8712 (101)
QB-99

QB MONITOR

QB-99'ERS U.G. NEWSLETTER

December 1987

The QB MONITOR is the Newsletter of the QB-99'ers User Group, is printed Sept. thru June and sent in exchange for other User Group Newsletters. Send Exchange Newsletter to Frank Cotty, Queensborough Community College, Bayside, NY 11364. Credit original sources.

The QB 99'ers meets the second Saturday of each month September through May, at Queensborough Community College, Bayside New York, room S225. See the calendar at right for the dates

Dues are due

In January all QB 99'ers must pay $10.00 Dues. Failure to pay dues will cause revocation of membership and termination of subscription to the Monitor. Forward your check for dues to Frank Cotty at Queensborough Community College.

Contents

Articles for the FEB issue must be in by JAN 29

Editor's Note

The current issue of the QB Monitor contains items of interest to most TI-99/4A computer users. Programming information in FORTH and in Extended BASIC. Both articles by very articulate and knowledgable programmers. For those without a printer Mike continues with FORTHward Ho! and Ed has contributed FORTUNE OF WHEELS. I'll let Ed describe that one for you.

Ed loves to chat with his fellow computerites. The result is often enlightening to those just sitting by and watching. Tom Freeman of the LA 99'ers wrote some programs in the TopIcs newsletter in addendum to Ed's "BASIC tinygram". His results are given here with Ed's last word (yes he said that). LA STYLER is Ed's last word on print styles.

For those with a RAVE 101 keyboard there also appears in this issue a listing of the Control Key Codes as used for the Rave keyboard.

Thanks to John Wilforth of the West Penn 99'ers we have the following listing of T.I. Vendors. This listing includes cnly current distributers of TI Hard and Software. I urge you to support these and other suppliers which have not been listed. If you write them they will return information to you regarding their products.

Horizon RAM Disk, P.O. Box 554, Walbridge, OH 43465

Quality 99 Software, 1884 Columbia Rd. #1021 Washington D.C. 20009-5161
--DM IV, Draw n' Plot, Screen DumpII

Amnion Helpline, 116 Carl St. San Francisco, CA 94117, Free not for profit help and freeware programs

Disk only software P.O. box 244 Lorton, VA 22079, Horizon, MYARC etc.

McCann Software, P.O. box 34160, Omaha, NE 68134. Printers Apprentice

DataBiotics P.O. Box 1194 Palos Verdes Estates, CA 90274

Tenex Computer Express, P.O. Box 6578, South Bend, IN 46660

Tex-Comp, P.O. Box 33084, Granada Hills, CA 91344

Texaments, 53 Center St. Pathogue, NY 11772

Triton Products Co., P.O. Box 8123, San Francisco, CA 94128

Genial Computerware, 835 Green Valley Drive, Philadelphia, PA 19128

Ryte Data, 210 Mountain St. Halburton, Ont. KOM 150

Micropendium, P.O. Box 1343, Round Rock, TX 78680

Corcomp, 2211-C Winston Rd., Anaheim, CA 92806

THIRD ANNUAL T.I.C.O.F.F. IS MARCH 26

T.I.C.O.F.F. '88

at Roselle Park High School

(Exit 137 - Garden State Parkway)

Saturday March 26, 1988

9 AM to 4 PM Admission $5.00

Featuring:

Huge Indoor Vendor's area, Work Shops - Fairware - Hardware/Software - Swap-Shop. For TI-99/4A, MS/DOS and IBM computers.

Pre-paid admissions get a free disk of IBM-TI software ---Send check and ASAE to TICOFF 185 W. Webster Ave., Roselle Park, N.J. 07204

For further information call (201) 241-45550/8902

FORTHward HO!   THE NEXT STEP

Now that we've gained a basic understanding of FORTH, lets become a little more intimate with the way FORTH handles, stores, and manipulates various types of data.

-by MikeH1   QB 99'ers

One of the final goals in learning how to program in a new language is to be able to write that all-time great program, isn't it.... Well, in order to accomplish this, we as programmers will need to know how the data to be used in our program is to be entered, stored, manipulated, and output.

Again, we will be using BASIC as a reference platform in order to more easily explain some of the new concepts we will be discovering in our trek down the road to becomming a proficient FORTH programmer.

ALL PROGRAMMING LANGUAGES ALLOW THE PROGRAMMER THE ABILITY TO RESERVE THE NECESSARY SPACE IN MEMORY FOR ALL DATA THAT WILL BE USED IN THE PROGRAM. THE PROCESS OF RESERVING MEMORY FOR DIFFERENT TYPES OF DATA IS REFERRED TO AS DECLARING VARIABLES.

In TI BASIC we never really had to declare a variable before using it (with the exception of arrays that contained more than 10 elements which had to be declared with DIM statements). FORTH is different. IN FORTH WE MUST EXPLICITLY DECLARE ALL VARIABLES BEFORE THEY CAN BE USED.

As in TI BASIC, FORTH allows us NUMERIC TYPE variable as well as STRING TYPE VARIABLES. There is, however, a slightly different way in which FORTH looks at these variables when compared to TI BASIC. This is important, so lets pay attention...

NUMERIC VARIABLES UP TO 255 ARE CONSIDERED TO BE ONE BYTE LONG, AND WILL ONLY REQUIRE 1 MEMORY CELL IN WHICH THEY CAN BE STORED. These are called single length variables, and can be declared as follows...

```
VARIABLE AGE
```

This statement declares the variable AGE, thereby reserving 1 byte (or 1 memory cell) in memory for it.

Now if we wanted to store a person's age in AGE, we would use the FORTH word ! (pronounced store) to put the persons age at the memory location referenced by the variable AGE.

```
12 AGE !
```

...is how it is done.

Supposed we wanted to find we didn't know what the persons age was, but it was already stored in the variable AGE, how would we get it? We would use the FORTH word @ (pronounced fetch) to get a 1 byte value from the variable AGE, and put it on the stack. To print it we use FORTHs dot command.

```
AGE @ .
```

...is how you do it.

Ok, lets look at NUMERIC VARIABLES THAT COMTAIN VALUES LARGER THAN 255. Since 255 is the largest value that can be stored in a single memory location, any number greater than 255 will need 2 memory locations in which to hold its data.

FORTH allows for this by providing the words necessary to declare a DOUBLE LENGTH VARIABLE, store data to a DOUBLE LENGTH VARIABLE, and to fetch data from a DOUBLE LENGTH VARIABLE (there is also provisions in FORTH for doing a lot more with double as well as single length variables...more on that later).

...uses the FORTH word 2VARIABLE

```
2VARIABLE YEAR
```

to declare the double length variable

```
1987. YEAR 2!
```

...stores the value 1987 to YEAR with the FORTH word 2! (pronounced 2store).

```
YEAR 2@ D.
```

...fetches 1987 from year with 2@ (pronounced 2fetch) and puts it on the stack where it is printed with the forth word D. (pronounced double-print ??).

Don't forget the period after 1987 (or any double length number).

Ok, that starts us on our way with numeric variables. Next we will look at STRING VARIABLES in FORTH.

In TI BASIC we didn't have to pre declare a string variable, and storing a value to a string variable was as simple as A$="hello there". In FORTH strings are handled a bit differently... (again). The simplest way, and the way that will serve to demonstrate how string variables are declared is as follows...

```
VARIABLE ANYNAME 30 ALLOT
```

....uses the familiar FORTH word VARIABLE to declare the variable ANYNAME, then 30 ALLOT uses the FORTH word ALLOT preceeded by the number 30 to reserve 30 more bytes in memory for the variable ANYNAME. We now have a 31 byte block of memroy reserved for a string of characters that are referenced by the variable name ANYNAME.

Now how do we store data to, and fetch data from, ANYNAME. Well, logically we would use a variation on the FORTH word ! (store) to store a string value... right??

Well... you're not quite right, but you're not entirely wrong either..

```
ANYNAME 30 EXPECT
```

...introduces the new FORTH word EXPECT. This causes the computer to immediately wait for the next string of characters entered terminated by a carriage return (the ENTER key) to be stored at the location ANYNAME in the order that they were entered. The number 30 preceding EXPECT is the maximum number of characters to EXPECT from the keyboard.

To see the string stored at ANYNAME, we would...

```
ANYNAME 20 TYPE
```

...TYPE is a FORTH WORD that types out the designated number of characters at ANYNAME.

There are other ways to declare and acces strings in FORTH, which we will go into later.

Ok, we've covered the basics of variables in FORTH as it applies to the TI-99/4A. This should provide a little food for thought. Take some time to do a little experimenting with what you've just learned...

Next installment will look at strings in depth, and cover loops, and conditional branching. Then we will be able to begin work on our first real program in FORTH.

till next time...

FORTUNE OF WHEELS
A Tinygram
by Ed Machonis

Last month I promised that I would have something this month for the people without printers. In lieu of Epsons and Axioms I give you a Fortune of Wheels. Just to prove that I CAN write a program that doesn't use a printer.

I must admit I had a little help. The Tinygram presented here is an enhanced version of son Michael's WORDGUESS which is on the TIMARC disk. Originally a TI Basic 10 Liner, it has been recast in Extended Basic and the hidden phrase display resembles that used in a popular TV show of similar name. Sorry, no Vanna White to turn over the letters. What do you expect from a TINYgram?

Unlike the TV show, where the amount of the prize depends on the random spin of a wheel, the prize in this game is proportionate to the relative difficulty of the puzzle and how quickly you solve it. The longer the phrase, the greater the prize; the fewer tries, the greater the prize.

Fortune of Wheels is a two player or two team game. The first player or team leaves the room or turn their backs to the screen while the second player or team enters the mystery phrase. As soon as ENTER is pressed the screen will clear. Alternatively, if you are sure of your typing, the TV brightness or contrast can be turned down, or the program revised to black out the screen during entry.

If you wish to black out the screen during entry of the mystery phrase, change Line 2 to read as follows:

```
2 CALL SCREEN(2):: INPUT M$
:: CALL CLEAR :: CALL SCREEN
(8):: L=LEN(M$)
```

The first player or team can now try to guess the individual letters or the entire phrase. You must enter the entire phrase to be recognized as a winner. If you do not enter the entire phrase, do not enter more than one character. Entering a wrong letter, or more than 1 letter, or an incorrect phrase will cost you a try and reduce your prize.

Cumulative totals can be kept on paper. (Horrors! Let it be a challenge to you. Either sharpen your pencils or your programming skills!) Negative amounts won (Possible!), should be subtracted from the cumulative totals.

This Tinygram is easy to type in, quick to load (cassette users take note), and FUN to play. It can be as simple or challenging as you and your opponents care to make it.

Minimum requirements are Console, Cassette Player, TV and Extended Basic.

```
1 ! *** FORTUNE OF WHEELS **
    *      A TINYGRAM      *
    * by Mike & Ed Machonis*
    ************************

2 CALL CLEAR :: INPUT "ENTER
 THE MYSTERY PHRASE    ":M$
:: CALL CLEAR :: L=LEN(M$)

3 D$=RPT$(CHR$(30),L):: FOR
J=1 TO L :: IF SEG$(M$,J,1)<
>" " THEN 4 ELSE D$=SEG$(D$,
1,J-1)&" "&SEG$(D$,J+1,L)

4 NEXT J :: PRINT D$

5 T=T+1 :: PRINT :"TRY No.";
T;: :: INPUT "TYPE LETTER O
R ENTIRE PHRASE":A$ :: IF LE
N(A$)>1 AND LEN(A$)<L THEN 5

6 W=L+1-T :: IF A$=M$ THEN 9

7 FOR J=1 TO L :: IF SEG$(M$
,J,1)=A$ THEN D$=SEG$(D$,1,J
-1)&A$&SEG$(D$,J+1,L)ELSE 8

8 NEXT J :: PRINT :D$ :: GOT
O 5

9 FOR J=1 TO W :: CALL SOUND
(200+J*10,330+40*J,0):: NEXT
 J :: PRINT :"YOU WIN ";STR$
(W);",000 WHEELS!":: :: INP
UT "PRESS ENTER TO PLAY AGAI
N":G$ :: T=0 :: GOTO 2
```

SETTING YOUR PRINTER

by Tom Freeman, LA 99ers
from an idea by Ed Machonis, QB-99ers

My article this month is going to go "back to basics" - literally! It began with a "BASIC Tinygram," as he called it, sent to us by Ed Machonis of Floral Park, NY, to show what could be done with just 1Ø lines of Basic code. It follows this paragraph in exactly the form that Ed sent it to us, with two exceptions: for some reason I typed an extra space before the ? in line 5, and I have provided the XBasic Checksums for all the programs in this article. Although this is a program that can be run in Basic as well as XB, I advise you to do your typing in XBasic and use the Checksum program, so as to ensure accuracy.

```
1 DIM P$(15)!155
2 READ P$(1),P$(2),P$(3),P$(
5),P$(6),P$(7),P$(8),P$(9),P
$(1Ø),P$(11),P$(12),P$(13),P
$(14),P$(15)!254
3 OPEN #1:"PIO" !253
4 PRINT :"1 PICA/RESET","9 T
EST","2 ELITE","1Ø EXIT","3
EXPANDED","11 SUPERSCRIPT","
4 COMPRESSED","12 SUBSCRIPT"
 !ØØ4
5 INPUT "5 EMPHASIZED  13 1/
2 LINE SP6 ITALIC       14 L
MARGIN 137 D'BLE STRIK 15 R
MARGIN 678 UNDERLINE      ?":
I !221
6 PRINT #1:CHR$(27)&P$(I)!16
Ø
7 IF I<>4 THEN 9 !2Ø3
8 PRINT #1:CHR$(27)&CHR$(15)
!233
9 IF I<>1Ø THEN 4 !244
1Ø DATA @,M,W1,E,4,G,-1," QU
ICK BROWN FOX JUMPS OVER THE
 LAZY RED DOG 123456789Ø TIM
ES",,SØ,S1,1,1,QC !Ø12
```

Ed's notes for this program included a warning that the next to last data item is the lower case letter l, not the number 1, and that the space following the quotation mark in line 1Ø is important (because each string sent to the printer is preceded by Esc - ASCII 27 - and Esc Q has an affect on the printer, whereas Esc space does not). Naturally you will need to check the specific codes for your printer - these are for an Epson RX-8Ø but most modern printers are compatible with it. The program is used by combining succesive entries.

After I typed in the program and ran it, I found that there were a few minor problems: 1) because the printer was opened as PIO, each time a code was sent to it, a line feed ensued, which you may not want, 2) for some strange reason my printer (Citizen MSP-1Ø) would not turn off underline with the Esc @ code, 3) there were no options to pick the left and right margins or the line feed- the C following the Q in the data for P$(15) defined a right margin of 67 and the carriage return that automatically followed the l defined a left margin of 13, and 4) typing line 2 was a pain! Therefore I revised the program slightly to fix these problems. The first was solved by opening the printer file as PIO.CR so there would be no line feeds (but note that I then had to add a carriage return and line feed to the test line), the second by putting in a specific option to turn off underline. For the third I put in a second input request to pick the actualy number desired, and for the fourth I read the data statements in a loop. What follows is my first revision of the program.

```
1ØØ DIM P$(16)!156
11Ø FOR X=1 TO 16 !126
12Ø READ P$(X)!2Ø8
13Ø NEXT X !238
14Ø OPEN #1:"PIO.CR" !195
15Ø PRINT :"1 PICA/RESET  9
 NO UNDRLINE2 ELITE","1Ø TES
T","3 EXPANDED","11 EXIT","4
 COMPRESSED  12 SUPERSCRIPT"
 !156
16Ø PRINT "5 EMPHASIZED  13
SUBSCRIPT  6 ITALIC      14
X/72 IN.LF 7 D'BLE STRIK 15
L MARGIN X" !Ø26
17Ø INPUT "8 UNDERLINE   16
R MARGIN X ":I !Ø32
18Ø IF I>16 THEN 15Ø !2Ø5
19Ø IF I<14 THEN 21Ø !ØØ6
2ØØ INPUT "X?":M !244
21Ø IF I<>1Ø THEN 24Ø !224
22Ø PRINT #1:P$(1Ø)&CHR$(13)
&CHR$(1Ø)!163
23Ø GOTO 15Ø !229
24Ø PRINT #1:CHR$(27)&P$(I)!
16Ø
25Ø IF I<14 THEN 27Ø !Ø67
26Ø PRINT #1:CHR$(M)!216
27Ø IF I<>11 THEN 15Ø !135
28Ø DATA @,M,W1," ",E,4,G,-1
,-Ø," QUICK BROWN FOX JUMPS
OVER THE LAZY RED DOG 123456
789Ø TIMES",,SØ,S1,1,1,Q !17
2
29Ø CLOSE #1 !151
```

Some things to note about this version. It is still a Basic program, although again I have provided checksums so you can type it with accuracy in XBasic. Also, the fourth data item did not have to be separately defined. Where you see a space on this page you should type CTRL O. Although you will still see a blank on the screen what is actually there is ASCII 143, which is an acceptable printer code for compressed mode. By the way, I believe I made a mistake in this version - the third to last data item which is presently a 1 should be an A. Type it the "wrong" way first, to get the correct checksum, then make the substitution.

My next version (which follows the 2nd below) merely put the above program into true XBasic format, with multiple statement lines. It actually takes up one bite MORE of code, despite being 11 program lines shorter, but it should be easier to type in. Note that the mistake mentioned above is corrected here, and that the 4th data item is still CTRL O.

```
1ØØ DIM P$(16):: FOR X=1 TO
16 :: READ P$(X):: NEXT X ::
 OPEN #1:"PIO.CR" !163
11Ø DISPLAY AT(3,1)ERASE ALL
:"1 PICA/RESET  9  NO UNDRLI
NE2 ELITE","1Ø TEST","3 EXPA
NDED","11 EXIT","4 COMPRESSE
D  12 SUPERSCRIPT" !131
12Ø DISPLAY AT(7,1):"5 EMPHA
SIZED  13 SUBSCRIPT  6 ITALI
C       14 X/72 IN.LF 7 D'BLE
 STRIK 15 L MARGIN X 8 UNDER
LINE    16 R MARGIN X" !168
13Ø ACCEPT AT(11,1)VALIDATE(
DIGIT)BEEP:I !Ø26
14Ø IF I>16 THEN 11Ø ELSE IF
 I>=14 THEN DISPLAY AT(12,1)
:"X?" :: ACCEPT AT(12,3)VALI
DATE(DIGIT)BEEP:M !225
15Ø IF I=1Ø THEN PRINT #1:P$
(1Ø)&CHR$(13)&CHR$(1Ø):: GOT
O 11Ø !Ø72
16Ø PRINT #1:CHR$(27)&P$(I):
: IF I>=14 THEN PRINT #1:CHR
$(M)!Ø36
17Ø IF I<>11 THEN 11Ø ELSE C
LOSE #1 !119
18Ø DATA @,M,W1," ",E,4,G,-1
,-Ø," QUICK BROWN FOX JUMPS
OVER THE LAZY RED DOG 123456
789Ø TIMES",,SØ,S1,A,1,Q !18
8
```

For the last version I decided to take a completely different approach. I noted that many current printers ive a "master" print control code, usually Esc ! n. Seven of the eight bits in the number n each control a print mode. For the Citizen MSP-1Ø, starting with the rightmost bit, they are elite/pica, no effect, compressed, emphasized, double strike, expanded, italics, and underline. The advantage of this method is that each mode can be toggled on and off separately by toggling the appropriate bit on and off. All bits "off" (ASCII Ø) is the equivalent of resetting to defaults, except that I continued to have the problem that even when I did this the underline was not turned off - must be some quirk in my printer! I decided that I would also like to be able to toggle near letter quality on and off, and that I wished to display on the screen what the current "settings" are.

To understand how I did this, you need to know how XBasic handles "logical operators." This will also be applicable to assembly language programming. There are four such expressions: AND, OR, XOR, and NOT. When used on numbers, they operate on full 16 bit numbers (which because the highest bit must be reserved for the sign of the number range from -32768 to 32767). NOT operates on a single number and reverses each bit in it. The other hree work on two numbers and produce a third. In the case of AND, corresponding bits are compared in the original two numbers, and a 1 put in that "place" if both bits were 1, otherwise a Ø. For OR, the result is a 1 if either number contained a 1 - only if both were Ø is the result a Ø. And finally XOR will place a 1 in the proper position in the result only if one of the numbers had a 1 there. If both were 1 or both were Ø then the result is a Ø. For you assembly language programmers exactly the same procedures apply, but see your manual for addressing modes.

Now we can combine these operators with the ASCII codes that must follow Esc ! to the printer. Since we want to treat each bit independently, the logical operators make it easy to reverse them or test them. Note that the first seven data items are numbers each of which have only one bit on, namely bit 1 and 3 to 8 (2 is not used). By using AND on this value and the current value of Q all the bits of Q except the one of current interest are turned off, and this particular bit is also off if it was off in Q (remember that AND insists that the bit be on in both numbers). The resultant number will still be a power of 2 however. By using the SGN function it becomes either a 1 or a Ø and this is listed on the screen to indicate the current state of the particular print mode. This is all done in line 13Ø.

The rest of the lines through 17Ø complete the setup of the menu. Note that I have also read some of the menu lines into an array with data statements - this was done so that I could use the SIZE command in line 15Ø and not erase to the end of the lines on the screen. Line 18Ø accepts the input number, and also sets M=Ø (used in menu items 1Ø to 13) because CHR$(M) will always be sent to the printer, but we want it to have meaning only for

1Ø-13 - CHR$(Ø) has no effect on the printer, unless it is needed by a previous code. Line 19Ø now sends the program to the appropriate line number. Line 2ØØ is for NLQ mode. The logical operator XOR is used here. Since it requires that only one of the two numbers operated on have a 1 in the bit position under consideration, we can reverse the state of the bit by doing an XOR with 1. Similarly line 23Ø does an appropriate bit reversal for each of the first 7 menu items by using XOR on Q and the current data item, which has only 1 bit turned on.

The rest of the program follows closely those that appear above. However please note the quoted string in line 29Ø. What looks like two spaces following the numbers is NOT - you should type CTRL J and CTRL M !! Also, type line 3ØØ carefully, or the screen setup will not be correct. The program is presented in 28 columns here, so "what you see is what you get" and the checksum should also help.

I might add that with careful attention to these operators you can use one variable to represent 16, if they are to be only 1 or Ø. Each variable that you are interested in can be one bit in the program variable, and you can use the logical operators to manipulate them.

This program was written more out of my interest in programming techniques and in teaching them to our readers. Hopefully it may also be of some use to you. Just remember not to turn off your printer after sending the codes to it!

```
100 DIM P$(16)!156
11Ø FOR X=1 TO 16 :: READ P$
(X):: NEXT X :: FOR X=1 TO 4
 :: READ T$(X):: NEXT X :: N
LQ$(1)="ON" :: NLQ$(Ø)="OFF"
 :: OPEN #1:"PIO.CR" !141
12Ø DISPLAY AT(3,1)ERASE ALL
:"MODE","1=ON,Ø=OFF","1 ELIT
E/PICA":"2 COMPRESSED":"3 EM
PHASIZED":"4 DOUBLE STRIKE":
"5 EXPANDED":"6 ITALICS":"7
UNDERLINE" !1Ø9
13Ø DISPLAY AT(13,19):"12" :
: FOR X=14 TO 16 :: DISPLAY
AT(X,19):"Ø" :: NEXT X !ØØ7
14Ø FOR X=1 TO 7 :: DISPLAY
AT(X+3,14):SGN(Q AND VAL(P$(
X))):: NEXT X !18Ø
15Ø DISPLAY AT(11,1):"8 SUPE
RSCRIPT":"9 SUBSCRIPT" :: FO
R X=1 TO 4 :: DISPLAY AT(X+1
2,1)SIZE(18):T$(X):: NEXT X
!233
16Ø DISPLAY AT(17,1)SIZE(23)
:"14 NEAR LETTER QUALITY" !2
19
17Ø DISPLAY AT(18,1):"15 TES
T":"16 RESET":"17 EXIT" !251
18Ø ACCEPT AT(21,1)VALIDATE(
DIGIT," ")SIZE(-2)BEEP:I ::
M=Ø !Ø81
19Ø IF I>17 THEN 18Ø ELSE ON
 I GOTO 23Ø,23Ø,23Ø,23Ø,23Ø,
23Ø,23Ø,25Ø,25Ø,24Ø,24Ø,24Ø,
24Ø,2ØØ,26Ø,22Ø,28Ø !Ø32
2ØØ P=P XOR 1 :: IF P THEN P
$(14)="x1" ELSE P$(14)="xØ"
!Ø26
21Ø GOTO 25Ø !Ø73
22Ø Q,P=Ø :: GOTO 25Ø !214
23Ø Q=Q XOR VAL(P$(I)):: GOT
O 27Ø !1Ø9
24Ø ACCEPT AT(I+3,19)VALIDAT
E(DIGIT," ")SIZE(-2)BEEP:M !
226
25Ø PRINT #1:CHR$(27):: DISP
LAY AT(17,24):NLQ$(P)!213
26Ø PRINT #1:P$(I)&CHR$(M)::
 IF I=16 THEN 13Ø ELSE 14Ø !
2Ø1
27Ø PRINT #1:CHR$(27)&"!"&CH
R$(Q):: GOTO 14Ø !ØØ8
28Ø CLOSE #1 !151
29Ø DATA 1,4,8,16,32,64,128,
SØ,S1,A,1,Q,N,x1,"QUICK BROW
N FOX JUMPS OVER THE LAZY RE
D DOG 123456789Ø  ",@ !Ø95
3ØØ DATA 1Ø X/72 IN. LF  X=,
11 L MARGIN     X=,12 R MARG
IN     X=,13 SKIP X LINES X=
 !Ø61
```

CONTROL KEY CODES AND THE RAVE 101 KEYBOARD....by Ed Machonis - QB-99'ers

If you are using a RAVE 101 Keyboard, pressing CTRL J or CTRL M will not produce CHR$(10) or CHR$(13). Actually, the Control key codes produced by either RAVE or TI-99 keyboards range from 129 thru 159. (Pg III-2 User's Ref. Guide) Most printers apparently subtract 128 and interpret the codes as 1 thru 31. The RAVE 101 Keyboard requires different key presses from those shown in the Reference Guide for CTRL codes 10, 13, and 27 thru 31. The following table shows the key presses required for CTRL codes 1 - 31.

1	CTRL A	2	CTRL B	3	CTRL C	4	CTRL D	5	CTRL E
6	CTRL F	7	CTRL G	8	CTRL H	9	CTRL I	10	CTRL [
11	CTRL K	12	CTRL L	13	CTRL \	14	CTRL N	15	CTRL O
16	CTRL P	17	CTRL Q	18	CTRL R	19	CTRL S	20	CTRL T
21	CTRL U	22	CTRL V	23	CTRL W	24	CTRL X	25	CTRL Y
26	CTRL [illegible]	27	FCTN SHFT.	28	FCTN SHFT;	29	SHFT F11	30	SHFT F8
31	SHFT F[illegible]								

LA STYLER

by Ed Machonis

Based on a Program by:
Tom Freeman LA 99ers

ANOTHER Print Styler??? I can hear the "Who needs it?"s. If you own a printer, you do! I promise this is the last styler we will print. <GRIN>

But there is more here than meets the eye. First, read the accompanying article by Tom Freeman, reprinted from the August issue of LA Topics, paying particular attention to the last paragraph. As you may have gathered, ulterior motives are afoot.

These short programs are not only useful utilities, they are excellent learning tools. (For the programmer as well as the reader.) I, for one, learned a great deal from Tom's article.

Did you notice these techniques: Displaying menu text using a FOR-NEXT loop in Line 150 and placing the text in a DATA statement in Line 300? Using loops in Line 110 to read in the data? Using ON GOTO in Line 190 to branch to the program sector associated with the menu selection? And, the essence of the program, use of the "master" print control code, ESC !n.

This was my first contact with the master print control code as my, anything but "current", Epson RX-80 does not support it. Another first was the use of the XOR "logical operator". I have never seen the Exclusive OR function used in a program before, in fact, I didn't even know it was available in Extended Basic. Although I did have a passing acquaintance with Exclusive OR gates when I dabbled in digital electronics, using them in programs just never occurred to me. Yet it's all there in the manual, just another case of "In One Eye and Out the Other!"

I typed in Tom's last version, knowing it could not possibly work with my RX-80. When I saw the menu and status display, I knew it was a program I wanted to have. When a print style is selected a "1" appears alongside the selected style on the menu. If that selection is repeated, the print style is canceled and a "0" replaces the 1. Thus individual print modes can be turned off without using the master reset code, which cancels all selections.

I was able to adapt Tom's program so that it would work with my RX-80. I have called the result LA STYLER in deference to its origins. It should work with most Epson Compatibles.

Tom toggled the print modes on and off by reversing the state of the respective master control code bit using XOR. Since I did not have a master control code, a different approach was required.

Both cancel and enable codes for each mode were placed in the DATA statements. P$ was made into a two dimensional array, with 16 rows (for menu Items 1 thru 16) and two columns, the first holding the cancel code and the second the enable code. Where cancel and enable were not applicable, as in menu items 10 thru 16, the same DATA was placed in each column of the respective element.

A second array, N, was added and is used to display the status of modes 1 thru 9 on the menu, and to point to the appropriate column of the P$ array when the print code is sent to the printer. N is equal to either 0 or 1 and is toggled by the XOR function in Line 190.

As an example, let us say you elect to turn on Compressed, Selection 2 on the menu. P$(2,0) holds the cancel Compressed code, and P$(2,1) has the enable Compressed code. Since this mode had not previously been selected, N(2) has a value of 0. In Line 190 the XOR operator will change the value of N(2) to 1 and the enable Compressed code is sent to the printer in Line 270. If Compressed is selected again, XOR will change the value of N(2) from 1 to 0 and cancel Compressed is sent to the printer.

RX-80 mode priorities are 1. Elite, 2. Emphasized, 3. Compressed and 4. Pica (Default Mode). Once a mode is enabled, a lower priority mode cannot be enabled; sending its print code to the printer will have no effect. Line 135 was added so that the status display would accurately reflect the priorities enforced by the printer.

On the RX-80, Subscript and Superscript modes toggle each other on and off, depending on the last one selected. Line 136 was added to reflect this toggle.

Another change deemed desireable was adding the provision for user input of text as provided in STYLEALINE, which appeared in a recent issue of the Monitor. Line 275 made this possible.

LA STYLER.........................Cont'd

Use of this feature caused the menu to scroll off the screen. An M array was added to store the Line Space, Margin, and Skip Over Perf values so that they could be displayed when the menu was restored. (Oh the tangled web we weave - When we alter what others conceive!) One plus, we can place values into this array (Line 100) which reflect initial printer status.

Another change was to reset the printer upon initialization (End of Line 110) so that the initial menu display truly reflects printer status

Although the RX-80 does not have a Near Letter Quality mode, provisions for it were left in for those with printers supporting this mode. Print codes are compatible with the Epson LX-80. The values in the N array (Shades of FDR) are used to display status of this mode, which enabled elimination of the NLQ$ array used in Tom's program.

The 3rd and 4th DATA items in Line 290 are not blank spaces but CHR$(18) (Type CONTROL plus R) and CHR$(15) (CONTROL O), respectively. Similarly, the apparent two blank spaces at the end of EACH quoted string in line 295 are actually (CHR$(10) (CONTROL J) and CHR$(13) (CONTROL M). The blank space at the beginning of EACH quoted string is a true blank space and required in this program (as it saves sending an additional Escape code to the printer).

If you have a modern printer that supports a master print control code, you undoubtedly will want to use Tom Freeman's program which is considerably shorter and, most likely, faster. If, on the other hand, you have an obsolete 3 year old printer like mine, you may find LA STYLER useful. In any event, there is something to be learned from both programs. And remember - Obsolescence is a state of mind!

```
90 ! *** LA STYLER ***
   by Ed Machonis QB-99ers
   Based on a Program by
   Tom Freeman LA 99ers

100 DIM P$(16,2),N(18):: M(1
)=12 :: M(2),M(4)=0 :: M(3)=
80

110 FOR X=1 TO 16 :: FOR J=0
 TO 1 :: READ P$(X,J):: NEXT
 J :: NEXT X :: FOR X=1 TO 4
 :: READ T$(X):: NEXT X :: N
LQ$(1)="ON" :: NLQ$(0)="OFF"
 :: OPEN #1:"PIO.CR" :: PRIN
T #1:CHR$(27)&"@"

120 DISPLAY AT(3,1)ERASE ALL
:"MODE","1=ON,0=OFF","1  ELI
TE":"2  COMPRESSED":"3  EMPH
ASIZED":"4  DOUBLE STRIKE":"
5  EXPANDED":"6  ITALICS":"7
  UNDERLINE"

125 DISPLAY AT(11,1):"8  SUP
ERSCRIPT":"9  SUBSCRIPT"

135 IF N(1)=1 THEN N(2),N(3)
=0 ELSE IF N(3)=1 THEN N(2)=
0

136 IF I=8 THEN N(9)=0 ELSE
IF I=9 THEN N(8) [illegible]

140 FOR X=1 TO 9 :: DISPLAY
AT(X+3,18):N(X):: NEXT X

150 FOR X=1 TO 4 :: DISPLAY
AT(X+12,1):T$(X);STR$(M(X)):
: NEXT X

160 DISPLAY AT(17,1)SIZE(23)
:"14 NEAR LETTER QUALITY"

170 DISPLAY AT(18,1):"15 TES
T":"16 RESET/PICA":"17 INPUT
 TEXT":"18 EXIT"

180 ACCEPT AT(22,1)VALIDATE(
DIGIT," ")SIZE(-2)BEEP:I

190 IF I>18 THEN 180 ELSE N(
I)=N(I)XOR 1 :: ON I GOTO 27
0,270,270,270,270,270,270,27
0,270,240,240,240,240,210,27
0,220,275,280

210 DISPLAY AT(17,24):NLQ$(N
(14)):: GOTO 270

220 FOR X=1 TO 14 :: N(X)=0
:: NEXT X :: GOTO 270

240 ACCEPT AT(I+3,19)VALIDAT
E(DIGIT," ")SIZE(-2)BEEP:M(I
-9)

260 PRINT #1:CHR$(27)&P$(I,N
(I))&CHR$(M(I-9)):: GOTO 135

270 PRINT #1:CHR$(27)&P$(I,N
(I)):: IF I=16 THEN M(1)=12
:: M(2),M(4)=0 :: M(3)=80 ::
 GOTO 135 :: ELSE 135

275 PRINT "INPUT A LINE OF T
EXT":"(ZZZ RETURNS TO MENU)"
 :: LINPUT A$ :: IF A$="ZZZ"
 OR A$="zzz" THEN 120 ELSE P
RINT #1:A$&CHR$(10)&CHR$(13)
:: GOTO 275

280 CLOSE #1

290 DATA P,M, , ,F,E,H,G,W0,
W1,5,4,-0,-1,T,S0,T,S1,A,A,1
,1,Q,Q,N,N,x0,x1

295 DATA " QUICK BROWN FOX J
UMPS OVER THE LAZY RED DOG 1
234567890  "," QUICK BROWN F
OX JUMPS OVER THE LAZY RED D
OG 1234567890  ",@,@

300 DATA 10 X/72 IN. LF  X=,
11 L MARGIN     X=,12 R MARG
IN     X=,13 SKIP X LINES X=
```