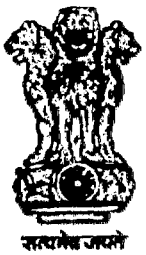
B.A.R.C.-1374



B.A.R.C.-1374

OUTPUT FORMATTING IN APPLE-SOFT BASIC

by

N. S. Navale

Ore Dressing Section,

Materials Group

1987

B.A.R.C. - 1374

GOVERNMENT OF INDIA
ATOMIC ENERGY COMMISSION

B.A.R.C. - 1374

OUTPUT FORMATTING IN APPLE-SOFT BASIC

by

A.S. Navale
Ore Dressing Section, Hyderabad
Materials Group

BHABHA ATOMIC RESEARCH CENTRE
BOMBAY, INDIA

1987

INIS Subject Category : F51.00

Descriptors

BASIC

A CODES

PROGRAMMING

READOUT SYSTEMS

ABSTRACT

Personal computers are being used extensively in various fields. BASIC is a very popular and widely used language in personal computers. Apple computer is one of the popular machines used for scientific and engineering applications. Presenting output from computers in a neat and easy to read form is very important. Languages like FORTRAN have a utlity command 'FORMAT' which takes care of the formatting of the output in user-defined form. In some versions of BASIC a PRINT USING facility is available but it is not as powerful as the FORTRAN statement 'FORMAT'. Applesoft basic does not have even this PRINT USING command. Programmers have to write their own program segments to handle output formatting in Applesoft BASIC. Generally, such user written programs are of limited use as they can not be used easily with other programs. A general purpose and easily transportable subroutine in Applesoft BASIC is presented here for handling output formatting in user-defined structure. The subroutine is nearly as powerful as the FORMAT statement in FORTRAN. It can also be used in other versions of BASIC with very little modifications.

OUTPUT FORMATTING IN APPLE-SOFT BASIC

by

A.S.Navale

The lack of a FORTRAN-like FORMAT statement in BASIC and the non-availability of PRINT USING utility in Applesoft BASIC compel the programmers to write their own output handling subroutines. Many-a-times such subroutines may not work with other programs without modificatations. A few programs have been published (1-3) and some are commercially available (4) for handling numerical output. But these programs can not handle text which also is a part of the output. A subroutine is presented here, which handles formatting of text and numbers, just like the FORMAT statement in FORTRAN. The subroutine works as follows:

The format information is placed in the string variable Z$ in the main program. Whenever output is required, the data is passed on to the subroutine which will handle the formatting and printing.

The subroutine handles the following types of output:

1) Numerical data:

a) Floating point representation (F-format)

b) Scientific notation (E-format)

c) Integer representation (I-format)

2) Alphanumeric data:

a) Text passed on from the main program (A-format)

b) Text contained within the format string (*-format)

c) Any number of spaces (X-format)

d) Any number of line feeds (/-format)

A brief description of each of these formats and their syntax is given below:

1) Numerical output formatting:

The number to be formatted is put in the numerical variable Z and passed on to the subroutine by a GOSUB command.

a) F-format: The syntax of this format is nFxy, where x = number of digits to be printed in the integer portion and y =

number of digits to be printed after the decimal point. n = the number of times the F-format is to be used. The number is printed in x+y+2 columns including the sign of the number and the decimal point.

b) E-format: The syntax for this format is nExy where n, x and y have the same meaning as above. However, if x>1, (x-1) blank spaces will be added to the left of the number being printed and always only one digit is printed in the integer portion.

c) I-format: The syntax is nIx where n and x have the same meaning as defined earlier. The number occupies x+1 columns in the output. If the number being printed contains a fractional part, it will be truncated without rounding i.e. only the integer part will be printed. Thus, both 9.001 and 9.999 will be printed as 9 only. In contrast, the output from F-format with no fractional part requested (i.e. y=0) will be an integer properly rounded. The two formats can be distinguished by the presence of the decimal point as the last character in the output from F-format while the output from I-format does not contain any decimal point.

In all the above formats, the negative numbers are printed with the minus sign as the first character and positive numbers have a space as the first character. If the formatted number can not be accommodated within the specified width, the out put for

that number will be a string of asterisks filling the entire width allotted to the number, a minus sign being the first character if the number is negative.

2) Text output formatting:

A-format: This is used when text generated in the main program is to be formatted and printed. The text to be printed is placed in the string variable Z1$ and passed on to the subroutine. This format has the syntax nAx where x = the maximum number of characters to be printed and n = the number of times the A-format is to be repeated. If x > the length of the string to be printed then the string will be appended with blank spaces so that its length equals x. If x < the length of the string to be printed then only x number of characters starting from the left-most character will be printed. Thus the output will always occupy x columns.

*-format: This is used when the same string is to be printed again and again. The string to be printed is placed between two astersks in the format string Z$. e.g. *Text to be Printed*. All the characters between the two asterisks will be printed as they are. The data enclosed between the two asterisks may contain anything except asterisks. The output does not include the asterisks.

X-format: This is useful for printing a predetermined number of blank spaces. The syntax is nX where n = the number of blank spaces to be printed.

/-format: This is useful for adding a specified number of line-feed characters. The syntax is n/ (or ///.... n times).

The syntax of the format string, Z$, is as follows:

Z$ = "format information" : Z1 = 0

where Z1 is the pointer which points to the character in Z$ being read by the subroutine. Every time a new format information is put in Z$, the variable Z1 should be initialised to zero.

Z$ can contain any combination of the above described formats in the required order. Each type of format is separated by a comma. Blank spaces in Z$ are ignored except when they occur in the *-format. Thus the subroutine accepts only the following characters in Z$: A, E, F, I, *, /, Blank space and the digits 0 to 9. Any other character will result in an error message (except when the occurance is in *-format).

It should be noted that the subroutine always inhibits a carriage return till it encounters a '/' in the format string.

Therefore provision should be made in Z$ to take care of line feed wherever necessary by incorporating a '/' in Z$ at appropriate positions. Also, the subroutine does not activate or de-activate the printer. These functions should be performed by the main program. If the subroutine is used to save data to a disk, adequate error handling facility should be provided to handle error arising from a wrong format specification.

The following example shows how to use the subroutine. The output is presented in Table 1. The Table was generated using EPSON LX-80 dot matrix printer connected to an Apple IIe computer. The subroutine can handle printer control codes also. The program used to generate Table 1 is listed under the heading "EXAMPLE" at the end of this text. Table 2 and Table 3 were also generated by the subroutine.

All numerical and string variable names used by the subroutine start with Z and the format string itself is Z$. Therefore it is better to avoid the variable names starting with Z in the main program. It is useful to save the subroutine as a text file and append it to the required program by an EXEC command.

Acknowledgements

The interest shown and the encouragement given by Dr. N.K. Rao and Mr. N.V. Iyer is gratefully acknowledged.

References:

1. B. G. M. Vandeginste and P. F. van der Wiel, Trends. Anal. Chem., 2 (1983), 220

2. J. Vindevogel, Trends. Anal. Chem., 5 (1986), 84

3. C. Peterson, Call A.P.P.L.E. in Depth, 1 (1981), 85

4. Macro Utilities Master, Heyden Data Systems, Heyden and Son Ltd., England.

SUBROUTINE FORMAT

```
9999  REM * SAVE THE SIGN OF THE NUMBER *
10000 Z0$ = " ": IF Z < 0 THEN Z0$ = "-":Z =  - Z
10010  IF Z2 > 0 THEN 10030
10020 Z2$ = "":Z3$ = "":Z4$ = "":Z5$ = "":Z6$ = "":Z7$ = "":
Z8$ = "":Z9$ = "":Z2 = 0:Z3 = 0:Z4 = 0:Z5 = 0
10030  IF Z6$ = "E" OR Z6$ = "F" THEN 10380
10040  IF Z6$ = "A" OR Z6$ = "I" THEN 10220
10050 Z1 = Z1 + 1: IF Z1 >  LEN (Z$) THEN Z1 = 0: GOTO 10020
10060 Z7$ =  MID$ (Z$,Z1,1): IF Z7$ = " " OR Z7$ = "," THEN
10050
10070  IF Z7$ = "E" OR Z7$ = "F" THEN Z6$ = Z7$: GOTO 10310
10080  IF Z7$ = "A" OR Z7$ = "I" THEN Z6$ = Z7$: GOTO 10160
10090  IF Z7$ = "X" THEN 10150
10100  IF Z7$ = "/" OR Z7$ = "*" THEN  GOSUB 10610: GOTO 100
50
10110  IF  ASC (Z7$) > 47 AND  ASC (Z7$) < 58 THEN 10140
10111  REM ****************************************
10112  REM *                                      *
10113  REM *                                      *
10114  REM *     LINE NO. 10120 AND 10130 MAY BE  *
10115  REM *     MODIFIED AS REQUIRED.            *
10116  REM *                                      *
10117  REM *                                      *
10118  REM ****************************************
10119  REM
10120  PRINT : PRINT  CHR$ (4);"PR#0": PRINT  CHR$ (7): PRIN
T : PRINT : FLASH : PRINT " INVALID FORMAT SPECIFIED ": NORM
```

```
AL
10130  END
10139  REM * MULTIPLIER FACTOR FOR THE FORMAT *
10140 Z2$ = Z2$ + Z7$:Z2 =  VAL (Z2$): GOTO 10050
10149  REM  ==  X FORMAT  ==
10150  FOR Z6 = 1 TO Z2: PRINT " ";: NEXT :Z2 = 1:Z2$ = "":
GOTO 10050
10159  REM  ==  A OR I  FORMAT  ==
10160 Z1 = Z1 + 1: IF Z1 >  LEN (Z$) THEN 10210
10170 Z7$ =  MID$ (Z$,Z1,1): IF Z7$ = " " THEN 10160
10180  IF Z7$ = "," THEN 10210
10190  IF  ASC (Z7$) > 57 AND  ASC (Z7$) < 48 THEN 10120
10200 Z8$ = Z8$ + Z7$: GOTO 10160
10210  IF Z8$ = "" THEN 10120
10220  IF Z6$ = "I" THEN 10250
10230  IF  LEN (Z1$) <  VAL (Z8$) THEN Z1$ = Z1$ + " ": GOTO
 10230
10240 Z9$ =  LEFT$ (Z1$, VAL (Z8$)): GOTO 10270
10250 Z =  INT (Z):Z9$ = Z0$ +  STR$ (Z): IF  LEN (Z9$) >  V
AL (Z8$) + 1 THEN Z9$ = Z0$ + "*": FOR Z6 = 1 TO  VAL (Z8$):
Z9$ = Z9$ + "*": NEXT : GOTO 10270
10260  IF  LEN (Z9$) <  VAL (Z8$) + 1 THEN Z9$ = " " + Z9$:
GOTO 10260
10270  PRINT Z9$;:Z2 = Z2 - 1
10280 Z7$ =  MID$ (Z$,(Z1 + 1),1): IF Z7$ = " " THEN Z1 = Z1
 + 1: GOTO 10280
10290  IF Z2 < 1 AND (Z7$ = "/" OR Z7$ = "*") THEN Z1 = Z1 +
 1: GOSUB 10610: GOTO 10280
10300  RETURN
10309  REM  ==  E OR F  FORMAT  ==
10310 Z1 = Z1 + 1: IF Z1 >  LEN (Z$) THEN 10360
10320 Z7$ =  MID$ (Z$,Z1,1): IF Z7$ = " " THEN 10310
```

```
10330  IF Z7$ = "," THEN 10360
10340  IF  ASC (Z7$) < 48 OR  ASC (Z7$) > 57 THEN 10120
10350 Z8$ = Z8$ + Z7$: GOTO 10310
10360  IF Z8$ = "" OR  LEN (Z8$) > 2 THEN 10120
10370 Z3 =  VAL ( LEFT$ (Z8$,1)):Z4 =  VAL ( RIGHT$ (Z8$,1))
10380  IF Z6$ = "F" THEN 10410
10390  IF Z >  = 10 THEN Z = Z / 10:Z5 = Z5 + 1: GOTO 10390
10400  IF Z < 1 THEN Z = Z * 10:Z5 = Z5 - 1: GOTO 10400
10410  IF Z > 10 ^ Z3 THEN Z9$ = Z0$ + "*": FOR Z6 = 1 TO (Z
3 + Z4):Z9$ = Z9$ + "*": NEXT : GOTO 10570
10420 Z9$ =  STR$ (Z):Z3$ = "":Z4$ = ".": IF Z < 1 THEN 1046
0
10430  FOR Z6 = 1 TO  LEN (Z9$):Z7$ =  MID$ (Z9$,Z6,1): IF Z
7$ = "." THEN Z6 =  LEN (Z9$): GOTO 10450
10440 Z3$ = Z3$ + Z7$
10450  NEXT
10460 Z = Z -  VAL (Z3$):Z =  INT (10 ^ Z4 * Z + 0.5) /  INT
 (10 ^ Z4): IF Z = 0 THEN 10500
10470  IF Z < 0.1 THEN Z = Z * 10:Z4$ = Z4$ + "0": GOTO 1047
0
10480 Z9$ =  STR$ (Z):Z9$ =  MID$ (Z9$,2, LEN (Z9$))
10490 Z4$ = Z4$ + Z9$
10500  IF  LEN (Z4$) < Z4 + 1 THEN Z4$ = Z4$ + "0": GOTO 105
00
10510 Z4$ =  LEFT$ (Z4$,Z4 + 1)
10520  IF Z3$ = "" THEN Z3$ = "0"
10530 Z3$ = Z0$ + Z3$
10540  IF  LEN (Z3$) < Z3 + 1 THEN Z3$ = " " + Z3$: GOTO 105
40
10550 Z9$ = Z3$ + Z4$:Z5$ = "+": IF Z5 < 0 THEN Z5$ = "-"
10560 Z5 =  ABS (Z5):Z5$ = Z5$ +  RIGHT$ (("00" +  STR$ (Z5)
),2)
```

```
10570  PRINT Z9$;:Z2 = Z2 - 1: IF Z6$ = "E" THEN  PRINT Z6$;
Z5$;:Z5$ = "":Z5 = 0
10580 Z7$ =  MID$ (Z$,(Z1 + 1),1): IF Z7$ = " " THEN Z1 = Z1
 + 1: GOTO 10580
10590  IF Z2 < 1 AND (Z7$ = "/" OR Z7$ = "*") THEN Z1 = Z1 +
 1: GOSUB 10610: GOTO 10580
10600  RETURN
10609  REM  ==  / OR *  FORMAT  ==
10610 Z7$ =  MID$ (Z$,Z1,1):Z8$ = ""
10620  IF Z7$ = "*" THEN 10650
10630  PRINT :Z2 = Z2 - 1: IF Z2 > 0 THEN 10630
10640 Z2 = 0:Z2$ = "": RETURN
10650 Z1 = Z1 + 1: IF Z1 >  LEN (Z$) THEN 10680
10660 Z7$ =  MID$ (Z$,Z1,1): IF Z7$ = "*" THEN 10680
10670 Z8$ = Z8$ + Z7$: GOTO 10650
10680  PRINT Z8$;:Z8$ = "": RETURN
10690  END
```

EXAMPLE

```
9  REM * DEMONSTRATION PROGRAM *
20  PR# 1: PRINT  CHR$ (27);"x"; CHR$ (1); CHR$ (27);"a"; CH
R$ (1); CHR$ (27);"-"; CHR$ (1);: REM  -  PRINTER CONTROL CO
DES  -
30  PRINT "TABLE 1"; CHR$ (27);"-"; CHR$ (0): PRINT : PRINT
40  PRINT "Some Properties of Noble Gases"; CHR$ (27);"a"; C
HR$ (0)
50  GOSUB 500
60 Z$ = "1X,A5,A8,4A12,/":Z1 = 0
70  FOR I = 1 TO 12: READ Z1$: GOSUB 10000: NEXT
80  DATA Gas,Atomic,Ionisn,Boiling,Heat of,Volume% in,      ,
No.,potl.,point,vaporsn.,atmosphere
90  GOSUB 500
100 Z$ = "A5,I2,4X,F22,* eV *,2X,F22,* K *,2X,F13,*kcal/mol*
,E32,/":Z1 = 0
110  FOR I = 1 TO 5: READ Z1$: GOSUB 10000
120  FOR J = 1 TO 5: READ Z: GOSUB 10000: NEXT J: PRINT : NE
XT I
130  PRINT : GOSUB 500
140  PR# 0
150  DATA He,2,24.58,4.18,0.022,5.24E-4
160  DATA Ne,10,21.56,27.13,0.44,0.00182
170  DATA Ar,18,15.76,87.29,1.5,0.934
180  DATA Kr,36,14,120.26,2.31,0.00114
190  DATA Xe,54,12.13,166.06,3.27,8.7E-6
200  END
500  PRINT  CHR$ (27);"-"; CHR$ (1);
510  FOR I = 1 TO 60: PRINT " ";: NEXT
520  PRINT  CHR$ (27);"-"; CHR$ (0): PRINT
530  RETURN
540  END
```

TABLE 1

Some Properties of Noble Gases

Gas	Atomic No.	Ionisn potl.	Boiling point	Heat of vaporsn.	Volume% in atmosphere
He	2	24.58 eV	4.18 K	0.022kcal/mol	5.24E-04
Ne	10	21.56 eV	27.13 K	0.440kcal/mol	1.82E-03
Ar	18	15.76 eV	87.29 K	1.500kcal/mol	9.34E-01
Kr	36	14.00 eV	***** K	2.310kcal/mol	1.14E-03
Xe	54	12.13 eV	***** K	3.270kcal/mol	8.70E-06

Table 2

Comparison of Results of Estimation of Tungsten and Molybdenum by three different Methods

No.	3 Wavelength method		Simult.Eqn.method		Individual estimation	
	W (ppm)	Mo (ppm)	W (ppm)	Mo (ppm)	W (ppm)	Mo (ppm)
1	95.51	2.98	95.32	2.77	95.40	3.00
2	23.70	22.40	24.50	23.00	24.00	22.51
3	51.89	52.15	51.57	51.93	51.75	52.52
4	18.59	27.81	18.94	28.08	19.00	28.15
5	18.49	52.44	19.04	52.88	18.50	52.45

Table 3

ABSORBANCE DATA FOR STANDARDS

No.	Composition		Absorbance			Abs.Difference	
	W (ppm)	Mo(ppm)	A1	A2	A3	D1	D2
1	0.00	0.00	0.000	0.000	0.000	0.000	0.000
2	0.00	25.00	0.401	0.562	0.400	0.001	0.325
3	0.00	50.00	0.802	1.123	0.801	0.001	0.648
4	25.00	0.00	0.312	0.194	0.078	0.234	0.000
5	25.00	25.00	0.713	0.756	0.478	0.235	0.325
6	25.00	50.00	1.114	1.318	0.880	0.234	0.649
7	50.00	0.00	0.625	0.388	0.155	0.470	-0.001
8	50.00	25.00	1.025	0.950	0.555	0.470	0.326
9	50.00	50.00	1.427	1.511	0.955	0.472	0.648

Published by Head, Library & Information Services, BARC, Bombay 400 085, India.