2

AAEC/E431

AAEC/E431

# AUSTRALIAN ATOMIC ENERGY COMMISSION

## RESEARCH ESTABLISHMENT

## LUCAS HEIGHTS

# SKAN - A FREE INPUT LABELLED OUTPUT VARIABLE DIMENSIONING ROUTINE FOR THE IBM360 COMPUTER

by

J.P. POLLARD

January 1978

ISBN 0 642 59633 6

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

# SKAN - A FREE INPUT LABELLED OUTPUT VARIABLE DIMENSIONING ROUTINE FOR THE IBM360 COMPUTER

by

J. P. POLLARD

ABSTRACT

A keyword directed free format input routine, which has been used in large neutronics codes, is described. The routine also sets up variable dimension addresses for use with side entry calls. In addition, as a program debugging aid, a labelled printer dump facility is provided. The suite of routines has been used on IBM360 computers.

National Library of Australia card number and ISBN 0 642 59633 6

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

IBM COMPUTERS; COMPUTER CODES; PROGRAMMING; REACTOR KINETICS

CONTENTS

## 1. INTRODUCTION

This report describes a suite of SKAN routines for the IBM360 which have been used, as the earlier version DTAV, in several modules of the AUS neutronics scheme [Robinson 1975]. The suite was based on three requirements outlined below.

(1) To ease the task of code authors desiring to use keyword data, *e.g.*
A=0.,1.,2.,3.

The SKAN input suite is a control program for the subroutine SCAN [Bennett & Pollard 1967]. SCAN readily enables numeric and alphanumeric data to be read from a card using the basic principle 'if you can read it then so should the routine'. For example

(i) 0.1 0.2 0.3 0. 0.
(ii) 1.E-1 2.-1 30.-2 0.E0 0.E+0
(iii) 0.1,0.2$0.3, 0.-0 0.+0 (special characters except as in (iv) are ignored)
(iv) 0.1(0.1)0.3,2*0. (*i.e.* a(in steps of b)to c, n*of these numbers)

punched freely on a card all represent the same numeric data. SKAN (keyword SCAN) in addition, enables keywords to control data storage so that the order of data becomes less important. For example, A=0.1,0.2,0.3,2*0 would pick up the same data as in the example above but would store them in elements A(1) to A(5) respectively.

(2) To ease program and data checkout by providing labelled printer dumps.

The DTADMP printer dump uses names and address of variables from the SKAN table to label output. The extent of dump can readily be changed during run time to suit the needs of the code user.

(3) To facilitate variable dimensioning features simply by setting up addresses for arrays with non-fixed dimensions, *e.g.*
... A(MAXX,MAXY,MAXZ,MAXG) .

Arrays with variable dimensions are assigned addresses as part of COMMON. The COMMON may be (a) extensive and preassigned, *e.g.*

```
COMMON STORAGE(100000)
```

or (b) minimal for various fixed arrays, *e.g.*

```
COMMON N,MAXX,...,WK(100),S,
```

the remaining storage being procured dynamically using the VARRAY subroutine of Cox & Pollard (Appendix A). Our variable dimensioned quantities for (a) and (b) would then be stored *as if we could* write in FORTRAN...

```
(a) COMMON N,MAXX,...,WK(100),S
    COMMON A(MAXX,MAXY,MAXZ,MAXG),...
```

or (b) COMMON N,MAXX,...,WK(100),S

COMMON HOLE(big enough to straddle program)

COMMON A(MAXX,MAXY,MAXZ,MAXG),...

Of course, SKAN is much easier to use when variable dimensions are not required; such a situation is described in Section 2. However, SKAN was written basically for large code authors who would overlay the extensive SKAN (and SCAN) routines with their own computational segments.

## 2. A SIMPLE CASE

To show the reader the data requirement, we begin with a simple program to input (i) array A and vector x, (ii) case number and then (iii) to calculate c=Ax, and (iv) to print c. The calculation begins after the case number is supplied. For subsequent cases we need not input both A and x. The full data would then be as follows (using catalogued procedures available at the AAEC Research Establishment). It is noted that the user's COMMON is supplied in both the FORTRAN compiler input and the job input, as each prepares its own independent table.

```
//Job card
// EXEC FORTGCLG,PLIB='AUS.POW3D'
//FORT.SYSIN DD *
C     A SIMPLE PROGRAM TO CALC C=AX
      COMMON NOCASE,A(4,4),X(4),C(4)
      CALL PRELUD(NOCASE,1,4,IC) - this sets up SKAN table from data on unit 4
    1 CALL SKAN(0,1,IC,1,NK)      - keyword read for data from unit 1
      IF(NK.NE.1)GO TO 1
C     ONE WORD PICKED UP SO MUST BE CASE NUMBER
      DO 2 I=1,4
      W=0.
      DO 3 J=1,4
    3 W=W+A(I,J)*X(J)
    2 C(I)=W
      WRITE(3,4)NOCASE,C
    4 FORMAT('bCASEbNO=',I4,'bC=',1P4E15.6) - b=blank
      GO TO 1
      END
/*
//GO.FT04F001 DD DSN=AUS.JPPSIMP,DISP=OLD
//GO.SYSIN DD *
      $COMMON                    - user's COMMON is read onto unit 4
```

```
      COMMON NOCASE,A(4,4),X(4),C(4) - if these data are not given then
      $                                previous unit 4 data are used
      END                            |
                                     | - terminal indicators
      $END                           |
* DATA FOLLOWS                         - an * in column 1 designates
*                                      - a comment card
A=4*1,4*2,4*3,4*4 X=0.1(0.1)0.4        - array data are given by columns
NOCASE=1
X=1.(2.)5.,8. NOCASE=2 ETC.            - data are normally columns 1 to 72
/*
//
```

From the example, note

(1) CALL PRELUD...

whose job it is to transfer

$COMMON ... $END

data (if given) from unit 1 to unit 4 and prepare the SKAN table (later also to establish variable dimensions). Note also that the statement

(2) CALL SKAN

which reads one (only) keyword and its data (hence the loop), is called repeatedly. The rather unusual way of terminating input when only one word is read is given here purely for simplicity of illustration. Normally, certain data options are designated as needing subsequent computation using the $MOD... feature (Section 3.2). Keywords such as START and STOP may be added to the SKAN table control to suit the requirements, and these typically START the calculation and STOP the run. Let us now look more deeply into PRELUD and SKAN.

## 2.1 PRELUD

The calling sequence is

CALL PRELUD(MIST,IT,ID,IC)

where MIST = first word of COMMON,

IT = input data unit number, probably 1,

ID = unit on which to find $COMMON...$END data

- PRELUD also transfers leading data from unit IT onto unit ID if it begins $COMMON (columns 7 to 13 of a card) and terminates $END (columns 7 to 10 of a card),

IC = SKAN (and SCAN) card column pointer which must be passed on from one routine call to the next as in Section 2.2.

Here (as elsewhere) input variables are underlined and output variables are overlined.

The basic structure of $COMMON data (to be described more fully in Section 7) are

```
            ⎧ $COMMON
block 1     ⎨ COMMON data including DIMENSION,REAL*8,INTEGER
            ⎪ :
            ⎩ $

blocks 2,3&4 ⎧ Data for variable dimensions go here if required
             ⎩ :

            ⎧ Standard data to be taken as default (all of COMMON is first
block 5     ⎨ zeroised), e.g. ERR=1.-4,MAX=100
            ⎪ :
            ⎩ END    - terminal indicator of block 5
block 6       $END   - terminal indicator of the lot.
```

Since the standard data may change from time to time, and later comparison of runs is made easier if the data are listed, the keyword PRINTABLE sets the block 5 data to print. In addition, if the keyword is repeated, the SKAN keyword table is also printed. Regardless of the print option set here, the user data supplied as input are always listed although such input may be turned off (see Section 3.4).

**NOTE** A variable or keyword name terminates at the last card column and must not continue from the end of one card to the beginning of the next.

## 2.2 SKAN

The calling sequence for keyword use is

CALL SKAN(0,IT,IC,NOP,NK) , where

IT = input data unit number, probably 1,

IC = card column pointer on exit, *e.g.* for a card punched from column 1 A=1.b gives IC=5, (the user may call in a fresh card with IC=73)

NOP = 
- 1 - seek next keyword and data including possible 'filling'
- 0 - seek next keyword only and don't read trailing data
- -1 - seek next keyword and data without 'filling' , and

NK = number of *4 words of core filled by data *as supplied e.g.* A=1.,2.would give NK=2 (or 4 if A is REAL*8) .

When data are undersupplied, *e.g.*

with COMMON A(10),B(10)

and A=3*1,3*2 B=10*3

then four elements of A (A(7)-A(10)) would not be filled (with zeros) unless NOP=1 and certain data options are met (Section 3.4). Normally filling does not take place unless set so that, for the simplest use, NOP=1 and NOP=-1 are effectively equivalent.

In the example above, A is designated the current keyword and B the interruptive keyword. Attributes of these keywords can be determined from a labelled COMMON area

COMMON/SKANLK/LK(5),KOP,LINK[,LAST,NSKIP]

where we would have

| | | |
|---|---|---|
| LK(1-2)='Abbbbbbb', | | name |
| LK(3) | = 1 | , *4 address relative to 1st COMMON |
| LK(4) | =10 | , *4 length |
| LK(5) | = 2 | , mode (1=INTEGER,2=REAL*4,3=REAL*8,4=alphanumeric) |
| KOP | = 0 | , option (set different from 0 with $MOD - Section 3.2) |
| LINK | = 5 | , table position of interruptive keyword passed from one SKAN call to the next (the table contains three preset keywords - see Section 3.1) |

and LAST and NSKIP (normally not required) are table position of current keyword and number of data words skipped over in locating current keyword respectively. LINK is passed from each SKAN call to the next.

The statement

CALL SKAN(0,IT,IC,0,NK)

is used to seek out the next keyword in the input data stream (without reading any trailing data). After a data inconsistency in the program, it may be decided to use the above feature to 'thumb through' to the next problem starting, say, with the keyword BEGIN. A following call with

CALL SKAN(0,IT,IC,1,NK)

will pick up from the last keyword (BEGIN in our example). However, memory of the keyword may be flushed out with the simple link destroying statement

LINK=0 .

In any case when NOP=0 is used, SKAN will not move away from a keyword unless the link is destroyed as indicated.

## 3. THE SIMPLE CASE MODIFIED

Let us see what modifications are required for the simple case described in Section 2 if we want the calculation to begin with the keyword START and the run to terminate with the keyword STOP. The calculation is now done in double precision. The following data are required:

```
//Job card
// EXEC FORTGCLG,PLIB='AUS.POW3D'
//FORT.SYSIN DD *
C     THE SIMPLE CASE MODIFIED
      COMMON/SKANLK/LK(5),KOP,LINK - note the labelled COMMON
      DIMENSION C(4)
      REAL*8 A,X(4),C              - not necessary to present this way
      COMMON CASE(2),A(4,4),X,C
      CALL PRELUD(CASE,1,4,IC)
    1 CALL SKAN(0,1,IC,1,NK)
      IF(KOP.EQ.0)GO TO 1
      GO TO (11,12),KOP
      STOP                         - end of file would STOP here as it
                                     returns KOP=9990
C     START
   11 DO 2 I=1,4
      W=0.
      DO 3 J=1,4
    3 W=W+A(I,J)*X(J)
    2 C(I)=W
      WRITE(3,4)CASE,C             - CASE is set to alpha with $MOD
    4 FORMAT(1X,2A4,',bC=',1P4E15.6)
      GO TO 1
C     STOP
   12 WRITE(3,5)
    5 FORMAT('0STOP RUN')
      STOP
      END
/*
//GO.FT04F001  DD DSN=AUS.JPPSIMPA,DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(TRK,(2,2)),
//             DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//GO.SYSIN DD *
      $COMMON
      DIMENSION C(4)
      REAL*8 A,X(4),C
      COMMON CASE(2),A(4,4),X,C
      $
```

```
$MOD(START,,,,1),(STOP,,,,2),(CASE,,,4,)$
PRINTABLE
*STANDARD DATA
X=4*1 CASE=THISJOB
END
      $END                          - note, still columns 7 to 10
*DATA FOLLOWS
A=4*1,4*2,4*3,4*4 X=0.1(0.1)0.4 CASE=AX START
X#C    CASE=AAX    START ##COMMENTS HERE
STOP
/*
//
```

From the example, note the use of

(1) GO TO (11,12),KOP

to change the program flow following keywords START and STOP respectively,

(2) $MOD...$

data which extend (or modify) the SKAN keyword table and

(3) # ,

which is used to extract data from keyword storage, etc.

### 3.1 KOP Feature

The general philosophy behind SKAN is that there is a bulk of trivial data (MAX=100 for example) which need no computation to follow - they simply set up part of the data requirement. For this type of data the default value

KOP=0

is returned from SKAN, which needs no special information beyond $COMMON...$. Other data exist, however, which require some, or a lot, of computation to follow. For example, it may be necessary to set up some program 'flags' following data, say

OUTPUT PRINT,AUS 10,

or we may perhaps want to begin an extensive calculation of neutron diffusion following data, say

START .

When computation is to follow (or a variable is alphanumeric), it is necessary to supply $MOD...$ data to extend or modify the SKAN table prepared from $COMMON...$ information. Basically the keywords are given option numbers, KOP=1,2,..., so that a simple computed GO TO can be used to separate out the different requirements following detection of different data.

The SKAN table, after PRELUD use, contains three built-in leading entries ('reserved keywords')

SKANEXTV - used by SKAN for normal SCAN use (Section 4)

END - which returns KOP=9993

ENDOFILE - which is set up by an end of file condition and which returns KOP=9990

so that in this example an end of file (or END) will cause a program STOP.

3.2 $MOD Feature

The $MOD data extend or modify the basic SKAN keyword table. The data consist of possible successions of modifications enclosed in parenthesis specified between $MOD and $. A single modification consists of the data (following the style of LK(1) to LK(6) of Section 2.2)

(NAME,[IADR],[LGH],[MODE],[KOP])

where NAME = a 1 to 8 alphanumeric character keyword name (may be a variable already in the table)

IADR = { *4 address relative to first COMMON;
*or* keyword name, in which case its address is used;
*or* [not given], in which case the entry is the address already given if NAME is a modification or 0 if NAME is an extension.

LGH, MODE and KOP are similar to IADR but for *4 length, mode (1=INTEGER,2=REAL*4,3=REAL*8,4=alphanumeric) and option respectively. For example

(START,,,,1)

would generate a new table entry of six words thus

'STAR','Tbbb',0,0,0,1 (and a KOP of 1 would be returned from SKAN following the keyword START)

and (CASE,,,4,) or (CASE,,,4)

would give the table modification

'CASE','bbbb',1,2,4,0 (and here SKAN would return alphanumeric data of one to eight alphanumeric characters filling 2,*4 words).

The above description gives enough detail of the $MOD feature for most uses (at least of the simple type). For more advanced use, however, the last three quantities can be given alternatively, thus

LGH=#

in which case the length attribute of the last given keyword is used; or

LGH=#A

in which case the length attribute of A is used. For example we could have

(START,CASE,#,#,1)

which is equivalent to

(START,1,2,4,1) - provided (START,...) follows (CASE,...)

Our input stream data could then contain

START MOATA,

which, following a SKAN call, would result in CASE='MOATAbbb' being stored and a KOP of 1 being returned.

It should be noted that, for example, using the data

$MOD(ARRAY,A,LA,MA,KA)$,

where ARRAY is to use the first LA words of A (say dimensioned A(100)), with LA(length), MA(mode) and KA(option) being stored in COMMON, then run time values are used. That is, if at any stage in the program we set

LA=10

then the next data for ARRAY will result in up to ten elements being stored in A. What happens is that LA,MA and KA are entered in the SKAN table as negative addresses of LA, *etc.* relative to beginning of COMMON and are treated as being indirect addresses. On the other hand, A is a direct address although run time indirect addressing is possible using, for example, #IA (see Section 7.4.2 for details).

3.3 <u># Feature</u>

We have already seen the use of # as a part of the $MOD data (Section 3.2). When # is encountered in normal keyword input (not part of $MOD data), there are the three meanings:

(1) #3

which increments the storage counter by three (or six for REAL*8 and alphanumeric data) so that three words of the keyword data are unaltered.

<u>Example</u>

In our modified case, if after

X=0.1(0.1)0.4,

we use

X#2,2*0.5,

then the vector X will contain the 4 values 0.1,0.2(unaltered original numbers), 0.5 and 0.5 (modified data).

(2) #KEY

which takes its data from the keyword KEY (and if not exhausted by this continues to read more).

Example

X#C

will result in the vector X being filled with numbers from the vector C. As a more elaborate example, if

X=0.1,0.2,0.1,0.2

is required, we could use, instead,

X=0.1,0.2#X

since the third data item will come from (the already read) first element of X,*etc*. Here and elsewhere, it should be noted that an oversupply of data is of no consequence; we may have

X=10000*0

to zeroise X.

(3) ##

which terminates the scanning operation for this card here so that comments that are not 'active' data may follow.

Example

With

OUTPUT AUS ## UNIT 10 IS DEFAULT

the active scanning of data for this card will terminate on encountering ##. Should more data be expected, the scanning will move on to the next card. This device simply saves cards or, alternatively, comment cards (an * in column 1) could be used to convey the same idea.

We now conclude this description of SKAN for non-advanced users with a feature used to set some SKAN options.

3.4 $OPT Feature

The feature $OPT...$ is used

(1) to set the extent of a data card (default columns 1 to 72) and the option to print input data, and

(2) to set 'flags' used in data filling.

The data supplied are always three integers

$OPT ℓ,m,n $ (*e.g.* $OPT 1,72,0$)

In case (1), if $\ell>0$

then the card width is from column $\ell$ (usually 1) to column m ($\leqslant 80$);

if $n>0$

then the input data will be printed (default n=1).

In case (2), if $\ell=0$

then the 'flags' used in data filling are KOPBTM=m and KOPTOP=n as described below. To use both features together would require, for example,

$OPT 1,72,0 $ $OPT 0,0,9999$ .

When data are undersupplied, *e.g.*

with     COMMON A(10),B(10)

and     A=3*1,3*2 B=10*3

then the default situation is that only six elements of A will have data transferred to them, A(1) to A(3)=1. and A(4) to A(6)=2. However, should the data filling 'flags' KOPBTM and KOPTOP be set, then the array will fill out with zeros (or blanks for alphanumeric data) provided

(i) NOP=1 in the SKAN call, *e.g.*

CALL SKAN(0,IT,IC,1,NK)

and (ii) the KOP for the returned keyword lies in the interval

KOPBTM $\leqslant$ KOP $\leqslant$ KOPTOP.

The default situation is set up with KOPBTM=10000 and KOPTOP=9999 so that no filling takes place; however, following

$OPT 0,0,9999$ (possibly part of the standard data)

filling will take place for all keywords (provided NOP=1). To establish an intermediate situation, we might have

$OPT 0,0,99 $

so that data not to be filled would be given keyword option numbers, KOP, greater than 99 and all other data would be filled out when undersupplied.

## 4. NORMAL SCAN-TYPE USE

Following particular keywords, an advanced user of SKAN may require to read trailing data with the equivalent of the normal SCAN [Bennett & Pollard 1967]. For example, we may have mixed alphanumeric and numeric mode data following a keyword, *e.g.*

OUTPUT PRINT, AUS 10

which we require to set

OUTPUT(1)='PRINTbbb' -OUTPUT is REAL*8

OUTPUT(2)='AUSbbbbb'

NOUT(1)=0

NOUT(2)=10

In this case the normal SCAN call is not permitted so the equivalent is used

CALL SKAN(M,IT,IC,VAR,NK,&RET)

where     M = data mode,

IT and IC are input unit and column counter as before (Section 2.2),

VAR = a variable or array address to hold returned data,

NK = number of equivalent *4 words read, and

&RET = return address should a keyword be encountered.

There are two ways of using SCAN (and the above equivalent); these are discussed briefly below.

(i) M=+ve called user directed

Should say M=1 be specified, then data to follow will be scanned looking for numbers which will be stored as integers, since

M=1 is for INTEGER data,

M=2 is for REAL*4 data,

M=3 is for REAL*8 data, and

M=4 is for alphanumeric data.

(SCAN has many more possibilities but these are not permitted with SKAN.) In our OUTPUT example, PRINT and AUS would be skipped over (not being keywords) and 10 would be located - this obviously does not meet the requirements for this example. In any case, NK is the number of required words to be read on input and the number of actual words returned on output perhaps stopped short by an interruptive keyword.

(ii) M=-ve called data directed

With the input specification M=-ve, only the next appearing data item is scanned (regardless of NK) and M is returned as the negative of the mode of encountered data and VAR(1) (and VAR(2) if M on output is -3 or -4) contains the data. This is different to SCAN, as if M=-1,-2 or -3 on input any encountered numeric data is automatically converted to mode 1,2 or 3 respectively on output. If this conversion feature is not required, then M=-4 should be used on input. If alphanumeric encountered data is a keyword, M=0 is set and the &RET return is made.

We can now tackle the OUTPUT example, segments of which are coded as follows:

```
      REAL*8 OUTPUT(10)
      COMMON OUTPUT,NOUT(10)
      REAL*8 VAR
      EQUIVALENCE(VAR,IVAR)
        .
        .
        .
    1 CALL SKAN(0,1,IC,1,NK)
        .
        .
        .
C     OUTPUT ONLY,SO FAR,AS $MOD SETS LGH TO 0
   11 DO 14 I=1,11
   12 M=-1
      CALL SKAN(M,1,IC,VAR,NK,&1)
```

```
      IF(M.EQ.-1)GO TO 13
C     PRINT FOUND, SAY
      IF(I.EQ.11)GO TO 1
      OUTPUT(I)=VAR
      NOUT(I)=0
      GO TO 14
C     10 FOUND, SAY
   13 IF(I.EQ.1)GO TO 12
      NOUT(I-1)=IVAR
      GO TO 12
   14 CONTINUE
      GO TO 1
      .
      .
      .
```

In our standard data

$MOD(OUTPUT,,0,,1)$

will stop OUTPUT from picking up trailing data. (Unless *all* the table entries for a keyword are nonzero, except KOP, data scanning will stop on the keyword itself.)

For user directed mode (M=+ve) data filling does not normally take place. However, provided the acceptable KOP interval is set up by the user (Section 3.4), data filling could then take place. In its data extraction, SKAN points to the first keyword table entry SKANEXTV (Section 3.1). The keyword option number may be changed from the default of zero with, say, the data

$MOD(SKANEXTV,,,,100)$

so that filling could then be on or off, even if data filling is required with normal keyword data.

NOTE Should SKAN(M,...) be interrupted by an intervening keyword, a following CALL SKAN(0,...) will pick up from the keyword. We may determine attributes (KOP,*etc.*) of the interruptive keyword with

CALL SKAN(0,IT,IC,0,NK)

which will enter the attributes into the labelled COMMON/SKANLK (Section 2.2). Should we want to move away from the interruptive keyword,we would set

LINK=0 .

## 5. PRINTER DUMP ROUTINE

A labelled printer dump of selected variables is given with the routine

CALL DTADMP(LIST,IDMP,IND)

where LIST is a vector of SKAN table pointers readily established as part of our input data which indicates arrays to be dumped onto the printer,

$$\text{IDMP}=\begin{cases}\leqslant 0 & \text{then immediate return is made,}\\ +\text{ve} & \text{then indicates 'segment' of LIST,}\end{cases}$$

and IND= an indicative counter to head output.

Returning to the modified case (Section 3), let us consider the following:

(1) print CASE and C if IDMP=1 -segment 1 of LIST, and

(2) print all variables (C to CASE) if IDMP=2 -segment 2.

Somewhere, probably in the standard data, the following requirement would be set up:

```
$MOD(LIST,,,,9991),(TO,,,,9992)[,(END,,,,9993) is already in table]$
LIST 1 CASE,C LIST 2 C TO CASE END
```

To use 'TO' (actually the keyword with KOP=9992), it must be noted that the order of the SKAN keyword table is in the order of appearance of each variable. (C is first as it was the first to appear with DIMENSION C(4).) Should LIST be required as part of the dump, it would not be trailed by a number. To assign sufficient storage for LIST, all entries in the LIST are counted, except the numbers and TO. Thus, for this example, we would need to have at least

COMMON LIST(7) .

[Since the same list establishing feature of SKAN may be useful to advanced users, the structure is indicated. For our example

```
LIST(1)= 3  - number of elements for segment 1 counting this number
    (2)= 7  - position of CASE in SKAN keyword table (3 default keywords)
    (3)= 4  - position of C in table
    (4)= 3  - number of elements for segment 2 counting this number
    (5)= 4  - position of C in table
    (6)= -7 - ve number means include all variables from 4 to 7
    (7)= 0  - terminal indicator.]
```

In the large codes used by the author, the dump was turned on when required by an intermediate vector. If we have the COMMON vector IDMP, then, at the end of reading all data, we might have

CALL DTADMP(LIST,IDMP(1),1)

(or IF(IDMP(1).GT.0)CALL DTADMP(LIST,IDMP(1),1)

when the calling time would add significantly to the overall time). Further, after all preliminary data are established ready for the main calculation, we might have

```
CALL DTADMP(LIST,IDMP(2),2) .
```

The vector IDMP would initially be zeroised (being part of COMMON, PRELUD would do this). In a rerun of a case producing unexplained results, we might use

```
IDMP=1,2
```

so that after all the data is read, a printer dump would be obtained according to segment 1 of LIST and, after all preliminary data are established, a dump would be obtained following segment 2.

All aspects of SKAN explained so far are for a user not requiring job time, or run time, variable dimensions. Before proceeding with variable dimensions, we introduce a FORTRAN preprocessor which inserts COMMON cards and side entry calls into coding prior to FORTRAN compilation.

## 6. UPDATE - A FORTRAN PREPROCESSOR

In writing programs containing many FORTRAN routines (say 100), maintaining the same COMMON cards (and up-to-date side entries for variable dimensions) in all routines is extremely tedious and a potential source of program error. This possible error is minimised by using UPDATE which inserts COMMON and side entries into source coding. Although the COMMON layout with SKAN is fairly general, when UPDATE is used, it restricts the layout to some extent.

Input layout to UPDATE consists of

```
     $OVERLAY
OVERLAY A
INSERT...
     .
     .
     .
     $END
```

Optional

(i) overlay cards to go on unit 5.

```
     $COMMON
     REAL*8...
     .
     .
     .
     $END
```

Optional

(ii) COMMON cards and standard data to go on unit 4 (same as used by PRELUD of SKAN suite).

```
     SUBROUTINE...
CINSERT COMMON
     .
     .
     .
     END
```

(iii) routines to be updated with COMMON and to be written on unit 8 for later compilation.

In (iii) of the above data, the FORTRAN source coding is simply copied except that the control card (punched from column 1)

CINSERT

inserts data prepared from unit 4 as requested on the control card. We may use

CINSERT <u>COMMON</u> <u>ENTRY</u> SABC

where (1) COMMON means the fixed (first given) COMMON of PRELUD,

and (2) ENTRY means the variable dimensions and side entry list of (trailing cards of a second given) COMMON of PRELUD and here SABC is to be the side entry name (to say routine ABC).

The COMMON must be given in a particular layout.

```
$COMMON         - must start in column 7
COMMON          - column 7, no further data on the card
1MAXA,MAXB,...  - continuation, all punched together
2,A,B,C(b5,10)  - blanks are permitted between brackets
 .
 .
 .
 $              - terminating $ must be in column 7
```

The reason for these restrictions is that in PRELUD, a comma may be replaced by # or blank (see Section 7.4). We therefore adopt the transforming rule here that any intervening # or blank is converted to a comma. The data (which make sense with variable dimensions, Section 7.4)

1LIST#THELOT,A,B,C,D,X,DELTA,GAMAVE

would be transformed to

1LIST,THELOT,A,B,C,D,X,DELTA,GAMAVE

Note that because of the transforming rule a second continuation card

2bA,B

would be wrongly transformed to

2,A,B

ENTRY data are prepared from a second supplied $COMMON data set. As many cards as there are in the fixed (first given) COMMON are counted off and the remaining data up to, but not including, a card starting

COMMON in column 7

are set aside for dimensions of the side entry. The data given after COMMON then form the side entry list so that dimensions should not be given here; rather they must be given earlier, *e.g.*

DIMENSION B(MAXB) .

For example

```
          $COMMON
          REAL*8 X                           four cards
          COMMON
block 1-  1MAXA,MAXB,MAXC
          2,N,X(10),Y(5,5),ETC
          $                                  - end of first COMMON
          PRELUDE(STANDARD)
block 2-  :                                  Section 7.2
          END
block 3-                                     - not supplied here (Section 7.3)
          $COMMON
          REAL*8 X
          COMMON                             these four cards are exact
         1MAXA,MAXB,MAXC                     copies of the earlier ones
         2,N,X(10),Y(5,5),ETC
block 4-  REAL*8 A(MAXA,MAXA)                dimensions passed on with
          DIMENSION B(MAXB),C(MAXC)          side entry list
          COMMON                             - a marker to indicate that list follows
         1A,B,C,                             - side entry list
          $                                  - end of second COMMON
          $MOD...
          *STANDARD DATA                     not used by UPDATE but supplied
block 5-        :                            for SKAN use
          END
block 6-  $END                               - terminal indicator of the lot .
```

The inserted coding derived from the above would be

```
CINSERT COMMON ENTRY SABC
      REAL*8 X
      COMMON
     1MAXA,MAXB,MAXC
     2,N,X(10),Y(5,5),ETC
      GO TO 9999
      ENTRY SABC (
     1A,B,C
     X)
      REAL*8 A(MAXA,MAXA)
```

```
      DIMENSION B(MAXB),C(MAXC)
      RETURN
 9999 CONTINUE
```

An example of the control cards (as used at the AAEC Research Establishment) and a typical job is given as Appendix B.

## 7. VARIABLE DIMENSIONS

Let us now consider each block of data to be supplied to the PRELUD routine, from $COMMON to $END. PRELUD reads its data from the unit ID, indicated in the calling sequence (Section 2.1).

### 7.1 Fixed COMMON

The requirement for variable dimensions is indicated to the PRELUD routine by the user assigning to COMMON the 'reserved keyword' END. For example, for our block 1 of data (see Sections 2.1 and 6) the user might have

```
 $COMMON
 COMMON            - following the restricted layout required
1MAXX,MAXY         - by the UPDATE preprocessor
2,A(3),ETC
X,END(6)           - need not be given last but would normally
 $
```

Should END be given then

(1) END(2 or more)

is used to request additional core for variable dimensions to be procured (by VARRAY; see Appendix A) from the region available to the job as it is currently running (specified, for example, with REGION.GO=480K), or

(2) END or END(1)

is used to tell PRELUD that the variable dimensions are to be stored immediately after fixed COMMON.

When procured core is used (case (1)), all of the remaining region is taken by PRELUD except for some core held back for use as I/O buffers. PRELUD is told the extent of core required for I/O buffers with the size of END;

END(6),

for example, will hold back 6 K bytes of storage (and at the same time will waste six words of storage through the assignment of these words to END never to be used for data storage).

When fixed COMMON is simply extended (case (2)), a subroutine CMKEYS must be supplied which returns to PRELUD the total extent of COMMON available. In addition, the size of the SKAN keyword table is returned and this may need to be extended. The size of the table should be at least

6*(number of user keywords+4) words.

An example of the type of routine to be supplied is given below:

```
      SUBROUTINE CMKEYS(LBCOM,LKEYS)
C     SETS DEFAULT SIZES FOR COMMON AND KEY-TABLE
C     1ST ARGUMENT IS COMMON SIZE (*4 BYTE WORDS)
C     2ND ARGUMENT IS SIZE OF KEY-TABLE USED BY SKAN
      COMMON STORGE(40000)             - default 2
      COMMON/SCANT/KEYS(1200)          - default 1000
      LBCOM=40000                      - same as dimensioned
      LKEYS=1200                       - same as dimensioned
      RETURN
      END
```

In the simple use of SKAN, the routine need not be supplied as the size of fixed COMMON is taken to be the maximum of LBCOM returned by CMKEYS and the size determined from $COMMON...$ data.

We now consider the other blocks of data supplied to the PRELUD routine when variable dimensions are required.

### 7.2 PRELUDE(STANDARD) Data

Default sizes of arrays are given here, for example

```
PRELUDE(STANDARD)
MAXA=10
MAXB=10
MAXC#MAXB
END
```

The data may be strung out to give this block a distinctive appearance or allow it to be entered compactly using columns 1 to 72.

### 7.3 PRELUDE (User's) Data

The PRELUDE data, which are simply the requirement for the current job, are then given as first data *on the input unit IT,* probably 1 (Section 2.1); these need not be given if they are not required. The overall PRELUDE assignment is then on the basis of last defined values from those given from unit ID (Section 2.1) and unit IT. Our data could be, for example,

PRELUDE MAXA=20 END or

PRELUDE MAXA=20, PRINTABLE END .

With the latter option, the special keyword PRINTABLE (originally occupying position 1 of the SKAN table) sets the option to print from here on; this includes a table of keywords at the end of all PRELUD routine processing.

Of course a job may be set up with a set of independent requirements. Having detected a keyword, set aside for the job, which tells the program that a new set of requirements follows (including a possible PRELUDE), the program arranges to

```
CALL PRELUD(...)
```

again. All the storage is cleared (as it is for the first call) as if beginning a new job. In the neutronics code POW [Pollard 1974] for example the already heavily used END keyword is used for this purpose.

### 7.3.1 PRELUC routine

Following the PRELUDE data, the routine PRELUD calls a user's supplied routine (if required) to calculate array sizes such as

```
MAXC=MAXA+MAXB .
```

The call is

```
CALL PRELUC(LCOM,LMCOM) ,
```

where LCOM is the length (in *4 byte words) of fixed COMMON supplied as block 1 data, and LMCOM is the total length of main COMMON available. (If variable dimensioning with VARRAY procured core is being used (Section 7.1), then a hole exists in blank COMMON from LBCOM+1 to IGETM-1 where LBCOM and IGETM are part of the labelled COMMON,

```
COMMON/CSIZE/LBCOM,LSCOM,IGETM,LGETM .)
```

As a simple example such a routine may have

```
      SUBROUTINE PRELUC(LCOM,LMCOM)
C     CALCULATES ARRAY SIZES
CINSERT COMMON                        - see Section 6
      MAXC=MAXA
      IF(MAXC.LT.MAXB)MAXC=MAXB
      RETURN
      END
```

The default version of PRELUC simply returns.

## 7.4 Variable 'COMMON'

Block 4 of data (again from unit ID supplied as part of the call to PRELUD) consists of four segments:

(1) a repeat of the cards supplied as block 1 for fixed COMMON, except that the $ card is not yet given,

(2) a set of variable dimensions, *e.g.*
REAL*8 X(MAXX),

(3) a further COMMON statement used as a trigger, and

(4) a list of variables follows for use directly as side entry variables as indicated in Section 6.

PRELUD sets up a SKAN keyword table which treats all variables as equivalent vectors. The information thereby lost means that a genuine EQUIVALENCE statement is not permitted in the $COMMON... $ data. However, a limited equivalence for variable dimensions may be obtained with this feature. In the list of variables ((4) above), the storage counter for the next available address may be prevented from incrementing (using #). If we have

```
DIMENSION A(MAX),B(MAX),C(MAX)
REAL*8 D(MAX)
COMMON
1A#D,B,C
```

then, after D is assigned (and *8 byte boundary alignment assured), the storage counter is not incremented, so B and C share the storage of D (B the first MAX *4 words and C the second MAX *4 words). It is for this reason (and a feature no longer used which is that, with the comma missing, the variable to follow is not entered into the SKAN table) that the UPDATE program (Section 6) must transform the data to ensure compatibility with FORTRAN. It should be noted that when # is used in the COMMON (side entry) list, the starting address is always set to give *8 byte boundary alignment.

### 7.4.1 PRELUE routine

Following the reading of block 4 data and assignment of addresses, a call is made thus

CALL PRELUE($\overline{\text{LCOM}}$,LMCOM),

where the arguments are the same as for PRELUC (Section 7.3.1) except that LCOM is the length of COMMON used. The intention is that the user may require to assign all the remaining core to, say, TEMP. The side entry list might consist of

```
1A,FLUX#TEMP,B    .
```

The address (and length) of TEMP may be modified in the same technique employed in run time indirect addressing (Section 7.4.2). In PRELUE, the coding

```
IF(LCOM.LT.LMCOM)LCOM=LMCOM
```

would indicate that all of the remaining core has been used.

For most applications the default version, which simply returns, would more than likely be adequate.

### 7.4.2 Run-time indirect addressing

Even while running, that is after PRELUD assignments, addresses (and lengths) of 'EQUIVALENCED' variables can be changed. (When addresses are changed the necessary side entry calls must be made as indicated in Section 7.7.) As an example, if we have

```
          DIMENSION TEMP(MAXT)
             .
block 4      :
data      COMMON
         1A#TEMP,B,C
```

then later (maybe as part of standard data Section 7.5) we would have

$MOD(TEMP,#ITEMP)$,

where ITEMP is a variable of COMMON that contains (indirectly) the address of TEMP (as it is preceded by #). The required address may be set in ITEMP at whatever stage of the program is desired (provided the necessary side entry calls are subsequently made).

(It is noted that to specify a run time variable length, if it is important to do so, we would use, for example

$MOD(TEMP,#ITEMP,LTEMP)$ .

The # has a different meaning in positions other than the second, see Section 3.2, as the addresses there are already indirect and set at run time.)

7.5 Standard Data

Various default values of data are set up in block 5 including all of the likely $MOD data required to run the program in a standard way. The data are normally supplied in five segments

| | | |
|---|---|---|
| (1) | $MOD(START,,,,1),etc $ | - may be more than 1 $MOD |
| (2) | PRINTABLE | - a keyword to set to print what follows |
| (3) | *STANDARD DATA | - a comment to label output |
| (4) | ERR=1.-4,MAXIT=100 | - standard data |
| (5) | END | - end of block |

The default 'scan width' is from column 1 to column 72.

7.6 $END

All of the data read from unit ID by PRELUD (Section 2.1) conclude with the terminal indicator

$END (columns 7 to 10).

The reason for having two terminal indicators END and $END is somewhat historical as other data used by the UPDATE program previously appeared between these two indicators. (Coding could be entered with the UPDATE preprocessor.)

We now consider the manner of making the side entry calls in order to make variable dimensioning effective.

7.7 Side Entry Calls

In a non-overlaid program, all side entry calls could be made initially (except that they would need to be repeated with run time indirect addressing).

In the normal large code situation, all of the input would form an overlay sharing the same core space as that of computational segments. For this case, it would be necessary to recall side entries when an overlay has freshly entered core. In the example given in Appendix B, INPUT and START share the same core space and so our coding could be given as

```
C     MAIN DRIVING CONTROL PROG
      EXTERNAL SINPUT
      EXTERNAL SSTART
CINSERT COMMON
      CALL PRELUD(MIST,1,4,IC)
    2 CALL SIDEN(SINPUT)
      CALL INPUT(IC,&3)
      CALL SIDEN(SSTART)
      CALL START
      GO TO 2
    3 CALL VEXIT(ISTOP)
      END
```

The basic call required to set the side entry addresses into a routine is

CALL SIDEN(SNAME)

where SNAME is a side entry external name. The data addresses held in the SKAN table are passed on to the nominated side entry by the routine SIDEN.

We also note the manner of terminating a run with a call to VEXIT.

### 7.8 Run Termination with VEXIT

The routine VEXIT called

CALL VEXIT(ISTOP)

where ISTOP is a returned program condition code (say zero), should be used at the termination of a run using variable dimensions. The routine releases any VARRAY procured core (Section 7.1) and sets up a return condition code. In the AUS scheme [Robinson 1975] for example, the running, or not, of subsequent modules depends on the value of the returned condition code.

## 8. CONCLUSION

The facilities made available by SKAN as the earlier version DTAV have been used to develop and run production neutronics modules of the AUS scheme [Robinson 1975]. The present version, which is easier to use, will be used in some modules presently being developed. Experience with the facilities has shown that the use of keyword data and variable dimensions need no longer be considered a task to avoid on account of tedious programming details. Those not familiar with the need for free format input in neutronics codes should refer to the GYMEA report [Pollard & Robinson 1969].

9. ACKNOWLEDGEMENTS

Thanks are due to users of this suite of modules, particularly Mrs Baiba Harrington, Dr Ken Maher and Mr Stewart Whittlestone who were willing to 'give it a go' in its early stages of development.

10. REFERENCES

Bennett, N.W. & Pollard, J.P. [1967] - SCAN - A free input subroutine for the IBM360. AAEC/TM399.

Pollard, J.P. [1974] - AUS module POW - A general purpose 0,1 and 2D, multi-group neutron diffusion code including feedback-free kinetics. AAEC/E269.

Pollard, J.P. & Robinson, G.S. [1969] - GYMEA - A nuclide depletion, space independent, multigroup neutron diffusion, data preparation code (IBM360 version). Unpublished, but an updated version of AAEC/E147.

Robinson, G.S. [1975] - AUS - The Australian modular scheme for reactor neutronic computations. AAEC/E369.

## APPENDIX A

### VARRAY FOR DYNAMIC PROCUREMENT OF CORE STORAGE

(G.W. Cox & J.P. Pollard)

For some applications (particularly large programs already written) the switch over to dynamically obtained core rather than an extensive fixed COMMON block for variable dimensions is usually difficult. VARRAY was written to make the switch as simple as possible. The idea is that all addresses are taken relative to some array A which may be a DIMENSIONed variable held in the program itself or part of labelled or blank COMMON.

(1) The extent of core available to the program is determined thus:

DIMENSION A(1)

CALL NARRAY(NARD) } if all the user's region is required, except for
NARD=NARD-1500 } 1500 words for later use of buffers for I/O

(in any case NARD, say, is the number of *4 words of storage required) then

CALL VARRAY(A,IARD,NARD)

returns IARD as the address relative to A (treated as a *4 vector array) such that A(IARD) = 1st word of storage made available with VARRAY

A(IARD+1) = 2nd " " " " " " "

A(IARD+NARD-1) = last word of storage made available with VARRAY.

(2) Possibly, other subroutines could then be called:

CALL ABC(A(IARD),NARD)

⋮

then SUBROUTINE ABC(V,N)

DIMENSION V(N) so that here V(1)=first word of storage from VARRAY, *etc.*

(3) The storage is always aligned to a *8 boundary.

(4) Successive calls can be made to VARRAY, possibly from different subroutines, and each parcel of core can be released for subsequent use with the statement

CALL VARRAY(A,IARD,-NARD[,&99])

which releases only that core previously procured with arguments A,IARD,NARD and which returns *via* the optional error return (in this case, statement 99), if no such core was made available. Of course, haphazard procuring and releasing of core could result in fragmentation of the user's region, for example,

user's core region |←——released——→|←——retained——→|←——released——→| .

(5) Release of all core procured by VARRAY is carried out with the statement

CALL VARRAY no arguments

APPENDIX A (continued)

and such release must be carried out to terminate a multistep job. Instead of STOP n as a program terminator we would then use

```
CALL VEXIT(N)
```

where we have

```
SUBROUTINE VEXIT(N)
CALL VARRAY
CALL CEXIT(N) - this is equivalent to STOP n except that STOP is not
                typed
STOP          - to avoid compiler error
END
```

## APPENDIX B

## SKAN TEST EXAMPLE

The test example uses a method of implicit non-stationary iteration [MINI, Barry & Pollard 1977†] to solve the set of linear equations

$$Ax=c \quad ,$$

where A is a symmetric diagonally dominant matrix with -ve off diagonal elements. The comments in the program listing give an idea of the input. Of course, the example is purely illustrative of SKAN features and is hardly worthwhile, for example, to overlay. Nevertheless, some facets desired in a large code are covered.

First a listing of all cards required to run the job at the AAEC Research Establishment on an IBM360/65 computer is given. This is followed by a listing of the run.

---

† Barry, J.M. & Pollard, J.P. [1978] - Method of implicit non-stationary iteration for solving neutron diffusion linear equations. To be published.

## INPUT

```
//JPPSS     JOB ('PH100930/P22PHNRK',N1),J.P.POLLARD,
//                 CLASS=B,
//                 TIME=2
/*ROUTE   PRINT PHYS
/*JOBPARM L=4
//UPY EXEC PROGEXEC,LIB='AUS.POW',MEM='UPDATE',LIST=0
//GO.FT04F001 DD DSN=AUS.POW3DCOM,
//                 DISP=OLD
//GO.FT05F001 DD DSN=AUS.POW3DOVL,
//                 DISP=OLD
//GO.FT08F001 DD DSN=&POW3D,DISP=(NEW,PASS),
//                 UNIT=SYSDA,SPACE=(TRK,(40,20)),
//                 DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//GO.SYSIN DD *
      $OVERLAY
 OVERLAY A
 INSERT INPUT
 INSERT SKAN
 INSERT PRELUD
 INSERT DTAIN
 INSERT DTAIM
 INSERT DTALST
 INSERT DTADCM
 INSERT CLSCAN
 INSERT CSCAN
 OVERLAY A
 INSERT START
      $END
      $COMMON
*     SKAN TEST EXAMPLE (SOLVES AX=C USING MINI)
      REAL*8 NAME,GAMMA,RTEMP
      COMMON
     1 MIST,MAXN,MAXLOT,N
     2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
     3,IROW,ICOL,INROW,INCOL,ISTOP
     X,END(6)
      $
      PRELUDE (STANDARD)
      MAXN=20
      MIST=100
      END
      $COMMON
*     SKAN TEST EXAMPLE (SOLVES AX=C USING MINI)
      REAL*8 NAME,GAMMA,RTEMP
      COMMON
     1 MIST,MAXN,MAXLOT,N
     2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
     3,IROW,ICOL,INROW,INCOL,ISTOP
     X,END(6)
      REAL*8 THELOT(MAXLOT),A(MAXN,MAXN),B(MAXN,MAXN),C(MAXN),D(MAXN)
     1,X(MAXN),DELTA(MAXN),GAMAVE(MAXN)
      DIMENSION LIST(MIST)
      COMMON
     1 LIST#THELOT,A,B,C,D,X,DELTA,GAMAVE
      $
$MOD(NAME,,,4),(ROW,IROW,4,1,1),(COL,ICOL,3,1,2),(ELM,IROW,4,1,3)
(START,,,,4),(STOP,ISTOP,#,#,5)             $
$MOD(LIST,,,9991),(TO,,,,9992)$
$OPT 0,0,9999$
*FILLS OUT ALL DATA
PRINTABLE
*STANDARD DATA FOLLOWS
NAME=THISJOB,MAX=100,ERR=1.-4,N#MAXN
LIST 1 NAME LIST 2 LIST 3 NIT,X END
      END
      $END
C     SKAN TEST EXAMPLE
C     SOLVES AX=C USING MINI
C     DATA ...
C     N=ORDER, A=FULL MATRIX DATA, ROW(1)A=DATA FOR ROW(1)
C     COL(2)A=DATA FOR COL(2), C=FULL VECTOR DATA
C     ELM(1,1)A=1, ELM(2)C=0,
C     NAME=A DESCRIPTIVE NAME
C     CAN HAVE B INSTEAD OF A AND D INSTEAD OF C (AS HOLDING STORE)
C     LIST 1 NAME LIST 2 LIST 3 NIT,X END (2=PRINT DURING ITERATION)
C     START BEGINS CALC  STOP TERMINATES RUN
      EXTERNAL SINPUT
      EXTERNAL SSTART
CINSERT COMMON
    3 WRITE(3,1)
```

```
    1 FORMAT('0**SKAN TEST EXAMPLE'/)
      CALL PRELUD (MIST,4,IC)
    2 CALL SIDEN(SINPUT)
      CALL INPUT(IC,&3,&4)
C     START CALLS START. END BEGINS AFRESH. STOP STOPS
      CALL SIDEN(SSTART)
      CALL START
      GO TO 2
    4 WRITE(3,5)ISTOP
    5 FORMAT('0**SKAN TEST EXAMPLE STOP',I3)
      CALL VEXIT(ISTOP)
C     TO PREVENT COMPILER DIAGNOSTIC
      STOP
      END
      SUBROUTINE INPUT(IC,*,*)
C     SETS UP INPUT DATA REQUIRED
CINSERT COMMON ENTRY SINPUT
      REAL*8 RCOM(1)
      EQUIVALENCE (MIST,RCOM(1))
      COMMON/SKANLK/LK(5),KOP,LINK
      IT=1
C     KEYWORD CONTROLLED SKAN
    1 CALL SKAN(0,IT,IC, 1,NK)
      IF(KOP.EQ.0)GO TO 1
      GO TO (11,12,13,14,15),KOP
C     END BEGINS AFRESH
      IF(KOP.EQ.9993)RETURN1
C     ENDOFILE
      ISTOP=0
      GO TO 15
C     ROW(1)
   11 INCOL=1
      GO TO 13
C     COL(1)
   12 INROW=1
      IROW=0
C     ELM(1,2)
   13 IF(IROW.EQ.0)IROW=1
      IF(ICOL.EQ.0)ICOL=1
C     FIND FOLLOWING KEYWORD , SAY A
      CALL SKAN(0,IT,IC, 0,NK)
C     FREE SKAN TO MOVE ON PAST KEYWORD
      LINK=0
C     IADRES IS FOR REAL*8 VAR RELATIVE TO 1ST OF COMMON
      IADRES=(LK(3)-1)/2+1
C     PICK UP DATA ONE WORD AT A TIME
      DO 21 I=1,N
C     2,*4 WORDS FOR REAL*8
      NK=2
      CALL SKAN(    3,IT,IC,RTEMP,NK,&22)
      II=IADRES+IROW-1+(ICOL-1)*MAXN
      RCOM(II)=RTEMP
      IROW=IROW+INROW
   21 ICOL=ICOL+INCOL
C     THUMB THRO TO NEXT KEYWORD
   22 CALL SKAN(0,IT,IC, 0,NK)
      GO TO 1
C     START
   14 RETURN
C     STOP
   15 RETURN2
      END
      SUBROUTINE START
C     SOLN OF AX=C USING MINT
CINSERT COMMON ENTRY SSTART
      REAL*8 U,V,W1,W2,AIJ,XI,XJ,GIJD
      CALL DTADMP(LIST,1,1)
      IF(MAX.LE.0)RETURN
      DO 21 I=1,N
   21 DELTA(I)=0.D0
      DO 1 NIT=1,MAX
      ERROR=-1.
      DO 20 II=1,N
   20 GAMAVE(II)=0.D0
      NAVE=0
      DO 2 I=1,N
      V=A(I,I)
      U=C(I)-V*X(I)
      IF(I.EQ.1)GO TO 4
      I1=I-1
```

```
      DO 3 J=1,I1
    3 U=U-A(I,J)*X(J)
    4 W1=0.D0
      W2=0.D0
      IF(I.EQ.N)GO TO 10
      I1=I+1
      DO 5 J=I1,N
      AIJ=A(I,J)
      IF(AIJ.EQ.0.D0)GO TO 5
      XI=DABS(DELTA(I))
      IF(XI.EQ.0.D0)GO TO 6
      XJ=DABS(DELTA(J))
      IF(XJ.GT.XI)GO TO 7
C     NORMAL
      GAMMA=XJ/XI
      GO TO 8
C     UPEND
    7 GAMMA=XI/XJ
    8 XI=X(I)
      XJ=X(J)
      IF(XJ.GT.XI)GO TO 9
C     DOWNHILL
      GIJD=XJ/XI
      IF(GAMMA.GT.GIJD)GAMMA=GIJD
    9 W2=W2+AIJ*GAMMA
    6 W1=W1+AIJ*X(J)
      GAMAVE(I)=GAMAVE(I)+GAMMA
      NAVE=NAVE+1
    5 CONTINUE
      V=V+W2
   10 DELTA(I)=(U-W1)/V
      IF(ERROR.LT.ERR)ERROR=DABS(DELTA(I)/X(I))
      X(I)=X(I)+DELTA(I)
    2 CONTINUE
      IF(NAVE.EQ.0)GO TO 24
      DO 23 I=1,N
   23 GAMAVE(I)=GAMAVE(I)/NAVE
   24 CALL DTADMP(LIST,2,2)
      IF(ERROR.LE.ERR)GO TO 11
    1 CONTINUE
C     NOT CONVERGED IN MAX ITERATIONS
   11 CONTINUE
      CALL DTADMP(LIST,3,3)
      RETURN
      END
      SUBROUTINE PRELUC(LCOM,LMCOM)
CINSERT COMMON
      MAXLOT=MAXN*(5+2*MAXN)
      RETURN
      END
/*
//COM EXEC FORTHC,REGION.FORT=240K
//FORT.SYSIN DD DSN=&POW3D,DISP=(OLD,DELETE)
//POW3D EXEC PROGUPD,LIB='AUS.POW3D',OVC='AUS.POW3DOVL',
//             RLSE=,PLIB='AUS.POW',PLIB1='PHYS.FORTLIB',GO=0
//GO EXEC PROGEXEC,LIB='AUS.POW3D',LIST=0,REGGO=240K
//GO.FT04F001 DD DSN=AUS.POW3DCOM,
//             DISP=SHR
//GO.SYSIN DD *
*SKAN TEST DATA
NAME=HARDJOB N=4
A=1,-0.01,2*0         16*0  ##TO FILL OUT REST OF 1ST ROW
-0.01,1,-0.99,0       16*0  ##NORMALLY WE WOULD REDUCE SIZE OF A DYNAMICALLY
0,-0.99,1,-0.01       16*0
2*0,-0.01,1
$OPT 1,80,1$                ##CHANGE TO 80 COL CARDS
            LIST 1 NAME,A,X,C LIST 2 LIST 3 NAME,NIT,ERROR,X,DELTA,GAMAVE, END
B#A                         ##SAVE A COPY OF A IN B
*A BAD TRIAL SOLN FOR LATER
D=1,-3,1,+3,1,-3,1,+3
C=4*1 X=4*1 START
X=#2,2*1,-10                ##TRIAL SOLN 1ST 2 ELMS EXACT, REST 1,-10
START
ELM(1,2)A=-0.0075,ELM(2,1)A=-0.0075 X#D START
A#B                         ##RESTORE ORIG DATA
ROW(4)A=0 0 -0.0075,1 COL(4)A=0 0 -0.0075,1,DO
C,ELM(4)C=1,X#D START       ##C NO DATA = 20*0 AS DATA FILL SET ON
END                         ##BEGINS AFRESH
PRELUDE PRINTABLE END
*ONLY GIVES TABLE LIST BUT NORMALLY WE WOULD HAVE FRESH DATA
STOP 2
/*
//
```

# OUTPUT

```
                                   H A S P   S Y S T E M   L O G

$ 10.24.36 JOB  95 -- JPPSS    -- BEGINNING EXEC - INIT 6 - CLASS B
$ 10.28.26 JOB  95 END(6 ) HCC= 002


 HASP STATISTICS --    243 CARDS READ --     691 LINES PRINTED --     0 CARDS PUNCHED --   3.83 MINUTES EXECUTION TIME

                 --      0 BLOCKS OF PLOT OUTPUT      0 BLOCKS OF PAPER TAPE OUTPUT

//JPPSS     JOB (*********/P22PHNRK*.N1).J.P.POLLARD.                       JOB  95
//                CLASS=H,
//                TIME=2
***ROUTE PRINT PHYS
***JOBPARM1=4
//UPY EXEC PROGEXEC.LIB='AUS.POW',MEM='UPDATE',LIST=0
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /STEP0    / START 77139.1024
IEF374I STEP /STEP0    / STOP  77139.1024 CPU    0MIN 00.02SEC MAIN   4K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF202I - STEP - LIST      , WAS NOT RUN BECAUSE OF CONDITION CODES.************************************************************************
IEF373I STEP /LIST     / START 77139.1024
IEF374I STEP /LIST     / STOP  77139.1025 CPU    0MIN 00.00SEC MAIN   0K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LKED     / START 77139.1025
IEF374I STEP /LKED     / STOP  77139.1025 CPU    0MIN 00.04SEC MAIN   4K LCS    0K
*** CONDITION CODE =  000(HEX)
//GO.FT04F001 DD DSN=AUS.POW3DCOM,
//                DISP=OLD
//GO.FT05F001 DD DSN=AUS.POW3DOVL,
//                DISP=OLD
//GO.FT08F001 DD DSN=&POW3D,DISP=(NEW,PASS),
//                UNIT=SYSDA,SPACE=(TRK,(40,20)),
//                DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
//GO.SYSIN DD *
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /GO       / START 77139.1025
IEF374I STEP /GO       / STOP  77139.1025 CPU    0MIN 02.15SEC MAIN  62K LCS    0K
*** CONDITION CODE =  000(HEX)
//COM EXEC FORTHC,REGION.FORT=240K
//FORT.SYSIN DD DSN=&POW3D,DISP=(OLD,DELETE)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /FORT     / START 77139.1025
IEF374I STEP /FORT     / STOP  77139.1026 CPU    0MIN 03.54SEC MAIN 240K LCS    0K
*** CONDITION CODE =  000(HEX)
//POW3D EXEC PROGUPD,LIB='AUS.POW3D',OVC='AUS.POW3DOVL',
//                PLSF=,PLIB='AUS.POW',PLIB1='PHYS.FORTLIB',GO=0
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF287I   AUS.POW3D                                          NOT RECTLGD 2
IEF287I   VOL SER NOS= AAE003.
IEF373I STEP /STEP0    / START 77139.1026
IEF374I STEP /STEP0    / STOP  77139.1026 CPU    0MIN 00.09SEC MAIN   8K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LIBUPD   / START 77139.1026
IEF374I STEP /LIBUPD   / STOP  77139.1026 CPU    0MIN 02.62SEC MAIN 106K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LKED     / START 77139.1026
IEF374I STEP /LKED     / STOP  77139.1027 CPU    0MIN 03.14SEC MAIN 100K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /COMPRESS/ START 77139.1027
IEF374I STEP /COMPRESS/ STOP  77139.1027 CPU    0MIN 00.20SEC MAIN  10K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LIST     / START 77139.1027
IEF374I STEP /LIST     / STOP  77139.1028 CPU    0MIN 01.35SEC MAIN  12K LCS    0K
*** CONDITION CODE =  000(HEX)
IEF202I - STEP - GO        , WAS NOT RUN BECAUSE OF CONDITION CODES.************************************************************************

IEF373I STEP /GO       / START 77139.1028
IEF374I STEP /GO       / STOP  77139.1028 CPU    0MIN 00.00SEC MAIN   0K LCS    0K
```

```
*** CONDITION CODE =   000(HEX)
//GO EXEC PROGEXEC,LIB='AUS.POW3D',LIST=0,REGGO=240K
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /STEP0    / START 77139.1028
IEF374I STEP /STEP0    / STOP  77139.1028 CPU    0MIN 00.02SEC MAIN   4K LCS    0K
*** CONDITION CODE =   000(HEX)
IEF202I - STEP - LIST     - WAS NOT RUN BECAUSE OF CONDITION CODES.*****************************************************************
IEF373I STEP /LIST     / START 77139.1028
IEF374I STEP /LIST     / STOP  77139.1028 CPU    0MIN 00.00SEC MAIN   0K LCS    0K
*** CONDITION CODE =   000(HEX)
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LKED     / START 77139.1028
IEF374I STEP /LKED     / STOP  77139.1028 CPU    0MIN 00.02SEC MAIN   4K LCS    0K
*** CONDITION CODE =   000(HEX)
//GO.FT04F001 DD DSN=AUS.POW3DCOM,
//             DISP=SHR
//GO.SYSIN DD *
IEF142I - STEP WAS EXECUTED - COND CODE 0002
IEF373I STEP /GO       / START 77139.1028
IEF374I STEP /GO       / STOP  77139.1028 CPU    0MIN 04.42SEC MAIN 240K LCS    0K
*** CONDITION CODE =   002(HEX)
IEF375I  JOB /JPPSS    / START 77139.1024
IEF376I  JOB /JPPSS    / STOP  77139.1028 CPU    0MIN 17.61SEC
HIGHEST CONDN CODE =   002(HEX)


UPDATE

CINSERT COMMON
      END
      SUBROUTINE INPUT(IC,*,*)
CINSERT COMMON ENTRY SINPUT
C     END BEGINS AFRESH
      END
      SUBROUTINE START
CINSERT COMMON ENTRY SSTART
      END
      SUBROUTINE PRELUC(LCOM,LMCOM)
CINSERT COMMON
      END

END


 LEVEL 20.1 (AUG 71)                         OS/360  FORTRAN H                                    DATE  77.139/10.25.41

          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
                C      SKAN TEST EXAMPLE
                C      SOLVES AX=C USING MINT
                C      DATA ...
                C      N=ORDER, A=FULL MATRIX DATA, ROW(1)A=DATA FOR ROW(1)
                C      COL(2)A=DATA FOR COL(2), C=FULL VECTOR DATA
                C      ELM(1,1)A=1, ELM(2)C=0,
                C      NAME=A DESCRIPTIVE NAME
                C      CAN HAVE B INSTEAD OF A AND D INSTEAD OF C (AS HOLDING STORE)
                C      LIST 1 NAME LIST 2 LIST 3 NIT,X,END (2=PRINT DURING ITERATION)
                C      START BEGINS CALC  STOP TERMINATES RUN
  ISN 0002             EXTERNAL SINPUT
  ISN 0003             EXTERNAL SSTART
                CINSERT COMMON
                C      SKAN TEST EXAMPLE (SOLVES AX=C USING MINT)
  ISN 0004             REAL*8 NAME,GAMMA,RTEMP
  ISN 0005             COMMON
                      1 MIST,MAXN,MAXLOT,N
                      2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
                      3,IROW,ICOL,INROW,INCOL,ISTOP
                      X,END(6)
  ISN 0006           3 WRITE(3,1)
  ISN 0007           1 FORMAT('0*SKAN TEST EXAMPLE'/)
  ISN 0008             CALL PRELUD (MISTA,IC)
  ISN 0009           2 CALL SIDEN(SINPUT)
  ISN 0010             CALL INPUT(IC,&3,&4)
                C      START CALLS START, END BEGINS AFRESH, STOP STOPS
  ISN 0011             CALL SIDEN(SSTART)
  ISN 0012             CALL START
  ISN 0013             GO TO 2
```

```
   ISN 0014          4 WRITE(3,5)ISTOP
   ISN 0015          5 FORMAT('0**SKAN TEST EXAMPLE STOP',I3)
   ISN 0016            CALL VEXIT(ISTOP)
               C       TO PREVENT COMPILER DIAGNOSTIC
   ISN 0017            STOP
   ISN 0018            END
*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,

*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF

*STATISTICS*      SOURCE STATEMENTS =       17 ,PROGRAM SIZE =       468

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******                                   113K BYTES OF CORE NOT USED

LEVEL 20.1 (AUG 71)                               OS/360  FORTRAN H                       DATE  77.139/10.25.46

        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,
                               SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
   ISN 0002            SUBROUTINE INPUT(IC,*,*)
               C       SETS UP INPUT DATA REQUIRED
               CINSERT COMMON ENTRY SINPUT
               C       SKAN TEST EXAMPLE (SOLVES AX=C USING MINT)
   ISN 0003            REAL*8 NAME,GAMMA,RTEMP
   ISN 0004            COMMON
                      1 MIST,MAXN,MAXLOT,N
                      2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
                      3,IROW,ICOL,INROW,INCOL,ISTOP
                      X,END(6)
   ISN 0005            GO TO 9999
   ISN 0006            ENTRY SINPUT  (
                      1 LIST,THELOT,A,B,C,D,X,DELTA,GAMAVE
                      X)
   ISN 0007            REAL*8 THELOT(MAXLOT),A(MAXN,MAXN),B(MAXN,MAXN),C(MAXN),D(MAXN)
                      1,X(MAXN),DELTA(MAXN),GAMAVE(MAXN)
   ISN 0008            DIMENSION LIST(MIST)
   ISN 0009            RETURN
   ISN 0010       9999 CONTINUE
   ISN 0011            REAL*8 RCOM(1)
   ISN 0012            EQUIVALENCE (MIST,RCOM(1))
   ISN 0013            COMMON/SKANLK/LK(5),KOP,LINK
   ISN 0014            IT=1
               C       KEYWORD CONTROLLED SKAN
   ISN 0015          1 CALL SKAN(0,IT,IC, 1,NK)
   ISN 0016            IF(KOP.EQ.0)GO TO 1
   ISN 0018            GO TO (11,12,13,14,15),KOP
               C       END BEGINS AFRESH
   ISN 0019            IF(KOP.EQ.9993)RETURN1
               C       ENDOFILE
   ISN 0021            ISTOP=0
   ISN 0022            GO TO 15
               C       ROW(1)
   ISN 0023         11 INCOL=1
   ISN 0024            GO TO 13
               C       COL(1)
   ISN 0025         12 INROW=1
   ISN 0026            IROW=0
               C       ELM(1,2)
   ISN 0027         13 IF(IROW.EQ.0)IROW=1
   ISN 0029            IF(ICOL.EQ.0)ICOL=1
               C       FIND FOLLOWING KEYWORD , SAY A
   ISN 0031            CALL SKAN(0,IT,IC, 0,NK)
               C       FREE SKAN TO MOVE ON PAST KEYWORD
   ISN 0032            LINK=0
               C       IADRES IS FOR REAL*8 VAR RELATIVE TO 1ST OF COMMON
   ISN 0033            IADRES=(LK(3)-1)/2+1
               C       PICK UP DATA ONE WORD AT A TIME
   ISN 0034            DO 21 I=1,N
               C       2,*4 WORDS FOR REAL*8
   ISN 0035            NK=2
                                                                                              PAGE 002

   ISN 0036            CALL SKAN(    3,IT,IC,RTEMP,NK,&22)
   ISN 0037            II=IADRES+IROW-1+(ICOL-1)*MAXN
   ISN 0038            RCOM(II)=RTEMP
   ISN 0039            IROW=IROW+INROW
   ISN 0040         21 ICOL=ICOL+INCOL
               C       THUMB THRO TO NEXT KEYWORD
```

```
ISN 0041         22 CALL SKAN(0,IT,IC, 0,NK)
ISN 0042            GO TO 1
               C    START
ISN 0043         14 RETURN
               C    STOP
ISN 0044         15 RETURN2
ISN 0045            END
*OPTIONS IN EFFECT*      NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,
*OPTIONS IN EFFECT*      SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
*STATISTICS*     SOURCE STATEMENTS =       44 ,PROGRAM SIZE =     1264
*STATISTICS*  NO  DIAGNOSTICS GENERATED
****** END OF COMPILATION ******                                   109K BYTES OF CORE NOT USED
LEVEL 20.1 (AUG 71)                           OS/360  FORTRAN H                              DATE  77.139/10.25.51
          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
ISN 0002            SUBROUTINE START
               C    SOLN OF AX=C USING MINT
               CINSERT COMMON ENTRY SSTART
               C    SKAN TEST EXAMPLE (SOLVES AX=C USING MINT)
ISN 0003            REAL*8 NAME,GAMMA,RTEMP
ISN 0004            COMMON
                   1 MIST,MAXN,MAXLOT,N
                   2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
                   3,IROW,ICOL,INROW,INCOL,ISTOP
                   X,END(6)
ISN 0005            GO TO 9999
ISN 0006            ENTRY SSTART  (
                   1 LIST,THELOT,A,B,C,D,X,DELTA,GAMAVE
                   X)
ISN 0007            REAL*8 THELOT(MAXLOT),A(MAXN,MAXN),B(MAXN,MAXN),C(MAXN),D(MAXN)
                   1,X(MAXN),DELTA(MAXN),GAMAVE(MAXN)
ISN 0008            DIMENSION LIST(MIST)
ISN 0009            RETURN
ISN 0010       9999 CONTINUE
ISN 0011            REAL*8 U,V,W1,W2,AIJ,XI,XJ,GIJD
ISN 0012            CALL DTADMP(LIST,1,1)
ISN 0013            IF(MAX.LE.0)RETURN
ISN 0015            DO 21 I=1,N
ISN 0016         21 DELTA(I)=0.D0
ISN 0017            DO 1 NIT=1,MAX
ISN 0018            ERROR=-1.
ISN 0019            DO 20 II=1,N
ISN 0020         20 GAMAVE(II)=0.D0
ISN 0021            NAVE=0
ISN 0022            DO 2 I=1,N
ISN 0023            V=A(I,I)
ISN 0024            U=C(I)-V*X(I)
ISN 0025            IF(I.EQ.1)GO TO 4
ISN 0027            I1=I-1
ISN 0028            DO 3 J=1,I1
ISN 0029          3 U=U-A(I,J)*X(J)
ISN 0030          4 W1=0.D0
ISN 0031            W2=0.D0
ISN 0032            IF(I.EQ.N)GO TO 10
ISN 0034            I1=I+1
ISN 0035            DO 5 J=I1,N
ISN 0036            AIJ=A(I,J)
ISN 0037            IF(AIJ.EQ.0.D0)GO TO 5
ISN 0039            XI=DABS(DELTA(I))
ISN 0040            IF(XI.EQ.0.D0)GO TO 6
ISN 0042            XJ=DABS(DELTA(J))
ISN 0043            IF(XJ.GT.XI)GO TO 7
               C    NORMAL
ISN 0045            GAMMA=XJ/XI
ISN 0046            GO TO 8
               C    UPEND
                                                                                           PAGE 002
ISN 0047          7 GAMMA=XI/XJ
ISN 0048          8 XI=X(I)
ISN 0049            XJ=X(J)
ISN 0050            IF(XJ.GT.XI)GO TO 9
               C    DOWNHILL
```

```
  ISN 0052            GIJD=XJ/XI
  ISN 0053            IF(GAMMA.GT.GIJD)GAMMA=GIJD
  ISN 0055          9 W2=W2+AIJ*GAMMA
  ISN 0056          6 W1=W1+AIJ*X(J)
  ISN 0057            GAMAVE(I)=GAMAVE(I)+GAMMA
  ISN 0058            NAVE=NAVE+1
  ISN 0059          5 CONTINUE
  ISN 0060            V=V+W2
  ISN 0061         10 DELTA(I)=(U-W1)/V
  ISN 0062            IF(ERROR.LT.ERR)ERROR=DABS(DELTA(I)/X(I))
  ISN 0064            X(I)=X(I)+DELTA(I)
  ISN 0065          2 CONTINUE
  ISN 0066            IF(NAVE.EQ.0)GO TO 24
  ISN 0068            DO 23 I=1,N
  ISN 0069         23 GAMAVE(I)=GAMAVE(I)/NAVE
  ISN 0070         24 CALL DTADMP(LIST,2,2)
  ISN 0071            IF(ERROR.LE.ERR)GO TO 11
  ISN 0073          1 CONTINUE
              C       NOT CONVERGED IN MAX ITERATIONS
  ISN 0074         11 CONTINUE
  ISN 0075            CALL DTADMP(LIST,3,3)
  ISN 0076            RETURN
  ISN 0077            END
*OPTIONS IN EFFECT*       NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF

*STATISTICS*      SOURCE STATEMENTS =        76 ,PROGRAM SIZE =       2164

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******                                      101K BYTES OF CORE NOT USED

 LEVEL 20.1 (AUG 71)                          OS/360  FORTRAN H                            DATE  77.139/10.25.55

         COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,
                            SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
  ISN 0002            SUBROUTINE PRELUC(LCOM,LMCOM)
              CINSERT COMMON
              C       SKAN TEST EXAMPLE (SOLVES AX=C USING MINT)
  ISN 0003            REAL*8 NAME,GAMMA,RTEMP
  ISN 0004            COMMON
                     1 MIST,MAXN,MAXLOT,N
                     2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR
                     3,IROW,ICOL,INROW,INCOL,ISTOP
                     X,END(6)
  ISN 0005            MAXLOT=MAXN*(5+2*MAXN)
  ISN 0006            RETURN
  ISN 0007            END
*OPTIONS IN EFFECT*       NAME=  MAIN,OPT=00,LINECNT=55,SIZE=0000K,

*OPTIONS IN EFFECT*       SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF

*STATISTICS*      SOURCE STATEMENTS =         6 ,PROGRAM SIZE =        236

*STATISTICS*  NO  DIAGNOSTICS GENERATED

****** END OF COMPILATION ******                                      113K BYTES OF CORE NOT USED

*STATISTICS*  NO  DIAGNOSTICS THIS STEP


*******************************
*  IBCOMOD LOADED AT 0015C558  *
*******************************


                 -------- EDITED SYSPRINT MESSAGES FROM LINKAGE EDITOR (LIBRARY UPDATE)--------

****MAIN      NOW REPLACED IN DATA SET
IEW0000        ALIAS    SINPUT                                                 PROGORG
****INPUT     NOW REPLACED IN DATA SET
****SINPUT    IS AN ALIAS FOR THIS MEMBER
IEW0000        ALIAS    SSTART                                                 PROGORG
****START     NOW REPLACED IN DATA SET
****SSTART    IS AN ALIAS FOR THIS MEMBER
****PRELUC    NOW REPLACED IN DATA SET
```

```
END OF SYSPRINT MESSAGES.


 F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED LIST,LET,OVLY
          DEFAULT OPTION(S) USED -  SIZE=(90112,6144)
IEW0000        INCLUDE SYSLIB(MAIN)
IEW0000        ENTRY MAIN
IEW0000        SETSSI 27713910
IEW0000       OVERLAY A
IEW0000       INSERT INPUT
IEW0000       INSERT SKAN
IEW0000       INSERT PRELUD
IEW0000       INSERT DTAIN
IEW0000       INSERT DTAIM
IEW0000       INSERT DTALST
IEW0000       INSERT DTADCM
IEW0000       INSERT CLSCAN
IEW0000       INSERT CSCAN
IEW0000       OVERLAY A
IEW0000       INSERT START
****PROGPGM     NOW REPLACED IN DATA SET


                                  PARTIAL DISK AAF003 CONTENTS     PAGE 1        19 MAY 77  10 HRS 27 MIN 46.11 SEC
DSNAME=AUS.POW3D,DCB=(RECFM=U,BLKSIZE=13030),SPACE=(TRK,(16,50,***))
       USES 62 TRACKS AND 11 BLOCKS, 1 EXTENT
         CKSCAN     (ALIAS)                         77.132.10  TTR=000320    CORE=0006E8
         CLSCAN     (SUBPROGRAM)           COMPILED 77.132.10  TTR=000320    CORE=0006E8
         CMKEYS     (SUBPROGRAM)           COMPILED 77.133.09  TTR=000723    CORE=0010A8
         CNSCAN     (ALIAS)                         77.132.10  TTR=000320    CORE=0006E8
         DTACOM     (ALIAS)                         77.138.14  TTR=001709    CORE=002828
         DTADCM     (SUBPROGRAM)           COMPILED 77.132.10  TTR=000416    CORE=000558
         DTADDC     (ALIAS)                         77.138.14  TTR=001709    CORE=002828
         DTADLS     (ALIAS)                         77.132.10  TTR=00040F    CORE=0004E0
         DTADMP     (SUBPROGRAM)           COMPILED 77.132.10  TTR=000405    CORE=000D00
         DTADPS     (ALIAS)                         77.132.10  TTR=000405    CORE=000D00
         DTAHOL     (ALIAS)                         77.138.14  TTR=001709    CORE=002828
         DTAIM      (SUBPROGRAM)           COMPILED 77.133.09  TTR=000715    CORE=0001B0
         DTAIN      (SUBPROGRAM)           COMPILED 77.138.14  TTR=001709    CORE=002828
         DTALST     (SUBPROGRAM)           COMPILED 77.132.10  TTR=00040F    CORE=0004E0
         DTARES     (SUBPROGRAM)           COMPILED 77.133.09  TTR=00070E    CORE=000330
         GARRAY     (SUBPROGRAM)           COMPILED 77.133.09  TTR=000804    CORE=0001D0
         INPUT      (SUBPROGRAM)           COMPILED 77.139.10  TTR=00370F    CORE=000578
         LOPREL     (SUBPROGRAM)           COMPILED 77.133.09  TTR=000613    CORE=0000D8
         MAIN       (SUBPROGRAM)           COMPILED 77.139.10  TTR=003708    CORE=000240
         PRELUC     (SUBPROGRAM)           COMPILED 77.139.10  TTR=00371F    CORE=000158
         PRELUD     (SUBPROGRAM)           COMPILED 77.139.10  TTR=003007    CORE=001308
         PRELUE     (SUBPROGRAM)           COMPILED 77.132.10  TTR=000312    CORE=0000C8
         PROGPGM    (COMPOSITE PROGRAM)      LOADED 77.139.10  TTR=003807    CORE=00F828
         SDTARE     (ALIAS)                         77.133.09  TTR=00070E    CORE=000330
         STDEN      (SUBPROGRAM)           COMPILED 77.133.09  TTR=000705    CORE=0011C0
         SINPUT     (ALIAS)                         77.139.10  TTR=00370F    CORE=000578
         SKAN       (SUBPROGRAM)           COMPILED 77.138.17  TTR=002008    CORE=0005F0
         SSTDEN     (ALIAS)                         77.133.09  TTR=000705    CORE=0011C0
         SSTART     (ALIAS)                         77.139.10  TTR=003717    CORE=0008E0
         START      (SUBPROGRAM)           COMPILED 77.139.10  TTR=003717    CORE=0008E0
         VEXIT      (SUBPROGRAM)           COMPILED 77.133.09  TTR=00071C    CORE=0000E8
      31 MEMBERS, 100 DIRECTORY BLOCKS, 6 IN USE
 A TOTAL OF AT LEAST 147 CYLINDERS AND 64 TRACKS UNUSED       (ON THIS DISK PACK).


**SKAN TEST EXAMPLE
                     PRELUDE (STANDARD)                                                             LAST VAR $COMMON
                     MAXN=20                                                                        LAST VAR $COMMON
                     MIST=100                                                                       LAST VAR MAXN
                     END                                                                            LAST VAR MIST
                 *SKAN TEST DATA                                                                    LAST VAR END
                 *STANDARD DATA FOLLOWS                                                             LAST VAR PRINTABL
                 NAME=THISJOB,MAX=100,ERR=1.-4,N#MAXN                                               LAST VAR PRINTABL
                 LIST 1 NAME LIST 2 LIST 3 NTT.X END                                                LAST VAR N
                     END                                                                            LAST VAR END
                 *4 ORDIN COMMON USED       25   00000019  (BASE 10 AND BASE 16 RESP)
```

```
            *4 DYNAM COMMON USED     1900   0000076C
            *4 COMMON WORDS LEFT    40102   00009CA6

            NAME=HARDJOB N=4                                                                        LAST VAR
            A=1,-0.01,2*0         16*0 ##TO FILL OUT REST OF 1ST ROW                                LAST VAR N
            -0.01,1,-0.99,0       16*0 ##NORMALLY WE WOULD REDUCE SIZE OF A DYNAMICALLY             LAST VAR A
            0,-0.99,1,-0.01       16*0                                                              LAST VAR A
            2*0,-0.01,1                                                                             LAST VAR A
            $OPT 1,80,1$               ##CHANGE TO 80 COL CARDS                                     LAST VAR A
                      LIST 1 NAME,A,X,C LIST 2 LIST 3 NAME,NIT,ERROR,X,DELTA,GAMAVE, END            LAST VAR $OPT
            B*A                        ##SAVE A COPY OF A IN B                                      LAST VAR END
            *A BAD TRIAL SOLN FOR LATER                                                             LAST VAR B
            D=1.-3,1.+3,1.-3,1.+3                                                                   LAST VAR B
            C=4*1 X=4*1 START                                                                       LAST VAR D
DUMP      1 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  1

****** DUMP OF NAME
   NAME  (     1)
        HARDJOB

****** DUMP OF A
   A      (     1)  A      (     2)  A      (     3)  A      (     4)  A      (     5)  A      (     6)  A      (     7)  A      (     8)
 1.000000000D+00 -1.000000016D-02  0.0              0.0              0.0              0.0              0.0              0.0

   A      (    17)  A      (    18)  A      (    19)  A      (    20)  A      (    21)  A      (    22)  A      (    23)  A      (    24)
 0.0              0.0              0.0              0.0             -1.000000016D-02  1.000000000D+00 -9.900000010D-01  0.0

   A      (    41)  A      (    42)  A      (    43)  A      (    44)  A      (    45)  A      (    46)  A      (    47)  A      (    48)
 0.0             -9.900000010D-01  1.000000000D+00 -1.000000016D-02  0.0              0.0              0.0              0.0

   A      (    57)  A      (    58)  A      (    59)  A      (    60)  A      (    61)  A      (    62)  A      (    63)  A      (    64)
 0.0              0.0              0.0              0.0              0.0              0.0             -1.000000016D-02  1.000000000D+00
****** DUMP OF X
   X      (     1)  X      (     2)  X      (     3)  X      (     4)  X      (     5)  X      (     6)  X      (     7)  X      (     8)
 1.000000000D+00  1.000000000D+00  1.000000000D+00  1.000000000D+00  0.0              0.0              0.0              0.0
****** DUMP OF C
   C      (     1)  C      (     2)  C      (     3)  C      (     4)  C      (     5)  C      (     6)  C      (     7)  C      (     8)
 1.000000000D+00  1.000000000D+00  1.000000000D+00  1.000000000D+00  0.0              0.0              0.0              0.0
DUMP      1 END


DUMP      3 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  3

****** DUMP OF NAME
   NAME  (     1)
        HARDJOB

****** DUMP OF NIT
   NIT   (     1)
               6

****** DUMP OF ERROR
   ERROR (     1)
    1.234955E-08

****** DUMP OF X
   X      (     1)  X      (     2)  X      (     3)  X      (     4)  X      (     5)  X      (     6)  X      (     7)  X      (     8)
 2.020203170D+00  1.020203010D+02  1.020203010D+02  2.020203180D+00  0.0              0.0              0.0              0.0
****** DUMP OF DELTA
   DELTA  (     1)  DELTA  (     2)  DELTA  (     3)  DELTA  (     4)  DELTA  (     5)  DELTA  (     6)  DELTA  (     7)  DELTA  (     8)
-4.989713020D-06 -2.519807740D-06 -2.494859180D-06 -2.494859590D-08  0.0              0.0              0.0              0.0
****** DUMP OF GAMAVE
   GAMAVE(     1)  GAMAVE(     2)  GAMAVE(     3)  GAMAVE(     4)  GAMAVE(     5)  GAMAVE(     6)  GAMAVE(     7)  GAMAVE(     8)
 1.716979450D-01  3.300330060D-01  3.333333380D-03  0.0              0.0              0.0              0.0              0.0
DUMP      3 END

            X=#2,2*1.-10               ##TRIAL SOLN 1ST 2 ELMS EXACT, REST 1.-10                    LAST VAR
            START                                                                                   LAST VAR X
DUMP      1 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  1
****** DUMP OF NAME
```

```
   NAME  (     1)
          HARDJOB
 ****** DUMP OF A
   A        (     1)    A        (     2)    A        (     3)    A        (     4)    A        (     5)    A        (     6)    A        (     7)    A        (     8)
 1.000000000D+00    -1.000000016D-02    0.0                 0.0                 0.0                 0.0                 0.0                 0.0
   A        (    17)    A        (    18)    A        (    19)    A        (    20)    A        (    21)    A        (    22)    A        (    23)    A        (    24)
 0.0                 0.0                 0.0                 0.0                -1.000000016D-02    1.000000000D+00    -9.900000010D-01    0.0
   A        (    41)    A        (    42)    A        (    43)    A        (    44)    A        (    45)    A        (    46)    A        (    47)    A        (    48)
 0.0                -9.900000010D-01    1.000000000D+00    -1.000000016D-02    0.0                 0.0                 0.0                 0.0
   A        (    57)    A        (    58)    A        (    59)    A        (    60)    A        (    61)    A        (    62)    A        (    63)    A        (    64)
 0.0                 0.0                 0.0                 0.0                 0.0                 0.0                -1.000000016D-02    1.000000000D+00
 ****** DUMP OF X
   X        (     1)    X        (     2)    X        (     3)    X        (     4)    X        (     5)    X        (     6)    X        (     7)    X        (     8)
 2.020203170D+00    1.020203010D+02    9.999999440D-11    9.999999440D-11    0.0                 0.0                 0.0                 0.0
 ****** DUMP OF C
   C        (     1)    C        (     2)    C        (     3)    C        (     4)    C        (     5)    C        (     6)    C        (     7)    C        (     8)
 1.000000000D+00    1.000000000D+00    1.000000000D+00    1.000000000D+00    0.0                 0.0                 0.0                 0.0
 DUMP       1 END


 DUMP       3 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  3
 ****** DUMP OF NAME
   NAME  (     1)
          HARDJOB
 ****** DUMP OF NTT
   NTT   (     1)
               7
 ****** DUMP OF ERROR
   ERROR (     1)
    4.543632E-08
 ****** DUMP OF X
   X        (     1)    X        (     2)    X        (     3)    X        (     4)    X        (     5)    X        (     6)    X        (     7)    X        (     8)
 2.020203180D+00    1.020203010D+02    1.020203010D+02    2.020203180D+00    0.0                 0.0                 0.0                 0.0
 ****** DUMP OF DELTA
   DELTA (     1)    DELTA (     2)    DELTA (     3)    DELTA (     4)    DELTA (     5)    DELTA (     6)    DELTA (     7)    DELTA (     8)
 1.835809740D-05    9.270849590D-06    9.179059090D-06    9.179060600D-08    0.0                 0.0                 0.0                 0.0
 ****** DUMP OF GAMAVE
   GAMAVE(     1)    GAMAVE(     2)    GAMAVE(     3)    GAMAVE(     4)    GAMAVE(     5)    GAMAVE(     6)    GAMAVE(     7)    GAMAVE(     8)
 1.683335220D-01    3.300330060D-01    3.333333880D-03    0.0                 0.0                 0.0                 0.0                 0.0
 DUMP       3 END

                      ELM(1,2)A=-0.0075,ELM(2,1)A=-0.0075 X*D START                                          LAST VAR
 DUMP       1 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  1
 ****** DUMP OF NAME
   NAME  (     1)
          HARDJOB
 ****** DUMP OF A
   A        (     1)    A        (     2)    A        (     3)    A        (     4)    A        (     5)    A        (     6)    A        (     7)    A        (     8)
 1.000000000D+00    -7.500000300D-03    0.0                 0.0                 0.0                 0.0                 0.0                 0.0
   A        (    17)    A        (    18)    A        (    19)    A        (    20)    A        (    21)    A        (    22)    A        (    23)    A        (    24)
 0.0                 0.0                 0.0                 0.0                -7.500000300D-03    1.000000000D+00    -9.900000010D-01    0.0
   A        (    41)    A        (    42)    A        (    43)    A        (    44)    A        (    45)    A        (    46)    A        (    47)    A        (    48)
 0.0                -9.900000010D-01    1.000000000D+00    -1.000000016D-02    0.0                 0.0                 0.0                 0.0
   A        (    57)    A        (    58)    A        (    59)    A        (    60)    A        (    61)    A        (    62)    A        (    63)    A        (    64)
 0.0                 0.0                 0.0                 0.0                 0.0                 0.0                -1.000000016D-02    1.000000000D+00
 ****** DUMP OF X
   X        (     1)    X        (     2)    X        (     3)    X        (     4)    X        (     5)    X        (     6)    X        (     7)    X        (     8)
```

```
  9.999999931D-04  1.000000000D+03  9.999999931D-04  1.000000000D+03  0.0             0.0             0.0             0.0

****** DUMP OF C
  C      (     1)  C      (     2)  C      (     3)  C      (     4)  C      (     5)  C      (     6)  C      (     7)  C      (     8)
  1.000000000D+00  1.000000000D+00  1.000000000D+00  1.000000000D+00  0.0             0.0             0.0             0.0

DUMP      1 END


DUMP      3 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  3

****** DUMP OF NAME
   NAME  (     1)
         HARDJOB

****** DUMP OF NTT
   NTT   (     1)
               7

****** DUMP OF ERROR
   ERROR (     1)
    4.362761E-09

****** DUMP OF X
  X      (     1)  X      (     2)  X      (     3)  X      (     4)  X      (     5)  X      (     6)  X      (     7)  X      (     8)
  1.762507390D+00  1.016674470D+02  1.016711390D+02  2.016711560D+00  0.0             0.0             0.0             0.0

****** DUMP OF DELTA
  DELTA  (     1)  DELTA  (     2)  DELTA  (     3)  DELTA  (     4)  DELTA  (     5)  DELTA  (     6)  DELTA  (     7)  DELTA  (     8)
 -2.346247420D-06 -8.886419630D-07 -8.798435330D-07 -8.798436650D-09  0.0             0.0             0.0             0.0

****** DUMP OF GAMAVE
  GAMAVE (     1)  GAMAVE (     2)  GAMAVE (     3)  GAMAVE (     4)  GAMAVE (     5)  GAMAVE (     6)  GAMAVE (     7)  GAMAVE (     8)
  1.262501260D-01  3.300330060D-01  3.333333880D-03  0.0             0.0             0.0             0.0             0.0

DUMP      3 END

                 A#R                          ##RESTORE ORIG DATA                                       LAST VAR
                 ROW(4)A=0 0 -0.0075,1  COL(4)A=0 0 -0.0075,1                                           LAST VAR A
                 C.ELM(4)C=1,X#D START    ##C NO DATA = 20*0 AS DATA FILL SET ON                        LAST VAR SKANEXTV

DUMP      1 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  1

****** DUMP OF NAME
   NAME  (     1)
         HARDJOB

****** DUMP OF A
  A      (     1)  A      (     2)  A      (     3)  A      (     4)  A      (     5)  A      (     6)  A      (     7)  A      (     8)
  1.000000000D+00 -1.000000160D-02  0.0             0.0             0.0             0.0             0.0             0.0

  A      (    17)  A      (    18)  A      (    19)  A      (    20)  A      (    21)  A      (    22)  A      (    23)  A      (    24)
  0.0             0.0             0.0             0.0             -1.000000160D-02 1.000000000D+00 -9.900000100D-01 0.0

  A      (    41)  A      (    42)  A      (    43)  A      (    44)  A      (    45)  A      (    46)  A      (    47)  A      (    48)
  0.0             -9.900000100D-01 1.000000000D+00 -7.500000300D-03 0.0             0.0             0.0             0.0

  A      (    57)  A      (    58)  A      (    59)  A      (    60)  A      (    61)  A      (    62)  A      (    63)  A      (    64)
  0.0             0.0             0.0             0.0             0.0             0.0             -7.500000300D-03 1.000000000D+00

****** DUMP OF X
  X      (     1)  X      (     2)  X      (     3)  X      (     4)  X      (     5)  X      (     6)  X      (     7)  X      (     8)
  9.999999310D-04  1.000000000D+03  9.999999310D-04  1.000000000D+03  0.0             0.0             0.0             0.0

****** DUMP OF C
  C      (     1)  C      (     2)  C      (     3)  C      (     4)  C      (     5)  C      (     6)  C      (     7)  C      (     8)

  0.0             0.0             0.0             1.000000000D+00  0.0             0.0             0.0             0.0

DUMP      1 END


DUMP      3 OF NON-ZERO ELEMENTS OF COMMON (OR LABELLED COMMON) -SEGMENT  3

****** DUMP OF NAME
   NAME  (     1)
         HARDJOB
```

```
****** DUMP OF NIT
   NIT    (      1)
                 7

****** DUMP OF ERROR
   ERROR  (      1)
    3.266262E-11

****** DUMP OF X
    X       (      1)   X       (      2)   X       (      3)   X       (      4)   X       (      5)   X       (      6)   X       (      7)   X       (      8)
  3.760687100-03  3.760686480-01  3.798293310-01  1.002848720+00  0.0             0.0             0.0             0.0

****** DUMP OF DELTA
   DELTA  (      1)   DELTA  (      2)   DELTA  (      3)   DELTA  (      4)   DELTA  (      5)   DELTA  (      6)   DELTA  (      7)   DELTA  (      8)
  8.754135060-09  4.411290010-09  4.367422830-09  3.275568170-11  0.0             0.0             0.0             0.0

****** DUMP OF GAMAVE
   GAMAVE(      1)   GAMAVE(      2)   GAMAVE(      3)   GAMAVE(      4)   GAMAVE(      5)   GAMAVE(      6)   GAMAVE(      7)   GAMAVE(      8)
  1.679697410-01  3.300185670-01  2.500000100-03  0.0             0.0             0.0             0.0             0.0

DUMP      3 END

                 END                          **BEGINS AFRESH                                  LAST VAR
**SKAN TEST EXAMPLE

                     PRELUDE (STANDARD)                                                        LAST VAR $COMMON
                     MAXN=20                                                                   LAST VAR $COMMON
                     MIST=100                                                                  LAST VAR MAXN
                     END                                                                       LAST VAR MIST
                 PRELUDE PRINTABLE END                                                         LAST VAR END
                     $COMMON                                                                   LAST VAR $COMMON
                 *   SKAN TEST EXAMPLE (SOLVES AX=C USING MINT)                                LAST VAR $COMMON
                     REAL*8 NAME,GAMMA,RTEMP                                                   LAST VAR $COMMON
                     COMMON                                                                    LAST VAR $COMMON
                    1 MIST,MAXN,MAXLOT,N                                                       LAST VAR $COMMON
                    2,NAME,GAMMA,RTEMP,MAX,ERR,NIT,ERROR                                       LAST VAR $COMMON
                    3,IROW,ICOL,INROW,INCOL,ISTOP                                              LAST VAR $COMMON
                    X,END(6)                                                                   LAST VAR $COMMON
                     REAL*8 THELOT(MAXLOT),A(MAXN,MAXN),B(MAXN,MAXN),C(MAXN),D(MAXN)           LAST VAR $COMMON
                    1,X(MAXN),DELTA(MAXN),GAMAVE(MAXN)                                         LAST VAR $COMMON
                     DIMENSION LIST(MIST)                                                      LAST VAR $COMMON
                     COMMON                                                                    LAST VAR $COMMON
                    1 LIST#THELOT,A,B,C,D,X,DELTA,GAMAVE                                       LAST VAR $COMMON
                     $                                                                         LAST VAR $COMMON
                 $MOD(NAME,,,4),(ROW,IROW,4,1,1),(COL,ICOL,3,1,2),(ELM,IROW,4,1,3)             LAST VAR $COMMON
                 (START,,,,4),(STOP,ISTOP,#,#,5)             $                                 LAST VAR $MOD
                 $MOD(LIST,,,,9991),(TO,,,,9992)$                                              LAST VAR $MOD

                 $OPT 0,0,9999$                                                                LAST VAR $MOD
                 *FILLS OUT ALL DATA                                                           LAST VAR $OPT
                 PRINTABLE                                                                     LAST VAR $OPT
                 *STANDARD DATA FOLLOWS                                                        LAST VAR PRINTABL
                 NAME=THISJOB,MAX=100,ERR=1.-4,N#MAXN                                          LAST VAR PRINTABL
                 LIST 1 NAME LIST 2 LIST 3 NIT,X END                                           LAST VAR N
                     END                                                                       LAST VAR END
```

TABLE USED WITH DTAIN FOR SCANNING DATA

| SEQ NO | VAR NAME | BASE ADDRESS COMMON | ABS(7) | LENGTH *4 WORDS | MODE | OPTION | VALUE |
|---|---|---|---|---|---|---|---|
| 1 | PRINTABL | 0 | 00000000 | 0 | 0 | 9994 | |
| 2 | END | 20 | 00144714 | 6 | 0 | 9993 | |
| 3 | ENDOFILE | 0 | 00000000 | 0 | 0 | 9990 | |
| 4 | NAME | 5 | 001446D8 | 2 | 4 | 0 | |
| 5 | GAMMA | 7 | 001446E0 | 2 | 3 | 0 | |
| 6 | RTEMP | 9 | 001446E8 | 2 | 3 | 0 | |
| 7 | MIST | 1 | 001446C8 | 1 | 0 | 0 | 100 |
| 8 | MAXN | 2 | 001446CC | 1 | 0 | 0 | 20 |
| 9 | MAXLOT | 3 | 001446D0 | 1 | 0 | 0 | 900 |
| 10 | N | 4 | 001446D4 | 1 | 1 | 0 | 20 |
| 11 | MAX | 11 | 001446F0 | 1 | 1 | 0 | 100 |
| 12 | ERR | 12 | 001446F4 | 1 | 2 | 0 | |
| 13 | NIT | 13 | 001446F8 | 1 | 1 | 0 | |
| 14 | ERROR | 14 | 001446FC | 1 | 2 | 0 | |
| 15 | IROW | 15 | 00144700 | 1 | 1 | 0 | |
| 16 | ICOL | 16 | 00144704 | 1 | 1 | 0 | |
| 17 | INROW | 17 | 00144708 | 1 | 1 | 0 | |
| 18 | INCOL | 18 | 0014470C | 1 | 1 | 0 | |

```
19  ISTOP          19  00144710      1     1     0
20  THELOT       9889  0014F148   1800     3     0
21  A            9889  0014F148    800     3     0
22  B           10689  0014FDC8    800     3     0
23  C           11489  0014FA48     40     3     0
24  D           11529  0014FAE8     40     3     0
25  X           11569  0014FB88     40     3     0
26  DELTA       11609  0014FC28     40     3     0
27  GAMAVE      11649  0014FCC8     40     3     0
28  LIST         9789  0014DFB8    100     1  9991
29  ROW            15  00144700      4     1     1
30  COL            16  00144704      3     1     2
31  ELM            15  00144700      4     1     3
32  START           0  00000000      0     0     4
33  STOP           19  00144710      1     1     5
34  TO              0  00000000      0     0  9992
    *LAST COM   11688  00002DA8

*4 ORDIN COMMON USED       25  00000019  (BASE 10 AND BASE 16 RESP)
*4 DYNAM COMMON USED     1900  0000076C
*4 COMMON WORDS LEFT    40102  00009CA6

*ONLY GIVES TABLE LIST BUT NORMALLY WE WOULD HAVE FRESH DATA          LAST VAR END
STOP 2                                                                LAST VAR END
**SKAN TEST EXAMPLE STOP  2
```