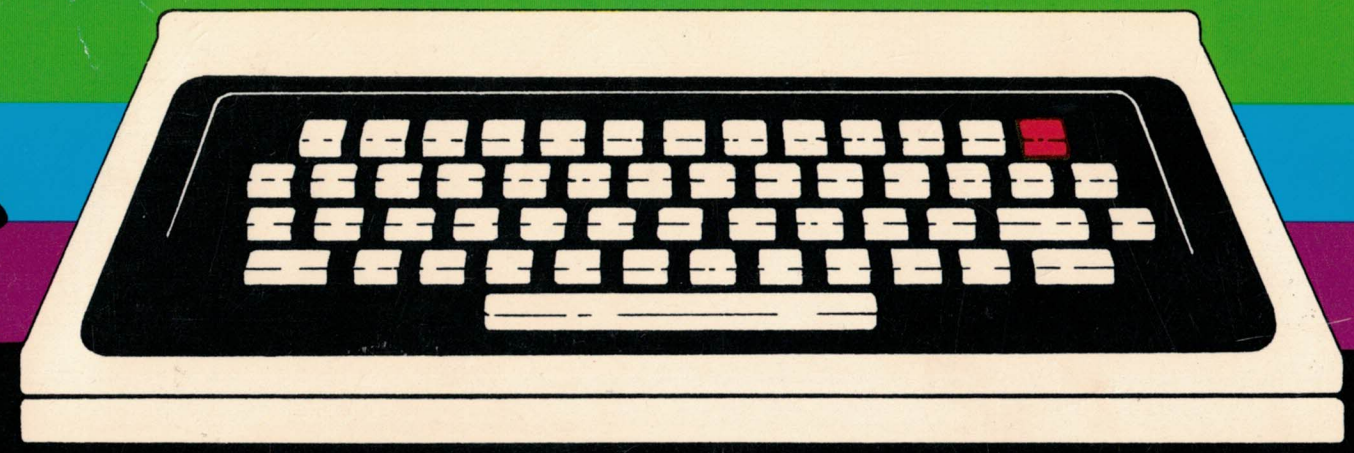


Radio Shack®

**GETTING
STARTED
WITH
COLOR
BASIC**

TRS-80™ COLOR COMPUTER





GETTING STARTED WITH COLOR BASIC

THE COLOR BASIC SYSTEM IS A COMPLETELY NEW
MANNER OF COLORING. IT IS THE ONLY SYSTEM
WHICH GIVES YOU THE BEST OF BOTH WORLDS.

IT IS THE ONLY SYSTEM WHICH GIVES YOU
THE BEST OF BOTH WORLDS. IT IS THE ONLY
SYSTEM WHICH GIVES YOU THE BEST OF BOTH
WORLDS. IT IS THE ONLY SYSTEM WHICH
GIVES YOU THE BEST OF BOTH WORLDS.

IT IS THE ONLY SYSTEM WHICH GIVES YOU
THE BEST OF BOTH WORLDS. IT IS THE ONLY
SYSTEM WHICH GIVES YOU THE BEST OF BOTH
WORLDS. IT IS THE ONLY SYSTEM WHICH
GIVES YOU THE BEST OF BOTH WORLDS.

IT IS THE ONLY SYSTEM WHICH GIVES YOU
THE BEST OF BOTH WORLDS. IT IS THE ONLY
SYSTEM WHICH GIVES YOU THE BEST OF BOTH
WORLDS. IT IS THE ONLY SYSTEM WHICH
GIVES YOU THE BEST OF BOTH WORLDS.

*Getting Started with COLOR BASIC: © 1980
Tandy Corporation, Fort Worth, Texas 76102
U.S.A. All Rights Reserved.*

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual or from the use of the information obtained herein.

*TRS-80 Color Computer System Software: © 1980
Tandy Corporation and Microsoft. All Rights Reserved.*

The system software in the Color Computer micro-computer is retained in a read-only memory (ROM) format. All portions of this system software, whether in the ROM format or other source code form format, and the ROM circuitry, are copyrighted and are the proprietary and trade secret information of Tandy Corporation and Microsoft. Use, reproduction or publication of any portion of this material without the prior written authorization by Tandy Corporation is strictly prohibited.

INTRODUCTION

This book is for those of you who don't know anything about Computers, and would like to be spared the long, technical explanations!

Using this as your guide, you'll be able to interact and enjoy your Computer *right away*. Later, you'll probably want to read some of our thorough and detailed books. For now, though, the easiest way to get started with your Computer is to use it and have fun doing it.

You'll find many of the things we have you do — especially in the first chapters — are games, songs, or other fun-type programs. This is not to say your TRS-80 can't be used for practical programs. It's every bit as powerful as other computers its size. We start you off with the fun programs because it's the easiest way for you to feel at ease with your Computer. Once you feel it's truly an extension of yourself, you'll find it much easier to write any kind of program you want.

So sit down and spend a couple of hours with it. Type away at it. Play with it. Try to make it do strange things. In other words. . . get to feeling comfortable with it. There's an infinite number of things it can do for you.

TO GET STARTED . . .

Connect your Computer by referring to the Chapters on "Installation" and "Operation", and to Figure 1 in your *TRS-80 Color Computer Operation Manual*.

Then power up your Computer:

- Turn ON your television set
- Select channel 3 or 4
- Set the antenna switch to "COMPUTER"
- Turn ON the Computer. The POWER button is on the left rear of your keyboard (when you're facing the front).

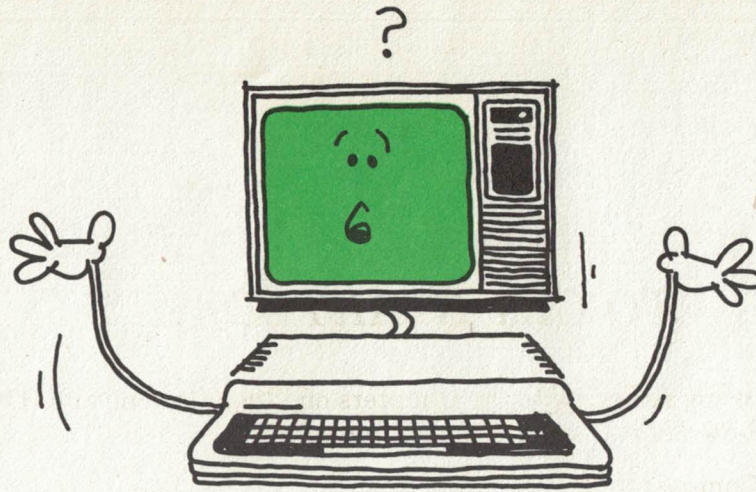
This message should appear:

```
COLOR BASIC v.r  
© 1980 TANDY  
OK
```

(*v.r* is two numbers specifying which version and release you have).

If you don't get this message, turn the computer on and off again. Adjust the Brightness and Contrast on your T.V. set. Check all the connections. If you still don't get this message, refer to "Troubleshooting and Maintenance" in the *TRS-80 Color Computer Operations Manual*.

Once you do get this message, you're ready to begin.



HOW DO YOU TALK TO A COMPUTER?

In this book, you'll learn how to talk to your Computer. That's all programming is, by the way. Once you learn how to communicate, you'll be able to get your Computer to do whatever you tell it. (well, almost).

The Computer understands a language called COLOR BASIC. COLOR BASIC is form of BASIC — Beginners All-purpose Symbolic Instruction Code. There are lots of computer languages. COLOR BASIC just happens to be the language your Computer understands.

We'll introduce BASIC words in the order that it's easiest to learn them. When you get mid-way in the book, you might forget what one of the words means. If this happens, simply look up the word in the back of the book or use your "Quick Reference Card" to find its meaning.

TABLE OF CONTENTS

CHAPTER 1: MEET YOUR COMPUTER.....	6
CHAPTER 2: YOUR COMPUTER NEVER FORGETS	
(unless you turn it off. . .).....	16
CHAPTER 3: SEE HOW EASY IT IS?	24
CHAPTER 4: COUNT THE BEAT	34
CHAPTER 5: SING OUT THE TIME	42
CHAPTER 6: DECISIONS, DECISIONS.....	54
CHAPTER 7: GAMES OF CHANCE.....	60
CHAPTER 8: SAVE IT ON TAPE	70
CHAPTER 9: COLOR YOUR SCREEN	76
CHAPTER 10: ONE FANTASTIC TEACHER.....	90
CHAPTER 11: HELP WITH MATH	102
CHAPTER 12: A GIFT WITH WORDS.....	112
CHAPTER 13: BEAT THE COMPUTER.....	124
WHAT NOW?	135
 APPENDIXES	
A / MUSICAL TONES	136
B / BASIC COLORS	138
C / PRINT @ SCREEN LOCATIONS	139
D / GRAPHICS SCREEN LOCATIONS	140
E / ANSWERS TO EXERCISES	141
F / SAMPLE PROGRAMS.....	144
G / ERROR MESSAGES	146
H / BASIC SUMMARY.....	148

CHAPTER 1



MEET YOUR COMPUTER



MEET YOUR COMPUTER

Is your Computer connected? Turned on? Ready to give it a first work-out?

In these first two Chapters, we're going to introduce you to your Computer — the way it thinks, some of its many talents, and even a couple of little quirks it has. By the time you finish these chapters, you'll be ready to program . . . *promise!*

Type away on the keyboard and then press the **ENTER** key.

Don't worry about anything but the last line of type on your screen. It should say:


OK

OK is the Computer's "prompt". It's telling you — "OK, enough foolishness . . . as soon as you are ready . . ." (It patiently waits for your command.) *You* are the Master — you can tell the Computer to do anything you wish.

Give it your first command. Type this *exactly* as it is below:

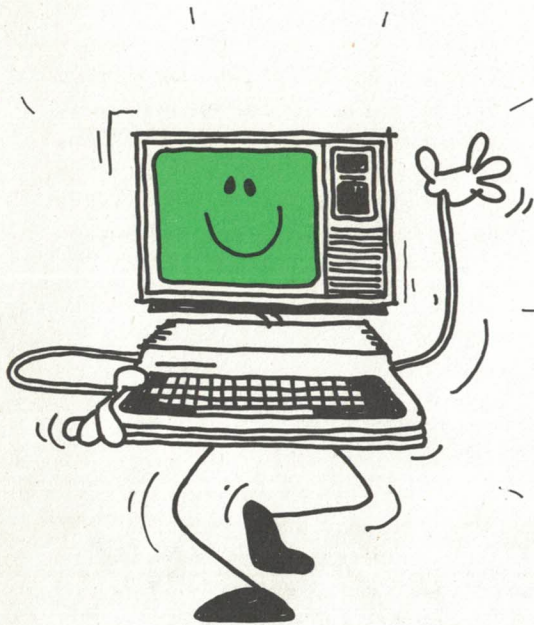
```
PRINT "HI, I'M YOUR COLOR COMPUTER"
```

When you reach the end of the line on your screen, keep on typing. The last part of the message will appear on the next line.




All letters you type should be **BLACK** with a **GREEN BACKGROUND**. If they're reversed (green with a black background), press the **SHIFT** and **0** (zero) keys at the same time.

See the blinking light? You can type something wherever you see it.



"Hi, I'm Your Color Computer!"

Now check your line. Did you put the quotation marks where we have them? If you made a mistake, no problem. Simply press the  key and the last character you typed will disappear. Press it again and the next to the last will disappear (. . . and so on and so on . . .).

Ready? This should be on your screen:

```
OK
PRINT "HI, I'M YOUR COLOR COMPUT
ER"
```

Press the **ENTER** key and watch. Your screen should look like this:

```
OK
PRINT "HI, I'M YOUR COLOR COMPUT
ER"
HI, I'M YOUR COLOR COMPUTER
OK
```

Your Computer just obeyed you by printing the message you had in quotes. Give it another message to print. Type:

```
PRINT "2"
```

Press **ENTER** The Computer again obeys you and prints your next message:

```
2
```

Try another one:

```
PRINT "2 + 2" ENTER
```

The Computer obeys you by printing:

```
2 + 2.
```

You probably expect a lot more than just an electronic mimic . . . like maybe

some answers! Well, try it without the quotation marks. Type:

```
PRINT 2 + 2 ENTER
```

Much better. This time the Computer prints the answer:

4

These quotation marks obviously must mean something. Try experimenting some more with them. Type each of these lines:

```
PRINT 5+4 ENTER  
PRINT "5+4" ENTER  
PRINT "5+4 EQUALS" 5+4 ENTER  
PRINT 6/2 " IS 6/2" ENTER  
PRINT "8/2" ENTER  
PRINT 8/2 ENTER
```

Have you come up with any conclusions on what the quotes do?



The Computer thinks of quotes like a journalist does. If the message is in quotes, the Computer must PRINT it exactly as it appears. If it's not in quotes, the Computer can interpret it by adding, subtracting, multiplying or dividing it.

RULES ON STRINGS VS NUMBERS

The Computer sees everything you type as *STRINGS* or *NUMBERS*. If it's in quotes, it's a *STRING*. The Computer sees it *EXACTLY* as it is. If it's not in quotes it's a *NUMBER*. The Computer will figure it out like a numerical problem.

A COLOR CALCULATOR, NO LESS!

Any arithmetic problem is a snap for your Computer. Let it do some long division. Type:

```
PRINT "3862 DIVIDED BY 13.2 IS" 3862/13.2 (ENTER)
```

Let's do a multiplication problem:

```
PRINT 1589 * 23 (ENTER)
```

Notice that the Computer's multiplication sign is an asterisk, rather than the X sign which you've always used in math. This is because the Computer is such a precise and literal creature that it would get the X multiplication sign mixed up with the X alphabetical character.

Try a few more problems:

```
PRINT "15 * 2 = " 15*2 (ENTER)
PRINT 18 * 18 " IS THE SQUARE OF 18" (ENTER)
PRINT 33.3/22.82 (ENTER)
```

Now it's your turn. Write two command lines which will print these two problems as well as their answers:

```
157 / 13.2 =
95 * 43 =
```

DO-IT-YOURSELF COMMAND LINES

Notice how the Computer handles parts in quotes vs. the parts not in quotes.

If you used "correct" command lines, this is what the Computer should have printed on your screen:

```
157 / 13.2 = 11.8939394
95 * 43 = 4085
```

Ready for the answers:

```
PRINT "157 / 13.2 =" 157/13.2
PRINT "95 * 43 =" 95*43
```

IT HAS ITS RULES . . .

By now, the Computer has probably printed some funny little messages on your screen. If it hasn't, type this line deliberately misspelling the word *PRINT*:

```
PRIINT "HI" ENTER
```

The Computer prints:

```
?SN ERROR
```

SN ERROR stands for "syntax" error. This is the Computer's way of saying "The command 'PRIINT' is not in my vocabulary . . . I have no earthly idea what you want me to do". Anytime you get a SN error, it's probably because you made some kind of typographical error.

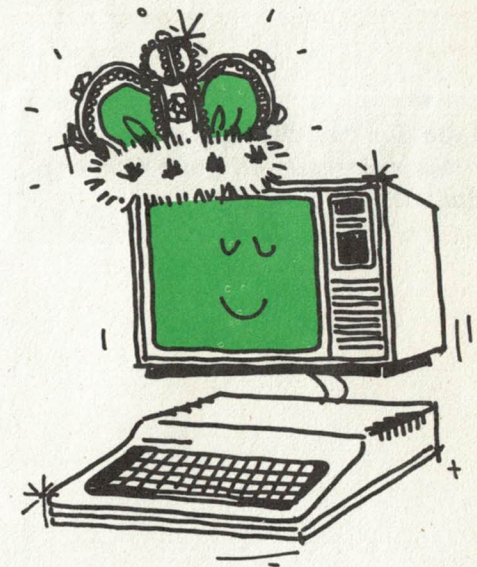
The Computer will also give you error messages when it *does* understand what you want it to do, but you're asking it to do something that it feels is *illogical* or *impossible*. For instance, try this:

```
PRINT 5/0 ENTER
```

The Computer prints:

```
?/0 ERROR
```

Actually there is no "correct" Command line. For that matter, there is no correct way of handling your Computer. There are many ways of getting it to do what you want. Relieved . . . Good!



Which means “Don’t ask me to divide by 0 — that’s impossible!!”

If you get a strange error message you don’t understand, flip back to the Appendix. We’ve listed all the error messages there and what probably caused them.

IT’S A SHOW OFF, TOO

So far, all you’ve seen your Computer do is silently print on a green screen. But your color Computer enjoys showing off. Type:

CLS(3) **ENTER**

Now your screen is a pretty shade of blue with a green stripe at the top. Your typed command told the Computer to clear the screen and print color number 3 — blue.

But why the green stripe? The Computer cannot type on a blue background. Anytime it types something on the screen, it must type it on a green background. Try typing some more characters. Notice that the Computer gives these characters a green background also.

Colors other than green are for printing graphics illustrations. We’ll spend lots more time with this color capability later.

Press **ENTER** so that you get the OK prompt on your screen. Type:

CLS(7) **ENTER**

Now you should have magenta (pinkish purple) on your screen with a green stripe at the top. Try some more colors if you like. Use any number from 0 to 8. Your color Computer has nine colors. Each color has a numeric code.



BUG: If you get a message saying **MICROSOFT** or an **?FC** Error message, it’s because you are using a number other than 0 through 8.

If you don’t get the right colors, refer to the color test in your Owner’s Manual.

Type CLS without a number code:

CLS **ENTER**

If you don't use a number code, the Computer assumes you just want a clear green screen.

COMPUTER SOUND OFF — ONE, TWO . . .

Type this:

SOUND 1, 100 **ENTER**

If you don't hear anything, turn up the volume and try again.

What you are hearing is 6 seconds of the lowest tone the Computer can hum. How about the highest tone? Type:

SOUND 255, 100 **ENTER**

OK, so it's got quite a hum-range . . . hope you're suitably impressed. Try some other numbers. Hope you like the Computer's voice (it's the only one it's got).

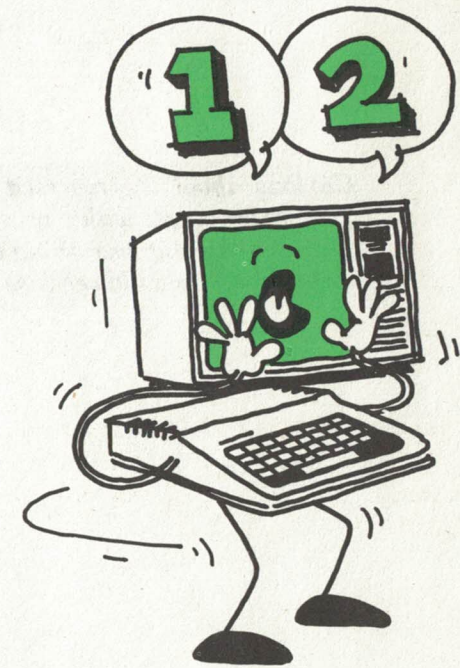
You want to know what the other number is for? (Or maybe you've already found out). The second number tells the Computer *how long* to hum the tone. You can use any number from 1 to 255. Try 1:

SOUND 128, 1 **ENTER**

and the Computer will hum the tone for about 6/100ths of a second. Try 10:

SOUND 128, 10 **ENTER**

The Computer sounds the tone for 6/10ths of a second. Try variations of both numbers, but stick to numbers between 1 and 255.



"Sound Off!"



BUG: Again, if you get an ?FC Error message, it's because you are using a number other than 1 through 255.

Curious about the reversed colors? They're for people with a printer. The printer will print everything typed in reversed colors as lower case letters.

ONE MORE THING . . .

Press the **SHIFT** and **0** (zero) keys, holding both down at the same time. Now type some letters. The letters you type should now be *green* on a *black background*. If they're not, try it again pressing **SHIFT** slightly before pressing **0**. Be sure to hold both keys down at the same time.

Now, with the colors "reversed", press **ENTER** and then type this simple command line:

PRINT "HI" **ENTER**

The Computer gives you an ?SN ERROR. It doesn't understand the command.

Press the **SHIFT** and **0** characters again and type some letters. They should be back to normal: *black* with the *green background*. Press **ENTER** and type the same command line again. This time, it'll work.

We just wanted to show you this in case you ever press **SHIFT** and **0** by a mistake. The computer can't understand any commands you type with reversed colors. If you ever find you're typing with these reversed colors, press the **SHIFT** and **0** keys to get the colors back to normal.

LEARNED IN CHAPTER 1

BASIC WORDS

PRINT
SOUND
CLS

KEYBOARD CHARACTERS



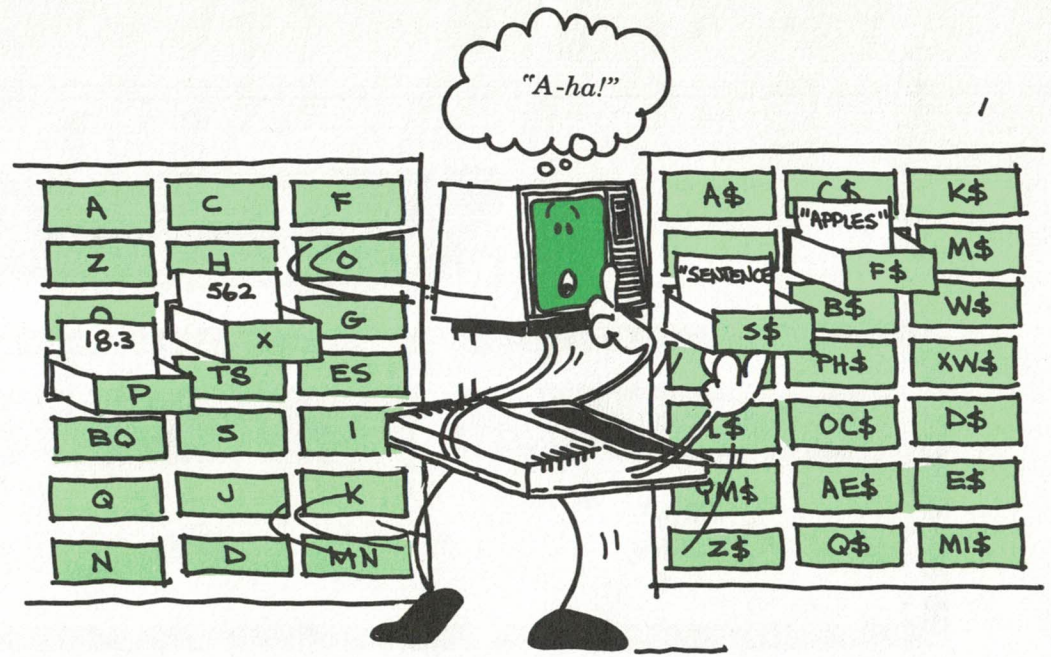
CONCEPTS

string vs. numbers
error messages

We'll put a list like this at the end of each chapter. It'll help you make sure you didn't miss anything.

NOTES:

CHAPTER 2



YOUR COMPUTER NEVER FORGETS
(... unless you turn it off ...)



YOUR COMPUTER NEVER FORGETS (... unless you turn it off...)

One of the things that makes your Computer so powerful is its ability to remember anything you ask it to. To make the Computer remember the number 13, type this:

```
A = 13 ENTER
```

Now type anything you want to confuse the Computer. When you're done, press **ENTER**. To see if the Computer remembers what A stands for, type:

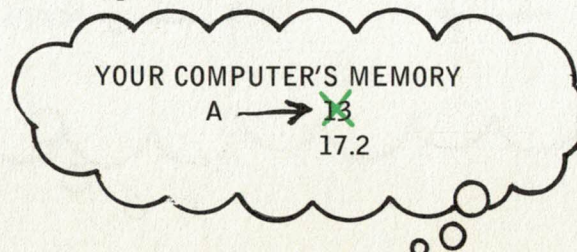
```
PRINT A ENTER
```

Your Computer will remember 13 as long as you have it *on* ... or until you do what we're going to do next. Type:

```
A = 17.2 ENTER
```

Now if you ask it to PRINT A, it will print the number 17.2.

This is what just happened in your Computer's memory:



*Did it get confused?
or forget?*



You don't have to use the letter A. You may use any letters from A to Z. (As a matter of fact, you can use any *two* letters). Try typing this:

```
B = 15 (ENTER)
C = 20 (ENTER)
BC = 25 (ENTER)
```

Have it print all your numbers. Type:

```
PRINT A, B, C, BC
```

To get it to remember a *string* of letters or numbers, put a dollar sign next to the letter. Type:

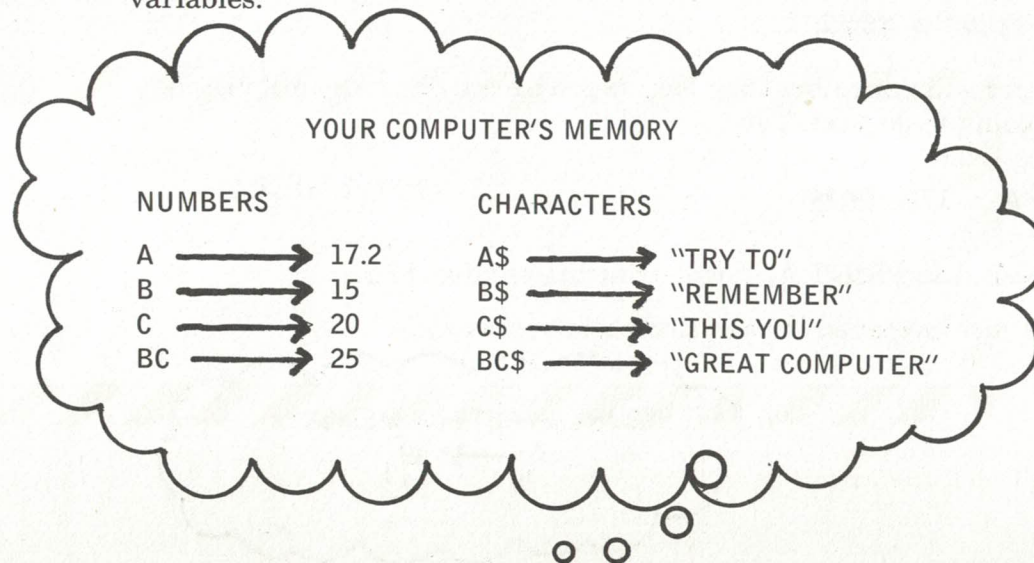
```
A$ = "TRY TO"
B$ = "REMEMBER"
C$ = "THIS YOU"
BC$ = "GREAT COMPUTER"
```

To the Computer, a dollar sign means it's a string.

Let's see how sharp your Computer is. Type:

```
PRINT A$, B$, C$, BC$ (ENTER)
```

Computer types call all these letters *variables*. So far, we've used these variables:



Try spot checking these variables to see if the Computer has remembered your information properly. For instance, type:

```
PRINT BC (ENTER)
```

To see if BC still contains 25.

You can think of these variables as little boxes where you can store your information. One set of boxes is for *strings*; the other set's for *numbers*. You use these variables to label each box.

Try to set the computer to remember a letter we haven't used yet. What happens . . . interesting . . .

THE COMPUTER IS FUSSY ABOUT ITS RULES

Do you think the Computer will accept these lines:

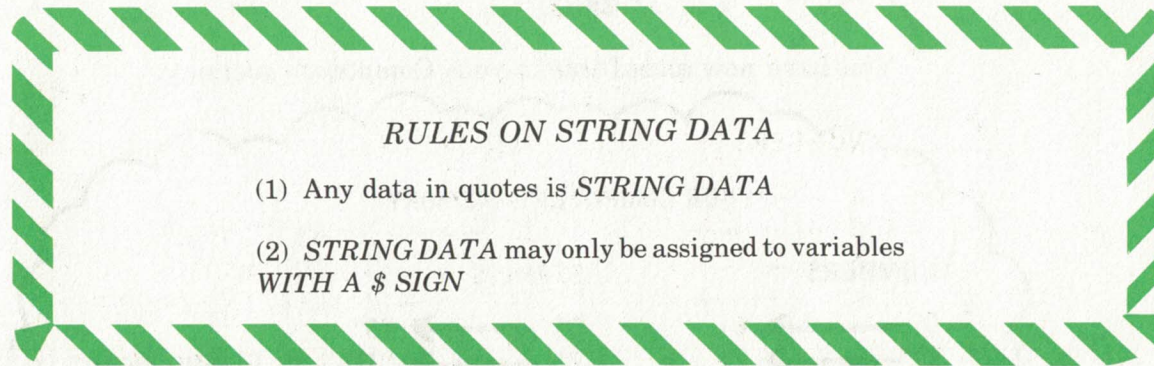
```
D = "6" (ENTER)  
Z = "THIS IS STRING DATA" (ENTER)
```

With both of these lines, the Computer responds with ?TM ERROR. It's telling you you've got to play according to *its rules*.

Like we said before, the Computer has it's rules and might get a little fussy with you if you don't play by them.

← *TM stands for Type MisMatch error. It means you didn't go by the rules.*

These are the rules you ignored:



To obey the Computer's rules, we have to put a dollar sign after D and Z. Type:

D\$ = "6" (ENTER)

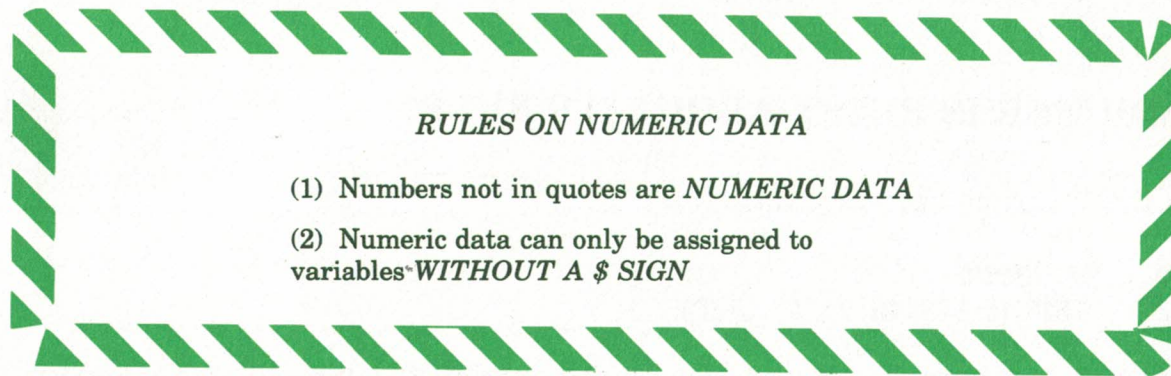
Z\$ = "THIS IS STRING DATA" (ENTER)

which the Computer accepts.

Do you think the Computer will accept this?

D\$ = 6 (ENTER)

These are the rules that this command ignored:



RULES ON NUMERIC DATA

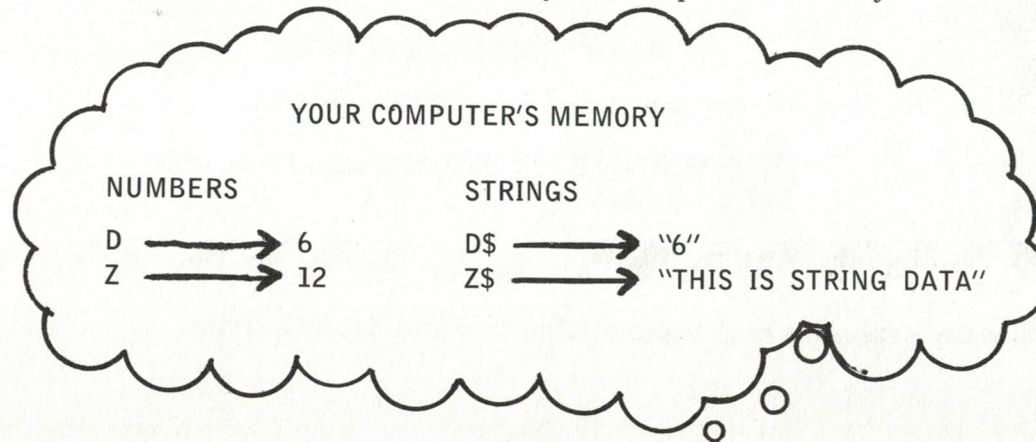
- (1) Numbers not in quotes are *NUMERIC DATA*
- (2) Numeric data can only be assigned to variables *WITHOUT A \$ SIGN*

Type this, which the Computer will accept:

D = 6 (ENTER)

Z = 12 (ENTER)

You have now added this to your Computer's memory.



Now you can do something interesting with these letters. Type:

```
PRINT D * 2 (ENTER)
```

The Computer prints the product of D times 2.

Try this line:

```
PRINT Z/D
```

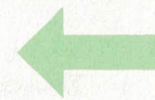
The Computer prints the quotient of Z divided by D.

Would this work:

```
PRINT D$ * 2 (ENTER)
```

Did you try it? This makes the Computer print the same ?TM ERROR. It *cannot multiply string data*.

Cross out the commands that the Computer will reject:



The computer remembers that D = 6.

EXERCISE WITH VARIABLES

```
F = 22.9999999  
M = "19.2"  
DZ$ = "REMEMBER THIS FOR ME"  
M$ = 15  
Z = F + F
```

Finished? This is what the Computer will *accept*.

```
F = 22.9999999  
DZ$ = "REMEMBER THIS FOR ME"  
Z = F + F
```

RULES ON VARIABLES

You may use any two characters from A-Z for a variable. If you want to assign it string data, put a dollar sign after it. Otherwise, it can only hold numeric data.

LEARNED IN CHAPTER 2

CONCEPTS

Variables
String vs. Numeric Variables

Now that you've learned how the Computer thinks it will be easy to write some programs. But before going to the next chapter, how about a break?

CHAPTER 3



SEE HOW EASY IT IS?



SEE HOW EASY IT IS?

Type:

NEW **(ENTER)**

This is just to erase anything that might be in the Computer's "memory".

Now type this line: Be sure you type the number 10 first — that's pretty important.

10 PRINT "HI, I'M YOUR COLOR COMPUTER" **(ENTER)**

Did you press **(ENTER)**? Nothing happened, did it? Nothing that you can see, that is. What you just did is type your first program. Type:

RUN **(ENTER)**

The Computer obediently runs your program. Type RUN again and again to your heart's content. The magic machine will run your program anytime you wish, as many times as you wish.

Since that worked so well, let's add another line to the program. Type:

20 PRINT "WHAT IS YOUR NAME?"

Now type:

LIST **(ENTER)**

Your Computer obediently LISTS your entire program. Your screen should look *exactly* like this:

```
10 PRINT "HI, I'M YOUR COLOR COM  
PUTER"  
20 PRINT "WHAT IS YOUR NAME?"
```

What do you think will happen when you RUN this? Try it. Type:

RUN **(ENTER)**

The Computer prints:

```
HI, I'M YOUR COLOR COMPUTER  
WHAT IS YOUR NAME?
```

Answer the Computer's question and then press **(ENTER)** What? There's that SN Error. The Computer didn't understand what you meant when you typed your name. In fact, the Computer can't understand anything unless you talk to it in its own way.

So let's use a word the Computer understands — INPUT. Type this line:

```
30 INPUT A$ (ENTER)
```

This tells the Computer to stop and wait for you to type something, which it will label as A\$. Add one more line to the program:

```
40 PRINT "HI, " A$ (ENTER)
```

Now list the program again to see if yours looks like mine. Type:

LIST **(ENTER)**

Your program should look like this:

*If you make a mistake after pressing **(ENTER)**, simply type the line over again.*

```
10 PRINT "HI, I'M YOUR COLOR COM
PUTER"
20 PRINT "WHAT IS YOUR NAME"
30 INPUT A$
40 PRINT "HI, " A$
```

Can you guess what will happen when you RUN it? Try it:

RUN (ENTER)

That worked well, didn't it? This is probably what happened when you ran the program (depending on what you typed as your name):

```
HI, I'M YOUR COLOR COMPUTER
WHAT IS YOUR NAME?
? JANE
HI, JANE
```

RUN the program again using different names:

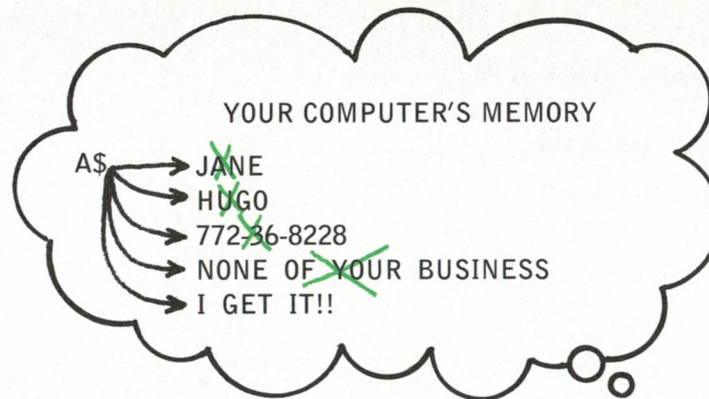
```
HI, I'M YOUR COLOR COMPUTER
WHAT IS YOUR NAME?
? HUGO
HI, HUGO
```

```
HI, I'M YOUR COLOR COMPUTER
WHAT IS YOUR NAME?
? 772-36-8228
HI, 772-36-8228
```

```
HI, I'M YOUR COLOR COMPUTER
WHAT IS YOUR NAME?
? NONE OF YOUR BUSINESS
HI, NONE OF YOUR BUSINESS
```

```
HI, I'M YOUR COLOR COMPUTER
WHAT IS YOUR NAME?
? I GET IT!!
HI, I GET IT!!
```

The Computer doesn't care what you want to call yourself. Here's what line 30 did to your Computer's memory each time you ran the program. (Assuming you gave it the same names we did):



There's an easier way to run your program over and over without having to type the RUN command. Type this line:

```
50 GOTO 10
```

Now RUN it the program runs over and over again without stopping. GOTO told the Computer to go back up to line 10:

```

10 PRINT "HI, I'M YOUR COLOR COMPUTER"
20 PRINT "WHAT IS YOUR NAME"
30 INPUT A$
40 PRINT "HI, " A$
50 GOTO 10

```

Your program will now run perpetually, because every time it hits line 50, the Computer goes up to line 10 again. We call this a "loop". The only way you can stop this endless loop is by pressing the **BREAK** key.

SPOTLIGHT YOUR NAME

Change line 50 so we can give your name the kind of attention it deserves. How do we change a program line? Simply by typing it over again, using the same line number. Type:

```
50 GOTO 40
```

To delete a program line, simply type and **ENTER** the line number. For example:

50 **ENTER**
erases line 50 from the program.

This is what the program looks like now:

```
10 PRINT "HI, I'M YOUR COLOR COMPUTER"  
20 PRINT "WHAT IS YOUR NAME"  
30 INPUT A$  
40 PRINT "HI, " A$  
50 GOTO 40
```

Type RUN and watch what this loop does. Press the **BREAK** key when you've seen enough.

There's a big change we can make simply by adding a comma or a semicolon. Try the comma first. Type line 40 again, but with a comma at the end:

```
40 PRINT A$,
```

RUN the program The comma seems to print everything in two columns.

Press **BREAK** and try the semicolon. Type:

```
40 PRINT A$;
```

and RUN You probably won't be able to tell what it's doing until you press **BREAK**. See how the semicolon crams everything together?

We're leaving out the "HI, " part this time.

Remember, if you make a mistake on one of the lines, simply type the line over again.

RULES ON PRINT PUNCTUATION

This is how punctuation at the end of a PRINT line makes the Computer PRINT:

- (1) a **COMMA** makes the Computer *PRINT* in columns.
- (2) a **SEMICOLON** makes the Computer cram the *PRINTing* together.
- (3) **NO PUNCTUATION** makes the Computer *PRINT* in rows.

COLOR/SOUND DEMONSTRATION

NEW (ENTER) . . . wish mine worked that easily!

In this program we are using T as a variable. However, we could use any letter.

Notice that Line 30 asks for T rather than T\$. This is because we want numeric data rather than string data.

Let's play around some more with your Computer's sound and color abilities. First clean out its memory. Remember how?

Now enter this program:

```
10 PRINT "TO MAKE ME CHANGE MY TONE"  
20 PRINT "TYPE IN A NUMBER FROM 1 TO 255"  
30 INPUT T  
40 SOUND T, 50  
50 GOTO 10
```

RUN through this program to get a sampling of some of the Computer's tones.



BUG: If you get a ?FC Error when you run this program, it's because you used a number other than 1 through 255. This error, like all errors, will make the Computer stop RUNning the program.

What would happen if we changed line 40 to:

```
40 SOUND 50, T
```

HINT: Look back in Chapter 1 where we talk about SOUND.

Did you figure it out? By making this change, the Computer hums the same tone every time, but hums it for a different length of time, depending on the number you type in.

Press (BREAK) first and then erase this program by typing NEW. Now see if you can write a program, similar to the one above, to make the Computer show a color you ask for. Remember, there are 9 colors, 0 through 8.

DO-IT-YOURSELF PROGRAM

*HINT: Line 40 could be:
40 CLS(T)*

This is our program:


```
10 PRINT "TO MAKE ME CHANGE MY COLOR"  
20 PRINT "TYPE A NUMBER BETWEEN 0 AND 8"  
30 INPUT T  
40 CLS(T)  
50 GOTO 10
```

ADD POLISH TO THE PROGRAM

Professional programmers would think that pressing the **BREAK** key was a rather sloppy way of getting the program to stop running. Why not get the Computer to politely ask us if we are ready to end it? Change Line 50 in the above program to:

```
50 PRINT "DO YOU WANT TO SEE ANOTHER COLOR"
```

Press **BREAK** before typing the line.



and add these lines:

```
60 INPUT R$
70 IF R$ = "YES" THEN 20
```

and RUN the program . . . Type YES and the program will keep on running. Type anything else and the program will stop.

This is what the program looks like:

```
10 PRINT "TO MAKE ME CHANGE COLORS"
20 PRINT "TYPE A NUMBER BETWEEN 0 AND 8"
30 INPUT T
40 CLS(T)
50 PRINT "DO YOU WANT TO SEE ANOTHER COLOR"
60 INPUT R$
70 IF R$ = "YES" THEN 20
```

R\$ = yes

Let's look at what these new lines did:


Line 50 simply printed a question.

Line 60 told the Computer to stop and wait for our answer -- R\$.

Line 70 told the Computer to go back to line 20 *IF* (and only *IF*) your answer (R\$) was *YES*. If not, the program simply ended since there are no more lines in the program.

You've covered a lot of ground in this chapter. Hope we're just whetting your appetite for more to come.

Don't worry if you don't understand everything perfectly yet. Just enjoy using your Computer.



Don't worry about this IF/THEN right now. We'll be devoting a whole chapter to it later.

LEARNED IN CHAPTER 3

BASIC WORDS

Characters

NEW
INPUT
GOTO
RUN
PRINT,
PRINT;
LIST
IF/THEN

CONCEPTS

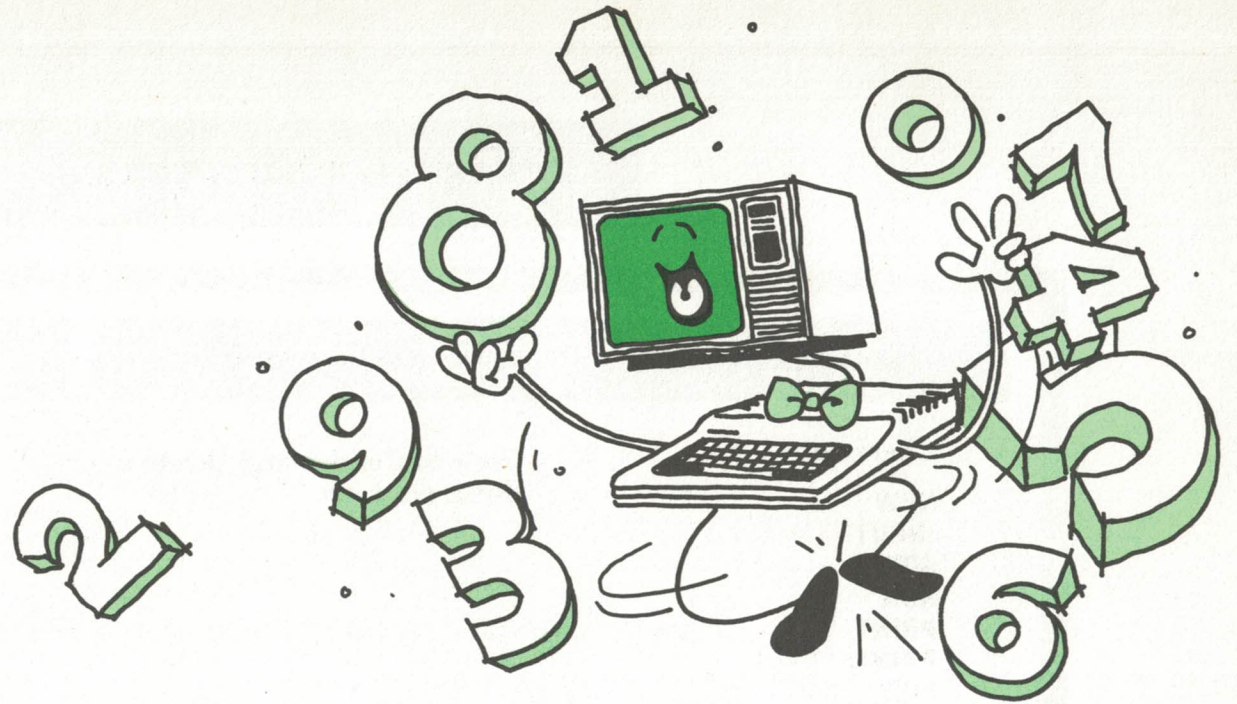
How to Change and Delete a
Program Line

KEYBOARD

BREAK

NOTES:

CHAPTER 4



COUNT THE BEAT



COUNT THE BEAT

In this Chapter we are going to do some experimenting with Computer sound effects. To do this, we have to first teach the Computer how to count.

Type this:

```
10 FOR X = 1 TO 10
20 PRINT "X = " X
30 NEXT X
40 PRINT "I HAVE FINISHED COUNTING"
```

RUN the program.

RUN the program several times, each time replacing line 10 with one of these lines:

```
10 FOR X = 1 TO 100
10 FOR X = 5 TO 15
10 FOR X = -2 TO 2
10 FOR X = 20 TO 24
```

Do you see what FOR and NEXT are making the Computer do? They are making it count. Let's study the last program we suggested you try:

The logic of this will become clear later.

*Remember to type
NEW **ENTER**
before typing a new program.*

```

10 FOR X = 20 TO 24
20 PRINT "X = " X
30 NEXT X
40 PRINT "I HAVE FINISHED COUNTING"

```

Line 10 tells the Computer that the first number should be 20 and the last number should be 24. It uses X to label these numbers.

Line 30 tells the Computer to keep going back up to line 10 for the next Number—the NEXT X—until it reaches the last number (24).

Look at line 20. Since line 20 is between the FOR and NEXT lines, the Computer must PRINT the value of X every time it counts:

```

X = 20
X = 21
X = 22
X = 23
X = 24

```

Add another line between FOR and NEXT:

```

15 PRINT "... COUNTING ..."

```

and RUN it. With every count, your Computer executes any lines you choose to insert between FOR and NEXT.

Write a program which will make the Computer print your name 10 times.

DO-IT-YOURSELF PROGRAM 4/A



HINT: The program must count to 10.

Write a program which will print the multiplication tables for 9 (9*1 through 9*10).

DO-IT-YOURSELF PROGRAM 4/B

*HINT: PRINT 9*X is a perfectly legitimate program line.*

Write a program which will print the multiplication tables for 9*1 through 9*25.

DO-IT-YOURSELF PROGRAM 4/C

HINT: By adding a comma in the PRINT line, you can get all the problems and results on your screen at once.

Finished? These are our programs:

```
Program 4/A
10 FOR X = 1 TO 10
20 PRINT "THOMAS"
30 NEXT X
```

```
Program 4/B
10 FOR X = 1 TO 10
20 PRINT "9**X"="9*X
30 NEXT X
```

```
Program 4/C
10 FOR X = 1 TO 25
20 PRINT "9**X"="9*X,
30 NEXT X
```

COUNTING BY TWOS

Now we'll make it count a little differently. Erase your program by typing NEW and then type our original program, using a new line 10:

```
10 FOR X = 2 TO 10 STEP 2
20 PRINT "X= " X
30 NEXT X
40 PRINT "I HAVE FINISHED COUNTING"
```

RUN the program . . . Do you see what the STEP 2 did? It makes the Computer count by 2's. Line 10 tells the Computer that:

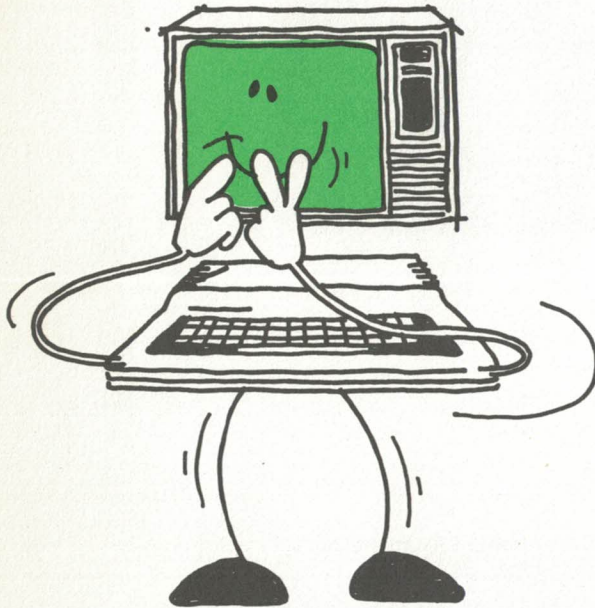
- the first X is 2
 - the last X is 10
- .. AND STEP 2 . . .
- all the Xs between 2 and 10 are 2 apart. . . that is 2, 4, 6, 8, and 10. (STEP 2 tells the Computer to add 2 to get each NEXT X.)

To make the Computer count by 3's, make all the Xs 3 apart. Try this for line 10:

```
10 FOR X = 3 TO 10 STEP 3
```

RUN the program. It should print this on your screen:

```
X = 3
X = 6
X = 9
```



"2, 4, 6, 8,..."

It passed up the last X (10) because $9 + 3 = 12$. Try a few more FOR . . . STEP lines so you can see more clearly how this works:

```
10 FOR X = 5 TO 50 STEP 5
10 FOR X = 10 TO 1 STEP -1
10 FOR X = 1 TO 20 STEP 4
```

COUNTING THE SOUNDS

Now that you've taught the Computer to count, you can add some sound. Erase your old program and type this:

```
10 FOR X = 1 TO 255
20 PRINT "TONE " X
30 SOUND X, 1
40 NEXT X
```

This program is making the Computer count from 1 to 255 (by ones). Each time it counts it does what lines 20 and 30 tell it to do:

- It PRINTs X, the current count (Line 20)
- It SOUNDS X's particular tone (Line 30)

For example:

- the first time the Computer got to FOR, in line 10, it made X equal to 1.
- then it went to line 20 and printed 1, the value of X.
- then, line 30 had it SOUND tone #1.
- then it went back up to line 10 and made X equal to 2
- etc.

What do you think the Computer will do if you make this change to line 10:

```
10 FOR X = 255 TO 1 STEP -1
```

Did you try it? Using STEP, change line 10 so the Computer will sound tones from:

You might be wondering about the programs we ran at the first of this Chapter where we didn't use STEP. If we leave out STEP, the Computer assumes we mean STEP 1.

Don't type the arrow of course. That's there to help you understand.

- (1) the bottom of its range to the top, humming every tenth note.
- (2) the top of its range to the bottom, humming every tenth note.
- (3) the middle of its range to the top, humming every fifth note.

PROGRAMMING EXERCISE

10 _____
10 _____
10 _____

*Try this: To pause the program while it is running press the **(SHIFT)** and **@** keys at the same time. Then press any key to continue.*

Ready for the answers?

```
10 FOR X = 1 TO 255 STEP 10  
10 FOR X = 255 TO 1 STEP -10  
10 FOR X = 128 TO 255 STEP 5
```

Now see if you can write a program which makes the Computer hum:

- (1) from the bottom of its range to the top, and then
- (2) from the top of its range back to the bottom

DO-IT-YOURSELF PROGRAM

BUT CAN IT SING?

Yes. Although your Computer is slightly off pitch, it can warble out most songs. The next chapter will show you how to teach it some of your favorite songs.



LEARNED IN CHAPTER 4

BASIC WORDS

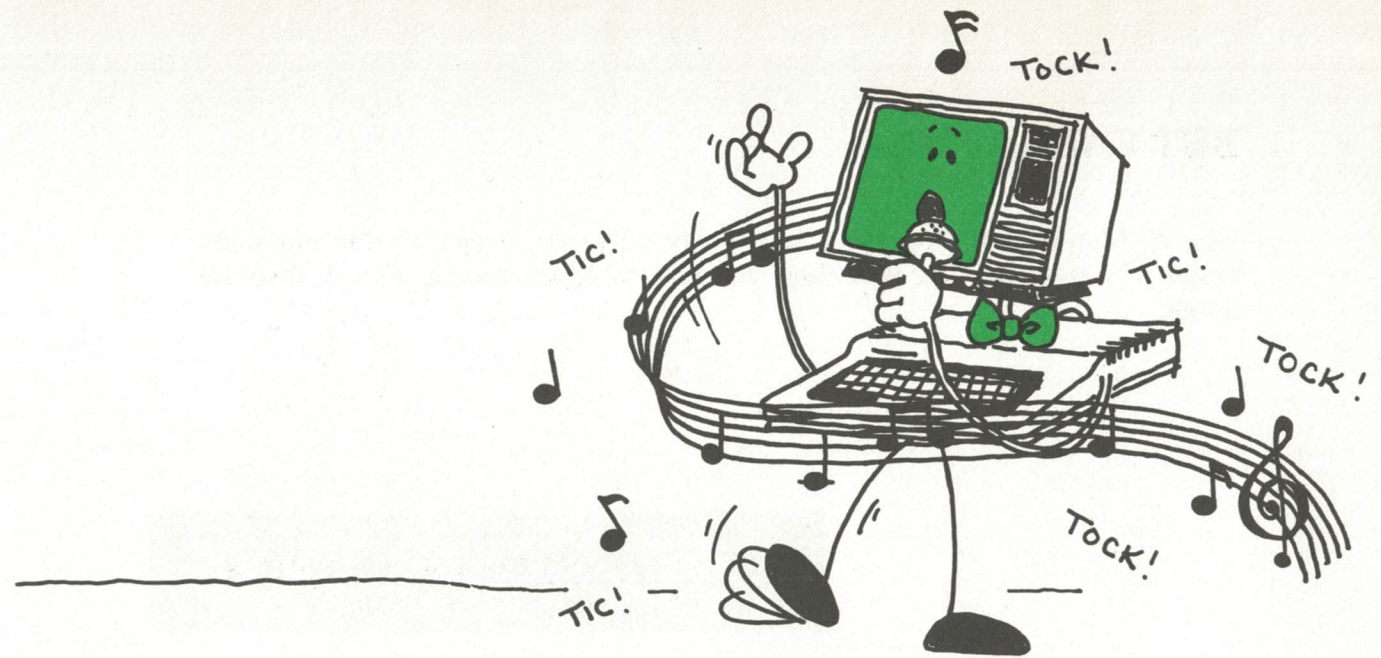
FOR ... TO ... STEP
NEXT

KEYBOARD CHARACTER

SHIFT @

NOTES:

CHAPTER 5



SING OUT THE TIME



SING OUT THE TIME

You're now ready to show your Computer how to do two things: tell time and sing (. . . well, as good as the Computer can sing. . .). Since they are actually closely related — especially to your Computer! — we're covering them both in the same Chapter.

Begin by typing this:

```
10 FOR Z = 1 TO 460 * 2
20 NEXT Z
30 PRINT "I COUNTED TO 920"
```

RUN the program. Be patient and wait a couple of seconds. Two seconds, to be precise. It takes your computer 2 seconds to count to 920.

Lines 10 and 20 set a *timer pause* in your program. By making the Computer count to 920, it keeps the Computer busy for 2 seconds.

As you can see, this gives us the makings of a stopwatch. Erase the program, and type this:

```
10 PRINT "HOW MANY SECONDS"
20 INPUT S
30 FOR Z = 1 TO 460*S
40 NEXT Z
50 PRINT S " SECONDS ARE UP!!!"
```

RUN it, inputting the number of seconds you want timed on your stopwatch.
It would be nice if the stopwatch could sound some kind of alarm. Add some lines to the end of the program to make it sound an alarm.

DO IT YOURSELF PROGRAM

Here's the program we wrote:

```
10 PRINT "HOW MANY SECONDS"  
20 INPUT S
```

```
30 FOR Z = 1 TO 460 * S  
40 NEXT Z
```

```
50 PRINT S " SECONDS ARE UP!!!"
```

```
60 FOR T = 120 TO 180  
70 SOUND T, 1  
80 NEXT T
```

```
90 FOR T = 150 TO 140 STEP -1  
100 SOUND T, 1  
110 NEXT T
```

```
120 GOTO 50
```

This is how computerized timers work.

Notice the GOTO line we added at the end of the program. This is so the message would print and the alarm would keep ringing over and over again until the nervous programmer must press the **BREAK** or **SHIFT @** keys to turn it off.

COUNTING WITHIN THE TIME

Before we go any further on the clock, we're going to have the Computer keep count *within* the time. This concept will become very clear to you shortly.

Type this new program:

```
10 FOR X = 1 TO 3
20 PRINT "X = " X
30 FOR Y = 1 TO 2
40 PRINT, "Y = " Y
50 NEXT Y
60 NEXT X
```

RUN it . . . This should be on your screen:

```
X = 1
      Y = 1
      Y = 2
X = 2
      Y = 1
      Y = 2
X = 3
      Y = 1
      Y = 2
```

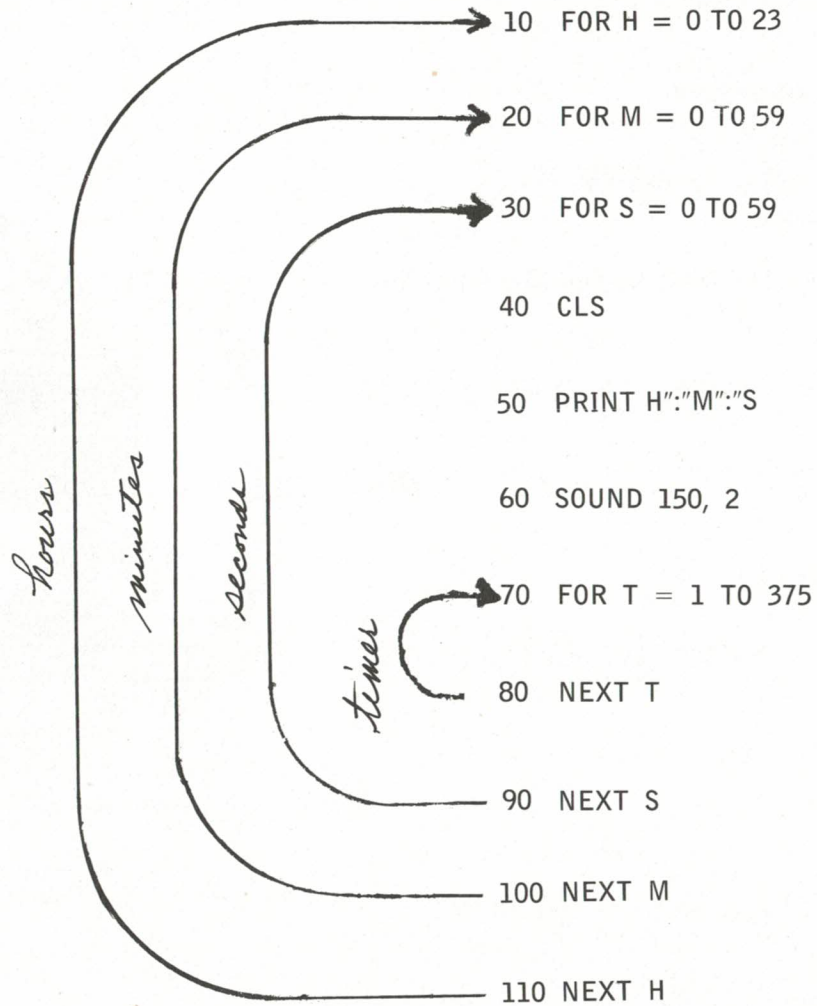
Call it a count within a count or a loop within a loop — whatever you prefer. Programmers call this a “nested loop”. This is what the program does:

- I. It counts X from 1 to 3. *Every time* it counts X, it does these things:
 - A. It *PRINTs* the value of X

Notice the comma in line 40. Try it without the comma. The comma makes "Y = " Y PRINT on the next column.



With this groundwork, it is easy to make a full fledged clock:



Here's an outline of what the Computer does in this program:

I. It counts the hours from 0 to 23. (Line 10)

Every time it counts a new hour:

A. It counts the minutes from 0 to 59. (Line 20)

Every time it counts a new minute:

1. It counts the seconds from 0 to 59. (Lines 30 and 90)

Every time it counts a new second:

a. It *CLear*s the Screen. (Line 40)

b. It *PRINT*s the hour, minute, and second. (Line 50)

c. It *SOUND*s a tone. (Line 60)

d. It pauses long enough for one second to pass. (Lines 70 and 80)

2. When it finishes counting all the 59 seconds,
it goes back up to line 20 for the next minute. (Line 100)

B. When it finishes counting all the 59 minutes,
it goes back up to line 10 for the next hour. (Line 110)

II. When it finishes counting all hours (0-23), the program ends.

Between lines 90 and 100 you can add some tones which will sound every minute. Write a program which does this.

*By adding this line:
120 GOTO 10
the clock will run perpetually.*

Having a tough time with this program? Skip it for now. It'll seem easy later.

DO-IT-YOURSELF PROGRAM

Write a program which makes your Computer show each of its nine colors for 1 second each:

DO-IT-YOURSELF PROGRAM

The answers to both of these programs are in the back.

But who said this Computer could make the Opera?

If you're a real music lover, you will probably want to purchase RADIO SHACK's "MUSIC" — Catalog number 26-3151. Then you will be able to compose songs on your Computer with perfect pitch.

FOR A COMPUTER, IT SINGS GREAT!

Now back to teaching your Computer how to sing. Flip back to the Appendix. We have a table, "Musical Tones", which shows the Computer's tone number for each note on the musical keyboard. For example, the Computer's tone number 89 corresponds to "middle C".

Unfortunately, your Computer can't exactly match most of the musical tones. That's why the Computer sings a little off key. . . But to those without perfect pitch, it can still sound very close to music.

Type this:

```
20 SOUND 125, 8
30 SOUND 108, 8
40 SOUND 89, 8
```

RUN the program. It is the first three notes of . . .well you know that, great piece!

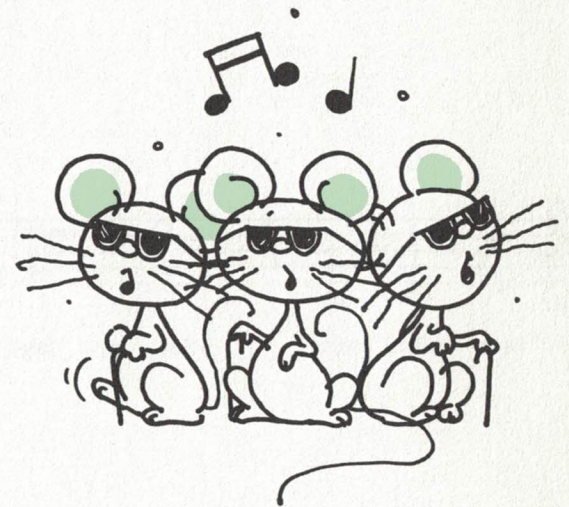
To get these first three notes to play over again, we can put a FOR/NEXT loop in the program:

```
10 FOR X = 1 TO 2
20 SOUND 125, 8
30 SOUND 108, 8
40 SOUND 89, 8
50 NEXT X
```

Now RUN the program again. It's missing a pause, isn't it? It's easy enough to put a *timer pause* in the program. Add these lines:

```
44 FOR Y = 1 TO 230
46 NEXT Y
```

and RUN it again. Now it's beginning to sound like the real thing!



Here is a program that gets through the first two phrases:



Three blind mice See how they run

THREE BLIND MICE

```

10  FOR X = 1 TO 2
20  SOUND 125, 8      "Three"
30  SOUND 108, 8     "blind"
40  SOUND 89, 8      "mice"

    repeat
    Pause
44  FOR Y = 1 TO 230  (pause)
46  NEXT Y

50  NEXT X

60  FOR X = 1 TO 2
70  SOUND 147, 8     "See"
80  SOUND 133, 4     "how"
90  SOUND 133, 4     "they"
100 SOUND 125, 8     "run"

    repeat
    Pause
110 FOR Y = 1 TO 230 (pause)
120 NEXT Y

130 NEXT X
    
```

Are your programs getting too long to list? Try this

LIST 10-48 (ENTER)

Only the first half of this program will be listed.

Finish the song, if you like. Or write a better one. Your Computer songs can certainly jazz up any program.

LEARNED IN CHAPTER 5

BASIC WORD

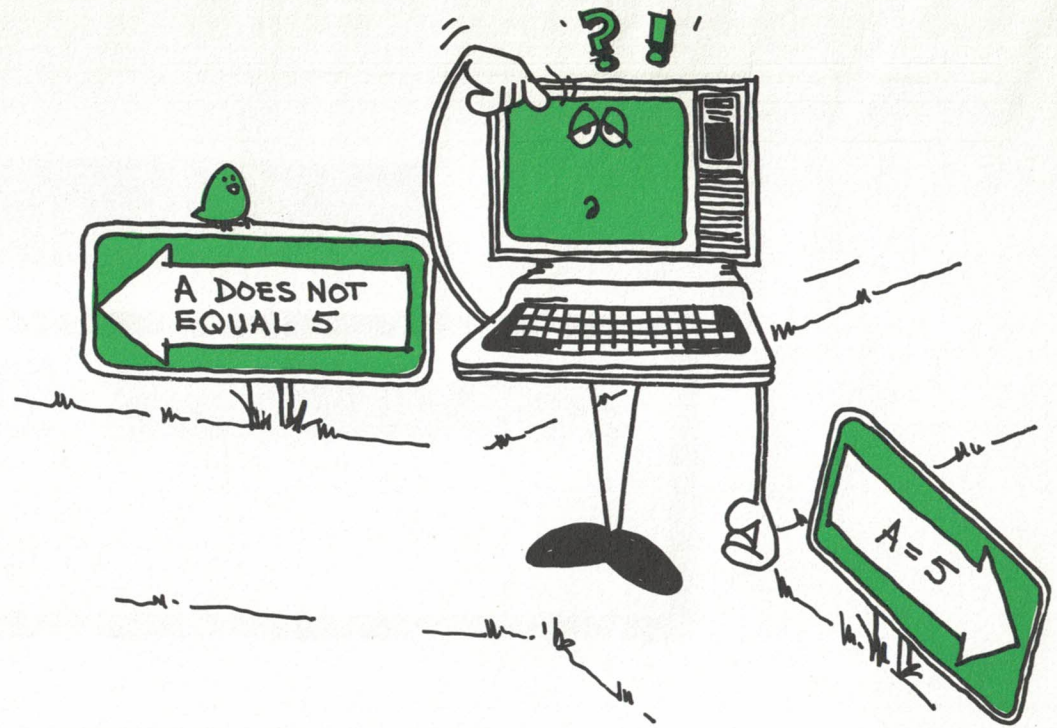
CLS

PROGRAMMING CONCEPT

Nested Loops

NOTES:

CHAPTER 6



DECISIONS, DECISIONS...



DECISIONS, DECISIONS. . .

Here's an easy decision for the Computer:

- (1) IF you type RED . . . *THEN* make the screen red
 . . . or
- (2) IF you type BLUE . . . *THEN* make the screen blue

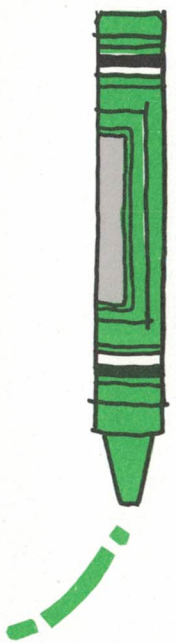
Easy enough? Let's make the Computer do it. Type this program:

```
10 PRINT "DO YOU WANT THE SCREEN RED OR BLUE?"
20 INPUT C$
30 IF C$ = "RED" THEN 100
40 IF C$ = "BLUE" THEN 200
100 CLS(4)
110 END
200 CLS(3)
```

Don't be confused by the arrows or the spaces between program lines. We just put them in to illustrate the flow of the program.

RUN the program several times, typing both RED and BLUE.

Let's see what the program is doing:



IF you type RED ... THEN ...

Line 30 sends your program down to line 100. Line 100 makes your screen red. At this point, we have to stop the Computer from going on to line 200.

Line 110 does just that. It ends your program right there... Once the Computer gets to line 110, it will never make it to 200.

... On the other hand ...

IF you type BLUE ... THEN ...

Line 40 sends your Computer down to line 200, which makes your screen blue. We do not have to put END on the next line. Since line 200 is the last line in the program, the Computer will end there anyway.

What happens if you type something other than RED or BLUE? Try running the program, typing GREEN in response to the Computer's question.

It makes the screen RED, right? Do you know why?

HINT: IF the condition is not true, the THEN part of the line is ignored and the Computer proceeds to the next program line.

There are two lines you could add to make the Computer ask you to type your answer again if you don't type RED or BLUE. We will give you the two lines, and let you figure out where to put them in the program:

PROGRAMMING EXERCISE

```
.... PRINT "YOU MUST TYPE EITHER RED OR BLUE"  
.... GOTO 20
```

insert the line numbers

HINT: The lines must come AFTER the Computer has had a chance to test your answer for RED or BLUE.

HINT: The lines must come BEFORE the Computer makes your screen RED.

Did you figure out where the two lines should go in the program? They must come after line 40 and before line 100:

```
50 PRINT "YOU MUST TYPE EITHER RED OR BLUE"  
60 GOTO 20
```

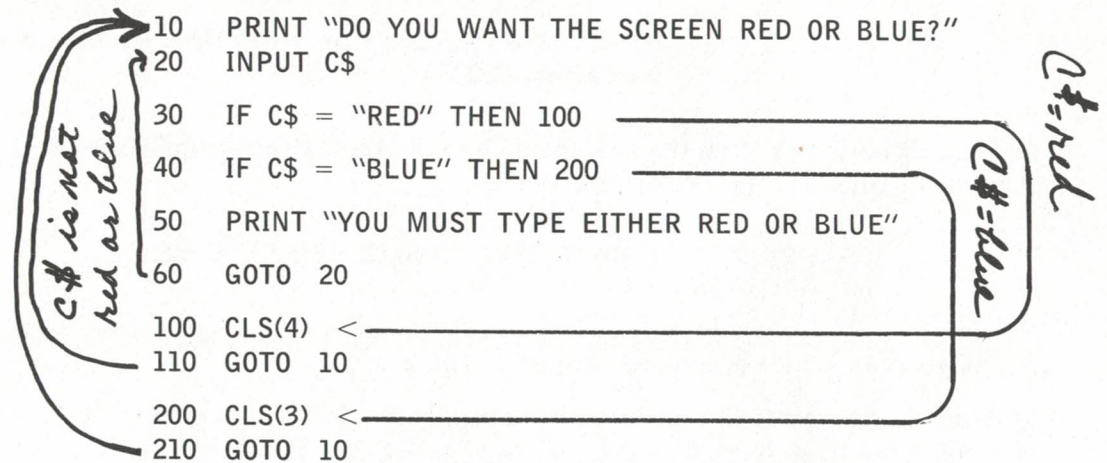
See if you can make one more change to the program:

Instead of having the Computer end the program after it makes the screen red or blue, have it go back and ask you to type RED or BLUE again.

DO IT YOURSELF PROGRAM

HINT: You will need to change line 110 and add line 210.

Have you got a program written? Look on the next page for a diagram of ours.



To trace the path the Computer takes down this program, simply go down, from one line to the next, following the arrows when told to. Notice the difference between the arrows going from the IF/THEN and the GOTO lines:

RULES ON IF/THEN AND GOTO

IF/THEN is conditional.

You only follow these arrows if the condition (C\$ = "RED" or C\$ = "BLUE") is true.

GOTO is unconditional.

You follow these arrows whenever you arrive at a GOTO line.

Although this chapter was short, you've learned one of the most important programming concepts. We will be getting the Computer to make decisions all through the rest of this book.

LEARNED IN CHAPTER 6

BASIC WORDS

IF/THEN
END

NOTES:

CHAPTER 7



GAMES OF CHANCE



GAMES OF CHANCE

Thanks to a BASIC word called RND, your Computer can play almost any kind of game involving chance or luck. Even if you don't plan to play games with your Computer, you'll want to know how to use RND and PRINT @ — the words we're introducing in this Chapter. We'll also show you some more uses for IF/THEN.

Type this:

```
10 PRINT RND(10)
```

RUN it. The Computer just picked a random number from 1 to 10. RUN it some more times. . .

It's as if the Computer is drawing a number from 1 to 10 out of a hat. The number the Computer picks is unpredictable. Type and RUN this program. Press **(BREAK)** when you satisfy yourself that the Computer is printing random numbers:

```
10 PRINT RND(10);  
20 GOTO 10
```

To have the Computer pick random numbers from 1 to 100, change line 10 to this:

```
10 PRINT RND(100);
```

*To make the Computer pause while running the program, press the **(SHIFT)** and @ keys at the same time. Press any key and the Computer will continue.*

and RUN. How would you change this program to have the Computer pick a random number from 1 to 255?

.....

The answer is:

```
10 PRINT RND(255);
```

A COMPLETELY RANDOM SHOW

Just for the fun of it, let's have the Computer compose a song made up of random tones. Type this:

```
10 T = RND(255)
20 SOUND T, 1
30 GOTO 10
```

RUN it. Great music, eh? Press **BREAK** when you've heard enough.


To add a random visual presentation to this program, add a couple of lines to make the Computer show a random color (1-8) just before it sounds each random tone.



DO IT YOURSELF PROGRAM

Here's our program:

```
10 T = RND(255)
14 C = RND(8)
16 CLS(C)
20 SOUND T, 1
30 GOTO 10
```




We'll show you a couple of simple games in this Chapter. Feel free to use your imagination to add interest to them — or invent your own games.

RUSSIAN ROULETTE

In our "Russian Roulette" game, the gun has 10 chambers. The Computer picks, at random, which of the 10 chambers will have the fatal bullet. Type:

```
10 PRINT "CHOOSE YOUR CHAMBER(1-10)"
20 INPUT X
30 IF X = RND(10) THEN 100
40 SOUND 200, 1
50 PRINT "--CLICK--"
60 GOTO 10

100 PRINT "BANG — YOU'RE DEAD"
```



First, in line 20, the player INPUTs X — a number from 1 to 10. Then the Computer compares X with RND(10) — a random number from 1 to 10.

Now look at the arrows we drew:

IF X is equal to RND(10), THEN the Computer goes down to 100.

IF X is not equal to RND(10), THEN the Computer "clicks" and goes back up to line 10 where you get another chance. . .

Let's make the dead routine in line 100 better. Type:


*Remember to always type:
NEW (ENTER)
before typing a new program.*

Remember how to list a portion of a program?

LIST 50-130

lists the middle portion of the program.

This will also help you in listing a long program. Press the **SHIFT** and **@** keys when the Computer first starts listing the program. The Computer will pause the "scrolling" on your display. Press any key to continue the listing.



```
100 FOR T = 133 TO 1 STEP -5
110 PRINT "          BANG!!!!!"
120 SOUND T, 1
130 NEXT T
140 CLS
150 PRINT @ 230, "SORRY, YOU'RE DEAD"
160 SOUND 1, 50
170 PRINT @ 390, "NEXT VICTIM, PLEASE"
```

RUN the program. Here's what happens in this program:

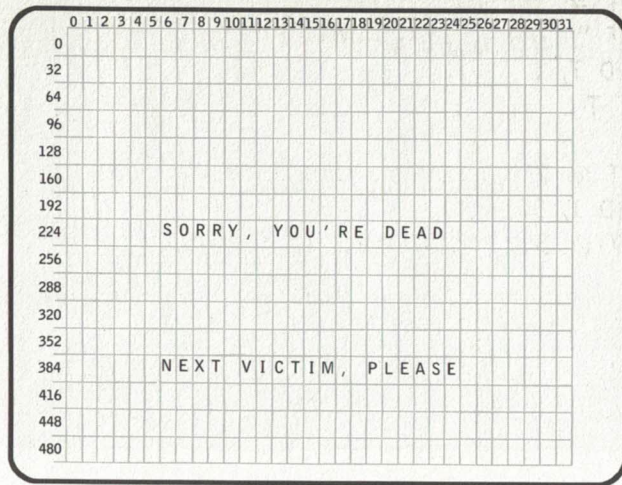
Lines 100 through 130 makes the Computer produce a sound of descending tones and print BANG!!!! over and over again on the screen.

Line 140 CLears the Screen. Since we did not choose a color number code, the Computer assumes we want the screen green.

Look at lines 150 and 170. Both of these lines use PRINT @. Here's the way PRINT @ works:

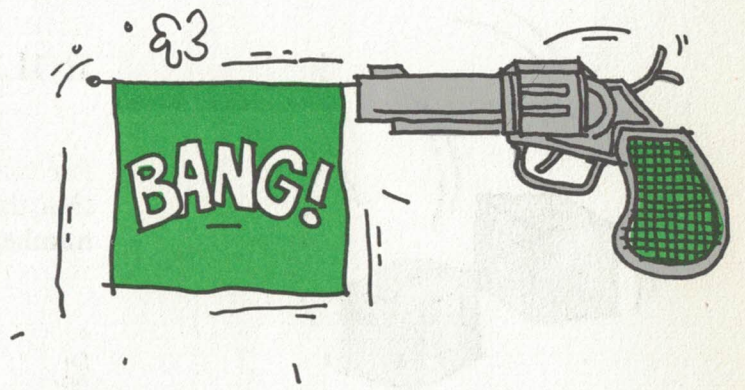
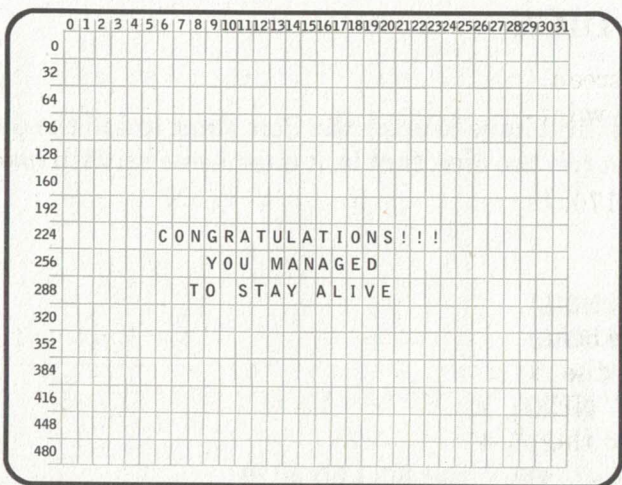
Notice the grid we have below, showing each of the 511 positions on your video screen. When writing the program, we wrote the two messages "SORRY, YOU'RE DEAD" and "NEXT VICTIM PLEASE" on this grid, positioning them where we wanted them on the screen.

SORRY, YOU'RE DEAD begins at location 230 (224 + 6). NEXT VICTIM PLEASE begins at location 390 (384 + 6). Using these numbers in the PRINT @ line, simply tells the Computer where we want the message printed.



We put this grid in the Appendix of this book "PRINT @ Screen Locations". Use it in planning your programs, since good screen formatting can add a great deal of interest to your programs.

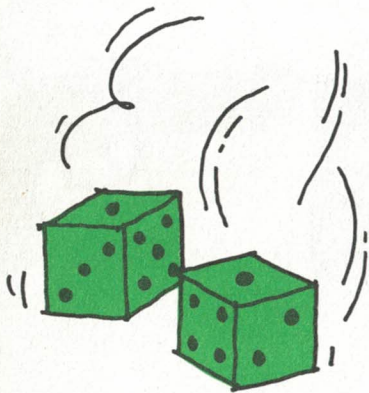
Change this program, so that if the player *DOES* manage to stay alive for 10 clicks, the Computer pronounces the player the winner, printing this message on the screen:



DO IT YOURSELF PROGRAM

HINT: You can use the FOR/NEXT loop, so that the Computer can keep count of the number of clicks.

Our answer is to this is in the book.



"Loser!"

ROLLING THE DICE

For our next game, we'll first have to teach the Computer to roll the dice. To do this, the Computer must roll *two* dice; that is, it must come up with *two* random numbers. Type:

```
A# = "YES"
10 CLS
20 X = RND(6)
30 Y = RND(6)
40 R = X + Y
50 PRINT @ 200, X
60 PRINT @ 214, Y
70 PRINT @ 394, "YOU ROLLED A" R
80 PRINT @ 454, "DO YOU WANT ANOTHER ROLL?"
90 INPUT A$
100 IF A$ = "YES" THEN 10
```

RUN the program. Let's look at it:

Line 10 tells the Computer to CLear the Screen.

Line 20 has the Computer pick a random number from 1 to 6 for one of the die.

Line 30 has the Computer pick a random number for the other die.

Line 40 simply adds the two dice to get the total roll.

Lines 50-70 PRINT the results of the roll on the screen.

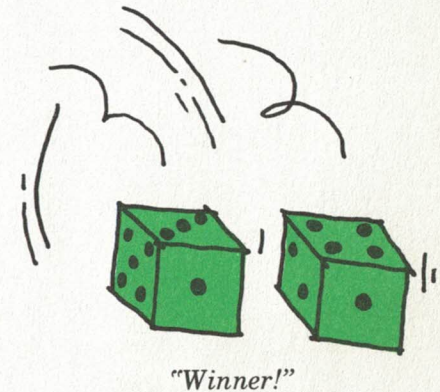
In line 90, you are able to INPUT whether you want the program to RUN again. IF you type YES, the Computer goes back to line 10 and runs the program again. Otherwise, since this is the last line in the program, the program ends.

CRAPS

Now that you know how to get the Computer to roll the dice, it should be fairly easy for you to write a Craps program. These are the rules of the game (in its simplest form):

1. The player rolls the two dice. If he rolls a sum of 2 ("snake eyes"), a 3 ("cock-eyes"), or a 12 ("boxcars") on the first roll, the player loses and the game is over.
2. If the player rolls a 7 or 11 on the first throw, ("a natural"), the player wins and the game is over.
3. If any other number is rolled on the first roll, it becomes the player's "point". He must keep rolling until he either "makes his point" by getting the same number again to win, or rolls a 7, and loses.

You already know more than enough to write this program. Do it. Make the Computer print it in an attractive format on your screen and keep the player informed on what is happening. It may take you awhile to finish, but give it your best. *Good luck!*



DO-IT-YOURSELF PROGRAM

Our answer to this is in the back.

LEARNED IN CHAPTER 7

BASIC WORD

**RND
PRINT @**

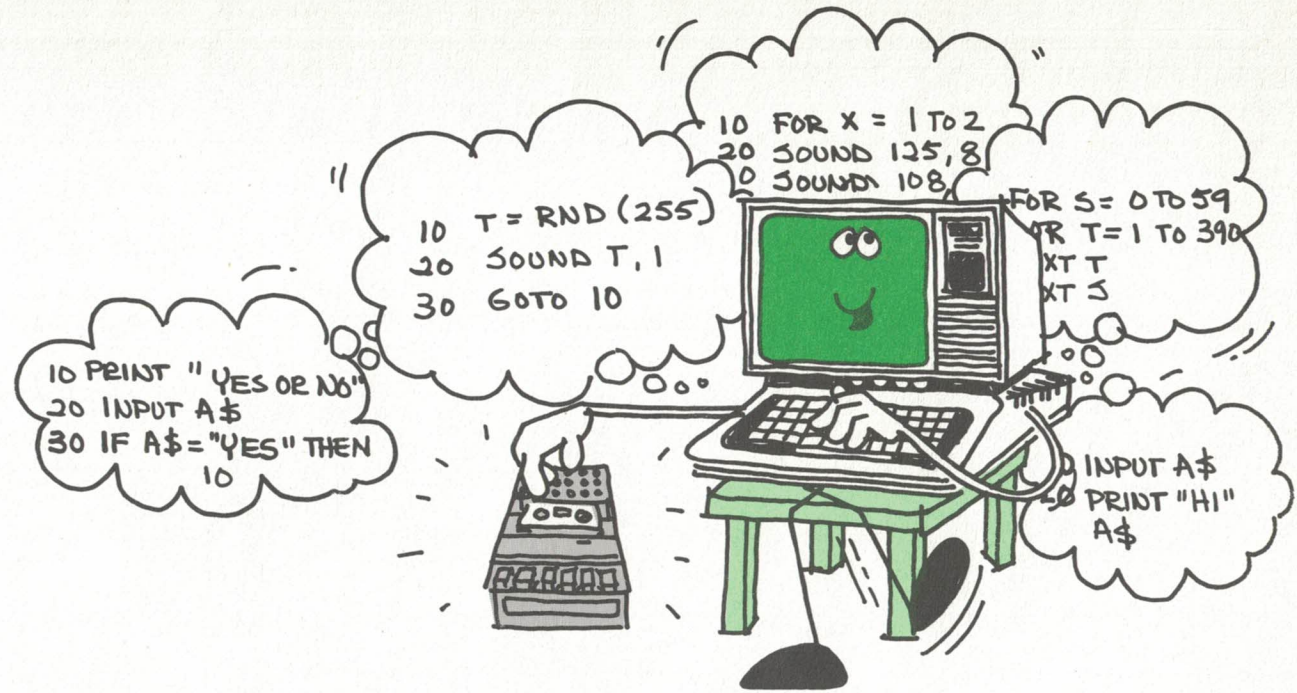
NOTES:

Blank lined area for notes on the left side of the page.

Blank lined area for notes on the right side of the page.



CHAPTER 8



SAVE IT ON TAPE



SAVE IT ON TAPE

You'll soon be writing longer and more powerful programs. Perhaps you already are. It certainly cramps your style to have the program disappear everytime you turn the Computer off!

You can "save" (make a copy of) any of your programs on cassette tape. Once the program's on tape, you'll be able to "load" the program back into your Computer's memory anytime you want. We recommend that you use Radio Shack's CTR-80A cassette recorder (catalog number 26-1206) along with Radio Shack's Computer Tapes (catalog number 26-301).

This chapter is only for those of you that have this type of cassette recorder and want to use it. If you don't, you'll probably want to skip this chapter for now, remembering that the information's here whenever you need it.

Once you're used to it, you'll find cassette tape easy to use. Simply follow these steps:

A. Connect the Tape Recorder

1. Locate the CTR-80A Cassette Recorder, Interconnecting Cable and Radio Shack Computer Recording Tape cassette.
2. Connect the short cable between the TAPE jack on the back of the TRS-80 and your Cassette Tape Recorder

If you are using a different type of cassette recorder, the connections might be different from the explanation in this chapter.

If you are using a tape other than Radio Shack's, you need to position it after the plastic "leader" at the beginning of the tape.

- The small grey plug goes into the REM jack on the Recorder.
- The large grey plug goes into the AUX jack.
- The black plug goes into the EAR jack.

3. Plug the Recorder into the wall outlet

B. Save a Program

1. Type any program into your Computer. RUN it to make sure it works.
2. Load the cassette tape, positioning it to the beginning of the tape. Press the PLAY and RECORD buttons *at the same time* until they lock.
3. Name the program you want to SAVE. You may use any name with 8 or fewer letters. For our example, we'll use "NAME".
4. SAVE on tape by typing this command:

You may substitute any name for NAME.

CSAVE "NAME" **ENTER**

The motor on the Recorder will start and you'll be recording the Computer's program on tape. Watch the screen. When:

OK

returns and the motor stops, your program is recorded on tape. It is also still in the Computer's memory. It has only been copied.

LOADING

Reversing the process and loading (copying) the program from tape into the Computer is just as easy:

1. Be sure the tape is fully rewound and the plugs are all in place.

2. Push the PLAY button down until it locks. Set the Volume Control to your CTR-80A's "Recommended Volume Level". Your CTR-80A Manual gives this recommended volume.
3. Type NEW to clear out any existing program.
4. Type the CLOAD command with the name of your program. For example:

CLOAD "NAME" **ENTER**

The Tape Recorder's motor will start. Watch your screen. The letter:

S

will appear at the top left hand corner. This means the Computer is Searching for your program. When the Computer has Found your program, it will print the letter F and the name of your program. For example, if your program name is NAME:

F NAME

will appear at the top of your screen. When the Computer prints:

OK

and the recorder motor stops, the program is "loaded" in memory. You may now RUN the program.

SAVING MORE THAN ONE PROGRAM

To SAVE more than one program on the same tape, you must make sure you are not recording on top of another program. This is an easy way to position the tape to the end of your last program:

1. Rewind the tape to the beginning.
2. Press the PLAY button until it locks

If you have several programs on tape, the Computer will print the name of each program it Finds on the tape preceding the one you want loaded.

If you try to load a program that's not on the tape, the Computer will not stop searching for it. Press the RESET button to stop searching.

-
3. Type **SKIPF** and the name of the last program on your tape. For example, if your last program is named "NAME", type:

SKIPF "NAME"

The Computer will notify you when it finds your program called NAME. When it reaches the end of NAME, the recorder's motor will stop and:

OK

will appear on your screen.

4. Once you've positioned the tape to the end of the last program, press the **RECORD** and **PLAY** buttons, name your program, and **CSAVE** it.

You may replace the name X with any name you know is NOT on the tape.

If you can't remember the name of your last program, type:

SKIPF "X"

and watch the screen. The Computer will give you the name of each program it encounters on the tape. It will print an **I/O ERROR** when it reaches the end of the tape, but don't worry about it. You've found what you were looking for — the name of the last program on the tape.

Now you can type the **SKIPF** command with the name of this last program. (Don't forget to rewind the tape first).

TIPS ON MAKING GOOD RECORDINGS

Here are some tips for making good recordings:

- When you're not using the Recorder for saving or loading, do not leave the **RECORD** or **PLAY** keys down. Press **STOP**.
- Don't attempt to re-record on a pre-recorded Computer tape. Even though the recording process erases the old recording, just enough information may

be left to confuse the new recording. If you want to use the same tape a second or third time, use a high-quality bulk tape eraser to be sure you erase everything.

- If you want to save a taped program permanently, break off the Erase Protect tab on the Cassette (see Tape Recorder's Manual). When the tab(s) has been broken off, you can't press the RECORD key on your Recorder. This will keep you from accidentally erasing that tape.

Now type as long of programs as you want, knowing you can make a permanent copy of them on tape. Happy recording!

LEARNED IN CHAPTER 8

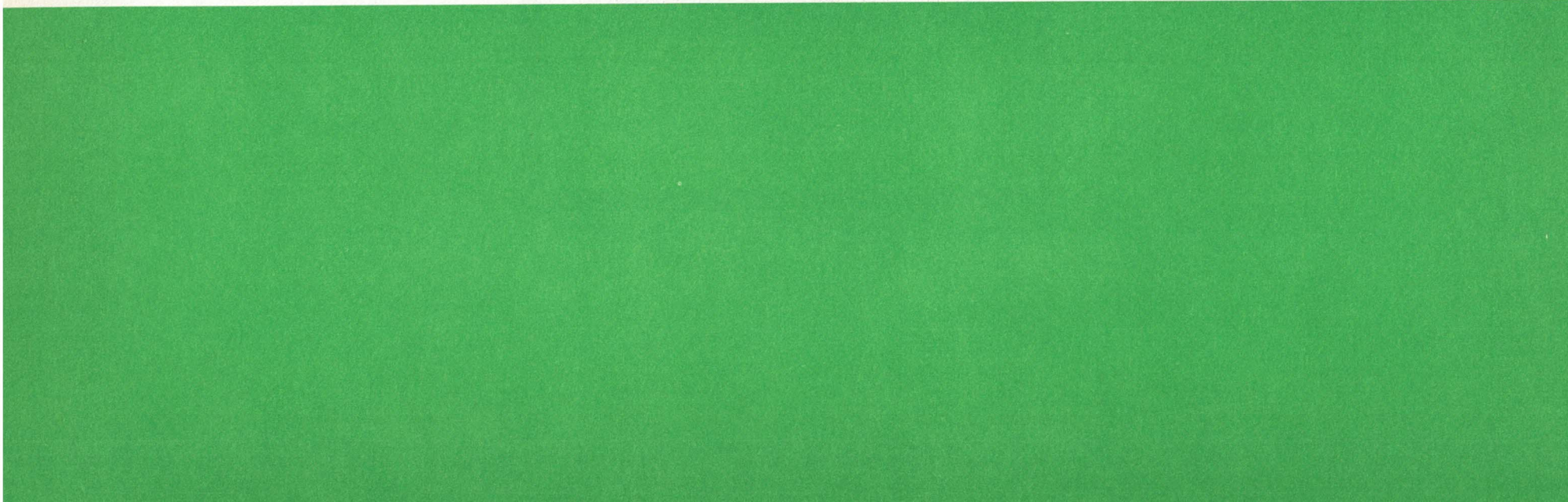
BASIC WORDS

CLOAD
CSAVE
SKIPF

CHAPTER 9



COLOR THE SCREEN





COLOR THE SCREEN

You've learned enough now to really start using the colors. Since color graphics ideas usually come very quickly to people — and the good graphics programs usually end up long — this Chapter just shows you how to get started. While going through this Chapter, you'll probably want to stop from time to time and add on to our programs or build your own. We hope you do. That's a fast way to learn.

To get started, type:

```
10 CLS(0)
```

to make the screen black. Add these two lines and RUN the program:

```
20 SET(0,0,3)  
30 GOTO 30
```

Do you see the blue dot? It's at the top left-hand corner of your screen. To put the dot at the bottom right-hand corner, change line 20 and RUN the program:

```
20 SET(63,31,3)
```

Want to put it in the middle of the screen? RUN the program using this for line 20:

Be sure to type line 30. We'll explain why later.



```
20 SET(31,14,3)
```

SET tells the Computer to SET a dot on your screen at a certain horizontal and vertical location.

- The first number you type is the horizontal location. This may be a number from 0 to 63.
- The second number is the vertical location. It may be a number between 0 and 31.

In the Appendix, there's a grid on your screen, "Graphics Screen Locations". The grid divides your screen into the 64 (0 to 63) horizontal locations and 32 (0 to 31) vertical locations. Use this grid in planning your graphics illustrations.

All of this explains what the first two numbers are for, but what about 3, the third number? Try using some numbers other than 3 for the third number. Type each of these lines and RUN the program:

```
20 SET(31,14,4)
20 SET(31,14,1)
```

Got it figured out? With number 4, you get a red dot, and with number 1 you get a green dot. The number codes are the same as the CLS number codes — 0 to 8. These are listed in your Appendix, "BASIC Colors".

Now, what's the GOTO line for? Try deleting the GOTO line from your program and RUN it:

```
10 CLS(0)
20 SET(31,14,1)
```

It looks like no dot was SET this time. Actually the dot was SET, but when the program ended, the Computer printed its OK message on top of the dot.

To avoid this, type the GOTO line at the end of the program. It sets up an infinite loop (going to itself over and over again) so that the program will never end.

The computer uses different screen locations for SET than PRINT @. That's why we have two grids in the Appendix. Be sure to use the one we call "Graphics Screen Locations".

SETTING TWO DOTS

To SET more than one dot, you need to do a little planning. Erase your program and RUN this program:

```
10 CLS(0)
20 SET(32,14,3)
30 SET(33,14,3)
40 GOTO 40
```

You should now have two blue dots—side by side—in the middle of your screen.

Now change the color of the right dot so you'll have one blue and one red dot. Type:

```
30 SET(33,14,4)
```

and RUN the program. . . *Both dots are red.*

Look again at the "Graphics Screen Locations" grid in your Appendix. Notice the darker lines group the dots together into blocks of four. For instance, the block in the middle of the grid contains these 4 dots:

	Horizontal	Vertical
Location	32	14
Location	33	14
Location	32	15
Location	33	15

Each dot within the block must either be:

1. the same color (colors 1-8)
- or
2. black

In our program, we tried to get the Computer to SET two dots with different colors — blue and red — within the same block. Since the Computer can't do that, it SETs both dots the second color — red.

Type this and RUN the program:



"Set Dot!"

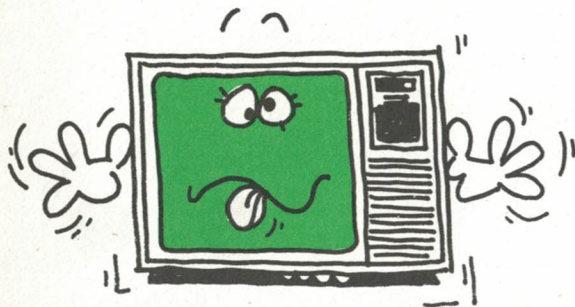
30 SET(34,14,4)

Since the dot in location 34, 14 is in a different block, the Computer can SET the two dots in different colors.

THE COMPUTER'S FACE

With this groundwork, you can draw whatever you want. We'll draw a simple picture of a Computer. First draw the top and the bottom of the head. We'll make it buff. Type:

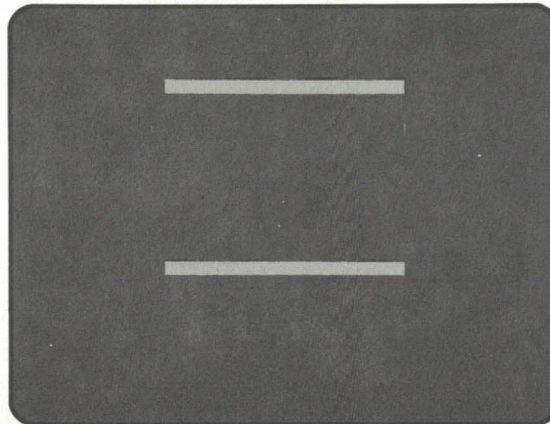
```
5 CLS(0)
10 FOR H = 15 TO 48
20 SET (H,5,5)
30 SET (H,20,5)
40 NEXT H
50 GOTO 50
```



"Funny Face!"

and RUN.

This is what you should have on your screen. (The lines should be buff rather than white, like we have them):



Lines 10 and 40 set up a FOR/NEXT loop for H — making the horizontal locations 15 through 48 for the top and the bottom lines.

Line 20 SETs the top line. The horizontal location is 15 through 48 and the vertical location is 5.

Line 30 SETs the bottom line. The horizontal location, again, is 15 through 48 and the vertical location is 20.

To SET the left and right sides of the head type these lines:

```
50 FOR V = 5 TO 20
60 SET(15,V,5)
70 SET(48,V,5)
80 NEXT V
90 GOTO 90
```

and RUN.

We'll make the nose orange. Type:

```
90 SET(32,13,8)
```

and the mouth red. Type:


```
100 FOR H = 28 TO 36
110 SET(H,16,4)
120 NEXT H
```

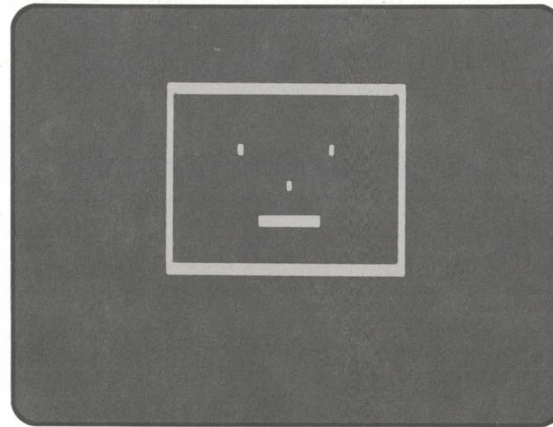
and blue eyes. Type:

```
130 SET(25,10,3)
140 SET(38,10,3)
150 GOTO 150
```

RUN the program. This is what your screen should look like now:

Notice we've changed line 50 — the GOTO line.





You don't need to tell the Computer the color of the dot to RESET (erase) it.

A BLINKING COMPUTER

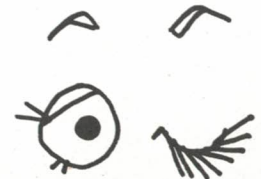
By adding a couple of lines, we can make the Computer blink. Type:

```
150 RESET(38,10)
```

and RUN the program. What you should have on your screen now is the same face as above, except the right eye is missing. RESET tells the Computer to erase the dot in the horizontal location 38 and the vertical location 10. That's the right eye.

To make it blink, we'll simply SET and RESET the right eye over and over again, by adding line 160:

```
160 GOTO 140
```



LIST your program to see if it still looks like mine:

```
5 CLS(0)
10 FOR H = 15 TO 48
20 SET(H,5,5)
30 SET(H,20,5)
40 NEXT H _____ face
50 FOR V = 5 TO 20
60 SET(15,V,5)
70 SET(48,V,5)
80 NEXT V
90 SET(32,13,8) _____ nose
100 FOR H = 28 TO 36
110 SET(H,16,4) _____ mouth
120 NEXT H
130 SET(25,10,3)
140 SET(38,10,3) _____ eyes
150 RESET(38,10) _____ blinker
160 GOTO 140
```



and RUN it . . . Try your hand at some pictures. I'm sure you have better artistic skills than we do.

THE BOUNCING DOT

By using SET and RESET, we can make a moving picture. Type and RUN these lines to make the dot go down:

```
5 CLS(0)
10 FOR V = 0 TO 31
20 SET(31,V,3)
30 RESET(31,V)
40 NEXT V
```

Remember to always erase your program before typing a NEW one.

Every dot that is SET on line 20 is RESET (erased) on line 30. Add these lines to make the dot go back up:

```
50 FOR V = 31 TO 0 STEP -1
60 SET(31,V,3)
70 RESET(31,V)
80 NEXT V
```

and this line to make the dot go up and down, over and over again:

```
90 GOTO 10
```

and RUN it. To slow the dot down — it will look a little better — change lines 30 and 70:

```
30 IF V > 0 THEN RESET(31,V-1)
70 IF V < 31 THEN RESET(31,V+1)
```

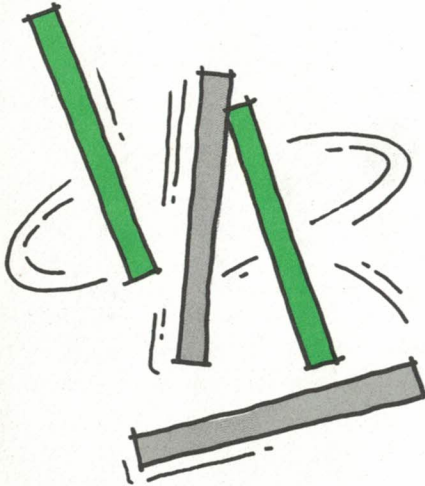
The > sign means the same as it does in math — *greater than*. The < sign means *less than*.

SET and RESET opens up all sorts of possibilities — moving targets, animated pictures, etc. Use your imagination in experimenting with this combination.

IF YOU HAVE THE JOYSTICKS...

If you have joysticks with your Computer, you have many more options open to you. If you haven't connected them yet, do it. Simply plug them in to the back of your Computer. They only fit in the correct slot, so don't worry about connecting them to the wrong one.

Now, type this short program which demonstrates how they work:



```

10 CLS
20 PRINT @ 0, JOYSTK(0);
30 PRINT @ 5, JOYSTK(1);
40 PRINT @ 10, JOYSTK(2);
50 PRINT @ 15, JOYSTK(3);
60 GOTO 20

```

Be sure to type the semicolons at the ends of lines 20, 30, 40, and 50.

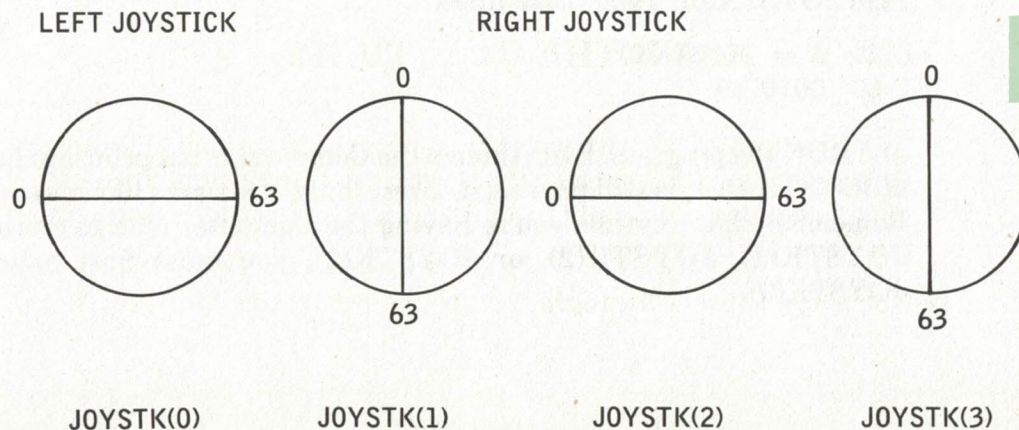
RUN the program. See the four numbers on your screen. These numbers tell the Computer the horizontal and vertical coordinates of your two joysticks' "floating switches".

Now to see what each of the four numbers are referring to. Grab the "floating switch" of one of your joysticks. Keeping it in the center, move it from left to right. Either the first number or the third number of your screen will change, going through all the numbers from 0 to 63. Move the "floating switch of your other joystick from left to right.

Place the joystick that makes the *first* number change on the *left* side.

Move the floating switches up and down, keeping them in the center. Moving the *left* joystick up and down makes the *second* number change from 0 to 63. Moving the *right* joystick up and down makes the *fourth* number change from 0 to 63.

This is how the Computer reads the position of your joysticks:



The second or fourth number might change also, but NOT from 0 to 63.

JOYSTK(0) and JOYSTK(1) tell the Computer to read the position of your *left* joystick:

- JOYSTK(0) makes it read the horizontal (left to right) coordinate.
- JOYSTK(1) makes it read the vertical (up and down) coordinate.

JOYSTK(2) and JOYSTK(3) tell the Computer to read the position of your *right* joystick:

- JOYSTK(2) makes it read the horizontal coordinate.
- JOYSTK(3) makes it read the vertical coordinate.

One more thing. Delete line 50 and RUN the program. It works almost the same, doesn't it, except it doesn't read JOYSTK(3) — the vertical position of your right joystick.

Now delete line 20 and change line 60:

```
60 GOTO 30
```

RUN the program. Move all the switches around. This time it doesn't work at all. The Computer will not read any of the coordinates unless you first have it read JOYSTK(0). Type these lines:

```
20 A = JOYSTK(0)  
60 GOTO 20
```

and RUN the program. Even though the Computer is not printing the location of JOYSTK(0), it is still reading it. Everything else works like it's supposed to. Remember that anytime you're having the Computer read to coordinates of JOYSTK(1), JOYSTK(2), or JOYSTK(3), you must first have it read JOYSTK(0).



MAKE PAINT BRUSHES OUT OF JOYSTICKS:

Type this:

```
10 CLS(0)
20 H = JOYSTK(0)
30 V = JOYSTK(1)
40 IF V > 31 THEN V = V - 32
80 SET(H,V,3)
90 GOTO 20
```

RUN it . . . Use the revolving switch of your *left* joystick to paint a picture. (Move the switch slowly so that the Computer has time to read its coordinates).

Line 20 reads H — the horizontal position of your left joystick. This could be a number from 0 to 63.

Line 30 reads V — its vertical position. This also could be a number from 0 to 63. Since the highest vertical position on your screen is 31, we had to add line 40 to the program. Line 40 makes V always equal to a number from 0 to 31.

Line 80 SETs a blue dot at H and V.

Line 90 goes back to get the next horizontal and vertical positions of your joysticks.

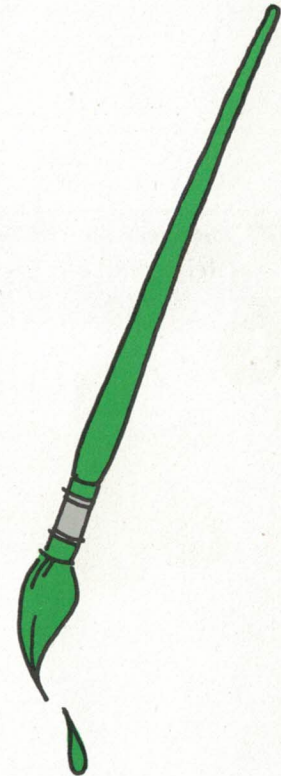
We haven't even used the right joystick. Perhaps we could use it for color. Add these lines:

```
50 C = JOYSTK(2)
60 IF C < 31 THEN C = 3
70 IF C > = 31 THEN C = 4
80 SET(H,V,C)
```

RUN the program. Move your right joystick to the right and the Computer makes C = 3. It SETs red dots. Move it to the left and the Computer makes C = 4 and SETs blue dots.

Want to make the buttons on your joysticks do something? Add these lines to the end of your program:

\geq means greater than or equal to



```
100 P = PEEK(65280)
110 PRINT P
120 GOTO 100
```

Now type:

```
RUN 100 (ENTER)
```

This tells the Computer to only RUN lines 100 through the end of the program. Your computer should be printing either 255 or 127 over and over again.

PEEK tells the Computer to look at a certain spot in its memory to see what number's there. We had it look at the number in location 65280. As long as you're not pressing either of the buttons, this spot contains the number 255 or 127.

If you press the buttons when you're not RUNning the program you will get @ABCDEFGF or HIJKLMNO.

Press the left button. When you press it, this memory location contains either the number 126 or 254.

Press the right button. This makes this memory location contain either the number 125 or 253.

Using this information, you can make the computer do whatever you want when you press one of the buttons. We'll make it go back to line 10 and CLS(0) — clear the screen to black — when you press the left button. Change lines 110 and 120:

```
110 IF P = 126 THEN 10
120 IF P = 254 THEN 10
```

Delete line 90 and add this line:

```
130 GOTO 20
```

RUN the program. Start your paintings. Press the left button when you want to clear the screen and start over again.

LEARNED IN CHAPTER 9

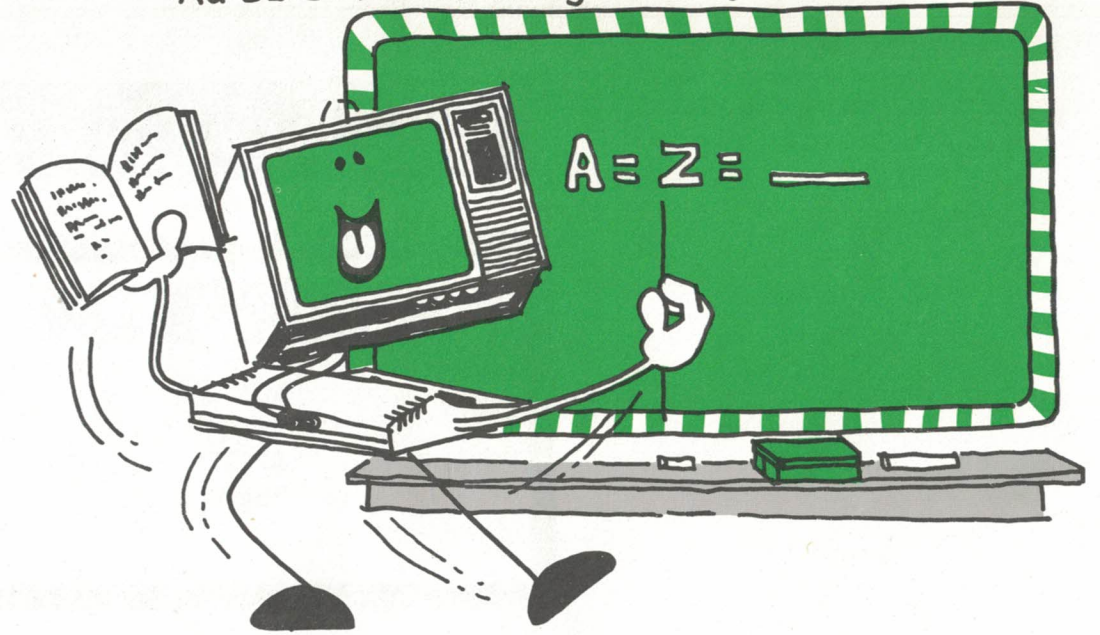
BASIC WORDS

SET
RESET
JOYSTK
PEEK

NOTES:

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo

CHAPTER 10



ONE FANTASTIC TEACHER

10

ONE FANTASTIC TEACHER

Your Computer has all the attributes of a natural born teacher. After all, it's patient, tireless, and detail conscious (. . . perhaps a bit nit-picky . . .). Depending on the programmer — we're talking about you, of course — it can be imaginative, consoling, and quite enthusiastic.

So lets get on with it! We can use RND to get the Computer to drill us on one math problem after the next. Type:

```
→ 10 CLS
   20 X = RND(15)
   30 Y = RND(15)
   40 PRINT "WHAT IS" X "*" Y
   45 INPUT A
   50 IF A = X * Y THEN 90

   60 PRINT "THE ANSWER IS" X*Y
   70 PRINT "BETTER LUCK NEXT TIME"
   80 GOTO 100

   90 PRINT "CORRECT!!!"

  100 PRINT "PRESS <ENTER> WHEN READY FOR ANOTHER"
  105 INPUT A$
  110 GOTO 10
```



This program will drill you on your multiplication tables, from 1 to 15, and check your answers.

How would you change this program to get the Computer to drill you on addition problems from 1 to 100:

DO-IT-YOURSELF PROGRAM

Here's the lines we changed:

```
20 X = RND(100)
30 Y = RND(100)
40 PRINT "WHAT IS" X "+" Y
45 INPUT A
50 IF A = X + Y THEN 90
60 PRINT "THE ANSWER IS" X + Y
```

To make the program a little more interesting we can have the Computer keep a running total of all the correct answers. Type:

```
15 T = T + 1
95 C = C + 1
98 PRINT "THAT IS" C "OUT OF" T "CORRECT ANSWERS"
```

T keeps a count of all the questions the Computer asks you. When you first

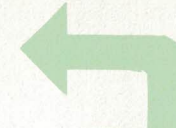
RUN the program T equals zero. Then, everytime the Computer gets to line 15, it adds 1 to T.

C does just about the same thing. It keeps a count of the number of correct answers. Since it is on line 95, the Computer will not increase C unless you get a correct answer.

There are many ways to make this program more entertaining. Add some lines to the program which will get the Computer to do one or more of the following:

1. Call you by name
2. Reward your correct answer with a sound and light show
3. Print the problem and messages attractively on your screen. (Use PRINT @ for this).
4. Keep a running total of the percent of correct answers.
5. End the program if you get 10 answers in a row correct.

Use your imagination on this one. We have a program in back which does all five of the above.



When you first turn on the Computer, all numeric variables equal 0. Also, when you type NEW (ENTER), all numeric variables equal 0.

There are many variations you could try with this program. For instance, the Computer could test you with business questions.

DO-IT-YOURSELF PROGRAM

FIRST BUILD YOUR COMPUTER'S VOCABULARY...

To build your Computer's vocabulary (so that it can build yours!) type and RUN this program:

```
10 DATA APPLES, ORANGES, PEARS
20 FOR X = 1 TO 3
30 READ F$
40 NEXT X
```

So what happened? Nothing? Nothing that you can see, that is. To see what the Computer is doing, add this line and RUN it:

```
35 PRINT "F$ = :" F$
```

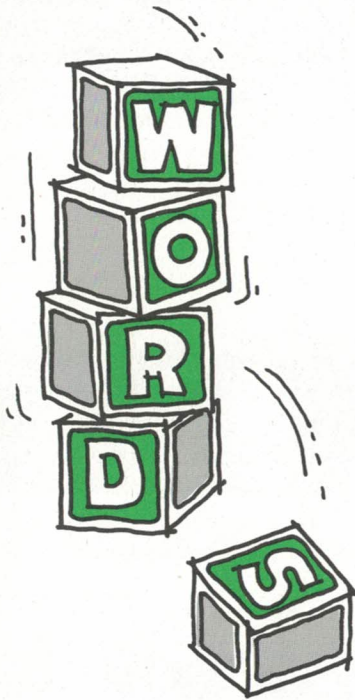
Line 30 tells the Computer to:

1. Look for a DATA line
2. READ the first item in the list — APPLES
3. Give APPLES an F\$ label
4. "Cross out" APPLES

The second time the Computer gets to line 30 it is told to do the same things:

1. Look for a DATA line
2. READ the first item — this time it is ORANGES
3. Give ORANGES the F\$ label
4. "Cross out" ORANGES

This is what is happening in your Computer's memory when you RUN the program:



What if you want the Computer to READ the same list over again? It's already crossed everything out . . . Type:

```
60 GOTO 10
```

and RUN the program. It prints ?OD ERROR IN 30. OD means Out of Data. The Computer has already crossed out its Data.

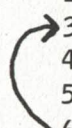
Type this line and RUN the program:

```
50 RESTORE
```


Now it's as if the Computer never crossed anything out. The Computer will READ the list over and over again.

The nice thing about DATA lines is that you can put them anywhere you want in the program. RUN each of these programs:

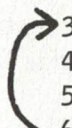
```
10 DATA APPLES
20 DATA ORANGES
30 FOR X = 1 TO 3
40 READ F$
50 PRINT "F$ = :" F$
60 NEXT X
70 DATA PEARS
```




```
10 DATA APPLES, ORANGES
20 DATA PEARS
30 FOR X = 1 TO 3
40 READ F$
50 PRINT "F$ = :" F$
60 NEXT X
```



```
30 FOR X = 1 TO 3
40 READ F$
50 PRINT "F$ = :" F$
60 NEXT X
70 DATA APPLES
80 DATA ORANGES
90 DATA PEARS
```



```
30 FOR X = 1 TO 3
40 READ F$
50 PRINT "F$ = :" F$
60 NEXT X
70 DATA APPLES, ORANGES, PEARS
```



*Remember how to make the Computer pause while RUNning a program? Press **SHIFT** @ Press any key to get it to continue.*

They all work the same, don't they? This knowledge should be handy for something . . .

...NOW HAVE IT BUILD YOUR VOCABULARY

Here's some words and definitions you might want to be tested on:

Words	Definitions
10	DATA TACITURN, HABITUALLY UNTALKATIVE
20	DATA LOQUACIOUS, VERY TALKATIVE
30	DATA VOCIFEROUS, LOUD AND VEHEMENT
40	DATA TERSE, CONCISE
50	DATA EFFUSIVE, DEMONSTRATIVE OR GUSHY

Now to get the Computer to pick out a word at random from the list. Hmm . . . there are ten items in the list. Maybe this will work:

```
60 N = RND(10)
70 FOR X = 1 TO N
80 READ A$
90 NEXT X
100 PRINT "THE RANDOM WORD IS :" A$
```

RUN it a couple of times to see if it works.

It doesn't quite work like we want it to. The Computer is just as likely to stop at a definition as a word. What we really want the Computer to do is to pick a random word from items 1, 3, 5, 7, or 9.

Fortunately, there is a word which will explain this to the Computer. Type:

```
65 IF INT(N/2) = N/2 THEN N = N - 1
```

RUN the program a few times. It should work now.

INT tells the Computer to only look at the whole portion of the number and



"Cataclysmic!"

ignore the decimal part. For instance, the Computer sees INT(3.9) as 3.

Here's how line 65 works. Say the random number the Computer picks is 10. The Computer does this calculation:

```
INT(10/2) = 10/2
INT(5) = 5
5 = 5
```

Since this is true, 5 *does* equal 5, the Computer completes the THEN portion of the line and makes N equal to 9 (10 - 1).

However, if the Computer picks 9, it does this:

```
INT(9/2) = 9/2
INT(4.5) = 4.5
4 = 4.5
```

Since this is not true, 4 *does not* equal 4.5, the Computer doesn't subtract 1 from N. 9 remains 9.

Now that the Computer is able to READ a random word, it must also READ the word's definition. You can do this simply by adding these lines to the end of the program:

```
110 READ B$
120 PRINT "THE DEFINITION IS :'" B$
```

RUN it several times now. To get the Computer to print one random word and definition after the next, add this line to the beginning of the program:

```
5 CLEAR 100
```

to give the Computer plenty of string space. And add these lines to the end of the program:

```
130 RESTORE
140 GOTO 60
```

So that the Computer can pick out a new random word and its definition from a

clean slate of data items.

Here is the way the entire program looks now:

```
5      CLEAR
10     DATA TACITURN, HABITUALLY UNTALKATIVE
20     DATA LOQUACIOUS, VERY TALKATIVE
30     DATA VOCIFEROUS, LOUD AND VEHEMENT
40     DATA TERSE, CONCISE
50     DATA EFFUSIVE, DEMONSTRATIVE OR GUSHY
60     N = RND(10)
65     IF INT(N/2) = N/2 THEN N = N - 1
70     FOR X = 1 TO N
80     READ A$
90     NEXT X
100    PRINT "A RANDOM WORD IS :" A$
110    READ B$
120    PRINT "ITS DEFINITION IS :" B$
130    RESTORE
140    GOTO 60
```

If you like, add some more words and definitions by adding DATA lines.

For variations on this program, you might try states and capitols, cities and countries, foreign words and meanings. Got more ideas?

Want to complete this program? Try it before turning the page to see ours. Program it so that the Computer will:

1. PRINT the definition ONLY
2. Ask you for the word
3. Compare the word with the correct random word
4. Tell you if your answer is correct. If your answer is incorrect, have it PRINT the correct word.

DO-IT-YOURSELF PROGRAM

Here's our program:

```
5 CLEAR 500
10 DATA TACITURN, HABITUALLY UNTALKATIVE
20 DATA LOQUACIOUS, VERY TALKATIVE
30 DATA VOCIFEROUS, LOUD AND VEHEMENT
40 DATA TERSE, CONCISE
50 DATA EFFUSIVE, DEMONSTRATIVE OR GUSHY
60 N = RND(10)
65 IF INT(N/2) = N/2 THEN N = N - 1
70 FOR X = 1 TO N
80     READ A$
90 NEXT X
110 READ B$
120 PRINT "WHAT WORD MEANS :'" B$
130 RESTORE
140 INPUT R$
150 IF R$ = A$ THEN 190
160 PRINT "WRONG"
170 PRINT "THE CORRECT WORD IS :'" A$
180 GOTO 60
190 PRINT "CORRECT"
200 GOTO 60
```

Feel free to dress the program up with a good looking screen format, sound, etc.

LEARNED IN CHAPTER 10

BASIC WORDS

DATA
READ
RESTORE
INT
CLEAR

NOTES:

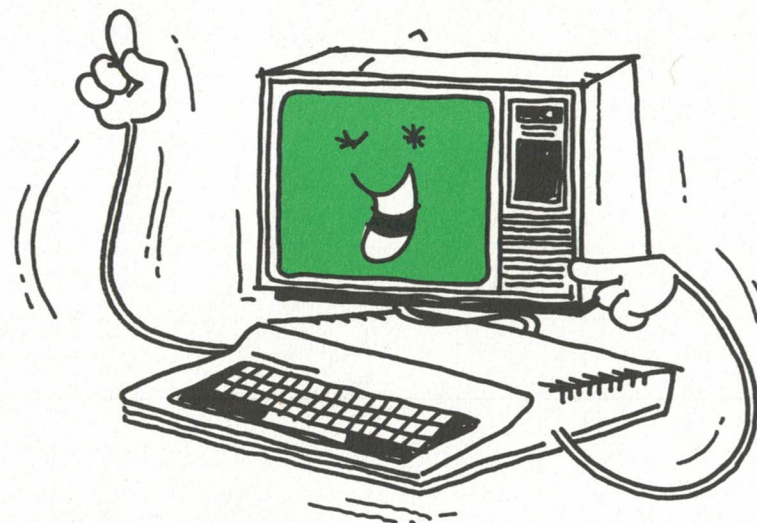


A series of horizontal lines for writing notes, located on the right side of the page.



CHAPTER 11

$$Ax (BY + C) - D + E (G/W) - F$$



HELP WITH MATH

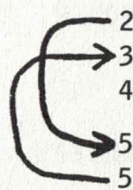


HELP WITH MATH

Solving complicated math formulas with super speed and precision is an area where your Computer shines. But before driving yourself crazy typing a bunch of math formulas, there are some shortcuts and hints you'll probably want to know about.

One easy way to handle complicated math formulas is by using SUBROUTINES. Type and RUN this program:

```
10 PRINT "EXECUTING THE MAIN PROGRAM"  
20 GOSUB 500  
30 PRINT "NOW BACK IN THE MAIN PROGRAM"  
40 END  
500 PRINT "EXECUTING THE SUBROUTINE"  
510 RETURN
```



Line 20 tells the Computer to GO to the SUBroutine beginning at line 500. RETURN tells the Computer to return back to the BASIC word immediately following GOSUB.

Delete line 40 and see what happens when you RUN the program.

.....

Did you delete it?



If you did, this is what's on your screen:

```
EXECUTING THE MAIN PROGRAM
EXECUTING THE SUBROUTINE
NOW BACK IN THE MAIN PROGRAM
EXECUTING THE SUBROUTINE
?RG ERROR IN 510
```

RG means RETURN without GOSUB. Can you see why deleting END in line 40 would cause this error?

At first, the Computer went through the program just like it did before you deleted the END line. It was sent to the subroutine in line 500 by GOSUB and it returned to the BASIC word immediately following GOSUB.

Then, since you deleted END, it went to the next line in the program which was the subroutine. When it got to RETURN, it did not know where to return to, since this time it had not been sent to the subroutine by a GOSUB.

Here's a subroutine which raises a number to any power you want:

See something different about INPUT? We can have the Computer PRINT a message before waiting for us to INPUT something.

```
10 INPUT "TYPE A NUMBER"; N
20 INPUT "TYPE THE POWER YOU WANT IT RAISED TO"; P
30 GOSUB 2000
40 PRINT : PRINT N " TO THE " P " POWER IS " E
50 GOTO 10
2000 REM FORMULA FOR RAISING A NUMBER TO A POWER
2010 E = 1
2020 FOR X = 1 TO P
2030 E = E * N
2040 NEXT X
2050 IF P = 0 THEN E = 1
2060 RETURN
```

Notice we introduced a couple of new things in the program.

Look at line 40. If you find it easier, you can combine two or more program lines into one, using a colon to separate the two lines. Line 40 contains the two lines:

```
PRINT  
and  
PRINT N " TO THE " P " POWER IS " E
```

Line 2000 has something else new — REM.

REM doesn't mean anything to the Computer. The Computer ignores any line beginning with REM. You can put REM lines anywhere you want in your program, so that you can remember what the program does. These REM lines will make *no difference to the way the program works*.

If you don't believe us, add these lines and RUN the program to see if they make any difference:

```
5 REM THIS IS A PECULIAR PROGRAM,  
17 REM THIS LINE SHOULD MESS UP THE PROGRAM  
45 REM THE NEXT LINE KEEPS THE SUBPROGRAM SEPARATED
```

Satisfied? . . . *Good*.

Change the program so that the Computer prints a table of squares (a number to the power of 2) for numbers, say, from 2 to 10.

PRINT by itself tells the Computer to skip a line.

DO-IT-YOURSELF PROGRAM

The answer is in the back of the book.

GIVE THE COMPUTER A LITTLE HELP

As math formulas get more complex, your Computer will need some help understanding them. For example, what if you want the Computer to solve this problem:

Divide the sum of $13 + 3$ by 8

You might want the Computer to solve the problem like this:

$$13 + 3 / 8 = 16 / 8 = 2$$

But your Computer solves it differently. Type this command line:

PRINT $13 + 3 / 8$ **ENTER**

What the Computer did was logical according to its rules:

RULES ON ARITHMETIC

The Computer solves arithmetic problems in this order

1. any multiplication and division operations are solved first
2. addition and subtraction operations are solved last
3. in case of a tie (that is, more than one multiplication/division or addition/subtraction operation) the operations are performed from left to right

The word "operation" means something you're getting the Computer to do. Here, we're talking about the "operations" of adding, subtracting, multiplying, and dividing.

In the problem above the Computer followed its rules. It performed the division operation first ($3/8 = .375$) and then the addition ($13 + .375 = 13.375$). To get the Computer to solve the problem differently, you can use parenthesis. Type this line:

```
PRINT (13 + 3) / 8 ENTER
```

Whenever the Computer sees an operation in parenthesis, it solves that before solving anything else.

What do you think the Computer will PRINT as the answers to each of these problems:

COMPUTER MATH EXERCISE

```
PRINT 10 - (5 - 1) / 2 _____  
PRINT 10 - 5 - 1 / 2 _____  
PRINT (10 - 5 - 1) / 2 _____  
PRINT (10 - 5) - 1 / 2 _____  
PRINT 10 - (5 - 1 / 2) _____
```

Finished? Type each of the command lines to check your answers.

What if you want the Computer to solve this problem?

Divide 10 minus the difference of 5 minus 1 by 2

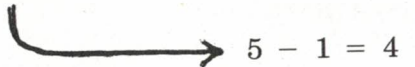
That is what you're actually asking the Computer to do:

```
(10 - (5 - 1)) / 2
```

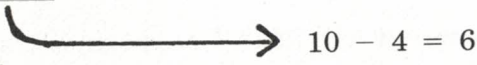
When the Computer sees a problem with more than one set of parenthesis, it

solves the inside parenthesis and then moves to the outside parenthesis. In other words, it does this:

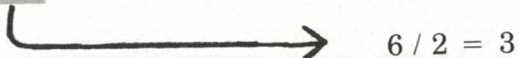
$$(10 - (5 - 1)) / 2$$



$$(10 - 4) / 2$$



$$6 / 2$$



RULES ON PARENTHESIS

1. When the Computer sees a problem containing parenthesis, it solves the operation inside the parenthesis **BEFORE** solving the rest of the operations.
2. If there are parenthesis inside parenthesis, the Computer solves the innermost parenthesis first and works its way out.

Insert parenthesis in the problem below so that the Computer will PRINT 28 as the answer:

COMPUTER MATH EXERCISE

PRINT 30 - 9 - 8 - 7 - 6

Answer:

```
PRINT 30 - (9 - (8 - (7 - 6)))
```

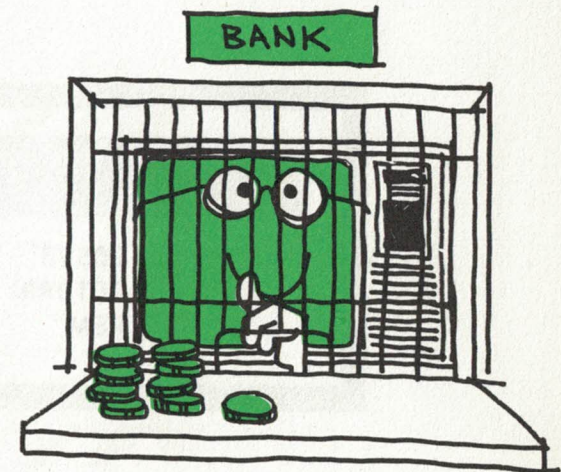
IS SAVING WORTH IT?

With what you've learned in this chapter, you can let the Computer do all the math by putting complicated math formulas in your subroutines. The program below uses two subroutines. It's for those of you who save by putting the same amount of money in the bank each month:

```
10 INPUT "YOUR MONTHLY DEPOSIT"; D
20 INPUT "BANK'S ANNUAL INTEREST RATE"; I
30 I = I/12 * .01
40 INPUT "NUMBER OF DEPOSITS"; P
50 GOSUB 1000
60 PRINT "YOU WILL HAVE $" FV " IN " P " MONTHS"
70 END

1000 REM COMPOUND MONTHLY INTEREST FORMULA
1010 N = 1 + I
1020 GOSUB 2000
1030 FV = D * ((E - 1) / I)
1040 RETURN

2000 REM FORMULA FOR RAISING A NUMBER TO A POWER
2010 E = 1
2020 FOR X = 1 TO P
2030 E = E * N
2040 NEXT X
2050 IF P = 0 THEN E = 1
2060 RETURN
```



"A PENNY SAVED..."

Notice that we have one subprogram "calling" another subroutine. This is perfectly OK with the Computer as long as you have a GOSUB to send the Computer to each subroutine, and a RETURN in each subroutine to return the

Computer to the BASIC word following each GOSUB.

If you will be using your Computer a lot to help with math or accounting, you might want to save all your math formulas on tape as subroutines. Then, when you are writing a program that uses one of them, simply load the subroutine from tape — it'll then be part of your program.

LEARNED IN CHAPTER 11

BASIC WORDS

GOSUB
RETURN
REM

BASIC SYMBOLS

()
:

BASIC CONCEPTS

Order of operations

NOTES:

Handwritten notes in pencil are faintly visible in the top left corner of the page. The rest of the page is filled with blank horizontal lines, organized into two columns.



CHAPTER 12



A GIFT WITH WORDS



A GIFT WITH WORDS

One of your Computer's greatest skills is its gift with words. It can tirelessly twist, combine, or separate words to anything you want. Because of this gift, you can teach it to read, write, and even carry on a halfway decent conversation.

For starters, see what you think of this:

```
10 PRINT "TYPE A SENTENCE"  
20 INPUT S$  
30 PRINT "YOUR SENTENCE HAS " LEN(S$) " CHARACTERS"  
40 INPUT "WANT TO TRY ANOTHER"; A$  
50 IF A$ = "YES" THEN 10
```

Impressed? LEN(S\$) tells the Computer to compute the LENGTH of the string S\$ — your sentence. Your obedient Computer counts every single character in the sentence, including spaces and punctuation marks.

Here's another skill it has. Erase your program and type this to make it compose a poem (of sorts):

```
10 A$ = "A ROSE"  
20 B$ = " "  
30 C$ = "IS A ROSE"  
40 D$ = B$ + C$  
50 E$ = "AND SO FORTH AND SO ON"  
60 F$ = A$ + D$ + D$ + B$ + E$  
70 PRINT F$
```

Not impressed? Well, later we'll talk about some practical ways to use this unusual skill.

Here the Computer combines strings. The plus sign tells it to do this. D\$ combines B\$ and C\$ to get "IS A ROSE", and you can see what strings are combined to form F\$.

A word of caution about combining strings — add this line to your program and RUN it:

```
80 G$ = F$ + F$ + F$ + F$ + F$ + F$ + F$
```

When you RUN this program, the Computer prints ?OS ERROR IN 80. OS means Out of String Space. The Computer only reserves about 50 characters for working with strings. Add this line to the beginning of the program for reserving plenty of string space:

```
5 CLEAR 500
```

RUN the program again. This takes care of the first problem, but there's still another.

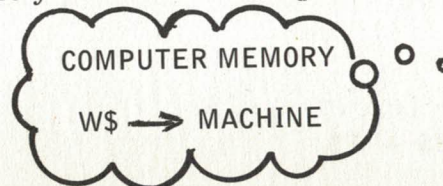
This time the Computer prints ?LS ERROR IN 80. LS means string too long. Line 80 asks the Computer to form a string — G\$ — with more than 255 characters. Your Computer simply can't manage to remember a string with that many characters.

Now that the Computer has combined strings, let's get it to take one apart. Type and RUN this program:

```
10 INPUT "TYPE A WORD"; W$
20 PRINT "THE FIRST LETTER IS : " LEFT$ (W$,1)
30 PRINT "THE LAST 2 LETTERS ARE : " RIGHT$ (W$,2)
40 GOTO 10
```

Here's what your Computer is doing:

In line 10 you INPUT a string for W\$. Let's say the string is MACHINE:



Your Computer then performs several calculations in lines 20 and 30 to get the first LEFT letter and the last 2 RIGHT letters of the string:



Try RUNNING the program a few times until you get the hang of it.

Add this line to the program:

```
5 CLEAR 500
```

So that your Computer will set aside plenty of space for working with strings. Now INPUT a sentence rather than a word.

.....

How would you change lines 20 and 30 so that the Computer will give you the first 5 letters and the last 6 letters of your string?

PROGRAMMING EXERCISE

```
20 .....  
30 .....
```

Answers:

```
20 PRINT "THE FIRST FIVE LETTERS ARE :" LEFT$ (W$,5)  
30 PRINT "THE LAST SIX LETTERS ARE :" RIGHT$ (W$,6)
```

.....

Erase your program and type this one:

```
10 CLEAR 500
20 INPUT "TYPE A SENTENCE"; S$
30 PRINT "TYPE A NUMBER FROM 1 TO " LEN(S$)
40 INPUT X
50 PRINT "THE MIDSTRING WILL BEGIN WITH CHARACTER " X
60 PRINT "TYPE A NUMBER FROM 1 TO " LEN(S$) - X + 1
70 INPUT Y
80 PRINT "THE MIDSTRING WILL BE" Y "CHARACTERS LONG"
90 PRINT "THIS MIDSTRING IS :" MID$(S$,X,Y)
100 GOTO 20
```

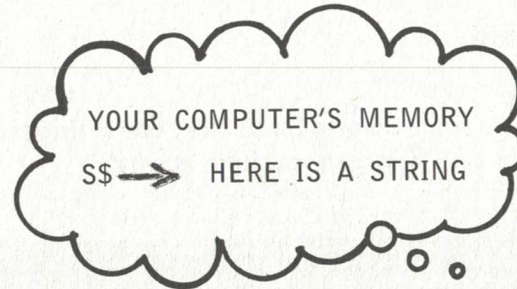
Remember how to erase a program?

Type:

NEW **ENTER**

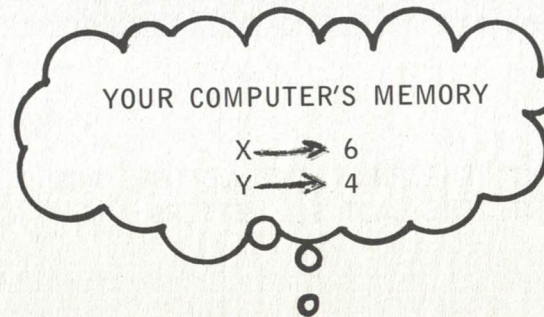
RUN this program a few times to see if you can figure out how MID\$ works.

Here's how the program works. Say you INPUT "HERE IS A STRING" for your sentence:



In line 30, the computer first calculates the LENGTH of S\$ — 16 characters. It then asks you to choose a number from 1 to 16. Let's say you pick the number 6.

The Computer then, in line 60, asks you to pick another number from 1 to 11 (16-6) plus 1. Say you pick the number 4.



In line 90, the Computer gives you a MID string of S\$ which begins at character number 6 and is 4 characters long:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
H	E	R	E		I	S		A		S	T	R	I	N	G

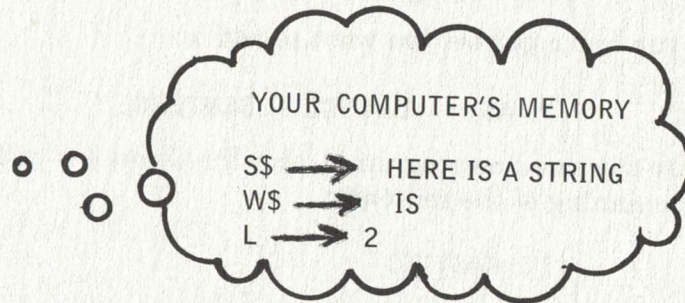
← 4 →
MID\$(S\$,6,4)

Here's something else you can do with MID\$. Erase your program and type:

```
10 INPUT "TYPE A SENTENCE"; S$
20 INPUT "TYPE A WORD IN THE SENTENCE"; W$
30 L = LEN(W$)
40 FOR X = 1 TO LEN(S$)
50 IF MID$(S$,X,L) = W$ THEN 90
60 NEXT X
70 PRINT "YOUR WORD ISN'T IN THE SENTENCE"
80 END
90 PRINT W$ "--BEGINS AT CHARACTER NO." X
```

RUN this program a few times. Here's how it works.

Say you type in the above sentence, and the word you INPUT for W\$ is "IS". In line 30, the Computer then calculates the LENGTH of W\$ — 2 characters.



The Computer then goes through the FOR/NEXT loop (lines 40-90) counting each character in S\$ beginning with character 1 and ending with character number LEN(S\$) — 16.

These types of programs can be used to sort through large files of information. For instance, by separating strings, you could look through a mailing list for TEXAS addresses.

Every time it counts a new character, the Computer looks at a new MID string. Each MID string begins at character X and is L or 2 characters long.

For example, when X equals 1, the Computer looks at this MID string:

```

  1
  H E R E I S A S T R I N G
  ←2→
MID$(S$,1,2)

```

The fourth time through the loop, when X equals 4, the Computer looks at this:

```

      4
  H E R E I S A S T R I N G
      ←2→
MID$(S$,4,2)

```

It finally finds the MID string it is looking for when X equals 6.

YOUR COMPUTER CAN BE A TOUGH EDITOR

Are you beginning to get a picture of the Computer hacking away at your sentences with a red pen? You can program it to help refine your writing and save you hours of typing and rewriting.

Say you had a phrase you want to add to:

```
10 A$ = "CHANGE A SENTENCE."
```

Add to this one-line program so that the Computer will insert these words at the beginning of the sentence:

```
IT'S EASY TO
```

and PRINT the new sentence:

```
IT'S EASY TO CHANGE A SENTENCE
```



"#@%|&\$!"

DO-IT-YOURSELF PROGRAM

This is our program:

```
10 A$ = "CHANGE A SENTENCE."  
20 B$ = "IT'S EASY TO"  
30 C$ = B$ + " " + A$  
40 PRINT C$
```

Now see if you can add to the program to get the Computer to:

1. Find the beginning location of the MID string:

A SENTENCE

2. Delete the words A SENTENCE, forming a new string:

IT'S EASY TO CHANGE

3. Add these words to the end of the new string:

ANYTHING YOU WANT

4. And PRINT the newly formed string:

IT'S EASY TO CHANGE ANYTHING YOU WANT

DO-IT-YOURSELF PROGRAM

HINT: To form the string IT'S EASY TO CHANGE, you need to get the LEFT portion of the string IT'S EASY TO CHANGE A SENTENCE.

.....

Answer:

```
10 A$ = "CHANGE A SENTENCE."  
20 B$ = "IT'S EASY TO"  
30 C$ = B$ + " " + A$  
40 PRINT C$  
50 Y = LEN ("A SENTENCE")  
60 FOR X = 1 TO LEN(C$)  
70 IF MID$(C$,X,Y) = "A SENTENCE" THEN 90  
80 NEXT X  
85 END  
90 D$ = LEFT$(C$,X - 1)  
100 E$ = D$ + "ANYTHING YOU WANT"  
110 PRINT E$
```

This type of program is the basis of a "word processing" program — a popular program for cutting down on office typing expenses.

IF YOU LIKE A CHALLENGE, TRY THIS. . .

Write a program in which:

1. The Computer asks you to INPUT:
 - a. a sentence
 - b. a phrase within the sentence to delete
 - c. a new phrase to replace the deleted phrase
2. The Computer then PRINTs a new sentence with your change intact.

This may take a while, but you have everything you need to write it. Our answer is in the back of the book:

DO-IT-YOURSELF CHALLENGER PROGRAM

LEARNED IN CHAPTER 12

BASIC WORDS	BASIC String OPERATOR
LEN LEFT\$ RIGHT\$ MID\$	+

NOTES:

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>
---	---

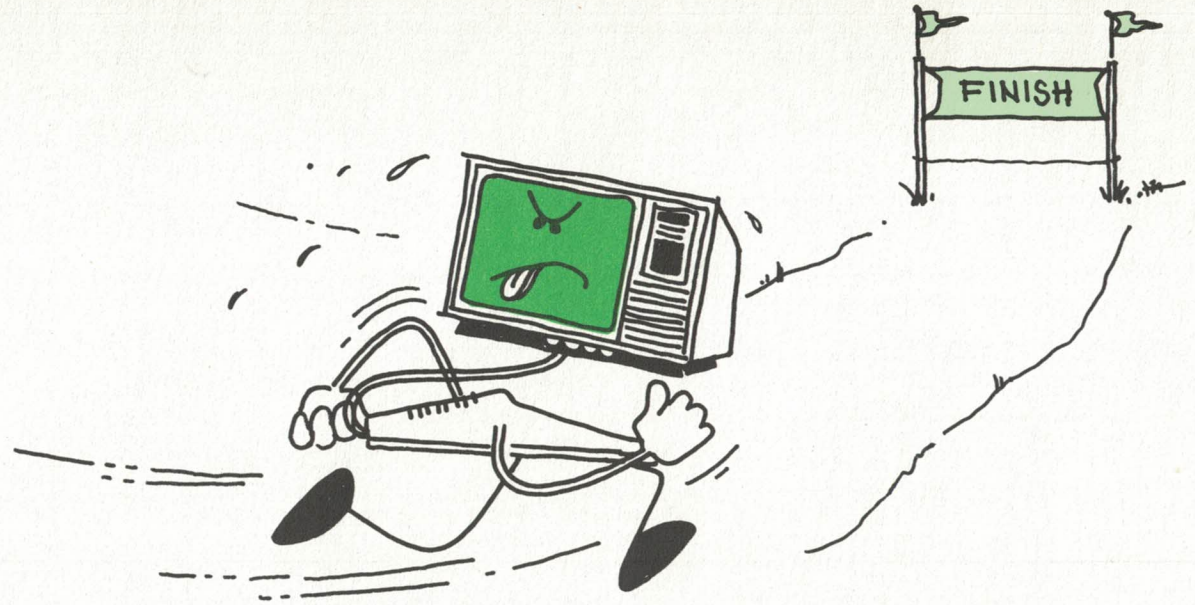


NOTES:

Handwritten notes on lined paper, including a faint pencil drawing of a mechanical device with a cylindrical component and a handle. The drawing is located in the upper left quadrant of the page. The rest of the page contains faint, illegible text and a large, faint watermark that reads "BEAT THE COMPUTER".



CHAPTER 13



BEAT THE COMPUTER





BEAT THE COMPUTER

You'll find the Computer much more adept by getting it to constantly watch and react to everything you do. By "watching you", we mean watching the keyboard to see if you are pressing something. The word INKEY\$ makes this possible.

Type this:

```
10 A$ = INKEY$
20 IF A$ <> "" GOTO 50
30 PRINT "YOU PRESSED NOTHING"
40 GOTO 10
50 PRINT "THE KEY YOU PRESSED IS---" A$
```



Remember what <> means? (It means "not equal to")

"" means an empty string — nothing

Press a key while RUNning this program.

INKEY\$ tells the Computer to look at the keyboard to see if you have pressed anything. The Computer does this with super-speed. You will have pressed absolutely nothing for at least the first 20 times the Computer checks.

The Computer labels this key, or this non-key (""), A\$. Then the Computer makes its decision:

If A\$ equals "" — *nothing* — the Computer prints "YOU PRESSED NOTHING" and goes back up to line 10 to check the keyboard again.

However, if A\$ equals *something*, the Computer goes to line 50 and prints the key.

For a constant look-out, type this and RUN the program:

```
60 GOTO 10
```

No matter how quick you are, the Computer is much faster! Erase line 30 to see what keys you're pressing.

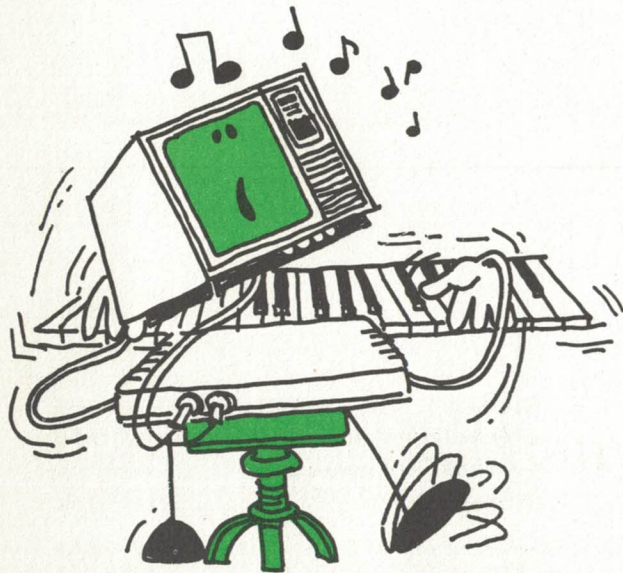
AN ELECTRONIC PIANO

Try using INKEY\$ to make a piano out of your keyboard. Look at that table in the Appendix, "Musical Tone Numbers". It lists these as the tones for middle C through the next higher C:

C - 89	E - 125	G - 147	B - 170
D - 108	F - 133	A - 159	C - 176

We can tell the Computer that if you press a certain key it should SOUND one of these tones. Erase your program and type:

```
10 A$ = INKEY$
20 IF A$ = "" THEN 10
30 IF A$ = "A" THEN T = 89
40 IF A$ = "S" THEN T = 108
50 IF A$ = "D" THEN T = 125
60 IF A$ = "F" THEN T = 133
70 IF A$ = "G" THEN T = 147
80 IF A$ = "H" THEN T = 159
90 IF A$ = "J" THEN T = 170
100 IF A$ = "K" THEN T = 176
110 IF T = 0 THEN 10
120 SOUND T, 5
130 T = 0
140 GOTO 10
```



RUN the program . . . Well, what are you waiting for? Play a tune. Type any of the keys on the third row down on your keyboard — from A to K.

*How would this change the program?
120 SOUND T, 1*

Why wouldn't the program work right if you use INPUT rather than INKEY\$?

.....

If you use INPUT the Computer will wait until you press **ENTER** before acknowledging what you type. With INKEY\$, it sees everything you type.

There's another way of writing this program using READ and DATA lines. Do you know how this would be done?

.....

This is what we came up with:


```
10 A$ = INKEY$
20 FOR X = 1 TO 8
30 READ B$, T
40 IF A$ = B$ THEN SOUND T, 5
50 NEXT X
60 RESTORE
70 GOTO 10
80 DATA A, 89, S, 108
90 DATA D, 125, F, 133
100 DATA G, 147, H, 159
110 DATA J, 170, K, 176
```

Using these DATA and READ lines will make it easier for you to add more tones to your Computer's repertoire.

BEAT THE COMPUTER

Type this program:

```
10 X = RND(4)
20 Y = RND(4)
30 PRINT "WHAT IS" X "+" Y
40 T = 0
50 A$ = INKEY$
60 T = T + 1
70 SOUND 128, 1
80 IF T = 15 THEN 200
90 IF A$ = "" THEN 50
100 GOTO 10
200 CLS(7)
210 SOUND 180, 30
220 PRINT "TOO LATE"
```



Here's what the program tells the Computer to do:

Lines 10, 20, and 30 gets the Computer to pick two random numbers and ask you what their sum is.

Line 40 sets T to 0. We will use T as a timer.

Line 50 gives you your first chance to answer the question — in one minute split second.

Line 60 adds 1 to the timer. T now equals 1. The next time the Computer gets to line 60 it again adds 1 to the timer to make T equal 2. Everytime the Computer executes line 60 it will add 1 to T.

Line 70's just there to make you nervous.

Line 80 tells the Computer you have 15 chances to answer. Once T equals 15, time's up. The Computer will insult you with lines 200, 210, and 220.

Line 90 says if you haven't answered yet to go back and give you another chance.

The Computer only gets to line 100 if you do answer. It will go back for another problem.

How would you get the Computer to give you three times as much time to answer each question?

Answer:

By changing this line:

```
80 IF T = 45 THEN 200
```

CHECKING YOUR ANSWERS

How would you get the Computer to check to see if your answer is correct? Would this work?

```
100 IF A$ = X + Y THEN 130
110 PRINT "WRONG", X "+" Y "=" X + Y
120 GOTO 10
130 PRINT "CORRECT"
140 GOTO 10
```

If you RUN this program (and answer on time), you'll get this error message:

```
?TM ERROR IN 100
```

That's because you can't make A\$, a *string*, equal to X + Y, *numbers*. You have to somehow change A\$ to a number.

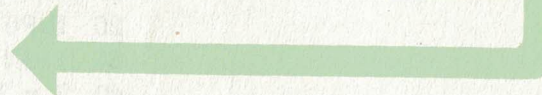
Fortunately, your Computer has a way of doing this. Change line 100 by typing:

```
100 IF VAL(A$) = X + Y THEN 130
```

VAL(A\$) turns A\$ into its numeric VALUE. If A\$ equals the string "5"; VAL(A\$) equals the number 5. (However, if VAL(A\$) equals the string "C"; VAL(A\$) equals 0 since "C" has no numeric value.)



Remember the problem of mixing strings with numbers. Chapter 2 will refresh your memory.



For those that want to make the program a bit more challenging, change these lines:

```
10 X = RND(49) + 4
20 Y = RND(49) + 4
90 B$ = B$ + A$
100 IF VAL(B$) = X + Y THEN 130
```

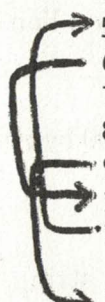
And add these lines:

```
45 B$ = ""
95 IF LEN(B$) <> 2 THEN 50
```

A COMPUTER TYPING TEST

Here's a program that will get the Computer to time how fast you type:

```
10 CLS
20 INPUT "PRESS <ENTER> WHEN READY TO TYPE THIS PHRASE"; E$
30 PRINT "NOW IS THE TIME FOR ALL GOOD MEN"
40 T = 1
50 A$ = INKEY$
60 IF A$ = "" THEN 100
70 PRINT A$;
80 B$ = B$ + A$
90 IF LEN(B$) = 32 THEN 120
100 T = T + 1
110 GOTO 50
120 S = T/74
130 M = S/60
140 R = 8/M
150 PRINT
160 PRINT "YOU TYPED AT —"R"—WDS/MIN"
```



Here's how this program works:

In line 40, we set the timer — T — to 1.

Line 50 gives you your first opportunity to type a key — A\$. If you're not quick enough, line 60 sends the Computer directly to line 100 and adds 1 to the timer.

Line 70 prints the key you typed.

Line 80 forms a string named B\$. Each time you type a key (A\$), it will be added to B\$. For example, if the first key you type is "N", then:

```
A$ = "N"  
and  
B$ = B$ + A$  
B$ = "" + "N"  
B$ = "N"
```

If the next key you type is "O", then:

```
A$ = "O"  
and  
B$ = B$ + A$  
B$ = "N" + "O"  
B$ = "NO"
```

And if the third key you type is "W", then:

```
A$ = "W"  
and  
B$ = "NO" + "W"  
B$ = "NOW"
```

When the LENgth of B\$ equals 32 characters (the length of "NOW IS THE TIME FOR ALL GOOD MEN"), the Computer assumes you've finished typing the phrase and goes to line 120 to compute your words per minute.

We could have made this calculation in one line by using parenthesis:

$$120 R = 8 / ((T / 74) / 60)$$

We calculate the words per minute in lines 120, 130, and 140 by dividing T by 74 (to get the seconds), S by 60 (to get the minutes), and then dividing the 8 words by M to get the rate of words per minute.

Change this program to get the Computer to check to see if you made a typographical error.

DO-IT-YOURSELF PROGRAM

Our answer is in the back of this book.

How about trying a variation of this program — a speed reading test.

LEARNED IN CHAPTER 13

BASIC WORDS

INKEY\$
VAL

NOTES:

WHAT NOW?

Congratulations! By now, you should feel in full control of your Computer. You might be so busy writing programs that you don't want to read anything else yet.

It's only fair to tell you that there are other BASIC words we haven't covered. They are all listed on your Quick Reference Card.

If you want more complete instructions on how to program, Radio Shack has plenty of good books:

BASIC Computer Language—catalog number 62-2016

BASIC Computer Programming—catalog number 62-2015

Level II Programming—catalog number 62-2061

Level II Programming Techniques—catalog number 62-2062

TRS-80 Assembly Language Programming—catalog number 62-2006

Understanding Digital Computers—catalog number 62-2027

Understanding Solid-State Electronics—catalog number 62-2035

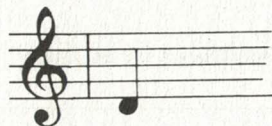
These books are written at all different levels. Browse through them at your Radio Shack store to see which are best for you.

MUSICAL TONES

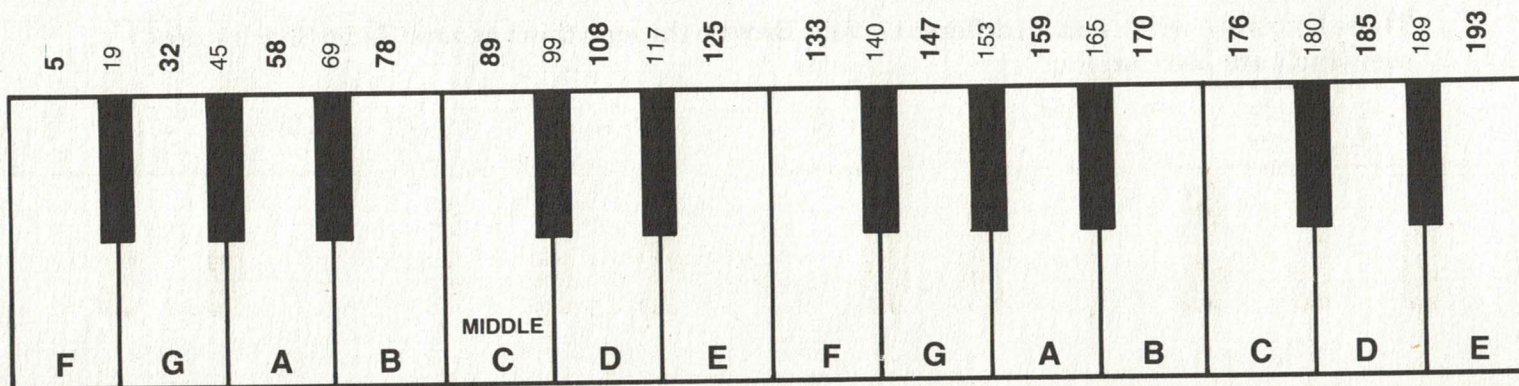
Your Computer can come fairly close to matching (although it can't *exactly* match) the musical tones shown below. You may either use the piano keyboard or the musical staff to produce electronic music.

If you're using the piano keyboard, the Computer tone for each key is directly over the key. For example, the Computer tone number for Middle C is 89.

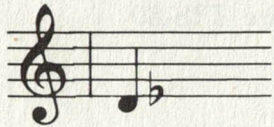
If you're using the musical staff, the tone number for each note is below the note. For example the tone number for:



is 108.

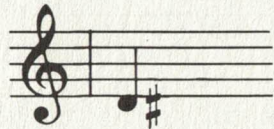


If the note is a flat, select the tone number immediately preceding the note.
For example:



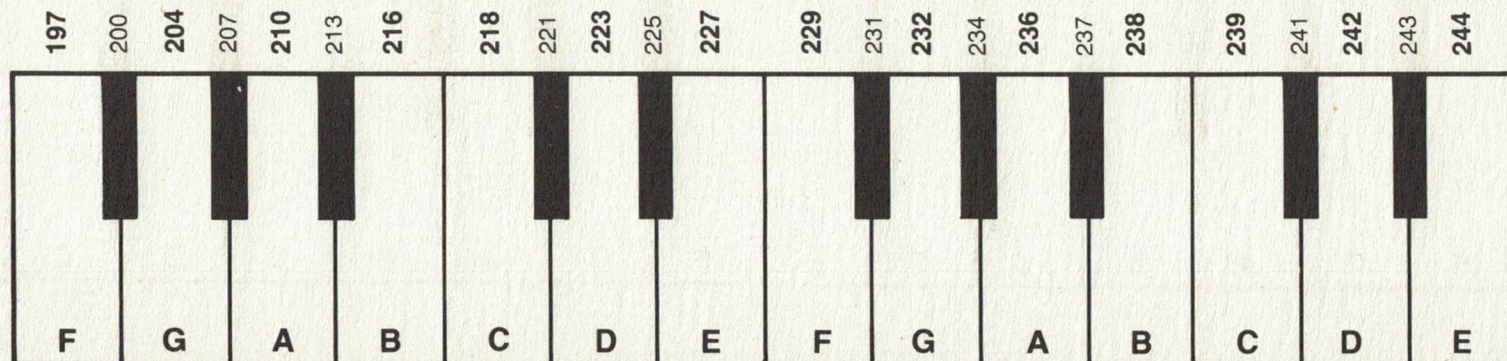
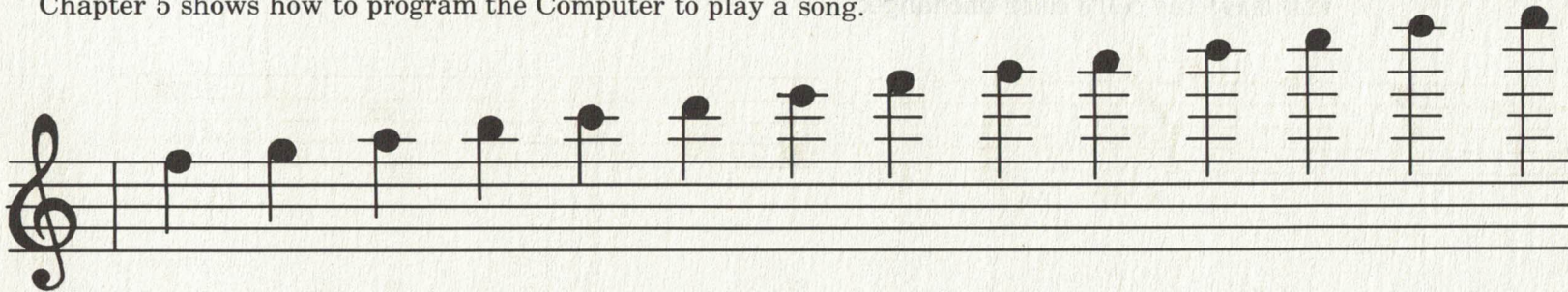
is 99.

If the note is a sharp, select the tone number immediately following the note.
For example:



is 117.

Chapter 5 shows how to program the Computer to play a song.



BASIC COLORS

These are the codes for each of the 9 colors you can create on the TRS-80
COLOR COMPUTER:

- 0 - black
- 1 - green
- 2 - yellow
- 3 - blue
- 4 - red
- 5 - buff
- 6 - cyan
- 7 - magenta
- 8 - orange

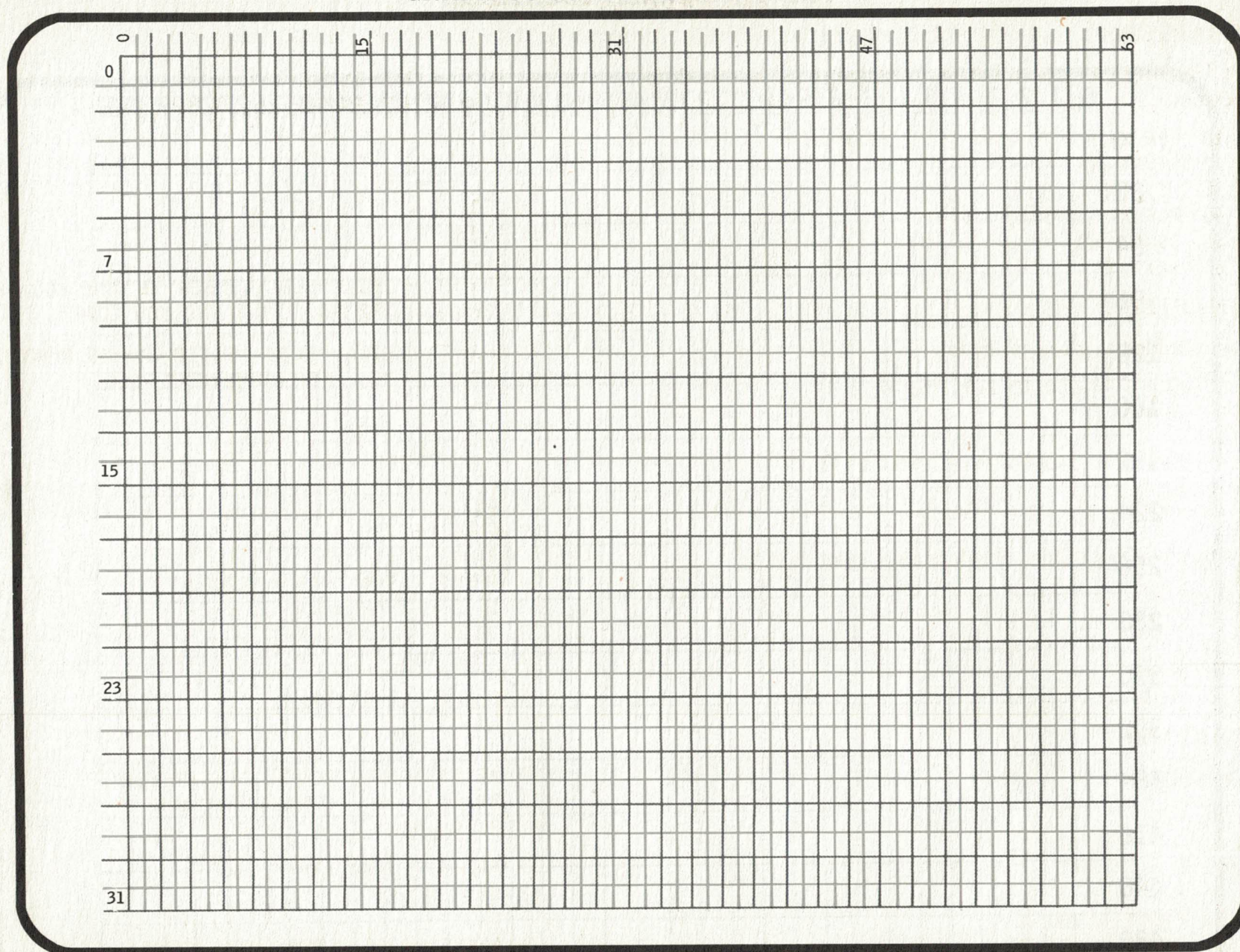
The color might vary in shade from these, depending on your T.V.

Color 0 is actually an absence of color. When using SET for graphics, color 0 will leave the cell's color unchanged.

PRINT @ SCREEN LOCATIONS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0																																	
32																																	
64																																	
96																																	
128																																	
160																																	
192																																	
224																																	
256																																	
288																																	
320																																	
352																																	
384																																	
416																																	
448																																	
480																																	

GRAPHICS SCREEN LOCATIONS



ANSWERS TO EXERCISES

CHAPTER 4

SOUNDing tones from bottom of range to top and back to bottom:

```
10 FOR X = 1 TO 255
20 SOUND X,1
30 NEXT X
40 FOR X = 255 TO 1 STEP -1
50 SOUND X,1
60 NEXT X
```

CHAPTER 5

Lines added to clock program:

```
92 FOR T = 200 TO 210 STEP 5
94 SOUND T, 1
95 NEXT T
97 FOR T = 210 TO 200 STEP -5
98 SOUND T, 1
99 NEXT T
```

Program which Shows 9 colors for 1 second each:

```
10 FOR C = 0 TO 8
20 CLS(C)
30 FOR X = 1 TO 460
40 NEXT X
50 NEXT C
```

CHAPTER 7

Craps Game

```
10 CLS
20 A = RND(6)
30 B = RND(6)
40 R = A + B
50 PRINT @ 200, A
60 PRINT @ 214, B
70 PRINT @ 394, "YOU ROLLED A" R
```

```
80 IF R = 2 THEN 500
90 IF R = 3 THEN 500
100 IF R = 12 THEN 500
110 IF R = 7 THEN 600
120 IF R = 11 THEN 600
130 FOR X = 1 TO 800
140 NEXT X
150 CLS
160 PRINT @ 195, "ROLL ANOTHER" R "AND YOU WIN"
170 PRINT @ 262, "ROLL A 7 AND YOU LOSE"

180 PRINT @ 420, "PRESS <ENTER> WHEN READY"
185 PRINT @ 456, "FOR YOUR NEXT ROLL"
190 INPUT A$
200 X = RND(6)
210 Y = RND(6)
220 Z = X + Y
225 CLS
230 PRINT @ 200, X
240 PRINT @ 214, Y
250 PRINT @ 394, "YOU ROLLED A" Z
260 IF Z = 2 THEN 500
270 IF Z = 7 THEN 600
280 GOTO 180

500 FOR X = 1 TO 1000
510 NEXT X
515 CLS
520 PRINT @ 230, "YOU'RE THE WINNER"
530 PRINT @ 294, "CONGRATULATIONS!!!"
540 GOTO 630

600 FOR X = 1 TO 1000
610 NEXT X
615 CLS
620 PRINT @ 264, "SORRY, YOU LOSE"
630 PRINT @ 458, "GAME'S OVER"
```

Russian Roulette program:

```
5 FOR N = 1 TO 10
10 PRINT "CHOOSE YOUR CHAMBER(1-10)"
20 INPUT X
30 IF X = RND(10) THEN 100
40 SOUND 200, 1
50 PRINT "--CLICK--"
60 NEXT N
65 CLS
70 PRINT @ 230, "CONGRATULATIONS!!!"
80 PRINT @ 265, "YOU MANAGED"
90 PRINT @ 296, "TO STAY ALIVE"
95 END
100 FOR T = 133 TO 1 STEP -5
110 PRINT "BANG!!!!!"
120 SOUND T, 1
130 NEXT T
140 CLS
150 PRINT @ 230, "SORRY, YOU'RE DEAD"
160 SOUND 1, 50
170 PRINT @ 390, "NEXT VICTIM, PLEASE"
```

CHAPTER 10

Test Your Arithmetic Program

```
5 CLS
6 PRINT @ 230, "YOUR NAME";
8 INPUT N$
10 CLS
15 T = T + 1
20 X = RND(100)
30 Y = RND(100)
40 PRINT @ 228, "WHAT IS" X "+" Y;
45 INPUT A
50 IF A = X + Y THEN 82
60 PRINT @ 326, "THE ANSWER IS" X + Y
70 PRINT @ 385, "BETTER LUCK NEXT TIME," N$
80 GOTO 100
82 CLS(7)
83 FOR M = 1 TO 4
84 SOUND 175, 1
85 SOUND 200, 1
86 NEXT M
87 CLS
90 PRINT @ 232, "CORRECT," N$ "!!!"
95 C = C + 1
97 PRINT @ 299, "THAT IS"
98 PRINT @ 322, C "OUT OF" T "CORRECT ANSWERS"
99 PRINT @ 362, C/T*100 "% CORRECT"
100 PRINT @ 420, "PRESS <ENTER> WHEN READY"
102 PRINT @ 458, "FOR ANOTHER"
105 INPUT A$
110 GOTO 10
```


CHAPTER 11

Table of Squares

```
5 CLS
7 PRINT @ 38, "TABLE OF SQUARES"
8 PRINT
10 P = 2
20 FOR N = 2 TO 10
25 GOSUB 2000
30 PRINT N "*" N "=" E,
40 NEXT N
50 END
2000 REM FORMULA FOR RAISING A NUMBER TO
    A POWER
2010 E = 1
2020 FOR X = 1 TO P
2030 E = E * N
2040 NEXT X
2050 IF P = 0 THEN E = 1
2060 RETURN
```

CHAPTER 12

Editing a Sentence

```
10 PRINT "TYPE A SENTENCE :"; S$
15 INPUT S$
20 PRINT "TYPE A PHRASE TO DELETE"
23 INPUT D$
25 L = LEN(D$)
30 PRINT "TYPE A REPLACEMENT PHRASE"
35 INPUT R$
40 FOR X = 1 TO LEN(S$)
50 IF MID$(S$,X,L) = D$ THEN 100
60 NEXT X
70 PRINT D$ "— IS NOT IN YOUR SENTENCE"
80 GOTO 20
100 E = X - 1 + LEN(D$)
110 NS$ = LEFT$(S$,X-1) + R$ + RIGHT$(S$,LEN(S$)
    - E)
120 PRINT "NEW SENTENCE IS :'"
130 PRINT NS$
```

CHAPTER 13

Computer Typing Test

```
10 CLS
20 INPUT "PRESS <ENTER> WHEN READY TO TYPE
    THIS PHRASE"; E$
30 PRINT "NOW IS THE TIME FOR ALL GOOD MEN"
40 T = 1
50 A$ = INKEY$
60 IF A$ = "" THEN 100
70 PRINT A$;
80 B$ = B$ + A$
90 IF LEN(B$) = 32 THEN 120
100 T = T + 1
110 GOTO 50
120 S = T/74
130 M = S/60
140 R = 8/M
142 FOR X = 1 TO 32
144 IF MID$("NOW IS THE TIME FOR ALL GOOD
    MEN",X,1) <> MID$(B$,X,1) THEN E = E + 1
146 NEXT X
150 PRINT
160 PRINT "YOU TYPED AT—" R "—WDS/MIN"
170 PRINT "WITH" E "ERRORS"
```

SAMPLE PROGRAMS

SPEED READING

```

10 REM * SPEED READING *
20 CLS : PRINT @ 32, "HOW MANY WORDS PER MINUTE"
30 INPUT "DO YOU READ"; WPM
40 FOR X = 1 TO 23
50 REM (23 LINES OF TEXT)
60 READ A$ : PRINT @ 256, A$
70 FOR Y = 1 TO (360/WPM) * 460 : NEXT Y
80 REM Y LOOP SETS LINES/MINUTE
90 NEXT X : END
100 DATA SCARLETT OHARA WAS NOT BEAUTIFUL
110 DATA BUT MEN SELDOM REALIZED IT WHEN
120 DATA CAUGHT BY HER OWN CHARM AS THE
130 DATA TARLETON TWINS WERE. IN HER FACE
140 DATA WERE TOO SHARPLY BLENDED
150 DATA THE DELICATE FEATURES OF HER
160 DATA MOTHER, A COAST ARISTOCRAT OF
170 DATA FRENCH DESCENT, AND THE HEAVY
180 DATA ONES OF HER FLORID IRISH FATHER
190 DATA BUT IT WAS AN ARRESTING FACE,
200 DATA POINTED OF CHIN, SQUARE OF JAW
210 DATA HER EYES WERE PALE GREEN
220 DATA WITHOUT A TOUCH OF HAZEL,
230 DATA STARRED WITH BRISTLY BLACK
240 DATA LASHES AND SLIGHTLY TILTED
250 DATA THE ENDS, ABOVE THEM, HER THICK
260 DATA BLACK BROWS SLANTED UPWARDS,
270 DATA CUTTING A STARTLING OBLIQUE LINE
280 DATA IN HER MAGNOLIA-WHITE SKIN—THAT
290 DATA SKIN SO PRIZED BY SOUTHERN WOMEN
300 DATA AND SO CAREFULLY GUARDED WITH
310 DATA BONNETS, VEILS, AND MITTENS
320 DATA AGAINST HOT GEORGIA SUNS.

```

MUSIC COMPOSER

```

20 INPUT "LENGTH(1-10)"; M
25 M = M * 4
30 INPUT "TEMPO (1-4)"; T1
40 IF T1 = 4 THEN 60
50 T = T1 : GOTO 70
60 T = 8
70 FOR K = 1 TO M*8
80 GOSUB 1000
90 B = RND(3) * T
100 SOUND P, B
110 CLS(S)
120 NEXT K
130 IF RND(10) <= 8 THEN 150
140 SOUND 125, 16*T
145 END
150 SOUND 90, 16*T
160 END
1000 X = RND(100)
1010 IF X <= 20 AND X <= 25 THEN P = 90 : S = 1
1020 IF X > 20 AND X <= 25 THEN P = 108 : S = 2
1030 IF X > 25 AND X <= 40 THEN P = 125 : S = 3
1040 IF X > 40 AND X <= 55 THEN P = 133 : S = 4
1050 IF X > 55 AND X <= 75 THEN P = 147 : S = 5
1060 IF X > 75 AND X <= 85 THEN P = 159 : S = 6
1070 IF X > 85 AND X <= 95 THEN P = 176 : S = 7
1080 IF X > 95 THEN P = 58 : S = 8
1090 RETURN

```

READY, AIM, FIRE

```
10 CLS : PRINT
20 PRINT "HIT 'Z' KEY TO AIM LEFT"
30 PRINT "HIT 'Y' KEY TO AIM RIGHT"
40 PRINT "HIT SPACE BAR TO FIRE"
50 FOR K = 1 TO 3000 : NEXT K
60 CLS(0)
70 CA = 464
80 PRINT @ CA, "*" ;
90 PRINT @ 495, "*****" ;
100 PRINT @ 32, " "
110 F = 0
120 PRINT @ 32 + I * 4, " " ;
130 IF I > 6 THEN I = 0
140 I = I + RND(10)/10
150 PRINT @ 32 + I * 4, "---->" ;
160 IF F = 0 THEN 500
170 RESET(MX,MY)
180 MX = MX - MD
190 MY = MY - 8
200 IF MX < 1 THEN 110
210 IF MX > 63 THEN 110
220 IF MY > 2 THEN SET(MX,MY,5) : GOTO 120
230 IF ABS(I*8-MX) > 4 THEN 110
240 FOR J = 1 TO 6
250 PRINT @ 32+4*I, "*****" ;
260 FOR K = 1 TO 50
270 NEXT K
272 PRINT @ 32 + I * 4, " "
274 FOR K = 1 TO 50
276 NEXT K
280 NEXT J
290 FOR K = 200 TO 10 STEP -3
300 SOUND K, 1
310 NEXT K
320 FOR K = 1 TO 7 : CLS(K)
330 FOR KK = 1 TO 50 : NEXT KK
340 NEXT K
350 CLS(0)
```

```
360 GOTO 60
500 Y$ = INKEY$
510 IF F = 1 THEN END
520 IF Y$ <> "Z" THEN 550
530 IF CA < 462 THEN 120
540 PRINT @ CA, CHR$(160);
545 CA = CA - 1
547 GOTO 580
550 IF Y$ <> "Y" THEN 590
560 IF CA > 466 THEN 120
570 PRINT @ CA, CHR$(160);
575 CA = CA + 1
580 PRINT @ CA, "*" ; : GOTO 120
590 IF Y$ <> " " THEN 120
600 F = 1 : MD = 464 - CA : MY = 20
610 MX = 32 - 3 * MD : SET(MX,MY,3) : GOTO 120
620 END
```

KALEIDOSCOPE

```
10 CLS(0)
20 X = RND(32) - 1
30 Y = RND(16) - 1
40 Z = RND(9) - 1
50 GOSUB 90
60 GOTO 20
90 IF Z = 0 THEN 150
95 IF RND(7) = 3 THEN 150
100 SET(31-X, 16+Y, Z)
110 SET(31-X, 15-Y, Z)
120 SET(32+X, 16+Y, Z)
130 SET(32+X, 15-Y, Z)
140 RETURN
150 RESET(31-X, 16+Y)
160 RESET(31-X, 15-Y)
170 RESET(32+X, 16+Y)
180 RESET(32+X, 15-Y)
190 RETURN
```

ERROR MESSAGES

The shaded errors are the errors you are most likely to come across while going through this book. While you might get some of the non-shaded errors, they are usually caused by more advanced programming techniques.

/0 Division by zero. The Computer was asked to divide a number by 0, which is impossible.

AO Attempt to open a data file which is already open.

BS Bad subscript. The subscripts in an array are out of range. Use DIM to dimension the array. For example, if you have A(12) in your program, without a preceding DIM line which dimensions array A for 12 or more elements, you will get this error.

CN Can't continue. If you use the command CONT and you are at the END of the program, you will get this error.

DD Attempt to redimension an array. An array can only be dimensioned once. For example, you cannot have DIM A(12) and DIM A(50) in the same program.

DN Device number error. Only three devices may be used with OPEN, CLOSE, PRINT, or INPUT: 0, - 1, or - 2. If you use another number you will get this error.

DS Direct statement. There is a direct statement in the data file. This could be caused if you load a program with no line numbers.

FC Illegal Function Call. This happens when you use a parameter (number) with a BASIC word that is out of range. For example SOUND (260,260) or CLS(10) will cause this error. Also RIGHT\$(S\$,20), when there are only 10 characters in S\$, would cause it. Other examples are a negative subscript, such as A(- 1), or a USR call before the address has been POKEd in.

FD Bad file data. This error occurs when you PRINT data to a file, or INPUT data from the file, using the wrong type of variable for the corresponding data. For example, INPUT # - 1, A, when the data in the file is a string, causes this error.

FM Bad file mode. This error occurs when you attempt to INPUT data from a file OPEN for OUTPUT (O), or PRINT data into a file OPEN for INPUT (I). 0

ID Illegal direct statement. You can only use INPUT as a line in the program, not as a command line.

-
- IE** Input past end of file. Use EOF to check to see when you've reached the end of the file. When you have, CLOSE it.
 - IO** Input/Output error. Often this is caused by trying to input a program or a data file from a bad tape.
 - LS** String too long. A string may only be 255 characters.
 - NF** NEXT without FOR. NEXT is being used without a matching FOR statement. This error also occurs when you have the NEXT lines reversed in a nested loop.
 - NO** File not open. You cannot input or output data to a file until you have OPENed it.
 - OD** Out of data. A READ was executed with insufficient DATA for it to READ. A DATA statement may have been left out of the program.
 - OM** Out of memory. All available memory has been used or reserved.
 - OS** Out of string space. There is not enough space in memory to do your string operations. Use CLEAR at the beginning of your program to reserve more string space.
 - OV** Overflow. The number is too large for the Computer to handle.
 - RG** RETURN without GOSUB. A RETURN line is in your program with no matching GOSUB.
 - SN** Syntax error. This could result from a mis-spelled command, incorrect punctuation, open parenthesis, or an illegal character. Type the program line or command over.
 - ST** String formula too complex. A string operation was too complex to handle. Break up the operation into shorter steps.
 - TM** Type Mismatch. This occurs when you try to assign numeric data to a string variable (A\$ = 3) or string data to a numeric variable (A = "DATA").
 - UL** Undefined line. You have a GOTO, GOSUB, or other branching line in the program asking the computer to go to an unexisting line number.

BASIC SUMMARY

This appendix contains the BASIC words, keyboard characters, symbols, and operators that we've explained in this book. For a complete listing of them, see the COLOR BASIC Quick Reference Card.


BASIC WORDS

WORD	PURPOSE	EXAMPLES	CHAPTER DISCUSSED
CLOAD	Loads the first program from cassette tape. You may specify the name of the program.	CLOAD CLOAD "PROGRAM"	8
CLS	Clears the screen to green, or to the color code you specify. See Appendix B for a list of the color codes.	CLS CLS(2)	1
CSAVE	Saves a program on cassette tape. You may use a program name with up to 8 letters.	CSAVE CSAVE "PROGRAM"	8
DATA	Stores data in your program. Use READ to assign this data to variables.	DATA 5, 3, PEARS DATA PAPER, PEN	10
END	Ends your program.	END	6
FOR. TO STEP/ NEXT	Creates a loop in your program which the Computer must repeat from the first number to the last number you specify. You may use STEP to specify how much to increment the number each time through the loop. If you omit STEP, 1 is the increment.	FOR X = 2 TO 5 /NEXT X FOR A=1 TO 10 STEP 5 /NEXT A FOR M = 30 TO 15 STEP -5 NEXT M	4&5
GOSUB	Sends the Computer to the subroutine beginning at the line number you specify.	GOSUB 500 GOSUB 5000	11
GOTO	Sends the Computer to the line number you specify.	GOTO 300	3

IF/THEN	Tests the relationship. If it is true, the Computer executes the instruction following THEN.	IF A=5 THEN 300 IF B\$="YES" THEN PRINT "XYZ"	6
INKEY\$	Strobes the keyboard and returns the key or non-key being pressed.	A\$=INKEY\$	13
INPUT	Causes the Computer to stop and await input from the keyboard.	INPUT X\$ INPUT "NAME"; N\$	3&11
INT	Converts a number to an integer.	X=INT(5.2)	10
JOYSTK	Returns the horizontal or vertical coordinate of the left or right joystick: 0—horizontal, left joystick 1—vertical, left joystick 2—horizontal, right joystick 3—vertical, right joystick	M=JOYSTK(0) H=JOYSTK(2)	9
LIST	Lists the entire program, or the lines in the program you specify.	LIST LIST 50-85 LIST 30 LIST -30 LIST 30-	3
NEW	Erases everything in memory.	NEW	3
PEEK	Returns the contents in the memory location you specify.	A=PEEK(32076)	9
PRINT	Prints the message you specify on the video screen.	PRINT "HI" PRINT A\$ PRINT A	1
PRINT @	Prints your message at the screen position you specify. See Appendix C for the screen positions.	PRINT "HI", 256 PRINT A\$, 331	7
READ	Reads the next item in the DATA line and assigns it to the variable you specify.	READ A\$ READ C, B	10

REM	Allows you to insert a comment in your program. The Computer ignores everything on a REM line.	REM THIS IS IGNORED	11
RESET	Erases the dot SET on the screen location you specify. See Appendix D for the screen locations.	RESET(14,15)	9
RESTORE	Sets the Computer's pointer back to the first item on the DATA lines.	RESTORE	10
RETURN	Returns the Computer from the subroutine to the BASIC word following GOSUB.	RETURN	11
RND	Returns a random integer between 1 and the number you specify.	A = RND(10)	7
RUN	Executes a program.	RUN	3
SET	Sets a dot at the screen location you specify, using the color you specify. See Appendix D for the screen locations and Appendix B for the color codes.	SET(14,13,3)	9
SKIPF	Skips to the end of the next program on cassette tape, or to the end of the program you specify.	SKIPF SKIPF "PROGRAM"	8
SOUND	Sounds the tone you specify for the duration you specify. Both the tone and the duration may be a number between 1 and 255.	SOUND 128, 3	1 & 5
VAL	Converts a string to a number.	A = VAL(B\$)	13

KEYBOARD CHARACTERS

CHARACTER	PURPOSE	CHAPTER DISCUSSED
	Backspaces the cursor (the blinking light)	1
ENTER	Tells Computer you've reached the end of your program line or command line.	1&3
BREAK	Stops execution of your program.	3
SHIFT @	Pauses execution of your program. Press any key to continue.	4

BASIC OPERATORS

OPERATOR	PURPOSE	CHAPTER DISCUSSED
+	Combines strings	12
+	Addition	1
-	Subtraction	1
*	Multiplication	1
/	Division	1
=	Equals	1
>	Greater Than	9
>= or =>	Greater than or equal to	9
<= or =<	Less than or equal to	9
<	Less than	9
<> or ><	Not equal to	13

BASIC SYMBOLS

SYMBOL	EXPLANATION	CHAPTER DISCUSSED
" "	Indicates that the data in quotes is a constant.	1
:	Separates program "statements" on the same line.	11
()	Tells the Computer to perform the operation in the inside parenthesis first.	11

NOTES:



RADIO SHACK SOFTWARE LICENSE

The following are the terms and conditions of the Radio Shack Software License for copies of Radio Shack software either purchased by the customer, or received with or as part of hardware purchased by customer:


- A. Radio Shack grants to *CUSTOMER* a personal, non-exclusive, paid up license to use the Radio Shack computer software programs received. Title to the media on which the software is recorded (cassette and/or disk) or stored (ROM) is transferred to the *CUSTOMER*, but not title to the software.
- B. In consideration for this license, *CUSTOMER* shall not reproduce copies of such software programs except to produce the number of copies required for personal use by *CUSTOMER* (if the software allows a backup copy to be made), and to include Radio Shack's copyright notice on all copies of programs reproduced in whole or in part.
- C. *CUSTOMER* may resell Radio Shack's system and applications software (modified or not, in whole or in part), provided *CUSTOMER* has purchased one copy of the software for each one resold. The provisions of this Software License (paragraphs A, B, and C) shall also be applicable to third parties purchasing such software from *CUSTOMER*.

IMPORTANT NOTE

All Radio Shack computer programs are licensed on an "as is" basis without warranty.

Radio Shack shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by computer equipment or programs sold by Radio Shack, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer or computer programs.

Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

RADIO SHACK  A DIVISION OF TANDY CORPORATION

U.S.A.
FORT WORTH, TEXAS 76102

CANADA
BARRIE, ONTARIO, L4M4W5

TANDY CORPORATION

AUSTRALIA
*280-316 VICTORIA ROAD
RYDAMERE, N.S.W. 2116*

BELGIUM
*PARC INDUSTRIEL NANINNE
5140 NANINNE*

UNITED KINGDOM
*BILSTON ROAD, WEDNESBURY
WEST MIDLANDS WS10 7JN*