,0,0)
84208              GO TO 115
84209        113 WRITE(XUO,114) IBUF(1,1,6), IBUF(2,1,6), KNTPLT
84210        114 FORMAT (6H0*****, 21HERROR IN PLOT REQUEST,//T14, 40HATTEMPTED TO
84211             1PLOT INDEPENDENT VARIABLE ',A10,A8, 38H) WITH AN INFINITE OR INDEF
```

Line Number	Explanation
84144-84155	The title of the plot and the grid is drawn. GRID draws in vertical and horizontal grid lines defining the mapping area defined by SET. The remaining operations are executed twice to achieve a darker print.
84160-84170	KEY(I) is filled with a list of all ID characters for variables in the Ith group.

Line Number	Explanation
84175-84190	The appropriate scale division and the list of ID characters for the first group are printed to the left of each horizontal grid line. The label of each later group is printed below the label of the previous group. WHER is the Y value at a particular grid line. IYCØL is the raster point position where the label of a particular group is to be plotted. (At any of the six grid lines, the label of group 1 is plotted at (9,IYCØL), group 2 at (9,IYCØL-18), etc.). FNDFMT determines the most significant 10-character format for a particular scale division. (RECALL. YLINE(I,K) contains the Kth scale division of the Ith group.)
84194-84202	The scale divisions of the independent variable are printed below the corresponding vertical grid line.
84204-84208	The name of the independent variable is printed below the X axis. If an indefinite or infinite value is encountered, a message is printed.

4.8. *Utility Routines*



Overview

The three utility routines for this chapter are:

(1) RØUND determines an appropriate rounded scaling for a graph of a function whose values range from the MIN to MAX, given a MIN and MAX value.

(2) FNDFMT determines the "best" 10-digit format for printout of the value, given a value, FNDFMT.

(3) CHECKZ determines whether a value is infinite, indefinite, or normally valued.

```
      SUBROUTINE ROUND (ZMIN,ZMAX,RNZMIN,RNZMAX,MAJDIV)
C
C.....GIVEN ZMIN AND ZMAX, THIS SUBROUTINE DETERMINES AN APPROPRIATE
C.....SCALING FOR A GRAPH OF A FUNCTION WHOSE VALUES RANGE FROM ZMIN
C.....TO ZMAX.
C.....RNZMIN AND RNZMAX ARE THE EXTREME VALUES OF THE GRAPH.
C.....MAJDIV IS THE NUMBER OF MAJOR DIVISIONS OF THE GRAPH.
C.....THE CASE WHERE ZMIN = ZMAX IS TREATED SEPARATELY.
C
      IF (ZMIN.NE.ZMAX) GO TO 45
      IF (ZMAX.NE.0.) GO TO 15
      RNZMIN=-1.
      RNZMAX=1.
      M=2
      GO TO 80
C
C.....SCALE Z UNTIL THE FIRST SIGNIFICANT DIGIT IS IN THE THOUSANDS
C.....PLACE AND ROUND AT THE DECIMAL PLACE.
C
   15 Z=ZMAX
      I=0
   20 IF (Z.GE.1000.) GO TO 25
      Z=Z*10.
      I=I-1
      GO TO 20
   25 IF (Z.LT.10000.) GO TO 30
      Z=Z/10.
      I=I+1
      GO TO 25
   30 Z=INT(Z+.5)
C
C.....DETERMINE THE NUMBER OF SIGNIFICANT DIGITS IN Z, TRUNCATE THE
C.....LAST ONE, AND USE THIS NUMBER AS A BASIS FOR SETTING THE
C.....GRAPH VALUES.
C
      Z=Z/10.
      I=I+1
   35 ZRND=INT(Z)
      IF (ZRND.NE.Z) GO TO 40
      Z=Z/10.
      I=I+1
      GO TO 35
   40 IF (Z.GE.0.) RNZMIN=ZRND-1.
      IF (Z.LT.0.) RNZMIN=ZRND-2.
      RNZMAX=RNZMIN+3.
C
C.....RESTORE THE NUMBERS TO THE ORIGINAL MAGNITUDE.
C
      RNZMIN=RNZMIN*10.**I
      RNZMAX=RNZMAX*10.**I
      M=3
      GO TO 80
C
C.....IN THE GENERAL CASE THE DIFFERENCE, ZMAX-ZMIN, IS TRUNCATED TO
C.....THE FIRST SIGNIFICANT DIGIT AND ENLARGED IF NECESSARY TO
C.....ENCOMPASS THE ENTIRE RANGE, ZMIN TO ZMAX.
C
   45 VAR=ZMAX-ZMIN
      I=0.
   50 IF (VAR.GE.1.) GO TO 55
      VAR=VAR*10.
      I=I-1
      GO TO 50
```

```
81063         55 IF (VAR.LT.10.) GO TO 60  .
81064            VAR=VAR/10.
81065            I=I+1
81066            GO TO 55
81067         60 RNVAR=INT(VAR)
81068            IF (RNVAR.EQ.VAR) GO TO 65
81069            IF (VAR.GT.0.) RNVAR=RNVAR+1.
81070            IF (VAR.LT.0.) RNVAR=RNVAR-1.
81071      C
81072      C.....TRUNCATE ZMIN AT THE SAME DECIMAL PLACE AS THE DIFFERENCE,
81073      C.....ZMAX-ZMIN, WAS TRUNCATED AND LOWER THIS VALUE IF NECESSARY
81074      C.....TO INSURE THAT IT IS LESS THAN ZMIN. THIS VALUE IS USED FOR
81075      C.....RNZMIN AND THE TRUNCATED DIFFERENCE, RNVAR, IS ADDED TO
81076      C.....OBTAIN RNZMAX (RNVAR IS ENLARGED IF NECESSARY TO INSURE
81077      C.....INCLUSION OF THE ENTIRE INTERVAL).
81078      C
81079         65 Z=ZMIN*10.**(-I)
81080            ZZ=ZMAX*10.**(-I)
81081            ZRND=INT(Z)
81082            IF (VAR.LT.0) GO TO 70
81083            IF (Z.GE.0.) RNZMIN=ZRND
81084            IF (Z.LT.0.) RNZMIN=ZRND-1.
81085            IF (RNZMIN+RNVAR.LT.ZZ) RNVAR=RNVAR+1.
81086            GO TO 75
81087         70 IF (Z.GT.0.) RNZMIN=ZRND+1.
81088            IF (Z.LE.0.) RNZMIN=ZRND
81089            IF (RNZMIN+RNVAR.GT.ZZ) RNVAR=RNVAR-1.
81090         75 RNZMAX=RNZMIN+RNVAR
81091      C
81092      C.....RESTORE THE NUMBERS TO THE ORIGINAL MAGNITUDE
81093      C
81094            RNZMIN=RNZMIN*10.**I
81095            RNZMAX=RNZMAX*10.**I
81096            M=AINT(ABS(RNVAR))
81097      C
81098      C.....ESTABLISH AN APPROPRIATE NUMBER OF MAJOR DIVISIONS
81099      C
81100         80 IF (M.LT.3.OR.M.EQ.5) MAJDIV=10
81101            IF (M.EQ.3.OR.M.EQ.4) MAJDIV=2*M
81102            IF (M.GT.5) MAJDIV=M
81103            RETURN
81104      C
81105            END
```

Line Number	Explanation
81000	ZMIN and ZMAX are the input arguments to RØUND (the minimum and maximum value to be plotted). RNZMIN and RNZMAX are the rounded values of ZMIN and ZMAX calculated by RØUND. MAJDIV is the number of major divisions the graph should have.
81009-81014	The input minimum and maximum values are both zero.
81019-81029	Input minimum and maximum are equal, but not zero. The maximum is divided (or multiplied) until the most significant digit is in the thousands place. Z is the four-digit representation of the maximum value (1000.<Z<10000.). I is the number of divisions (multiplications) by 10 of maximum necessary to produce Z.

Line Number	Explanation
81034-81041	Determine the number of significant digits in Z. ZRND contains the significant digits of Z less one.
81042-81044	RNZMIN and RNZMAX are assigned values, determined by the truncated significant digits of Z (will contain 1, 2, or 3 significant digits).
81048-81051	The rounded numbers are restored to the magnitude of the original maximum and minimum values.
81057-81070	The general case (the input maximum value is not equal to the input minimum value) is processed. The difference (ZMAX-ZMIN) is truncated to the most significant digit. VAR is a one-digit representation of ZMAX-ZMIN (1.<var<10.). I is the number of divisions (multiplications) necessary to produce VAR. RNVAR is a one-digit number that represents the entire range of values from ZMIN to ZMAX.
81079-81090	Truncate ZMIN and ZMAX by the amount ZMAX-ZMIN was truncated. Z is the truncated minimum. ZZ is the truncated maximum. ZRND, truncated Z (any fractional part of Z is removed); ZRND is used as the rounded minimum value. The rounded maximum value becomes the rounded minimum value plus the one-digit representation of the range of values. The rounded numbers are restored to the magnitude of the input arguments.
81096-81105	M is the integer representation of the range between the rounded maximum and minimum values. MAJDIV is the number of major divisions that are determined by M (6≤MAJDIV≤10).

```
82000                 IDENT CHECKZ
82001                 LIST  -R,-G
82002                 ENTRY CHECKZ
82003    .      *
82004           **********CHECKZ DETERMINES WHETHER AN ARGUMENT IS INDEFINITE, INFINITE,
82005           *         OR IS WITHIN BOUNDS.  IT RETURNS A VALUE = 1,2,3,4 FOR EITHER
82006           *         INDEFINITE, + INFINITE, - INFINITE, OR NORMAL VALUED VARIABLES
82007           *
82008           CHECKZ    BSS   1
82009                     SA2   X1               .X1=ADDRESS OF ARG, X2=CONTENTS OF ARG
82010                     SX6   2                .X6=2 FOR POSITIVE INFINITY
82011                     OR    X2,INFIN         .JUMP TO INFIN IF X2 IS POSITIVE
82012                     ID    X2,INDEF         .JUMP TO INDEF IF X2 IS INDEFINITE
82013                     SX6   4                .X6=4 FOR NORMAL VALUES
```

```
82014                     EQ      ENDIT
82015          INFIN      PL      X2,ENDIT         .IF XI IS POSITIVE GO TO ENDIT
82016                     SX6     3                .X6=3 FOR NEGATIVE INFINITY
82017                     EQ      ENDIT
82018          INDEF      SX6     1                .X6=1 FOR INDEFINITE
82019          ENDIT      SA2     A1+1             .X2 = ADDRESS OF 2ND ARG
82020                     SA6     X2               .X6=CONTENTS OF 2ND ARG = 1,2,3, OR4
82021                     SB2     X6               .B2=X6
82022                     SB2     B2-4
82023                     EQ      B2,CHECKZ        .IFB2=0, THEN NORMAL BOUNDS VAR AND RETURN
82024                     SA2     A1+3             .X2=ADDRESS OF ABNORMAL RETURN LOCATION
82025                     SB1     X2               .B1=ADDRESS OF ABNORMAL RETURN LOCATION
82026                     JP      B1               .JUMP TO LOCATION SPECIFIED BY RETURNS(MM)
82027    '                END
```

Line Number	Explanation
82000-82027	CHECKZ(i) determines whether the first argument is infinite, indefinite, or normal valued, (ii) sets IERR=1 If the argument is indefinite (see a call to CHECKZ). =2 For positive infinite. =3 For negative infinite. =4 If a normal value. and (iii) executes an abnormal return jump if the argument value is abnormal.

```
86000                     IDENT FNDFMT
86001                     LIST  -R,-G
86002                     ENTRY FNDFMT
86003          *
86004          **********FNDFMT DETERMINES THE MOST ACCURATE 10 FIELD LENGTH FORMAT
86005          *          FOR A GIVEN NUMBER
86006          *
86007          FNDFMT     BSS     1
86008                     SA2     X1               .X1=ADDRESS 1ST ARG, X2=C(X1)
86009                     BX0     X2               .X0=X2
86010                     AX0     60               .EXTEND SIGN BIT ACROSS X0
86011                     BX0     X2-X0            .X0=ABSOLUTE VALUE OF X2
86012                     SA3     SMALL            .X3=.00001
86013                     RX3     X3-X0
86014                     PL      X3,EFMT          .FORM E FORMAT IF ARG LE .00001
86015                     SA3     LARGE            .X3=1.E8
86016                     RX3     X0-X3
86017                     PL      X3,EFMT          .FORM E FORMAT IF ARG GE 1.E8
86018                     UX3     B1,X0            .B1=EXP-2000B, X3=COEF
86019                     SX4     B1+48            .X4=EXP WITH DECIMAL PT LEFT OF COEF
86020                     PX5     B0,X4            .X5=FLOAT VALUE OF X4
86021                     SA3     LOG2             .X3=LOG10(2)
86022                     RX6     X3*X5            .X6=BASE2 EXP * LOG10(2)
86023                     UX7     B7,X6            .B7=EXP-2000B, X7=COEF
86024                     LX7     B7,X7            .X7=INTEGER X6
86025                     SB3     X7               .B3=          INTEGER X6
86026                     PL      B3,CALC
```

```
86027                  SB3   B0
86028        CALC      SA3   B3+POWERS       .X3=1.EX WHERE X=INTEGER(EXP*LOG10(2)
86029                  RX4   X0/X3           .X4=ABS(X2)/1.EX
86030                  SA5   POWERS          .X5=1.
86031                  RX6   X4-X5           .ABS(X2)/1.EX -1.
86032                  PL    X6,FFMT         .X4-X5 GE 1. THEN 1.EX IS CORRECT EXT OF N
86033                  SB3   B3-1
86034        FFMT      SB4   B3-42B          .FORM HOLLERITH BIAS
86035                  SB4   -B4             .B4=NO DIGITS RT OF DECIMAL POINT
86036                  SA5   =06343357005555555558  .XK=F10. (HOLLERITH)
86037                  SX3   B4              .X3=B4
86038                  LX3   30
86039                  RX6   X5+X3           .X6=F10.X WHERE X GE 0 AND LT 9
86040        STOP      SA1   A1+1            .X1=RETURN ARG ADDRESS
86041                  SA6   X1              .X6 IS STORED AT C(X1)
86042                  EQ    FNDFMT          .RETURN TO ENTRY LINE
86043        EFMT      SA5   =05343357365555555558   .X5=E10.3  (HOLLERITH)
86044                  RX6   X5              .X6=E10.3
86045                  EQ    STOP
86046        LOG2      DATA  .301029957
86047        SMALL     DATA  1.E-5
86048        LARGE     DATA  1.E8
86049        POWERS    DATA  1.,1.E1,1.E2,1.E3,1.E4,1.E5,1.E6,1.E7,1.E8
86050                  END
```

Line Number	Explanation
	FNDFMT determines the best 10-digit format for the value of the first argument and returns this format via the second argument (see a call to FNDFMT). An E format is returned if arguments ≤.00001 or argument ≥ 1.E8. Otherwise, an F format is constructed which will preserve the greatest number of significant digits of the argument.
86007-86011	X0 contains the absolute value of the input argument.
86012-86014	An E format (E10.3) is returned if the number is less than .00001.
86015-86017	An E format is returned if the number (X0) is greater than 10^8.
86018-86026	The base 10 exponent of the number is calculated by multiplying the base 2 exponent by the log of 2. B3 is the base 10 exponent of the number (if the decimal point were to the right of the most significant digit).
86027	If the exponent is less than zero (i.e., a negative exponent), it is assumed zero (B3=0).
86028-86033	The absolute value of the input argument is divided by the calculated exponent times 10. If the resulting value is greater than or equal to 1, then the exponent is correct. Otherwise, subtract 1 from the exponent.

Line Number	Explanation
86034-86039	The BCD value of the number of digits right of decimal is calculated (B4). The digits right of point equal 8 - base 10 exponent. This number is filled into a format statement, F10.X, where X is the hollerith representation of the number of digits.
86040-86042	Store the format at the return argument address.
86043-86046	X5=E10.3 format. The number is too large or small to be represented by a F10.X format.

LITERATURE CITED

Gustafson, J. D., and G. S. Innis. 1973. SIMCOMP version 3.0 user's manual. US/IBP Grassland Biome Tech. Rep. No. 218. Colorado State Univ., Fort Collins. 149 p.