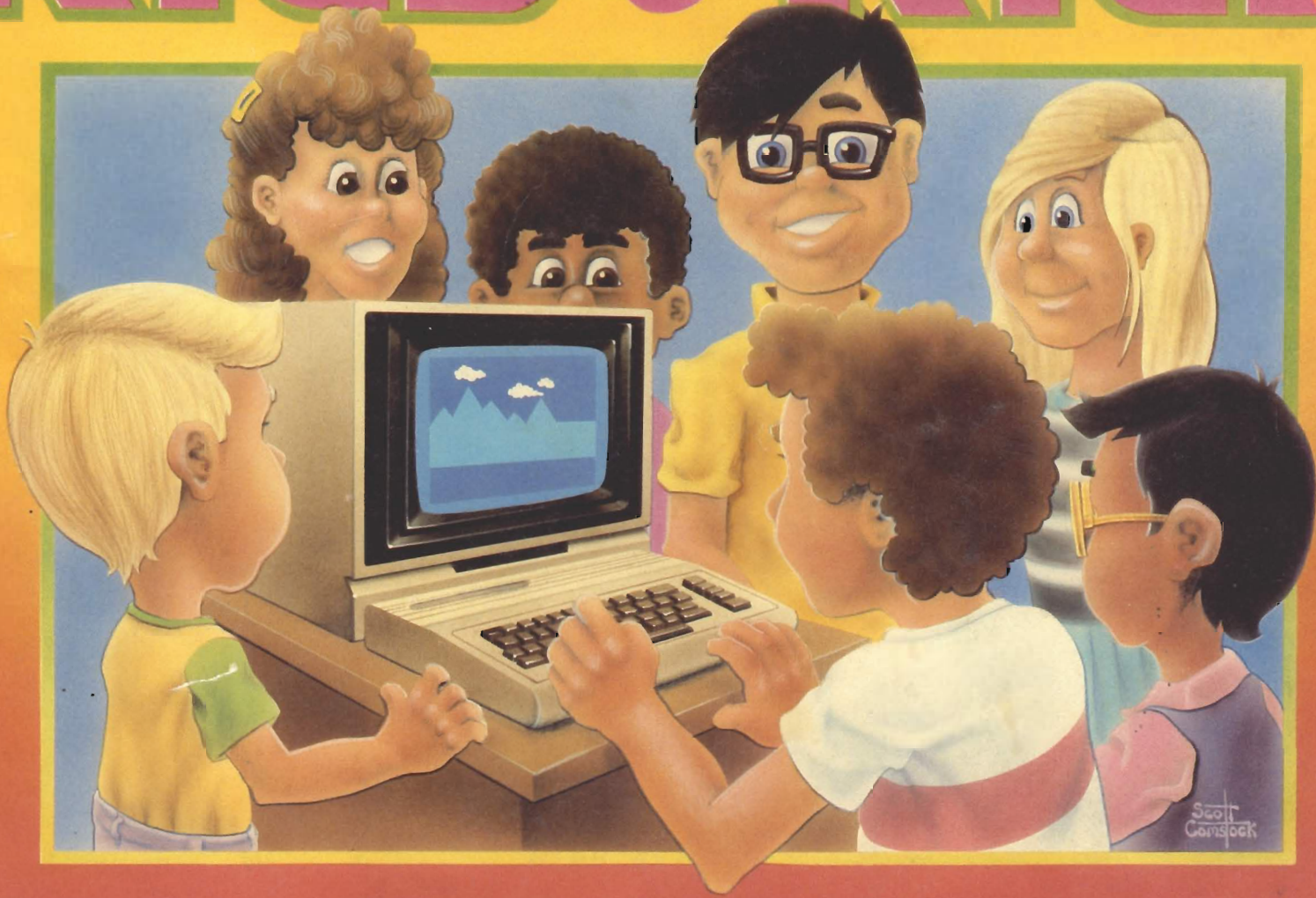
Reston Publishing Company Inc.

Kids to Kids

$9.95



ON THE COMMODORE 64

by
Sam Edge,
Billy Sanders
and
Kris Hauser

Kids to Kids

on the Commodore 64

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Kids to Kids

on the Commodore 64

by

Sam Edge, Billy Sanders and Kris Hauser

Illustrated by
Martin Cannon

Editor's Introduction
by
William B. Sanders, Ph.D.

DATAMOST INC.
The most out of our minds.™
20660 Nordhoff Street
Chatsworth, CA 91311-6152
(818) 709-1202

First Printing January, 1984
Second Printing June, 1984

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

ISBN 0-8359-3679-1

Printed in U.S.A.

Table of Contents

EDITOR'S INTRODUCTION

When our computer came into our house, our older son was the first to learn how to run and program it. I explained the various parts and commands, and got him started programming with a promise of his own computer once he mastered the essential commands and statements. Like all nine years olds, his main interest was in arcade games, and slowly he began wondering how he could make his own game. After some extensive work with the computer, he developed a rudimentary adventure game (motivated by an interest in making the adventure rather than the pure joy I always feel when I master the puzzle of programming).

Shortly after my oldest son became adept at working and programming his computer, his younger brother wanted to learn as well. I mumbled encouragement and went back to a matter engrossing my attention at the time deciding (in the back of my mind) to get around to helping him learn how to use the computer soon. The next day I noticed that he was already using it without my vaunted assistance. I asked him if his older brother knew about his using the computer, and he explained that he had taught him how to work it. Later on, I found that the older brother could communicate with his younger brother in a way that was clear, concise, and to the point.

Kids have always communicated better with one another than with adults, and this was simply another instance of that phenomenon. Also, kids seem to learn quickly from one another. Adults have ideas about what really interests kids, and sometimes they are right. However, kids seem to have a sixth sense about their mutual interests. And kids *never* want to be told what is good for them by adults. Since the passing on of valued knowledge from kids to kids has been going on

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

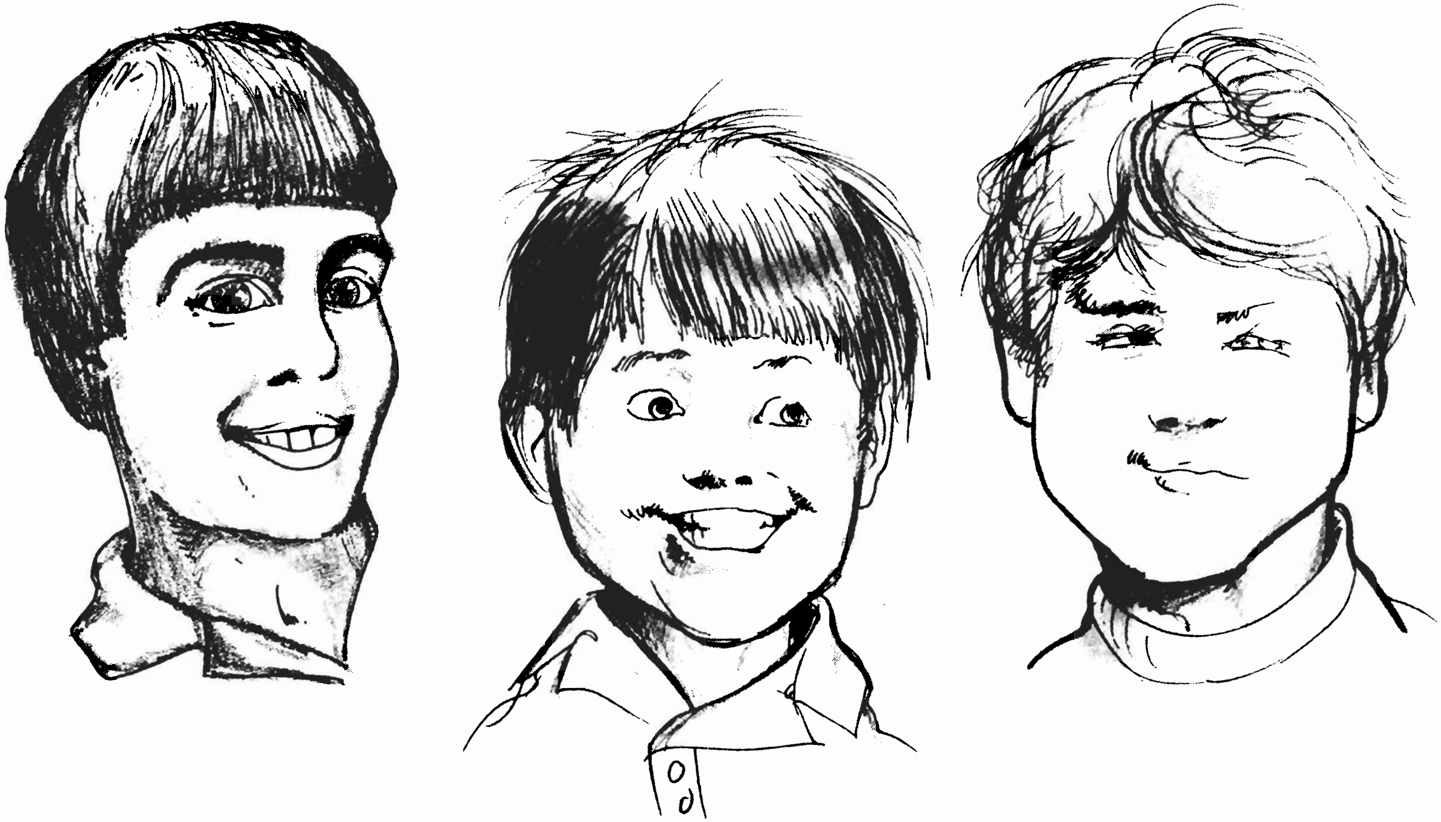
since the first cave kid told another about the best tree to climb, why not apply that process to learning about computers? After all, if one kid knows the "neatest" (if they still use that term) thing to do with a computer and how to program it, why not let him or her explain the whole process to another kid?

This book was born with the idea that kids can teach other kids better than anyone else. Adult intervention was given to aid and assist where required and to do all of the grimy work of formatting, correcting, and encouraging. To my surprise and delight, I was not overburdened with editorial tasks. The three young authors dived into their project with boundless enthusiasm and ideas Their energy level was always high, and discussions and phone calls concerning their opus occurred at all times of day and night.

The level of the book is introductory, but the authors did include some more advanced concepts near the end. This was done so that they could include some games and other programs they felt kids would enjoy. Otherwise, most of the material is designed to help kids get started using and programming their Commodore 64.

The three authors, Sam Edge, Billy Sanders, and Kris Hauser are 17, 11, and 14 years old respectively. They wanted me to mention several people who helped them. Sam would like to thank all his friends for their help, and also he is very grateful to his mother for all of her understanding support. Billy wants to thank Eric Goez for originally suggesting he create the book, his brother David and his parents. Kris is grateful to his friend Kenneth, his sister Jenny, and his mom for the encouragement and help they gave. Finally, as editor, I would like to thank Sam, Billy, and Kris for doing such an excellent job.

William B. Sanders, Ph.D.
Series Editor

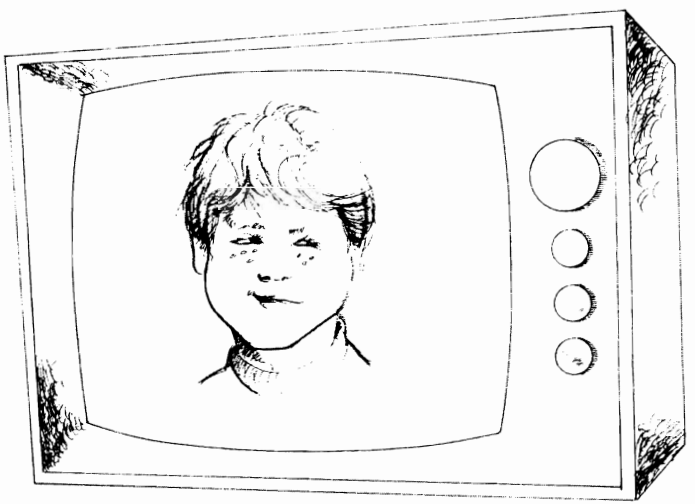


KIDS TO KIDS TO KIDS TO

1

Getting Set

Hi, we're Sam Edge, Billy Sanders and Kris Hauser. We are kids like you (17, 11 and 14 years old.) That's why this book is called KIDS TO KIDS ON THE COMMODORE 64. It's a book for kids from the point of view of three kids. We hope this book will help you understand how to use your Commodore 64.



Let's get started by connecting your computer to your TV set. You do that by taking a little black box called a switch box that came with your Commodore 64 and placing the switch box wires under the screws where it says VHF on your TV. Then just tighten the screws.

Next take out the long black cord from the box you got your computer in, and hook one end of the cord into the back of your computer in the port with a round hole in it. The other end of the

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

cord goes into the connecting hole on the switch box. Now you have made the connection between your computer and your TV set. Depending on where you live, either Channel 3 or 4 will be the best one to use for your computer. Right next to where you plugged in the cord that goes to the TV is a switch. If the switch is to the left it is set for Channel 3, and if to the right, for Channel 4.

Now that you have hooked up the TV to the computer, plug the computer into the wall. You do that by taking the power cord attached to the Power Supply box and plugging it into the socket labeled POWER on the right side of your computer. Take the other cord leading from the Power Supply box and plug it into the nearest wall outlet.

To turn it on you simply flip a little switch on the right side of the computer to the ON position. When you have done that, on the top of the screen the following should appear:

```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

This will be in the center of the screen. When you are done with that, play with it for a while. Press some different keys to see what happens. Some of the keys are regular and some are irregular. You may know the keys from A-Z pretty well. Also, you know the number keys and their shift statements, like !"#$%&'(). But what about the up, down, left, and right CRSR arrow keys and the RETURN, CTRL, RUN/STOP, CLR/HOME, INST/DEL, and RESTORE keys? Here is what they do. The up and down CRSR arrow keys move the cursor up and down the screen. The same thing happens with the left and right arrow keys. By pressing the keys, you can move the cursor down and to the right. If you hold down the SHIFT key and press the CRSR keys, the cursor will move up or to the left. The little white square moving around the screen is the cursor. (That's what CRSR stands for.) Play with these keys for a while so you can see how to use them.

CTRL means CONTROL. It is used with other keys to do things like change colors. Don't worry about it now, but later on we will use it when we talk about graphics. CLR/HOME is used to put

the cursor in the HOME position on your screen (in the upper left-hand corner). Press it and see where the cursor goes. Now press the SHIFT key and the CLR/HOME key and the screen will clear and put the cursor in the upper left-hand corner. The INST/DEL key is used to fix mistakes in your programs. When not using the SHIFT key, it DELetes the character to the left of the cursor. When SHIFTed, the key INSerts a space. Write some stuff on your screen, move the cursor around with the CRSR keys and see if you can delete characters and insert spaces.

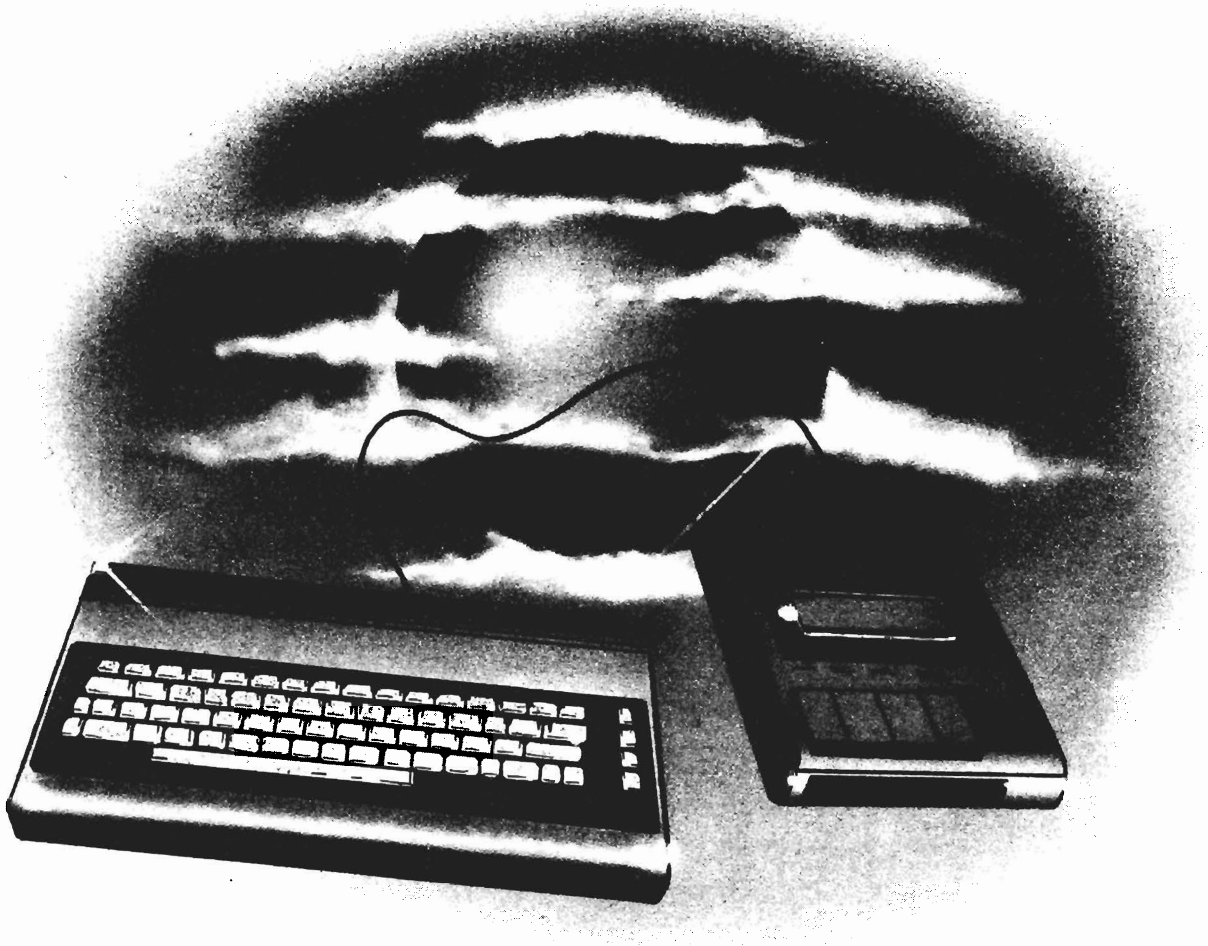
Press the RETURN key. That will end a program line and make the cursor jump a line. You will be using the RETURN key a lot! Press the RUN/STOP and RESTORE keys at the same time, and your screen will clear and READY. appears at the top of your screen. These are the "panic buttons," If your screen ever freezes up on you, press these keys to "reset" everything. To stop a program, just press the RUN/STOP key. Key in the following program to see how it works. (If something is between the curly brackets, it means to press that key. In line 10, put in quotation marks and press the SHIFT key and the

CLR/HOME key at the same time. A little heart will appear on the screen. <RETURN> means press the RETURN key.)

```
10 PRINT "{SHIFT CLR/HOME}"
<RETURN>
20 PRINT "COMPUTER" <RETURN>
30 GOTO 20 <RETURN>
RUN <RETURN>
<PRESS RUN/STOP>
```

When you press the RUN/STOP key, your computer will respond BREAK IN 20, indicating that the program was at line 20 when you pressed the RUN/STOP key.

Now you should have the computer hooked up to the TV and you should know what most of the keys mean. In the next chapter you will learn how to use the cassette tape and disk drive.



KIDS TO KIDS TO KIDS TO

2

How To Use The Cassette Recorder and Disk Drive

In this chapter you will be learning how to use the cassette recorder and the disk drive. If you want to use your cassette recorder, you have to hook it up first. Turn off your computer and TV. In fact, you should unplug both to be certain. The cord for the recorder is long and gray. It has a long flat end which plugs into the back of your computer right behind the 4, 5, and 6 number keys. *Note: We would strongly recommend that you buy the cassette recorders made by Commodore since not every cassette recorder will work with your computer.* Now plug in the computer, TV, and recorder and turn them all on.

Now that you have hooked it up and turned it on, here's how to use it. (You will need a tape

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

with programs on it. If you don't have any, get a blank tape and skip to the next section which explains how to save a program to tape.) The first word we will introduce to you is LOAD. LOAD means LOAD a program. How it works is simple. Put your tape with some programs on it into your recorder. You type LOAD, then quotation marks, the program's name, and then ending quotation marks.

LOAD "PROGRAM" <-Use the actual name of the program

Now press PLAY on your cassette recorder and the Commodore key (it's the one with the Commodore logo on it) on your computer and press RETURN. *Note: Make sure that you have fast forwarded the tape or rewound it to the program that you want. If this is not done, it will go searching for the program until it has found it or gives up. You have to watch the cassette counter. Those are the numbers on the cassette recorder that tell where the tape is. By writing down the numbers of the cassette counter, you can tell where your program begins. Then you don't have to wait all day for the recorder to find*

your program. The recorder will start moving the tape forward. When it finds the program it will say LOADING, and when it is finished it says READY. Type RUN and press RETURN, and you will have your program working. If you get an error when you type LOAD, make sure you have spelled the name of the program correctly, rewind the tape, and try again.

Sometimes the computer will give you an error because the program is not found or the cassette is not hooked up right. If that happens, make sure your cassette recorder is connected where it should be.

Saving Programs On Tape

So far you have learned how to load programs and search for them. How about learning how to save programs? Saving programs is simple. When you are finished with a program and you want to save it, all you do is type SAVE, quotation marks ("), a program name, and then ending quotation marks ("). See how easy it is! Enter the example program and use it for practice. At

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

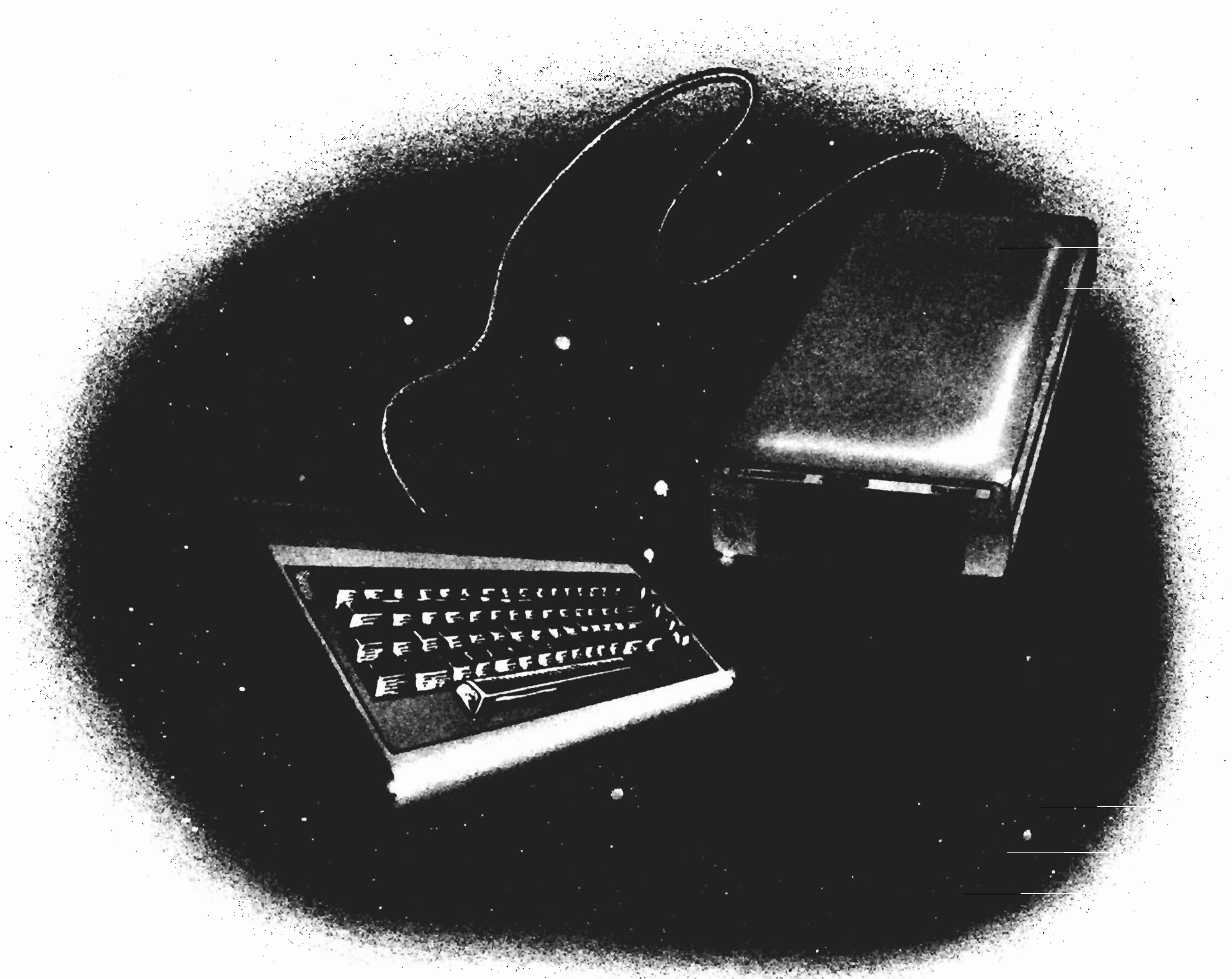
the end of every line press RETURN. (If this is your only program, go back to the section on LOAD and practice loading the program.)

Disk Drive

If you do not have a disk drive, you can skip to the next section. To use your Disk Drive, you first have to hook it up. Be sure to unplug your computer and disk drive *first*. You simply attach one end of the disk cable to the serial socket in back of the computer next to the video port. Connect the other end of the cable to the disk drive above the fuse. Now plug the power cord into the three-pronged socket between the ON/OFF switch and fuse on the disk drive and then plug the power cord into a wall outlet. Once you have finished that, find a little switch in the back of the disk drive and turn it on. Now that you have your disk drive all hooked up, you will learn how to use it.

```
1Ø PRINT "{SHIFT CLR/HOME}"
2Ø PRINT "SAVE ME"
3Ø END
```

SAVE "THIS PROGRAM" <-Use your program's real name



Once you have connected your drive, you have to format a disk for it to work. You format a disk by typing

```
OPEN15,8,15 <RETURN>
PRINT #15,"N0:DISKNAME,20"
<RETURN>
```

The red light on your drive will go on and it will start running. After the light on the drive goes out, your disk is formatted. Next type in:

```
PRINT#15,"I0" <RETURN>
```

Now your disk is all ready to use. Once you have formatted a disk, never do it again since you will erase all the programs on your disk if you do! However, each time you use your disk, you should initialize it with the PRINT#15,"I0" sequence. Now, key in

```
LOAD "$",8 <RETURN>
```

When the READY. prompt appears, type in:

```
LIST <RETURN>
```

Since you have just formatted a disk, when you LIST its contents, no programs will be there. For the sake of example, let's say there are some programs on that disk. This is what your screen would show:

```
LOAD "$",8
SEARCHING FOR $
LOADING
READY
LIST

Ø "DISKNAME          "          2Ø
3     "ZAP"                     PRG
2     "WSS GAME"                PRG
4     "SAMS GAME"               PRG
2     "KRIS FILE"               SEQ
READY.
```

You might be wondering what PRG and SEQ mean. Well, they have simple meanings. PRG means PRoGram. When you want to LOAD a program that says PRG beside it, you simply type LOAD, quotation marks, what the program's name is, and then ending quotation marks, a comma, and the number 8. When the

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

red light on the disk drive goes off and it says READY. below where you typed LOAD, type RUN, and you will have your program. SEQ means SEQuential file, and you need special programs to load a SEQ file. We will be using only PRG files in this book, but when you get more advanced you can get a book that will tell you how to use sequential files.

Now that you have learned how to LOAD programs, how about learning how to save them? Saving is simple. All you type is SAVE, quotation marks, your program's name, and then ending quotation marks and comma 8 (,8). The red light on your disk drive should go on, and when it goes off your program will be SAVEd on that disk. Whenever you type LOAD "$",8 and LIST, the program that you saved will show up on it. By the way, LOAD "$",8 stands for LOAD the directory of the disk.

In this chapter you have learned how to SAVE and LOAD on both the cassette recorder and the disk drive. These commands deal with I/O (Input/Ouput) to external devices. In the next chapter you will be learning how to use PRINT and math statements.

3

How To Use PRINT And Math Statements

In this chapter you will be learning how to use PRINT statements, first with words and then with math statements. PRINT statements are very easy to use, and they are also the most important statement in programming. When you are using the PRINT statement, you must always have quotation marks after it if you are not using variables. (You will learn about variables in the next chapter.) Look at the sample below to get a view of what a PRINT statement looks like.

```
10 PRINT "HI MOM AND DAD"
20 PRINT "HOW IS EVERYTHING GOING?"
30 PRINT "EVERYTHING IS FINE OVER
   HERE, SON."
```

When you are finished typing the program, type RUN, and on your TV screen it will show

```
HI MOM AND DAD
HOW IS EVERYTHING GOING?
EVERYTHING IS FINE OVER HERE, SON.
```

The numbers 10, 20, 30 stand for the line numbers. Every time you want to make a program you must type line numbers. Line numbers can range from 0 to 63999. Usually you start a program at line 10 and skip ten for each line. This is to allow for insertions and/or corrections. For example, if you number your program so that there is no gap between lines (1,2,3,4 . . .), and you want to add a line between 1 and 2, you will have to retype the entire program (except line 1). If your program is numbered by tens, like the above program, an additional line (say line 15) is put in its proper numerical order.

Enough with line numbers, let's get back to PRINT statements. A shortcut to writing PRINT is to simply type a question mark (?). Type the above sample program, but instead of typing PRINT, type a question mark. When you RUN the program, it should do exactly what the previous program did.

To get used to using PRINT statements, type in the next program. We will use the question mark this time. We will also introduce a single key statement. When you PRINT the SHIFT CLR/HOME key, it is like printing any other BASIC statement. Try putting in the following:

```
PRINT "{SHIFT CLR/HOME}"
<RETURN>
```

{SHIFT CLR/HOME} means press the SHIFT and the CLR/HOME keys together. You will see a little heart between the quotation marks. As soon as you press RETURN, your screen will go blank. That's because when SHIFT CLR/HOME is pressed (between quotation marks and preceded by a PRINT statement), it waits until the PRINT statement is executed to do what the keypress is supposed to do. Just pressing SHIFT CLR/HOME clears the screen immediately. To do it in a program, just put a PRINT statement in front of it, along with quotation marks. When we get to keyboard graphics, we use the keys a lot to PRINT the single key figures.

We also put in PRINT without PRINTing anything. This will put in a space between lines for

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


EVERYBODY CLEAR OUT!
I WANT THIS SCREEN EMPTY!

you. (Notice that we used both PRINT statements and question marks to create PRINT.)

```
10 PRINT "{SHIFT CLR/HOME}"
20 ? "WHAT MONSTER WOULD YOU
   LIKE TO BE?"
30 PRINT
40 ? "1. DRAGON"
50 ? "2. OGRE"
60 ? "3. WEREWOLF"
70 PRINT
80 ? "YOU GUYS SCARE ME!"
```

When you RUN the program, it will look like this,

```
WHAT MONSTER WOULD YOU LIKE TO BE?

1. DRAGON
2. OGRE
3. WEREWOLF

YOU GUYS SCARE ME!

READY.
```

Before we go on, we will learn another new statement, LIST. After RUNning your program,

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

type in the word LIST. This will LIST your program, line-by-line. (Remember to press RETURN after an immediate mode command and after typing in a program line. Words like RUN and LIST are immediate commands.) There's a surprise! The question marks have been changed to PRINT. It should look like this on your screen:

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "WHAT MONSTER WOULD
   YOU LIKE TO BE?"
30 PRINT
40 PRINT "1. DRAGON"
50 PRINT "2. OGRE"
60 PRINT "3. WEREWOLF"
70 PRINT
80 PRINT "YOU GUYS SCARE ME!"
```

For practice, try saving the program to tape and disk. We will call the program MONSTER.

<u>TAPE</u>

SAVE "MONSTER" <RETURN>

<u>DISK</u>

SAVE "MONSTER",8 <RETURN>

Now that you have learned how to PRINT letters, how about learning how to do arithmetic? Arithmetic is simple. All it involves is adding, subtracting, multiplying, and dividing. First let's start with adding. Adding is the first thing a kid should know in arithmetic. Look at the example below to see how to do an addition statement.

```
PRINT 2+3
```

When you press RETURN your screen will look like this:

```
PRINT 2+3
5
READY.
```

The PRINT statement tells the computer to PRINT the sum of 2+3 on the screen. If you put quotation marks around the 2+3, the computer would not PRINT the answer, but instead it would look like this:

```
PRINT "2+3"
2+3
READY.
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

So remember, if you put quotation marks around something, it will PRINT whatever is inside them. If there are no quotation marks around numbers, it will PRINT the numbers or their arithmetic results.

Another thing a kid should know is how to subtract. On your computer, subtracting is just like adding except instead of typing a plus sign you would type a minus sign. For instance, look at the sample to see what a subtraction problem might look like.

```
PRINT 23-18
5
READY.
```

That was easy, but don't use your computer to get the answers to your homework! You will never learn how to do math in your head.

Multiplying works the same way as addition and subtraction. Try typing this on your screen:

```
PRINT 6 * 7
42
READY.
```

When you are using your computer and you want to multiply, use the little star (called an asterisk) for the multiplication sign. You wouldn't use the X because to the computer it just means the letter X. Division is the same as all the others except it uses the slash mark (/). Type the following problem:

```
PRINT 4/2 <RETURN>
```

Right under it, it should say 2. For some division problems where you get fractions, your computer will print up to eight decimal places. When you get to junior high or high school, you will need all those decimal places. The next program gives examples of all the different math functions.

```
10 "{SHIFT CLR/HOME}"
20 PRINT "ADDITION PROBLEM: 28 + 49"
30 PRINT "THE ANSWER IS "; 28+49
40 PRINT
50 PRINT "SUBTRACTION PROBLEM:
   83 - 47"
60 PRINT "THE ANSWER IS "; 83-47
70 PRINT
80 PRINT "MULTIPLICATION PROBLEM:
   19 * 51"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
90 PRINT "THE ANSWER IS "; 19*51
100 PRINT
110 PRINT "DIVISION PROBLEM: 73 / 14"
120 PRINT "THE ANSWER IS "; 73/14
130 END
```

Notice that we were able to PRINT both the message THE ANSWER IS *and* the math problem on the same line. The single PRINT statement took care of printing both. We used the semicolon (;) to put the two together on a single line.

In this chapter you have learned about PRINT statements and how to do adding, subtracting, multiplying, and dividing. You also learned how to clearn the screen with SHIFT CLR/HOME, LIST a program, and use line numbers. In the next chapter you will learn how to use variables.

4

How To Use Variables

Numeric Variables

There are two kinds of variables we will show you in this chapter. The first one is called a NUMERIC variable. A numeric variable is like a slot where you keep a number. You name the slot with one or two letters and the name works just like a number. You can change the number in the slot and the letters that represent the slot will now hold the new number. Look at the example to see what a numeric variable program might look like.


```
10 PRINT "{SHIFT CLR/HOME}"
20 B=25
30 C=40
40 PRINT B + C
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

When you are done typing the program, type RUN. Below where you typed the program, it should say 65. The reason it should say 65 is because 25 + 40 equals 65. The B in line 20 stands for 25. That's why it says B=25. On the next line it says C=4Ø, so in the computer's memory the variable C stands for 4Ø. The line below that, line 40, instructs the computer to add B and C. It's the same as saying, "add 25 and 40". That is why it printed 65 below the program that we just showed you. Numeric variables are fun and easy to use. Here are two more programs you can use to see how numeric variables work. Notice how we PRINT both messages and numbers (variables) in the first program. If we use the semicolon (;), whatever we PRINT will be next to the first item we PRINTed. In the second program, before you RUN it, see if you can guess what the computer will PRINT to the screen. We also introduced a new statement, END, which tells the computer that it should stop. You do not need the END statement at the very end of a program, but in some cases we will look at later, you do need it.

```
1Ø PRINT "{SHIFT CLR/HOME}"
2Ø A = 1Ø
```

```
30 A1 = A * A
40 B = 20
50 B1 = B * B
60 PRINT "A = ";A
70 PRINT "A1 = ";A1
80 PRINT "B = ";B
90 PRINT "B1 = ";B1
100 PRINT "A1 TIMES B1 =";A1 * B1
110 END
RUN <RETURN>
```

```
10 PRINT "{SHIFT CLR/HOME}"
20 X = 50
30 Y = 60
40 Z = 70
50 A = X + Y
60 B = X + Z
70 C = Y + Z
80 PRINT A
90 PRINT B
100 PRINT C
110 END
RUN <RETURN>
```

Write some other programs of your own that use numeric variables.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

String Variables

Another kind of variable we are going to show you in this chapter is called a STRING variable. A string variable is like a numeric variable except it has a dollar sign on the end of it. String variables store character strings in slots like numeric variables store numbers. A string is any message you put in quotation marks. For example, if you say, A$ = "I'M A COMPUTER STAR", the message would be stored in the slot called A$. When you PRINT A$, your computer prints "I'M A COMPUTER STAR".

```
A$ = "I'M A COMPUTER STAR" <RETURN>
PRINT A$ <RETURN>
I'M A COMPUTER STAR
```

String variables take something really big and change it into something small and easy to print. Look at the sample to see what a string variable might look like.

```
10 PRINT "{SHIFT CLR/HOME}"
20 A$ = "<YOUR NAME>"
30 PRINT "HI ";A$
40 END
RUN <RETURN>
```

D-DOG

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

When you store your name in the slot called A$ in line 20, it will PRINT your name when you PRINT A$ in line 30. We added "HI" and a semicolon so that your computer would greet you. You can mix string and numeric variables in the same program if you want. Look at the next program.

```
10 PRINT "{SHIFT CLR/HOME}"
20 AG = <YOUR AGE>
30 N$ = "<YOUR NAME>"
40 PRINT N$; " IS ";AG; "YEARS OLD"
50 END
RUN <RETURN>
```

INPUT Statement

The last thing we are going to show you in this chapter is how to INPUT variables. Again, that is very simple. Instead of using a variable to equal something such as:

A$ = "AIRPLANE"

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

we enter the name of the string or the value of the numeric variable after we RUN the program.

```
10 INPUT A$
```

Look at the sample to see what a program with an INPUTed variable might look like.

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "WHAT IS YOUR NAME?"
30 INPUT N$
40 PRINT "HOW OLD ARE YOU?"
50 INPUT AG
60 PRINT "HI ";N$
70 PRINT "YOU ARE " ;AG; " YEARS OLD"
80 END
```

You saw another program similar to this one in this book, but by using the INPUT statement we were able to enter any name and age we wanted. INPUT can change the value of the variables when we RUN the program. The next program shows how we INPUT and PRINT two different strings for the same string variable.

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "RETURN WORD NUMBER ONE"
30 INPUT A$
40 PRINT A$
50 PRINT "RETURN WORD NUMBER TWO"
60 INPUT A$
70 PRINT A$
```

The first time we INPUT A$, it PRINTed the first word. The second time we INPUT A$ it PRINTed the second word. That shows you that you can change the contents of a variable while a program is being RUN. Change the program so that you can add two more words. To do that, just add some more lines to the program, beginning with line 80.

You can now see how important variables are. They are very useful and very helpful. Well, guess what? This is the end of the chapter. You have learned how to use two kinds of variables, numeric and string variables. You have also learned how to INPUT variables. In the next chapter you will be learning how to use different kinds of loops using GOTO statements and FOR/NEXT loops.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO

5

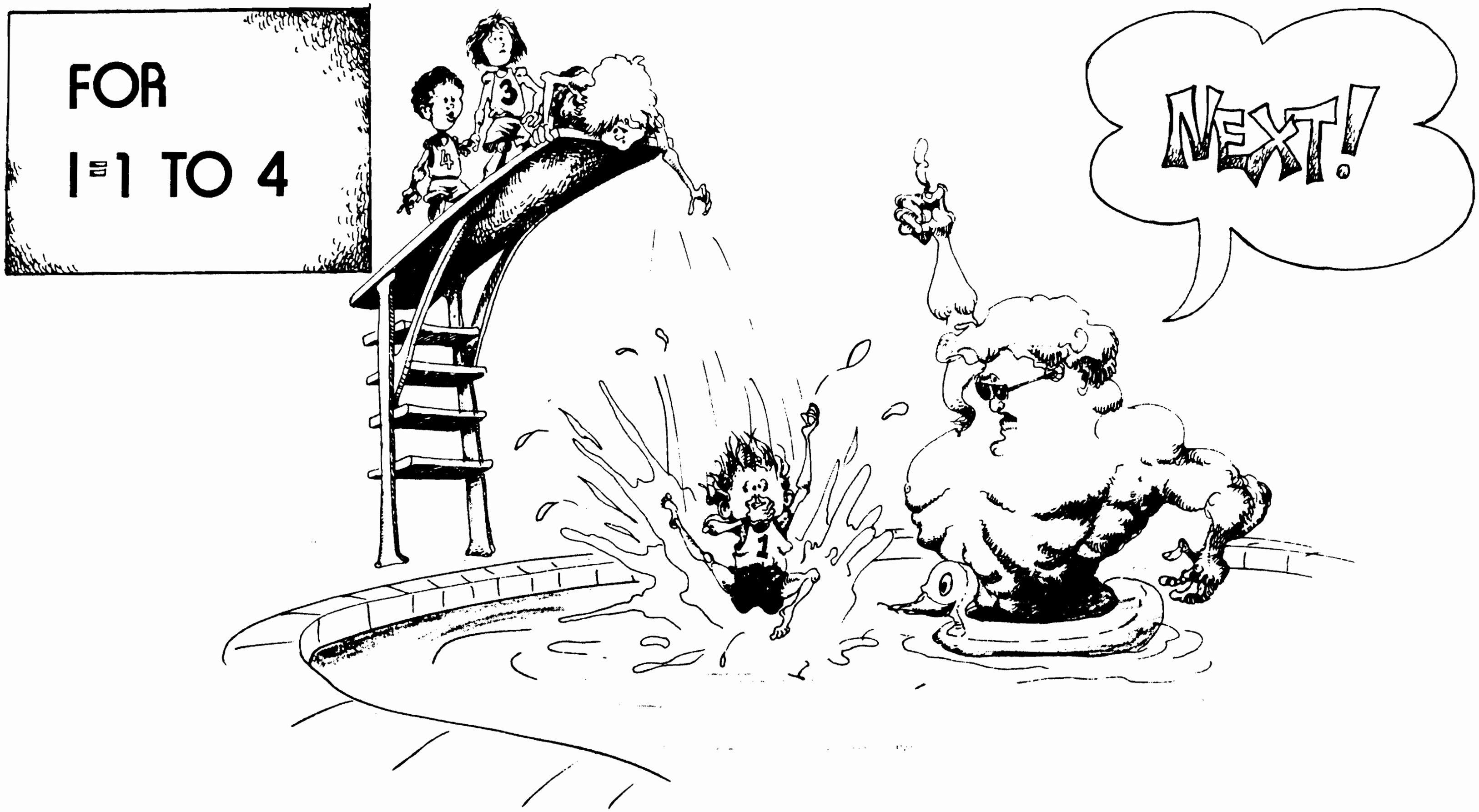
Using Loops

There are two kinds of loops we will be showing you in this chapter. They are called the FOR/NEXT loop and the GOTO loop. First, we will show you how to use a FOR/NEXT loop. Look at the example to see what a FOR/NEXT loop looks like. When you are finished looking at it, type it on your computer and RUN it to see what happens.


```
10 PRINT "{SHIFT CLR/HOME}"
20 FOR X=1 TO 10
30 PRINT X
40 NEXT X
50 END
```

In the second line, line 20, you will find that it says FOR X= 1 TO 10. That means that X equals 1 to 10. The letter X is a type of variable. The value of X begins at 1 and goes up to 10.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


FOR
1=1 TO 4
NEXT!

Each time the program hits the NEXT statement, it loops back to line 20 increasing the value of X by 1. That is why it is called a loop. It does this until the value of X is equal to 10 and then it leaves the loop and goes to the line after the NEXT statement. On line 30 it says to PRINT X. So the computer PRINTs the value of X. The first time through the loop, the value of X is 1, then 2, then 3, and so on until it reaches 10. That's why the computer PRINTs 1-10 on the screen. If the loop began with FOR X = 40 TO 50, it would PRINT 40, 41, 42, etc. until it reached 50. A good time to use FOR/NEXT loops is when you have to do the same thing several times. If you have 10 names to enter you could write a program that uses a loop from 1 to 10. Look at the next example.

```
10 PRINT "{SHIFT CLR/HOME}"
20 FOR X = 1 TO 10
30 PRINT "NAME ";X;
40 INPUT N$
50 NEXT X
60 END
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

If you did not have the FOR/NEXT loop, you would have had to do it the hard way. Look at the example of the hard way.

```
10 PRINT "{SHIFT CLR/HOME}"
20 REM THIS IS THE HARD WAY
30 PRINT "NAME 1 ";
40 INPUT N$
50 PRINT "NAME 2 ";
60 INPUT N$
70 PRINT "NAME 3 ";
80 INPUT N$
90 PRINT "NAME 4 ";
100 INPUT N$
110 PRINT "NAME 5 ";
120 INPUT N$
130 PRINT "NAME 6 ";
140 INPUT N$
150 PRINT "NAME 7 ";
160 INPUT N$
170 PRINT "NAME 8 ";
180 INPUT N$
190 PRINT "NAME 9 ";
200 INPUT N$
210 PRINT "NAME 10 ";
220 INPUT N$
230 END
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Using the FOR/NEXT loop it took only six lines to write the program. Using the old way took 22 lines (not counting the REM statement line). All a REM does is to let you put a comment in a program. It doesn't affect the program at all.

You can also change the value in a FOR/NEXT loop with something other than one. Your computer can count by twos, threes or any other number you choose. To do this, you have to use the STEP statement. It looks like this:

```
FOR X = 1 TO 100 STEP 2
```

Instead of counting from 1 to 100 by ones, it counts by twos. Enter the next program to see what happens when you use STEP.

```
10 PRINT "{SHIFT CLR/HOME}"
20 FOR X = 1 TO 100 STEP 2
30 PRINT X
40 NEXT X
50 END
```

Try changing the STEP value to see what happens. If you want to count backwards, use STEP

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

and a minus sign (–). For instance, you could have

```
FOR X = 1ØØ TO 1 STEP –1
```

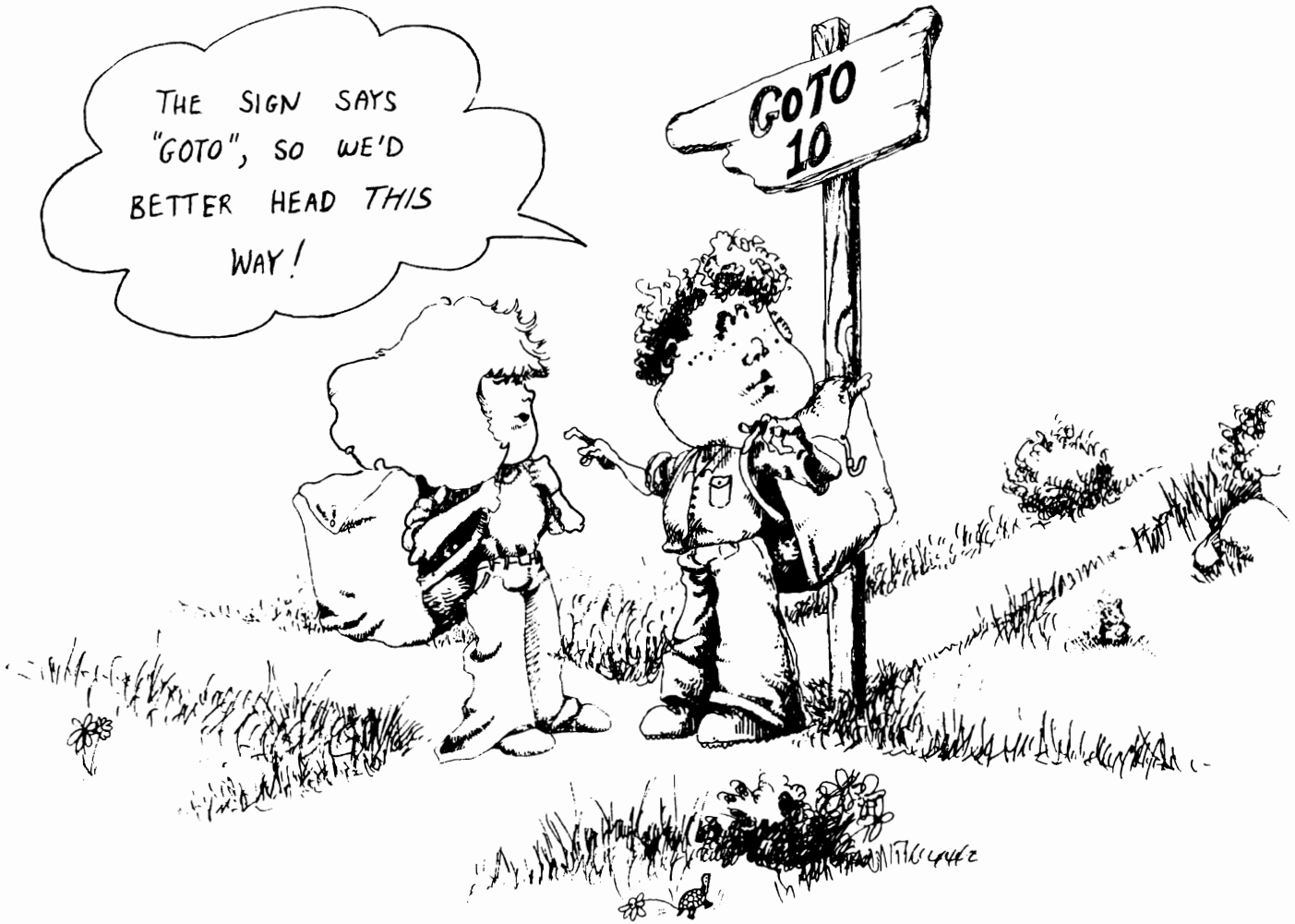
Here's a program that will count from 100 to 1.

```
1Ø PRINT "{SHIFT CLR/HOME}"
2Ø FOR X = 1ØØ TO 1 STEP –1
3Ø PRINT X,
4Ø NEXT X
5Ø END
```

Play with the statements to see what you get. The comma (,) in line 30 will print the numbers in four columns. Try changing the comma to a semicolon (;) and omitting the comma to see the different results on your screen.

Now that you know how to use FOR/NEXT loops, let's see how well you can do with the GOTO statement. Look at the next program to see what GOTO looks like in a program.

```
1Ø A$="FLOWER"
2Ø B$="BED"
3Ø PRINT A$;B$
4Ø GOTO 1Ø
```



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

On lines 10, 20, and 30 you see string variables that you learned about in the last chapter. What it says is that A$ = FLOWER and that B$, another string variable, = BED. And line 30 says to PRINT A$ and B$, so on your screen it would print FLOWERBED once. Line 40 says GOTO 1Ø, so what it is going to do is go back to 10 and do the program all over again. Your screen will fill up with FLOWERBED. It will keep on doing that until you either press the RUN/STOP key, turn it off, or until your computer breaks down. It is an endless loop. Endless loops are not used very much since you want your program to end some time. Later on we will see a better use for the GOTO statement when we look at branching and decision making. But since we are talking about endless loops, the next program shows how to make a TV commercial with an endless loop.

```
1Ø REM ******************************
2Ø REM TELEVISION COMMERCIAL
3Ø REM ******************************
4Ø PRINT "{SHIFT CLR/HOME}"
5Ø PRINT "SEE THE GREAT MOVIE"
6Ø PRINT "THE CAT THAT ATE MIAMI"
```

```
70 PRINT
80 FOR W = 1 TO 1000
90 NEXT W
100 GOTO 50
```

We used both a GOTO loop and a FOR/NEXT loop. The FOR/NEXT loop in lines 80 and 90 stopped the program for a couple of moments. The GOTO loop went back to line 50 and PRINTed the commercial message all over again. Try making your own commercial. For practice, try these next three programs.

```
10 FOR X=1 TO 1000
20 PRINT X
30 NEXT X
40 FOR V=1 TO 100
50 PRINT V,
60 NEXT V
70 PRINT "IF YOU CAN SAY ALL THOSE
   NUMBERS"
80 PRINT "IN ONE MINUTE YOU SHOULD
   BE ON "
90 PRINT "THAT'S INCREDIBLE!"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
1Ø PRINT "WHAT IS YOUR NAME?"
2Ø INPUT A$
3Ø PRINT A$
4Ø GOTO 3Ø
```

```
1Ø FOR I= 1 TO 5Ø
2Ø PRINT I
3Ø NEXT I
4Ø GOTO 1Ø
```

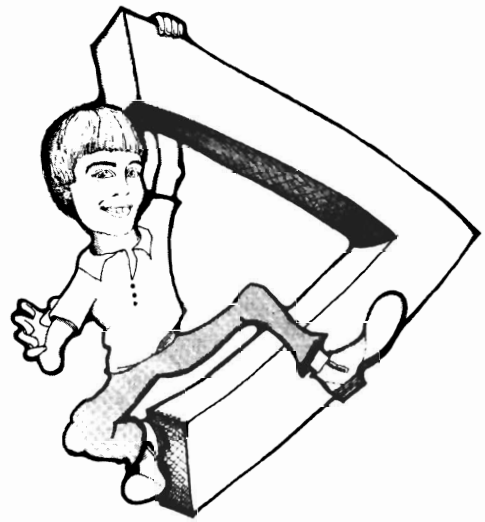
In this chapter you have learned how to use two kinds of loops. They are FOR/NEXT loops and GOTO loops. In the next chapter you will learn how to use branching and subroutines. We will see more of the GOTO statement there.

6

Decision Making

In this chapter we will explain and use the IF/THEN statement. It will enable you to do complex branching in your BASIC programs. There are two different types of branches or jumps: GOTOs and GOSUBs. These are direct branches.

We will start with the IF/THEN statement. This is called a "conditional branch" since it will jump or branch only under certain conditions. It is like saying, "IF you are going to walk in the rain THEN wear your raincoat." This means that you have to wear your raincoat only if you walk in the rain, but if you do not walk in the rain, you don't have to wear it. For example, in a program the same statement would be as follows:



```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "ENTER 1 FOR RAIN AND 2
    FOR NO RAIN"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
30 INPUT R
40 IF R = 1 THEN GOTO 100
50 IF R = 2 THEN GOTO 200
60 END
100 REM **********
110 REM RAIN
120 REM **********
130 PRINT "WEAR YOUR RAINCOAT!"
140 END
200 REM ************
210 REM NO RAIN
220 REM ************
230 PRINT "YOU DON'T HAVE TO WEAR
    YOUR RAINCOAT."
240 END
```

Now look at the next, more general program and we will explain the branches with the IF/THEN statement.

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "ENTER 1 OR 2"
30 INPUT N
40 IF N = 1 THEN GOTO 100
50 IF N = 2 THEN GOTO 200
60 END
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
1ØØ PRINT "YOU ENTERED ONE"
11Ø END
2ØØ PRINT "YOU ENTERED TWO"
```

Lines 40 and 50 used IF/THEN statements. The statements looked to see if the value of N was equal to 1 or 2. IF the value of N was 1 THEN the program branched to line 100. IF the value of N was 2 THEN the program branched to line 200. Notice that we used two END statements. If you did not enter a 1 or a 2 then there was no branch and instead the program just ENDed at line 60. In line 110 we also put in an END statement so that if you entered 1, the program would not continue to 200 and PRINT the statement YOU ENTERED TWO.

The next program is more complex, but it shows you the power you have using IF/THEN statements. It also introduces multiple statements on a single line. The RND function is explained below. To use multiple statements on a single line, we have to use the colon (:). The colon works just like putting in another line number, but you can save memory and time by using it instead of a new line.

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "WOULD YOU LIKE ME TO TELL
   YOU A JOKE?"
30 INPUT I$
40 IF I$ = "YES" THEN 100
50 IF I$ = "NO" THEN 70
60 GOTO 20 : REM *** GO ASK AGAIN ***
70 PRINT : REM *** SKIP A LINE ***
80 PRINT "SORRY TO HEAR IT, IT WAS
   A GOOD JOKE."
90 END
100 PRINT "WHY DID THE FOOL DRIVER
    MAKE" : PRINT "SEVEN PIT STOPS IN
    THE INDY 500?"
110 FOR A = 1 TO 1500 : NEXT A
    : REM *** WAIT A FEW SECONDS ***
120 PRINT : REM *** SKIP A LINE ***
130 PRINT "TWO FOR GAS AND FIVE
    FOR DIRECTIONS!!!"
140 END
```

In this program the computer asks you to INPUT whether or not you would like to see the joke. You answer by replying YES or NO. If your INPUT was YES, the computer branches to line 100 (look at line 40). If you typed NO, the com-

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

puter does not branch to line 100 because the INPUT results in line 40 being untrue. Conversely, line 50 is true (I$ = "NO"), so the program branches to line 70. Line 60 is referred to as an error trap. The only time the program will perform line 60 is when the INPUT is not either YES (the program branches to line 100) or NO (the program branches to line 70). Because YES and NO are considered to be "valid" INPUT, any other INPUT is considered to be be an error — thus the term for line 60 is error trap.

The following program demonstrates the RND (RaNDom) function (which will be explained later in the chapter) and also demonstrates some of the logical expressions shown below (called relationals).

SYMBOL	MEANING
=	EQUAL TO
<	LESS THAN
>	GREATER THAN
< >	NOT EQUAL TO
>=	GREATER THAN OR EQUAL TO
<=	LESS THAN OR EQUAL TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

All these functions can be used with the IF/THEN statement to form complex decision making. Type in the following program and RUN it to see how they work.

```
10 REM ***********************************
20 REM ***      HIGH LOW GAME          ***
30 REM ***********************************
40 PRINT "{SHIFT CLR/HOME}"
50 PRINT "I AM THINKING OF A NUMBER
   BETWEEN 1 AND"
55 PRINT "100. YOU MUST TRY TO
   GUESS MY NUMBER BY"
60 PRINT "TYPING IN DIFFERENT
   NUMBERS, AND I WILL"
65 PRINT "TELL YOU IF YOU ARE TOO
   HIGH OR TOO LOW"
70 PRINT "OR IF YOU HAVE IT."
80 RU = INT(RND(1)*(100)+1)
90 PRINT : REM *** SKIP A LINE ***
100 INPUT "ENTER GUESS"; G
110 IF G = RU THEN 180
    : REM *** CORRECT ***
120 IF G > RU THEN GOSUB 160
130 IF G < RU THEN GOSUB 170
140 GOTO 90
150 PRINT : REM *** SKIP A LINE ***
160 PRINT "IT WAS TOO HIGH" : RETURN
```

```
170 PRINT "IT WAS TOO LOW" : RETURN
180 FOR A = 1 TO 500 : NEXT A
    : REM SHORT PAUSE
190 PRINT "*** YOU GOT IT ***"
200 PRINT : REM *** SKIP A LINE ***
210 PRINT "DO YOU WANT TO
    PLAY AGAIN?"
220 INPUT I$
230 IF I$ = "YES" OR I$ = "Y"  THEN 80
240 IF I$ = "NO" OR I$= "N" THEN 260
250 PRINT "WHAT" : GOTO 210
260 PRINT
270 PRINT "OK, GOODBYE"
280 END
```

In this game you are asked to guess a number between 1 and 100. The computer will respond by telling you whether your guess was too high, too low, or just right. In line 80, the variable RU is given a RaNDom value between 1 and 100. To get a range of RaNDom numbers, enter the highest possible number you would like generated (in parentheses) after the multiplication sign (see line 80). For example, if you wanted a RaNDom number between 1 and 55, you would enter

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

The random numbers generated can be stored in variables, such as we did with RU. The INT function turns numbers into INTegers. Integers are whole numbers without fractions or decimals. Line 110 checks to see if the number you entered was equal to the computer's RANDOM number RU. If not, the program will continue to line 120 where, if the number INPUTed is greater than (>) the RaNDom number RU, the program will do a different branch called a GOSUB. When the computer is at line 160 it will do the following: One, it will print out IT WAS TOO HIGH, two, it will see that there is a RETURN; and the computer will go back to where the GOSUB left off at line 130. Now wasn't that SIMPLE! Even if we found that our number was greater than the RaNDom number, we still have to check to see if it is less than RU because the RETURN branched back to line 130, right after the last GOSUB. You might be able to figure out what's happening in 130. It's almost the same thing as in 120, except if your number was less than RU the GOSUB would branch to 170. Again after printing out

```
INT(RND(1)*55+1)
```


I SHALL RETURN!
GOSUB

the message, the RETURN will go back to where the last GOSUB left off, line 140. (Line 180 is a pause loop that serves as a short pause for you if your guess is correct.) Line 180 is where the computer branches if your guess is correct in line 110. In line 220 you're asked if you would like to play again, to which you reply either YES or NO. Did you notice that line 230 checks to see if you INPUTed YES or Y? That's where the OR function comes in. This function will let you compare two or more acceptable answers. It's the same in 240 except it checks for NO OR N. And if none of the answers were acceptable, line 250 tells you that your answer was bad and branches back to 210. Of course we always have to say "Goodbye" when people don't want to play anymore (lines 260-280).

There still might be some questions about the GOSUB/RETURN statements. If so, type the following and RUN it.

*** IMPORTANT ***

The following program must be spaced evenly. In the PRINT statements there are periods in

place of spaces. This is so you do not have to count the spaces in the line. For example, Line 30 has five periods (. . . . .). When typing in the line you should insert five spaces and then the message.

* * * EXAMPLE * * *

(THE BOOK)

```
30 PRINT " . . . . . THE COMMODORE 64
   MICROCOMPUTER"
```

(YOU TYPE)

```
30 PRINT "       THE COMMODORE 64
   MICROCOMPUTER"
```

```
10 PRINT "{SHIFT CLR/HOME}"
20 GOSUB 140 : REM *** PRINT 40
   STARS ***
30 PRINT " . . . . . THE COMMODORE 64
   MICROCOMPUTER"
40 GOSUB 140
50 PRINT : PRINT
60 GOSUB 140
70 PRINT " . . . . . THIS IS A DEMO OF
   THE"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
80 PRINT
90 PRINT "............. GOSUB/RETURN"
100 GOSUB 140
110 FOR V = 1 TO 10 : PRINT : NEXT
120 PRINT ".........<><> THE END
    <><>"
130 END
140 PRINT "******************************
    **********";
150 REM *********************************
160 REM ***        BRANCH BACK        ***
170 REM *********************************
180 RETURN
```

Let's discuss the flow of the program. Line 20 GOSUBs to line 140. Line 140 PRINTs out 40 stars (asterisks), then skips to line 180, and finally returns to the orignal GOSUB line (line 30). The program will continue to branch back and forth to the subroutine until it gets to line 130 and ENDs. Did you notice that instead of typing in several rows of stars, you just had to do it once? This illustrates one of the main uses of a GOSUB/RETURN loop. The big advantage here is the time we saved in typing and the amount of computer memory we saved. Line

110 creates space. How? It created ten vertical spaces (blank lines) in the display.

We finally made it. We're at the end of the chapter. Here are a few things to remember. With the IF/THEN statements, you can do complex testing of either numeric or string variables. Also, GOSUB/RETURN statements are good to use when a certain routine is to be used more than once.

KIDS TO KIDS TO KIDS TO

7

Arrays and READ/DATA

In this chapter we will discuss DIMs, ARRAYs, INPUTting, ARRAYs and how to use READ/DATA statements. The best way to think about ARRAYs is to think of them as a set of variables. To understand the use of an array, type in the following program.

```
NEW <RETURN>
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "ENTER FIVE NAMES."
30 INPUT A$
40 INPUT B$
50 INPUT C$
60 INPUT D$
70 INPUT E$
80 PRINT "{SHIFT CLR/HOME}"
90 REM ********************************
100 REM *** PRINT OUT THE NAMES ***
110 REM ********************************
```

```
120 PRINT "HIT RETURN TO SEE LIST
    OF NAMES."
130 INPUT AA$
140 PRINT A$
150 PRINT B$
160 PRINT C$
170 PRINT D$
180 PRINT E$
190 END
```

Though easy to understand, the program is almost useless. If there are only one or two names (or other types of INPUT), this method would be fine. What if you want to INPUT 10 or more items of data? Wouldn't that many INPUT statements be tedious? This is where arrays come in. Let's try the following program:

```
NEW <RETURN>
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT "ENTER FIVE NAMES"
30 PRINT : REM *** SKIP A LINE ***
40 REM ***
50 REM *** INPUT 5 NAMES ***
60 REM ***
70 FOR A = 1 TO 5 : REM *** SET LOOP ***
```

```
80 PRINT "ENTER NAME #" ; A;
90 INPUT NAME$(A)
100 NEXT A
110 PRINT "HIT RETURN TO
    SEE NAMES"
120 INPUT ET$
130 PRINT : REM *** SKIP A LINE ***
140 REM *** PRINT OUT NAMES ***
150 FOR A = 1 TO 5
160 PRINT "NAME #" ; A ; "=" ; NAME$(A)
170 NEXT A
180 END
```

OK, now you might see the purpose of an array. If not, read carefully. In line 70 we set up the loop from 1 to 5. In 90 we ask for an INPUT for the string variable NAME$(A). At this point in the program A=1 therefore you're INPUTting NAME$(1). We will continue doing so until the loop is done (A=5). In line 150 we'll again set up a loop from 1 to 5. Line 160 will PRINT the number we're on, PRINT out NAME$(1), and continue until it gets to A=5 or NAME$(5).

Let's say that you want to add more than just five names. You would adjust the loops to a higher

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 FOR A = 1 TO 5
20 PRINT "ENTER KID#"; A
30 INPUT KID$(A)
40 NEXT A
```

number. For example, FOR A = 1 TO 5 change to FOR A = 1 TO 20. Go ahead and change line 70 to:

```
70 FOR A = 1 TO 20
```

RUN IT !!!
You should have run into a problem (BAD SUBSCRIPT ERROR IN 90). When using string or variable arrays greater than 10, you must DIM (DIMension) the array that contains that particular string or variable. The usual way to DIM an ARRAY is to DIM it to the highest number of times we want to use the ARRAY. For example, to DIM the array NAME$ to 20 we would enter

```
DIM NAME$(20)
```

The reason for this is that we will be using the string NAME$ 20 different times, and the computer must make room for the ARRAY data. Remember that DIM reserves a certain amount of memory (which you specify). Let's fix the problem. Insert the following line:

```
65 DIM NAME$(20)
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

You should have no trouble getting to the 20th INPUT. What you have now is 20 variable names (NAME$(1) to NAME$(20)). Look at the following list comparing regular string variables with array variables. We only used an array of four in the example, but you can get the idea of how much easier it is to use arrays in certain instances (instead of variables).

Regular	Array
A$	A$(1)
B$	A$(2)
C$	A$(3)
D$	A$(4)

With arrays we can generate the variable names using FOR/NEXT loops as we did in our example program. It saves a lot of time keying in variable names and makes our programs more flexible.

It's time we learn two more different BASIC statements. These next statements are the READ/DATA statements. Type in the following program and RUN it.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 PRINT "{SHIFT CLR/HOME}"
20 FOR A = 1 TO 5
30 READ D$ : REM *** GET DATA ***
40 PRINT D$
50 NEXT A
60 DATA MONSTERS, GOBLINS, WITCHES,
   GHOSTS, VAMPIRES
70 END
```

The program first sets the loop from 1 to 5 in line 20. Then in 30 it READs the DATA from line 60, and stores it in D$. You must always separate each piece of DATA by a comma (,). This tells the computer where a new piece of data starts and where the old piece of data ends. The first time through the loop the variable D$ reads MONSTERS, then GOBLINS, and finally VAMPIRES. As the program is RUN, the values of D$ are as follows:

```
D$ = MONSTERS  <-A =1
D$ = GOBLINS   <-A =2
D$ = WITCHES   <-A =3
D$ = GHOSTS    <-A =4
D$ = VAMPIRES  <-A =5
```

In the next program we're going to tinker with transferring data from DATA statements into string arrays. This type of program would be useful when making a telephone/address file.

```
10 PRINT "{SHIFT CLR/HOME}"
20 C=1 : REM *** SET ARRAY POINTER
   TO ONE ***
30 READ B$ : REM *** GET DATA ***
40 IF B$ = "END" THEN GOTO 80
50 D$(C) = B$
60 C=C+1 : REM *** INCREASE
   COUNTER OF ARRAY ***
70 GOTO 30
80 PRINT : PRINT "HIT RETURN TO
   SEE NAMES"
90 INPUT ET$
100 REM ***
110 REM *** PRINT OUT NAMES ***
120 REM ***
130 FOR A = 1 TO C-1
140 PRINT "DATA #" ; A ; "=" ; D$(A)
150 NEXT A
160 END
170 REM ***
180 REM *** PLACE YOUR INFORMATION
    BELOW ***
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
190 REM ***
200 DATA DOG, CAT, COW, HORSE, PIG,
    GOAT, SHEEP, END
```

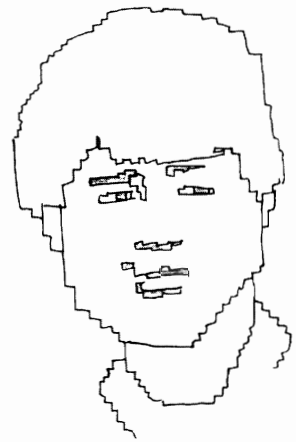
Notice we used the variable A in our FOR/NEXT loop in line 130 instead of C. It doesn't matter what variable names we use since all we want it to do is generate the numbers 1 to 7. That's because our array variables are actually D$(1) to D$(7) (not D$(C) or D$(A)). The C and the A just represent different numbers. Also, note how we used END as the last element in our DATA statement. When the computer read END it stopped READing DATA into the array and jumped to the routine for PRINTing the array to the screen.

Since you've had so much experience with the DATA/READ statements, why not make that telephone/address file for yourself? It's easy to do, but since your friends' names are probably something other than the barnyard characters we used, try putting names, addresses, and telephone numbers in your DATA statements. Use separate arrays to READ the different parts of a name/address/telephone list.

8

Commodore 64 Keyboard Graphics

This is one of our favorite chapters because it tells how to create pictures on the screen. We will show you how to make figures and give you some graphics so you will be able to create a game or a colorful program! First, we will show you how to work with the graphic characters that can be produced from the keyboard. Look at the program below to see what a program with graphics looks like, and when you are finished looking at it, type it in and RUN it.



```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT : PRINT
30 FOR H=1 TO 39
40 PRINT "{SHIFT Q}";
50 NEXT H
```

The program produces a line of circles for you. It looks like a string of pearls laid across your screen. Here's how it works: Line 10 clears the screen and line 20 takes you down two lines from the top. The FOR/NEXT loop in line 30 sets up a repeating sequence, and line 40 prints a circle to your screen. Since there is a semi-colon at the end of line 40, the graphic circles are PRINTed right next to each other, just like text. Line 50 loops back to line 30 until 39 circles are on your screen.

Let's see how to draw a vertical graphic line. This time we will use the graphic characters on the left side of the keys. (The circle, you will notice, is on the right side.) We got the character on the right side by pressing the SHIFT key and the key with the graphic character. To get the characters on the left side, press the COMMODORE key and the key with the desired graphic character.

```
10 PRINT "{SHIFT CLR/HOME}";
20 FOR V = 1 TO 22
30 PRINT SPC(5); "{COMMODORE-K}"
40 NEXT V
```

When you RUN the program, you'll get a solid vertical line. Notice that the graphic character on the K key is a half-block. Arranging the half-blocks one on top of another creates the vertical line. Line 30 uses the SPC statement to SPaCe the vertical line five spaces from the left side of the screen. The program PRINTs five spaces and a half-block each time line 30 is performed. Thus, the vertical line is made up of 22 half-blocks, with each one indented five SPaCes.

When using keyboard graphics, treat graphic characters like any other text. Each single character or character set must be enclosed in quotes. You can have many different characters, each in its own set of quotation marks, or you can define a series of characters in a single string. For example, if you want to make a box with graphic characters, you may want to define the parts of the box as string variables, like the following program does.

```
10 PRINT "{SHIFT CLR/HOME}"
20 T$ = "{COMMODORE-A}" +
   "{COMMODORE-S}"
30 B$ = "{COMMODORE-Z}" +
   "{COMMODORE-X}"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
40 PRINT : PRINT
50 PRINT SPC(5); T$ : REM BOX TOP
60 PRINT SPC(5); B$ : REM BOX BOTTOM
```

When you RUN the program, you will see a box. We did something a little bit different. We wanted both strings (B$ and T$) to be combinations of two keys; therefore, we used a plus sign (+). You can tie text and graphic strings together by using the plus sign. Remember, the plus sign must be *outside* of the quotation marks. You can even combine T$ and B$ together, but the result would not look like a box. This process of merging strings is called concatenation. You might think it would have been easier just to PRINT the graphic characters directly rather than defining them as strings. Actually, this is correct, but only if you don't plan to draw the box more than once. Add the following lines to the program. This should make it clear just why it is often a good idea to define a character set as a string variable.

```
70 FOR X=1 TO 20
80 PRINT T$;
90 NEXT X
100 FOR Y = 1 TO 20
```

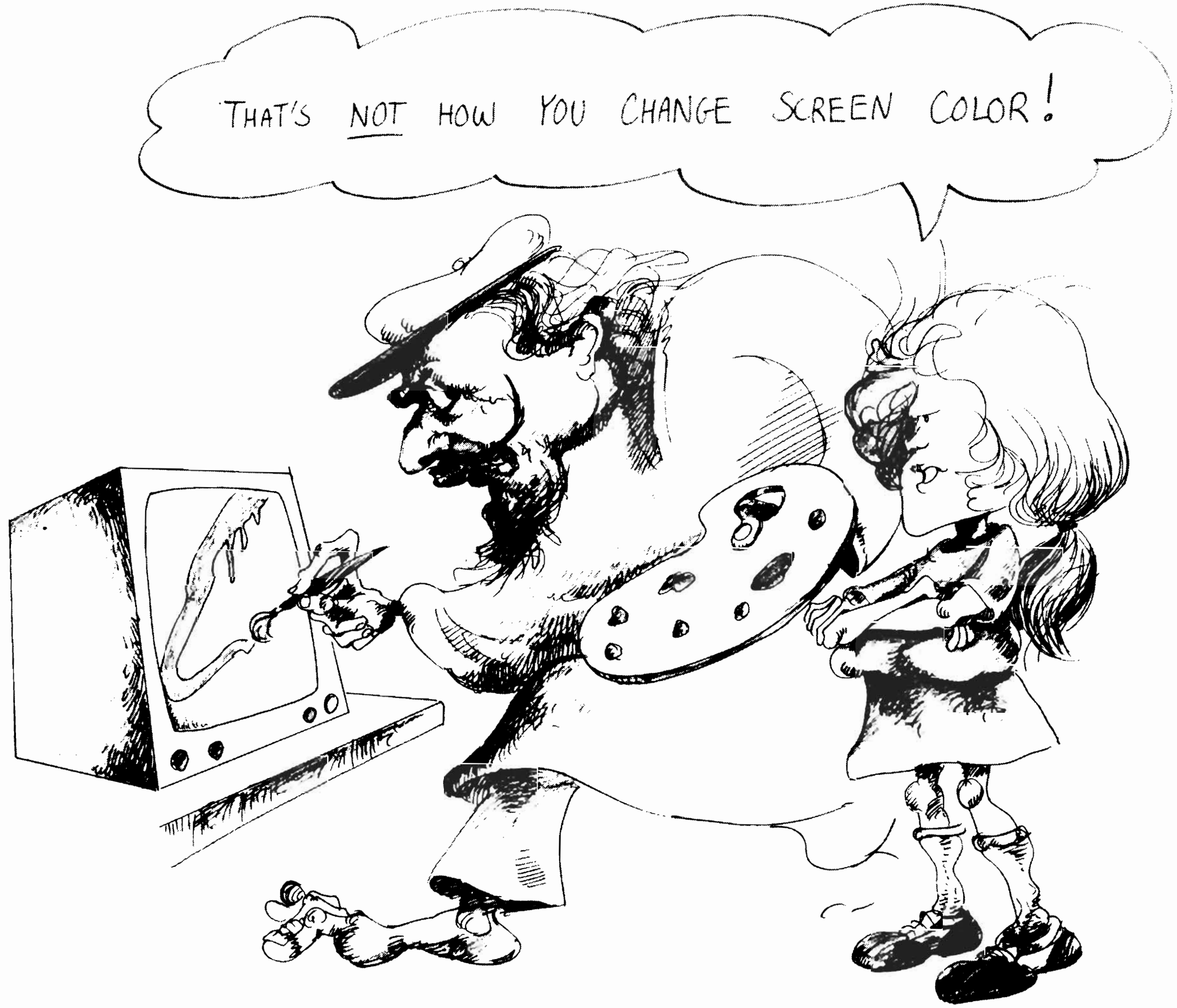
```
110 PRINT B$;
120 NEXT Y
```

It's about time you learned how to create different colors on your Commodore 64. Look at the keys on the top of your keyboard. Each has a color code printed on its front (numbered from 1-10). The following chart shows which key corresponds to which color. Using the ConTRoL key and the color keys, you can draw with the following colors:

CTRL-1 = BLACK
CTRL-2 = WHITE
CTRL-3 = RED
CTRL-4 = CYAN
CTRL-5 = PURPLE
CTRL-6 = GREEN
CTRL-7 = BLUE
CTRL-8 = YELLOW
CTRL-9 = REVERSE
CTRL-0 = REVERSE OFF

If you make a mess on the screen, just press the RUN/STOP key and the RESTORE key to get everything back to normal.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


THAT'S NOT HOW YOU CHANGE SCREEN COLOR!

Notice that there are a total of eight colors. Those colors can do a lot of different things besides coloring your text and graphic characters. Look at the program below to see how to use different colors in your programs. When you are finished looking at it, type it on your computer.

```
10 PRINT "{SHIFT CLR/HOME}"
20 S$ = " " : REM A SPACE
30 SS$ = S$ + S$ + S$ + S$ + S$ + S$
40 PRINT "{CTRL-9}{CTRL-1}"; SS$ : PRINT
50 PRINT "{CTRL-9}{CTRL-2}"; SS$ : PRINT
60 PRINT "{CTRL-9}{CTRL-3}"; SS$ : PRINT
70 PRINT "{CTRL-9}{CTRL-4}"; SS$ : PRINT
80 PRINT "{CTRL-9}{CTRL-5}"; SS$ : PRINT
90 PRINT "{CTRL-9}{CTRL-6}"; SS$ : PRINT
100 PRINT "{CTRL-9}{CTRL-7}"; SS$ : PRINT
110 PRINT "{CTRL-9}{CTRL-8}"; SS$ : PRINT
```

The above program should have made a set of color bars. If the colors do not look right, adjust your TV set or monitor. Normally when you press a color key, the letters or graphic characters are the corresponding color, but when the ReVerSe (CTRL-9) is on, the background assumes the indicated color. Since we made a space with S$ and we made a series of spaces

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


COLOR 8

with SS$, all that showed was the background, so we were able to make colored spaces look like bars.

Now that you know how to make lines and color, it is time for the grand finale. We will give you a program that will create a figure. Look below to see what the program looks like. Note the similarities in making vertical and horizontal lines. When you are finished looking at it, type it in (substituting spaces for the periods).

```
10 PRINT "{SHIFT CLR/HOME}" : PRINT
   "{CTRL-2}"
20 PRINT " . . . . . {COMMODORE-L}
   {COMMODORE-K}"
30 PRINT " . . . . . {COMMODORE-L}
   {COMMODORE-K}"
40 PRINT " . . . . . {COMMODORE-K}
   {COMMODORE-L}"
50 PRINT " . . . . . {COMMODORE-K}
   {COMMODORE-L}"
60 PRINT " . . . . {COMMODORE-K}U.
   {COMMODORE-L}"
70 PRINT " . . . . {COMMODORE-K}S.
   {COMMODORE-L}"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
80 PRINT "....{COMMODORE-K}A.
   {COMMODORE-L}"
90 PRINT "...{SHIFT £}....
   {COMMODORE-*}"
100 PRINT".....{COMMODORE-*}
    {SHIFT £}"
```

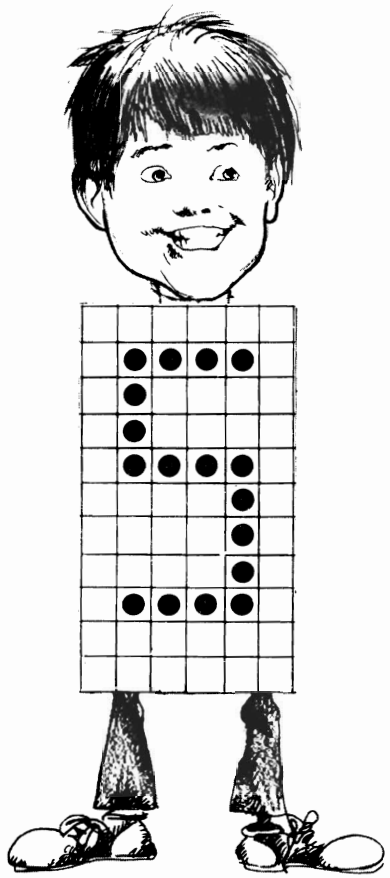
When you have finished typing it in, type RUN. If correctly typed, it should draw a white rocket ship. All we did was create strings with graphic characters and print them on the screen. Try creating some of your own characters.

In the next chapter we will show you how to use Sprite Graphics. They are more difficult to create, but you can do more with them. To make it easier for you, we have included a program to create sprites.

9

Sprite Graphics

Wouldn't it be neat if we could make the high resolution graphics in our Commodore 64 come alive? We thought so. The best and the simplest way possible to create graphics on your Commodore is called Sprite Graphics. You may ask, "What is a sprite?" A sprite is a picture in your computer's memory that can be easily displayed on your T.V. screen or monitor. Sprites have 16 different colors to choose from. Another neat feature is you can E-X-P-A-N-D your sprites to twice their horizontal size (side to side) and vertical size (up and down). Once you have expanded them you can reduce them back to normal size. Up to eight different sprites may be displayed on the screen. We will demonstrate how to display only one sprite. Each sprite is 24 (horizontal dots) by 21 (vertical dots). This means that there is a possible 504 dots in one sprite (21 * 24 = 504). Below is a grid that shows the 21 by 24 sprite block. You can use this grid to develop your own sprites.



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

SPRITE GRID

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9																								
10																								
11																								
12																								
13																								
14																								
15																								
16																								
17																								
18																								
19																								
20																								
21																								

In the normal fashion of making sprites, you would spend a great deal of time drawing the design you want on graph paper. This is fine, but when it comes time for you to calculate the numbers necessary for the sprite, it becomes a hassle. This is why we have developed the program called Sprite Creator. This program will enable you to plot a design or figure on the screen. When finished plotting your design, the computer will start calculating the data for the sprite. When the computer has completed the calculations, it will then display the information for your sprite. You should get a piece of paper and copy down the numbers that the computer displays on the screen. (The information you will see is the sprite data. We will show you what to do with the numbers a little later in a program to display your sprite.) After copying the information, the computer will display the sprite you have just created. The sprite you will see will be expanded to twice its normal size.

Before typing in the Sprite Creator program, we thought we would show you what a sprite looks like. Type in the following program and RUN it. Make sure that all of the DATA is correct, or the sprite will be incorrect.

```
10 PRINT "{SHIFT CLR/HOME}"
20 REM * *********************************
30 REM ***      POKE SPRITE DATA      ***
40 REM ***       INTO MEMORY.         ***
50 REM * *********************************
60 FOR X=832 TO 894
70 READ S
80 POKE X,S
90 NEXT X
100 REM *********************************
110 REM ***        SPRITE DATA        ***
120 REM *********************************
130 DATA 0,24,0,0,36,0,28,66,56
140 DATA 31,129,248,31,129,248
150 DATA 28,129,56,0,129,0,0,129,0
160 DATA 0,129,0,0,129,0,0,129,0
170 DATA 0,129,0,0,129,0,0,129,0
180 DATA 0,129,0,0,129,0,28,129
190 DATA 56,31,129,248,31,129,248
200 DATA 28,129,56,0,126,0
5000 REM ***
5010 REM *** SPRITE MOVER ***
5020 REM ***
5030 PRINT
5040 REM ***
5050 REM *** GET THE SPRITE READY ***
```

```
5060 REM ***
5070 POKE 2040,13
5080 REM ***
5090 REM *** TURN ON SPRITE #1 ***
5100 REM ***
5110 POKE 53269,1
5120 REM ***
5130 REM *** SPRITE COLOR (0 TO 15) ***
5140 REM ***
5150 POKE 53287,1
5160 REM ***
5170 REM *** EXPAND SPRITE ACROSS ***
5180 REM ***
5190 POKE 53277,1
5200 REM ***
5210 REM *** EXPAND SPRITE UP ***
5220 REM ***
5230 POKE 53271,1
5240 REM ***
5250 REM *** SET SPRITE
     LOCATION ON ***
5260 REM *** SCREEN
     UP/DOWN 0-255 ***
5270 REM ***
5280 POKE 53249,129
5290 REM ***
```

```
5300 REM *** MOVE SPRITE ACROSS ***
5310 REM *** SCREEN 0-347 ***
5320 REM ***
5330 FOR X=24 TO 255
5360 POKE 53248,X
5380 NEXT X
5390 GOTO 5330
```

If you've typed in the program correctly you should see a white skateboard moving about the screen. (Below is the sprite grid we used to make the skateboard). By drawing it first, it was a lot easier (using the Sprite Creator program). The program has been filled with numerous REMark statements to make the program easier to understand. To make sure you understand the operation of the program, we will explain its operations line-by-line. Line 5030 clears the screen. Line 5070 gets the sprite ready (this tells the computer where in memory the sprite is stored. Our sprite is stored at locations 832-894). Line 5150 sets the sprite color (0 to 15). Line 5190 expands the sprite across. (If you're not sure about this, re-read the last page.) Line 5230 expands the sprite up. The next few lines are the most important ones to understand because they control

the movement of the sprite. Line 5280 controls the vertical (up/down) movement of the sprite. There are 256 different positions that the sprite may occupy. For example, if you wanted to place the sprite at the very top of the screen, you type POKE 53249,50. The reason we would POKE a 50 in memory and not a 0 (on most computers 0 would be the beginning or the top of the screen, but in our case 50 is the top of the screen) is because the screen window starts at 50; anything less than 50 would only show a portion of the sprite being displayed on the screen. Now let's say we want to display the sprite at the very bottom of the screen. We would type POKE 53249,208; 208 is the highest number we can use and still completely display the sprite. Just remember that the numbers 50 through 208 are the numbers that we can use to see the complete sprite on the screen. Any other number will only display your sprite partially. Lines 5330-5390 will do the moving of the sprite. The loop is set from 24 to 255. The effect of the skateboard moving freely on the screen is simple to do. All you have to do is POKE 53248,X (X is the value of the loop) and the sprite will have the effect of moving by con-

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

tinuing through the loop. This is the same way video games get the effect of movement. That's really about all we have to do to display and move the sprite on the screen.
Here's something that you should try. The skateboard you saw on your screen was enlarged to twice its normal size. To see the sprite at its normal size, replace line 5190 with POKE 53277,Ø and line 5230 with POKE 53271,Ø. Now RUN the program. By expanding and shrinking your sprites, you can produce many interesting effects. Try expanding the vertical and leaving the horizontal as is. Don't be afraid of experimenting with the programs we have given you.

* * * The best way to learn is to play * * *

When you save the the sprite demo program, choose the size you want your sprite to be on the screen. The reason is, you will be using lines 5000-5390 in the next program. By saving the sprite demo program, you will save yourself a great deal of typing. To start, erase lines 10 through 130. The only program lines in memory should be lines 5000 through 5390. When you are finished typing the program, come back

here and read how to properly operate the Sprite Creator program. Go to it!!!

```
100 REM ********************************
110 REM ***       SET UP ARRAYS       ***
120 REM ********************************
130 DIM D(505),S(63)
140 B(8)=1:B(7)=2
150 B(6)=4:B(5)=8
160 B(4)=16:B(3)=32
170 B(2)=64:B(1)=128
180 REM ********************************
190 REM ***       MAIN ROUTINE        ***
200 REM ********************************
210 POKE 53280,6:POKE 53281,6
220 PRINT "{SHIFT CLR/HOME}
    {CTRL-WHT}";
230 FOR I=1 TO 21
240 PRINT " ..........................":REM
    24 PERIODS; DO NOT USE SPACES
250 NEXT I
260 PRINT "{HOME}"
300 POKE 1864,32
310 PRINT TAB( 28)"SPRITE";
320 PRINT TAB(68)"CREATOR";
330 PRINT TAB(146)"SPACE=ON/OFF";
```

```
340 PRINT TAB(228)" I"
350 PRINT TAB(28 )"J K";
360 PRINT TAB(68)" M"
370 PRINT TAB(106)"F1=FINISHED"
380 X=1024
390 I=PEEK(X)
400 POKE X,224
410 F=PEEK(197)
420 POKE X,I
430 IF F< >64 THEN GOSUB 450
440 GOTO 390
450 IF NOT F=37 THEN 470
460 IF PEEK(X+1)< >32 AND (X+1)
    <1864 THEN X=X+1:RETURN
470 IF NOT F=34 THEN 490
480 IF PEEK(X-1)< >32 AND (X-1)>1023
    THEN X=X-1:RETURN
490 IF NOT F=36 THEN 510
500 IF PEEK(X+40)< >32 AND (X+40)
    <1884 THEN X=X+40:RETURN
510 IF NOT F=33 THEN 530
520 IF PEEK(X-40)< >32 AND (X-40)
    >1023 THEN X=X-40:RETURN
530 IF NOT F=60 THEN 560
540 IF PEEK(X)< >46 THEN
    POKE X,46:RETURN
```

```
550 POKE X,81:RETURN
560 IF F=4 THEN GOTO 580
570 RETURN
580 REM ********************************
590 REM ***      CALCULATE THE       ***
595 REM ***           SPRITE         ***
600 REM ********************************
610 C=1:Q=1
620 FOR I=1024 TO 1863
630 IF PEEK(I)=46 THEN D(C)=0
    :GOTO 650
640 D(C)=1
650 C=C+1
660 POKE I,PEEK(I) OR 128
670 Q=Q+1
680 IF Q=25 THEN Q=1:I=I+16
690 NEXT I
700 REM ********************************
710 REM ***      CALCULATE THE       ***
715 REM ***         NUMBERS          ***
720 REM ********************************
730 C1=1:U=1:FL=0:T=0
740 FOR I=1TO504
750 IF D(I)=1 THEN T=T+B(C1):FL=1
760 C1=C1+1
770 IF C1=9 THEN C1=1:GOTO 800
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
780 NEXT I
790 GOTO 87Ø
8ØØ IF FL=Ø THEN S(U)=Ø:GOTO 82Ø
81Ø S(U)=T
82Ø T=Ø:FL=Ø:U=U+1:GOTO 78Ø
83Ø REM ********************************
84Ø REM ***     PLACE SPRITE DATA     ***
85Ø REM ***         IN MEMORY         ***
86Ø REM ********************************
87Ø FOR I=1 TO 63
88Ø POKE I+831,S(I)
89Ø NEXT I
9ØØ REM ********************************
91Ø REM ***    PRINT OUT SPRITE       ***
915 REM ***             DATA          ***
92Ø REM ********************************
93Ø PRINT "{SHIFT CLR/HOME}"
94Ø PRINT "WRITE DOWN THE
    FOLLOWING INFORMATION,"
95Ø PRINT "AND KEEP IT IN A SAFE PLACE."
96Ø PRINT "THIS INFORMATION IS THE
    ACTUAL DATA FOR"
965 PRINT "THE SPRITE. TO USE THIS
    INFORMATION,"
97Ø PRINT "REFER TO THE BOOK."
98Ø PRINT "GET READY TO WRITE
    DOWN THE DATA."
```

```
 990 PRINT:PRINT:PRINT
1000 PRINT SPC(6)"PRESS 'RETURN'
     WHEN READY";
1010 POKE 198,0
1020 INPUT A$
1030 PRINT "{SHIFT CLR/HOME}"
1040 PRINT SPC(10)"*** SPRITE DATA ***"
1050 FOR X=1 TO 63
1060 PRINT S(X)",",S(X+1)",",S(X+2)
1070 X=X+2
1080 NEXT X
1090 PRINT SPC(3)"PRESS 'RETURN'
     TO SEE THE SPRITE ";
1100 INPUT A$
5000 REM ***
5010 REM *** SPRITE MOVER ***
5020 REM ***
5030 PRINT "{SHIFT CLR/HOME}"
5040 REM ***
5050 REM *** GET THE SPRITE READY ***
5060 REM ***
5070 POKE 2040,13
5080 REM ***
5090 REM *** TURN ON SPRITE #1 ***
5100 REM ***
5110 POKE 53269,1
5120 REM ***
```

```
5130 REM *** SPRITE COLOR (0 TO 15) ***
5140 REM ***
5150 POKE 53287,1
5160 REM ***
5170 REM *** EXPAND SPRITE ACROSS ***
5180 REM ***
5190 POKE 53277,1
5200 REM ***
5210 REM *** EXPAND SPRITE UP ***
5220 REM ***
5230 POKE 53271,1
5240 REM ***
5250 REM *** SET SPRITE
     LOCATION ON ***
5260 REM *** SCREEN
     UP/DOWN 0-255 ***
5270 REM ***
5280 POKE 53249,129
5290 REM ***
5300 REM *** MOVE SPRITE ACROSS ***
5310 REM *** SCREEN 0-347 ***
5320 REM ***
5330 FOR X=24 TO 255
5360 POKE 53248,X
5380 NEXT X
5390 GOTO 5330
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

At this point, you should have the Sprite Creator program typed in and be ready to discuss the program. If possible, while reading the program operation it is a good idea to have the program in memory and running. This allows you to try the different functions while learning about them. To begin, when using the sprite Creator program, you move around the screen using the I,J,K, and M keys. Find them on the keyboard. These keys are placed just right for up/down, left/right moves.

```
  I
J   K
  M
```

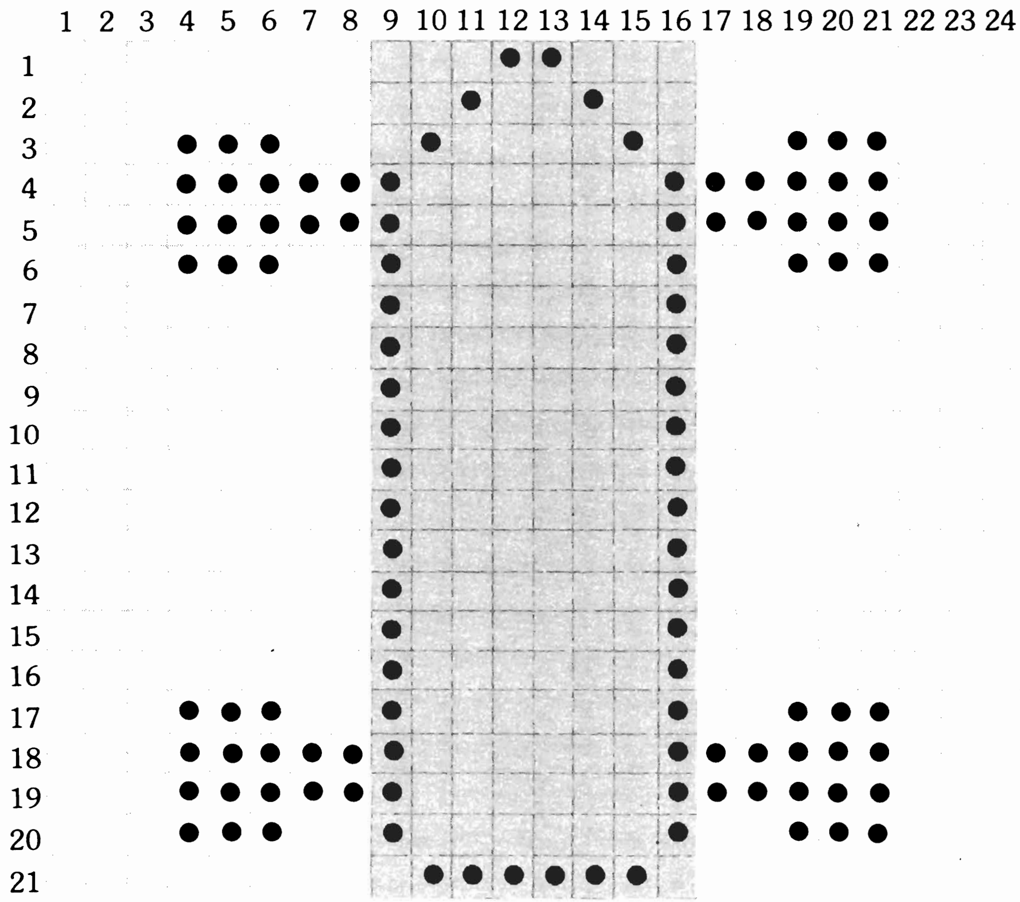
This is what it looks like on your keyboard.

I is for moving up, M is for moving down, J is for moving left, and K is for moving right. To place a dot over the period, hit the SPACE BAR, and to re-place a dot and place the period back, you again hit the SPACE BAR. Be sure to tap the SPACE BAR when using it. The reason being, the way the program is set up, the SPACE BAR is very sensitive. When you are finished, hit the

function 1 (F1) key on the right-hand side of your computer. The program will now calculate all of the necessary data for your sprite. This takes approximately 30 seconds. Once this is done, you should write down all data that is shown on the screen. The correct way of copying the data displayed on the screen onto paper is to go along each line and copy the three pieces of the sprite data one after another. Then drop down one line and continue in the same manner until finished. There will be 63 numbers to copy down. When finished copying, the computer tells you to hit the RETURN key to see the sprite. By doing so, you should see the sprite you just created. To stop the program, you must hit the RUN/STOP key. This will force the computer to stop RUNning the program.

Before going any further, you should experiment by making different designs. For example, make a better skateboard or even a picture of your best friend. When you think you're ready to continue, go on to the next section.

SKATEBOARD DEMO PROGRAM 1



1	2	3
0	24	0
0	36	0
28	66	56
31	129	248
31	129	248
28	129	56
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
0	129	0
28	129	56
31	129	248
31	129	248
28	129	56
0	126	0

5365 POKE 53249,X

At this point you should be ready to display your own sprite. You must have the data for your sprite written down, ready for transfer to DATA statements. Load in the sprite demo program. Replace the skateboard data with your own data that you have written down. Place about six to nine pieces of data per DATA statement. (Just a reminder, make sure you place a comma (,) after each piece of data.) When finished, RUN the program. You should now see your sprite moving along the screen. If your sprite did not turn out right, there are a couple of things you can do to find your problem. (1) Check all of the data in the DATA statements. (2) Did you copy the data correctly from the computer? (Remember that you write the data displayed on the screen to the paper by going along each line and copying the three pieces of the sprite data that is on the line. Then drop down one line and continue in the same manner until finished.) If your sprite is still incorrect, go back and check to make sure you have entered the Sprite Creator program correctly.

So far we've learned how to create, display and move a sprite. Let's experiment a little more with

the movement of the sprite. Load in the sprite demo program again (it doesn't matter what sprite image you have in the computer as long as you have the sprite mover program at line 5000) and add the following lines:

```
5330 X = INT (RND(1)*231)+1
5340 Y = INT (RND(1)*161)+1
5350 X = X + 24 : Y = Y + 50
5360 POKE 53248,X
5370 POKE 53249,Y
5380 FOR LOOP = 1 TO 940 : NEXT LOOP
5390 GOTO 5330
```

This little add on will place the sprite randomly around the screen. To see the sprite change colors while moving around, insert the following line:

```
5345 POKE 53287, INT(RND(1)*15+1)
```

For an even more interesting sprite moving routine, type in the following lines:

```
5330 H=53248:V=53249
5340 POKE V,50
5350 FOR X=24 TO 255
```

```
5360 POKE H,X
5370 NEXT X
5380 FOR Y=51 TO 208
5390 POKE V,Y
5400 NEXT Y
5410 FOR X=254 TO 24 STEP -1
5420 POKE H,X
5430 NEXT X
5440 FOR Y=207 TO 50 STEP -1
5450 POKE V,Y
5460 NEXT Y
5470 GOTO 5350
```

Experiment with what you have just learned. As we said before, the best way to learn is to play! That about wraps it up for this chapter. We want you to know, this was the most enjoyable chapter so far. Happy SPRITEing!

10

Sound and Music On The Commodore 64

This chapter deals with computer generated sound. Your Commodore 64 has the capability of sounding like a whole symphony. The heart to all the music is the 6581 SID (Sound Interface Device) chip. You need not be a music expert to generate fantastic and exciting sounds with your Commodore. You produce sound by POKEing into specified memory locations. (We will give you a full list of these memory locations at the end of this chapter.) We'll start off by giving you an example of the sound on the Commodore. Type in and RUN the following program.

```
10 PRINT"{SHIFT CLR/HOME}"
20 FOR T=54272 TO 54296 : POKE T,0
   : NEXT T
30 POKE 54277,6
40 POKE 54278,127
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


PITCH
CRACK!
RUMBLE

```
50 POKE 54296,15
60 READ N,DUR
70 IF N=0 AND DUR =0 THEN END
80 POKE 54273,INT(N/256)
90 POKE 54272,N-INT(N/256)*256
100 POKE 54276,33
110 FOR LOOP = 1 TO DUR : NEXT LOOP
120 POKE 54276,32
130 FOR LOOP =1 TO 45 : NEXT LOOP
140 GOTO 60
150 REM ********************************
160 REM ***       NOTE DATA FOR       ***
170 REM ***     MUSIC. ALL MUSIC      ***
180 REM ***     DATA SHOULD GO        ***
185 REM ***           HERE            ***
190 REM ********************************
200 DATA 2145,32,2703,32,2864,32,
    3215,128,0,128
210 DATA 2145,32,2703,32,2864,32,
    3215,128,0,128
220 DATA 2145,32,2703,32,2864,32,
    3215,32
230 DATA 2703,32,2145,32,2703,32,
    2408,128
240 DATA 2703,32,2703,128,2408,
    32,2145,32
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
250 DATA 2145,32,2703,128,3215,
     128,3215,32,2864,128
260 DATA 2703,32,2864,32,3215,32,
     2703,128,2145,32
270 DATA 2408,32,2145,128,2145,16
280 DATA 0,32,0,0
```

If the program was typed in correctly, you should have heard a Dixieland tune on your C-64. We will now explain what happened in this program. Line 20 clears the SID chip so that we will have a fresh start. If we neglected to clear the SID chip, it would be like us eating from dirty dishes. Who knows what is on them? Line 30 sets up the SID chip. This tells the SID chip to prepare itself to play music. Same for line 40. Line 50 is very interesting. It sets the overall volume of the sound produced by the SID chip. The sound ranges from 1 (the softest) to 15 (the loudest). You may set this to whatever level you prefer. Line 60 READs the note and the DURation value from the data table beginning at line 200. We'll explain (later) how to write your own songs. Line 70 checks to see if N (N = Note) and DUR (DUR = DURation) are both equal to zero. If so, the computer will END the program.

(This is how we tell the computer that the song is over.) Lines 80 and 90 tell the computer what note has been called for by POKEing the value of the note into memory. Line 100 tells the SID chip to start producing sound. Once this is done, we must tell the chip when to stop. Line 110 contains a FOR/NEXT loop. This loop extends the length of each note. When the delay is over, the program drops down to the next line and turns off the note being played. Line 120 turns off the sound. Line 130 is also a loop. It puts a delay between the playing of each note. If you want the computer to play the notes more quickly, shorten the length of the loop (or take it out). For example, change 50 to 10 and note the difference in the rate the song is played. Line 140 goes back to line 60 and starts the entire process over again until both N and DUR equal zero.

Now that we've discussed the program, let's experiment with it. First we'll change the volume control. In line 50, change 15 to 4 and RUN the program. Put in any values you wish. Remember, volume ranges between 1 and 15. Now let's change the overall sound of the music. Line 100

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

POKEs 33 into memory location 54276. Try POKEing the following values into 54276: 17 (run the program), 65 (run the program), and 129 (run the program). The value 17 will give an organ sound; 65 will give a short note; 129 will give a sound similar to a gunshot. Try other values. Believe me, it won't hurt your SID chip at all.

That's about all the goofing around we'll do for right now. It's time to learn how to compose our own music. To help make things simpler, we've put together a chart of all the possible notes that the computer can play. Each note has a numeric value. When you have found a note you wish to play, look at the Note Chart to find its numeric value. Place the number in a DATA statement (beginning at line 200 of the music demo program). After getting the note value, you must next tell the computer how long you want the note to be played. You do this by looking at the Note Length chart. By looking at the note on the music sheet and comparing it to the appropriate note on the Note Length chart, you can determine the value for the note. Place the length value next to the note value (separated by a comma). Let's say you want to play a G note.

MUSIC NOTES AND VALUES

#	NOTE	NOTE NUMBER
OCTAVE 1		
1	C	2145
2	C#	2273
3	D	2408
4	D#	2551
5	E	2703
6	F	2864
7	F#	3034
8	G	3215
9	G#	3406
10	A	3608
11	A#	3823
12	B	4050
13	C	4291
OCTAVE 2		
14	C#	4547
15	D	4817
16	D#	5103
17	E	5407
18	F	5728
19	F#	6069
20	G	6430
21	G#	6812
22	A	7217
23	A#	7647
24	B	8101
25	C	8583

#	NOTE	NOTE NUMBER
OCTAVE 3		
26	C#	9094
27	D	9634
28	D#	10207
29	E	10814
30	F	11457
31	F#	12139
32	G	12860
33	G#	13625
34	A	14435
35	A#	15294
36	B	16203
37	C	17167

NOTE LENGTHS

1/16 note	=	8
1/8 note	=	16
1/4 note	=	32
1/2 note	=	64
1 or whole note	=	128

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Look down the chart until you find G and write down its numeric value. Now that you have the note, set the note length to 1/2 note. The value for a half note is 64. The value for G, in octave one, is 3215. If you were to put this into a DATA statement, you would type: DATA 3215,64,Ø,Ø. Remember, only include Ø,Ø when you want to stop. For those of you who don't read music, we have a chart just for you. The chart has all of the notes drawn on a musical scale bar. All you have to do is match your note with the identically positioned note on the chart. Once you have identified the note (A,B,C,D,E,F,G, Sharps, and Flats), you can go back to the Note Chart and find the number value. There is one more thing to explain about the Note Chart. Next to the note, have you noticed a 1,2, or 3? This is the octave range for the note. To put it simply, an octave is a range or a level of sound. You should try to use the octave that sounds best with your particular song. If your song is high and bright, you would want to use the second or third octave range. If your song uses low to normal tones, you would probably want to use the first octave. All of this may be confusing, but hang in there. If all else fails, you can always experiment with the charts until you feel confident to move on.

When you have finally copied down all the information and are ready to hear your song, delete from line 200 to the end and put in your own data. Each DATA statement should contain from six to nine pieces of data. When you are finished, place two zeros at the end of your final DATA statements. This tells the computer that the data has come to an end. We've covered lots of material in the last few pages. You should slowly reread these pages, covering each item thoroughly and completely. This chapter, along with the chapter on sprites, is probably the most difficult to master. It takes time and patience to do so. For those of you who are interested, we've included another song. Just replace the existing DATA statements with the data given below. Happy composing!

```
200 REM **********************************
210 REM ***      DO-DAH DO-DAH       ***
220 REM **********************************
230 DATA 3215,32,3215,32,3215,
     32,2703,32,3215,32
240 DATA 3608,32,3215,32,2703,128,0,
     8,2703,32,2408,64,0,16
250 DATA 2703,32,2408,64,0,16,3215,
     32,3215,32,3215,32
```

```
260 DATA 2703,32,3215,32,3608,32,
     3215,32,2703,64
270 DATA 2408,64,2703,32,2408,
     32,2145,64
280 DATA 0,128,0,0
```

11

How To Make a Game

In this chapter we'll discuss how to put together a simple game. In developing a game, a major objective should be the purpose or goal of the game, such as catching bombs or shooting down green invaders. It shouldn't be so hard that people won't play because they always lose. Of course it certainly should not be too easy, enabling anyone to play for hours on end and to always win. That's not very interesting either!

In the example game program at the end of the chapter, we will be using the space bar to fire missiles at a space alien who bobs above us. We will be using a lot of movement and our cursor keys in the program. In order to create movement, called animation, we will do the same thing that cartoon makers have always done. A picture is drawn, removed, and then drawn again in a different place. When this is done rapidly, it looks like the figure is moving. To see how this

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

works, look at the following program. We will begin with side to side animation. To make it clear, we will define a keyboard character as a string. We will also create a string to erase the first drawing. The following steps will be used:

Animation

STEP 1 : PRINT a character on the screen.

STEP 2 : Leave the character there for a moment with a delay FOR/NEXT loop.

STEP 3 : Using the CRSR keys in a PRINT statement, move the cursor over the character and then erase it by PRINTing a space over it.

STEP 4 : Move the cursor again and PRINT the character again, delay, erase and go back to Step 1.

Horizontal Animation

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT : PRINT : PRINT
30 HEART$ =  "{SHIFT S}"
```

```
40 PRINT TAB(10); HEART$;
50 FOR P = 1 TO 100 : NEXT P : REM
   DELAY LOOP
60 PRINT "{LEFT CRSR} "; : REM LEFT
   CRSR ARROW AND A SPACE
70 PRINT "{RIGHT    CRSR}"; HEART$
80 FOR P = 1 TO 100 : NEXT P : REM
   DELAY LOOP
90 GOTO 10
```

To understand what the above program does, there's a new statement we have to learn, TAB. TAB works like the TAB on a typewriter. In a PRINT statement, it moves the cursor over the indicated number of spaces. In line 40 we use TAB to move the cursor to position 10 across the screen. Here's what happens in the program:

Erase
. . . . . ♥
Pause
<– Move left Erase
–> Move right ♥
Pause
Do it again

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


GOOD RESPONSE---

Now let's do the same thing moving up and down the screen instead of across.

Vertical Animation

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT : PRINT : PRINT
30 HEART$ = "{SHIFT-S}"
40 PRINT TAB(10); HEART$;
50 FOR P = 1 TO 100 : NEXT P : REM
   DELAY LOOP
60 PRINT "{LEFT CRSR} "; : REM LEFT
   CRSR ARROW AND A SPACE
70 PRINT "{LEFT CRSR} {DOWN CRSR}";
   HEART$
80 FOR P = 1 TO 100 : NEXT P : REM
   DELAY LOOP
90 GOTO 10
```

All we had to do to change movement from horizontal to vertical was to change line 70. Instead of moving the cursor to the right, we moved it to the left and down.

Now let's shoot a missile and a ray across the screen. We will use a FOR/NEXT loop to move from the left to the right. Our missile will be made up of hyphens and greater than (>) signs. We will define it as a string.

Fire Missile

```
10 PRINT "{SHIFT CLR/HOME}"
20 MISSILE$ = "->" : ERASE$ = "  "
   : REM 2 SPACES IN ERASE$
30 FOR V = 1 TO 10 : PRINT : NEXT V
40 FOR FIRE = 1 TO 20
50 PRINT MISSILE$;
60 FOR P = 1 TO 30 : NEXT P
70 PRINT "{LEFT CRSR} {LEFT CRSR}";
   ERASE$;
80 NEXT FIRE
```

Next, using color, we will fire a "death beam" across the screen. As everyone knows, a death beam is red, so our's will be red.

Death Beam

```
10 PRINT "{SHIFT CLR/HOME}"
20 BEAM$ = "{COMMODORE-U}"
```

```
30 FOR V = 1 TO 10 : PRINT : NEXT V
40 FOR FIRE = 1 TO 35
50 PRINT "{CTRL-3}";BEAM$;
60 FOR P = 1 TO 10 : NEXT P
70 NEXT FIRE
80 PRINT "{CTRL-2}"
```

Drawing the death beam was even easier than drawing the missile since we did not have to erase it. If you want a faster beam, take out line 50. Note that we ended the program by changing the character color to white. White is easier to read than red.

Next we will want to control the fire or movement from the keyboard. To do that, we need a command which will read the entire keyboard. Fortunately the Commodore 64 has such a command, it's called a GET statement. GET looks at the keyboard to see if any key has been pressed. If so, it stores the numeric value of the pressed key in a variable location. If no key has been pressed, then the program skips to the next line. Since the computer reads more quickly than you can press keys, we have to put in a loop. The loop continually checks to see if any key has

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

been pressed. We'll fix up the death beam program so that it will fire only if we hit the space bar (using GET).

```
10 PRINT "{SHIFT CLR/HOME}"
20 BEAM$ = "{COMMODORE-U}"
30 FOR V = 1 TO 10 : PRINT : NEXT V
40 GET KEY$ : IF KEY$ < > CHR$(32)
   THEN 30
50 FOR FIRE = 1 TO 35
60 PRINT "{CTRL-3}";BEAM$;
70 FOR P = 1 TO 10 : NEXT P
80 NEXT FIRE
90 PRINT "{CTRL-2}"
```

We sneaked in a new function called CHR$ on you. The CHR$ (called CHaRacter $tring) function is a code for the different keys. In Appendix F of your Commodore 64 User's Guide, there is a complete list of the CHR$ values. You will see that the value 32 is for the space bar. Line 30 says to look at what key has been pressed, and if that key is not (< >) the space bar, to loop back to the same line. In this way, the computer waits until you hit the space bar before doing anything else. Without using the loop, the program would just go on and fire the beam without waiting for you to hit a key.

Try experimenting to see what you can come up with, and come back here a little later on to continue with this chapter.

We're going to end this chapter with a simple game named ALIEN BOOM. The game is simple and straightforward. To control it you use only the space bar. You have a base, and when the alien is directly above you, you shoot a missile by pressing the space bar. When you hit the alien, it goes "BOOM!!!" and gives you one point. When you want to quit, just press Q and the game ends and your final score will be shown. We used the RND function to randomly place the alien in different positions across the top of the screen. To make the alien easier to hit, increase the value in the FOR/NEXT loop in line 100. To make it more difficult, reduce the value in line 100.

```
10 PRINT"{SHIFT CLR/HOME}" : PRINT
   "{CTRL-2}"
20 ALIEN$="={SHIFT-X}="
30 BASE$="{SHIFT-O}{SHIFT-Z}{SHIFT-P}"
40 ERASE$ = " " : REM 3 SPACES
50 GOSUB 160
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
6Ø AP = INT(RND (1)*37 + 1)
7Ø PRINT "{CLR/HOME}" : REM NO
   SHIFT - INVERSE S
8Ø PRINT TAB(AP);ALIEN$;
9Ø PRINT "{CLR/HOME}" : REM NO
   SHIFT - INVERSE S
1ØØ FOR X=1 TO 2ØØ : NEXT X
11Ø GET KEY$
12Ø IF KEY$ = "Q" THEN 4ØØ
13Ø IF KEY$ = CHR$(32) THEN GOSUB
    22Ø : REM CHR$(32) = SPACE BAR
14Ø PRINT TAB(AP);ERASE$;
15Ø GOTO 6Ø
16Ø REM ****
17Ø REM BASE
18Ø REM ****
19Ø FOR V=1 TO 2Ø : PRINT : NEXT
2ØØ PRINT TAB(2Ø);BASE$
21Ø RETURN
22Ø REM ********
23Ø REM FIRE
24Ø REM ********
25Ø FOR V=1 TO 18 : PRINT : NEXT
26Ø FOR F=1 TO 2Ø
27Ø PRINT TAB(21); "*";:FOR W=1 TO 2Ø
    : NEXT
```

```
275 PRINT "{LEFT CRSR} ";:REM NOTE
    SPACE AFTER LEFT CRSR ARROW
280 PRINT "{LEFT CRSR}";"{UP-CRSR}";
290 NEXT F
300 IF AP = 20 THEN PRINT " BOOM !!! "
    : FOR PA=1 TO 800 : NEXT
    : SC=SC + 1 : GOTO 10
310 PRINT "{CLR/HOME}" : REM NO
    SHIFT - INVERSE S
320 RETURN
400 REM *****************
410 REM TERMINATION
420 REM *****************
430 PRINT "{SHIFT CLR/HOME}"
440 PRINT : PRINT
450 PRINT "YOUR FINAL SCORE WAS ";SC
```

Well, we used a lot of different commands to make the game, and some of them are advanced. However, to make a good game, we sometimes have to get a little advanced. You have been taught the majority of statements used in the program. If you study the program carefully, you can probably understand how it works. With practice, you can create your own games. See if you can add some sound to the game and move the base around.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

12

How To Use a Printer

Printers are like computer typewriters, but they can do a lot more than a typewriter. There is a special set of printer commands to learn, and that is what this whole chapter is all about. We are going to show you how to do some things, like make a LISTing of a program to your printer and how to print a sentence or two (or three or four or more!).

Right now we are going to show you how to LIST a program to your printer. The command for that is LIST used in conjunction with a sequence of other commands. First, we have to OPEN a channel to the printer. Make sure there's some paper in your printer and turn it on. Then key in the following:

```
OPEN7,4
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

The "7" is any number between 1 and 255, and the "4" is the device number of your printer. In the back of your 1525 printer is a switch for 4 or 5. Make sure the switch is on 4. Next, key in the following little program:

```
10 REM THIS WILL BE
20 REM LISTED TO THE PRINTER
30 REM IF ALL GOES WELL!
```

Now enter the following:

```
CMD7
LIST
```

Your printer should have LISTed the program to your printer instead of to the screen. The CMD command tells the output to go to a specified channel. Since we opened the channel to the printer with 7, CMD7 sends the output to the printer. To turn it off, enter

```
PRINT#7
CLOSE7
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

As you can see, it is possible to have everything go to your printer using the CMD command. However, there will be times when you want some output to go to the screen and other to go to the printer. To do that we use PRINT# instead of CMD. The PRINT# statement sends the output to the desired channel. In our example, we used channel 7, so if we OPEN7,4 we would PRINT#7 to get our output to go to the printer. Notice in the next program how we have some output go to the screen and the rest go to the printer. Also notice how we put a comma (,) after PRINT#7.

```
10 OPEN 7,4
20 PRINT "{SHIFT CLR/HOME}"
30 PRINT "THIS GOES TO THE SCREEN"
40 PRINT#7, "HI BILLY, HOW'S IT GOING?"
50 PRINT#7, "FINE SAM, HOW'S IT GOING
   OVER THERE?"
60 PRINT#7, "FINE. HAVE YOU SEEN
   KRIS TODAY?"
70 PRINT#7, "YEAH, HE'S WORKING
   ON HIS COMPUTER"
80 PRINT#7, "I THINK I'LL WRITE HIM
   A LETTER ON MY PRINTER"
90 PRINT#7 : CLOSE7
```

Now that you have finished the program, turn on your printer and RUN it. Everything with PRINT#7 in front of it was printed on your printer, but the line "THIS GOES TO THE SCREEN" was only printed on the screen. That's because PRINT instead of PRINT# was used in line 30.

Right now we will show you how to print lower case letters and strings. Lower case letters are simple since all you have to do is to use lower case letters in your program, and they will be transferred to your printer. Strings are printed to the printer in the same way as they are to the screen.

```
10 j$ = "My best friend's name is Jonny"
20 open 7,4
30 print#7, j$
40 print#7 : close7
```

Make sure your printer is on. Now type RUN, and on your printer it should say

My best friend's name is Jonny

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

The only two letters that should be capitalized are the M in "My" and the J in "Jonny." To get lower case on the Commodore 64, just press the COMMODORE KEY. Once you do that, if your letters are all upper case, they will now be all lower case. To get upper case, all you do is to press SHIFT and the desired letter, just like on a typewriter.

Our next program is a simple one to turn your printer into a typewriter. When you RUN the program it will clear the screen and put a question mark on the left side of the screen. Type in messages, and every time you hit RETURN, the text will be printed to your printer. Be sure not to put in too many letters — about 200 — before you press RETURN. You can use it to write letters to your friends. When you are finished, press "#" when the question mark appears and it will end.

Kid Processor

```
10 PRINT "{SHIFT CLR/HOME}"
20 OPEN 7,4
30 INPUT S$
40 IF S$ = "#" THEN END
```

```
50 GOSUB 100
60 GOTO 20
100 REM ************************
110 REM OUTPUT TO PRINTER
100 REM ************************
130 PRINT#7,S$
140 RETURN
```

In this chapter you have learned how to make a printout of a program, print sentences out to the printer, and how to print out lower case letters on your printer. You also learned how to LIST a program to the printer. This is useful for debugging programs and for sending copies of programs to friends. In the next chapter you will learn to your homework on the computer.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO

13

Doing Your Homework On The Computer

In this chapter we'll show you how you can do your homework so that even your parents will want you to use your computer when doing homework. You think that's impossible? Well, type the following program and you'll see why it's not impossible. *Note: Where you see periods in the program listing, put spaces.*

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT " ..... C-64 PROBLEM
   CALCULATOR"
30 FOR V = 1 TO 8 : PRINT : NEXT
40 PRINT " ..... <1> .. ADDITION"
50 PRINT " ..... <2> .. SUBTRACTION"
60 PRINT " ..... <3> ..
   MULTIPLICATION"
70 PRINT " ..... <4> .. DIVISION"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
80 PRINT " ........ <5> .. QUIT"
90 PRINT : PRINT : PRINT
100 INPUT "PICK A FUNCTION ";A
110 PRINT "{SHIFT CLR/HOME}"
120 IF A = 1 THEN A$ = "ADDITION"
    : GOTO 210
130 IF A = 2 THEN A$ = "SUBTRACTION"
    : GOTO 210
140 IF A = 3 THEN A$ =
    "MULTIPLICATION": GOTO 210
150 IF A = 4 THEN A$ = "DIVISION"
    : GOTO 210
160 IF A = 5 THEN 410
170 GOTO 10
180 REM *********************************
190 REM ***        MAIN ROUTINE        ***
200 REM *********************************
210 PRINT : PRINT : PRINT TAB(15); A$
220 FOR V = 1 TO 7 : PRINT : NEXT V
230 INPUT "ENTER THE FIRST NUMBER ";B
240 PRINT : PRINT
250 INPUT "ENTER THE SECOND
    NUMBER ";C
260 PRINT : PRINT
270 REM *********************************
280 REM *      CALCULATE ANSWER         *
290 REM *********************************
```

```
300 IF A = 1 THEN D = B + C
310 IF A = 2 THEN D = B - C
320 IF A = 3 THEN D = B * C
330 IF A = 4 THEN D = B / C
340 REM ** ENTER THE ANSWER
350 INPUT "WHAT IS YOUR ANSWER ";A
360 IF A < > D THEN 420: REM
    *** WRONG ANSWER ***
370 PRINT "YOU'RE RIGHT!"
380 PRINT : PRINT
390 INPUT "DO YOU WANT TO TRY
    ANOTHER ";A$
400 IF A$ = "YES" OR A$ = "Y" THEN 10
410 PRINT : PRINT "OK, GOODBYE": END
420 PRINT "SORRY, THE ANSWER IS ";D
430 GOTO 380
```

You should notice that the program is quite short. There was no need to write four virtually identical programs. We could have made the program out of three main GOSUB routines combined into one main program, but instead we did it a little differently.

Line 210 is the start of the main routine that asks for both numbers. You can also refer to the numbers as the NUMERATOR (which is the top

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

or first number) and the DENOMINATOR (which is the bottom or last number). This really isn't too important for now, but it's good to know. Your parents should really appreciate a program such as this, especially if they also have trouble with your math homework.

The Commodore 64 can also help with spelling. The following can virtually guarantee a perfect score on your next spelling test if you study with the program. Here's a little program that will go over your spelling words. See if you can improve it so that it mixes up the words.

```
10 REM ****************************
20 REM ENTER SPELLING WORDS
30 REM ****************************
40 PRINT "{SHIFT CLR/HOME}" : CLR
   : RESTORE : PRINT : PRINT
50 INPUT "HOW MANY SPELLING
   WORDS";N
60 PRINT : DIM SP$(N) : PRINT
70 FOR X = 1 TO N
80 PRINT "SPELLING WORD #";X;
90 INPUT SP$(X)
100 NEXT X
```

```
200 REM **********************
210 REM SPELLING QUIZ
220 REM **********************
230 FOR X = N TO 1 STEP -1
240 PRINT "{SHIFT CLR/HOME}"
250 FOR V = 1 TO 10 : PRINT : NEXT V
260 L = LEN(SP$(X)) : PL=20-L/2
270 PRINT CHR$(18); SPC(PL); SP$(X)
280 FOR P = 1 TO 200 : NEXT P
290 PRINT CHR$(146);CHR$(147)
300 INPUT "ENTER CORRECT
    SPELLING ";CS$
310 IF CS$ = SP$(X) THEN 500
320 PRINT : PRINT "THAT'S NOT QUITE
    CORRECT" : PRINT "TRY AGAIN"
330 FOR P = 1 TO 400 : NEXT P
340 PRINT "{SHIFT CLR/HOME}"
    : GOTO 300
350 NEXT X
360 PRINT : PRINT "ANOTHER SPELLING
    LIST?(Y/N)";
370 GET AN$ : IF AN$="" THEN 370
380 IF AN$="Y" THEN 10
390 PRINT "{SHIFT CLR/HOME}" : FOR
    V=1 TO 10 : PRINT : NEXT V
400 PRINT CHR$(18);SPC(5); "GOOD
    LUCK ON YOUR SPELLING TEST!" : END
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

```
500 REM **********************
510 REM CORRECT ANSWER
520 REM **********************
530 PRINT "{SHIFT CLR/HOME}" : FOR
    I=1 TO 10 : PRINT : NEXT
540 PRINT CHR$(18);"THAT'S CORRECT!!!"
550 FOR HOLD = 1 TO 500 : NEXT HOLD
560 GOTO 350
```

WoRd PrOcEsSiNg

Have you ever wished you could be a fancy typist and have great looking school reports? Well, wish no more. The power is right at your fingertips (with the help of your Commodore 64 computer). Most computers have the capability of using a word processor. For the Commodore 64 computer there are several different ones from which to choose. Here are just a few:

Easy Script (Commodore) — Simple to learn and use, and good for just about everything you need for school.

Bank Street Writer — Designed with kids in mind. Easy to learn. It has a good manual to show you how to use it.

WordPro 3 Plus (Professional Software, Inc.) — A fast and very good word processor. It has a lot of functions, and when you have more advanced papers and reports to write for school, this one would be good.

Finally, if you have a lot of money, Script 64 (Computer Marketing Services, Inc.) has a word processor with a built-in dictionary to check your spelling. If you misspell a word, it will tell you, but you have to look up the correct spelling yourself. All of these are excellent word processors. Word processors (W/Ps) are usually very easy to use and understand. This is called being "user friendly." The word processor will allow you to do such things as realign text, move blocks of words, and easily re-write essays.

* * * E x a m p l e * * *

This is a dictionary definition of the word "computer" : a programmable electronic device that can store, retrieve, and process data.

Definition of the word "memory": any of a number of devices used by a computer for the

internal storage of information. Memory is always expressed as a factor of two. For example, 16K is actually 16,384 (or 2 ^ 14).

This is a dictionary definition of the word "computer" : a programmable electronic device that can store, retrieve, and process data.

Definition of the word "memory": any of a number of devices used by a computer for the internal storage of information. Memory is always expressed as a factor of two. For example, 16K is actually 16,384 (or 2 ^ 14).

There sure is quite a difference between the nonaligned definition and the aligned definition. Of course this is a little exaggerated, but it illustrates the benefits of the align function. We'll now give you a short list of functions that most word processor have for writing letters, documentation, and other written material:

<1> WordWrap: When at the end of a line and you are still typing a word, the WordWrap function will bring the entire word down to the next line.

<2> Block Delete: Deletes a block of text that you may not want.

<3> Copy Block: Copies a block of text and places it where indicated by the user.

<4> Line Delete: Deletes a single line.

<5> Find: The word processor will find the character you indicate.

<6> Replace: Searches the entire document for a character or word and replaces it with another one.

<7> Save: Saves text to tape or to disk.

<8> Load: Loads text from tape or disk.

<9> Append: Tacks on a file of text to the end of the existing file in memory (combining letters).

<10> Print: Sends the file in memory to a printer.

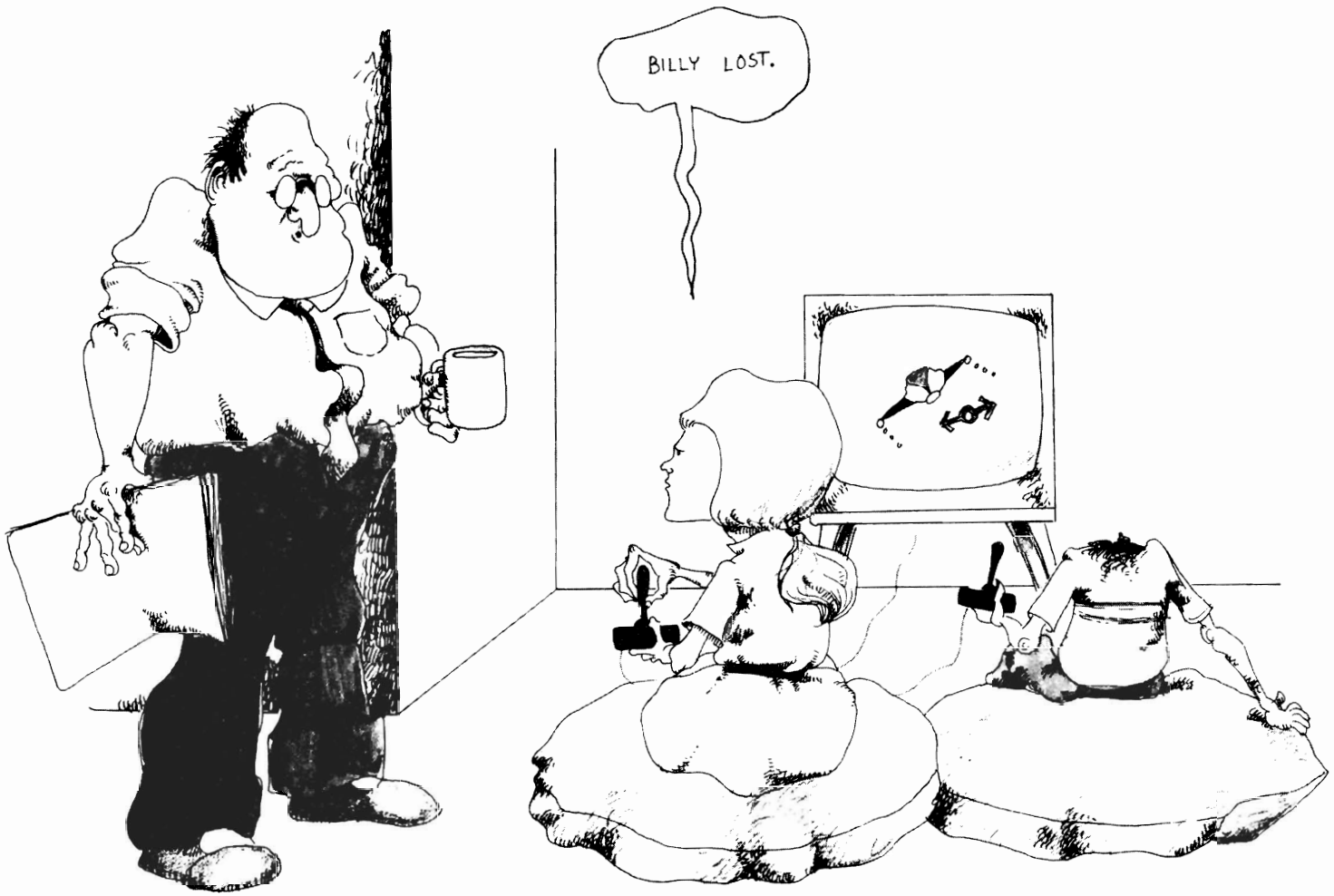
KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Written Assignments on a Word Processor

Whenever we're given a written assignment like a report or grammar homework, after we write the assignment, our parents go over it and check spelling and English. Then we have to write the whole thing over again. With a word processor though, it's really simple. All we have to do is to go back and change the mistakes, and then send it to the printer. The printer will print it as many times as you want. You don't have to retype anything except the mistakes. It helps you concentrate on correct grammar and spelling instead of all the work in redoing the whole paper. Also, when the paper is turned in, it looks much better and is clearer than a hand-written assignment.

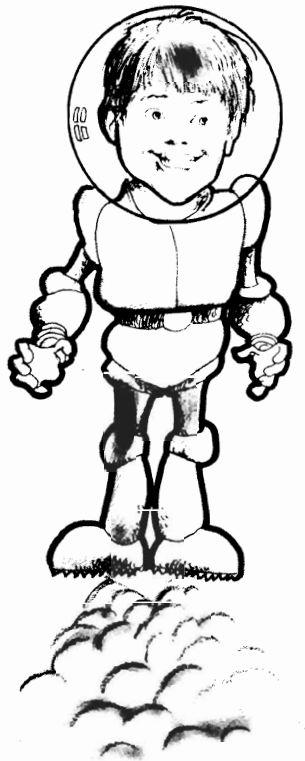
There are many, many more such commands as these that will enable you to produce great looking material. Through practice you will find that the word processor can be a very useful tool in doing just about anything.


BILLY LOST.

KIDS TO KIDS TO KIDS TO

14

Our Favorite Games and Programs



Hello there. In this chapter you will not be learning any new words, nor will you be learning anything about programming your computer. We want to tell you about games, for you, your friends and your family. After we are finished with that, we will discuss some of the top programs to get (if you are interested in that facet of programming).

The first game is CHOPLIFTER by Broderbund Software. In this game, you pilot a helicopter that takes off from your headquarters, blasts open barracks, rescues men, and brings them back to your headquarters. There are three enemies that oppose you: a tank, a jet plane, and an alien in a space pod. The game ends when all the hostages in all the barracks are

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

either dead or safely back at headquarters.

Another great game is MR. ROBOT AND HIS ROBOT FACTORY by DATAMOST. You are a robot who must collect all the power pills in the robot factory. There are moving treadmills, ladders, escalators, poles to slide down, trampolines to bounce on, magnets, bombs, and transporter disks. The best part about this game is that you can make your own game screens using the joystick to pick up and place the game objects.

Some of the best games are adventure games. These games require both fast reflexes and quick thinking. ZORK I, ZORK II, and ZORK III by Infocom are addictive adventure games. You are confronted by a lot of different obstacles and interesting opponents. The games, which require thought and planning , will keep you up all night. Infocom also makes two popular investigation games: DEADLINE and WITNESS. These games put you in the shoes of a detective trying to solve a mystery. The TEMPLE OF

APSHAI and GATEWAY TO APSHAI by Epyx are also excellent adventure games. These games let you create a superhero, wizard, dwarf, elf, or some other character. You venture into the underground world of monsters, looking for treasures. When you find the treasures and escape, you can rearm your character and go back down into the temple. These games have great graphics and are a lot of fun to play.

Programming Helpers

Some devices and programs help you do all sorts of things with your computer and help you program. These programs are called "utilities," and there are quite a few for use with the Commodore 64. The most useful is a device called the VIC-TREE by Skyles Electric Works. It plugs into the right slot in the back of your computer. This program gives you 42 additional commands and makes working with your disk system a lot easier. You can renumber your program lines, access your printer, chain programs, have automatic line numbering, and do all kinds of other helpful things you need in programming.

One version of the VIC-TREE comes with a cable for parallel printers in case you do not have a serial printer.

Communications

How would you like to talk to another computer or send your friend a program over the telephone lines? With a modem and communications software you can do that. There are a lot of modems for the Commodore 64. The Vicmodem is inexpensive, and with the right communications software, it works fine. With the modem, you will need a communications program. Some modems give you the communications software free, but sometimes it isn't very good. (The tape that comes with the Vicmodem isn't so hot.) We found that SMART 64 TERMINAL by Microtechnic Solutions, Inc. is an excellent one. It is a little hard to learn how to use, but once you get used to it, there is not much it can't do. It will even change a program file into a sequential file so you can send your friends your programs over the phone lines.

*** WANNA GO TO JAIL? ***

Some kids who have modems have been calling places that need special passwords. They break into these systems by illegally using stolen passwords. It's really easy to do once you have the password, and most of these systems are poorly protected. Usually, though, there's nothing for kids in these networks, but if they get caught, the FBI or police take their computers away and make their parents pay a fine.

Since communications with computers is a lot of fun, there ought to be something for kids. Well there is, and it's free! Go to your computer store and ask them for the number of a local bulletin board. When you get the number, call up the bulletin board and leave a message for SYSOP (that's short for System Operator.) Leave your name and phone number and the secret code word you want for your own password. Then, when you get your password, you can call them up on your computer and talk with other kids (and even adults) who also have computers. Some places even have bulletin boards for kids.

Other Neat Stuff

A good way to get a lot of free programs is to join a Commodore 64 Club. Most cities have clubs for Commodore 64 users, and usually your computer store or a teacher will be able to tell you who to call. When you are a member, the club will give you, or sell you (for little more than the price of a diskette), a whole load of public domain software. There are a lot of neat games and other programs that clubs have. Some clubs have special groups for kids too.

If you want to save money on programs but you want to increase your program library, a book (THE COMMODORE 64 EXPERIENCE (DATAMOST) by Mike Klein), will give you plenty of programs free. Also, there are several magazines like *Compute! Gazette, Commander* and *Powerplay* that have lots of programs in them.

Another magazine you might like to see is *JOYSTIK*. It's a magazine that gives you tips on winning at home computer and arcade games. They also have information on a lot of new games available for the Commodore 64. If you

can't find it in stores, write to

JOYSTIK
3841 W. Oakton St.
Skokie, IL 60076.

That's about it for our favorite games, programs, and tips on some other things you might like. We like games a lot, but it's even more fun to write your own games and programs. In the next chapter we've written a bunch of programs you might like, including an arcade game. In no time, you'll be able to write your own.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


PROFITS HAVE INCREASED FIFTY PERCENT SINCE WE COMPUTERIZED THE VOODOO DIVISION!
COMPETITION
PINS

15

Commodore 64 Programs

We've filled this chapter with several different types of programs for you to enjoy and use on your Commodore Computer. There will be some advanced programs, but don't worry about understanding them right away. The first program is called "The Messenger." This program is really fun to use when you and a friend want to send each other secret messages. Only you and your friend can understand the code (with the help of your computer.) The computer will give you a menu to choose from. You choose to either scramble a message, unscramble a message, or end. Have fun and happy scrambling!

```
10 CLR
20 PRINT "{SHIFT CLR/HOME}"
30 PRINT SPC(11) "THE MESSENGER"
40 PRINT:PRINT
50 PRINT " <1> SCRAMBLE MESSAGE"
   : PRINT
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
6Ø PRINT " <2> UNSCRAMBLE
    MESSAGE" : PRINT
7Ø PRINT " <3> QUIT" : PRINT
8Ø INPUT " ENTER CHOICE ";A
9Ø ON A GOSUB 15Ø,27Ø,39Ø
1ØØ PRINT
11Ø PRINT SPC(3) "HIT THE 'RETURN'
     KEY WHEN READY"
12Ø INPUT A$
13Ø GOTO 1Ø
14Ø END
15Ø PRINT "{SHIFT CLR/HOME}"
16Ø INPUT "ENTER MESSAGE TO
     SCRAMBLE ";V$
17Ø FOR X = 1 TO LEN (V$)
18Ø D = ASC (MID$ (V$,X,1))
19Ø IF D < 65 OR D > 9Ø THEN 22Ø
2ØØ D = D - 2
21Ø IF D < 65 THEN D = D + 26
22Ø M$ = M$ + CHR$ (D)
23Ø NEXT X
24Ø PRINT : PRINT
25Ø PRINT "MESSAGE: ";M$
26Ø RETURN
27Ø PRINT "{SHIFT CLR/HOME}"
28Ø INPUT " ENTER MESSAGE TO
     UNSCRAMBLE"; V$
```

```
290 FOR X = 1 TO LEN(V$)
300 D = ASC (MID$(V$,X,1))
310 IF D < 65 OR D > 90 THEN 340
320 D = D + 2
330 IF D > 90 THEN D = D - 26
340 M$ = M$ + CHR$ (D)
350 NEXT X
360 PRINT : PRINT
370 PRINT "MESSAGE: "; M$
380 RETURN
390 PRINT "OK . . . . " : END
```

The next program is a number guessing game. The computer RaNDomly generate a number between 1 and 100. Your job is to guess it in the fewest possible tries. Hint: For your first try, start with the number 50. Have a blast!

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT " NUMBER GUESSING GAME"
30 PRINT
40 PRINT "I'M THINKING OF A NUMBER
   BETWEEN 1 AND"
50 PRINT "100. YOU WILL ENTER A
   NUMBER, AND I"
60 PRINT "WILL TELL YOU IF YOUR
   GUESS WAS TOO"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
7Ø PRINT " 'HIGH' OR TOO 'LOW'. AND
   MOST IMPORTANT"
8Ø PRINT "IF YOUR GUESS WAS
   CORRECT. I WILL ALSO"
9Ø PRINT "TELL YOU HOW MANY TIMES
   IT TOOK YOU TO"
1ØØ PRINT "GUESS THE NUMBER."
11Ø CA = INT(RND(1)*1ØØ+1)
12Ø T=Ø
13Ø PRINT
14Ø INPUT "ENTER YOUR GUESS ";G
15Ø T=T+1
16Ø IF G<CA THEN PRINT
    "*** TOO LOW ***":GOTO 13Ø
17Ø IF G>CA THEN PRINT
    "*** TOO HIGH ***":GOTO 13Ø
18Ø PRINT:PRINT
19Ø PRINT "YOU GOT IT!!!"
2ØØ PRINT
21Ø PRINT "IT TOOK YOU"T"GUESSES"
22Ø PRINT
23Ø INPUT"DO YOU WANT TO PLAY
    AGAIN ";A$
24Ø IF A$ = "N" OR A$="NO" THEN PRINT
    "SEE YOU LATER!":END
25Ø GOTO 11Ø
```

The next program can be enjoyed by people of all ages. The game, Computer Hangman, needs two players. One player types in a message or word for the other player to guess. The player trying to guess the message or word has 15 attempts. Every time you guess a letter and it is in the puzzle, your remaining guesses stay the same. For each incorrect guess, the total number of remaining guesses is reduced by one. On the screen it will show how many guesses are left and the used letters. If at the end of the game you haven't completely guessed your opponent's message or word, the computer will display the answer. Have fun, and don't get hung.

```
10 PRINT "{SHIFT CLR/HOME}"
20 PRINT SPC(11) "COMPUTER
   HANGMAN"
30 PRINT : PRINT
40 INPUT "ENTER SECRET WORD OR
   PHRASE ";X$
50 PRINT "{SHIFT CLR/HOME}"
60 DIM U(LEN(X$))
70 T = 15
80 FOR A = 1 TO LEN (X$)
90 IF MID$(X$,A,1)=" " THEN U(A) = 32
   : GOTO 110
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
100 U(A) = 45
110 NEXT A
120 F1 = Ø
130 PRINT : PRINT
140 PRINT "TRY TO GUESS THIS:"
150 FOR A = 1 TO LEN(X$)
160 PRINT CHR$(U(A));
170 NEXT A
180 PRINT
190 S$ = ""
200 PRINT : PRINT
210 INPUT "ENTER YOUR LETTER ";S$
220 E = LEN(S$)
230 IF E < 1 OR E > 1 THEN PRINT
    "IMPROPER INPUT, TRY AGAIN"
    : GOTO 130
240 GOSUB 340
250 PRINT : PRINT
260 T$ = T$ + S$
270 PRINT "LETTERS USED: "T$
280 IF F1 = Ø THEN 300
290 GOTO 120
300 T = T - 1
310 IF T = Ø THEN 410
320 PRINT"YOU HAVE"T"GUESSES LEFT."
330 GOTO 120
```

```
340 F = 0
350 FOR A = 1 TO LEN(X$)
360 IF MID$(X$,A,1) = S$ THEN
      U(A) = ASC(S$) : F1 = 1
370 IF U(A) = 45 THEN F = 1
380 NEXT A
390 IF F = 0 THEN PRINT X$ : PRINT
      "YOU GOT IT !!!" : END
400 RETURN
410 PRINT "H A N G M A N !!!"
420 PRINT : PRINT
430 PRINT "THIS SECRET IS:"
440 PRINT X$
450 END
```

The next program can help you learn your multiplication tables. The computer will display two randomly generated numbers which you try to correctly multiply. If your answer happens to be wrong, the computer will display the correct answer. After the 10th answer, the computer will give a score of correct and incorrect answers, and the percentage of correct answers. Have fun.

```
10 PRINT "{SHIFT CLR/HOME}"
20 T = 0
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS


-- NO, J.B., I THINK YOU'RE SUPPOSED TO LEAVE ONE DOT, THEN EAT THE POWER PILL AND ATTACK THE GHOSTS ---

```
30 PRINT SPC(8) "MULTIPLICATION
   MASTER"
40 FOR A = 1 TO 10
50 PRINT : PRINT
60 D = INT (RND(1)*9+1)
70 N = INT (RND(1)*9+1)
80 PRINT D;"X"; N;"="
90 PRINT
100 INPUT "WHAT IS THE ANSWER ";Y
110 IF Y = D*N THEN 210
120 PRINT
130 PRINT "SORRY, THE RIGHT
    ANSWER IS";D*N
140 NEXT A
150 PRINT : PRINT : PRINT
160 PRINT "*** THAT'S IT. ***" : PRINT
170 PRINT "YOU GOT";T;"RIGHT AND";
    10-T;"WRONG."
180 PRINT
190 PRINT "YOUR PERCENTAGE RATE IS";
    T * 10;"%"
200 END
210 T = T + 1
220 PRINT : PRINT
230 PRINT " ******************************"
240 PRINT " ****      CORRECT      *****"
250 PRINT " ******************************"
```

```
260 PRINT : PRINT
270 GOTO 140
```

This next program can be very helpful in alphabetizing a list of names. The computer will ask you how many names to alphabetize. When you have entered the list, the computer will put it in alphabetical order. This process can take a while, depending on the number of names. Be patient. The wait is worth it.

```
10 PRINT "{SHIFT CLR/HOME}"
20 INPUT "HOW MANY WORDS TO
   ENTER ";X
30 DIM R$(X) : T = X
40 FOR A = 1 TO X
50 PRINT "ENTER WORD #";A
60 PRINT : PRINT
70 INPUT R$(A)
80 NEXT A
90 N = T
100 F = 0 : X = 1
110 IF R$ (X) < = R$ (X+1) THEN 160
120 T$ = R$ (X+1)
130 R$ (X + 1) = R$ (X)
140 R$ (X) = T$
```

```
150 F = 1
160 X = X + 1 : IF X < N THEN 110
170 IF F = 1 THEN 100
180 PRINT : PRINT
190 PRINT "WORDS IN ALPHABETICAL
    ORDER."
200 PRINT
210 FOR Y = 1 TO T
220 PRINT "#";Y;R$(Y)
230 NEXT Y
240 END
```

Have you ever wanted to protect your programs from snooping individuals? Let's say that you want a program for yourself and your friends, and if anyone else tries to run it the program will "blow up," lock up the computer, and generally go nuts. The following program does just that! Put the program you want to protect between lines 1 and 60000. Then if someone tries to run it without the password, they'll be in for a surprise.

You can test it out to see how it works when you put in the wrong password. *Before* you run the program, be sure to save it to tape or disk. You will see why!

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
1 GOSUB 60000
::::
:::: <- Your main program goes here.
::::
59999 END
60000 PRINT "{SHIFT CLR/HOME}" : C = 0
60010 FOR X = 1 TO 10 : PRINT : NEXT X
60020 PW$ = "<YOUR SECRET
        PASSWORD>"
60030 PRINT CHR$(18); " ENTER THE
        PASSWORD "; CHR$(146);
60040 INPUT WP$
60050 IF WP$ = PW$ THEN GOTO 60500
60100 REM ************************
60110 REM ILLEGAL PASSWORD
60120 REM ************************
60130 FOR X = 1 TO 200
60140 PRINT "ILLEGAL ENTRY!!";
60150 POKE 53281,C
60160 POKE 53280,C
60170 C = C + 1 : IF C = 16 THEN C = 0
60180 POKE 54296,C
60190 POKE 54277,100
60200 POKE 54278,100
60210 POKE 54276,129
60220 POKE 54273,100
60230 POKE 54272,100
```

```
60240 NEXT X
60250 POKE 768,255 : PRINT PEEK(768)
60260 SYS 0
60270 GOTO 60130
60500 REM ************************
60510 REM CORRECT PASSWORD
60520 REM ************************
60530 PN$ ="<YOUR NAME>"
60540 PG$ ="COMPUTER
      PROGRAMMER"
60560 FOR X = 1 TO LEN(PN$)
60570 PRINT MID$(PN$,X,1)
60580 FOR H = 1 TO 50 : NEXT H
60590 NEXT X
60600 FOR X = 1 TO LEN(PG$)
60610 PRINT MID$(PG$,X,1)
60620 FOR H = 1 TO 50 : NEXT H
60630 NEXT X
60640 PRINT : PRINT CHR$(18); " HIT
      ANY KEY TO PROCEED"; CHR$(146);
60650 GET A$ : IF A$ = "" THEN 60650
60660 PRINT "{SHIFT CLR/HOME}"
60670 RETURN
```

Okay gang, that does it for another chapter. Enjoy programming your Commodore 64. In time you will be able to create all the programs you want!

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

* * * GLOSSARY * * *

ARRAY An arrangement or pattern of data in succession, such as a table of numbers. Also a sequentially stored set of variables.

ASCII An acronym for American Standard Code for Information Interchange. It is a standard code adopted by most computers to remain compatible with other machines.

BASIC An acronym for Beginner's All-purpose Symbolic Instruction Code. This language is easy to learn and easy to use.

BINARY Pertains to the number system with only two digits, 1 and 0.

BOOLEAN ALGEBRA A portion of logic which is similar to algebra, but deals with logical relationships, not numerical ones.

BRANCH Alternative directions a program can take.

BUG A mistake in a computer program, or a malfunction in the computer's hardware.

CASSETTE A magnetic tape device used for program storage, and the storage of data and files.

CELL A storage place in memory with the capacity to hold only one byte.

CENTRAL PROCESSING UNIT The main component of any computer, the brain of the operation's. The CPU on the Commodore 64 is a 6510 microprocessor.

CHARACTER Any letter, number, or symbol stored by the computer.

COBOL An acronym for COmmon Business Oriented Language. A high level language for the development of complex business programs.

COMPUTER A device that executes mathematical or logical operations.

COMPUTER OPERATOR A person who has some familiarity with computers and is able to operate peripheral devices.

COMPUTER UTILITY A program or device that allows the user to gain better use of the computer's functions.

CRT An acronym for Cathode Ray Tube. The computer's TV or monitor is the CRT.

DATA Data can be computer software, a vital part of the computer's operation, or a statement in BASIC.

DEBUG To find and fix a problem that might exist in a program.

DECIMAL A base 10 numbering system using the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

DELETE To completely eliminate or remove.

DIGITAL COMPUTER A device that operates on a binary system (zeros and ones).

DISK DRIVE A form of storage for programs, files, data using hard or floppy disks or diskettes.

DISKETTE The media for the disk drive.

DOS An acronym for Disk Operating System.

EDIT To check, change, or insert data into a program.

ERROR A problem or bug in a program, usually caused by the operator of the computer.

FILES Programs or data usually stored on tape or diskette which can be called up again for later use.

FIRMWARE Software that is permanently stored in the computer. Storage media is ROM (Read Only Memory).

FLOWCHART A diagram that shows the logical operation of a program with the use of various symbols.

FORTH A high level fast language that uses user defined words to create programs. This language is available for the Commodore 64.

FORTRAN Stands for FORmula TRANslator. A high level language used primarily for making highly complex scientific and engineering computations.

GLITCH A sudden jump in the AC line voltage or other source voltages that may cause unexpected wipeouts of computer memory.

GRAPHICS A mode that allows the computer to form colorful or complex visual displays and drawings.

HARD COPY A printed copy of the output of a program, listing or graphic display.

HARDWARE The physical part of the computer (keyboard, CPU, RAM chips, ROM chips, etc.).

HEXADECIMAL A base 16 numbering system including 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

HOME COMPUTER A small low cost computer. (We think they were invented so that kids could have a computer in their home.)

INPUT Information inserted into the computer.

INPUT/OUTPUT A means of sending and receiving information, such as a disk drive or cassette deck. Usually abbreviated as I/O.

INTEGRATED CIRCUIT A complex complete circuit that requires minimal parts to get a desired circuit operation, such as a computer.

I/O An abbreviation for Input/Output.

K An abbreviation for kilo or 1000. (Actually refers to 1024 or 2^10).

KEYBOARD A group of keys manually operated for inputting information into the computer.

KEYWORD A major word element in a programming language. In BASIC, RUN, FOR, NEXT, GOTO, and PRINT are keywords.

LIGHT PEN An electronic device that allows input to the computer by the use of light on the CRT screen.

LOAD To read in information from an external device such as a disk drive and/or cassette deck.

MEMORY The part of the computer that allows storage of programs and data.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

MICROCOMPUTER A small, low cost computer that is generally used for home and small business use (and made especially for kids).

PASCAL A highly structured programming language originally used to help students learn structured programming.

PROGRAM A set of instructions that guide and inform the computer on exactly what the user wants to do.

RAM An acronym for Random Access Memory. The process of obtaining data from or placing data into storage (memory).

RUN A single statement for executing a program.

SAVE To save information to an external device such as a disk drive or cassette deck.

STRING A variable that stores any kind of characters generated by the computer in ASCII form.

SYNTAX The grammatical and base structural laws of a language.

WHOLE NUMBER A number without any fractional parts; e.g., 12, 45, 54, 99 , not 1 2/3, 4/5. Also referred to as an INTEGER. any fractional parts; e.g., 12, 45, 54, 99 , not 1 2/3, 4/5. Also referred to as an INTEGER number.

WORD A complete word is comprised of 8 bits.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO K

O KIDS TO K

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID TO

Kids to Kids ON THE COMMODORE 64

Written by kids for kids, this unique book explains BASIC programming on the Commodore 64. Created from the idea that kids can teach other kids better than anyone else, the material is designed to help you get started using and programming your Commodore 64.

You'll learn how to use the cassette and disk drive, PRINT and math statements, variables, loops, branching and subroutines, and arrays. Two chapters are devoted to sound and graphics and another will teach you how to write an original game. Before long, you'll be using your Commodore 64 to finish your homework in record time!

As an added bonus, the authors discuss their favorite programs and games. Complete with a computer glossary, KIDS TO KIDS ON THE COMMODORE 64 will teach you the magic of programming in simple, straightforward language.

Also available for the Radio Shack Color Computer and the Apple II, II+ and //e.

Other popular computer books by DATAMOST:

Compumath Magic
Computer Playground Apple II, II+ & //e
Computer Playground Atari 400/800/1200
Computer Playground Commodore 64/VIC-20
Computer Playground TI-99/4A
by M. J. Winter

Games Apples Play
Games Ataris Play
Games TIs Play
by Mike Weinstock & Mark Capella

The Atari Experience
by Adrien Z. Lamothe
The Commodore Experience
by Mike Dean Klein

Apple Almanac
The Elementary Apple
The Elementary Atari
The Elementary Commodore 64
The Elementary IBM-PC
The Elementary Timex/Sinclair
The Elementary VIC-20
The Elementary TI-99/4A
by William Sanders

How to Write an Apple Program
How to Write an IBM-PC Program
How to Write a TRS-80 Program
How to Write a Program Vol. II
Computer in Your Pocket
by Ed Faulk

Kids to Kids on the Apple
Kids to Kids on the Color Computer
by Billy Sanders & Sam Edge

Kids & the Apple
Kids & the Atari
Kids & the Commodore 64
Kids & the IBM-PC/PCjr
Kids & the Panasonic
Kids & the TI-99/4A
Kids & the VIC-20
by Ed Carlson

Using 6502 Assembly Language
p-Source
by Randy Hyde

DATAMOST INC™
20660 Nordhoff Street, Chatsworth, CA 91311-6152
(818) 709-1202

ISBN 0-8359-3679-1

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

0 21898 36799