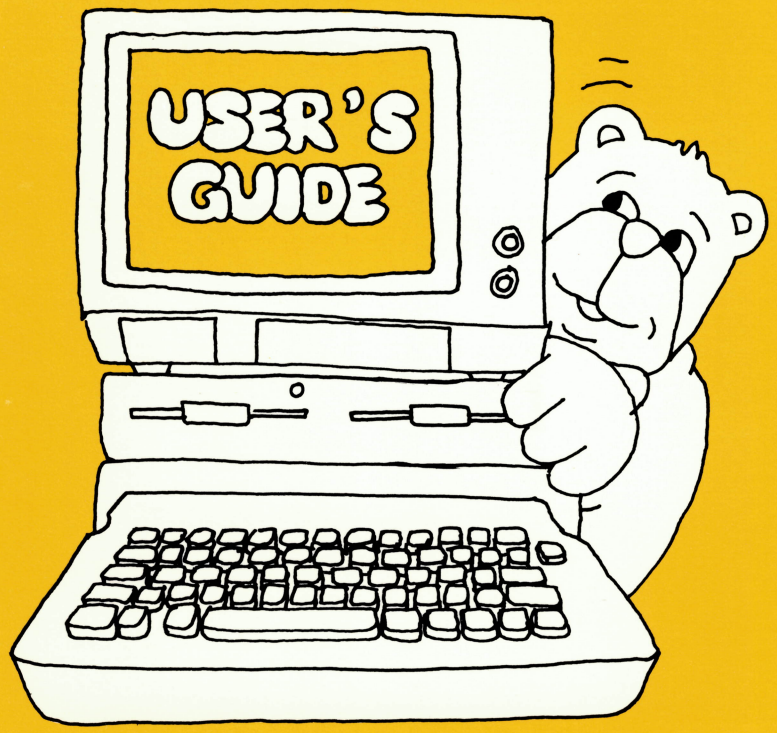
Weekly Reader
Family Software
Middletown, CT 06457

4433

STICKYBEAR® BASIC



STICKYBEAR BASIC

By Richard Hefter and Steve Worthington

Published by Optimum Resource, Inc., Norfolk, Connecticut.
Distributed by Weekly Reader Family Software, 245 Long Hill Road, Middletown, CT 06457.

Contents


```
YARRAY(CTR)=Y
CTR=CTR+1
IF Y>MAXY THEN MAXY=Y
IF Y<MINY THEN MINY=Y
NEXT X
```

When this has executed, you will have an array of Y values (unscaled), the minimum Y, and the maximum Y. The range is the maximum minus the minimum. The scale factor is the range divided by the number of vertical pixels in the plot area (160). To plot these, we need only a small loop

```
HGR
HCOLOR=7
SCALE=(MAXY-MINY)/160
FOR SX=0 TO 279
SY=YARRAY(SX)/SCALE
PLOT SX,SY
NEXT SX
```

These are the bare necessities for putting together the PLOTTER program. See if you can take this information and use it to write a working version of PLOTTER. Remember...Experiment and have fun!

program must calculate all Y values before the Y range can be known. This is usually done inside the loop above. The Y variable will have to be stored in an array. Dimension the array to 280 at the beginning of the program.

```
DIM YARRAY(280)
```

Before entering the loop, initialize an array pointer to 1. Inside the loop

```
GOSUB 1000
YARRAY(CTR)=Y
CTR=CTR+1
```

This will set the element of YARRAY to the corresponding value of Y at that X; but to calculate the range, you need the minimum and maximum. This is where things get tricky. To calculate the min and max, just

```
IF Y>MAXY THEN MAXY=Y
IF Y<MINY THEN MINY=Y
```

The tricky part is making sure that MINY and MAXY are initialized before entering the loop. Do this by assigning both MINY and MAXY to the value of Y at the first point. Before entering the loop, you need

```
X=START
GOSUB 1000
MINY=Y
MAXY=Y
```

Now, put together the whole calculation loop. It will look something like

```
X=START
GOSUB 1000
MINY=Y
MAXY=Y
CTR=0
FOR X=START TO END STEP INCREMENT
GOSUB 1000
```


What Is *Stickybear BASIC ?*

The *Stickybear BASIC* disk contains an interactive tutorial and example programs which are designed to help you learn the BASIC programming language. BASIC is an acronym for Beginner's All-purpose Symbolic Instruction Code.

The tutorial on the disk is divided into eight sections describing types of BASIC statements. There are example programs in each section for you to experiment with. In addition, there are many useful short programs on the disk that you can modify to suit your own needs.

MAKE A COPY OF THE MASTER DISK! *Stickybear BASIC* is not copy-protected and we suggest that you make a copy of the disk and work with the copy. Once you have the original safely tucked away, you can experiment without worrying about losing the original programs. *Stickybear BASIC* can be copied using COPYA (under DOS 3.3) or the ProDOS System Utilities.

The best way to learn a computer language is by using it. Keep experimenting. Change the example programs, and write simple programs of your own. The more you work with BASIC, the easier it will be to learn the language.

HAVE FUN! Don't take the computer too seriously. The more fun you have programming, the quicker you will learn. Don't be disappointed if your program doesn't work on the first try, most don't.

To Begin

Stickybear BASIC will run on any Apple, Apple II, Apple II Plus, Apple //c or Apple //e with at least 48K and a disk drive.

To start, put the program disk into drive 1 and turn on your Apple. If you have Autostart, you will see the title panel displayed. If your Apple does not have the Autostart ROM, you will see the monitor cursor *. Type 6, then type P while holding down the key marked Control (6 CTRL P), then press the RETURN key.

After the title, credits, and copyright, the program will ask you to enter your name. Type in your name and then press the RETURN key. Your name can be up to ten letters long. Now when the program wants to tell you to do something, it can refer to you by name.

The Main Menu

After entering your name, the program will proceed to the Main Menu:

STICKYBEAR BASIC

1. GETTING STARTED
2. LINE NUMBERS
3. PRINT STATEMENTS
4. VARIABLES
5. KEYBOARD INPUT
6. BRANCHING AND LOOPS
7. SOUND
8. GRAPHICS
9. EXIT TO BASIC

Use the up and down arrows or the I and M keys to move the cursor to the section you want. You can also press the corresponding number to move the cursor. Press RETURN when the cursor is pointing to the section you want. Each time you complete a section, you will see a screen that says:

PRESS Y TO GO ON TO THE NEXT SECTION, ANY OTHER KEY TO RETURN TO MENU.

You can also press the ESC (Escape) key at any time to return to the Main Menu.

Stickybear BASIC is divided into eight tutorial sections. The last option in the Main Menu allows you to get out of the program to type in your own BASIC programs.

If you select option 9, you will be asked if you wish to exit the tutorial. If you answer yes, you will see the BASIC cursor]. You

After checking if the guess is high or low, increment the counter of guesses and go back to get a new one if the counter is less than 50. Otherwise, the program will print the "You will never guess..." message, tell the user the number, and start again.

PLOTTER -- A program to plot two-dimensional equations. This project will require lots of time to program. Write the program one step at a time and make sure each part works before going on to the next.

Program Specification:

The program will plot a graph of an equation expressed in the form Y=fn(X). For example, Y=5X+3. The equation will be entered as a subroutine like

```
1000 Y=23 * X^3 + 5 * X^2 + 12 * X + 6
1010 RETURN
```

The program will plot the equation in Hires graphics with four lines of text mode at the bottom of the screen. We will leave off the complications of axes and labels at this time. The user will input the range of the X variable and the starting point of the X variable.

Some of the things you must consider when writing a program like this are:

The range of the X variable will vary with each equation, or with different plots of the same equation. You must scale your graph to fit the range you are interested in. If X can vary from 0 to 25000, then you want your steps (280 steps across the whole screen) to be 25000/280 units each. You can do this automatically by setting up your FOR-NEXT loop like

```
INCREMENT=(END-START)/280
FOR X=START TO END STEP INCREMENT
```

The range of the Y variable will also vary with each equation or with the different plots of the same equation. You must also scale the vertical on your graph. This requires a little more thought because the

Some of the things you must consider when writing a program like this are

To get the random number, use the statement

```
10 NUMBER = INT(RND(1)*100+1)
```

The RND function will return a number between 0 and 1. Multiplying this by 100 will produce a number between 0 and 100, excluding 100. Adding 1 gives a number 1 through 100. We then take the integer so that we do not have a number such as 26.54272.

Next we need to set the number of guesses to 0. This needs to be done with each new number.

```
20 NUMTRIES=0
```

The program then asks the user for a guess. Remember to give the user clear instructions. You can use a statement like

```
30 PRINT "Guess a number from 1 to 100"
40 INPUT "Enter your guess and press return";GUESS
```

This will get the user's guess. If the guess is correct, print the congratulations and go back and start over. The easiest way to do this is to check if the guess is incorrect and if it is, to jump over the lines of code that handle a correct guess. This will look like

```
50 IF GUESS <> NUMBER THEN 100
60 PRINT "Congrat...
70 GOTO 10
```

After that the program is really three IF-THEN statements in the form

```
100 NUMTRIES=NUMTRIES+1
110 IF GUESS > NUMBER THEN PRINT "Try a lower
number"
120 IF GUESS < NUMBER ...
130 IF NUMTRIES < 50 THEN 40
```


can type in your own BASIC programs at this point or RUN any of the BASIC programs that come with the disk. To return to Stickybear BASIC, reboot the disk as described above or simply type BRUN BASIC and press RETURN.

Control Q (CTRL Q) Quiet
Press the Control key while pressing the Q key (CTRL Q) to turn off the sound for quiet tutoring. Apple //c users can simply turn the volume up or down by using the adjustment knob located on the left of the machine.

It's On The Disk

The tutorial sections on the disk contain all the instructions you need to get started, along with some interesting exercises. You should boot the disk now and start working with the program. Remember to experiment, and have fun!

The rest of this manual contains additional information and helpful hints. You can use it as needed.

Section 1 • Getting Started

This section describes some of the DOS commands that you will need to write programs. Take the time to become familiar with them. Also take the time to learn the editing features of your computer. On the Apple II you can retype data by running the cursor over the line using the right arrow. The Apple II Reference Manual fully describes how to move the cursor on the screen.

There are more advanced commands that you may find helpful after you have been programming for a while. Some of them are

TRACE -- This will list each line number as it is executed. This can be very helpful if you need to know what path your program is taking at a branch.

NOTRACE -- Turns off the trace command.

CONT -- Continue from where a program was interrupted by a Control C or STOP instruction.

Other terms that you will need to know are

PIXEL -- One dot or picture element on the screen in Hires graphics. A Hires screen is made up of 280 x 192 pixels.

SUBROUTINES -- A sequence of instructions for performing a specified task that can be used repeatedly.

USER -- This is the person running the program and typing in the data. This is the person the program is written for.

ARRAY -- A list - A numeric array is a list of numbers. Arrays are defined at the beginning of a BASIC program using the DIM statement. The statement DIM A(10) specifies an array named A which can contains ten elements. You can access any element of the array by number. The statement X=A(5) sets X to the the value of the fifth element of the array A.

BUG -- An error in a program.

DEBUG -- To remove bugs from a program. Programmers spend most of their time debugging programs.

Section 2 • Line Numbers

After using the tutorial on disk, you will have a good idea how to work with line numbers. Remember to space out your line numbers, put subroutines at high numbers (such as 10000), and type a line number and press RETURN to replace the old line.

It is usually a good idea to leave space at the beginning of the program. This space can be used to add lines that initialize variables and define arrays that you might need later in the program.

ART DRIFTING CIRCLE • another demonstration using high-resolution graphics
REVERSE PRINT • A print backward program using string commands
PERPETUAL CALENDAR • A useful program to print calendars for any year
JOYSTICK • A demonstration of reading the joystick position
JOYSTICK2 • A drawing program based on the "JOYSTICK" program
RANDOM SENTENCE • A silly word game using random numbers
SOUND EXAMPLE • An example program that uses the sound utility
SOUND EXAMPLE2 • Examples of some different sounds
LORES DEMO • A demonstration of the low-resolution colors
HIRES DEMO • A demonstration of the high-resolution colors

BASIC programs for you to write:

After you have worked with the tutorial on the Stickybear Basic disk and have experimented with the sample programs included, you may want to try creating some programs of your own from scratch. We have presented two projects with some hints and a starting point...the rest is up to you.

NUMBER GUESS -- This is a simple program to get you used to writing in BASIC. Take your time writing it and keep in mind what you have learned so far.

Program Specification:

The program will pick a random number from 1 through 100. The program will then ask the user to guess what the number is. If the guess is too low, the program will respond "Try a higher number." If the number is too high, the program will respond "Try a lower number." If the answer is correct the program will respond "Congratulations! You guessed the number in X tries." X is the number of guesses the user made. The program will then accept the next try. If the user guesses more than 50 times without getting it right, the program will print the message "You will never guess it. The number is Y." Y is the correct number.

Some hints on making different types of sounds can be found in the program "SOUND EXAMPLE2."

Section 8 • Graphics

The Apple computer offers a variety of graphics modes. The mode that is used most often is the Hires (High Resolution) mode. Many of the demo programs contained on the Stickybear BASIC disk use Hires graphics. Spend some time looking at and using these programs.

The fastest way to start using Hires graphics is to buy graphics utilities. There are many companies that offer packages that allow you to draw pictures that can be loaded by a BASIC program; design special Applesoft shapes which can be drawn anywhere on the screen; or add animation to your own BASIC programs.

Helpful Hints

Save a copy of your program each time you get a part of it working. This will give you a backup if you totally mess up the program trying to get the next part working.

Build yourself a library of standard subroutines. These subroutines can then be combined to write new programs more easily.

Subscribe to hobbyist computer magazines. Many of these magazines provide programs to type in. These magazines also give many helpful programming hints.

BASIC programs on the Stickybear BASIC disk:

The following is a list of some of the BASIC programs included with *Stickybear BASIC*. Use the CATALOG command to get a full list of all programs contained on the disk. Any filename that is displayed with an A in front of it is a BASIC program. RUN each program.

ART TWIST1 • A demonstration using high-resolution graphics
ART CIRCLE • A demonstration using low-resolution graphics

Section 3 • Print Statements

The PRINT statement is the BASIC program's way of passing information to the outside world. In addition to using the PRINT statement to output your results as described in the tutorial, you can use the PRINT statement to debug your program. Usually, a program will not work as expected on the first try. When this happens, insert PRINT statements to see what is happening at each step along the way. For example

```
10 X=4
20 I=1
30 Y=X*I
40 IF Y<> 100 THEN 30
50 I=I+1
60 PRINT "The number is ";I
```

This program contains a very common type of error. When it is run the program will sit and calculate forever, but will never get to the final PRINT statement. To find out what is going wrong, we insert the following line:

```
35 PRINT "X=";X;"  Y=";Y;" I=";I
```

When we RUN the program now, the screen will be filled with the following line repeated endlessly:

```
X=5  Y=5  I=1
```

We can now see that the reason the program isn't working is that the variable I is remaining constant (and hence X and Y). The solution is now obvious, just move the statement that increments I inside the loop. We delete line 50 (type 50 and press RETURN to delete the line). We replace line 35 with

```
35 I=I+1
```

Now when we run the program, we are presented with the correct result (20).

This is just one example of how PRINT statements are used to debug programs. In general, adding PRINT statements is an alternative to using the TRACE command to see what your program is doing.

Section 4 • Variables

Variables will become second nature to you as you write more BASIC programs, but it is important not to fall into some of the traps that BASIC variables present. The most common problem people have is forgetting that only the first two characters of a variable name count. Don't have one variable GOLD and another GOOD. Sooner or later, you will change GOOD and then use GOLD in a calculation and get the wrong answer. Another thing to be careful of is loop counters. Remember that if I is your loop counter in a FOR-NEXT loop, you should not assign a value to it within the loop.

Try to use the clearest variable names possible, but remember that variable names take up memory space and processing time. As your programs grow large, a name like "MILEAGEFROMSTART" can become quite a burden. In a program that has more than one mileage variable, it is best to put the word mileage at the end of each variable to avoid assigning the same variable two names. So instead of "MILEAGEFROMSTART," try "TOTMILES," which conveys the same idea.

Section 5 • Keyboard Input

As you have learned from the tutorial, there are two ways to get input from the keyboard, the INPUT and the GET statements. Some hints for using these statements effectively are

• The GET statement has only limited use in good programming. One use of the GET statement is when the user is making a menu choice. Be sure to give clear instructions to the user like "Press A,B, or C." The other common use is in games where the keyboard is used to control the player's "playing piece."

• Keep input simple. Don't ask a user to input name and address on the same line. It is much simpler for the user to enter the name (first, last); street address (number, street name); city; state and Zip Code on separate lines. This will save the user from getting terribly confused if an error is made.

• With complicated or very important input, give the user a chance to verify that the input is correct. This requires that you print out what you input and ask the user to verify it before going on.

Section 6 • Branching and Loops

Branches and loops make BASIC a very powerful problem-solving language, but they are also the cause of many programming bugs. Some of the things to look out for when writing programs with branches and loops are

• Be sure that you have clearly defined the conditions for a branch. There is a world of difference between I>6 and I>=6.

• When writing FOR NEXT loops watch out for "fenceposts." Fenceposts are the first and last times through the loops. These are the places where most loop errors occur.

• When a loop is based on a variable, look carefully at every statement that changes that variable.

• When writing programs with complex loops, add PRINT statements to help you test the program and get it working. Delete the PRINT statements when the loops are working correctly.

Section 7 • Sound

The *Stickybear BASIC* disk contains an assembly language utility which allows you to program sounds from BASIC. The BASIC program, which uses this utility, puts the assembly language program in memory using POKE statements. After you have read through the sound tutorial, spend some time experimenting with the program "SOUND EXAMPLE."