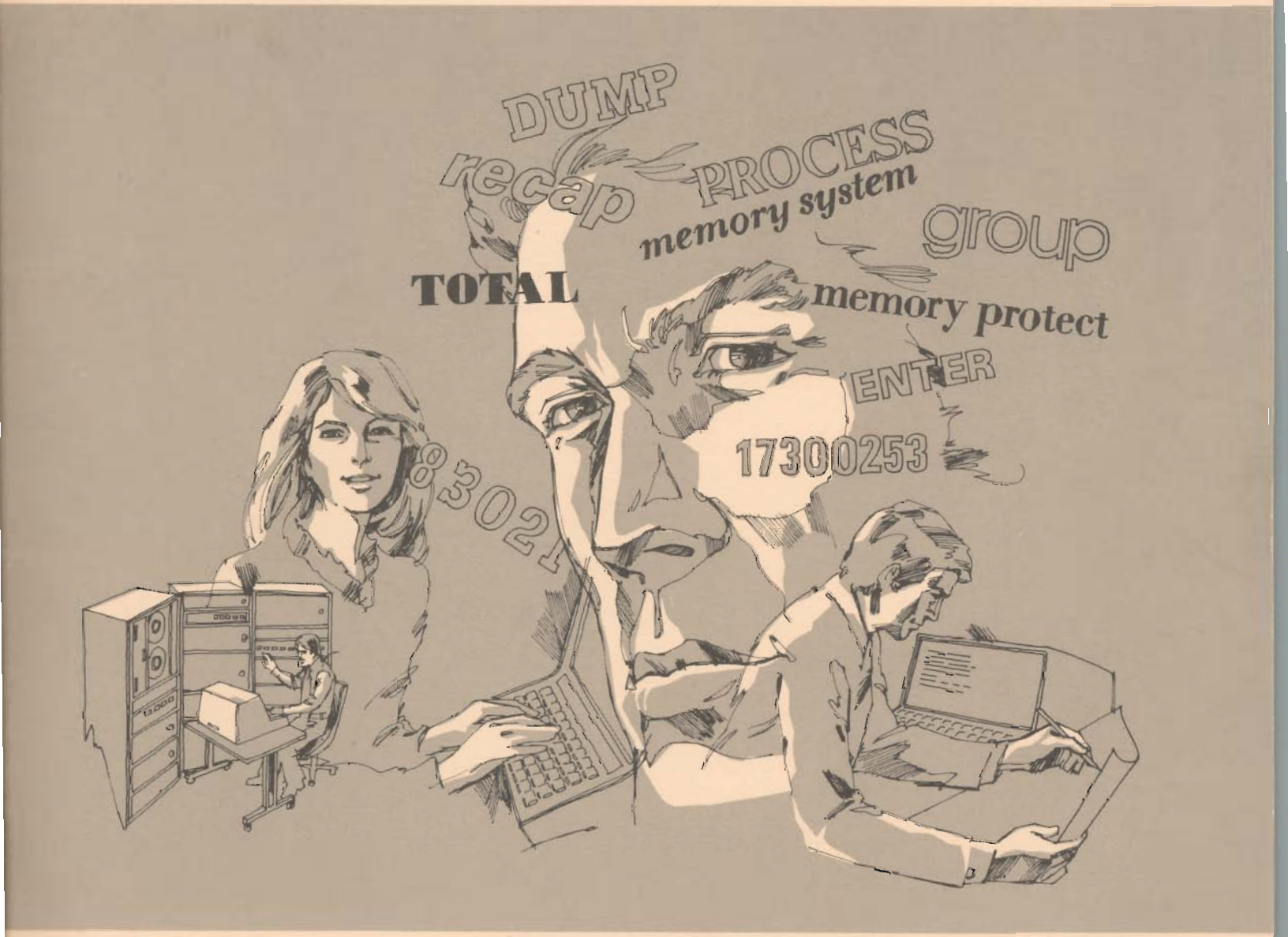
BASIC/3000

Programming Examples



HEWLETT hp PACKARD

5952-4646

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

FORWARD

The examples in this book represent typical programs as written by typical BASIC users. BASIC/3000's advanced language extensions were used when they appeared appropriate. Some of the examples resemble sample programs demonstrated by other mini-computer manufacturers. These are included to give comparative programming techniques of BASIC/3000.

BASIC/3000 allows and encourages a "STRUCTURED" programming approach, but does not enforce this upon the user that has developed an alternative style. BASIC/3000 constructions maximize the readability of finished programs by other persons.

Each program emphasizes a point, as well as shows the language necessary to achieve the result. When in doubt as to the meaning of a statement, refer to the BASIC/3000 Programmers Pocket Guide.

TABLE OF CONTENTS

PREFACE

Some of the major features of BASIC/3000 are:

Types of Variables

- Four Types . . . simultaneously
  - Integer
  - Real
  - Long
  - Complex
- Simple Definition

Call FORTRAN, COBOL, SPL, SUBROUTINES

Superior File Capabilities

- Consistent SYNTAX for ease of use
- Three File Types, each with Direct and Sequential access
- Runtime redefinition
- All systems resources available (tape, line printer, etc.)

Powerful Use of Multi-Programming Executive

Advanced Language Syntax

Extensive Built-in Functions

I. RANGE – Numeric Range of BASIC/3000.

RANGE computes the area of a circle with its radius increasing in steps by the square root of 10. Note that the last area calculated (1.5708×10^{76}) is near the limit of BASIC/3000, yet precision is fully maintained.

```
>LIST
RANGE
  10 PRINT LIN(2),"RADIUS","AREA",LIN(1)
  20 A=.5
  30 A=10*A
  40 PRINT SQR(A),PIX(A)
  50 GOTO 30
>RUN
RANGE


RADIUS          AREA

 2.23607         15.708
 7.07107         157.08
 22.3607         1570.8
 70.7107         15708.
 223.607         157080.
 707.107         1.57080E+06
   ●               ●
   ●               ●
   ●               ●
 2.23607E+36     1.57080E+73
 7.07107E+36     1.57080E+74
 2.23607E+37     1.57080E+75
 7.07107E+37     1.57080E+76
 2.23607E+38
WARNING: OVERFLOW - RESULT= + OR - 1E77 IN LINE 40 IN RANGE
                 1.15792E+77

*CONTROL-Y BREAK IN LINE 40 IN RANGE
>
```

II. SAMPLE – Size of User Space With BASIC/3000.

SAMPLE SORT demonstrates the size of available user space (32K words total). This is a simple sort* program that will sort a large number of records in core. It also demonstrates the following features:

LINE 5	• Integers are simply referenced
LINE 10	• String array is dimensioned at 2500 records of 20 characters each (11 words each X2500 = 27,500 words)
LINE 80	• CONVERT and RND functions
LINE 760	• Multiple assignment statements per line


```
>LIST
SAMPLE
    5 INTEGER C,I,J,P
   10 DIM C$[20],A$[2500,20]
   40 I=RND(-1)
   50 INPUT "TEST SORT HOW MANY(MAX 2500)?  ",C
   70 FOR J=1 TO C
   80   CONVERT INT(RND(1)*10000)+10000 TO A$[J]
  100 NEXT J
  110 PRINT C;"RANDOM NUMBERS, HAVE BEEN SET UP--SORT STARTS..."
  699 REDIM A$[C]
  700 FOR P=1 TO C-1
  720   FOR I=2 TO C+1-P
  725     J=I-1
  750     IF A$[I]>A$[J] THEN 810
  760     C$=A$[I],A$[I]=A$[J],A$[J]=C$
  810   NEXT I
  820 NEXT P
  830 MAT PRINT A$;
>
```

*Actual sorts of large data files would be accomplished using the extremely efficient SORT/3000 subsystem, rather than an inefficient sort of this type.

III. DIGITS – Precision of BASIC/3000.

DIGITS demonstrates the following features

LINE 100	• 12 digit long-precision variables may be declared as necessary, the system in *not* dedicated to LONG.
LINE 110	• BASIC/3000 automatically continues program lines, as necessary, for the list device.
LINE 120	• The ":" in the input statement allows input to be buffered—the user is asked for input only as necessary.
	• ('10'10'10) shows a way special characters can be put into an output string. ('10 is a line feed.)
	• The input statement allows an output question to be inserted.
LINE 130	• One form of formatted print statement is shown.
	• The "SQR(N)" routine is automatically made LONG precision by it's LONG precision argument.
CONTROL-Y	• Suspends any program unless the feature is disabled. "RESUME" would continue operation.
	• "ABORT" terminates the run of the program.

```
>LIST
DIGITS
 100 LONG N
 110 PRINT &
     "PROGRAM PRINT SQUARE ROOTS TO 12 DECIMAL DIGITS OF PRECISION"
 120 INPUT :'10'10'10"YOUR NUMBER(S)",N:
 130 PRINT USING "15A,5D.16D";"SQUARE ROOT IS ",SQR(N)
 150 GOTO 120
>RUN
DIGITS
PROGRAM PRINT SQUARE ROOTS TO 12 DECIMAL DIGITS OF PRECISION


YOUR NUMBER(S)2
SQUARE ROOT IS      1.4142135623730000


YOUR NUMBER(S)20000,2000000
SQUARE ROOT IS    141.4213562375000000
SQUARE ROOT IS   1414.2135623730000000


YOUR NUMBER(S)625,624.99999999
SQUARE ROOT IS     25.0000000000000000
SQUARE ROOT IS     24.9999999998300000
```

```
YOUR NUMBER(S)9,9.0000000001
SQUARE ROOT IS      3.0000000000000000
SQUARE ROOT IS      3.0000030000140000


YOUR NUMBER(S)

*CONTROL-Y BREAK IN LINE 120 IN DIGITS
>
```

IV. INDEX – Indexed Sequential With BASIC/3000.

INDEX is an example of a simple indexed sequential access method that could be used effectively with a very large master parts file. The program uses an index file "IX1Ø" to point to parts (with subassemblies) in master file "PROD1Ø". It demonstrates the following features:

MAIN PROGRAM
LINES 100–330

- Several constructs of IF-THEN, IF—THEN DO, and ELSE statements.
- PRINT USING statements.
- String and numeric function calls.
- FOR loop in PRINT statement.
- CONVERT statement.
- ROW function.

```
>LIST
INDEX
 100 REM     ****MANUFACTURING EXAMPLE--INDEXED SEQUENTIAL ACCESS****
 110 REM
 120 FILES IX10,PROD10
 130 DIM T6$[3],S[5]
 140 PRINT USING 150
 150 IMAGE //"SAMPLE INDEXED SEQUENTIAL SEARCH"//"TYPE '0' WHEN FINISHE&
D..."/
 160 INPUT :'10'10'10"PART NUMBER(S):  ",T6:
 170 IF T6<>0 THEN DO
 180    CONVERT T6 TO T6$
 190    T6$=FNL$(T6$)
 200    R=FNM(T6$,1)
 210    IF R THEN DO
 220      P=FNA(2,R,T6$,S[*])
 230      IF P THEN PRINT USING 320;T6$,R,(FOR K1=1 TO ROW(S),S[K1])
 240      ELSE PRINT USING 310;T6$
 250    DOEND
 260    ELSE PRINT USING 330;T6$
 270    GOTO 160
 280 DOEND
 290 PRINT USING 300
 300 IMAGE /"PNUM,  PART NUMBER;"/"REC,   RECORD NUMBER ADDRESS."
 310 IMAGE 2X,3A,X,"IS NOT IN THE FILE."
 320 IMAGE "PNUM(",3A,"), REC(",2D,"), SUBASSEMBLY (",4(D,X),D,")"
 330 IMAGE 2X,3A,X,"IS NOT IN THE INDEX."
```

INDEX SEQUENTIAL SCAN ROUTINES

LINES 9000–9530

- Sample of numeric functions.
- Direct and sequential single variable and matrix file I/O.
- IF—THEN—ELSE DO statements.

```
9000 REM
9010 REM
9020 REM INDEXED SEQUENTIAL SCAN ROUTINES.
9030 REM
9040 REM SCAN ROUTINE.
9050 REM FNA(F,R6,P4$,S2(*)).
9060 REM R6,  START SCAN AT RECORD NUMBER R6;
9070 REM P4$,  TARGET PRODUCT NUMBER;
9080 REM Z9$,  PRODUCT NUMBER CURRENTLY ACCESSED;
9090 REM S2(*),  SUBASSEMBLY TABLE.
9100 DEF FNA(F,R6,P4$,S2[*])
9110   DIM Z9$[3]
9120   IF R6=0 THEN RETURN 0
9130   RESTORE #F
9140   READ #F,R6;Z9$
9150   Z9$=FNL$(Z9$)
9160   MAT READ #F;S2
9170   REM
9180   REM SCAN
9190   IF P4$=Z9$ THEN RETURN 1
9200   ELSE DO
9210     IF P4$>Z9$ THEN DO
9220       REM
9230       REM GET THE NEXT RECORD
9240       READ #F;Z9$
9250       Z9$=FNL$(Z9$)
9260       MAT READ #F;S2
9270       GOTO 9190
9280     DOEND
9290     ELSE DO
9300       REM
9310       REM P4$ IS NOT IN THE FILE
9320       MAT S2=ZER
9330       RETURN 0
9340     DOEND
9350   DOEND
9360 FNEND
9370 REM
9380 REM INDEX ROUTINE.
9390 DEF FNM(M2$,F)
9400   DIM P1$[3]
9410   REAL R,I1
9420   ON END #F THEN 9510
9430   RESTORE #F
9440   READ #F;P1$,R
9450   IF M2$<P1$ THEN RETURN 0
9460   READ #F;P1$
9470   FOR I1=1 TO 1 STEP 0
9480     IF M2$<P1$ THEN RETURN R
9490     READ #F;R,P1$
9500   NEXT I1
9510   IF M2$=P1$ THEN RETURN R
9520   ELSE RETURN 0
9530 FNEND
```

RIGHT JUSTIFY STRING ROUTINE

LINES 9540–9620

- Sample of a string function.
- DEB$, LEN functions.
- Normal string operations.

```
9540 REM
9550 REM RIGHT JUSTIFY STRING ROUTINE.
9560 DEF FNL$(Z6$)
9570   DIM T1$[3],S1$[3]
9580   S1$=DEB$(Z6$)
9590   T1$[1;3]=" "
9600   T1$[4-LEN(S1$)]=S1$
9610   RETURN T1$
9620 FNEND
>
```

Sample RUN of INDEX

```
>RUN
INDEX


SAMPLE INDEXED SEQUENTIAL SEARCH

TYPE '0' WHEN FINISHED...



PART NUMBER(S):  102
PNUM(102), REC( 2), SUBASSEMBLY (5 1 1 4 1)



PART NUMBER(S):  993,502,50,602,5,258,0
PNUM(993), REC(13), SUBASSEMBLY (8 0 4 6 9)
  502 IS NOT IN THE FILE.
PNUM( 50), REC( 1), SUBASSEMBLY (8 0 8 4 0)
PNUM(602), REC( 8), SUBASSEMBLY (2 0 5 9 1)
    5 IS NOT IN THE INDEX.
PNUM(258), REC( 3), SUBASSEMBLY (9 6 5 4 4)

PNUM,  PART NUMBER;
REC,   RECORD NUMBER ADDRESS.

>
```

V. FNP$ – "COBOL Plus" Formatting With BASIC/3000.

PICTURE FORMATTING

The function FNP$ is used to edit a integer number (data type INTEGER, REAL OR LONG), up to 11 digits if LONG (called "D"). The function edits according to a format mask (called "F$") and returns edited results in FNP$.

PICTURE FORMAT FORM

Format masks are used to edit the integer number D. Format masks consist of placement holders, sign characters and insertion characters.

REPLACEMENT

9 (NUMERIC REPLACEMENT HOLDER).

Each 9 of F$ is replaced by a decimal digit in the corresponding position of D; result returns as FNP$.

Z (ZERO SUPPRESSION REPLACEMENT HOLDER).

The position of the Z in F$ is replaced by a decimal digit in the corresponding position of D. Zeros to the left of the first significant position in F$ are replaced by blanks, results return as FNP$.

* (ASTERISK REPLACEMENT HOLDER).

Asterisks rather than blanks are inserted to the left of the first significant decimal digit in the FNP$ result.

$ (DOLLAR SIGN REPLACEMENT HOLDER).

A dollar sign is inserted to the left of the first significant decimal digit in the FNP$ result, and is to the left of the position that defined the zero suppression. Any zero in the remaining non-significant positions are replaced by blanks.

SIGN CHARACTER

DB (DEBIT).

These two characters are placed in the right-most position of the format mask F$. If D is positive, the characters remain in the edited output. If F$'s value is negative, DB is replaced by two blanks. When DB is present in F$, no data is edited into the last two positions but only into the edit characters to the left.

CR (CREDIT)

These two characters are placed in the rightmost positions of the format mask, F$. If D is negative, the characters remain in the edited output. If F$'s value is positive, CR is replaced by two blanks. When CR is present in F$, no data is edited into the last two positions but only into the edit characters to the left.

SS (DEBIT OR CREDIT: DB, CR).

These two characters are placed in the rightmost position of the format F$. If D is positive, SS is replaced with DB. If D is negative, SS is replaced with CR. When SS is present in F$, no data is edited into the last two positions but only into the edit characters to the left.

+ (PLUS)

This character placed in the rightmost position of F$ is treated similarly to DB.

– (MINUS)

This character placed in the rightmost position of F$ is treated similarly to CR.

S (PLUS OR MINUS: +, –).

This character placed in the rightmost position F$ is treated similarly to SS.

NONE OF THE ABOVE. A (–) minus in D remains in the position in formatted output.

INSERTION CHARACTERS

All other characters in F$ not defined about are insertion characters and remain in FNP$ result.

FNP$ OPERATION

FNP$ formatting requires passing an integer number (Integer, Real, or Long) in D; a format in F$ when calling FNP$. The formatted output is returned.

Example:

```
600 D="-346789"
610 F$="$$$$,$$$,ZZZ.99 CR"
630 PRINT USING "19A";FNP$(D,F$)
```

Result:

```
(        $3,467.89 CR)
```

NOTE:
* The characters returned in FNP$ using F$ mask from right to left. Only the characters 9, Z, *, and S are replaced by numbers.
* If all numeric characters have not been placed in the edit mask when the end of the edit mask is reached, the entire edited output is filled with asterisks and editing terminates.

FNPS LISTING

A listing of FNP$ follows. FNP$ is fairly long, but then this formatting is the most extensive available. Large segments of code can be removed if the user does not want all the features. For example, if one wants only the sign character option of 'NONE OF THE ABOVE' just DELETE 9034-9110.

```
>LIST
DN
9000 DEF FNP$(LONG D,F$[*])
9002   INTEGER D4,D5,D6,D7,D8,D9
9004   DIM P$[30],D$[20]
9006   REM EDIT D (A LONG PRECISION INTEGER) WITH FORMAT IN F$
9008   REM  RETURN FNP$ AS EDITED RESULT.
9010   IF D>9.9999999999L10 OR D<-9.9999999999L10 THEN PRINT &
     "MAX NUMBER EXCEEDED"
9012   CONVERT D TO D$
9014   D6=POS(D$,"L"),D7=POS(D$,".")
9016   IF D7=0 THEN D7=D6
9018   CONVERT D$[D6+1] TO D9
9020   IF D$[1,1]="-" THEN D9=D9+1
9022   D$=D$[1,D7-1]+D$[D7+1;(D6-D7-1 MAX 0)]
9024   D$[1,D9+1]=D$+"00000000000"
9026   P$=F$
9028   D4=1
9030   D5=LEN(P$)
9032   D6=LEN(D$)
9034   REM DB SIGN:    'DB' OR  BLANK
9036   IF D5-1<1 THEN 9114
9038   IF P$[D5-1,D5]="DB" THEN 9042
9040   GOTO 9050
9042   IF D$[1,1]<>"-" THEN 9114
9044   P$[D5-1,D5]="  "
9046   GOTO 9110
9048   REM  CR SIGN:  BLANK OR 'CR'
9050   IF P$[D5-1,D5]="CR" THEN 9054
9052   GOTO 9062
9054   IF D$[1,1]="-" THEN 9110
9056   P$[D5-1,D5]="  "
9058   GOTO 9114
9060   REM  + SIGN:   '+' OR BLANK
9062   IF P$[D5,D5]="+" THEN 9066
9064   GOTO 9074
9066   IF D$[1,1]<>"-" THEN 9114
9068   P$[D5,D5]=" "
9070   GOTO 9110
9072   REM  - SIGN:  BLANK OR '-'
9074   IF P$[D5,D5]="-" THEN 9078
9076   GOTO 9086
9078   IF D$[1,1]="-" THEN 9110
9080   P$[D5,D5]=" "
9082   GOTO 9114
9084   REM SS SIGN:       'DB' OR 'CR'
9086   IF P$[D5-1,D5]<>"SS" THEN 9100
9088   IF D$[1,1]="-" THEN 9094
9090   P$[D5-1,D5]="DB"
9092   GOTO 9114
9094   P$[D5-1,D5]="CR"
9096   GOTO 9110
9098   REM  S SIGN:   '+' OR '-'
9100   IF P$[D5,D5]<>"S" THEN 9114
9102   IF D$[1,1]="-" THEN 9108
9104   P$[D5,D5]="+"
```

```
9106    GOTO 9114
9108    P$[D5,D5]="-"
9110    D4=2
9112    REM LOAD D$ INTO P$ FROM RIGHT TO LEFT
9114    FOR D8=D5 TO 1 STEP -1
9116      IF D6<D4 THEN 9142
9118      IF P$[D8,D8]="Z" THEN 9134
9120      IF P$[D8,D8]="$" THEN 9134
9122      IF P$[D8,D8]="9" THEN 9128
9124      IF P$[D8,D8]="*" THEN 9134
9126      GOTO 9138
9128      IF D$[D6,D6]<>"-" THEN 9134
9130      P$[D8,D8]="0"
9132      GOTO 9138
9134      P$[D8,D8]=D$[D6,D6]
9136      D6=D6-1
9138    NEXT D8
9140    REM WITH D$ NOW IN P$--FIXUP P$ TO LEFT TO FIRST '$' IF ANY
9142    FOR D9=D8 TO 1 STEP -1
9144      IF P$[D9,D9]<>"Z" THEN 9150
9146      P$[D9,D9]=" "
9148      GOTO 9188
9150      IF P$[D9,D9]<>"9" THEN 9156
9152      P$[D9,D9]="0"
9154      GOTO 9188
9156      IF P$[D9,D9]<>"." THEN 9170
9158      IF D9+1>D5 THEN 9162
9160      IF P$[D9+1,D9+1]<>" " THEN 9188
9162      IF D9-1<1 THEN 9188
9164      IF P$[D9-1,D9-1]="$" THEN 9188
9166      P$[D9,D9]=P$[D9-1,D9-1]
9168      GOTO 9144
9170      IF P$[D9,D9]<>"," THEN 9182
9172      IF D9-1>0 THEN 9178
9174      P$[D9,D9]=" "
9176      GOTO 9188
9178      P$[D9,D9]=P$[D9-1,D9-1]
9180      GOTO 9144
9182      IF P$[D9,D9]="*" THEN 9188
9184      IF P$[D9,D9]<>"$" THEN 9188
9186      GOTO 9192
9188    NEXT D9
9190    REM  FIXUP P$ TO THE LEFT OF FIRST $
9192    FOR D7=1 TO D9-1
9194      IF P$[D7,D7]="$" THEN 9200
9196      IF P$[D7,D7]="," THEN 9200
9198      GOTO 9202
9200      P$[D7,D7]=" "
9202    NEXT D7
9204    REM FILL WITH '*' IF NUMBER EXCEEDED FORMAT
9206    IF D6<D4 THEN 9214
9208    FOR D7=1 TO D5
9210      P$[D7]="*"
9212    NEXT D7
9214    RETURN P$
9216  FNEND
>
```

VI. JOURNAL – Use of BASIC/3000 in DP Shop.

JOURNAL demonstrates the type of program that might be used to replace traditional key-punch operation in a data processing shop. JOURNAL provides rapid on-line input in a user oriented form. It also provides character, size and some logical error checking while producing an ASCII file. This formatted ASCII is a card deck replacement for input for a COBOL system. The Program is designed to use a 10-Key accounting keyboard for all JOURNAL entries and includes some user written SPL subroutines that facilitate this operation.

JOURNAL demonstrates the following features:

Line 190	CONVERT numeric to ASCII
Line 310	SYSTEM statement to BUILD a non-BASIC ASCII file
Line 360	SYSTEM statement used to PURGE a file
Line 410	File assignment statement
Line 440	Turn off all BREAKS with BRK function
Line 450	**USER WRITTEN SPL ROUTINES CALL NO ECHO** Turns off character echo to program
Line 460	**CALL PERIOD** make a period (.) the line termination character. When used with most ten key entry pads this means at no time does the hand have to leave the pad.
Lines 490 640	CALL INPUT (S$) works in conjunction with a "period" terminator to bypass the standard operation of BASIC.
Lines 1410 1420	CALL ECHO and CR restore normal operation
Lines 650, 690 and others	LENGTH of string functions
Line 670	DEB$ function to remove blanks
Line 700	POSITION function
Lines 1040 1220–1240 and others	STRING operators
Line 1260	PRINT to ASCII FILE (Direct Access)
Line 1130–1400 and others	Examples of IF-THEN-DO-ELSE DO constructions

```
 10 PRINT "JOURNAL ENTRY.";LIN(6)
 20 FILES *
 30 DIM J$[8],F4$[3],S$[22],E$[35],X$[10]
 40 INTEGER S1,F1,D1,S2,F2,D2,S3,S4,F4,F,R,I,P1
 50 INTEGER G3,T3
 60 LONG T1,T2,G1,G2,S
 70 T1=T2=G1=G2=T3=G3=0
 80 REM INITIALIZE FIELD DISCRIPTIONS.
 90 REM FIELD START(S#),FINISH(F#),LENGTH(D#)
100 REM RULE: 3 <= D1 < D2 <= 10
110 REM JOURNAL CODE
120 S1=5,F1=7,D1=F1-S1+1
130 REM ACCOUNT NUMBER
140 S2=8,F2=13,D2=F2-S2+1
150 REM DR(+) OR CR(-) SYMBOL POSITION
160 S3=14
170 REM AMOUNT  (RULE: AMOUNT <= 11 DIGITS)
180 S4=15,F4=26
190 CONVERT F4 TO F4$
200 REM           SUMMARY OF OUTPUT RECORD
210 REM  YR MO JOURNAL-CODE ACCOUNT-NUMBER DR-CR-CODE AMOUNT
220 REM
230 REM VALID INPUT CHARACTERS
240 X$="0123456789"
250 REM INITIALIZE OUTPUT RECORD TO BLANKS AND RECORD COUNTER (R)
260 E$[1,F4]=" ",E$[S3,S3]="+",R=0
270 REM INITIALIZE JOURNAL-CODE FLAG TO FALSE
280 P1=0
290 REM INITIALIZE FILE AND INPUT DATE
300 INPUT "OUTPUT FILE NAME:  ",J$
310 SYSTEM F,"BUILD "+J$+";REC=-"+F4$+",,,ASCII"
320 IF F THEN DO
330   PRINT J$;" EXISTS; PURGE IT (Y OR N)";
340   INPUT S$
350   IF S$[1,1]="Y" THEN DO
360     SYSTEM F,"PURGE "+J$
370     GOTO 310
380   DOEND
390   GOTO 300
400 DOEND
410 ASSIGN J$,1,F
420 IF F>1 THEN 290
430 REM DISABLE ALL BREAKS, STOP ECHO, AND MAKE PERIOD--LINE TERMINATOR
440 I=BRK(0)
450 CALL NOECHO
460 CALL PERIOD
470 PRINT LIN(2),"NOECHO USE PERIOD '.' TO TERMINATE LINES";LIN(1)
```

```
 480 PRINT "YEAR-MONTH #";
 490 CALL INPUT(S$)
 500 IF S$[LEN(S$);1]="." THEN S$=S$[1,LEN(S$)-1]
 510 IF LEN(S$)<>4 THEN DO
 520   PRINT "ENTRY MUST HAVE FOUR DIGITS"
 530   PRINT "EX. 7504 = 1975, FOURTH MONTH"
 540   GOTO 480
 550 DOEND
 560 REM PUT DATE IN OUTPUT RECORD
 570 E$[1,4]=S$[1,4]
 580 PRINT LIN(2),"YEAR = ";E$[1,2]
 590 PRINT "MONTH= ";E$[3,4]
 600 REM
 610 REM START OF JOURNAL ENTRY PROGRAM
 620 REM
 630 PRINT "START ENTRY.";LIN(6);"DR'S";LIN(2);"JOU CODE #";
 640 CALL INPUT(S$)
 650 IF S$[LEN(S$);1]="." THEN S$=S$[1,LEN(S$)-1]
 660 REM REMOVE ANY LEADING OR TRAILING BLANKS AND '.'
 670 S$=DEB$(S$)
 680 REM CHECK FOR BAD CHARACTERS IN ENTRY
 690 FOR I=1 TO LEN(S$)
 700   IF POS(X$,S$[I,I])<>0 THEN 740
 710   IF S$[1,1]="E" THEN 1400
 720   PRINT "BAD CHARACTER:  '";S$[I,I];"'"
 730   GOTO 640
 740 NEXT I
 750 IF (NOT P1) THEN 1020
 760 IF LEN(S$)<=D2 THEN DO
 770   REM S$ MAY BE A 1)COMMAND, 2)JOURNAL-CODE, 3)BAD INPUT.
 780   IF S$="7" THEN DO
 790     REM CHANGE DR-CR FIELD TO DR(+)
 800     E$[S3,S3]="+"
 810     PRINT "DR'S"
 820     GOTO 640
 830   DOEND
 840   IF S$="8" THEN DO
 850     REM CHANGE DR-CR FIELD TO CR(-)
 860     E$[S3,S3]="-"
 870     PRINT "CR'S"
 880     GOTO 640
 890   DOEND
 900   IF S$="9" THEN DO
 910     REM REQUEST A NEW JOURNAL CODE--SET P1 TO FALSE
 920     P1=0
 930     PRINT LIN(2)," TOTAL:"
 940     PRINT USING 1500," DEBITS",T1/100
 950     PRINT
 960     PRINT USING 1500," CREDITS",T2/100
 970     PRINT
 980     PRINT USING 1510," ENTRIES",T3
 990     PRINT LIN(3);"JOU CODE #";
1000     GOTO 640
1010   DOEND
```

```
1020    IF LEN(S$)=D1 AND (NOT P1) THEN DO
1030      REM CHANGE JOURNAL-CODE AND SET P1 TO TRUE
1040      E$[S1,F1]=S$[1,D1]
1050      P1=1
1060      G1=G1+T1,G2=G2+T2,G3=G3+T3,T1=T2=T3=0
1070      PRINT "    ";E$[S1,F1]
1080      GOTO 640
1090    DOEND
1100    PRINT "IMPROPER ENTRY"
1110    GOTO 640
1120  DOEND
1130  ELSE DO
1140    REM S$ IS A COMBINATION ACCOUNT-NUMBER AND AMOUNT
1150    REM PUT ACCOUNT-NUMBER AND AMOUNT IN FIELDS AND PRINT
1160    IF LEN(S$)>D2+(F4-S4+1) THEN DO
1170      PRINT "ENTRY TOO LARGE"
1180      GOTO 640
1190    DOEND
1200    ELSE DO
1210      REM OUTPUT RECORD AFTER STEPPING RECORD COUNT R.
1220      E$[S2,F2]=S$[1,D2]
1230      E$[S4,F4]="0000000000000000000"
1240      E$[F4-LEN(S$[D2])+2]=S$[D2+1]
1250      R=R+1
1260      PRINT #1,R;E$[1,F4]
1270      CONVERT S$[D2+1] TO S
1280      PRINT USING 1500;S$[1,D2],S/100
1290      T3=T3+1
1300      IF E$[S3,S3]="+" THEN DO
1310        T1=INT(T1+S)
1320        PRINT "DR"
1330      DOEND
1340      ELSE DO
1350        T2=INT(T2+S)
1360        PRINT "CR"
1370      DOEND
1380      GOTO 640
1390    DOEND
1400  DOEND
1410  CALL CR
1420  CALL ECHO
1430  PRINT LIN(3);" GRAND TOTAL:"
1440  G1=G1+T1,G2=G2+T2,G3=G3+T3
1450  PRINT USING 1500;" DEBITS",G1/100
1460  PRINT
1470  PRINT USING 1500;" CREDITS",G2/100
1480  PRINT
1490  PRINT USING 1510;" ENTRIES",G3
1500  IMAGE #,8A,X,9D.DD,X
1510  IMAGE 8A,X,9D
```

Sample Run of Journal

INPUT AS IT IS TYPED INTO JOURNAL:

```
SEPT15
7409.
452.
110000456.
23000045678.
8.
2120012367.
11000045678.
444000456.
7.
2440002367.
453.
12220067898.
13330012345678.
8.
45550067898.
46660012345678.
9.
END.
```

ASCII DISC FILE CREATED BY JOURNAL:

```
7409452110000+000000000456
7409452230000+000000045678
7409452212001-000000002367
7409452110000-000000045678
7409452444000-000000000456
7409452244000+000000002367
7409453122200+000000067898
7409453133300+000012345678
7409453455500-000000067898
7409453466600-000012345678
```

Actual appearance of Journal Entry Program on terminal:

```
>RUN
JOU
JOURNAL ENTRY.


OUTPUT FILE NAME:  SEPT15


NOECHO USE PERIOD '.' TO TERMINATE LINES

YEAR-MONTH #

YEAR = 74
MONTH= Ø9
START ENTRY.
```

```
DR'S

JOU CODE #    452
110000            4.56 DR
230000          456.78 DR
CR'S
212001           23.67 CR
110000          456.78 CR
444000            4.56 CR
DR'S
244000           23.67 DR


 TOTAL:
 DEBITS         485.01
 CREDITS        485.01
 ENTRIES          6
```

```
JOU CODE #    453
122200          678.98 DR
133300       123456.78 DR
CR'S
455500          678.98 CR
466600       123456.78 CR


 TOTAL:
 DEBITS      124135.76
 CREDITS     124135.76
 ENTRIES           4


JOU CODE #


 GRAND TOTAL:
 DEBITS      124620.77
 CREDITS     124620.77
 ENTRIES           10

>
```

VII. CALLS – 5 User Written Subroutines in SPL for BASIC/3000.

USER written SPL subroutines (procedures) that are called from BASIC program JOURNAL.

```
$CONTROL SEGMENT=UTILITY
$CONTROL USLINIT,SUBPROGRAM
BEGIN

PROCEDURE PERIOD; BEGIN LOGICAL X; INTRINSIC FCONTROL;
X:=LOGICAL(%56); FCONTROL (2,25,X); END;

PROCEDURE CR; BEGIN LOGICAL X; INTRINSIC FCONTROL;
X:=0; FCONTROL (2,25,X); END;

PROCEDURE ECHO; BEGIN LOGICAL X; INTRINSIC FCONTROL;
FCONTROL (2,12,X); FCONTROL(2,18,X); END;

PROCEDURE NOECHO; BEGIN LOGICAL X; INTRINSIC FCONTROL;
FCONTROL (2,13,X); FCONTROL(2,19,X); END;

PROCEDURE INPUT(S); BYTE ARRAY S;BEGIN INTRINSIC FREAD;
S(-1):=BYTE(FREAD(2,S,-INTEGER(S(-2))));END;

END.
```

Sales and service from 172 offices in 65 countries.
1501 Page Mill Road, Palo Alto, California 94304