STRINGS AND FORMATTING

Some information on and examples of string handling and formatting options for the ATARI 400/800™ Home Computer System.

1) String Handling
2) String Array Emulation
3) Double-subscript String Arrays
4) Inverting Characters
5) Formatting Options

ATARI INC.
CONSUMER PRODUCT SERVICE
PRODUCT SUPPORT GROUP
1312 Crossman Avenue
Sunnyvale, CA 94086

(800) 672-1404 inside CA
(800) 538-8543 outside CA

DEMOPAC #1
Rev.2 9-82 JB

DISCLAIMER OF WARRANTY ON PROGRAMS CONTAINED HEREIN

All computer programs contained herein are distributed on an "as is" basis by Atari, Inc. ("Atari") without warranty of any kind. Any statements concerning the capabilities or utility of the computer programs are not to be construed as express or implied warranties.

Atari shall have no liability or responsibility to the user or any other person or entity with respect to any claim, loss, liability, or damage caused or alleged to be caused directly or indirectly by the computer programs, contained herein. The entire risk as to the quality and performance of such programs is with the user.

Every effort has been made to ensure the accuracy of this document. However, because of ongoing improvements and updating of our computer software and hardware, Atari cannot guarantee the accuracy of printed material after the date of publication and disclaims any liability for changes, errors, or omissions.

Correspondence regarding this pack should be forwarded to Manager of Technical Support, Consumer Product Service, Atari, Incorporated, 1312 Crossman Avenue, Sunnyvale, CA 94086.

STRING HANDLING
ATARI 8K BASIC vs. ATARI Microsoft BASIC
JB 3/82

The major difference between ATARI Microsoft and ATARI 8K BASIC is in the handling of string variables. Here is an overview of the ATARI 8K approach to strings, and a comparison with the ATARI Microsoft method.

It is often necessary to split strings into pieces called substrings. In ATARI Microsoft BASIC, this is accomplished with special functions, MID$,RIGHT$ and LEFT$. In ATARI 8K BASIC, strings are split easily by using a subscript on the string variable. For example, A$(5,10) results in a substring which starts at the fifth character of A$ and ends at the tenth character. If only one number is given in the subscript, the substring will start with that character and end with the last character of the string.

Here is a table of the ATARI 8K equivalents of ATARI Microsoft string functions:

ATARI Microsoft:	ATARI 8K:
MID$(A$,X,Y)	A$(X,Y)
LEFT$(A$,X)	A$(1,X)
RIGHT$(A$,X)	A$(LEN(A$)-X+1)

The function LEN(A$) is the same in both types of BASIC, and returns the length, or number of characters (including blanks) of the string A$. This function is also used in concatenation of strings, that is, putting two strings together into one string. In ATARI Microsoft, concatenation is accomplished with a plus sign. In ATARI 8K, the second string is concatenated to the first by making it a substring which starts just after the last character. Here is an example of two types of concatenation in both BASICs:

ATARI Microsoft:	ATARI 8K:
A$=A$+B$	A$(LEN(A$)+1)=B$
C$=A$+B$	C$=A$ C$(LEN(C$)+1)=B$

In ATARI Microsoft, the subscript on the string indicates a string array, which is handled just like a numeric array. In ATARI 8K BASIC, however, a string array is kept in a very long string, which is put together using concatenation, and taken apart with string splitting, as shown above. Here is an example of a simple string array in both types of BASIC:

ATARI Microsoft:	ATARI 8K:
A$(1)="AAA"	ARRAY$="AAABBBCCC"
A$(2)="BBB"	ARRAY$(1,3)="AAA"
A$(3)="CCC"	ARRAY$(4,6)="BBB"
	ARRAY$(7,9)="CCC"

When using the long-string method, it is often helpful to make all of the substrings in the array the same length, so that it is easy to calculate the position in the array. This can be done by padding the smaller substrings with blanks. Remember, blank spaces count in the length of the string.

```
1 REM STRINGARRAY
2 REM WBB/JB 3/82
3 REM A demonstration of the use of a long-string variable
4 REM to emulate a string array.
5 REM This example keeps a list of customer names.
6 REM It will handle up to 100 names, each up to 30 characters long.
7 REM **********************************************************************
10 DIM ARRAY$(100*30),NAME$(30),BLANK$(30),YN$(1)
11 REM array$ holds 100 names
12 REM name$ is used to accept input name
13 REM blank$ is used to blank-fill, so that lengths are even
14 REM yn$ accepts yes/no answers
20 BLANK$="                                        ":REM initialize 30-space string
30 FOR I=1 TO 100:REM initialize long-string to reserve memory space
40 ARRAY$(I*30-29,I*30)=BLANK$
50 NEXT I:REM fill w/spaces-length of array$ is now 3000
52 REM **********************************************************************
53 REM get customer number and verify that it doesn't already exist
100 PRINT :PRINT "CUSTOMER NUMBER (1-100)(0=END)";
110 INPUT I
120 IF I=0 THEN 500:REM if it's the end, go print the list
130 IF ARRAY$(I*30-29,I*30)=BLANK$ THEN 400:REM if new number, go get name
140 PRINT "CUSTOMER NUMBER ";I;" IS ASSIGNED TO:"
150 PRINT ARRAY$(I*30-29,I*30)
160 PRINT :REM if number is already in use, print out number and name
170 PRINT "DO YOU WISH TO REPLACE WITH NEW NAME (Y/N)";
180 INPUT YN$
190 IF YN$="N" THEN 100
200 IF YN$<>"Y" THEN 170
201 REM if you want to replace it or it's a new number, go ahead
202 REM **********************************************************************
400 PRINT "CUSTOMER NAME ";
410 INPUT NAME$:REM get new name
419 REM fill up name with blanks, in case it is less than 30 characters
420 NAME$(LEN(NAME$)+1)=BLANK$
429 REM concatenate new name into array
430 ARRAY$(I*30-29,I*30)=NAME$
440 GOTO 100:REM go get the next one
498 REM **********************************************************************
499 REM print out the customer list
500 PRINT "DO YOU WANT TO PRINT THE CUSTOMER"
510 PRINT " LIST ON THE SCREEN (Y/N) ";
520 INPUT YN$
530 IF YN$="N" THEN 600
540 IF YN$<>"Y" THEN 500
541 REM if the answer is yes, go ahead and print on screen
550 PRINT "NUMBER","CUSTOMER NAME":PRINT
560 FOR I=1 TO 100
570 IF ARRAY$(I*30-29,I*30)<>BLANK$ THEN PRINT I,ARRAY$(I*30-29,I*30)
580 NEXT I
590 PRINT :PRINT "** END OF LIST **":PRINT
599 REM **********************************************************************
600 PRINT "DO YOU WANT TO PRINT THE CUSTOMER"
610 PRINT "LIST ON THE PRINTER (Y/N)";
620 INPUT YN$
630 IF YN$="N" THEN END
640 IF YN$<>"Y" THEN 600
641 REM if the answer is yes, go ahead and print on printer
650 LPRINT "NUMBER","CUSTOMER NAME":LPRINT
660 FOR I=1 TO 100
670 IF ARRAY$(I*30-29,I*30)<>BLANK$ THEN LPRINT I,ARRAY$(I*30-29,I*30)
680 NEXT I
690 LPRINT :LPRINT "** END OF LIST **":LPRINT
700 END
```

DOUBLE-SUBSCRIPTS
Emulating Two-Dimensional String Arrays
With ATARI 8K BASIC
PY/JB 3/82

ROW \ COL	1	2	3	4
1	x x x 1 2 3	x x x 4 5 6	x x x 7 8 9	x x x 10 11 12
2	x x x 13 14 15	x x x 16 17 18	x x x 19 20 21	x x x 22 23 24
3	x x x 25 26 27	x x x 28 29 30	x x x 31 32 33	x x x 34 35 36

A$= "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

1,2,3... ...35,36

To find the starting location of a particular substring in the array A$, use the following formula:

((COL-1)*CHAR)+((ROW-1)*CHAR*NUMCOL)+1

ROW = row number
COL = column number
CHAR = number of characters per element of array
NUMCOL = total number of columns

In the example given, CHAR = 3, and NUMCOL = 4. In order to find the starting location of A$(ROW,COL), where ROW=2 and COL=4, perform the following calculation:

((4-1)*3)+((2-1)*3*4)+1 = ((3)*3)+((1)*3*4)+1 = (9)+(12)+1 = 22

Thus, the starting character of substring A$(2,4) is character number 22. This substring is addressed as A$(22,24).

```
1 REM INVERT A STRING
2 REM PY/JB 3/82
3 REM turn a string into inverse video
4 REM ************************************************
10 DIM NAME$(50):REM dimension a string to a length of 50 characters
20 PRINT "TYPE IN A NAME"
30 INPUT NAME$
40 FOR I=1 TO LEN(NAME$):REM go through characters one at a time
50 NAME$(I,I)=CHR$(ASC(NAME$(I,I))+128)
55 REM add 128 to each character number to make it inverse video
60 NEXT I
70 PRINT NAME$:REM display inverse name
80 GOTO 20:REM try another one
```

FORMATTING with ATARI 8K BASIC

DEB 3/82

Every computer has some way of placing text where you want it, in order to create exactly the effect you need. Most computers have special formatting commands. With ATARI 8K BASIC, there are several methods to choose from, depending on your needs. Using the TAB key allows quick movement across the screen to a designated column. Using a comma in the PRINT statement will automatically allow a number of spaces between fields. The POSITION statement can be used to put the cursor in any specified row or column, in any mode. In addition to these basic methods, there are special procedures for formatting printed output, and for such functions as right alignment. Here is a brief description of various formatting options with examples.

FORMATTING WITH THE TAB KEY

In order to produce the control characters for TABbing, the following key sequences are used. These characters will appear on your screen but not in a program listing.

TAB: ► Press the ESC key, then the TAB key
Clear TAB: [◄] Press the ESC key, then the CTRL and TAB key simultaneously
Set TAB: [►] Press the ESC key, then the SHIFT and TAB key simultaneously

On powerup, the TAB key advances the cursor to 5 default settings on one physical line. These are column positions 7,15,23,31, and 39. To clear the default TAB setting, type:

PRINT " ► [◄] ► [◄] ► [◄] ► [◄] ► [◄] "

After the ready prompt appears, press the TAB key. The cursor will remain in the first column. To set new tabs type:

PRINT "(insert 10 spaces)[►](insert 10 spaces)[►](insert 10 spaces)[►]"

If the TAB key is now pressed, the cursor stops in columns 2,12,22,32.

Remember to add one space in formatting to get desired column widths.

To set formatted columns from within a program requires planning on the part of the programmer. The following program illustrates 8 columns with 5 spaces between each. These will then be cleared out to set up 3 columns with 16 spaces between each. Press RESET and type:

```
NEW
10 PRINT "►[◄]►[◄]►[◄]►[◄]►[◄] ":REM clear out default tabs
20 PRINT "(insert 6 spaces)[►](insert 6 spaces) [►]
   (insert 6 spaces)[►](insert 6 spaces) [►](insert 6 spaces)[►]
   (insert 6 spaces)[►] ":REM set columns
30 PRINT "A ►B► C ► D ► E ► F ► G ► H ►I ►J► K► L ►M► N ":REM use
   TAB to separate fields
40 PRINT " ►[◄]►[◄]►[◄]►[◄]►[◄]►[◄] ":REM clear columns
50 PRINT "(insert 17 spaces) [►](insert 17 spaces) [►]
   (insert 17 spaces) [►] ":REM set new columns
60 PRINT "A ► B ► C ► D► E ► F"
```

TABbing brings columns to the next print zone as long as the length of the string is smaller than the print zone.

FORMATTING WITH THE COMMA

The comma in a PRINT statement sets up 10 spaces between each field with the default line length of 38 characters. This results in diagonal lines rather than columns. For an example of this, press RESET and type:

```
PRINT 1,2,3,4,5,6,7,8
```

The left margin is controlled by location 82. For a full 40 column line, change the left margin to column 0 with POKE 82,0. Since 10 goes into 40, you will get regular columns. Press RESET and type:

```
POKE 82,0:PRINT 1,2,3,4,5,6,7,8
```

You may also regulate the print zone for the comma with location 201. This example gives 7 spaces between fields, then resets the width to 19. Press RESET and type:

```
NEW
10 POKE 82,0:REM allow 40 characters
20 POKE 201,8:REM set comma spacing to 7
30 PRINT 1,2,3,4,5,6,7,8,9,0,1,2,3:REM print to screen
40 POKE 201,20:REM reset comma spacing to 19
50 PRINT 1,2,3,4,5,6,7,8,9,0,1,2,3:REM print to screen
```

Remember to add one space in formatting to get desired column widths.

You may also shorten the right margin by using POKE 83. The default is 39. This example sets a 30 column screen. Press RESET and type:

```
NEW
10 POKE 82,1:REM start left margin in position 1
20 PRINT "This line has old margins of 2 and 39":REM the next line has
   the new margins
30 POKE 83,30:REM stop right margin in position 30
40 PRINT "123456789012345678901234567890123456789012345678901"
```

FORMATTING WITH THE POSITION STATEMENT

Words as well as numbers can be put on the screen through the use of the POSITION statement. In the following example, the word "ATARI" is positioned on the screen three times by designating the X and Y coordinates in three POSITION statements. Press RESET and type:

```
NEW
10 POSITION 8,2:PRINT "ATARI":REM go to position 8,2 and print word
20 POSITION 18,12:PRINT "ATARI":REM go to position 18,12 and print word
30 POSITION 28,22:PRINT "ATARI":REM go to position 28,22 and print word
40 GOTO 40:REM keep it on the screen
```

POSITIONING IN GRAPHICS WINDOW

The POSITION statement can also be used to write information in the graphics window of text modes 1 and 2. Press RESET and type:

```
NEW
10 GRAPHICS 2
20 POSITION 5,5:REM position cursor in row 5, column 5
30 PRINT #6;"ATARI"
40 GOTO 40:REM keep it on the screen
```

POSITIONING IN TEXT WINDOW

The POSITION statement moves the cursor in the graphics window. If you wish to position the cursor in the text window, you must POKE directly into the text window cursor locations, 656 and 657 (decimal). Press RESET and type:

```
NEW
10 GRAPHICS 2
20 POKE 656,0:REM move cursor to row 0
30 POKE 657,2:REM move cursor to column 2
40 PRINT"LINE 0":REM type "LINE 0" in this position
50 POKE 656,1:REM move cursor to row 1,
60 POKE 657,12:REM move cursor to column 12
70 PRINT"LINE 1":REM type "LINE 1" in this position
80 POKE 656,2:REM move cursor to row 2
90 POKE 657,22:REM move cursor to column 22
100 PRINT"LINE 2":REM type "LINE 2" in this position
110 POKE 656,3:REM move cursor to row 3
120 POKE 657,32:REM move cursor to column 32
130 PRINT"LINE 3";:REM type "LINE 3" in this position
140 GOTO 140:REM keep it on the screen
```

RIGHT ALIGNMENT

If your strings are not the same length, pad the shorter one with spaces. The following example illustrates a string being concatenated to a string of spaces to allow for a three character number. Press RESET and type:

```
NEW
10 DIM A$(2), B$(3):REM dimension strings
20 PRINT "Type in a one or two digit number ...";:INPUT A$
30 ALEN=LEN(A$)
40 B$="(insert 2 spaces)": REM a string of spaces
50 B$=B$(1,3-ALEN):REM allow a three digit number
60 B$(LEN(B$)+1)=A$:REM concatenate the strings
70 PRINT LEN (B$)
80 PRINT B$:REM note indent from left side of screen
```

PRINTER FORMATTING

Tabs on the printer can be set with a string of spaces, as in the following example. Press RESET and type:

```
NEW
10 DIM TAB$(80),A$(35):REM dimension strings
20 X=25
30 TAB$="(insert 80 spaces)":REM set up a string of spaces
40 A$="This line is indented by 25 spaces":REM message to be printed
50 OPEN #1,8,0,"P:":REM open printer for output
60 PRINT #1;TAB$(1,X);A$:REM output to the printer
70 CLOSE #1
```