"SNAP" & "SHOW" ON THE BALLY 4K

by ED GROEBE

Two of the new commands on the BALLY expansion units allow you to copy a picture from the screen, store it in memory and then display it again. These two commands are called SNAP (to save & store) and SHOW (to display).

Well you can do the same kind of thing with the 4K BALLY. Of course it's not as easy, but the process can be shown with program below. It will copy and display a 2Ø x 2Ø pixtel picture in the screen's center.

```
 5 CLEAR;BOX Ø,Ø,2Ø,2Ø,1;BOX Ø,Ø,1Ø,1Ø,3
 1Ø NT=Ø;Z=Ø;FOR CY=-1ØTO 9;FOR CX=-1ØTO
    9;@(Z)=PX(CX,CY);Z=Z+1;NEXT CX;
    NEXT CY
 1ØØ CLEAR ;IF KP<1GOTO 1ØØ
 2ØØ FOR Z=ØTO 399;CY=Z÷2Ø+1;CX=RM;
    BOX CX,CY,1,1,@(Z);NEXT Z;CX=-79
```

PICTURE

Instruction 5 will put a picture of a box in the designated area, as an example. Any other picture can be substituted for this box, so long as it's in the 2ØX2Ø pixtel space in the center of the screen.

SNAP

The function of instruction 1Ø is to scan the area to be copied, pixtel by pixtel -- from left to right across 2Ø pixtels and progressively up 2Ø lines through the designated area.

The condition of each of these 4ØØ pixtels is stored in memory as a string, starting at @(Ø) and ending at @(399).

A "1" is stored if the pixtel is "on" and a "Ø" is stored if it's "off". These 1's & Ø's are used later, when the picture is reproduced, for setting the last condition of the "BOX" command.

When the scan is completed and the status of each pixtel is stored, instruction 1ØØ will clear the screen and wait for you to press any key. <u>When you are ready to see the picture displayed, press any key on the keypad</u>.

SHOW

By following instruction 2ØØ the computer will step through the 4ØØ memory locations, calculate the corresponding pixtel location from the value of Z and draw a box, one pixtel high and wide.

Each box will either be "white", if the value of @(Z) is"1"; or "black", if the value is "Ø".

In this program the reproduced picture is shown in the screen area to the right and above the center. With a different program, the area to be copied and the place it's displayed could be under control of a joystick.

MEMORY

Each pixtel condition requires a memory location which in turn requires two bytes for storage. This demonstratic requires 8ØØ bytes to store the 4ØØ pixtel conditions. Because of this it's not practical to copy a picture much bigger than this and still leave enough memory space to have a useful program.

EDGE SOFTWARE

SNAP AND SHOW UPDATES to the program on page 48 were received from Ron Picardi and Bert Holmes. These techniques drastically reduce the amount of memory required to store a picture for future re-display. The program by Ron Picardi uses a string storage method that is limited to a 15 pixel wide swath which keeps the scheme from being adversely influenced by the 16th nibble of a byte.

"Try the following listing for a SNAP routine for a pixel area of 15 by 15. It uses 30 bytes as opposed to the 450 bytes needed by the scheme of page 48.
"It works by making each string store the condition of 15 pixels. Each pixel is represented by 2 raised to a power from 0 to 14. Example, a line with pixel 1, 3, and 15 would contain a string number made up of the sum of 1, 4, and 16384. <note that we are using 2 raised to n-1 power>. The number 32767 would mean that all 15 pixels were 'on'.

this is the SNAP subroutine:

```
10    FOR Y= A TO B; W=1; Z= Z+1; FOR X= C TO D;
IF PX(X,Y) `(Z)= `(Z)+W
20  IF W< 16384 W = Wx2
30  NEXT X; NEXT Y; RETURN
```

now for the DRAW subroutine:

```
50  FOR Y= A TO B; W=16384; Z=Z+1; FOR X= D TO C
STEP -1; IF @(Z)-W> -1 BOX X,Y,1,1,1
60  W= W-2; NEXT X; NEXT Y; RETURN
```

the value of D-C must be 15, while the value of B-C can be 15, or any other positive value."

The next method by Bert Holmes describes a method of storing a 20 x 20 pixel space. In Bert's words:
"With a little effort, the SNAPand DRAW programs can be made almost 15 times more efficient! - and one variation would allow you to move it around the screen without losing previous information.
"The SNAP subroutine uses 8 variables (I used the last 7 'T-Z' but the user could change these to fit his needs.)

S= Subroutine Line Number <It will be 31000 in this example>

T= This value is the top of the memory used to store the picture. I decided to use the Top of Program Memory so it wouldn't move as your program is modified. It is decremented by 2 as each double byte is filled.

U= The Counter to keep track of how many pixels have been accumulated for the active byte. (0 is empty, while 15 is full)

V= The accumulated value for the active byte.

W= Width of picture area (whole pixels)

Z= Height of above picture area.

X= Reference position of picture area

Y= Reference position of picture area (the center of the picture area is positioned, as in the BOX command).

These last two items must be set by the CALLing program

```
30000    S=31000;T=  <top of memory  (32767 ) is
used in this example>
30005  W=      ;Z=    <you set these>
30010  V=(ZxW÷15+(RM>0)+2)x2;GOSUB S
     <this stores the number of two-byte values
used>
30020  V=Zx256+W; GOSUB S
      <stores the height and width as a single
value>
30030  U=0; V=0
    <initializes variables for picture>
30040  FOR CY=Y-Z÷2 TO Y+Z÷2 RM-1
    <range of picture vertical extent>
30050  FOR CX=X-W÷2 TO X+W÷2+RM-1
    <-ange of picture horizontal extent>
30060  U= U+1; V= 2xV+PX(CX,CY)
    <accumulates value and increments count>
30070  IF U=15 GOSUB S; U=0; V=0
    <stores value when full and resets value and
counter>
30080  NEXT CX; NEXT CY
    <end loop statements>
30090  IF U>0 GOSUB S
    <stores leftovers, if any>
30100  RETURN
      <T=next  memory space available for next
picture>
31000  %(T)=V; T= T-2; RETURN
    <the subsubroutine to store value and update
the Top Memory Pointer>
```

"Using this routine, the 20x20 pixel picture described on page 48 is stored in 58 bytes instead of 800. The ratio would be even better as the picture size increased.

"Now to DRAW the captured picture, we have a similar set of variables, but only U, X, and Y need be specified as the others are in %(T). One can store more than one picture if U is set equal to the picture number (start with 1).

```
60000  S=61000: T= <top of picture storage area,
same as above>
60010    U=U-1;  IF  U    T=T-%(T): GOTO <A LINE
NUMBER THAT CONTROLS THE NEXT PICTURE>
60020  T= T-2
    <increment  pointer>
60030  GOSUB S; Z=V÷256; W= RM; GOSUB S; U=0
    <get Z, W, V, and set U>
60040  FOR CY=Y-Z÷2 TO Y+Z÷2+RM-1
    <develop vertical range>
60050  FOR CX=X-W÷2 TO X+W÷2+RM-1
    <develop horizontal range>
60060  V= V÷2; U= U+1
    <extract value increment counter>
60070  BOX CX,CY,1,1,RM
    <print pixel if there>
60080  IF U=15  U=0; GOSUB S
    <old data used up, get new>
60090  NEXT CX; NEXT CY; RETURN
    <end loops>
61000  V=%(T); T= T-2; RETURN
      <subsubroutine to retrieve data and update
pointer>
```

"This will print your picture in the area specified by the X,Y coordinates and the stored values of W and Z. In doing so any previous information will be obliterated. If it is desired to save the background information, change line 60070 to

```
60070 IF RM BOX CX,CY,1,1,3
```

(——►)

which will reverse or "Exclusive OR" (XOR) the picture with the existing background. Calling the subroutine a second time with the same values will therefore erase the picture, leaving the background."

Ron Picardi adds: If the subroutines were written in assembly code, you would have a true SNAP command, as these routines in Basic are quite slow.