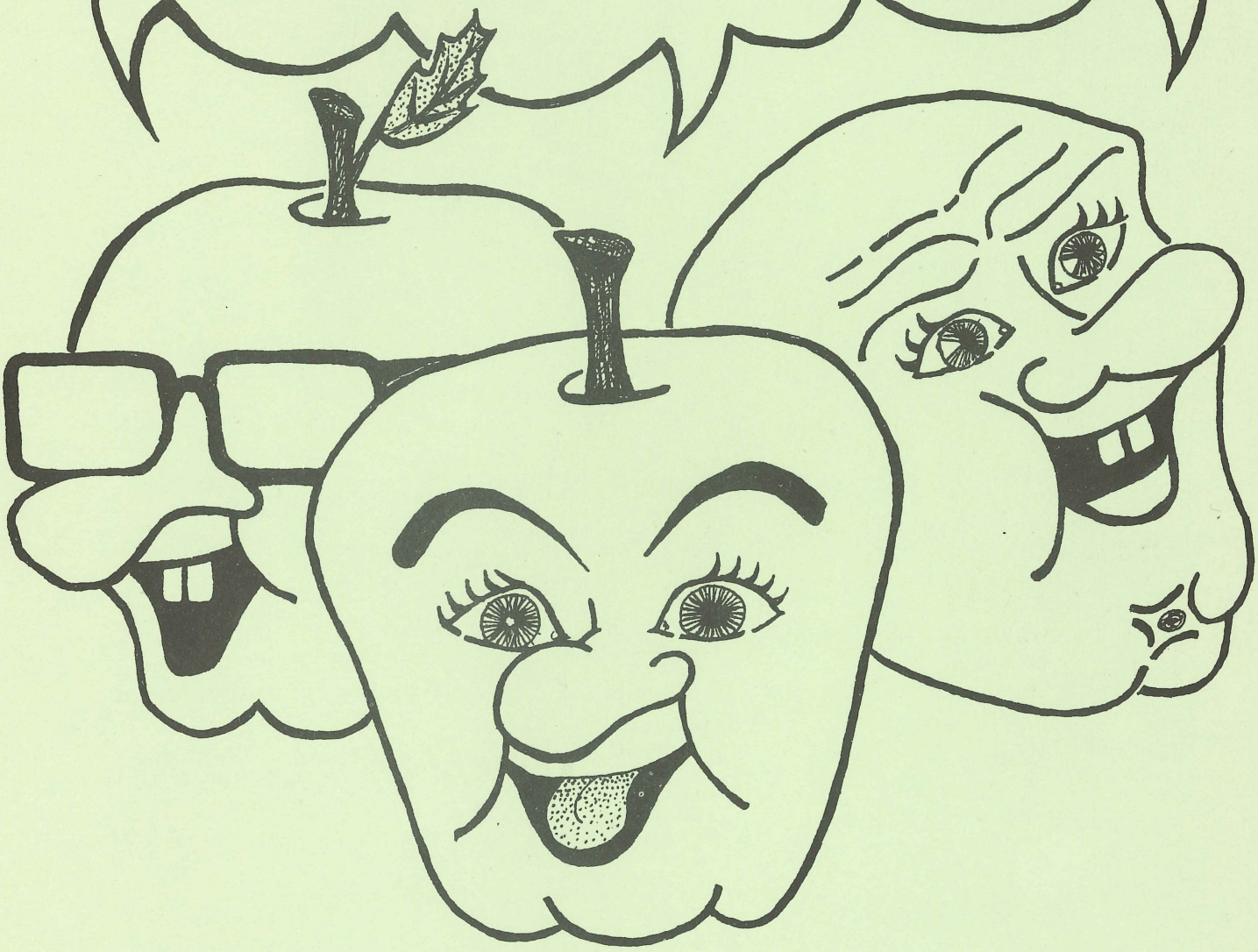
BABBLE



DON WORTH

The
Software Factory

A a B b C c D d E e F f G g H h I i J j K k L l M m N n O o P p Q q R r S s
T t U u V v W w X x Y y Z z 0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f G g
H h I i J j K k L l M m N n O o P p Q q R r S s T t U u V v W w X x Y y Z z

DON WORTH

BABBLE

ILLUSTRATIONS BY STEPHEN WORTH

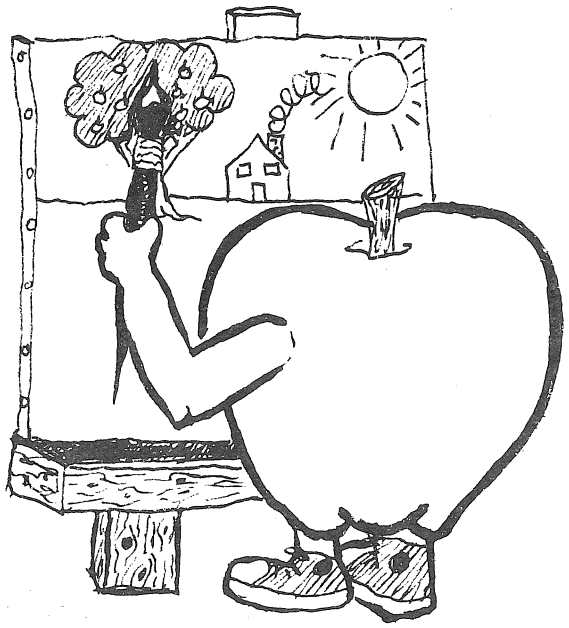
The
Software Factory
P.O. BOX 904
CHATSWORTH, CA 91311

A a B b C c D d E e F f G g H h I i J j K k L l M m N n O o P p Q q R r S s
T t U u V v W w X x Y y Z z 0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f G g
H h I i J j K k L l M m N n O o P p Q q R r S s T t U u V v W w X x Y y Z z

DEDICATED TO

JIM BALTER

WHOSE "GRAMMAR" PROGRAM,
LONG AGO, PROVIDED THE
INSPIRATION FOR BABBLE.



BABBLE

WHAT IS BABBLE?

In some ways BABBLE is a programming language like BASIC or PASCAL but, unlike these general purpose languages, BABBLE is highly specialized. "Programming" in BABBLE is much simpler than in BASIC, as you define the rules the computer will follow to create random or fixed patterns using language, music, or graphics. A BABBLE program can be as simple as a random sentence "babbler" or as complex as a meaningful prose generator. The only limitation is your imagination. Some of the things BABBLE can do are:

* Complete a partially written story, filling in the blanks with words, either chosen at random from lists you specify, or obtained from the keyboard at execution time. The stories generated this way are often hilarious!
* Construct random poetry and jingles from building block words and phrases, laid out by grammatical rules.
* Generate random stories from small sections or sentence patterns.
* Create fixed or random graphic displays, block letters, and animation more concisely and easily than in BASIC.
* Mix text, graphics, sound (and anything else you can think of through calls to your own assembly language subroutines).

THE PROGRAM

BABBLE is written entirely in 6502 machine language to provide greater speed in less memory. The package consists of three combined program components, occupying a total of 6K bytes of memory starting at location $8ØØ. The remainder of the machine (excluding DOS, if a diskette drive is used) may be used to load and run your own BABBLE programs. This means that BABBLE will run on tape based systems from 16K up and on diskette based systems from 32K up. BABBLE can be made to run on smaller systems, but this is not recommended.

Although written in assembly language, BABBLE does make use of some subroutines in the integer BASIC ROM (notably the random number generator). To insure the availability of these subroutines and to make running the program easier, BABBLE has been made to look like an integer BASIC program.

A a B b C c D d E e F f G g H h I i J j K k L l M m N n O o P p Q q R r S s
T t U u V v W w X x Y y Z z 0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f G g
H h I i J j K k L l M m N n O o P p Q q R r S s T t U u V v W w X x Y y Z z

GETTING STARTED

The BABBLE package consists of a cassette or diskette and this booklet. The actual program is identical for both cassette and diskette, so if you don't have a diskette drive now but end up buying one later, you need only LOAD the program from cassette and SAVE it on diskette.

If you have the diskette version of BABBLE you may run the program by merely booting the diskette. BABBLE will begin execution automatically. If DOS is already active you may type:

```
RUN BABBLE
```

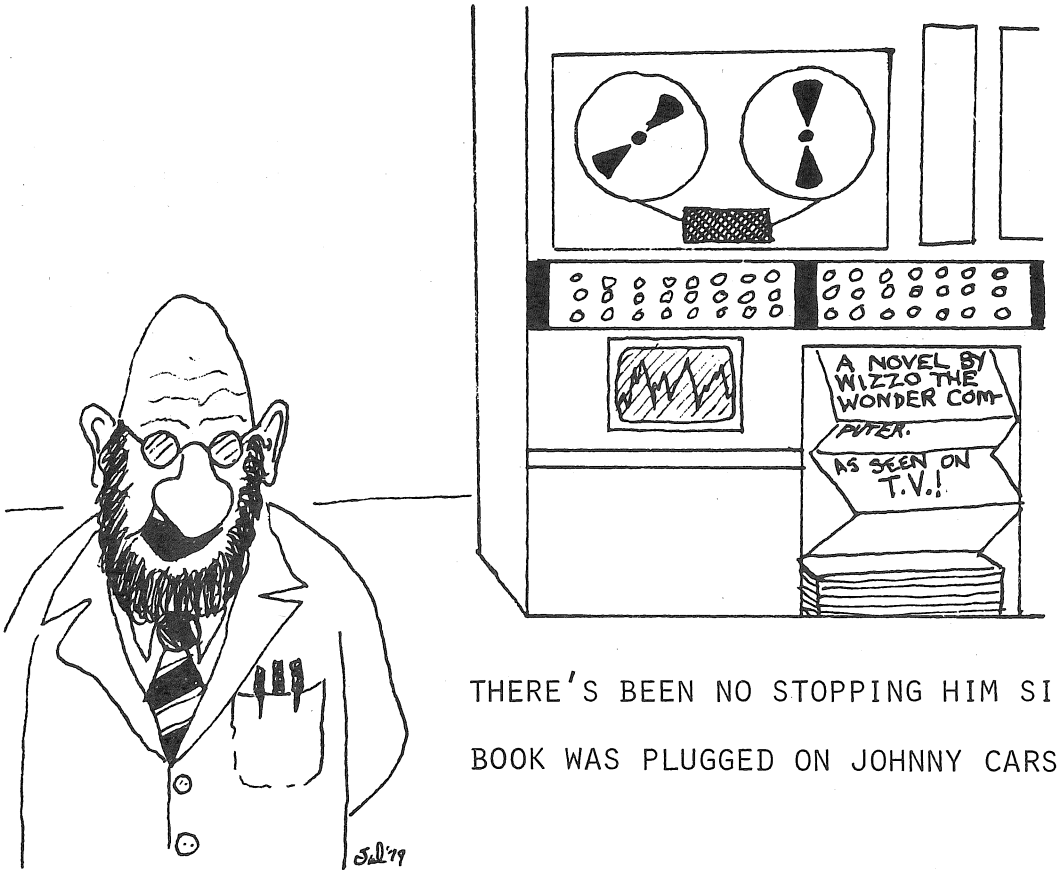
Integer BASIC will be activated and BABBLE will begin executing. At this point you will find yourself in the BABBLE editor. On the screen you will find a list of demonstration BABBLE programs (STORY, JINGLE, or GRAPHICS, for example). To try one of these type "L" for "load" followed by a space and the name and hit RETURN. For example:

```
L STORY
```

The demonstration program "STORY" will be loaded from diskette, the drive will stop, and the screen will change. Now type:

```
B
```

to begin the demonstration. After the demo you will be returned to the editor where you may load another program if you wish. Further use of the editor will be covered later.



THERE'S BEEN NO STOPPING HIM SINCE HIS BOOK WAS PLUGGED ON JOHNNY CARSON

If you are a tape user of BABBLE, go into integer BASIC and LOAD the first file on the cassette (side A) as follows:

```
(reset)
(ctl)B                    (Get into BASIC)
>LOAD                     (LOAD first file on side A)
>RUN                      (Begin running BABBLE)
```

(If you wish to make a copy of BABBLE to another cassette or to diskette, do so before RUNning the program.) At this point you will find yourself in the BABBLE editor. On the opposite side of the tape (side B) are several demonstration programs. To load the first, type:

```
L
```

hit RETURN and begin playing side B of the tape. When the file is loaded (the screen will change to show the new program) stop the recorder and type:

```
B
```

to start the demo. When the program finishes you will be returned to the BABBLE editor where you may load the next consecutive demo file from the tape with the "L" command in the same way.

A BABBLE TUTORIAL

Perhaps the easiest way to learn how to use BABBLE is to actually enter a program and run it. Before reading further, LOAD and RUN BABBLE, then perform the example session given here on your own APPLE.

When you first run BABBLE you will be in the editor. To clear out any left over BABBLE programs and to prepare to enter a new one, type:

```
N
```

(Just like the BASIC command, NEW) and hit RETURN. The screen will clear and the editor's text file will be empty. Add lines of BABBLE statements by typing the following lines, ending each with a RETURN (type them exactly as shown, including spaces). Don't worry about what they mean yet.

```
ASENTENCE=NAME / VERB / OBJECT .
AVERB=LOVES=KISSES=IS
ANAME=DON=BOB=FRANK
AOBJECT=ARTICLE / THING
AARTICLE=A=THIS=THE=MY
ATHING=DOG=TREE=HOUSE=CAR
A/=' '
```

The "A" at the beginning of each line is a command to the editor to add the text that follows as a new line to the end of the program. As each line is added, it will appear at the top of the screen and the line count at the bottom will increase by one.

When you have entered all seven lines, type:

    1

followed by a RETURN. The entire program will appear on the screen like this:

```
                    B A B B L E
>SENTENCE=NAME / VERB / OBJECT
>VERB=LOVES=KISSES=IS
>NAME=DON=BOB=FRANK
>OBJECT=ARTICLE / THING
>ARTICLE=A=THIS=THE=MY
>THING=DOG=TREE=HOUSE=CAR
>/=' '
                    1 OF 7
?
```

The "1" command tells the BABBLE editor to position its screen display "window" so that the first line in the program (line 1) appears on the top line. The top line is also called the "current" line, since this is the line you may currently modify if you wish. The bottom of the display always tells you which line is current and how many lines are in the program (in this example, "1 OF 7" means line 1 is current and there are 7 lines in the program). Unlike BASIC, where line numbers are permanently assigned to a line of text, BABBLE's line numbers are implicit. In other words, the fourth line of text in the file is line 4. If a line of text is inserted before line 4, the new line automatically becomes line 4 and the old line 4 is now line 5.

Now try out the program by typing:

    B

and hitting RETURN. This command tells the editor to begin BABBLING the program you have entered. A screen will flash by quickly (if you blink you'll miss it) that says "CHECKING FOR ERRORS" and then the results of the program will appear, something like this:

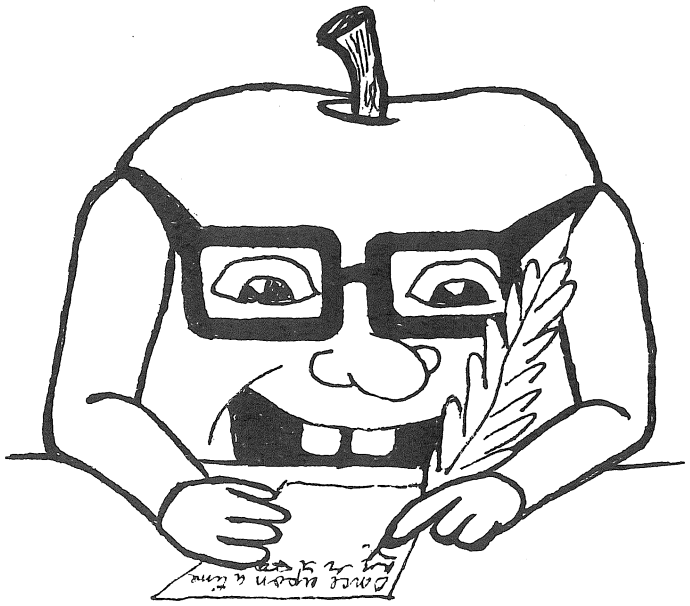
FRANK KISSES MY HOUSE.

BABBLE AGAIN? (Y/N)

The sentence at the top of the screen will probably be different because BABBLE is driven by a random number generator. If you want to see another random sentence, type:

    Y

(no RETURN is needed here). Each time you hit "Y", your BABBLE program is executed again from the beginning. If you type "N" (or anything other than "Y") you will be returned to the BABBLE editor. Type "N" now.



Back in the editor, suppose we want to add the names "CARLEY", "LYNN", and "ESTHER" as possibilities to the sentences BABBLE generates. To do this we must first make the line containing the names "current" by typing its number:

```
3
```

and hitting RETURN. The line:

```
NAME=DON=BOB=FRANK
```

will appear at the top of the screen. To replace it type:

```
RNAME=DON=BOB=FRANK=CARLEY=LYNN=ESTHER
```

and hit RETURN. The "R" command replaces the current line with the text that follows. Now type "B" to try out the new program.

Let's try inserting a line into the program between lines 1 and 2. To do this, first make line 2 the "current" line. By now you realize that this is done by typing:

```
2
```

and RETURN. Then type:

```
ISENTENCE=NAME / VERB / NAME .
```

The "I" command inserts a new line, containing the text that follows, before the "current" line. Sentences like:

```
DON LOVES CARLEY.
```

are now possible. Try "B"ABBLING the program. Using the commands outlined above, try adding or replacing the NAME words, THING words, or ARTICLE words with your own to create the silliest sentences you can think of. You're beginning to BABBLE!

EDITOR COMMANDS

Now that you've begun to get the hang of how to use the BABBLE editor (even if writing your own programs is yet to come), try some of these commands (of course, some of them you already know):

N - New file. Delete all lines in the editor file so that a new program may be typed in.

number - A line number (3, for example). Make that line "current" by positioning it at the top of the screen.

Rtext - Replace the "current" line with the text that follows.

R - "R" by itself deletes the "current" line from the file.

Itext - Insert a new line containing the text before the "current" line.

Atext - Add a new line containing the text at the end of the file.

U - Move up one line in the file.

Unumber - Move up 'number' lines. U5 moves up 5 lines, for example.

D - Move down one line in the file.

Dnumber - Move down 'number' lines.

S - Save the program on tape (as you would a BASIC program).

S filename - Save the program on disk into a DOS text file. (The last drive referenced is used.)

L - Load a program from tape. This does a New first, just as the BASIC LOAD command does.

L filename - Load a program from a DOS text file on diskette.

B - Begin BABBLING the program.

Bnumber - BABBLE 'number' times in a row.

T - Test BABBLE a program. Same as "B" except this command is used when testing a new BABBLE program. This will be discussed more later.

E - Exit from BABBLE.

If you hit RETURN without typing anything, a D10 command is done for you. This way you can "page" through a long program to see all of it. Do not enter any command line to BABBLE longer than 128 characters.

NOTE: Since the BABBLE editor operates on standard DOS T (text) type files, disk users may even use it to create EXEC files, move text files from disk to disk (or to tape), or update input files to your BASIC programs.

BASIC BABBLING

To understand how one writes a BABBLE program, let's look at our example again:

```
SENTENCE=NAME / VERB / OBJECT .
VERB=LOVES=KISSES=IS
NAME=DON=BOB=FRANK
OBJECT=ARTICLE / THING
ARTICLE=A=THIS=THE=MY
THING=DOG=TREE=HOUSE=CAR
/=' '
```

The above program consists of seven statements or "rules" that the computer will follow to generate, in this case, sentences. Each rule is one or more definitions for a given word. The word "SENTENCE" is defined to be (or will be replaced by) the word "NAME", followed by the word "/", followed by the word "VERB", then another "/", the word "OBJECT", and finally the word ".". Thus, a rule has the form:

word-being-defined=word1 word2 word3 ... etc.

Notice that the words in the definition must be separated by one or more spaces. BABBLE's job is to generate the first word you define by replacing it with its definition, replacing each word in that definition with their definitions, and so on, until the "terminal words" (words that are not defined) are encountered and printed.

If BABBLE encounters a word that has more than one definition, it must pick one at random:

VERB=LOVES=KISSES=IS

can also be written...

```
VERB=LOVES
VERB=KISSES
VERB=IS
```

The word VERB is multiply defined (it has three definitions) so each time BABBLE wants to generate it, one definition is chosen at random (VERB=KISSES, for example).

Each time BABBLE encounters a word in a definition, it scans all the statements for a definition of the word. If it can't find one, the word is "terminal" and must be printed as is.

Thus, in the above example, "SENTENCE" becomes "NAME", "/", "VERB", "/", "OBJECT", and ".". "NAME" becomes "DON", "BOB", or "FRANK". "/" becomes a space. "VERB" becomes "LOVES", "KISSES", or "IS", and so on. What makes things interesting is the fact that BABBLE picks multiple definitions randomly. Thus,

BOB KISSES MY HOUSE.

is just as much a possibility as...

DON IS A DOG.

This, basically, is all there is to a BABBLE program. There are a few additional ways to make it easier to write programs:

```
WORD=3*NAME
```

is the same as writing...

```
WORD=NAME NAME NAME
```

but shorter. Also...

```
WORD=(3)NAME=THING
```

is the same as...

```
WORD=NAME=NAME=NAME=THING
```

or...

```
WORD=NAME
WORD=NAME
WORD=NAME
WORD=THING
```

This implies that "NAME" will be chosen three times as often as a definition for "WORD" as would "THING".

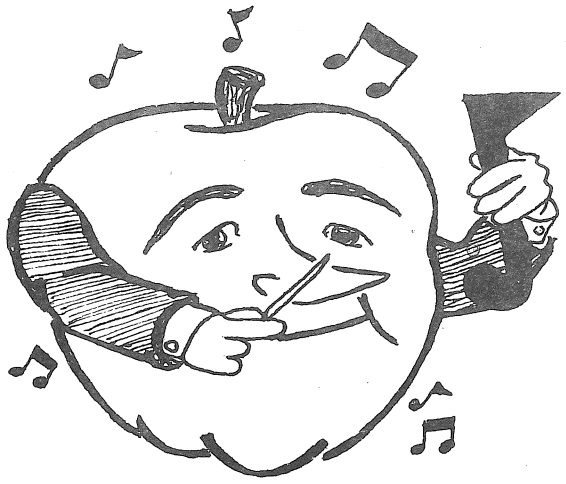
```
(3)WORD=NAME
WORD=THING
```

would have produced the same results. Here the whole statement is duplicated 3 times. So...

```
WORD=NOUN
(2)WORD=VERB=(2)THING
```

can be thought of as...

```
WORD=NOUN
WORD=VERB
WORD=THING
WORD=THING
WORD=VERB
WORD=THING
WORD=THING
```

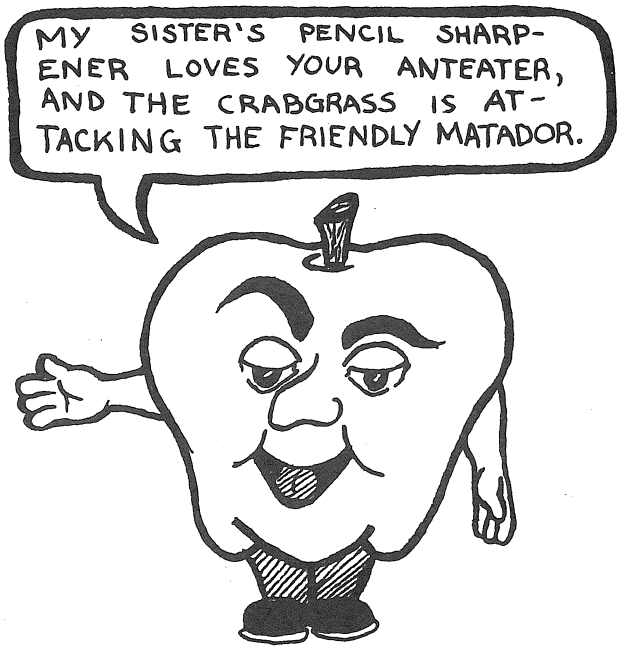


You may continue a multiple definition line as follows:

```
NAME=DON=BOB=FRANK=CARLEY=LYNN=ESTHER
=MELISSA=GEORGE=FRED
```

You may put your BABBLE statements in any order and they will work the same way (except continuations like that above) but remember that the first word defined is the one BABBLE will generate.

APPLES SAY THE DARNEDEST THINGS



LIMITATIONS

A BABBLE word may not begin with a numeral (0-9) nor may it contain any of the following characters:

= space # ' ( ) *

If one of these is to be used, enclose the word in single quotes:

```
WORD='E=MC**2 WON''T WORK (VERY OFTEN)'
```

Notice that single quotes within the word must appear twice. Only one ' will be printed, however.

Words may not exceed the length of a line (128 characters) but you can have the effect of large words by using definitions like the following:

```
BIGWORD=PART1 PART2 PART3
PART1=SUPERCALI
PART2=FRAGILISTIC
PART3=EXPIALIDOCIOUS
```

Try to avoid recursive definitions (defining a word in terms of itself), for instance:

```
VERB=ADVERB / VERB
(3)VERB=HITS=LOVES=TOUCHES
```

BABBLE will allow you to "nest" up to 80 levels deep before a terminal word is found, but defining a word in terms of itself will quickly use up these levels. The program:

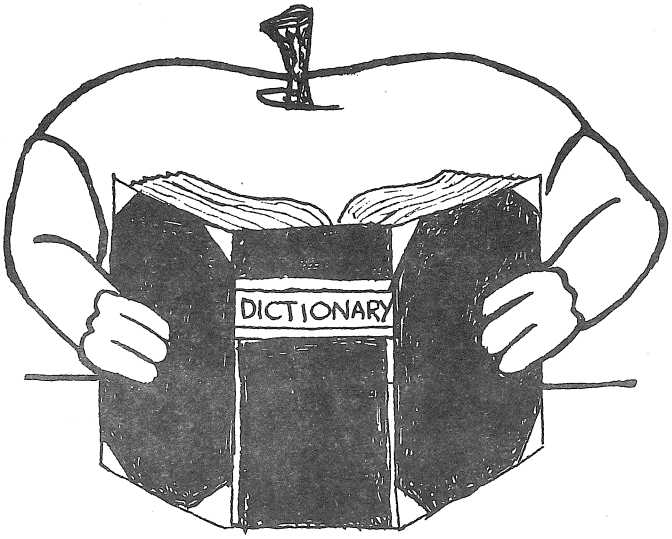
```
P=P
```

will fail in this way. Try it and see what happens.

Numbers used as multiplication and duplication factors may not exceed 255. To create 1000 occurances of a word, do this:

```
WORD=100*WORD1
WORD1=10*'PRINT THIS 1000 TIMES'
```

In this manner you can generate incredibly huge quantities.



SPECIAL WORDS

The following special words may be used in your BABBLE programs to do graphics, music, textual formatting, and to control execution. Whenever BABBLE encounters one of these words, instead of printing it, its function is performed. As an example, the program:

```
WORD=#C #>12 #^17 HELLO
```

clears the screen and writes "HELLO" in the exact center.

#C - Clear text screen, home cursor
#S - Skip to next line on screen
#Snumber - Skip 'number' lines on screen
#^number - Position to line 'number' (VTAB) (1-24)
#^ - Position to a random line on screen
#>number - Position to column 'number' (HTAB) (1-40)
#> - Position to a random column on screen
#E - End this execution of BABBLE now
#W - Wait for any keyboard input before proceeding
#Wnumber - Wait 'number' tenths of a second or for any keyboard input before proceeding (#W10 waits one second)
#Bnumber - Beep the APPLE's speaker. 'number' is a tone value (0-255)
#B - Random beep tone
#I - Set inverse mode
#N - Set normal mode
#F - Set flashing mode
#G - Set LORES mixed graphics mode and clear graphics screen
#T - Set text mode only
#Xnumber - Set next horizontal plot position to 'number' (0-39)
#X - Pick a random horizontal plot position
#Ynumber - Set next vertical plot position to 'number' (0-39)
#Y - Pick a random vertical plot position
#U - Move next plot position up one
#Unumber - Move next plot position up 'number'
#D - Move next plot position down one
#Dnumber - Move next plot position down 'number'
#R - Move next plot position right one
#Rnumber - Move next plot position right 'number'
#L - Move next plot position left one
#Lnumber - Move next plot position left 'number'
#Pnumber - Plot color 'number' (0-15) at current plot position (EX: #X10 #Y10 #P15 would put a white brick at 10,10)
#P - Pick any random color and plot it at the current position
#Hnumber - Draw a horizontal line 'number' bricks long from the current plot position to the right, using the last color used by #P
#H - Draw a random length horizontal line as above
#Vnumber - Draw a vertical line 'number' bricks long from the current plot position down, using the last color used by #P
#V - Draw a random length vertical line as above
#J - Jump back to the BABBLE editor immediately
#Jnumber - Jump (CALL) to subroutine (machine language) at location 'number' (0-65535) X and Y regs contain #X and #Y values
#?'text'* - At compilation time, prompts user with the text string. Whatever is entered replaces this word in the program. This allows dynamic "fill in the blanks" in stories or sentences.
#Mnumber* - Memorize the current value of the random number generator seed and store it in memory variable 'number' (0-15)
#M * - Same as #M0
#Anumber* - Alter random number generator seed using value in memory variable 'number'. If original seed is active, #M it into variable 0 first
#A * - Restore original seed from memory variable 0

* See ADVANCED TOPICS section for more details on these.

ADVANCED TOPICS

If you have read everything up to this point there is a good possibility that you are lost. BABBLE is a very powerful and, in places, complex program, and the only way to teach you to use it, short of writing a monstrous manual, is to provide examples. It would be a good idea for you to do the following:

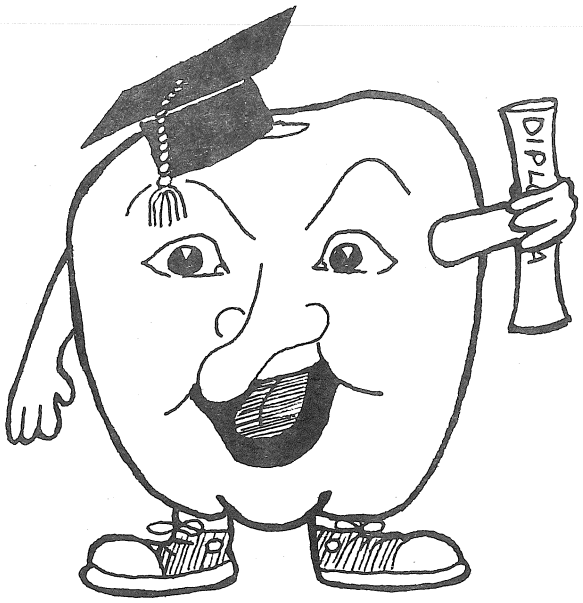
* Run each of the demonstration programs
* Look at their 'source' code in the editor to see how they work. If you don't understand something, don't worry about it yet.
* Try writing a simple program of your own. Start small and add to it as you try out new special words, etc.

The remainder of this manual will cover some special cases which may be of use to you and might clarify techniques used in the demo programs.

RANDOM SEEDS

Within BABBLE there are 16 "memory variables" into which you may save the state or "seed" of the random number generator. Using the #M and #A words you can actually control which words BABBLE will pick. Suppose you are creating rhymes:

```
VERSE1=ROSES / ARE / RED
VERSE2=VIOLETS / ARE / #M1 COLOR
VERSE3=SUGAR / IS / SWEET
VERSE4=AND / YOU / ARE / #A1 WHAT
COLOR=BLUE=GREEN=WHITE=BLACK
WHAT=#A TOO=#A MEAN=#A LIGHT=#A BACK
/=' '
```



In this way the random number generator is altered in VERSE4 to the same value it had when picking a color in VERSE2. By putting words that rhyme in the same relative positions in the definitions for COLOR and WHAT, no matter which color is chosen, the proper rhyming word will also be chosen. Look at JINGLE or STORY for more examples of this.

FILLING IN THE BLANKS

It is relatively easy to produce fill-in-the-blanks stories using the #? special word. Consider this example:

```
STORY='ONCE UPON A TIME I' #?'GIVE ME A VERB' ' MY ' #?'GIVE ME A BODY PART' .
VERB=BROKE=BENT=KICKED
PART=LEG=BACK=HEAD
```

When this is compiled, the user will be asked for a verb and a body part. If he answers "VERB" to the first question, since the word "VERB" has a definition, a word is chosen for him. If he says:

```
GIVE ME A VERB
?QUICKLY BIT
GIVE ME A BODY PART
?BELLY BUTTON
```

he gets...

ONCE UPON A TIME I QUICKLY BIT MY BELLY BUTTON.

See the demo STORY for more examples of how this is done.

THE TEST COMMAND

If you are having trouble with a BABBLE program you are writing, type T in the editor. Your program will be compiled as always but, instead of executing it right away, BABBLE will give you a list of the terminal words (words without any definition). While executing, BABBLE will print each non-terminal word in inverse mode as it is encountered to allow you to trace its path through your program. Also, all special words are printed without being executed.

IF YOU HIT RESET

If you accidentally (or intentionally) hit RESET, you can get back into BABBLE without losing your file by going into the monitor and typing:

```
*803G
```

If you want to reBABBLE a program after answering the BABBLE AGAIN? (Y/N) question with a N, you can avoid recompiling by hitting RESET (to get into the monitor) and typing:

```
*1800G
```

STOPPING BABBLE

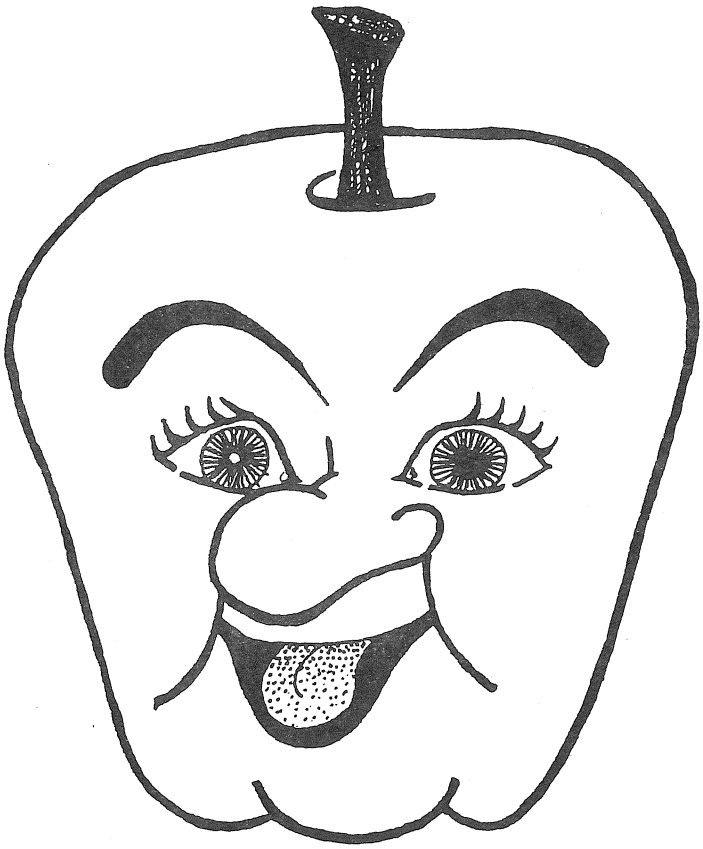
If your program runs amuk, don't hit RESET, type control-C as you would in BASIC. At any time during execution you can halt BABBLE temporarily by hitting any key. Hit any key again to resume where you left off.

POSTSCRIPT

I hope you find BABBLE as intriguing to play with as I do. I would be very interested in collecting well written BABBLE programs to add to or replace those in future copies of the program. If you have written one, contact me at the SOFTWARE FACTORY. Here are some ideas to get you started:

RANDOM BASIC PROGRAMS (syntactically correct but who knows what they would do?)
MORE FILL-IN-THE-BLANKS STORIES
RANDOM HORROR OR SCI-FI STORIES
STRANGE AND WONDERFUL GRAPHICS DISPLAYS
ANIMATION
HIRES (using the #J interface)
RANDOM "FOREIGN" LANGUAGES

DON D. WORTH



AS I JUST EMBARKED ON BOARD SHIP
I MET A BEAST UPON ITS HIP
IT CALLED ITSELF MACARONE
I 'SPECT IT WAS PART JAPANEE

AS I WAS STROLLING IN THE SUN
I MET THIS SON WHO SHOT A GUN
AS I APPROACHED HE TURNED TO SCOLD
I DASHED RIGHT BACK TO MY HOUSEHOLD

WHILE I WAS BASKING THROUGH THE ZOO
I MET SOME BRIGHT BLUE CURLIQUE
THAT IT WAS STRANGE I'LL CLARIFY
I WONDER WHY IT STOLE YOUR PIE

AS I WAS PLAYING IN THE DARK
I FOUND SOME DAMSEL HID BY BARK
HER SHOES WERE FULL OF OPIUM
I GUESS SHE THOUGHT I WAS HER MUM

WHILE I WAS WALKING IN THE PARK
I MET THIS CREATURE IN AN ARK
AT SUCH A SIGHT I ATE YOUR HAT
IT TOOK ME FOR A DEMOCRAT

HERE IS A THING YOU NEVER KNEW
I FOUND THIS BRIGHT GREEN CURLIQUE
IT WAS SO WEIRD I DID POPEYE
I THINK IT WAS AS WEIRD AS I

WHEN I WAS WALKING IN THE DARK
I MET THIS HORSEMAN WITH A SHARK
HE WAS SO BENT HE COULD NOT FLEE
I DON'T KNOW WHAT HE THOUGHT OF THEE

HERE IS A THING YOU CAN'T ESCHEW
I SAW SOME BRIGHT BLUE COCKATOO
HE SMELLED JUST LIKE A BIG POLECAT
I WENT AND GAVE HIM TIT-FOR-TAT

I MET A THING THAT MADE ME RUN
I CAME UPON A SKELETON
IT CALLED ITSELF ANTIGONE
I GUESS IT WAS PART CHIMPANZEE

PLEASE LISTEN UP, MERCI BEAUCOUP
I MET A LULU FROM PERU
WHEN I CAME BY SHE DRANK SOME RUM
SHE REALLY WAS QUITE CUMBERSOME

HERE'S A THING YOU CAN NOT EQUATE
I SAW A GUY WHOSE NAME WAS KATE
AT SUCH A SIGHT I SHOUTED "SCAT!"
HE TOOK ME FOR AN ACROBAT

WHILE I WAS PLAYING THROUGH THE ZOO
I SPIED A BRIGHT BLUE KANGAROO
WHEN I APPROACHED SHE KISSED A PLUM
SHE WAS FORSOOTH QUITE QUARRELSOME

ONCE WHEN I WAS ON AN ESTATE
I MET SOME MAN WHO ATE AND ATE
AT SUCH A SIGHT I ATE THAT RAT
HE TOOK ME FOR A DIPLOMAT

THE OTHER DAY I HAD SOME FUN
I CHANCED UPON A HAIRY HUN
HE WAS A STRANGE SIGHT TO UNFOLD
AND NEVER HAVE I FELT SO OLD

AS I WAS BASKING IN THE SUN
I SAW THIS NUN WHO SHOT A GUN
SHE WAS SO BENT SHE COULD NOT SKI
I WONDER WHAT SHE THOUGHT OF TEA

THIS STORY YOU CAN NOT BERATE
I MET SOME BEAST WHO COULD INFLATE
WHEN I APPROACHED IT TURNED TO GOLD
I RAN RIGHT BACK TO MY STRONGHOLD

AS I JUST STARTED ON A TRIP
I SPIED A THING UPON ITS HIP
THAT IT WAS STRANGE I'LL CERTIFY
I WONDER WHY IT ATE MY EYE

I FOUND A THING SECOND TO NONE
THERE WAS THIS BUN WHO SHOT A GUN
THAT IT WAS ODD I'LL AMPLIFY
I WONDER WHY IT KISSED MY THIGH

AS I WAS WHISTLING IN THE SUN
THERE WAS A NUN WHO TOLD A PUN
SHE WAS A QUEER SIGHT TO BEHOLD
AND NEVER HAVE I FELT SO OLD

ONCE WHEN I WAS UPON A DATE
I SAW SOME THING THAT DID GYRATE
IT SMELLED JUST LIKE MY BROTHER NAT
I WENT AND GAVE IT THIS AND THAT

WHEN I WAS PLAYING IN THE PARK
I SPIED A DAMSEL HID BY BARK
HER PANTS WERE FULL OF RADIUM
I 'SPECT SHE THOUGHT I WAS HER CHUM

I SPIED A THING THAT MADE ME RUN
THERE WAS A SON WHO SHOT A GUN
AS I CAME BY HE STOLE SOME GUM
HE REALLY WAS QUITE FROLICSOME

The
Software Factory
P.O. BOX 904
CHATSWORTH, CA 91311