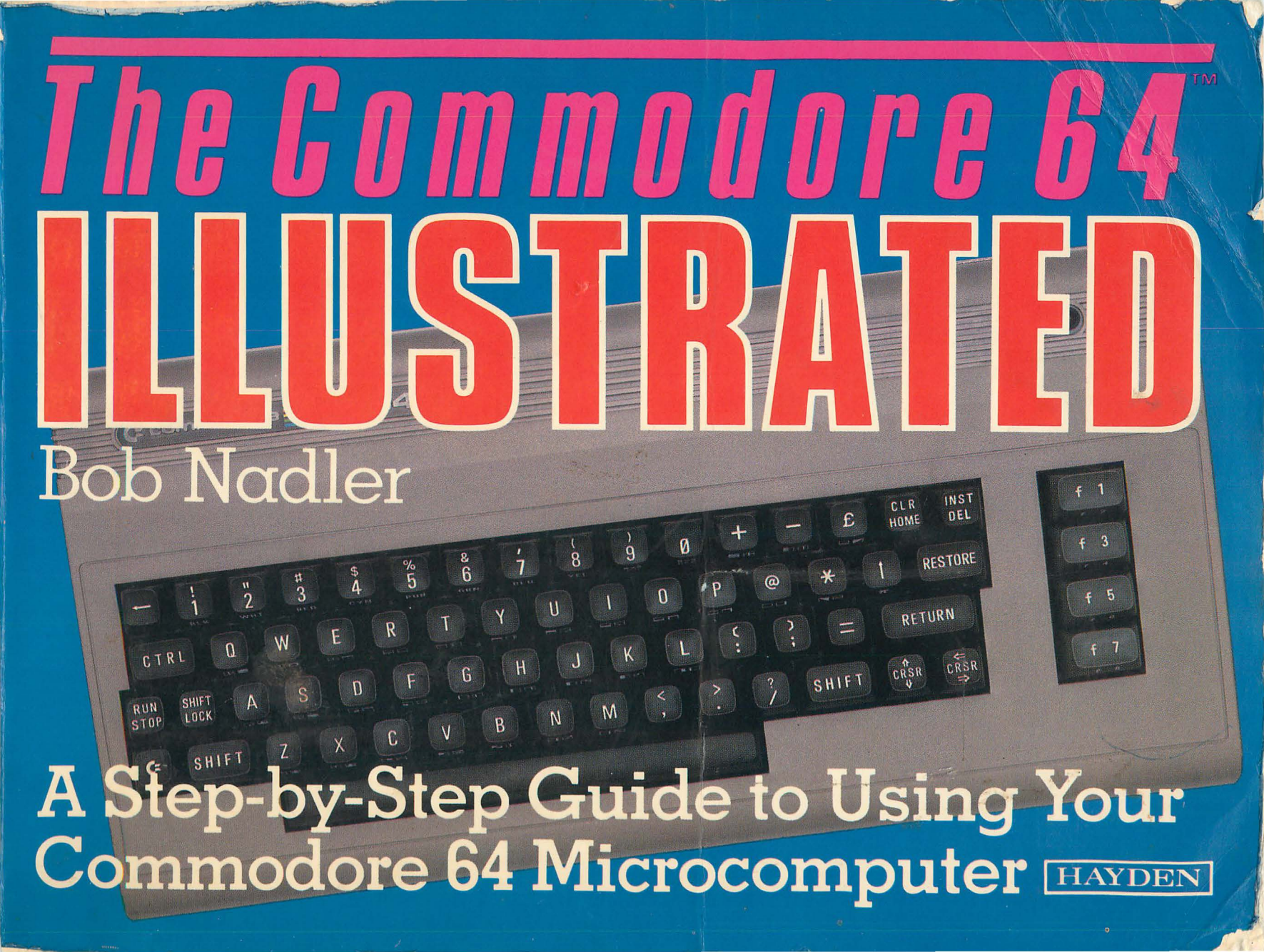
The Commodore 64™ ILLUSTRATED

Bob Nadler



A Step-by-Step Guide to Using Your Commodore 64 Microcomputer

HAYDEN

The Commodore 64™ ILLUSTRATED

The Commodore 64™ ILLUSTRATED

Bob Nadler

HAYDEN BOOK COMPANY
a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

To Gardy. Thanks.

Acquisitions Editor: PRIJONO HARDJOWIROGO
Developmental Editor: KAREN PASTUZYN
Production Editor: LORI WILLIAMS
Text and Cover Design: JOHN M-RÖBLIN
Cover Photo: LOU ODOR
Compositor: McFARLAND GRAPHICS AND DESIGN
Printed and bound by: THE BOOK PRESS

Library of Congress Cataloging in Publication Data

Nadler, Bob.
The Commodore 64 illustrated.

1. Commodore 64 (Computer) I. Title. II. Title: Commodore sixty-four illustrated.
QA76.8.C64N33 1984 001.64 84-3794
ISBN 0-8104-6453-5

Printed in the United States of America

1 2 3 4 5 6 7 8 9 PRINTING

84 85 86 87 88 89 90 91 92 YEAR

PREFACE

User-friendly is the most abused expression in the computer business. You are forever being told that this program or that computer is user-friendly. Not all of them truly are.

In good conscience, however, I will tell you that this is a user-friendly book. It was specifically planned, designed, and produced to be just that.

This book will literally *show* you each minute step along the path between opening the box your computer is packed in and writing your first simple programs. That is all it will do (that's enough for one small book to accomplish).

When you finish this book you will not be a master programmer, but you won't be afraid to sit down at your computer and use the machine. You will have begun to program. It's getting started that's hard, and sometimes frightening. The extreme simplicity of this book's picture-with-caption format—its user-friendliness, if you will—will allay those fears and get you into computing without discomfort. What more can you ask of a friend?

Bob Nadler

EQUIPMENT NEEDED

To use the programs in this book on a Commodore 64, you will need the following equipment:

- Commodore 64, Commodore Cassette Tape Recorder (Datassette), and a black-and-white TV (color optional)

CONTENTS

INTRODUCTION

This is one of the very few places in this book where you will encounter extensive uninterrupted text. All these words up front are needed to explain why I chose to do this book the way I did.

I set out to produce a book that relies almost entirely on photo illustrations with accompanying short captions, to literally *show* you just how very simple using and programming your Commodore 64 computer really is. Owning a computer and not being able to program it is frustrating indeed.

You won't be expected to bring any knowledge or understanding of computers or of programming to this book. As a matter of fact, the less you already know about these topics, the better equipped you are to benefit from this presentation. If you have been confused and alarmed by articles and lectures telling you just how difficult computing can be, the utter simplicity of this presentation may just boggle your mind.

I'll begin by *showing* you what to do with all the cables and parts you got with your Commodore 64. You will see just how they get attached to the TV set, the electrical outlet, and the computer in order to get started. Later in the presentation, and only when you realistically need to know it and can benefit from the knowledge, I'll explain how to attach and to use a Datassette with your computer, to save and reuse valuable data.

When you have mastered getting it all together, I'll begin to explain the mysteries of the keyboard. It probably looks formidable to you, what with words and symbols printed on the fronts of the keys as well as on their tops, right along with the letters, numbers, and punctuation marks you have come to expect on a typewriter keyboard. Don't worry! Once you see that there is a simple logic to it all, and that a few easy rules explain how it all operates, you will be hunting and pecking along with the best.

You will see what a cursor is and what it does. You will learn to edit—to make deletions, additions, changes, and corrections. You will discover how to call on the computer's powerful mathematical and logic functions. If the section on math seems too terrifying to deal with, just skip over to the next topic that catches your interest. Actually it won't matter whether you understand the math at all because each subject is fully illustrated with specific examples. So you can simply punch the keys exactly as shown

and get the impressive mathematical solutions the machine is capable of providing—all without the foggiest notion of how or why the math works.

You will see how to use your new computer in two completely different operational ways. I'll show you how to get results in the immediate mode so that you can use the computer as a powerful calculator or as an attention-grabbing electronic bulletin board. Once you have mastered the principles involved, I'll ease you into the delayed, or programmed, mode, so gently that you won't even realize that you are writing programs until I congratulate you for your abilities in the field.

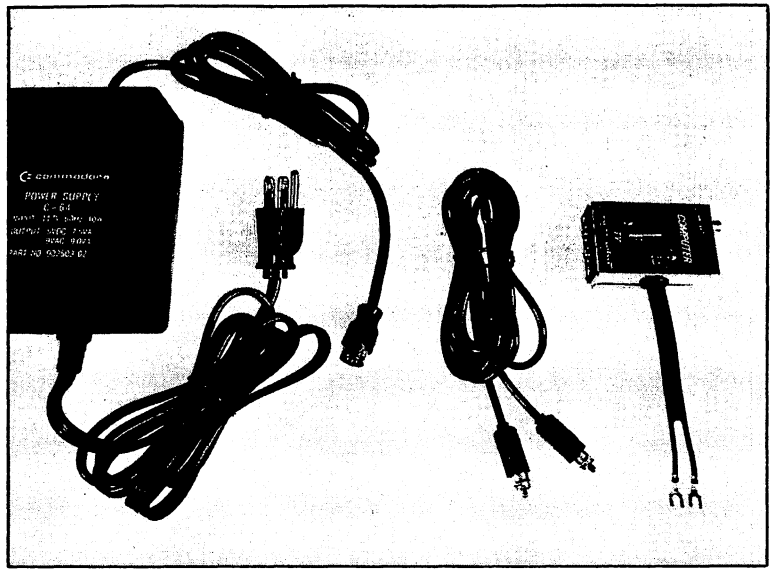
That's when I'll show you how to connect the Commodore Cassette Tape Recorder (Datassette) to enable you to save and recall programs to and from tape. Actually there is very little to it. But the process can be just a little bit nerve-wracking and confusing until you have done it a few times and you get used to it. Being able to see the whole procedure here, in these pages, will eliminate most of the usual beginner's confusion, and remove most, if not all, of the anxiety from the learning process.

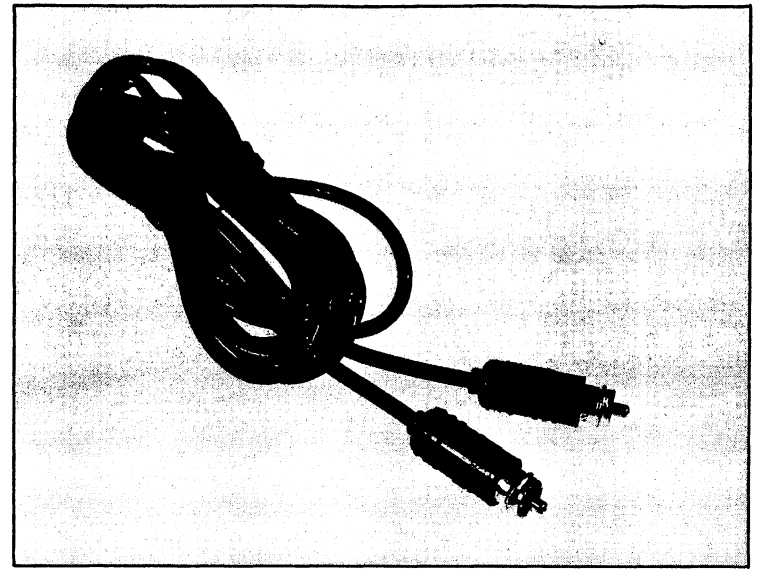
I will also show you examples of the Commodore 64's error codes. Yes, the little devil actually checks up on you and lets you know when you blow it. While it does use English to tell you what you did wrong, the messages are always terse, and often less than perfectly clear. The wrong-headed examples shown here will illustrate the computer's response to goofy errors and, perhaps, save you from making similar mistakes.

Your tour through the wonders your computer is capable of performing wouldn't be complete unless I showed you some of the graphics and color capabilities, and a few simple programming tricks and shortcuts, which can point the way for you to go on to bigger and better efforts in computing.

I would be less than candid if I didn't tell you that I have very deliberately ignored one entire aspect of the machine's abilities—sound. I did so because there was no way I could devise to *show* you anything useful. Instead, I chose to use the space available to its best advantage, and concentrated on the capabilities of the computer that could be fully illustrated with this book's unique format.

When you have finished this book, I am sure that you will feel, as I and millions of others do, that the computer is merely a tool, and a simple one at that. With your fears allayed and a whole new set of skills in place, you will be ready to join the world of high-tech. What are you waiting for? Let's get started!

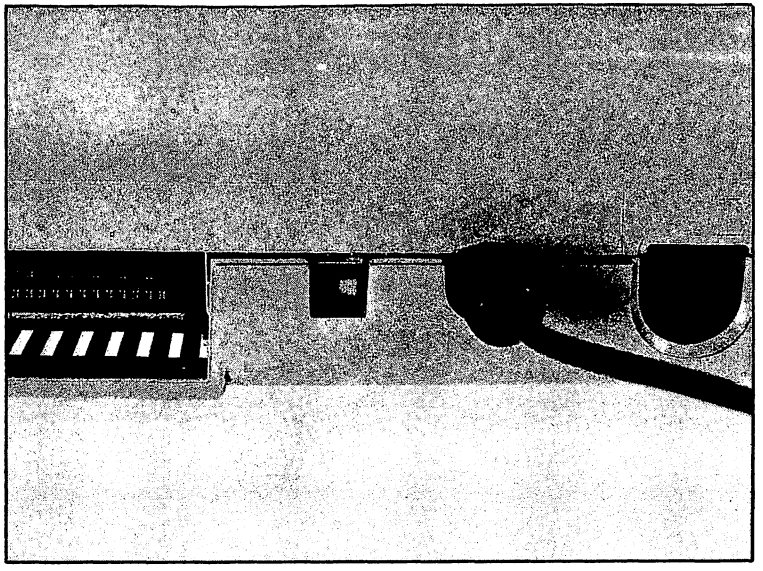




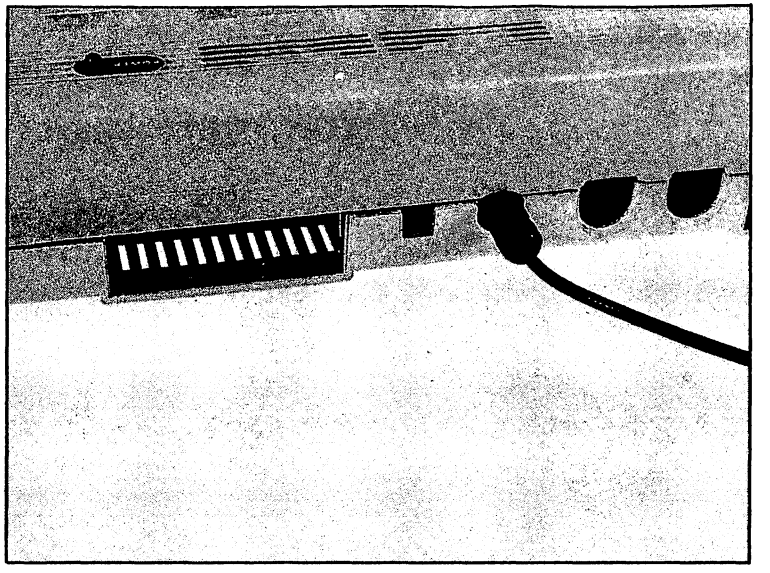
GETTING STARTED

1. The Commodore 64 computer comes equipped with (from left to right) a power supply, a TV-connector cable, and a TV-input-selector switch.

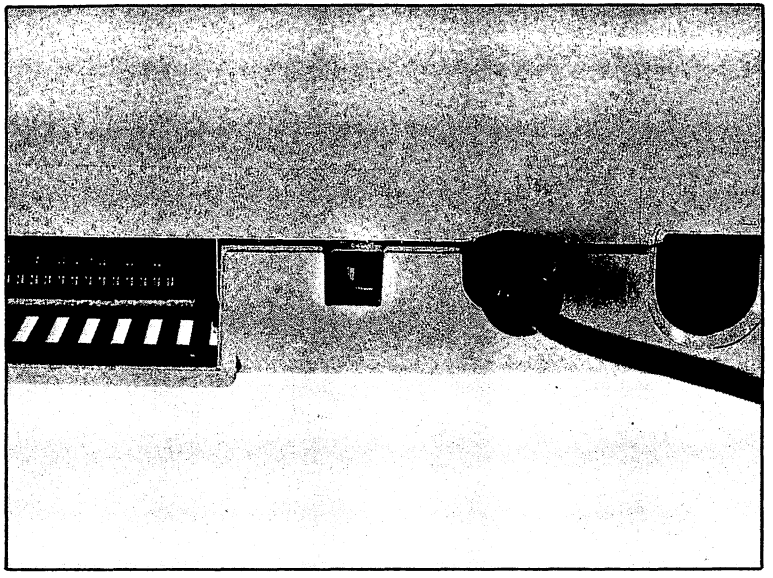
2. The TV-connector cable has a male RCA plug at each end.

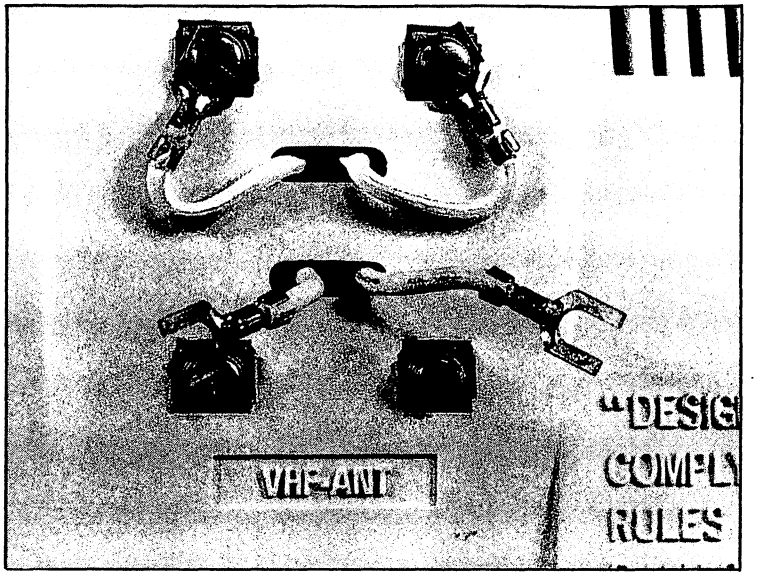


3. Push one of the RCA plugs on one end of the TV-connector cable into the socket just to the right of center at the rear of the C64. (For the sake of brevity, from now on I'll refer to the computer by this name.)



4. The small slide switch located just to the left of the TV-output socket is the channel-selector switch. It is shown here in its Channel 4 position, closest to the TV-output socket.

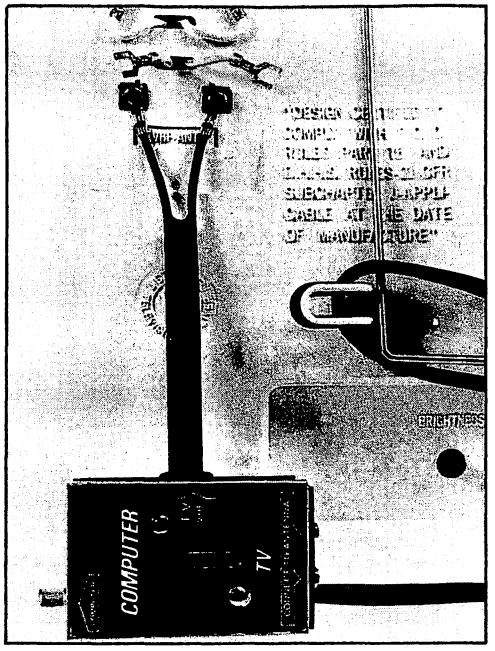




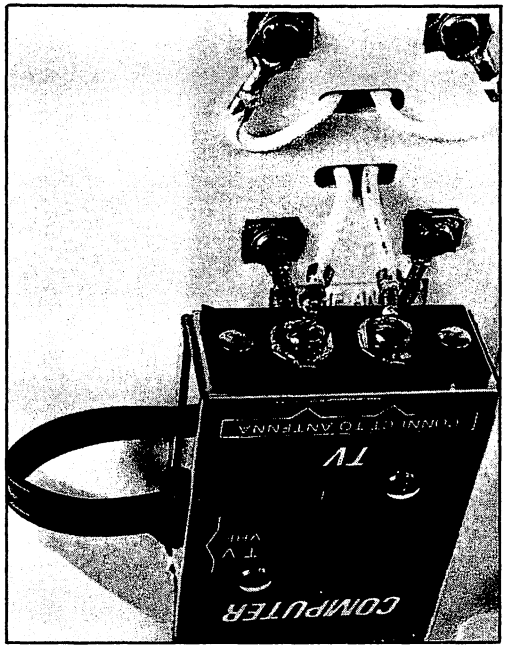
5. If Channel 4 has a strong TV signal on it in your area, use the tip of a pencil or the blade of a small screwdriver to move the switch to the Channel 3 position, away from the TV-output socket. If both channels have TV signals on them in your area, set the switch to the channel with the weakest signal.

6. To attach the TV-input-selector switch, first locate the VHF-antenna terminals on your television set. If your set has leads already attached to those terminals, remove them.

NOTE: I'd strongly recommend that you initially use a black-and-white TV, or a green or amber monitor, and only connect a color set when you reach the section in this book dealing specifically with color. The text display on a black-and-white set (or a green or amber monitor) will be clearer and easier to read than it will be on a color TV. However, if a color TV is the only one available for use with your C64, it will be satisfactory. See illustrations 328 through 334 for instructions on adjusting a color TV set for use as a monitor.

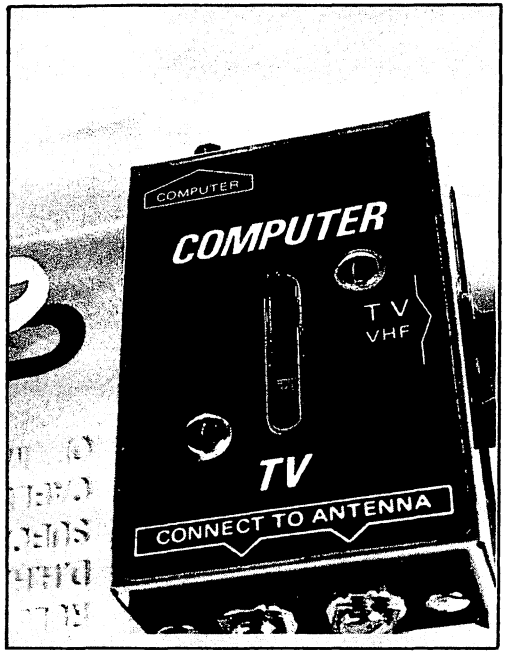


7. Attach the leads coming from the switch box to the VHF-antenna terminals.

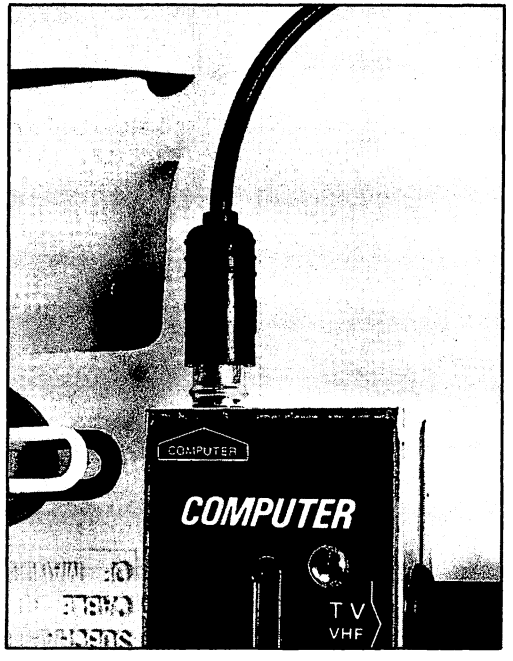


8. Then attach the wires you originally removed from the VHF-antenna-input screws to the two screws at the end of the switch box.

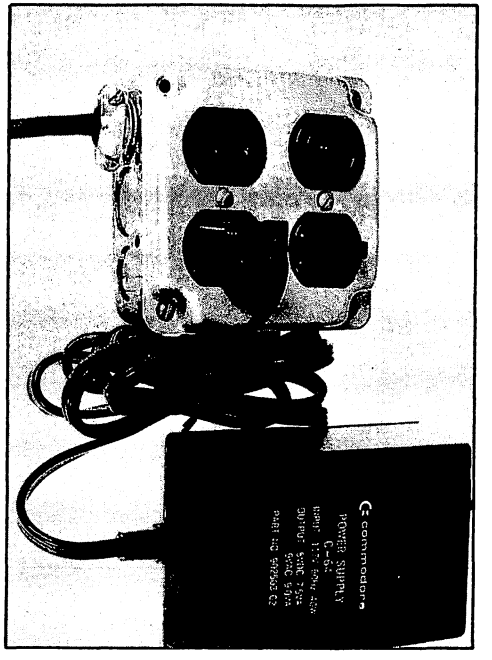
9. If your set has no wires on the VHF-antenna-input terminals but does incorporate a switch marked "Internal/External," attach the switch box to the terminals and move the switch to the "External" position. (You will have to move it back to the "Internal" position to watch TV.)



10. Be sure that the TV-input-selector switch is set at its "Computer" position. You can move it back to the "TV" position to watch TV.

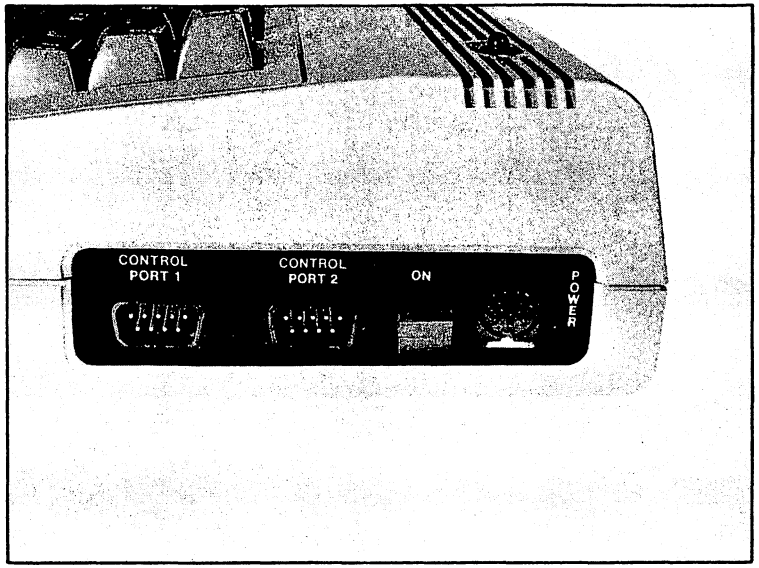


11. Push the RCA plug at the free end of the TV-connector cable into the socket labeled "Computer" on the TV-input-selector switch.

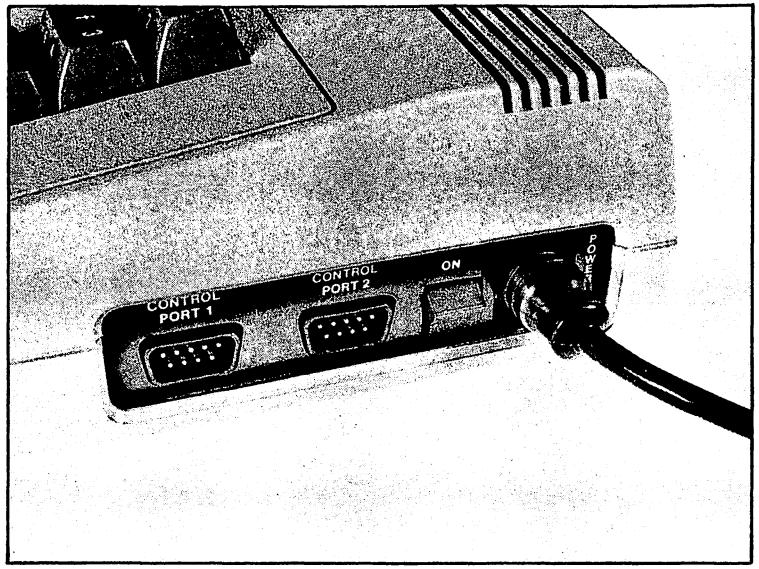


12. The power supply has two cables attached to it. Plug the cable equipped with a three-prong ac plug into a wall outlet or extension cord.

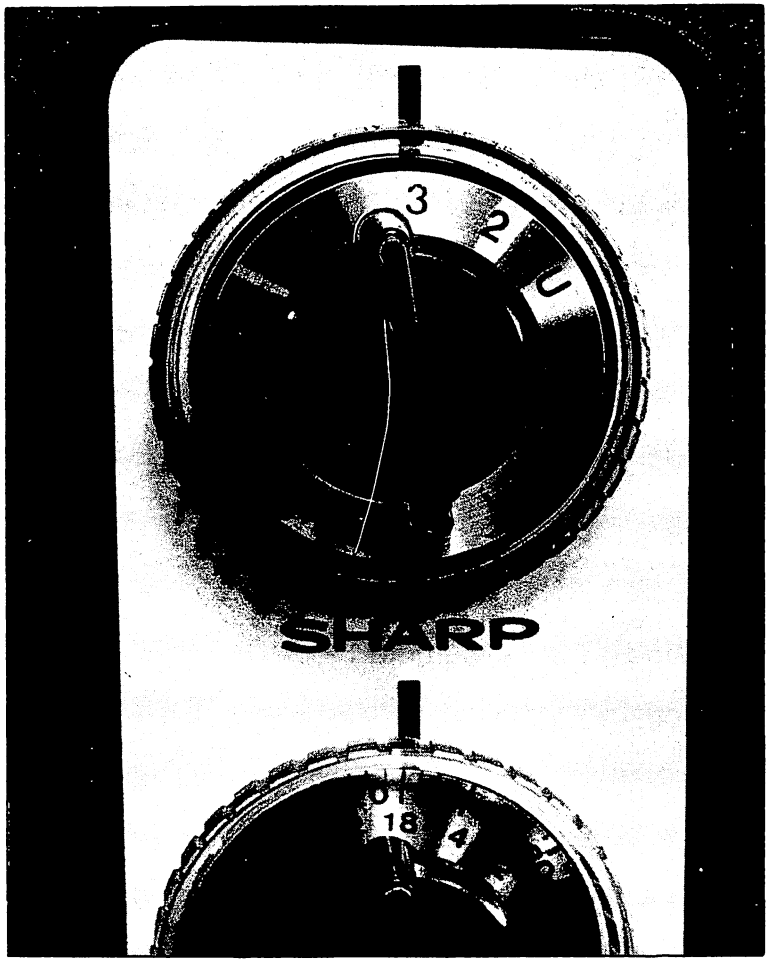
NOTE: It will be a good idea to unplug this cable when you are through using the computer, and reconnect it when you wish to use the machine again.



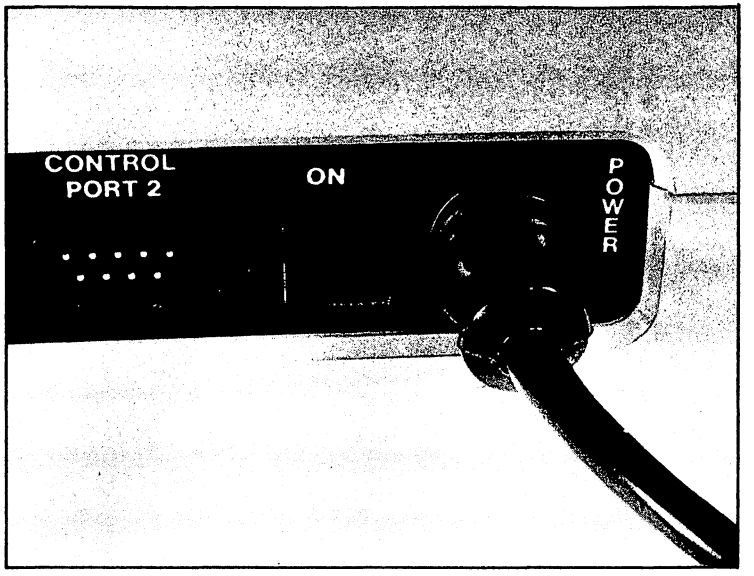
13. Locate the power switch on the right side of the C64. Make sure it is in the "Off" (down) position.



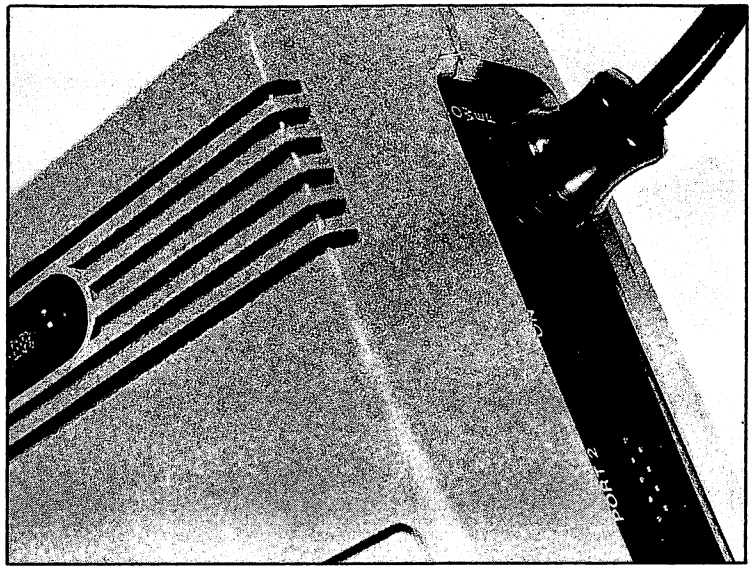
14. Then plug the other cable from the power supply into the socket marked "Power" at the rear of the right side of the computer.



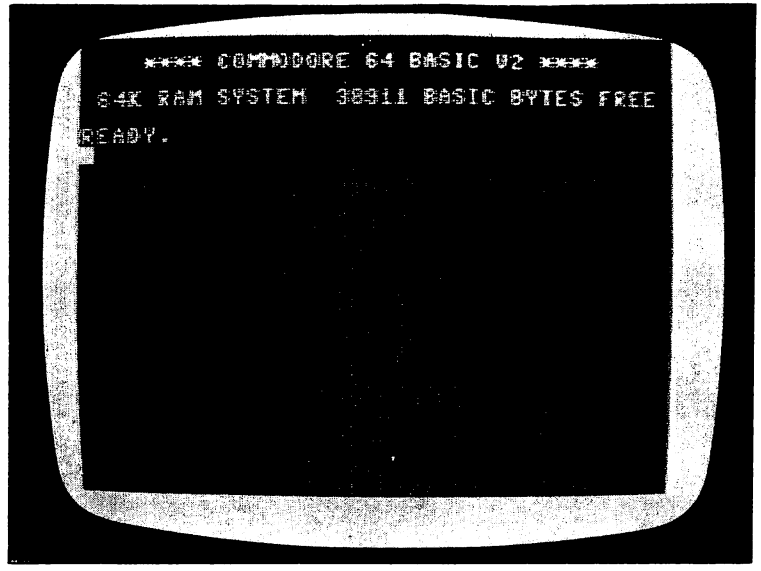
15. Be sure your TV set is plugged in. Turn the TV set on and turn the channel-selector switch to either Channel 3 or Channel 4, depending on which channel is vacant in your area.



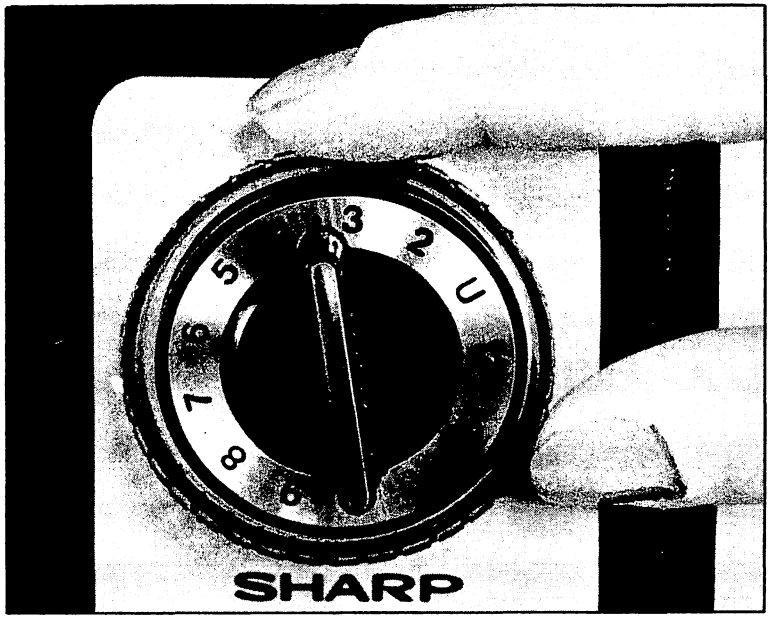
16. Now push the power switch to the "On" (up) position.

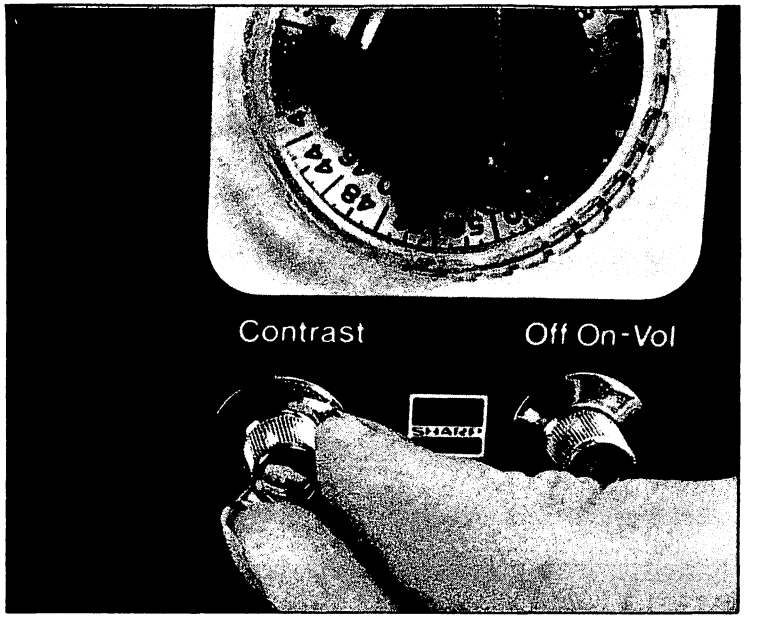


17. The red "Power" lamp on the top-right corner should light up. If it doesn't, be sure you have connected both of the power-supply cables properly (see illustrations 12 through 14).



18. After about three seconds your properly adjusted TV screen should look like this.

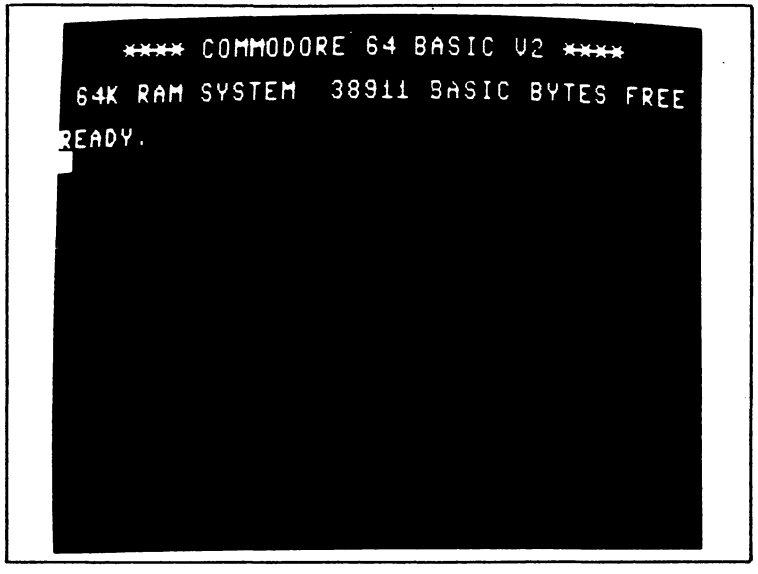


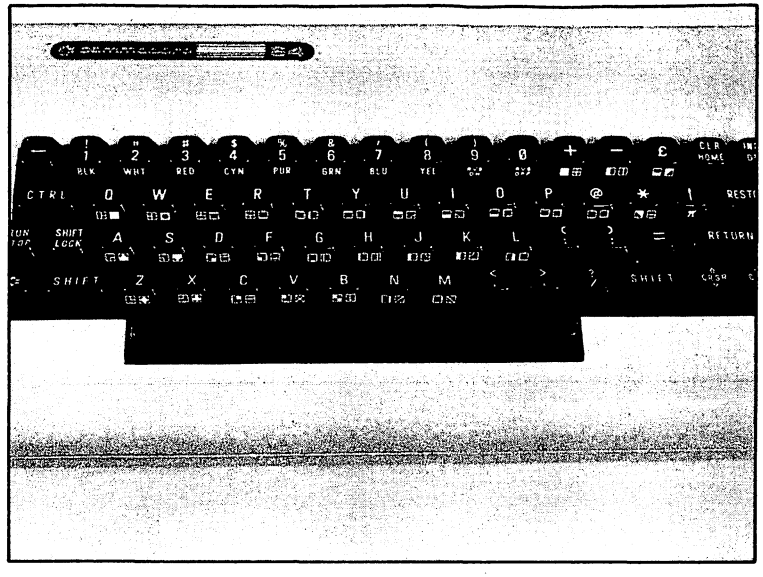


19. If the words on the screen are still unclear, try adjusting the fine-tuning control on your TV set. It is usually located concentric to the channel-selector knob.

20. Further adjustment to the TV's "Brightness" and "Contrast" controls may be necessary to achieve a good screen presentation. If you have connected your computer to a color TV set, turn to illustration 328 to see how your screen should look, and for information on how to adjust the controls.

NOTE: While a standard black-and-white TV display is adequate for direct viewing, such presentations photograph very poorly. For that reason I have used a monitor cable to connect the C64 to a green video monitor. If you compare the screen shown in illustration 21 with the one shown in illustration 18 you will immediately note the difference in sharpness and contrast, and the fact that the central portion of the monitor's screen is black, while the TV's is gray. You can convert the TV's background to black by typing in: POKE 53281,240 and pressing RETURN if you wish. All the screen illustrations offered hereafter, until the section on color is reached, have been photographed from a green monitor screen.



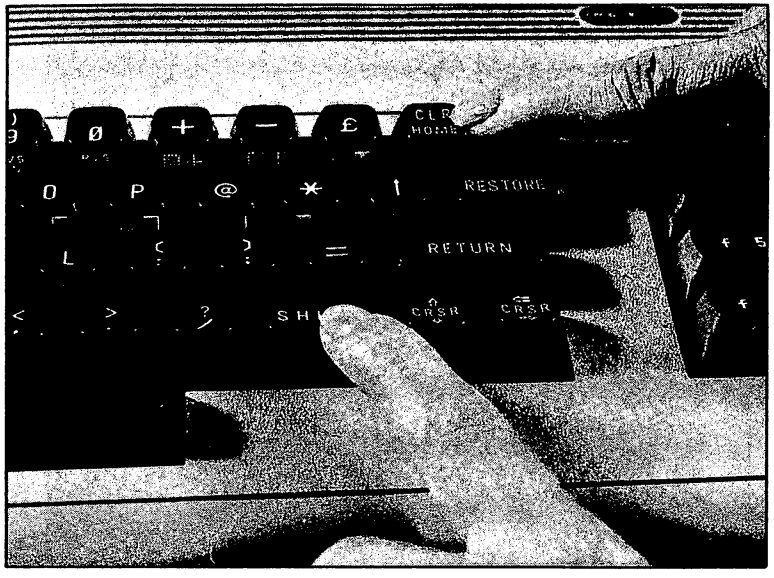


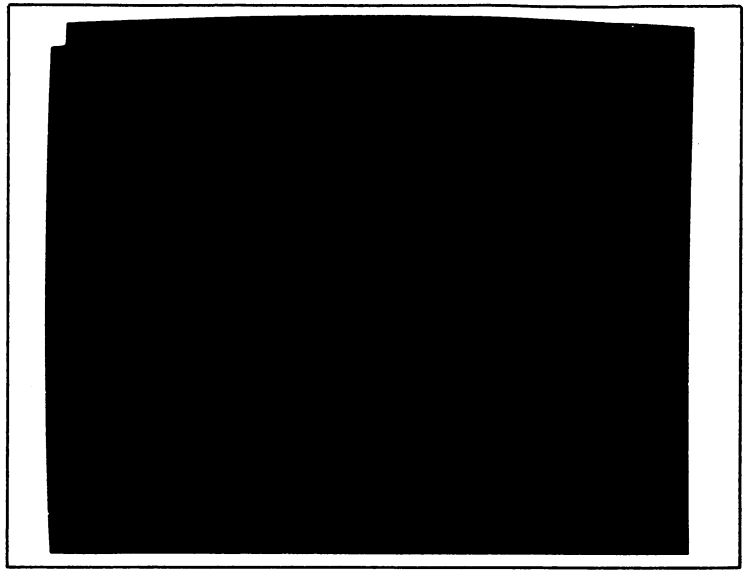
21. The top line advises that your C64 has Commodore Business Machines' BASIC language, Version Two installed. The next line indicates the amount of memory which is installed and which is available to you. Only 38,911 bytes of random access memory (RAM) of the installed 64K bytes are accessible. Some memory is taken up by the internal workings of the computer. The next line tells you that the computer is READY for use. The blinking white box below that is called a *cursor*. It indicates the screen position which will be taken by any character you may input from the keyboard.

THE KEYBOARD

Screen Clearing

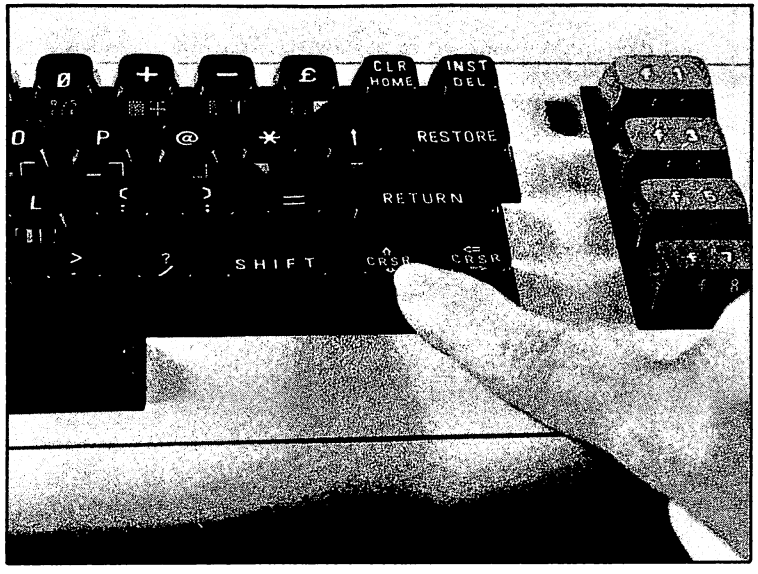
22. The C64 keyboard layout is somewhat similar to that of a typewriter, although the differences are profound and touch typists will have a hard time adjusting to them. They should be particularly wary of the use of the right-hand SHIFT key.

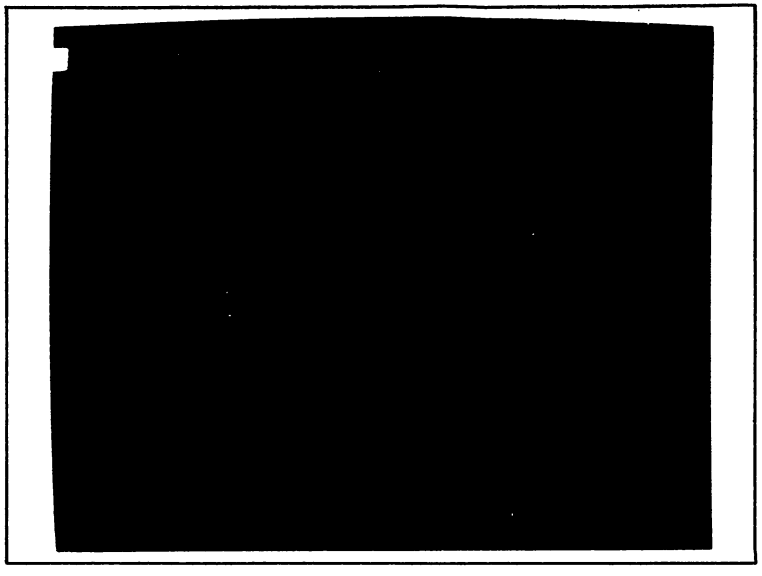




23. Now that we know the language the computer speaks, the amount of memory we have available, and that the machine is ready for use, the material printed on the TV screen is of no further value. Let's get rid of it so that we can begin with an uncluttered screen. To do so, press the SHIFT key and hold it down while you press the CLR/HOME key.

24. That should result in your screen looking like this. It's like putting a fresh sheet of paper in a typewriter.

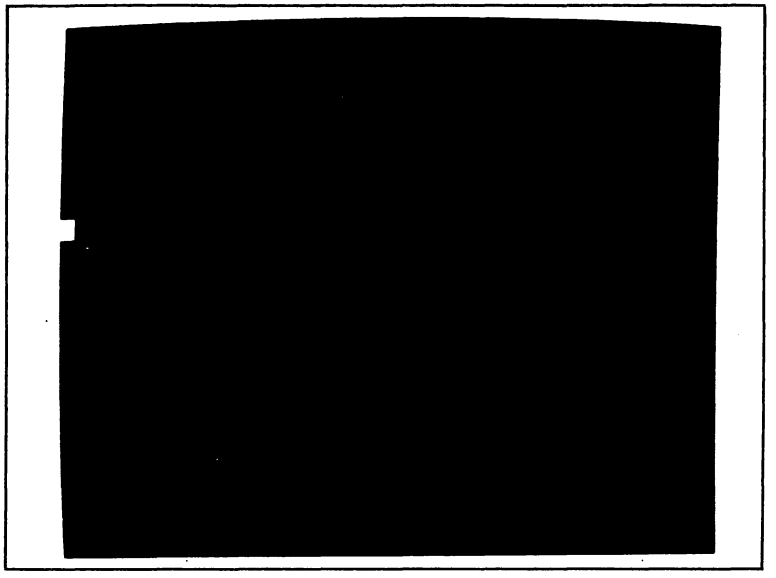




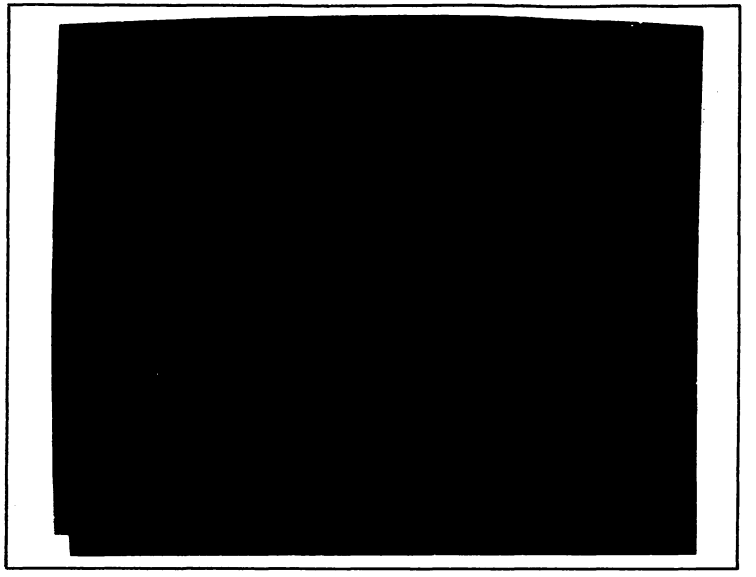
Moving the Cursor

25. Clearing the screen has automatically placed the blinking cursor in a position in the upper-left corner of the screen. This location is called the home position. You can move the cursor to any position on the screen through the use of two keys. To see how this is accomplished, press the vertical CRSR key once.

26. Notice that the cursor has moved one line (row) down the screen.

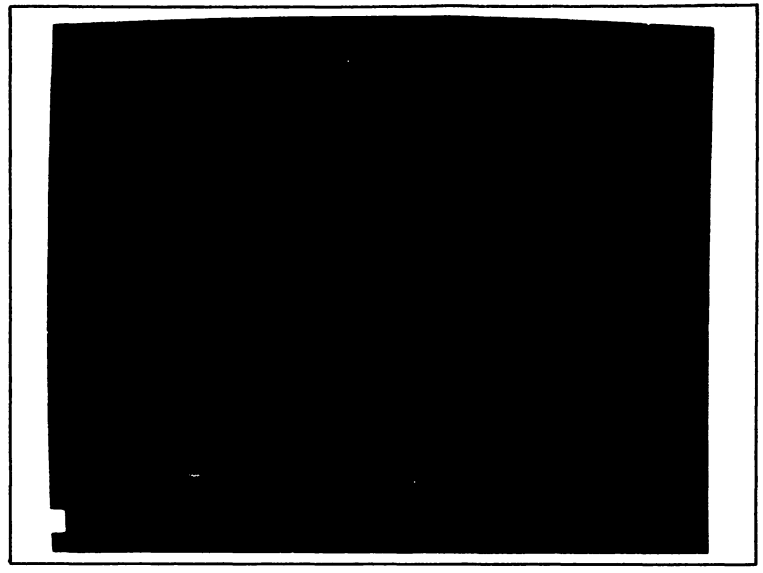


27. Now if you press and hold the vertical CRSR key down for a moment, you will notice that the key has an automatic repeat feature and that the cursor continues to move toward the bottom of the screen for as long as you hold the key down.

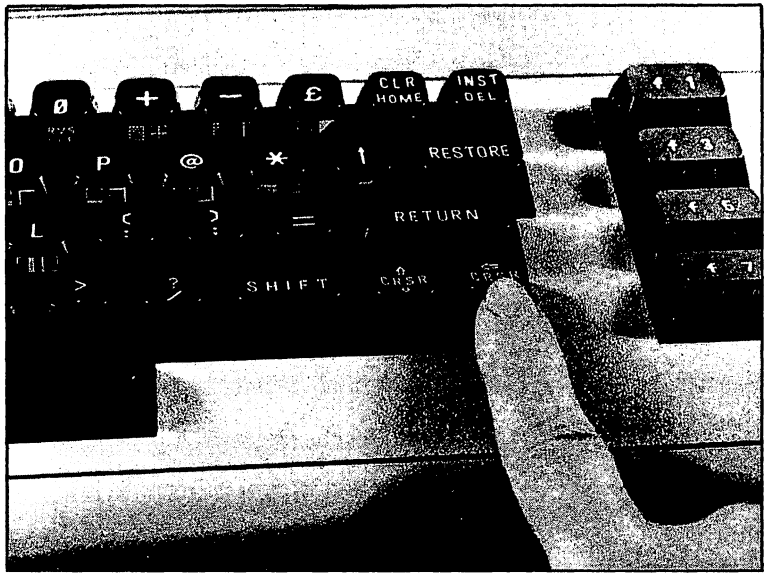


28. If you continue to hold the key down, the cursor will move to the last row, at the bottom of the screen, and remain there, flashing on and off.

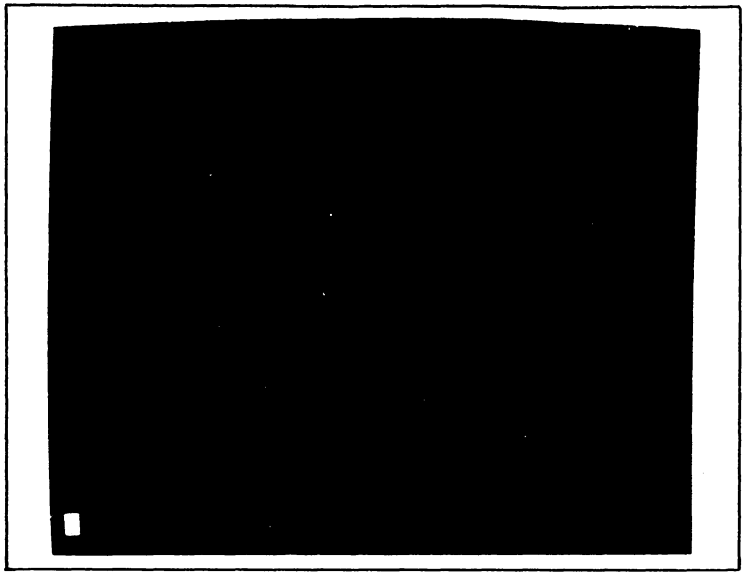
29. Press the SHIFT key and hold it down while you press the vertical CRSR key once.



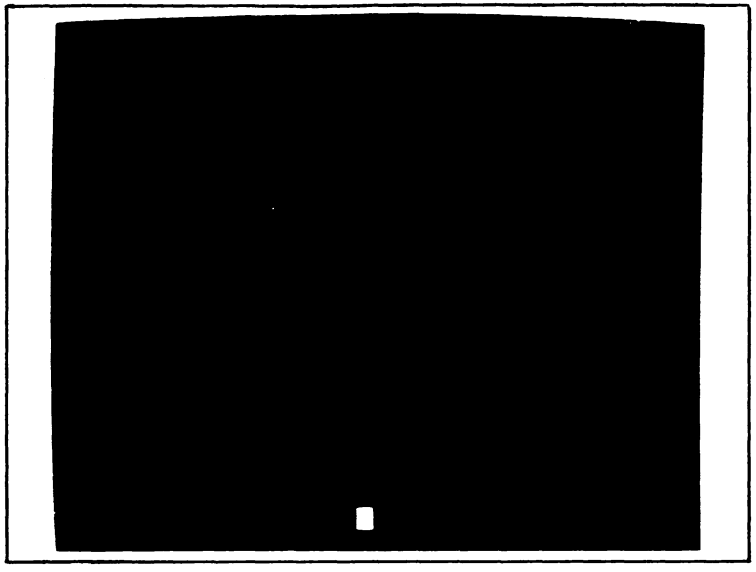
30. Notice that the cursor has moved up one row. Using the SHIFT key in conjunction with the CRSR key reverses the direction in which the cursor is moved. The repeat feature is also available when the CRSR key is used with the SHIFT key.



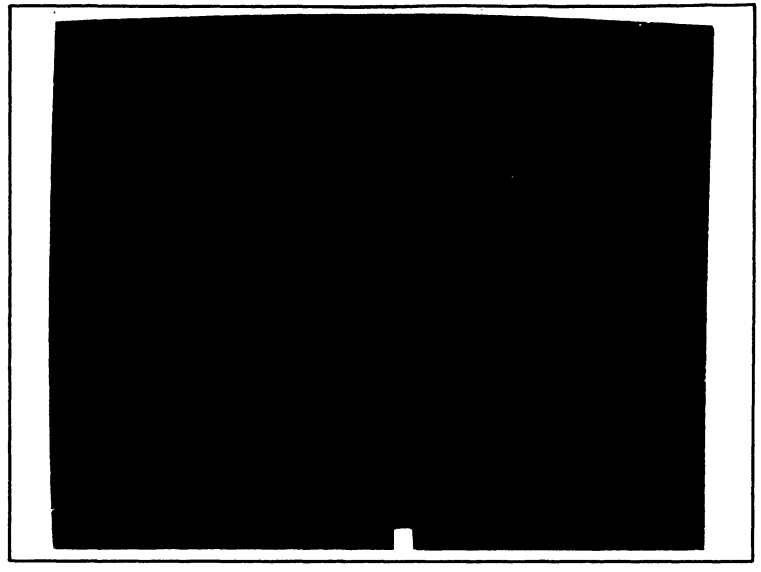
31. Now press the horizontal CRSR key once.



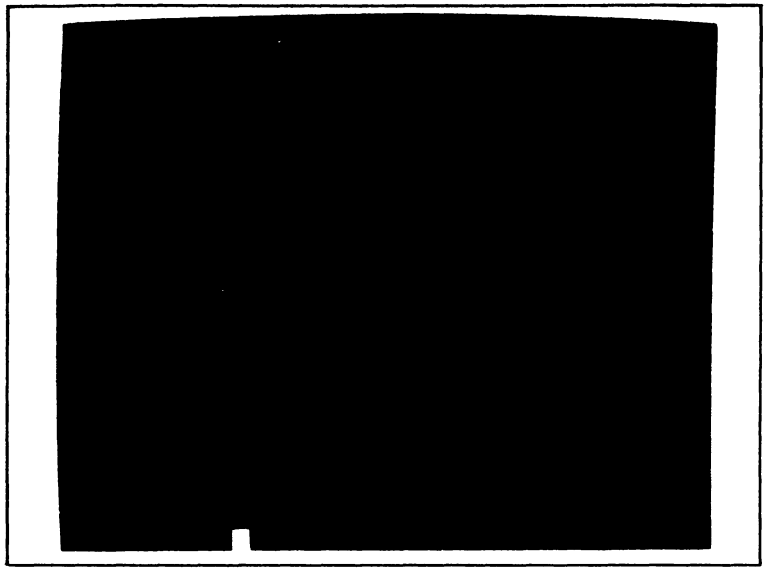
32. The cursor will move one column to the right.



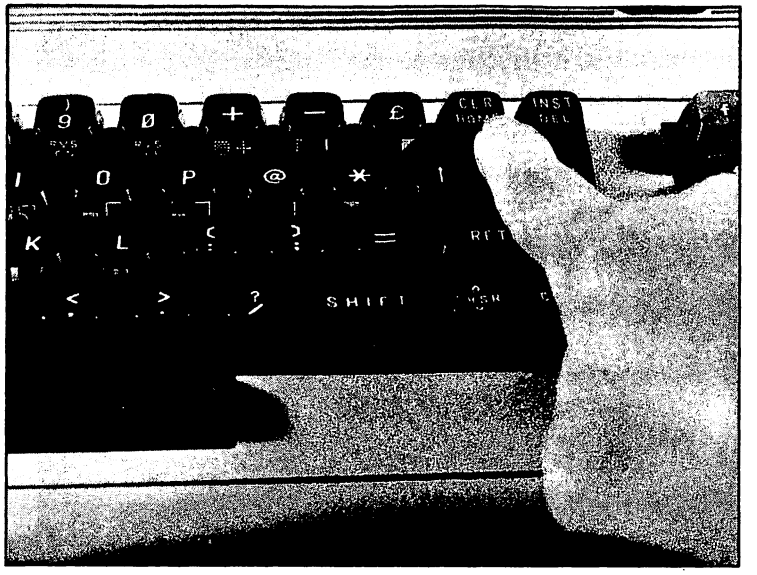
33. Press the horizontal CRSR key and hold it down for a moment and the cursor will move to the right.



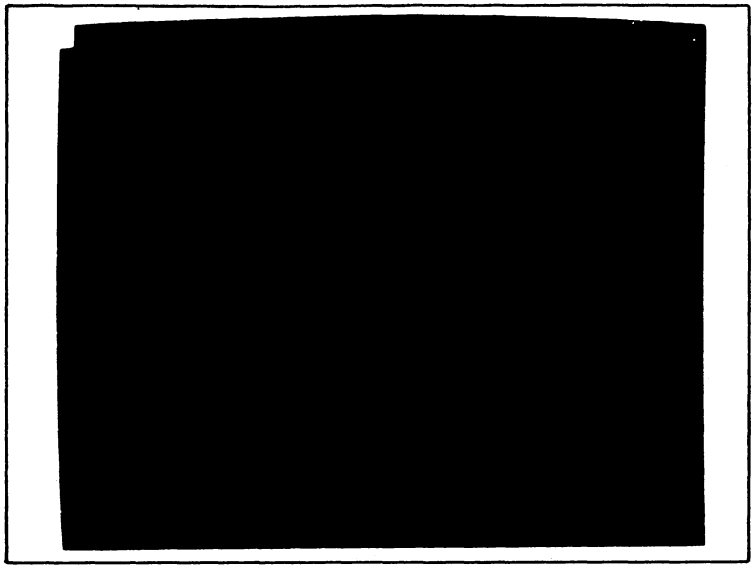
34. If you hold the horizontal CRSR key down long enough, the cursor will travel to the right screen border, disappear, and then reappear, one row later, at the left screen border, and continue back across the screen. It will keep doing this for as long as you keep the horizontal CRSR key depressed.



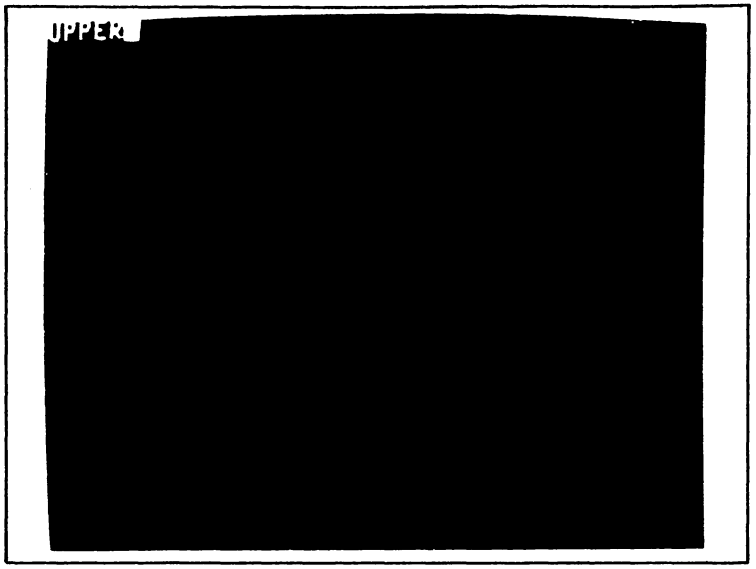
35. Now press and hold the SHIFT key down while you press the horizontal CRSR key and hold it down momentarily. Notice that now the cursor moves from right to left.



36. Regardless of where the cursor may be on the screen, a single press of the CLR/HOME key . . .



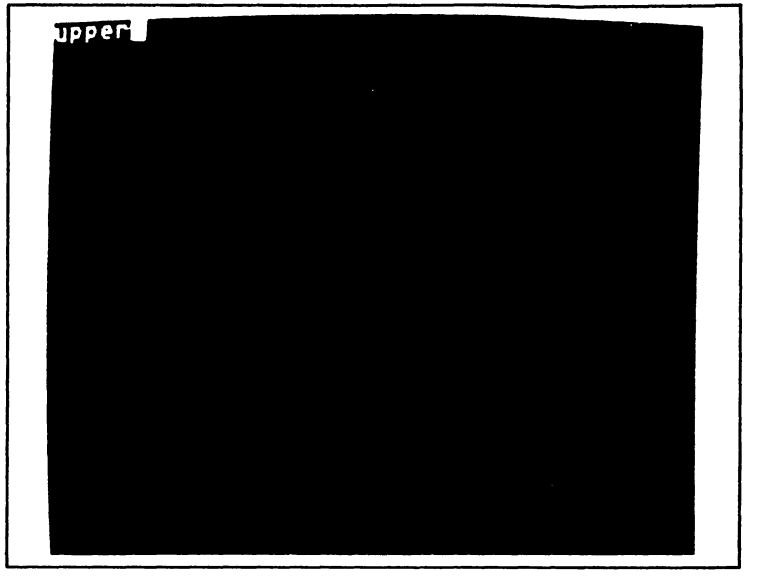
37. . . . will return the cursor to its home position.



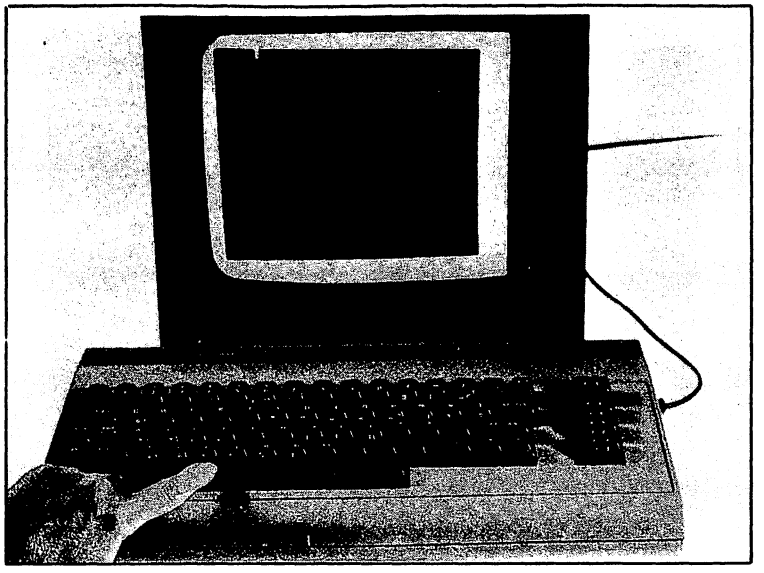
Selecting Upper or Lower Case

38. Press the following keys: U P P E R. Now your screen should look like this. Notice that each letter appeared in the location the cursor last occupied, and that the cursor moved over a column as each letter appeared.

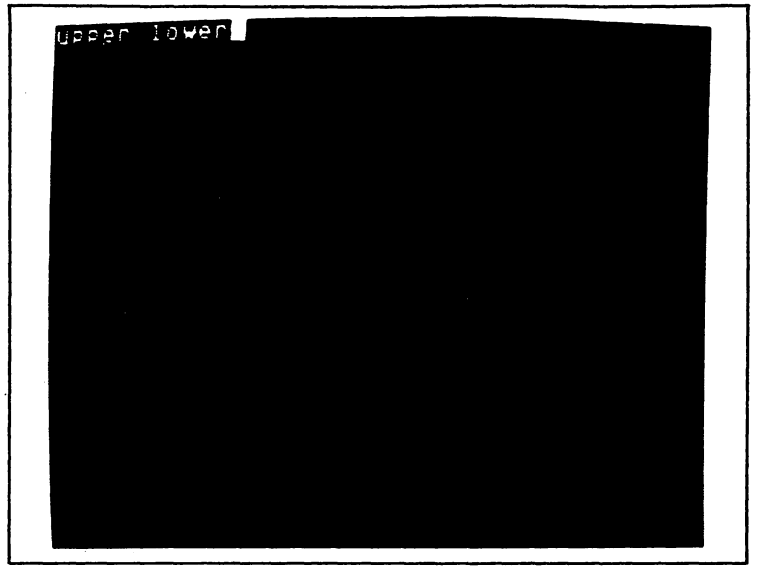
39. Press the COMMODORE and SHIFT keys simultaneously.



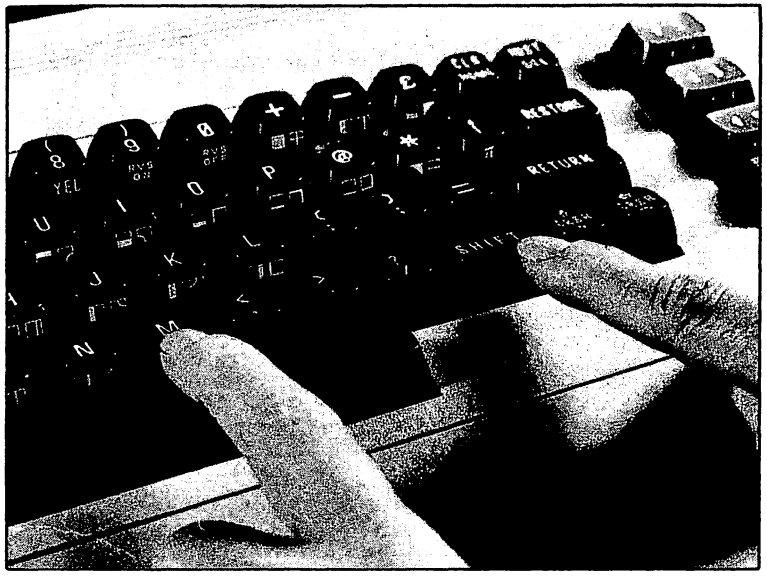
40. Now your upper has become a lower. All of the upper-case letters you just typed have been shifted to their lower case.



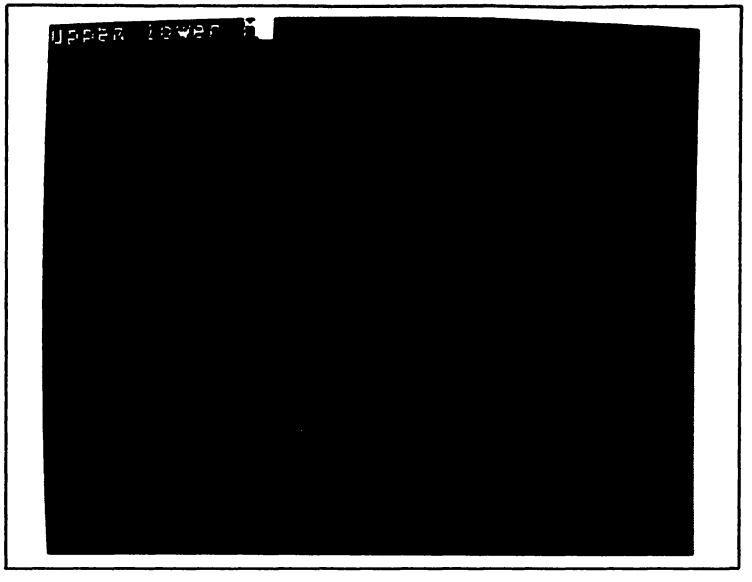
41. Press the SPACE bar once to move the cursor one space to the right. (The SPACE bar also has a repeat feature, so if it is held down, the cursor will keep moving to the right.)



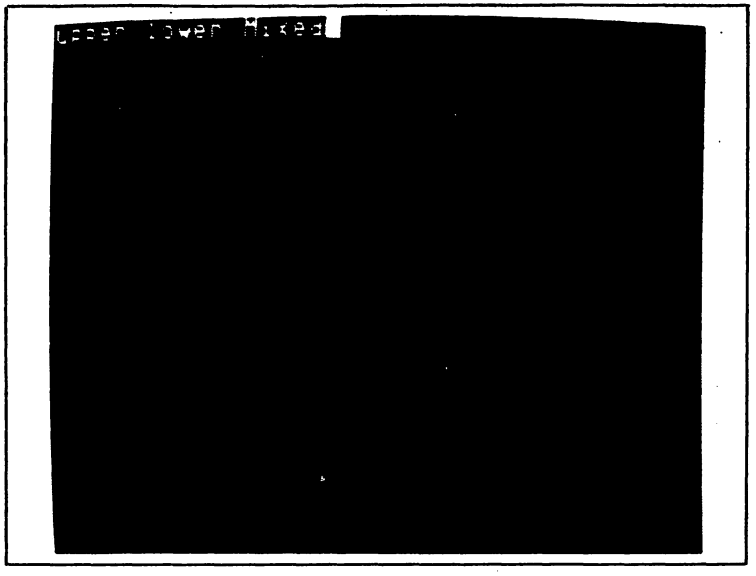
42. Now press the following keys: l o w e r. Your screen should look like this. All the characters you just typed have been printed on the screen in lower case, as they would be on a normal typewriter.



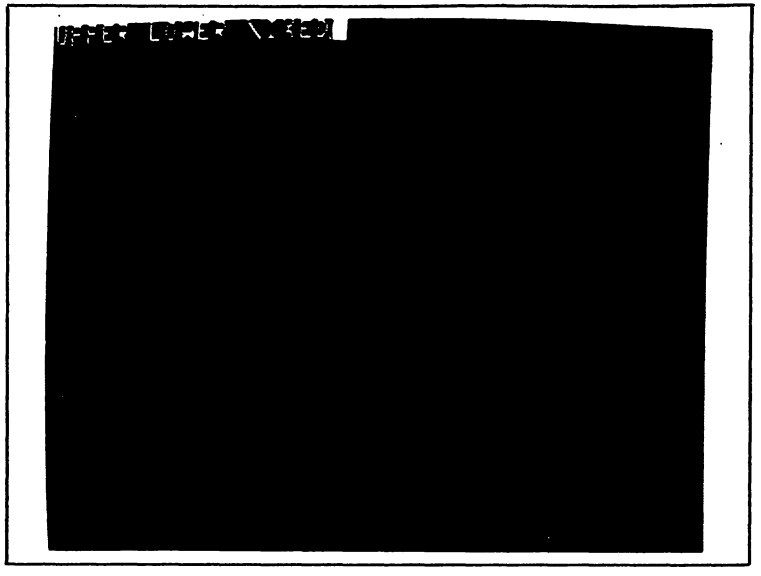
43. Press the SPACE bar once, then press and hold the SHIFT key while you press the M key.



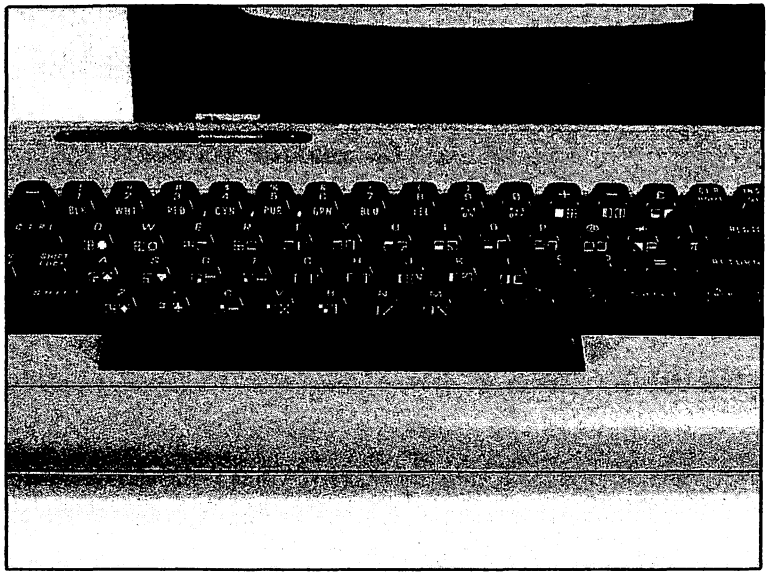
44. Notice that the M was printed on the screen as a capital letter. (With the C64 in its lower-case mode it operates like a typewriter.)

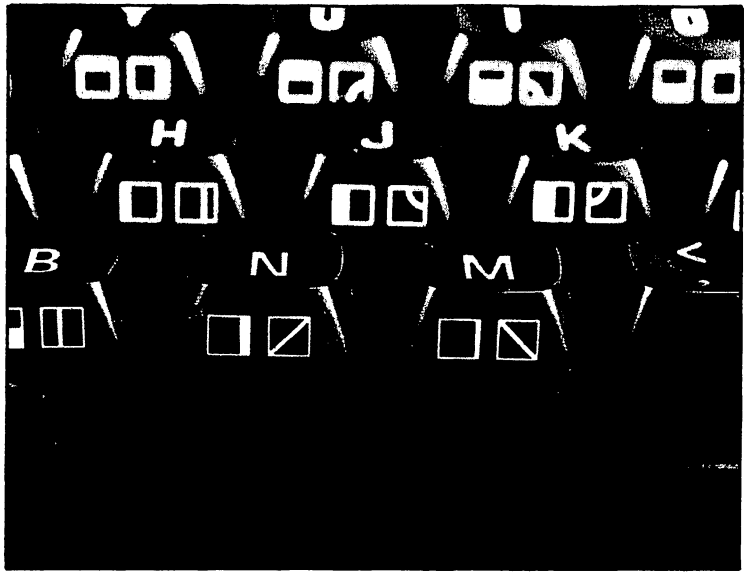


45. Now press: i x e d, and your screen should look like this.



46. You can shift back to the upper-case mode again, but you won't get exactly what you might expect. Try it. Press the COMMODORE and SHIFT keys at the same time and your screen will look like this.

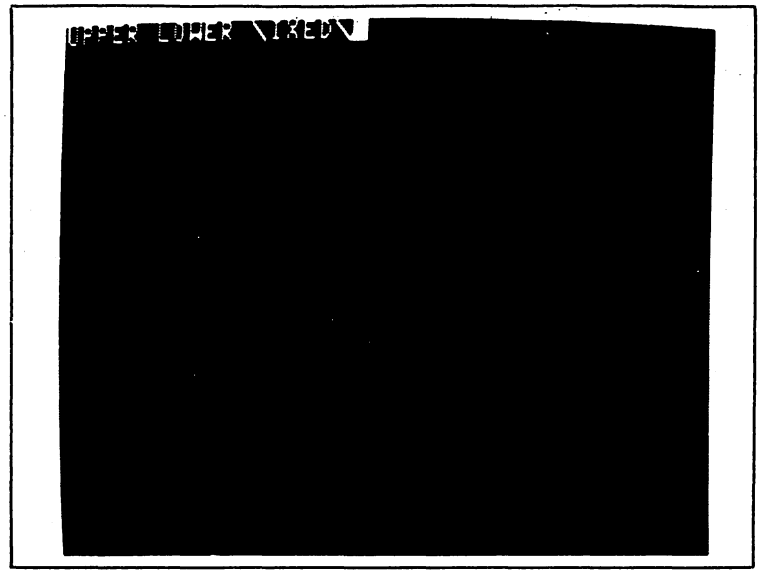




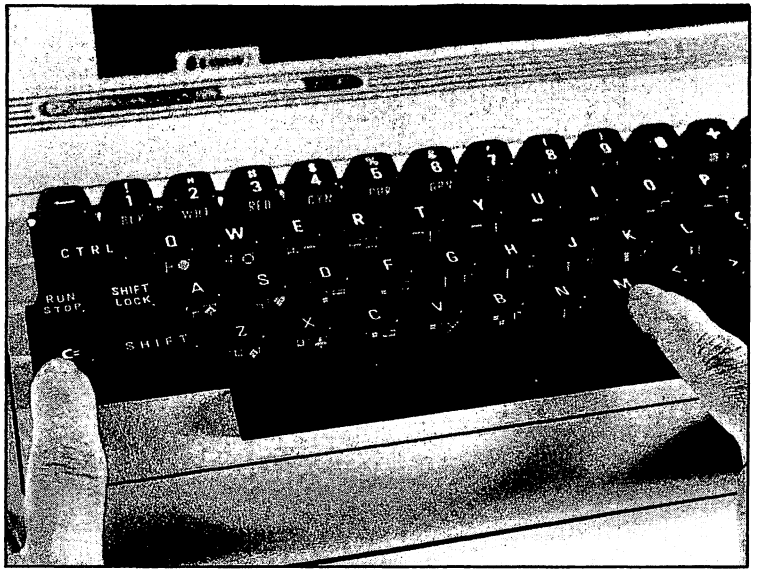
Graphic Symbols

47. The explanation for this peculiar state of affairs has to do with the computer's ability to produce on the screen all of the graphic characters shown on the key fronts.

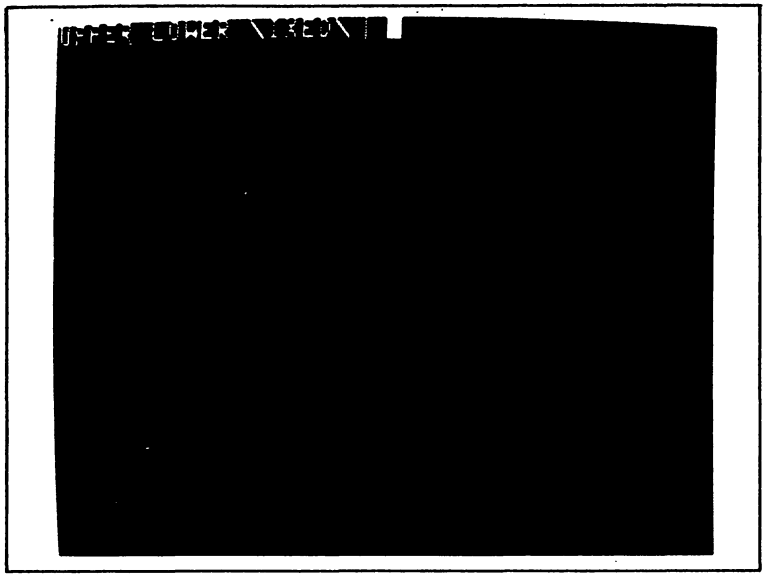
48. If you look at the front of the M key, you will notice two boxes. The box on the right shows a diagonal running from the lower-right corner to the upper-left corner. If you now press the SHIFT and the M keys at the same time, that is just what will appear on the screen.



49. Now your screen looks like this.



50. If you press the COMMODORE key while you press the M key you will get a thin vertical line printed on the screen. (See the right side of the left box on the front of the M key.)

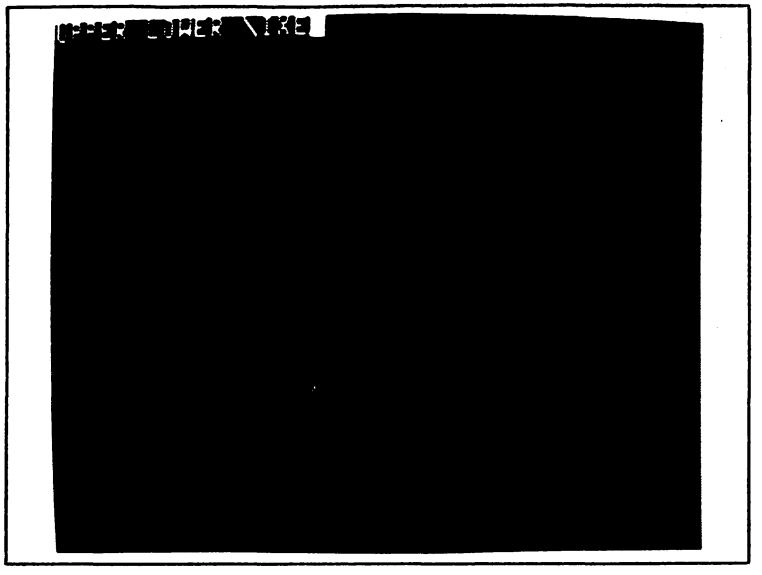


51. I've moved the cursor one space to the right to show you the vertical line more clearly here.

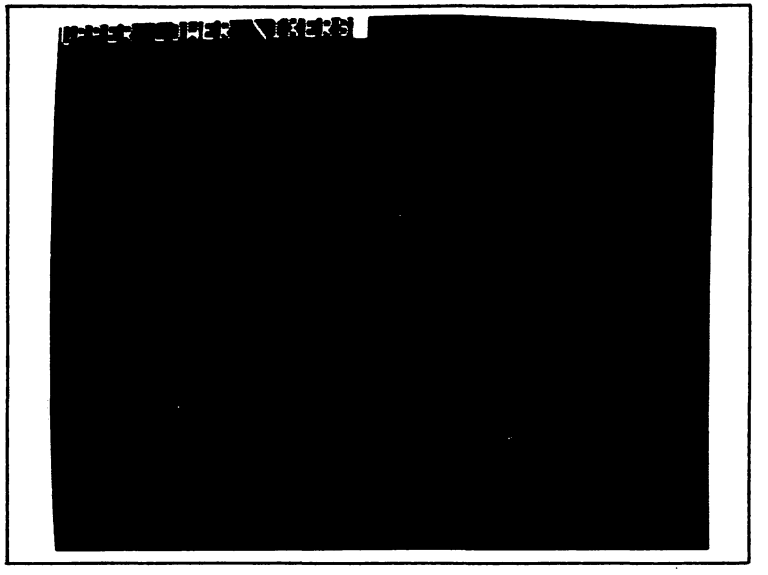
NOTE: When you turn the computer on, it will automatically begin to operate in its upper-case mode. In that mode you have the option of putting on the screen all the available characters, punctuation marks, symbols, and so on, that appear on the key tops, as well as all the graphic symbols that appear on the key fronts. If you want to use mixed cases you should shift the machine into its lower-case mode immediately, before doing anything else. In the lower-case mode , only the characters, punctuation marks, symbols, and the like, which appear on the key tops can be put on the screen. Always remember that you will get some very peculiar results if you decide to shift back from lower-case to upper-case operation.

Making Corrections

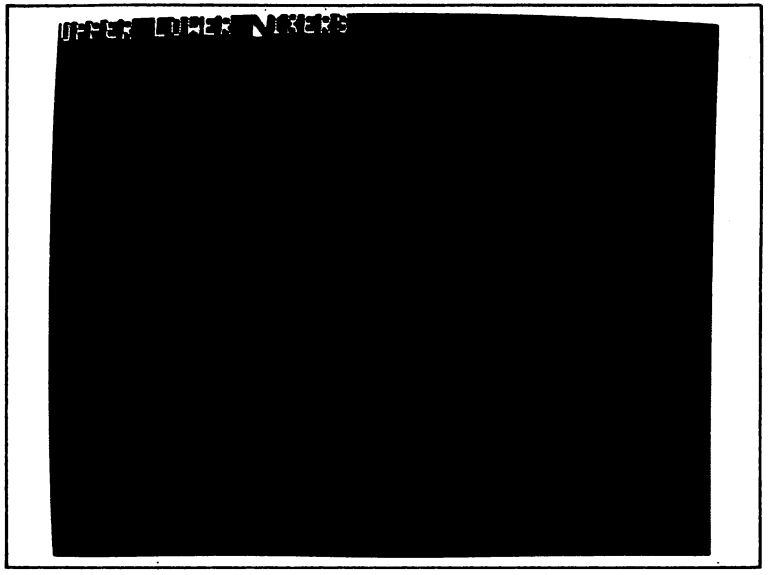
52. The line on the screen in illustration 51 presents a perfect opportunity to illustrate how easily errors can be corrected with the C64. Let's change this line to read: UPPERMOST LOWLY MIXERS. Begin by pressing the INST/DEL key four times.



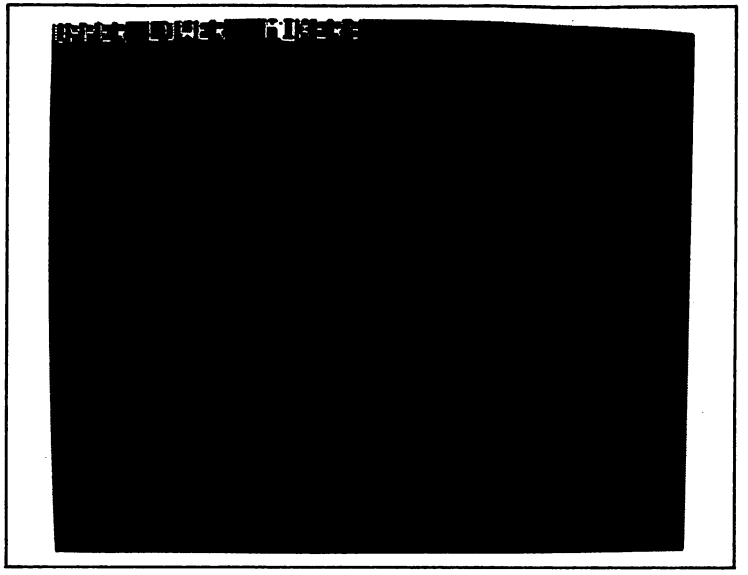
53. Doing so will erase the vertical line, the diagonal, and the D, so that your screen now looks like this.



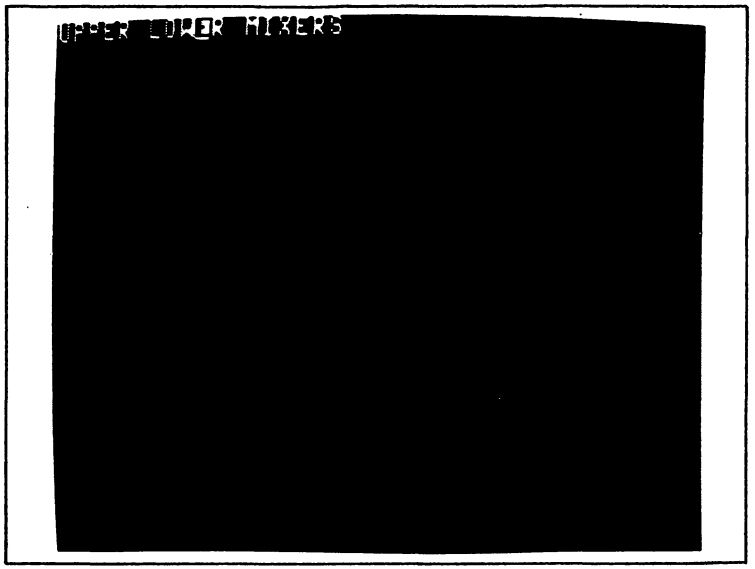
54. Press the R and S keys to bring your screen to this.



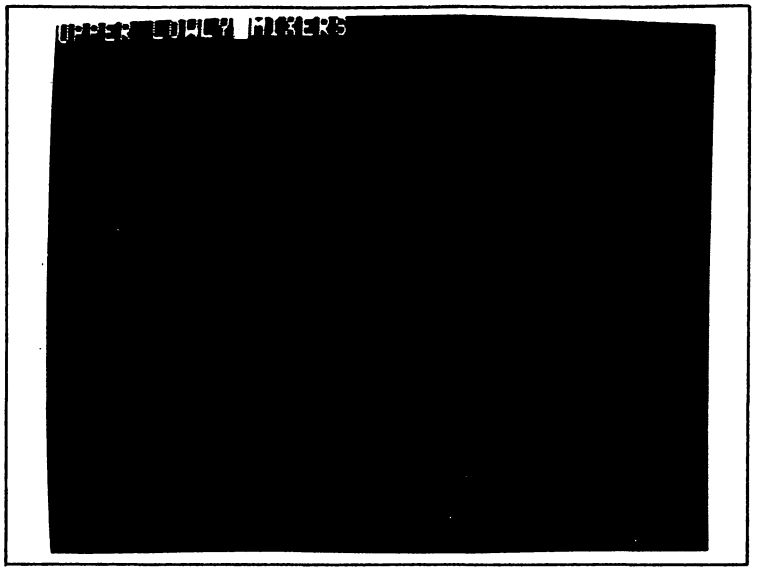
55. Now press the SHIFT and horizontal CRSR keys simultaneously to position the cursor over the diagonal.



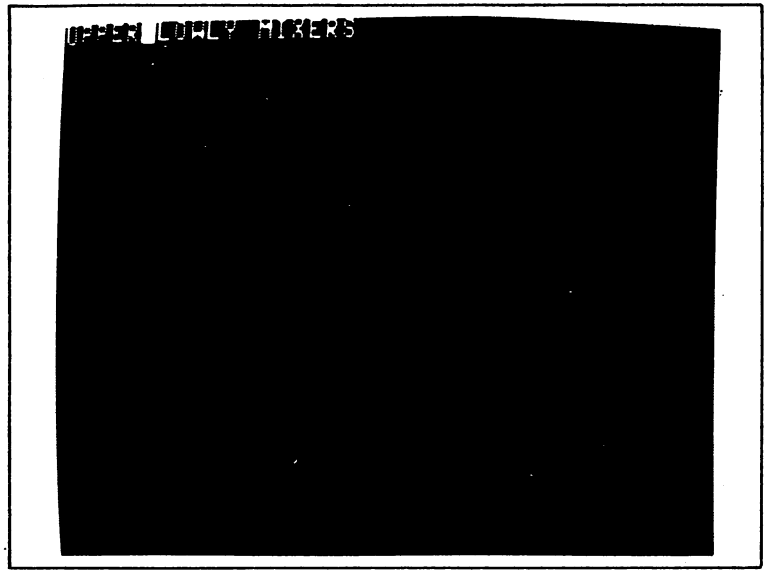
56. Press the M key and you have corrected the final word.



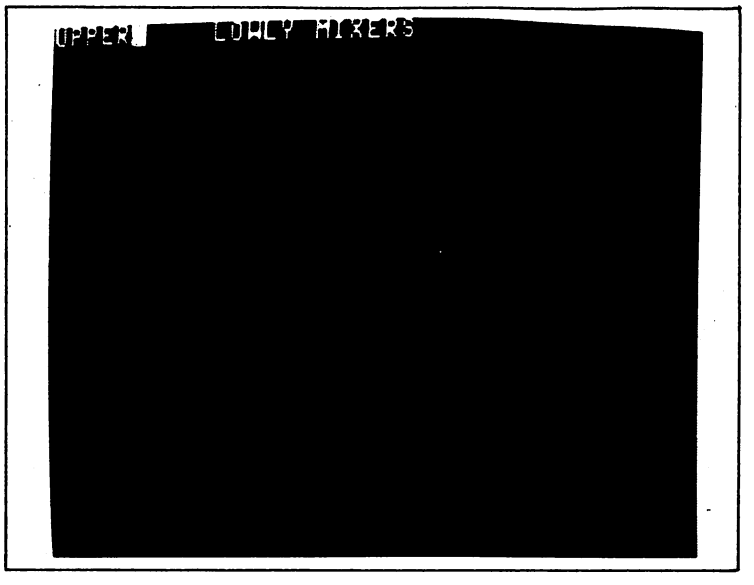
57. Now press the SHIFT and horizontal CRSR keys four times to bring the cursor over the E in LOWER.



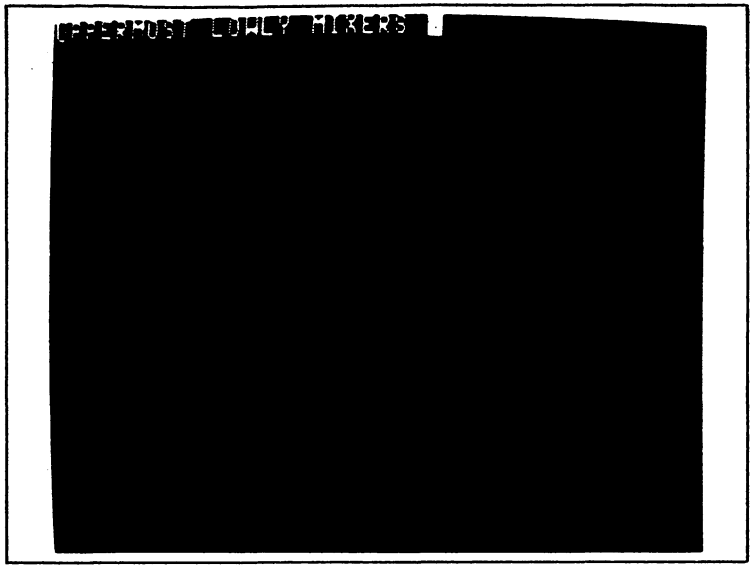
58. Press the L and Y keys and those letters will replace the ones already on the screen.



59. Use the SHIFT and horizontal CRSR keys to position the cursor between UPPER and LOWLY.

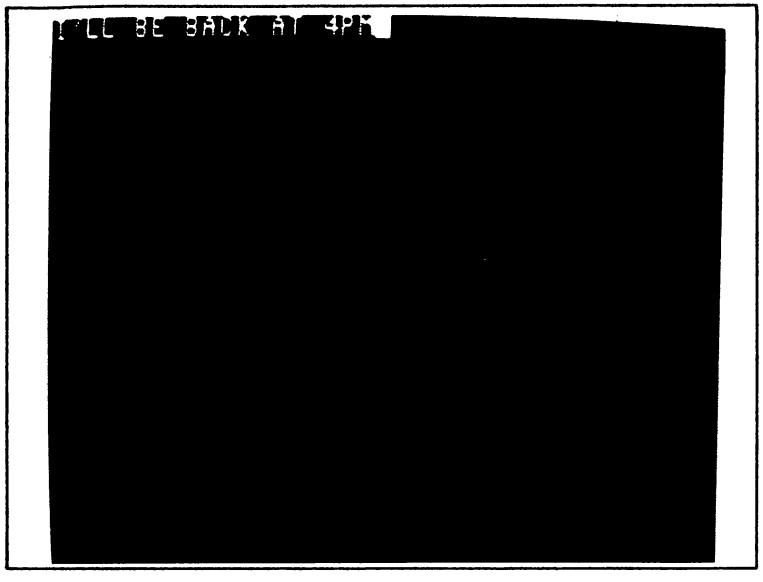


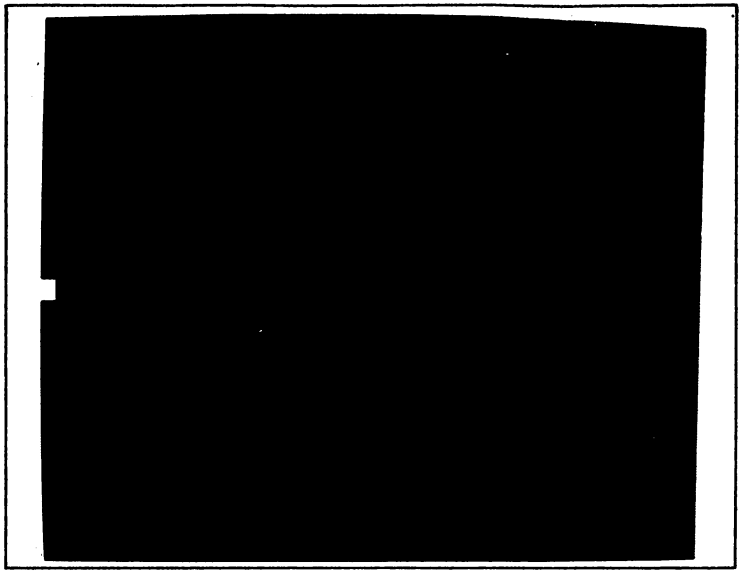
60. Press and hold the SHIFT key and press the INST/DEL key four times to provide space enough to INSerT the needed correction.



61. Finally, type in the characters: M O S T, and you have completed the corrections to the line. You can move the cursor out of the way by pressing either of the CRSR keys to allow you to better admire your work.

62. Remember, you can always clear the entire screen by pressing the SHIFT and CLR/HOME keys.





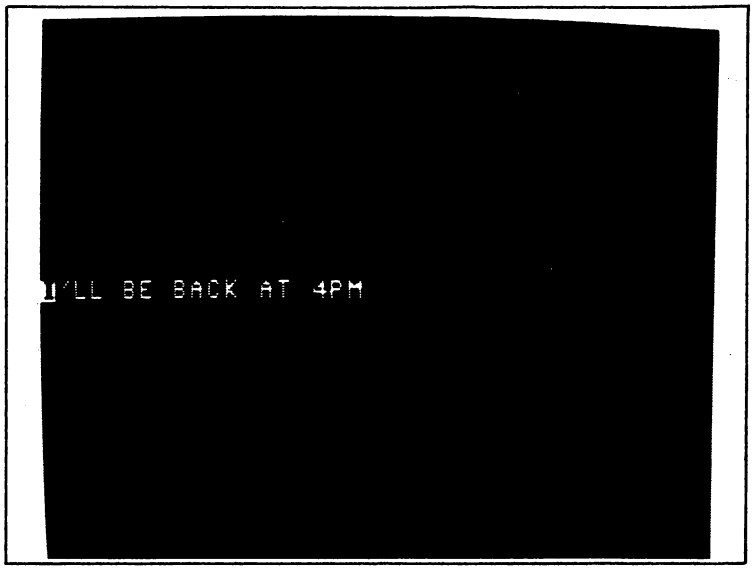
TELLING YOUR COMPUTER WHAT TO DO—IMMEDIATELY

Using the Computer as a Bulletin Board

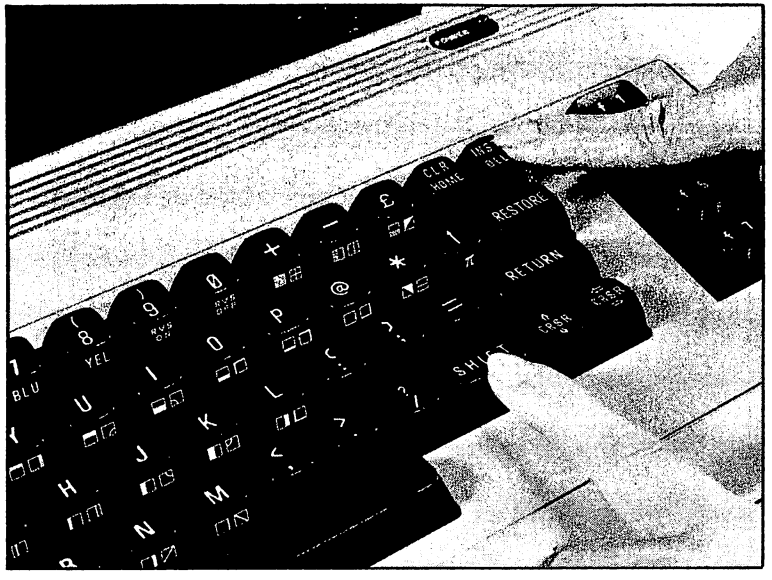
63. The C64 has two modes of operation—immediate and delayed. One of the applications for the immediate mode is to use the machine as an electronic bulletin board. Let's suppose you have an important message you wish to bring to someone's attention. You could simply type it into the computer and leave it on the screen for that person to see. For example, type in: I'LL BE BACK AT 4PM.

64. That's not bad. But your message might get more attention if it was placed right in the middle of the screen. It is easy to put it there. First clear the screen by pressing SHIFT and CLR/HOME. The screen is divided into 25 rows and 40 columns, so to place your message in the center of the screen, begin by moving the cursor down the left side of the screen to row 12 by pressing the vertical CRSR key 12 times.

65. Then type your message in. That looks as if it might attract a little more attention, doesn't it?

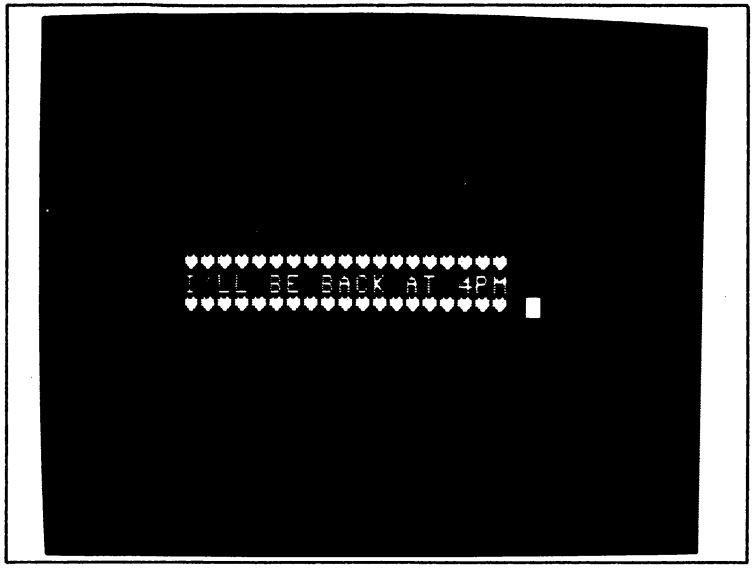


66. If you are a compulsive neatnik, you can center the line by first moving the cursor back to the beginning of the line, positioned over the letter I.

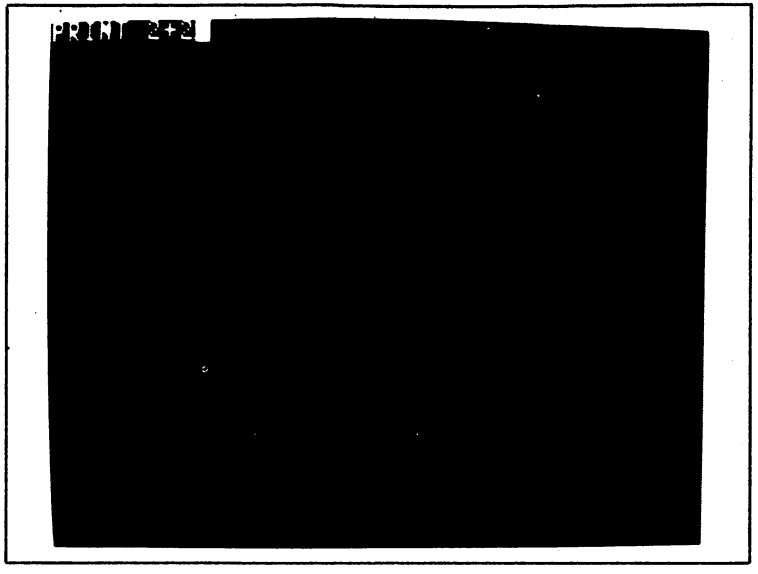


67. Now press and hold the SHIFT key down while you press the INST/DEL key nine times.

68. That looks neater, doesn't it?



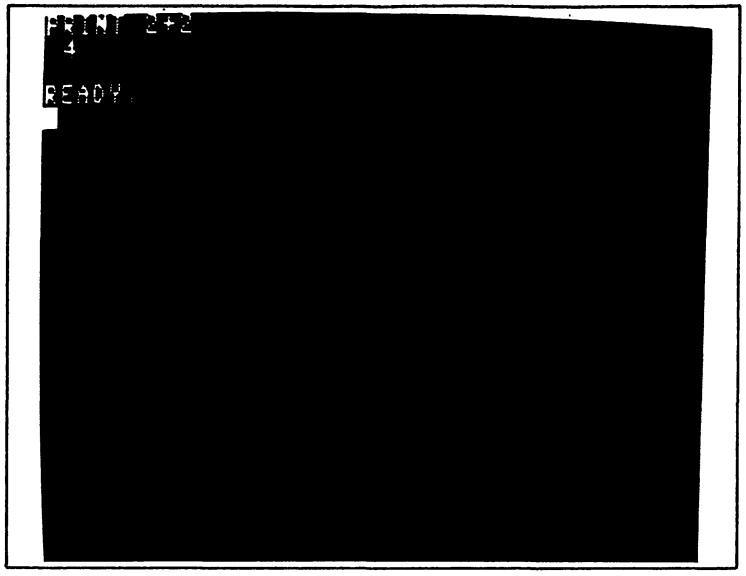
69. You are not limited to one-liners when you use the computer as a bulletin board. You can type as long a message as the screen can hold (that's 1,000 characters, or about 167 words). If you have the patience and artistic talent, you can also use the graphic elements on the key fronts to draw designs, pictures, or diagrams as part of your message.



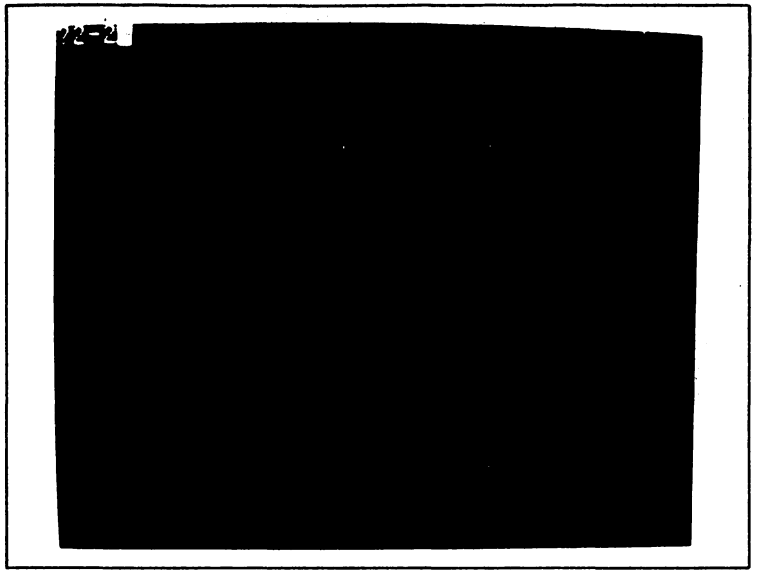
Using the Computer as a Simple Calculator

70. Perhaps a better application for the C64, in the immediate mode, is as a calculator. To try this application, lets add two and two. First clear the screen, then type in: PRINT 2+2. You'll find the plus sign on the top row of keys next to the zero.

71. Press RETURN . . .



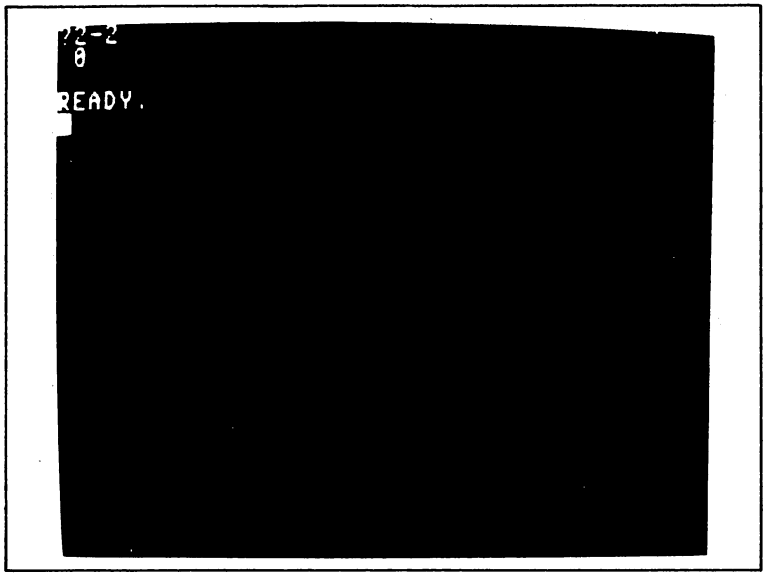
72. . . . and your screen will look like this. The answer, 4, appears on the line below your command. The computer, having done your bidding, then goes on to announce that it is ready for your next command.



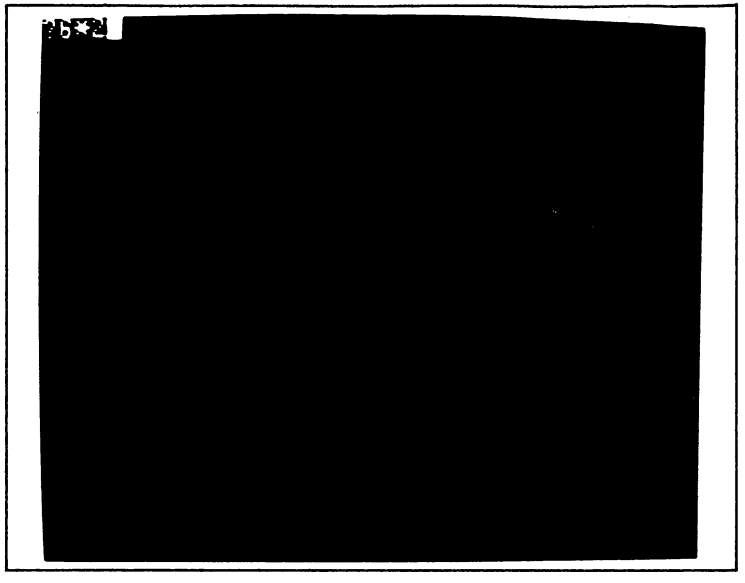
73. Let's check out the machine's ability to do some more very basic arithmetic by subtracting two from two. This time, however, we will use a time-saving abbreviation. In BASIC, the language the computer understands, the question mark is used as an abbreviation for the command PRINT. (To get it, press SHIFT as you press the question mark key.) Clear the screen by pressing the SHIFT and CLR/HOME keys and type in: ?2−2.

NOTE: Virtually all the BASIC key words can be abbreviated with the C64 by typing the first letter of the key word normally, and then typing the second letter SHIFTed. The two exceptions are PRINT (which is abbreviated with a question mark) and INPUT (which has no abbreviation).

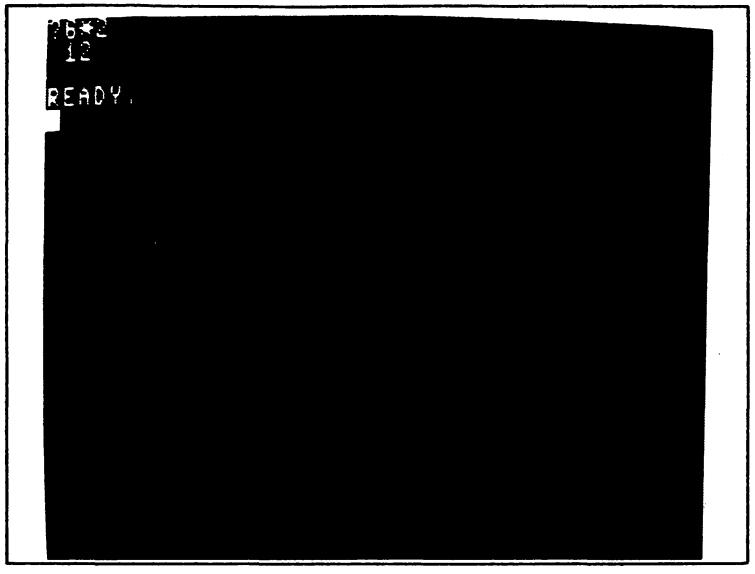
74. Press RETURN.



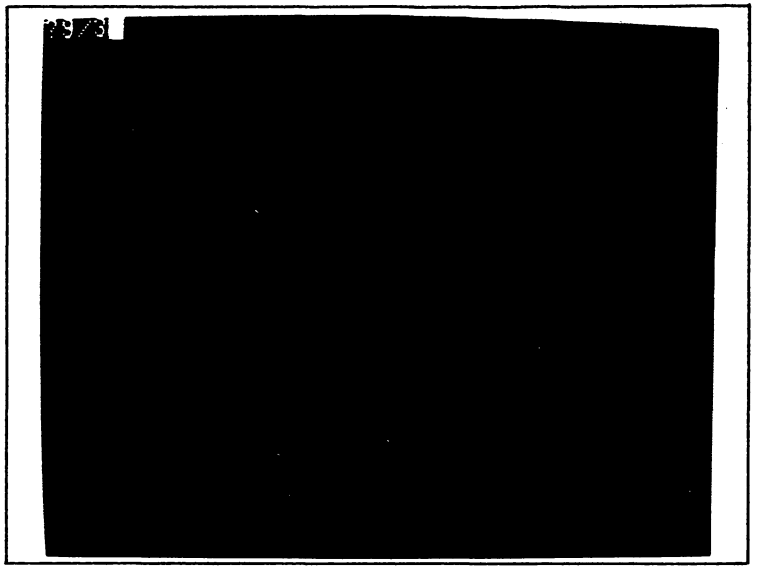
75. The computer gives you the answer, 0, on the next line, and then goes on to tell you it's ready for more.



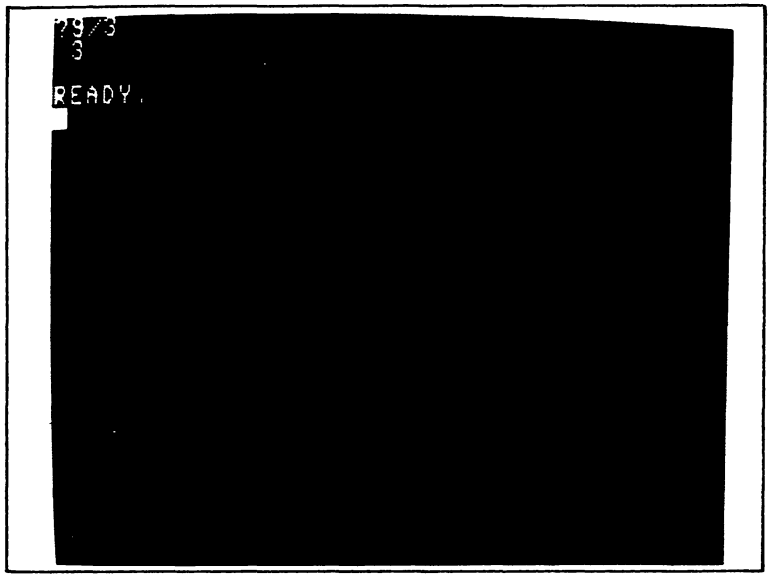
76. This time let's ask the machine to multiply six times two. Clear the screen, then type in: ?6*2. You'll find the asterisk (*) third from the right in the second row of keys from the top. In BASIC the asterisk is used (instead of the more familiar X) as the multiplication sign.



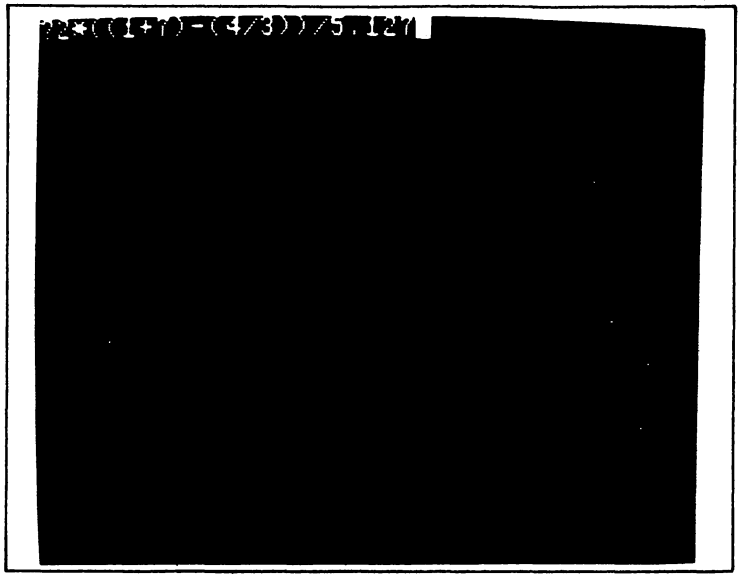
77. Press RETURN and your screen will look like this.



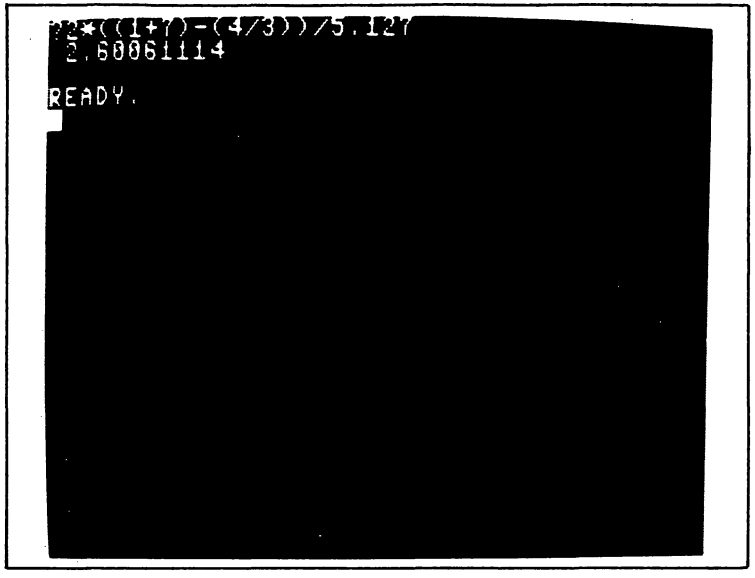
78. In BASIC the slash (/) represents the divide-by sign. To divide nine by three, first clear the screen, then type in: ?9/3. You'll find the slash on the bottom row of keys, adjacent to the right SHIFT key.



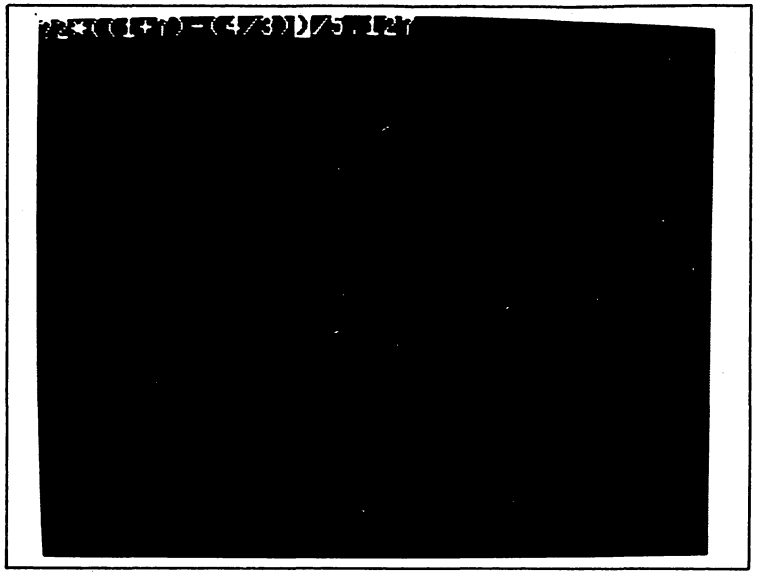
79. Press RETURN and your screen will look like this.



80. If your math is up to it, you can use standard algebraic notation to solve rather complex problems in the immediate mode. For example, first clear the screen. Then type in: ?2*((1+7)−(4/3))/5.127.



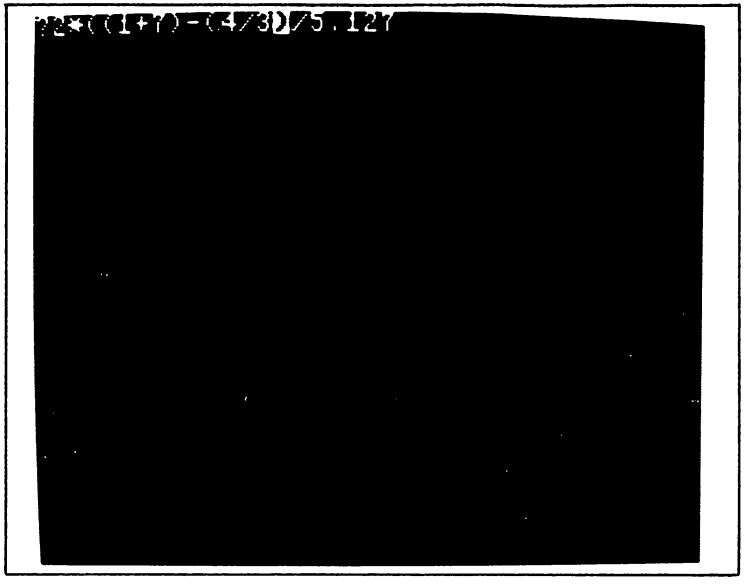
81. Press RETURN and the answer, 2.60061114, will appear, and the computer will be ready for the next problem.



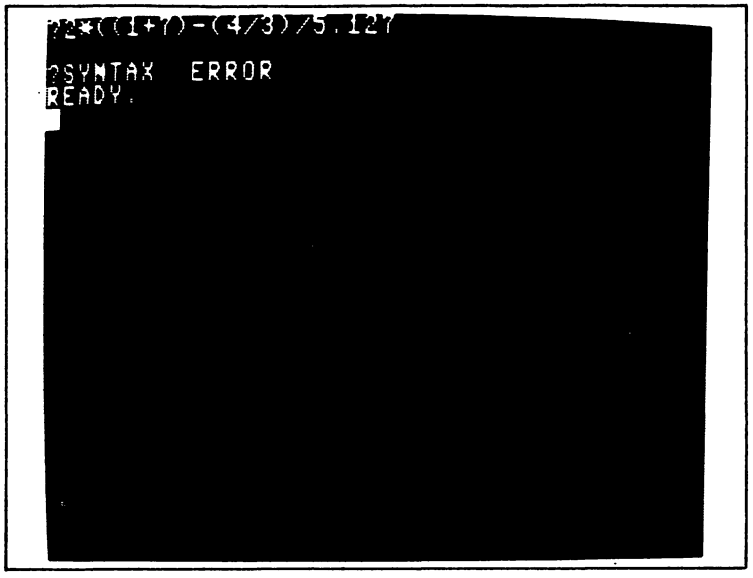
Syntax Errors

82. All computers are extremely sensitive about the way you talk to them. They can be more finicky than lawyers. If you don't speak precisely, they will absolutely refuse to hear what you have to say. When you make such a communications error talking to the C64 it will tell you you have made a syntax error and refuse to do your bidding. To see this, press and hold the INST/DEL key down until READY and the answer have been erased. Then move the cursor, by means of the two CRSR keys and the SHIFT key, until it is over the last parenthesis in the equation the machine just solved, as shown here.

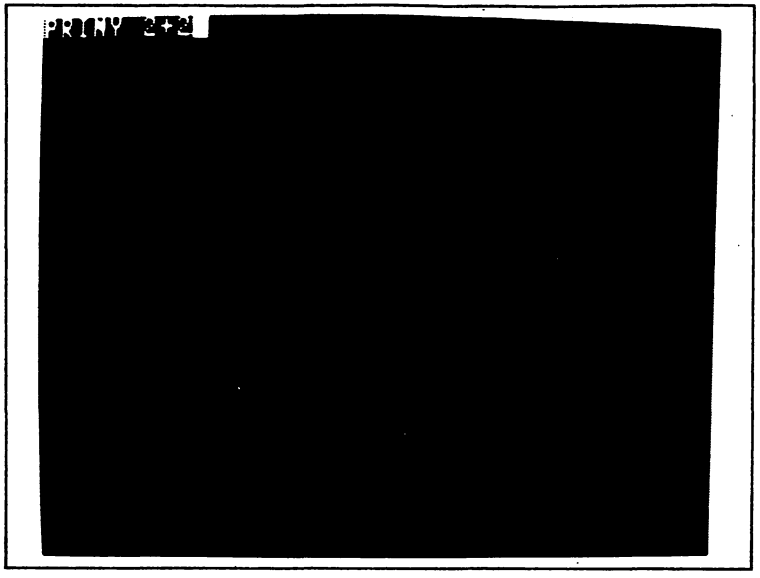
83. Press the INST/DEL key once to erase the last parenthesis.



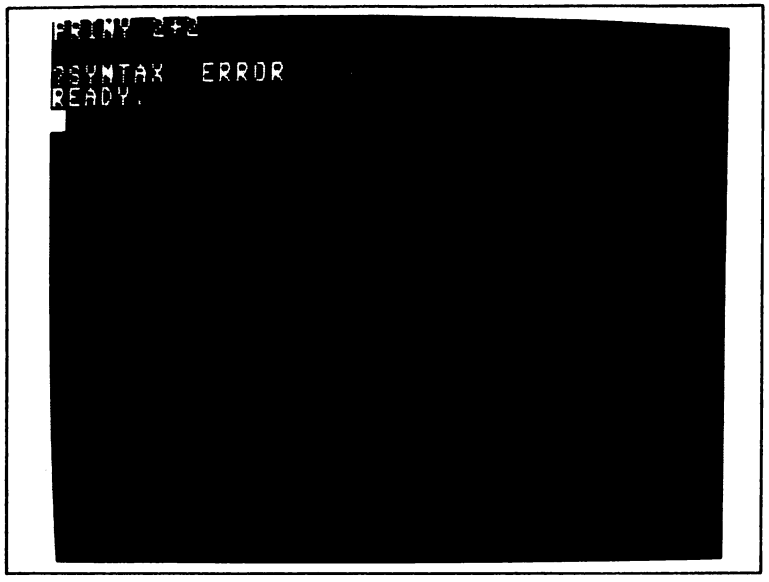
84. Now your screen will look like this, and the equation will be an invalid one which the machine cannot solve.

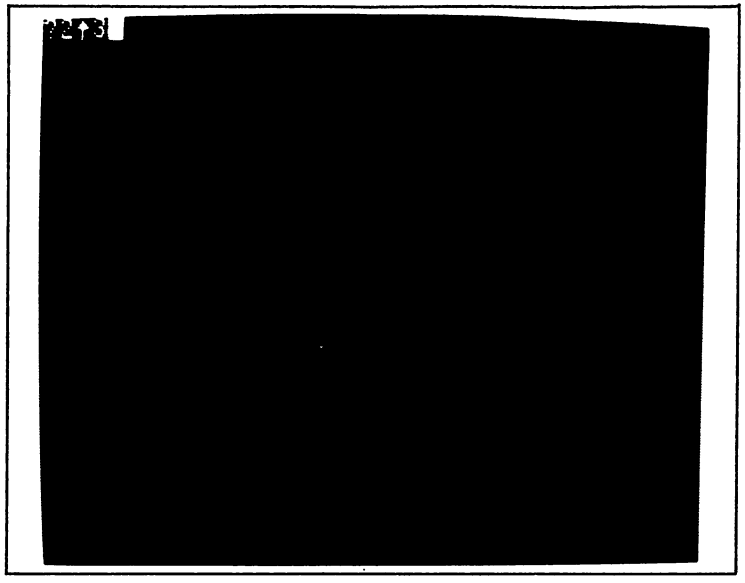


85. Press RETURN and the machine will politely tell you to go fly a kite. The computer reports that you have made a syntax error, but it doesn't tell you where or what kind. You have to discover that for yourself.



86. Even simple spelling or punctuation errors will cause the C64 to reject your efforts. For example, let's first clear the screen and try to add two and two again, but this time type in: PRINY 2+2.



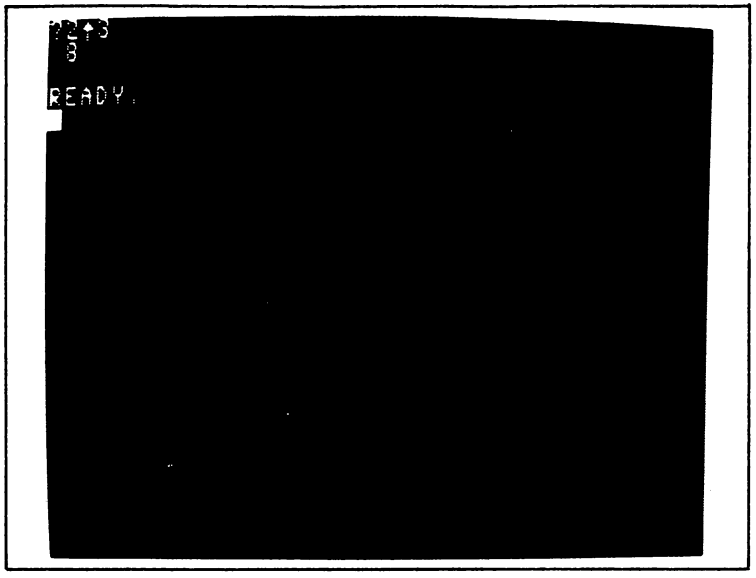


87. Press RETURN and the computer will again tell you, via its SYNTAX ERROR message, that it can't do what you asked it to do because it can't understand what it is that you want it to do.

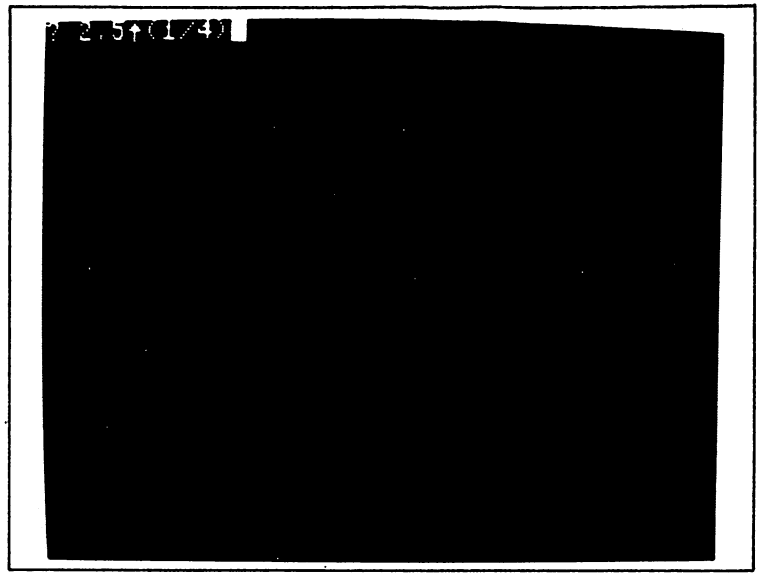
SUGGESTION: Move the cursor up to the offending letter Y, change it to a T, and press RETURN, to convince yourself that the C64 is just looking out for your best interests when it refuses to accept gibberish. Always remember the great computer truism: *garbage in = garbage out.* Be careful of what you tell your computer.

Using the C64 to Solve More Complex Problems

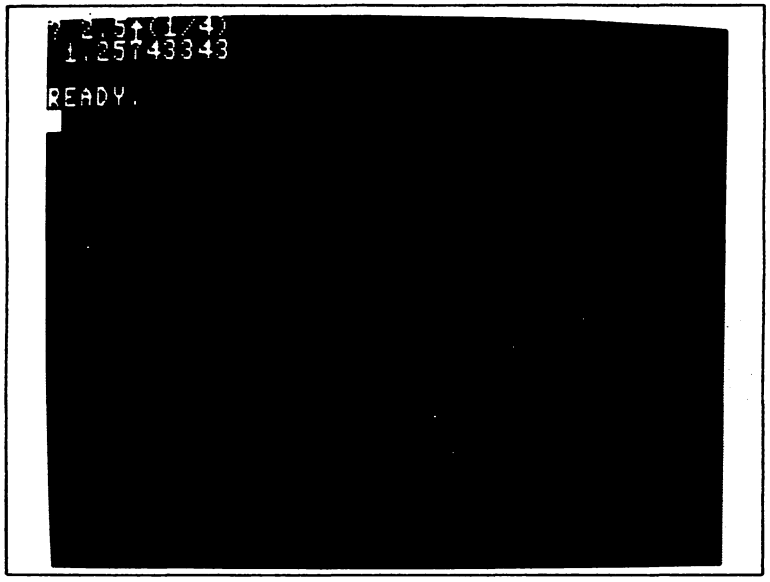
88. To raise a number to a power, an arrow symbol is used. It is the up-arrow found near the right end of the second row of keys. For example, to raise 2 to the third power (2 cubed) clear the screen, then type in: ?2↑3.



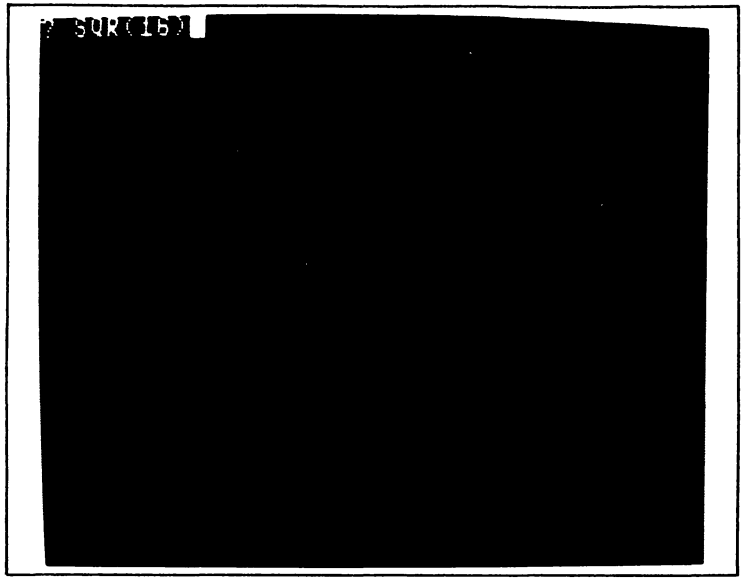
89. Press RETURN and the answer, 8, is immediately provided.



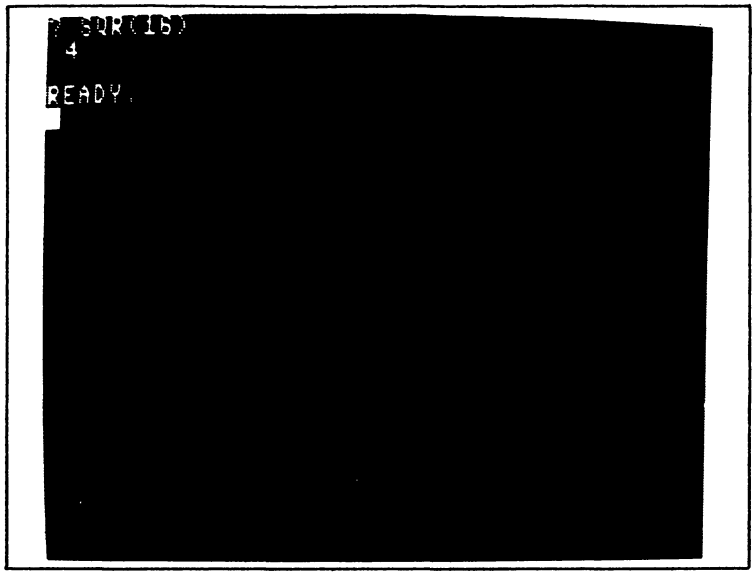
90. To find the nth root of any number with the C64, use the form shown in this example. For instance, to find the fourth root of 2.5, first clear the screen, then type in: ? 2.5↑(1/4).



91. Press RETURN and you will be immediately provided with the answer, 1.25743343, shown here.

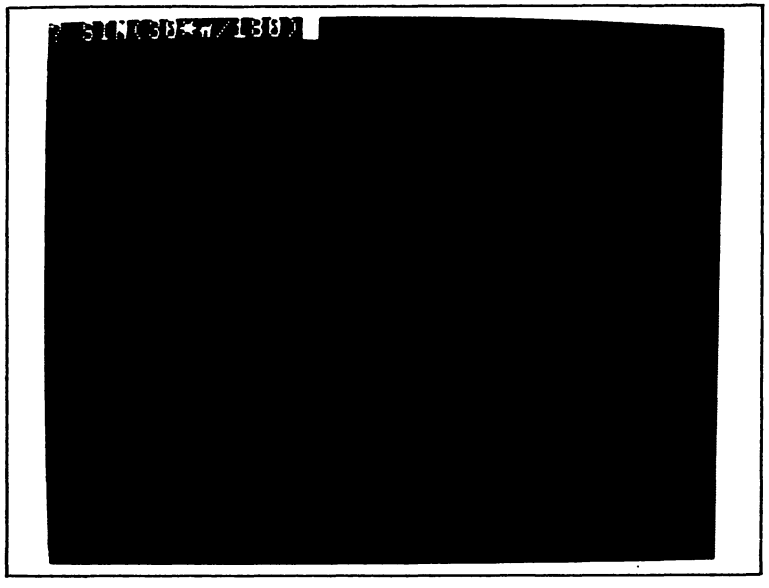


92. Square roots are commonly needed, so the C64 has been provided with a direct means to obtain them. To find the square root of 16 you can type in either: ? 16↑(1/2) or you can type in: ? SQR(16). Try it the first way, clear the screen, and then try it the second way.

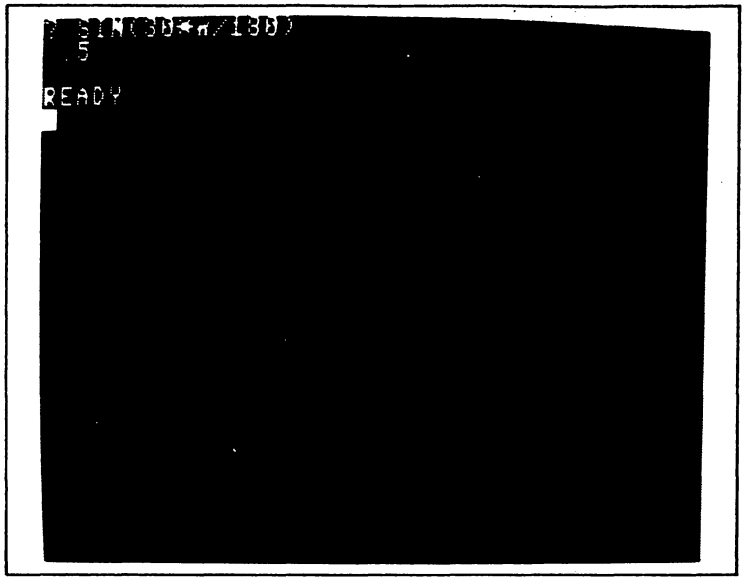


93. If you've typed the line correctly, when you press RETURN you will get the answer, 4.

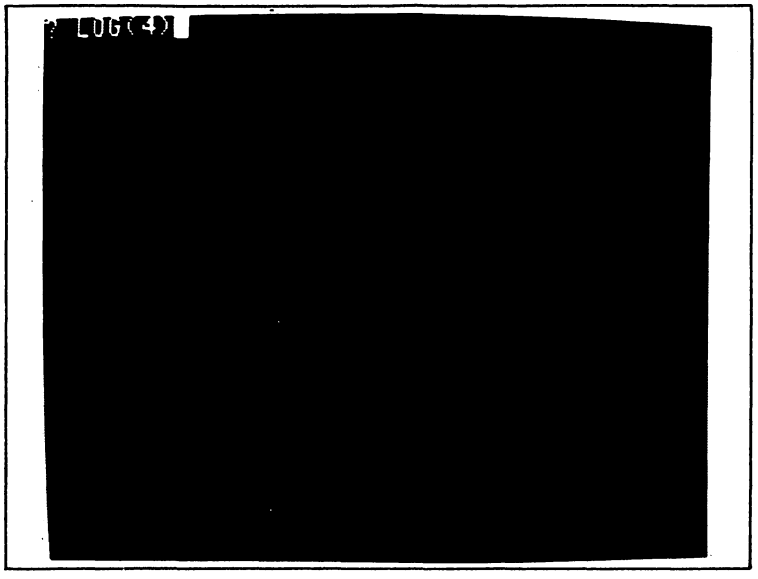
94. Three trigonometry functions are available: SIN, COS, and TAN. All other trigonometry functions must be obtained through manipulation of the available three. Be careful when using these functions, however, as they work only in radians, not in degrees. You can convert degrees to radians by multiplying by PI/180. This chore is eased somewhat by the availability of PI on the keyboard. (The Greek symbol for PI is printed on the front of the UP-ARROW key.) To get it printed on the screen you need only push the SHIFT key and hold it down while you press the UP-ARROW key.



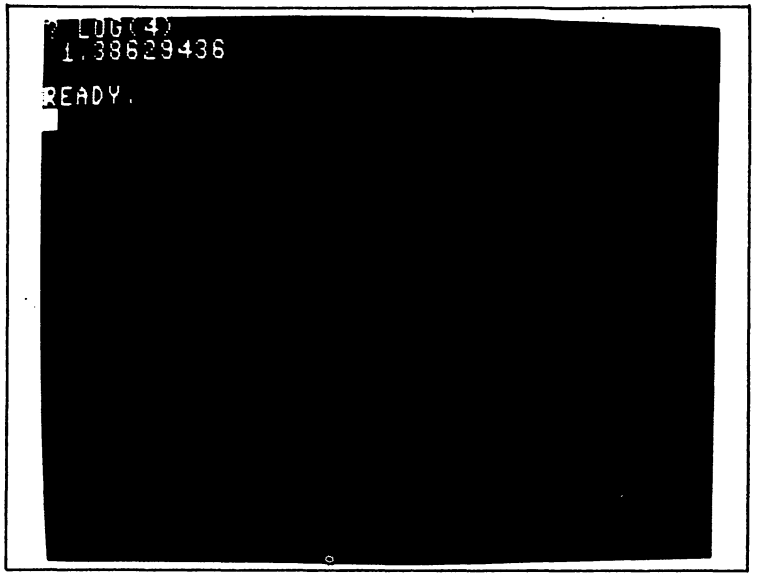
95. To find the sine of 30 degrees, first clear the screen, then type in: ? SIN(30*π/180).



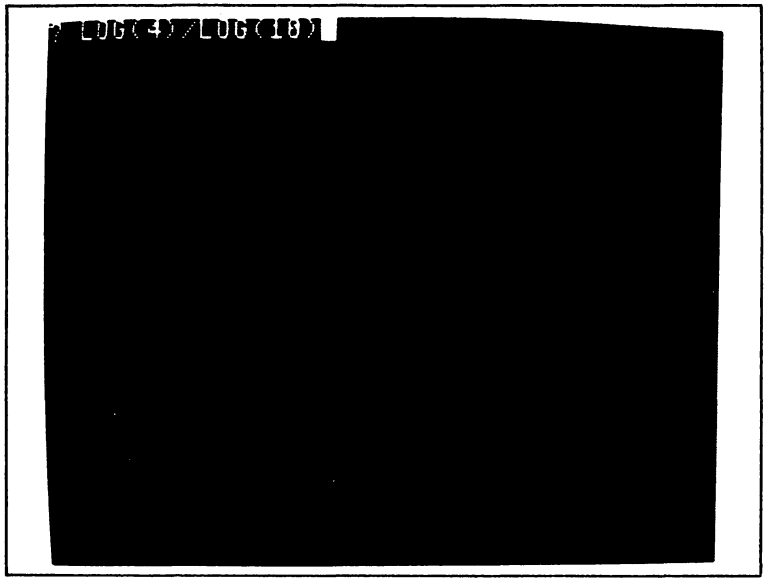
96. When you press RETURN, you will obtain the answer, .5.



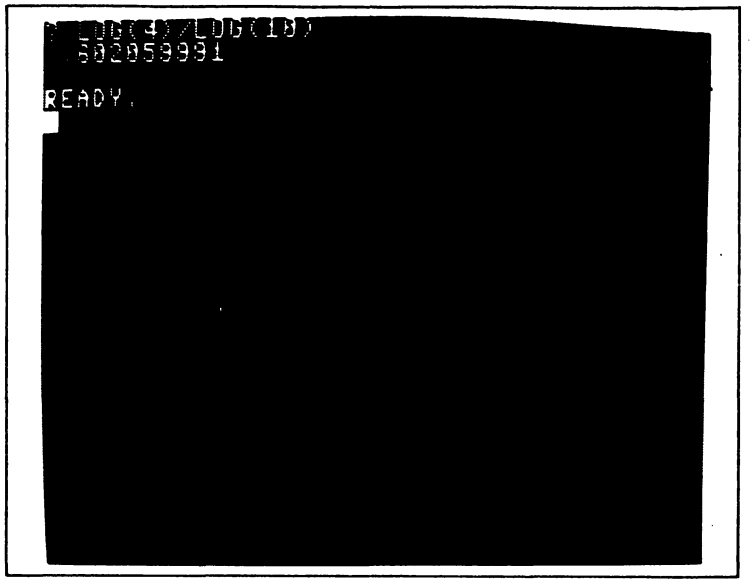
97. To find the natural logarithm (the log to the base e) of any number, use the function LOG. For example, to find the natural log of 4, first clear the screen, then type in: ? LOG(4).



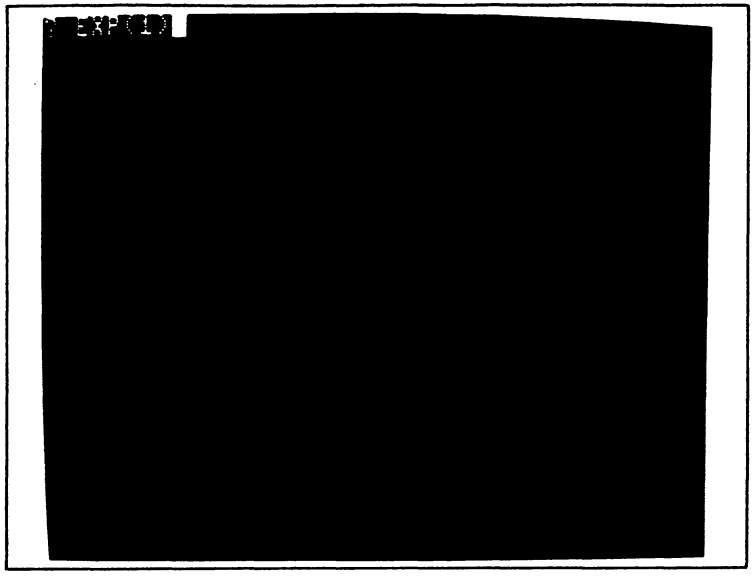
98. After pressing RETURN, you will get the answer, 1.38629436.



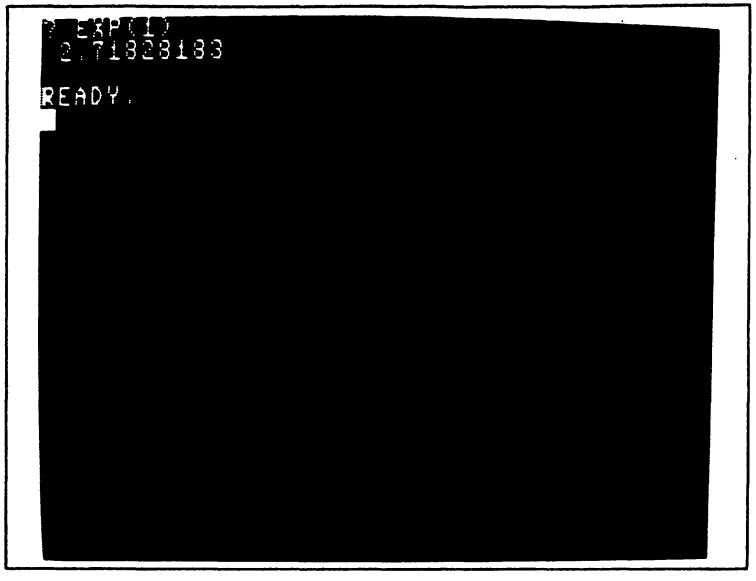
99. To find the log to the base 10 of any number, simply divide the natural log of the number by the natural log of 10. For example, the log to the base 10 of 4 is found by first clearing the screen, then typing in: ? LOG(4)/ LOG(10).



100. Press RETURN and the answer, .602059991, is yours.

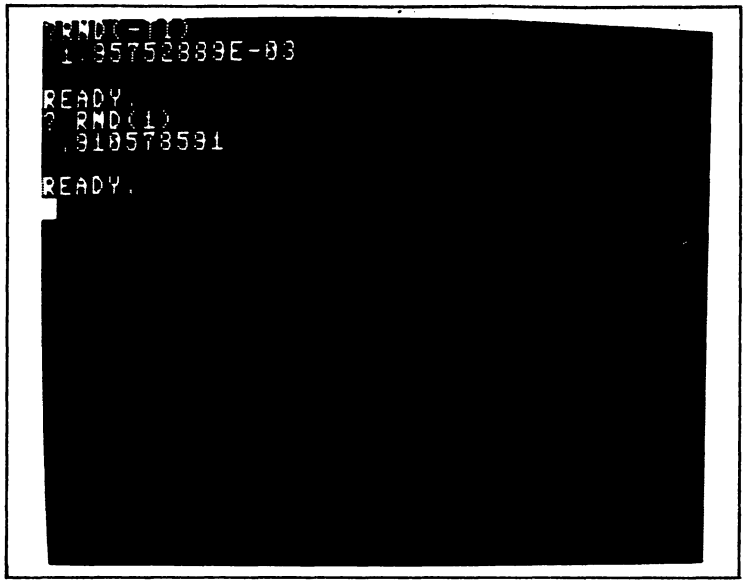


101. If you know the natural log of a number and you want to find out what the number is, use the EXP (exponential) function. For example, to find the number whose natural log is 1, begin by clearing the screen, then type in: ? EXP(1).

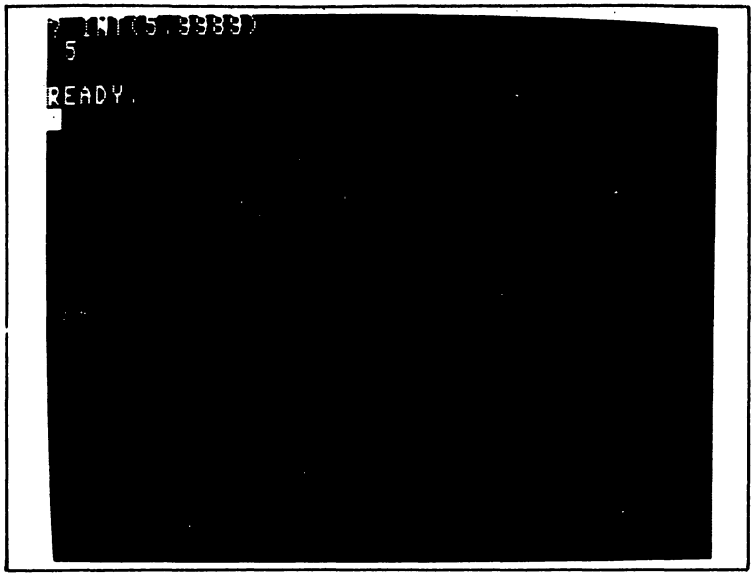


102. After RETURN is pressed you will be presented with the answer, 2.71828183.

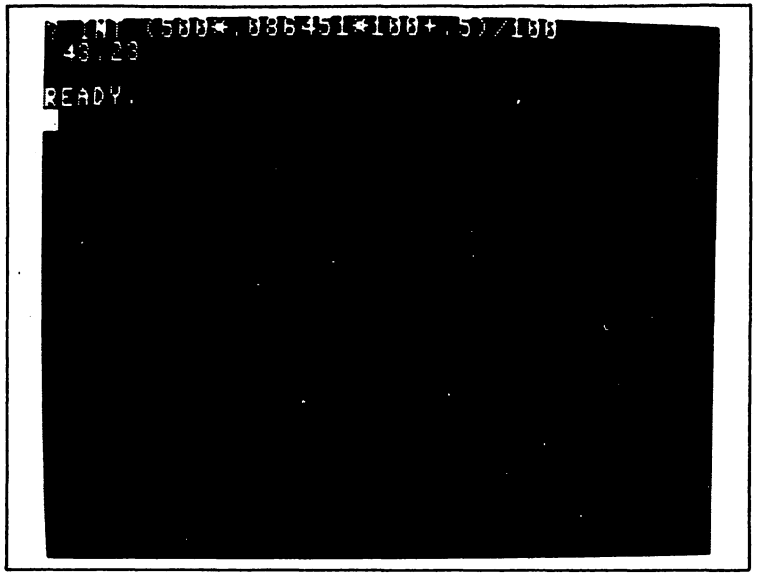
103. Getting a random number from the C64 is best handled in two steps. First clear the screen, then type in: ? RND(—TI) (and press RETURN). That will gain you entry into a fixed sequence of numbers stored in the machine at some random point determined by the computer's internal clock (TI). Ignore the number provided (which may be given in scientific notation).



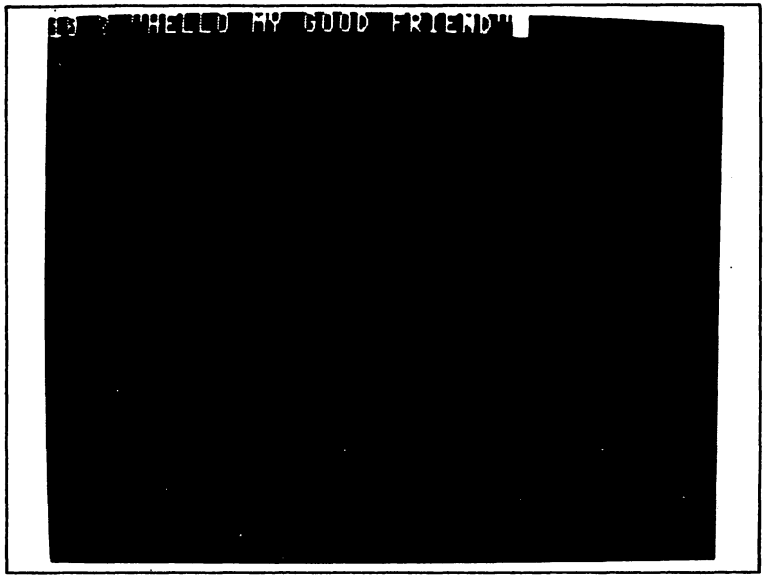
104. Then obtain your random number by typing in: ? RND(1) (and pressing RETURN). You won't get the same result shown here, because if you did it would hardly be a random number, would it?

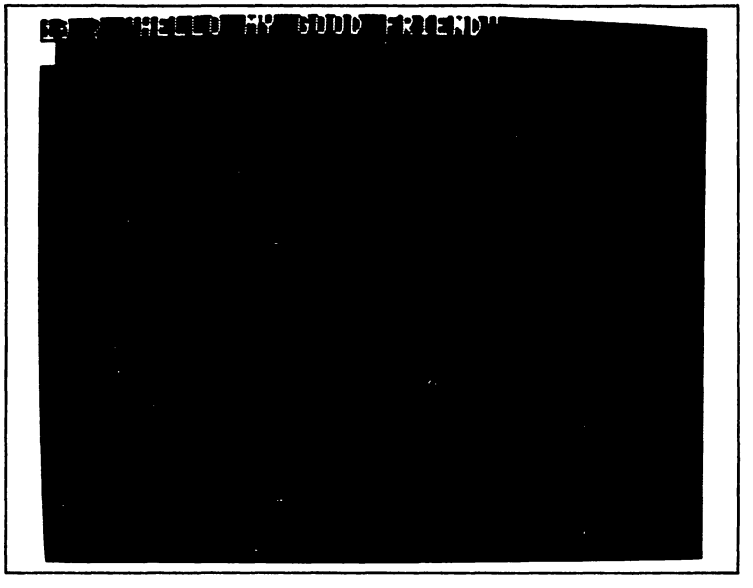


105. If lots of numbers after the decimal point aren't your thing, you can get rid of them quickly with the INT function. INT will provide an integer answer eliminating any decimal values. But always remember that it rounds in one direction only, and that is down. For example, clear the screen, then type in: ? INT(5.9989) (and press RETURN). See what I mean? The answer obtained is 5.



106. The INT function can be used to limit decimal places quite simply and effectively. To limit X number of dollars to two decimal places, rounding to the nearest penny, simply use $=INT(X* 100+0.5)/100. To see how it works, let's say that you earn interest at the annual rate of 8.6451 percent on your principal of $500. Your check, at the end of the year, would be equal to INT(500* .086451 * 100+.5)/100. Type all that in, preceded by a question mark, press RETURN, and you will be rewarded with a check for $43.23.



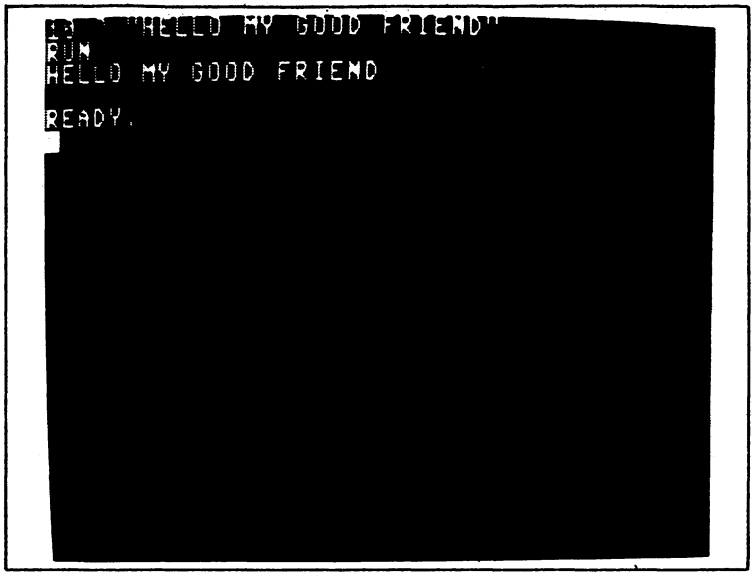


TELLING THE C64 WHAT TO DO IN A LITTLE WHILE—VERY BASIC PROGRAMMING IN BASIC

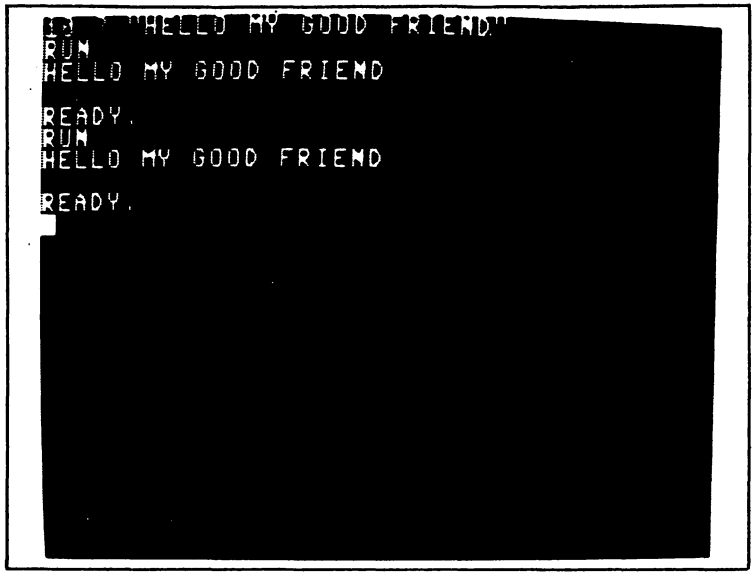
Entering and Running Simple Programs

107. Frankly, most uses of the C64 in the immediate mode are just a bit strained. Cork bulletin boards and pocket calculators are really much more practical. It was useful to go into them at such great length, however, if only to get you used to operating the machine. Computers really come into their own in the delayed, or programmed, mode of operation. The thing that tells the computer you want to operate in a programmed mode is a line number. To illustrate this, first clear the screen, then type in: 10 ? "HELLO MY GOOD FRIEND."

108. Now press RETURN and notice that nothing much seems to happen except that the cursor moves down to the next line. What you can't see happening, however, is that the program line is instantly added to the computer's memory, where it will remain until the machine is turned off, or until you change it.



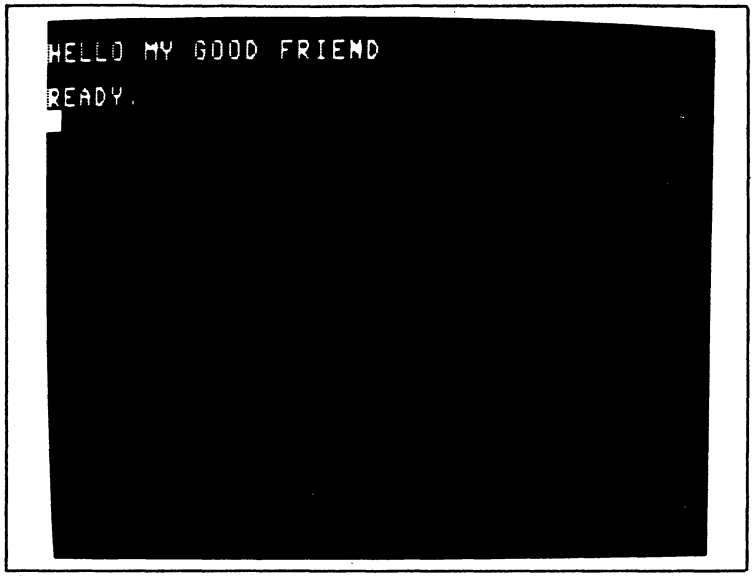
109. Now type in: RUN (and press RETURN). Notice that the C64 has done what you asked it to do. It PRINTed everything between the quotation marks, which was, of course, HELLO MY GOOD FRIEND.



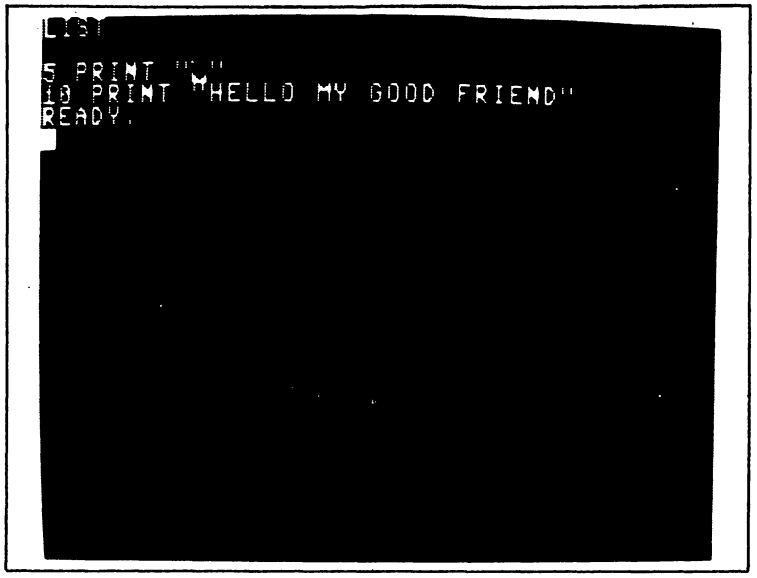
110. Doesn't that give you a sense of power—commanding a computer to do your bidding! Go ahead, try it again. Type in: RUN (and press RETURN). There! You did it again.

111. Still and all, you must admit that the message kind of gets lost amid all that screen clutter. Try this to clear things up; type in: 5 ? " ♥."

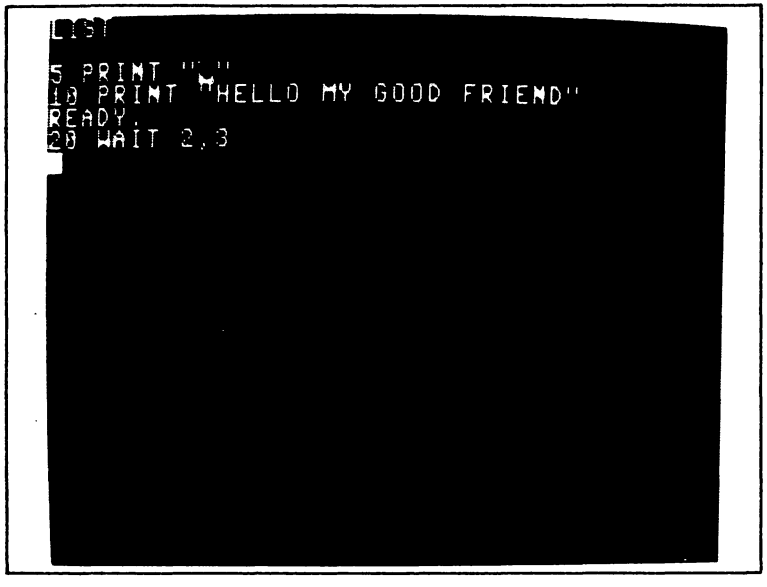
112. The way you get the reverse-video heart to appear on the screen between the quotation marks is to press the SHIFT key and hold it down while you press the CLR/HOME key.



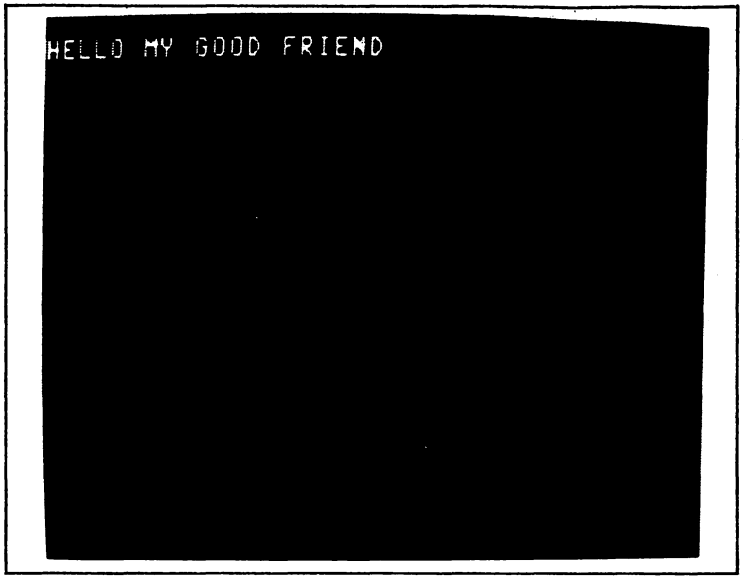
113. Press RETURN to enter the program line into memory. Then type in RUN and press RETURN again. There now. That's a little better. The C64 did as we commanded; it executed the lowest-numbered program line first—that was line 5—and cleared the screen. Then it went on to execute the command it found on the next-highest-numbered program line—line 10—and PRINTed the message we asked it to. But we still have the word READY and the blinking cursor appearing on the screen to distract attention from the message.



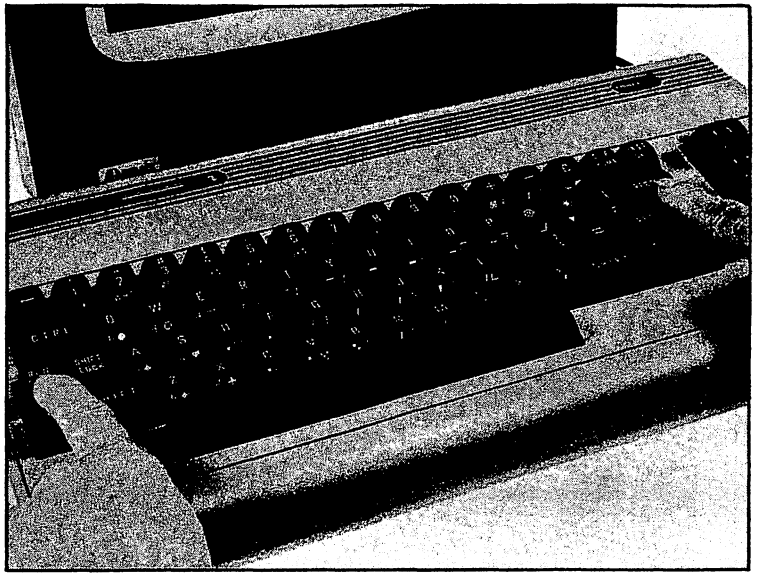
114. The little two-line program is stored in the computer's Random Access Memory (RAM). Let's get it back on the screen and see if we can do something about getting rid of the distractions. To get the program, simply clear the screen, then type in: LIST (and press RETURN). Notice that the computer has changed the question marks into the command PRINT, and it has arranged the program lines in their proper numerical order, even though you didn't enter them that way.



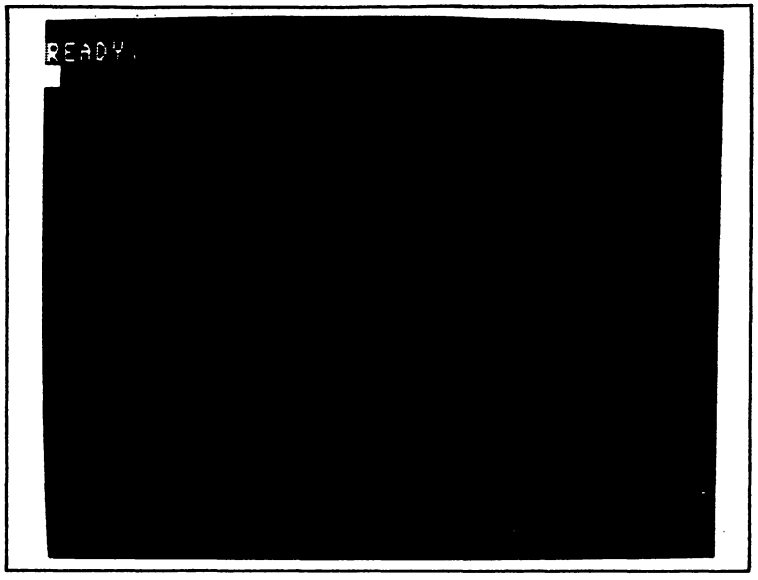
115. To get rid of the word READY and the blinking cursor, we can tell the C64 it has to stop running the program and wait. Type in: 20 WAIT 2,3 (and press RETURN).



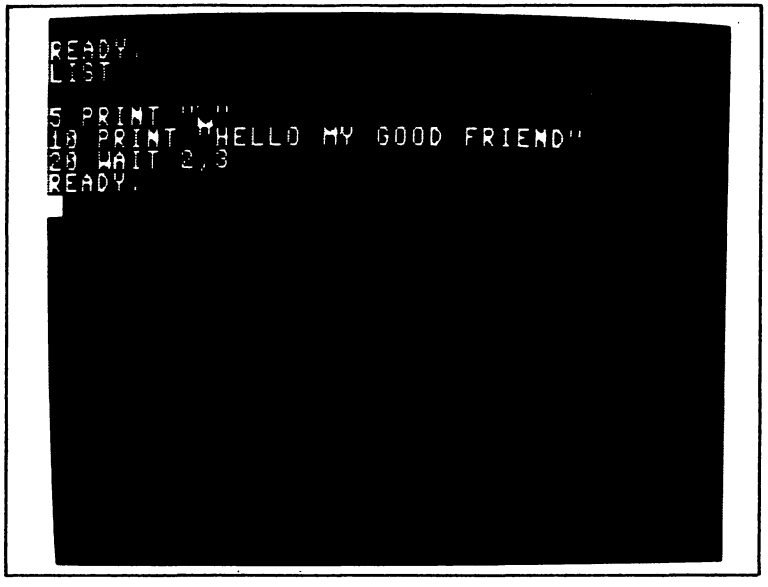
116. Now type in: RUN (and press RETURN) and your message doesn't share the screen with anything else.

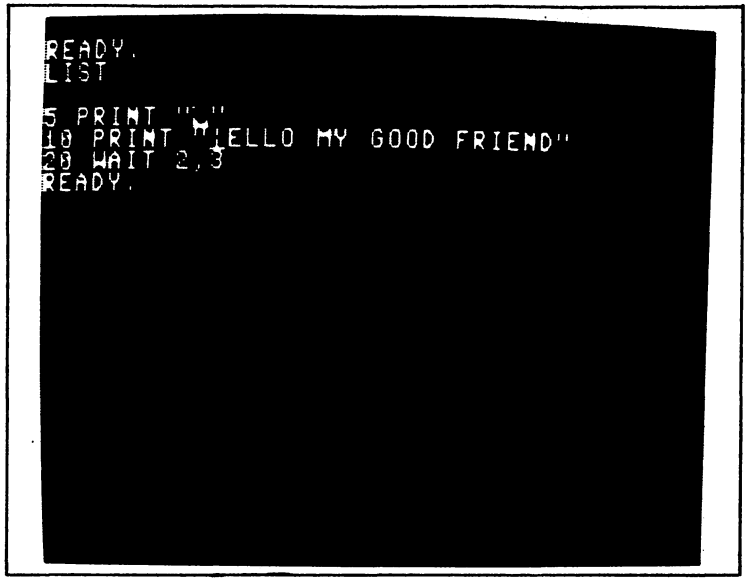


117. The WAIT command is too complicated to really explain here (the numbers following the word are memory locations and the machine automatically manipulates the contents of those locations), but its use essentially paralyzes the C64. There is now no single key you can push that will get the machine to come back to life. Try it, and see if I'm not right. To rescue the C64, press the RUN/STOP key while you press the RESTORE key.



118. Now your screen looks like this.



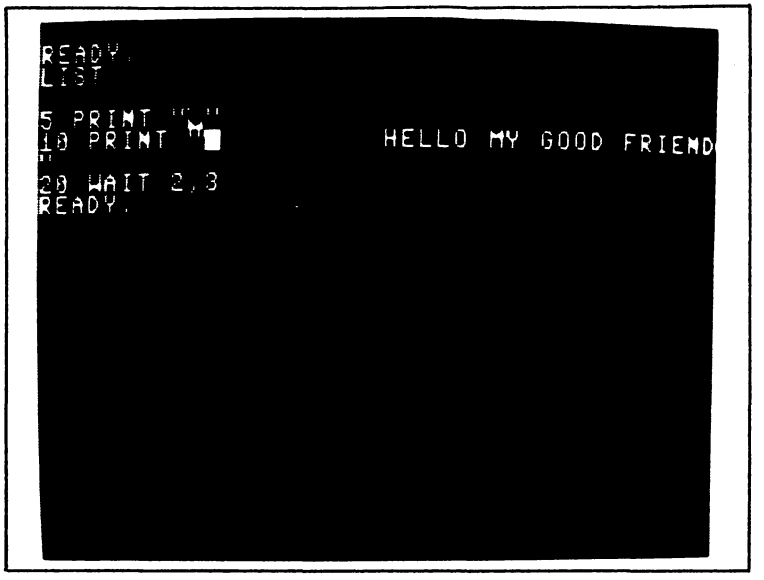


Program Editing and Screen Layout

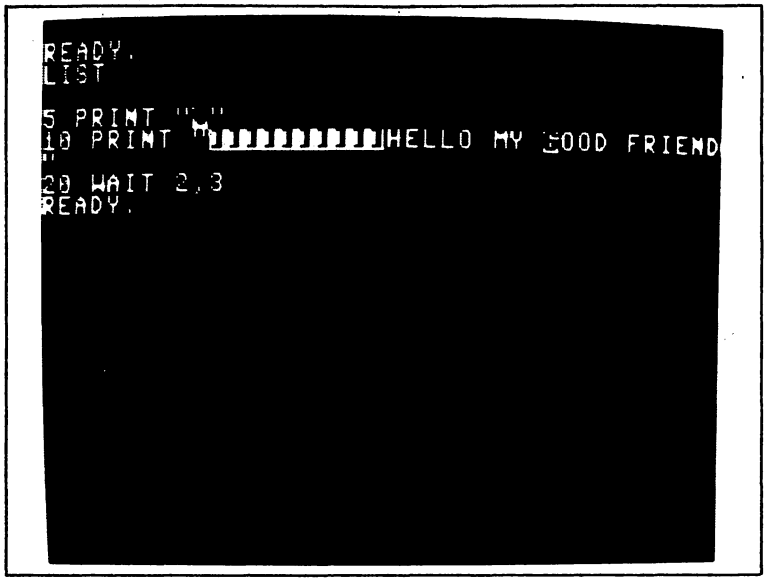
119. You have already discovered that the screen layout consists of 25 rows and 40 columns. We can use that knowledge to edit our little program to PRINT the message in the center of the screen. Begin by typing: LIST (and pressing RETURN) to get the program back on the screen.

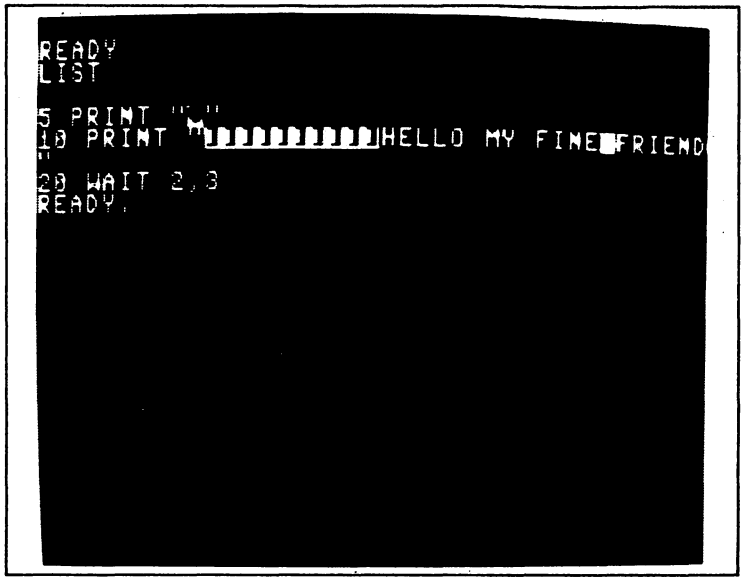
120. But before we do, just to see how easy it is to edit program lines on the C64, let's change the message slightly. Use the SHIFT key and the vertical CRSR key to get the cursor up to line 10, then use the horizontal CRSR key to position the cursor over the letter H in the word HELLO.

121. Now press the SHIFT key and hold it down while you press the INST/DEL key ten times.



122. Notice that doing this has moved all the words within the quotation marks ten spaces away from the left quotation mark, and therefore, ten columns in from the left margin when the words are later PRINTed on the screen. You can use this procedure to INSerT space in any line. The space can then be filled with new material if you wish.

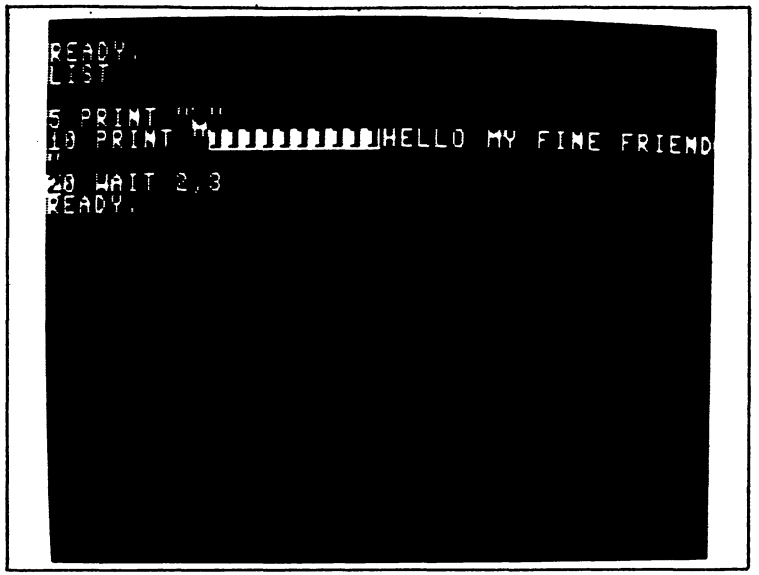




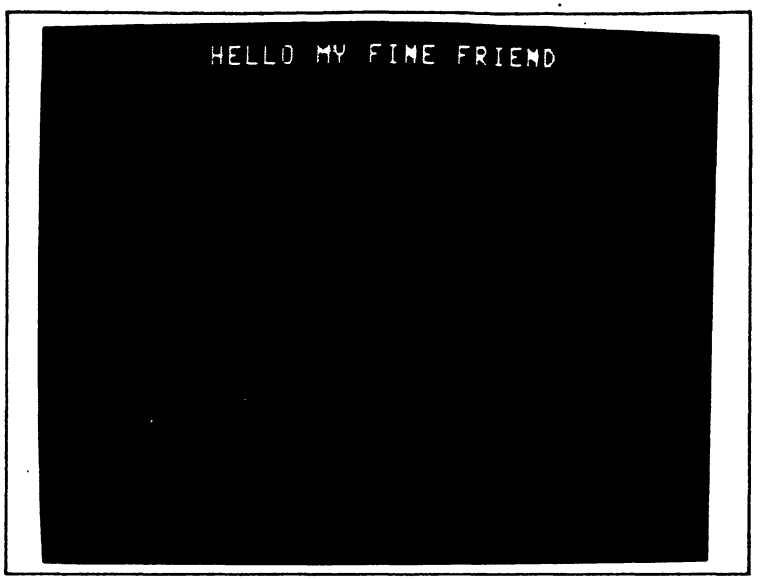
123. Now use the horizontal CRSR key to position the cursor over the G in the word GOOD. Notice that when you move the cursor, inverse-video right-hand brackets remain in the space you just created. Don't worry—they're just reminders that you did some editing. They won't be PRINTed later.

NOTE: Among the computer's less endearing qualities is its tendency to leave inverse-video (black-on-white) characters on the screen where and when you least expect them during editing. Perhaps the most aggravating of these are the marks left when you call for cursor movement. Once the C64 starts leaving these marks, often the best course of action is to simply press RETURN and then use the SHIFT and CRSR keys to bring the cursor back to where you left off editing, and start again.

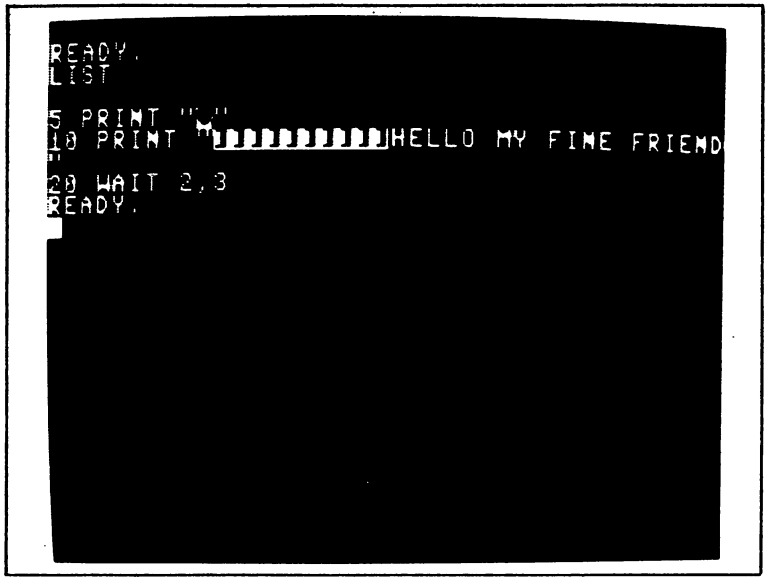
124. Now type in the word FINE.



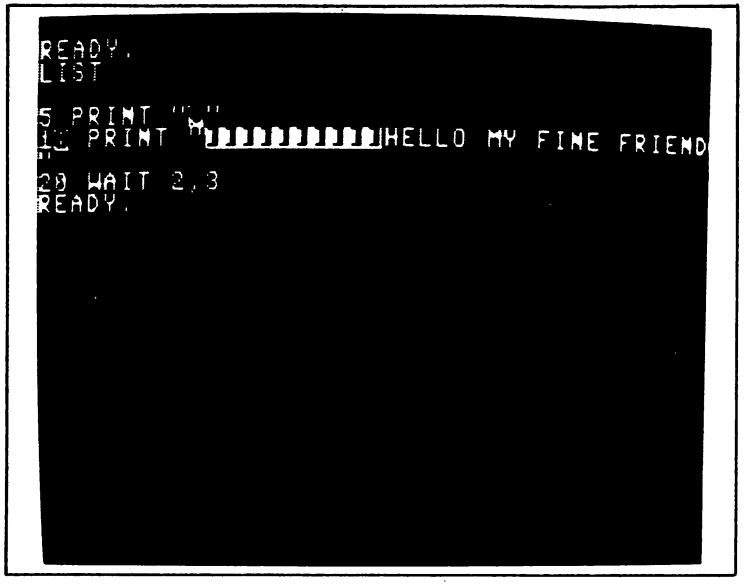
125. Press the RETURN key to enter the edited line into memory. Note that the cursor moves down to the next line number.



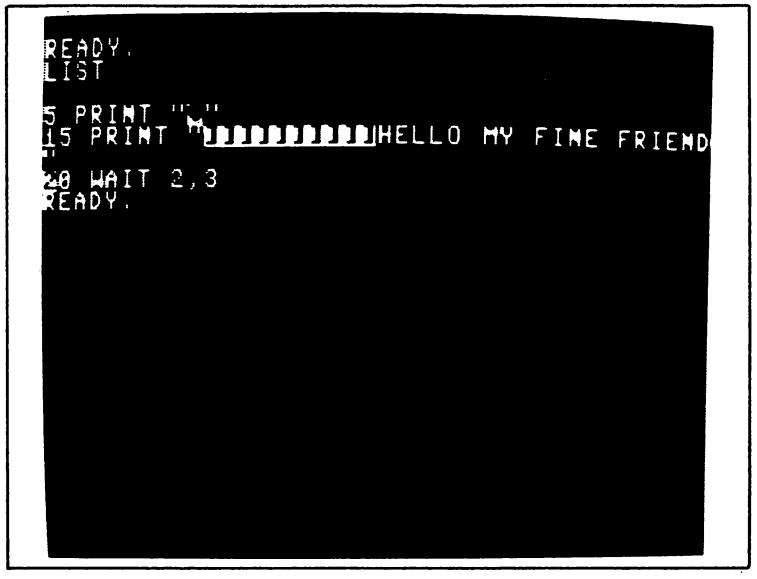
126. Clear the screen, then type in: RUN (and press RETURN), and your screen should now look like this. Notice that the line is now centered at the top of the screen because of the space we added between the left quotation mark and the first word.



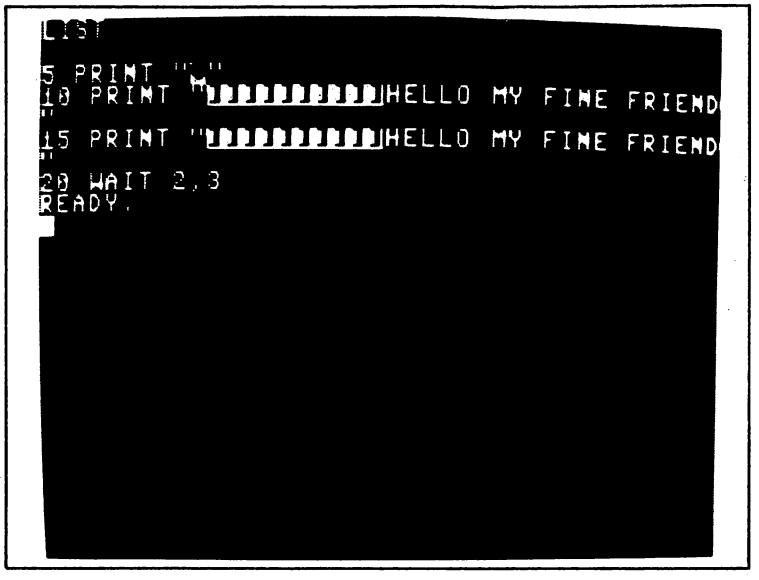
127. Press the RUN/STOP and RESTORE keys. Then type in: LIST (and press RETURN) to get the program listed again.



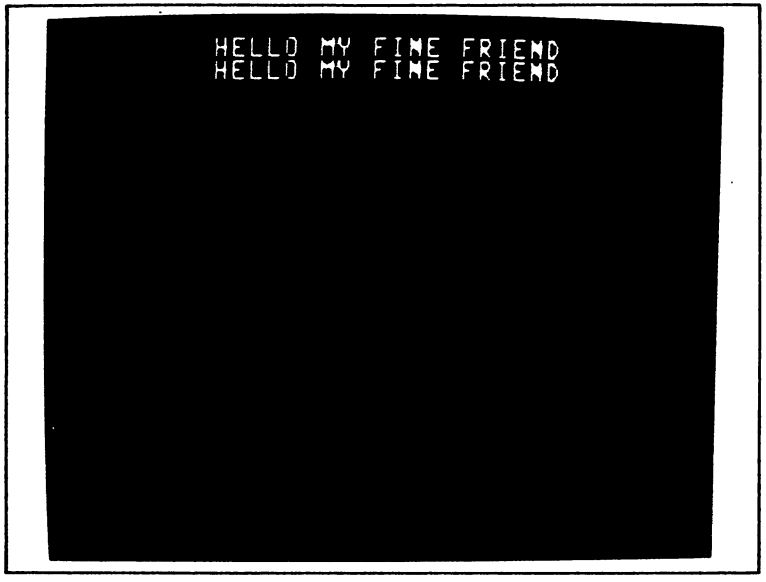
128. You can alter line numbers at will. For example, move the cursor up to the 0 in line number 10, as shown here.



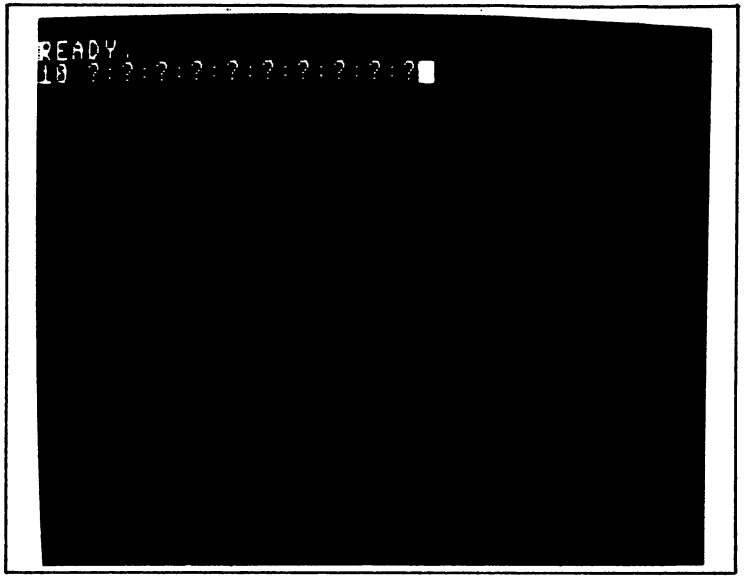
129. Type a 5 so that the line number showing on the screen is 15. Press RETURN to enter the change into memory. Note that the cursor moves to the next line number to indicate that line 15 has been entered.



130. Clear the screen, then type in: LIST (and press RETURN) to get the program listed again. Now notice that you still have line 10, but you also have an identical line, numbered 15.

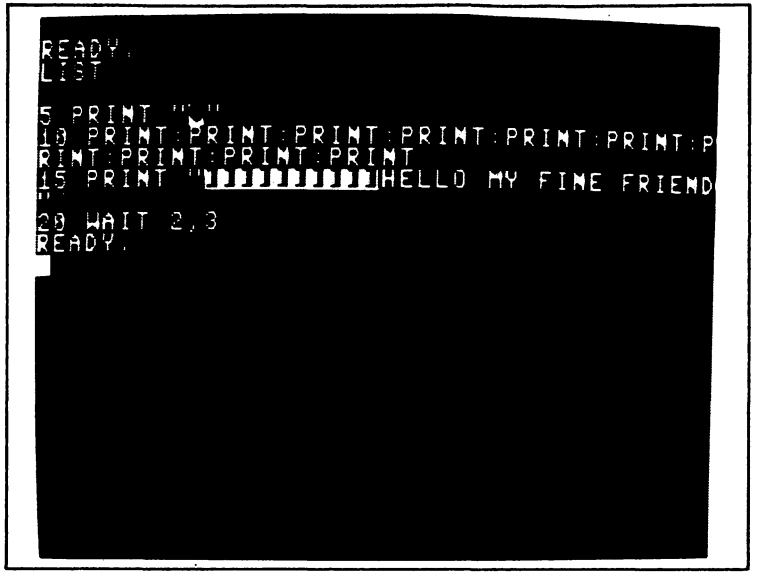


131. RUN the program, just for the heck of it. Both line 10 and line 15 are PRINTed by the ever-faithful C64.

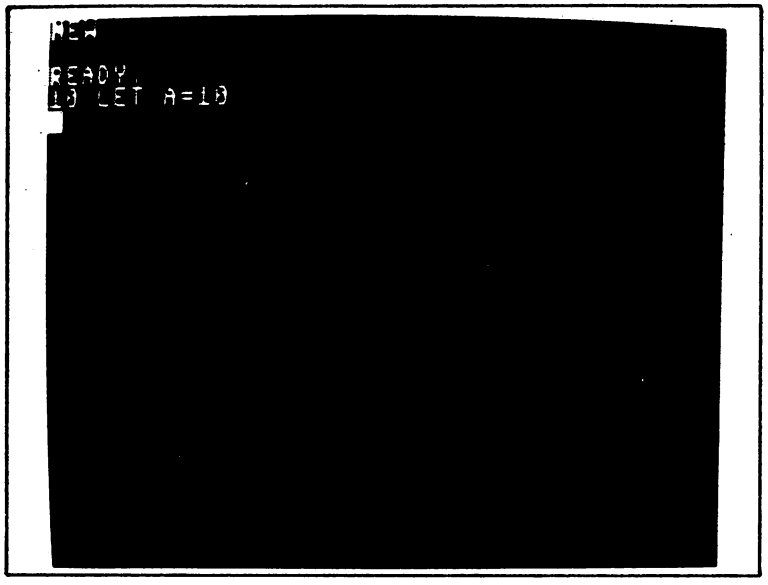


132. There is no need to have the listing of the program on the screen to make changes. Wake the computer by pressing the RUN/STOP and RESTORE keys. Then type in: 10 ?:?:?:?:?:?:?:?:?:? (yes, that's ten question marks, each separated from the last by a colon) and press RETURN.

133. Now RUN the program. The line 10 we just entered has replaced the old line 10 in the program. The new line 10 commanded the C64 to PRINT 10 times, but it never told it what to print. When the computer gets such PRINT instructions it simply PRINTs an empty line for each.



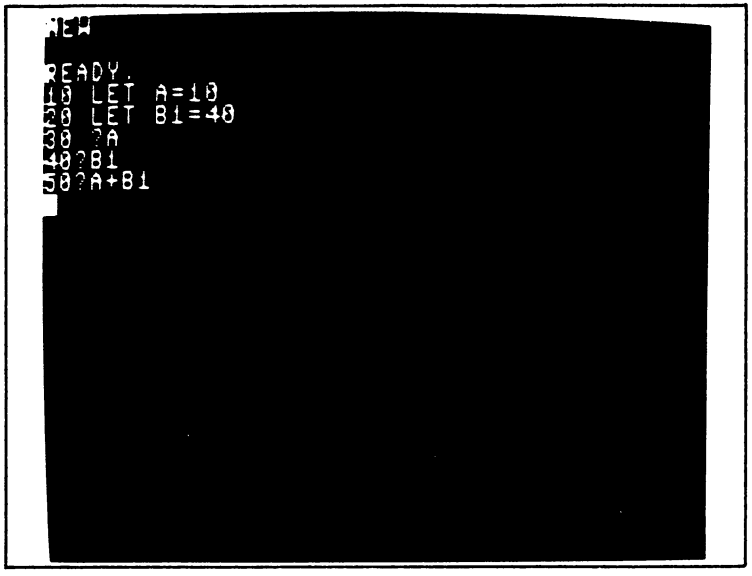
134. Press RUN/STOP and RESTORE, then type: LIST (and press RETURN) to get the program LISTed again. Notice that the computer has converted each of the question marks into the word PRINT. *You can enter multiple commands on a single program line if you separate each command from the next one with a colon.*



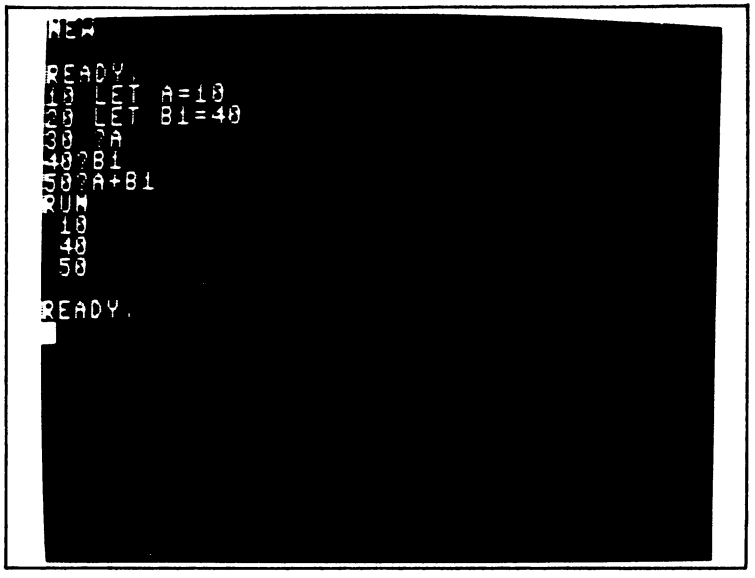
Numeric Variables

135. Writing any kind of meaningful program is going to involve working with variables. There are several types of variables, of which the two most basic are numeric and string. Numeric variables are used to represent only numbers. Every numeric variable must have a different name. The rules concerning these names are fairly straightforward. A numeric-variable name must begin with a letter. It should be no more than two characters long. The last character can be either a letter or a number. Valid numeric-variable names might be: X, XX, X1, and so forth. Let's look at a program example. First clear the screen, then type: NEW (and press RETURN) to clear the C64's memory. Now type in: 10 LET A = 10 (press RETURN). Your screen should now look like this.

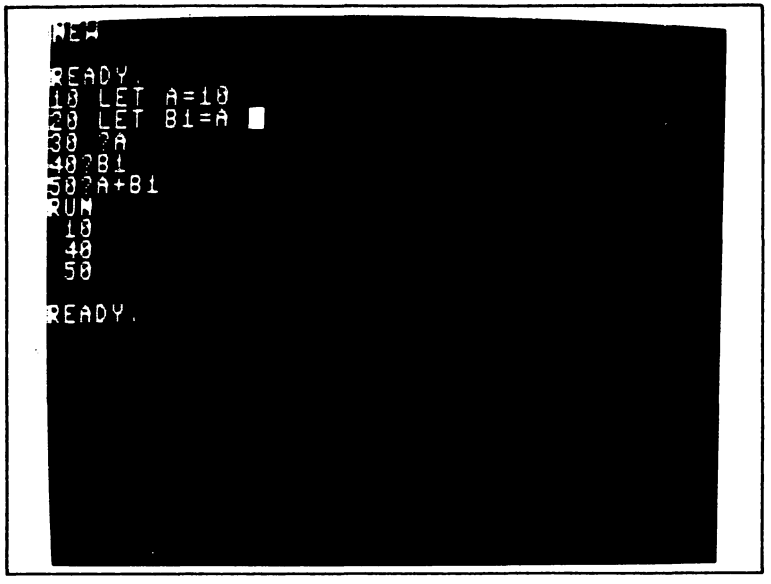
NOTE: You *can* use more than two characters as a numeric-variable name, but the computer will ignore the extras. TOM, DICK, and HARRY are all all right to use as numeric-variable names, but the machine won't be able to differentiate between BOB and BORIS because it reads and identifies only the first two characters in a variable name.



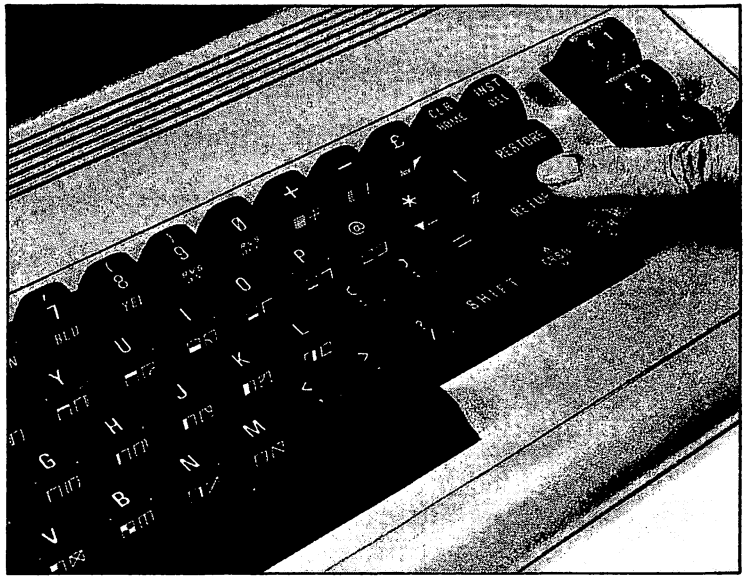
136. Now type the following program lines, and enter each into memory by pressing RETURN after each line is completed: 20 LET B1 = 40 (press RETURN), 30 ?A (press RETURN), 40?B1 (press RETURN), 50?A+B1 (press RETURN). Now the screen should look like this.



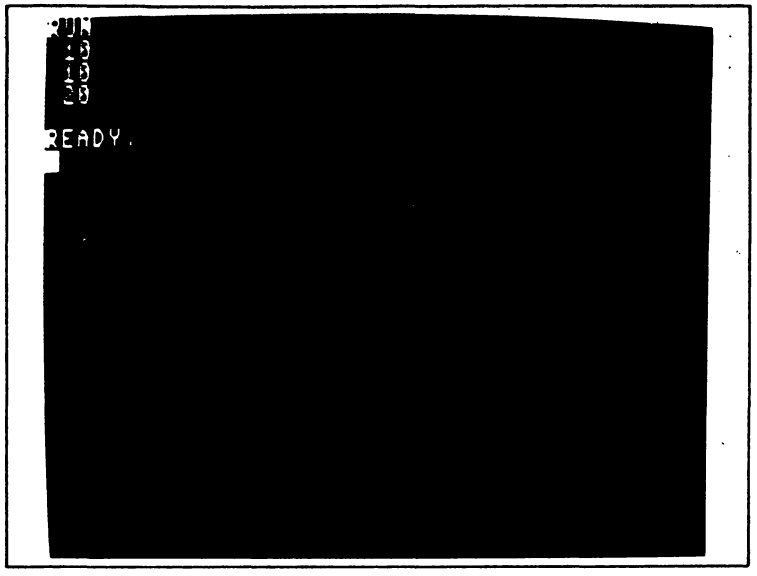
137. RUN the program, and your screen will change to this. The computer has done just what you told it to—PRINTed A (10), PRINTed B1 (40), and PRINTed A+B1 (50).

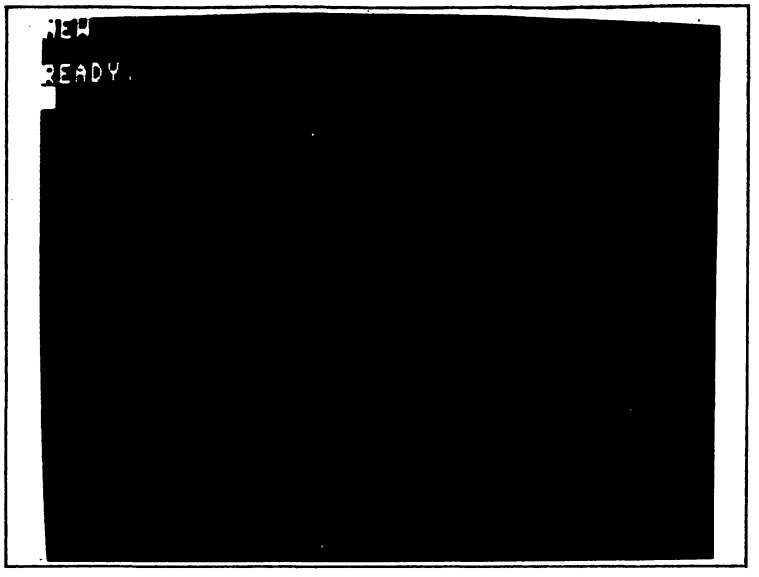


138. The numeric variables, A and B1, you just used were fixed variables (I know that sounds silly, but their values were fixed at 10 and 40, weren't they?). To see how a variable-numeric variable works, use the SHIFT and vertical CRSR keys to bring the cursor back up to program line 20. Use the horizontal CRSR key to move the cursor to the number 4. Then type the letter A, followed by a space, so that now your screen looks like this.



139. Enter the edited program line 20 into memory by pressing RETURN.



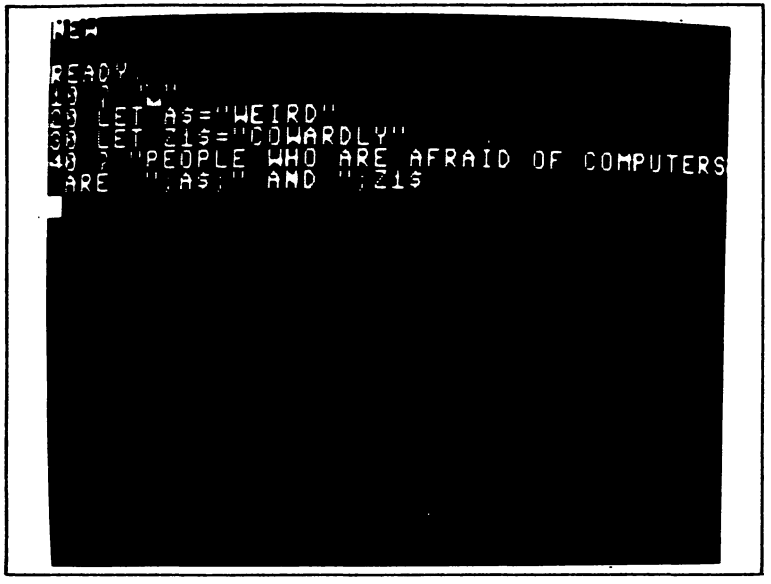


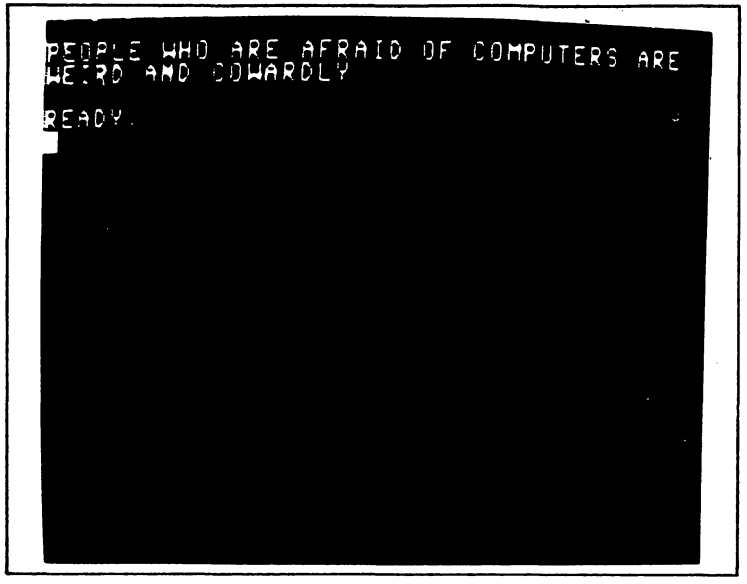
140. Now clear the screen (press SHIFT and CLR/HOME) to get rid of the clutter. RUN the edited program. Now your screen should look like this. Again the C64 did just what you asked it to. It PRINTed A (10), PRINTed B1 (10 because B1=A and A=10), and PRINTed A+B1 (20 because 10+10=20).

NOTE: The use of the command LET is optional with the C64. I have used it to let you know that it exists, but you can eliminate it entirely, as I eventually will, if you wish.

String Variables

141. String variables usually (but not always) consist of letters. The name of each string variable should consist of one or two characters followed by a dollar sign. The first character must be a letter. The second can be either a letter or a number. For example, a string variable could be named A$, AA$, or A1$. When defining the value of a string variable, be sure to enclose the value in quotation marks. If you don't, the computer will generate an error report telling you that you have a type mismatch. Before going on to examples of programming with string variables, however, be sure to clear the screen by pressing SHIFT and CLR/HOME; clear the



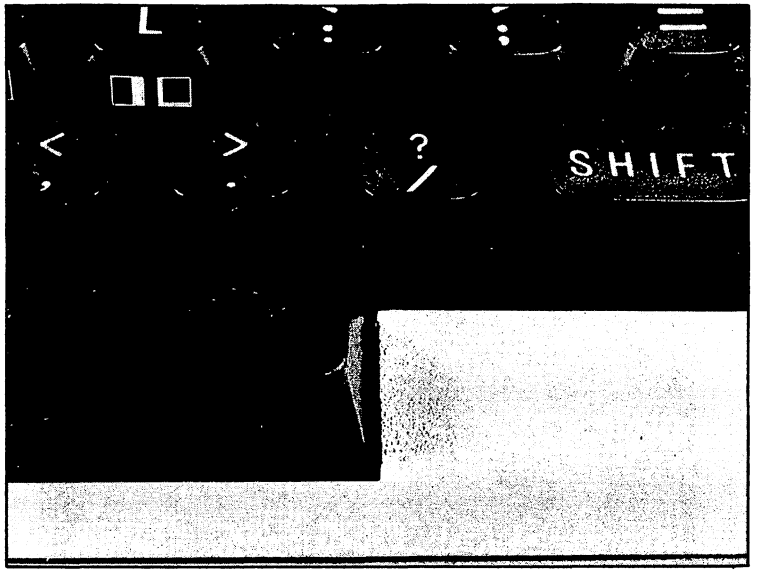


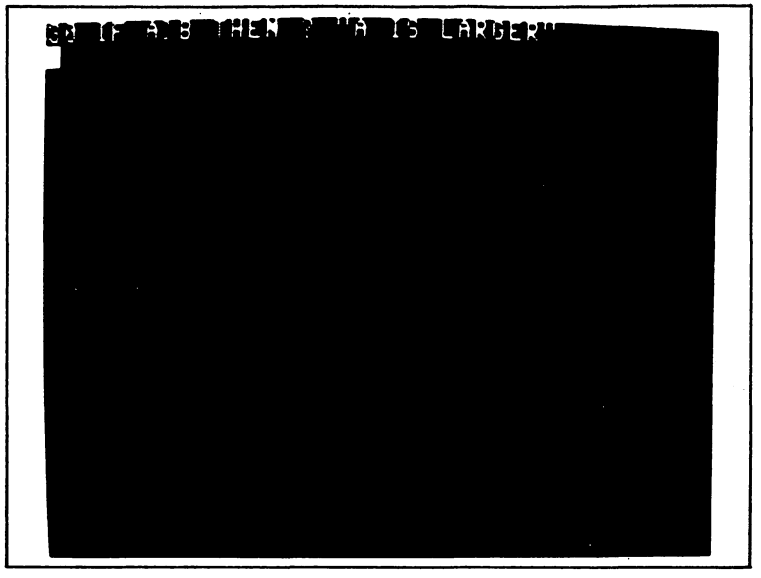
computer's memory by typing: NEW (and pressing RETURN).

NOTE: You can use more than two characters as a string-variable name, but the computer will ignore the extras. NAME$, SEX$, and RACE$ are all fine to use as string-variable names, but the machine won't be able to differentiate between STREET$ and STATE$, because it reads and identifies only the first two characters in a variable name.

142. Now type in: 10 ? " ♥ " (press RETURN), 20 LET A$="WEIRD" (press RETURN), 30 LET Z1$="COWARDLY" (press RETURN), 40 ? "PEOPLE WHO ARE AFRAID OF COMPUTERS ARE ";A$;" AND ";Z1$ (press RETURN). Now your screen should look like this. Be very careful of the spacing and be sure to get all the quotation marks (") and semicolons (;) typed in exactly as shown. You get the dollar sign by pressing and holding the SHIFT key down while you press the $/4 key.

143. Type: RUN (and press RETURN) and the computer will express its opinion of such folk.

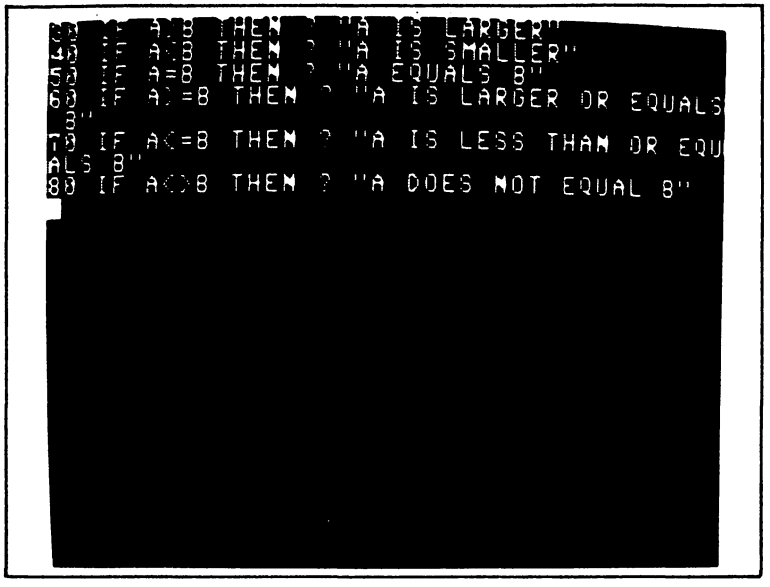




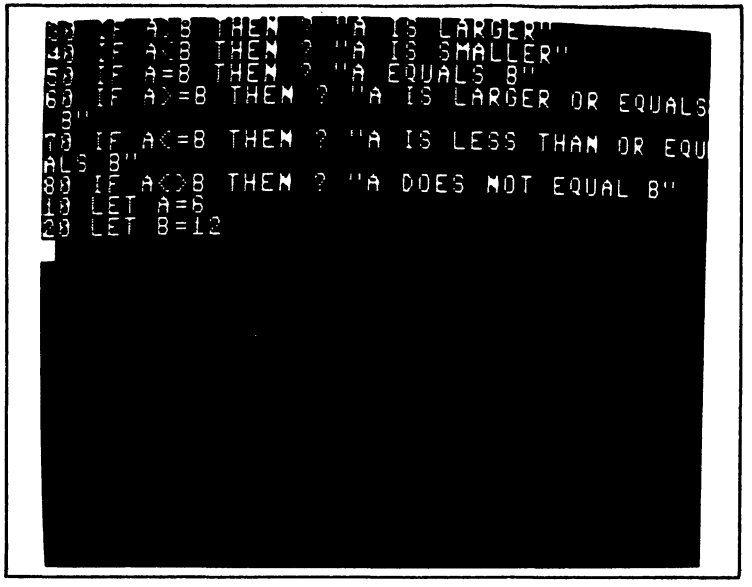
Some Logical Considerations

144. We have far from exhausted the supply of things to be learned about numeric and string variables, but we have reached a point where what has so far been presented can be put to very good use, if only a bit more knowledge is added to it. Let's begin by looking at some simple mathematical symbols: <, >, <=, and >=. The latter two are obtained with two key presses. You'll find the first two above the comma and period, so they call for the use of the SHIFT key to get them on the screen.

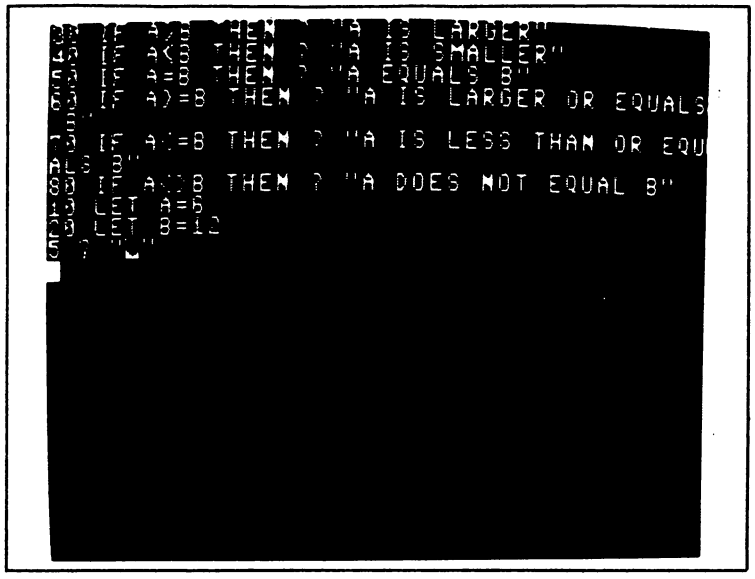
145. The symbol > has a big side (on the left) and a little side (on the right). If it is placed between two numeric variables, it indicates that the variable on the left is greater (has a larger numerical value) than the one on its right. For example, clear memory (NEW) and clear the screen (SHIFT CLR/HOME), then type in: 30 IF A>B THEN ? "A IS LARGER" (press RETURN).



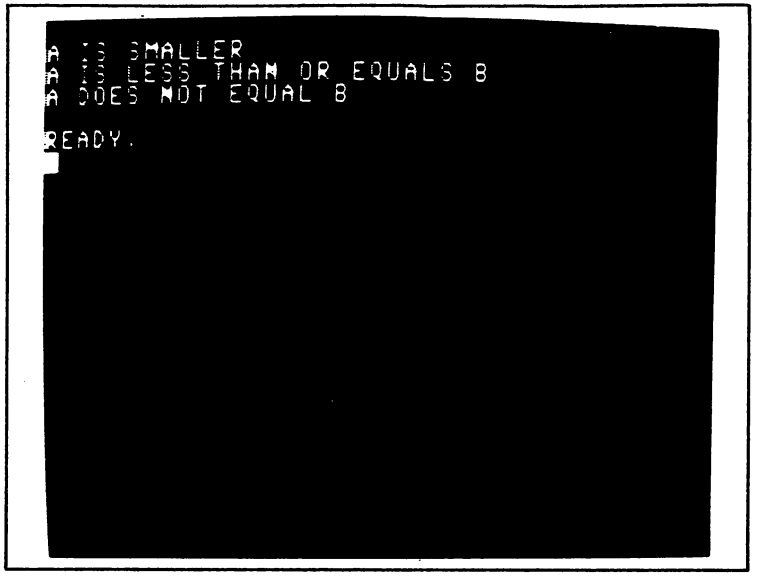
146. Now type in: 40 IF A<B THEN ? "A IS SMALLER" (press RETURN), 50 IF A=B THEN ? "A EQUALS B" (press RETURN), 60 IF A>=B THEN ? "A IS LARGER OR EQUALS B" (press RETURN), 70 IF A<=B THEN ? "A IS LESS THAN OR EQUALS B" (press RETURN), 80 IF A<>B THEN ? "A DOES NOT EQUAL B" (press RETURN).



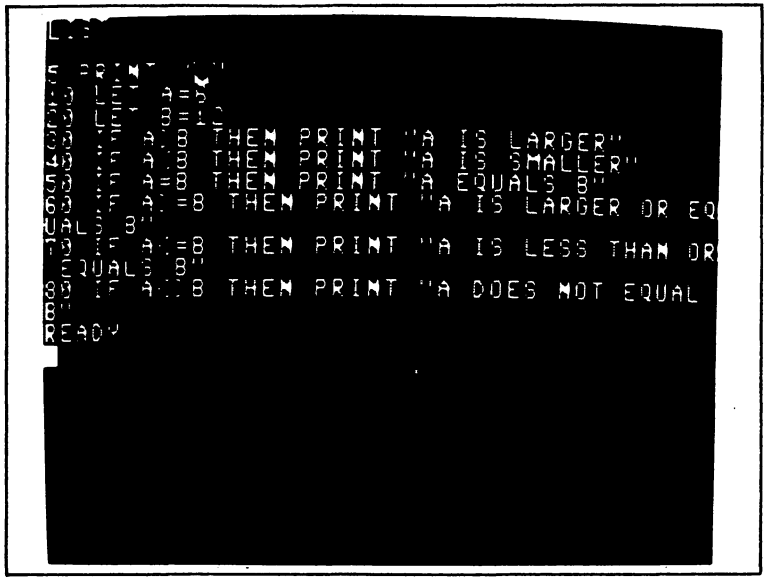
147. Now that we have established all of the conditional PRINT statements, we had best also establish some values of A and B for them to measure. Type in: 10 LET A=6 (press RETURN), 20 LET B=12 (press RETURN).



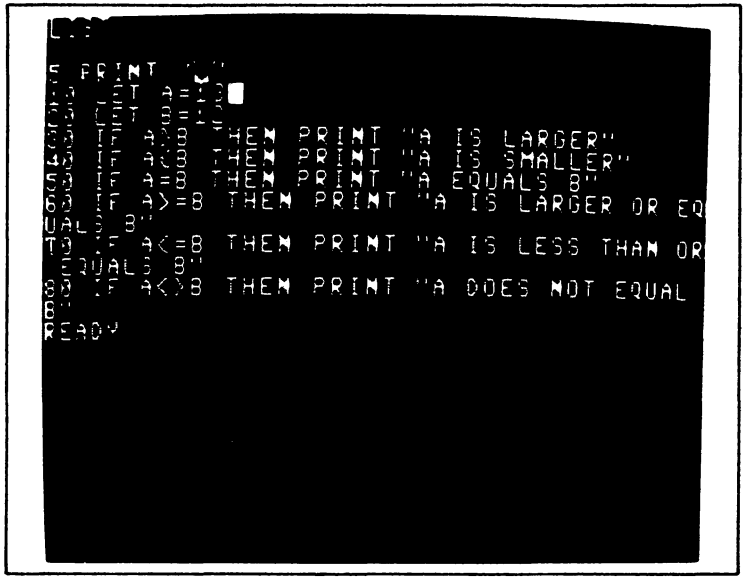
148. And just for good measure, to clear the screen each time this program is RUN, type in: 5 ? "♥" (and press RETURN). Remember, to get the inverse-video heart, you must press and hold the SHIFT key while pressing the CLR/HOME key.



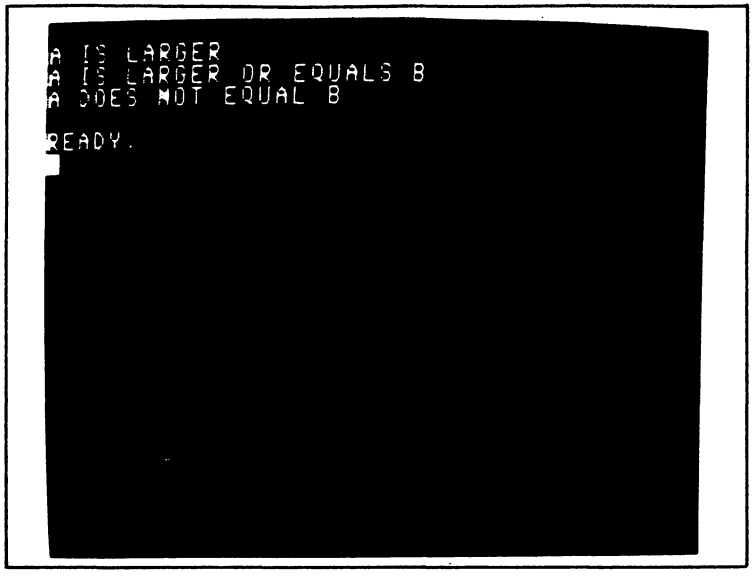
149. Now RUN the program, and these are the three lines you get. Quite rightly, too, because since A (6) is obviously smaller than B (12), the condition of line 40 was met, and its PRINT command was executed. Since A is smaller than B, one of the two acceptable conditions for line 70 was met, and its PRINT command was executed. And because A does not equal B, the condition of line 80 was met, and its PRINT command was executed. Since none of the conditions of lines 30, 50, and 60 were met, the computer did not execute their PRINT commands. And having done everything it was told to do, the C64 announces that it is ready for more and blinks its cursor at us.



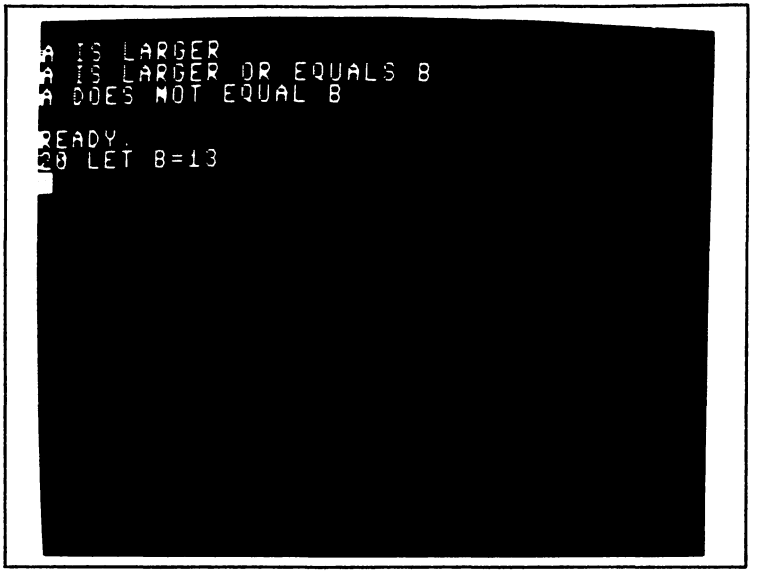
150. Clear the screen, then type: LIST (press RETURN).



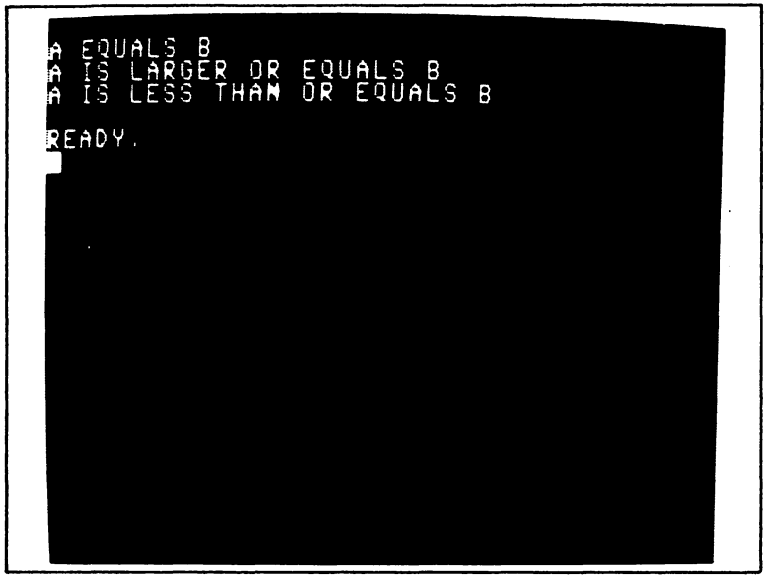
151. Use the SHIFT and CRSR keys to position the cursor over the 6 in program line 10. Type in: 13.



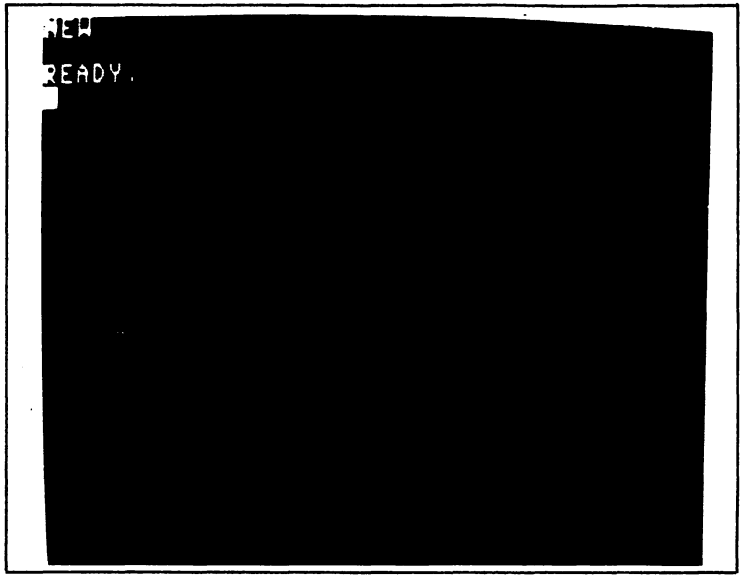
152. Press RETURN to enter the edited program line to memory. Clear the screen, then RUN the program. And again, the computer has done exactly what you asked it to do. It found that A is now larger than B; and since that satisfied the condition of line 30, it executed that line's PRINT command and gave you a message advising you of the fact. Then it went on and found that the required conditions in lines 60 and 80 were met, and executed their PRINT commands. That's where it ran out of instructions, so having done its job, it stopped.



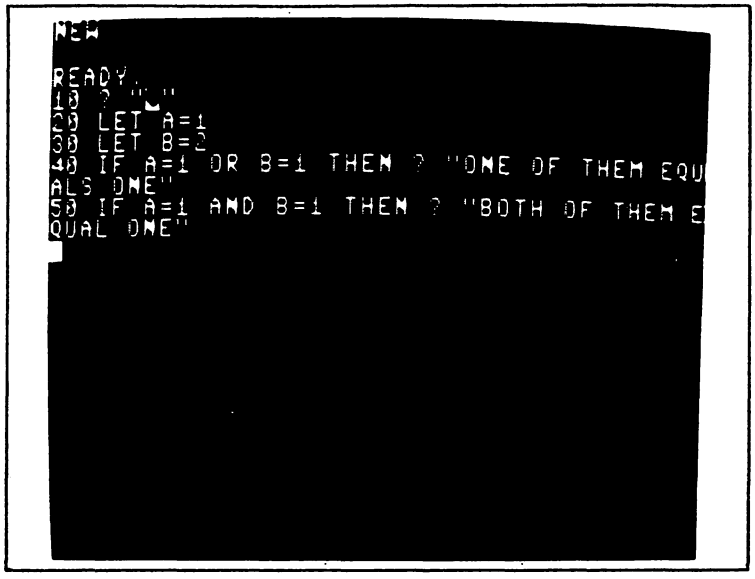
153. Now, without bothering to LIST the program again, simply type in: 20 LET B=13 (and press RETURN). Doing so will automatically replace the program line 20 in memory with this new one, which is identical in everything but the value of B.



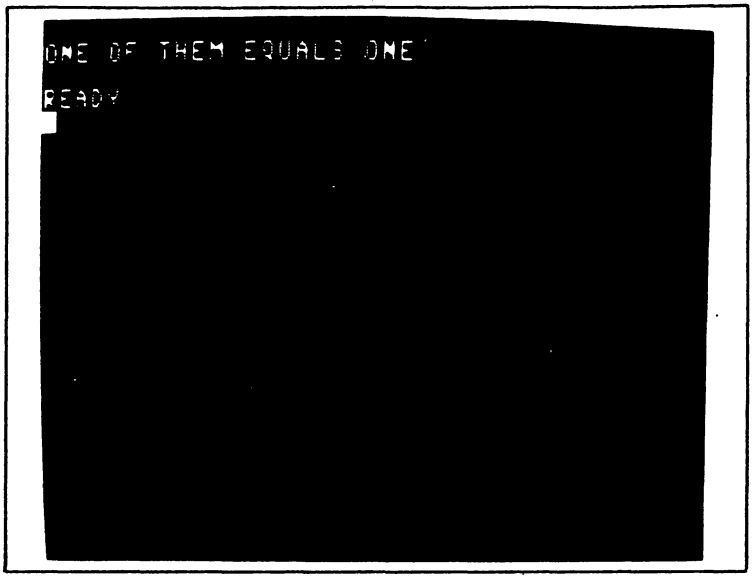
154. RUN the new program. Now the required conditions of lines 50, 60, and 70 have been met, their PRINT commands have been executed, and the computer has run out of things to do, so it let us know that it was ready for more.



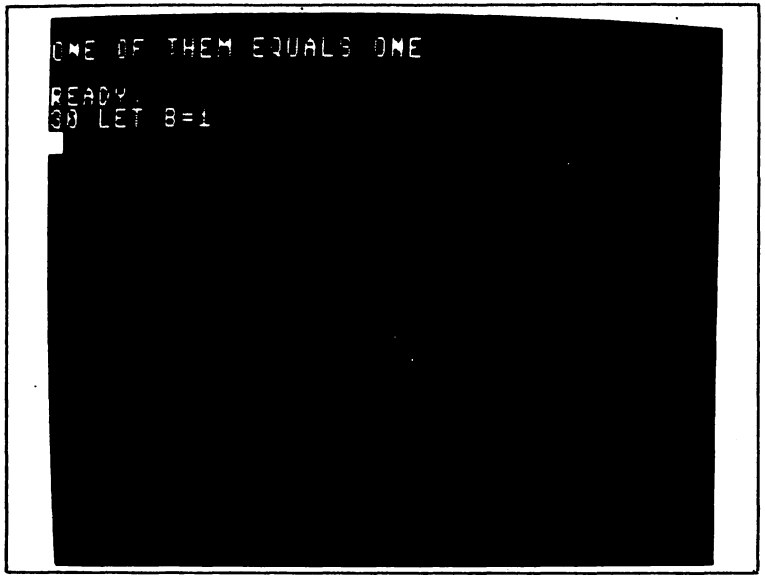
155. Let's clear the screen, then clear out the computer's memory, and move on to a few more conditional words we can profitably use in programming. Press the SHIFT and CLR/HOME keys; then type NEW (and press RETURN).



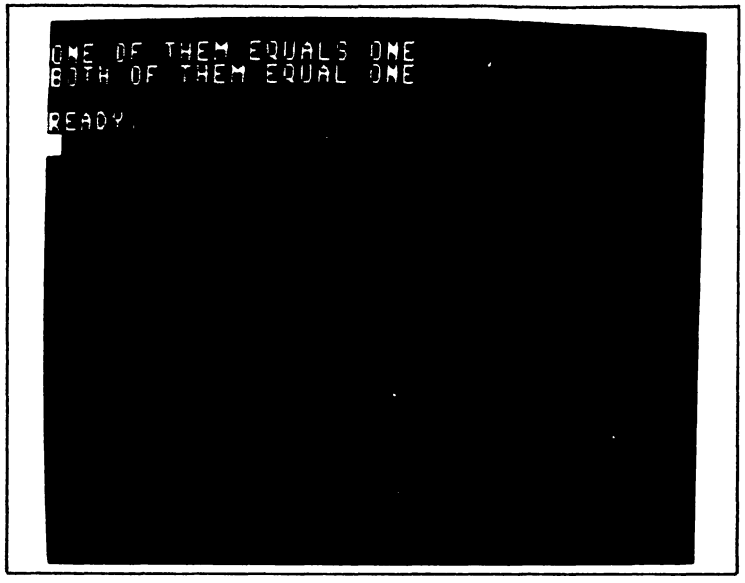
156. Caretully type in: 10 ?"♥" (press RETURN), 20 LET A=1 (press RETURN), 30 LET B=2 (press RETURN), 40 IF A=1 OR B=1 THEN ? "ONE OF THEM EQUALS ONE" (press RETURN), 50 IF A=1 AND B=1 THEN ? "BOTH OF THEM EQUAL ONE" (press RETURN).



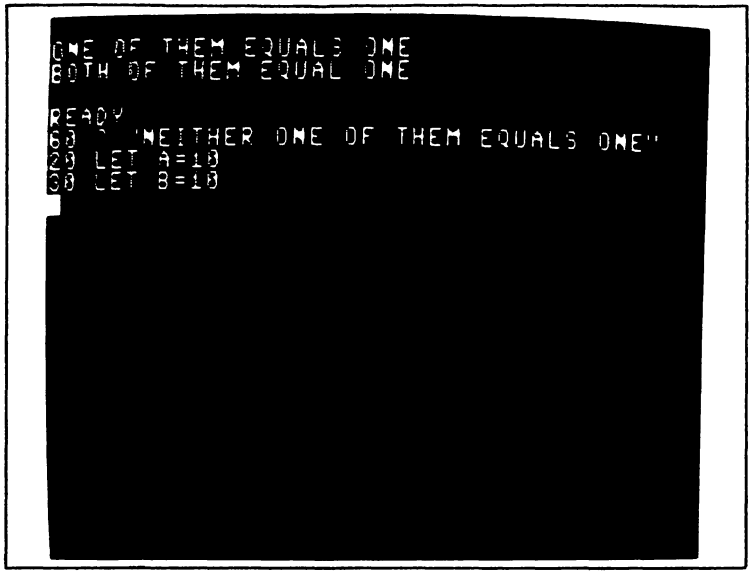
157. RUN the program. The computer checked the values of A and B against the conditions of lines 40 and 50 and found that only line 40's conditions were met. So it executed the PRINT command on line 40, and having nothing else to do, advised that it was ready for more.



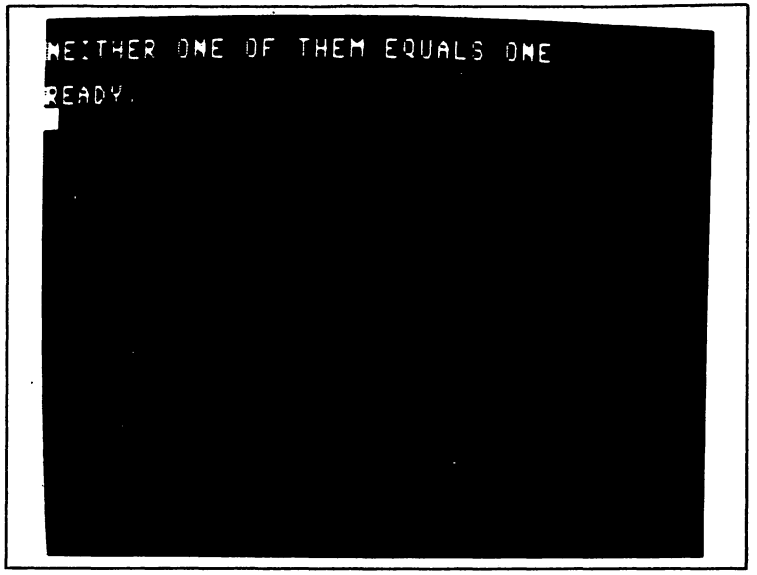
158. Change the value of B in memory to 1, by typing in: 30 LET B=1 (press RETURN).



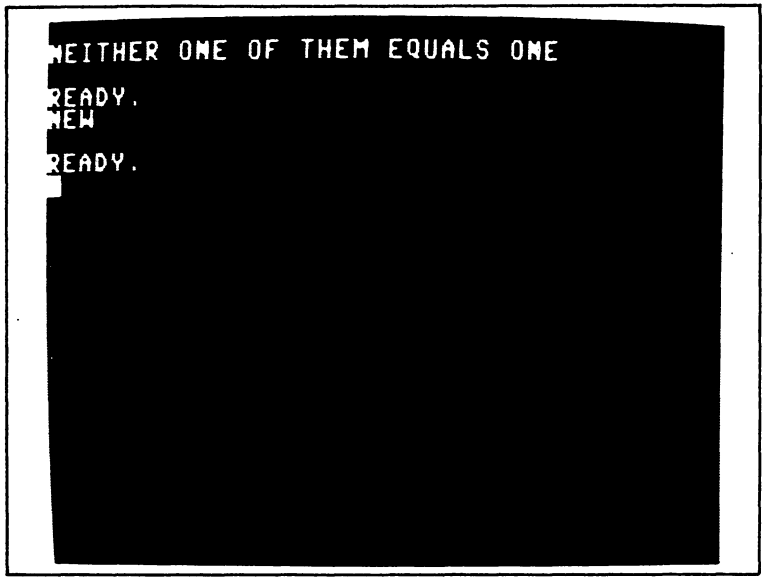
159. Now RUN the program. You can't fool the C64. It checked the latest values of A and B against the conditions of the program lines and found that now the conditions for both line 40 and line 50 were met, so it executed the PRINT commands of both lines.

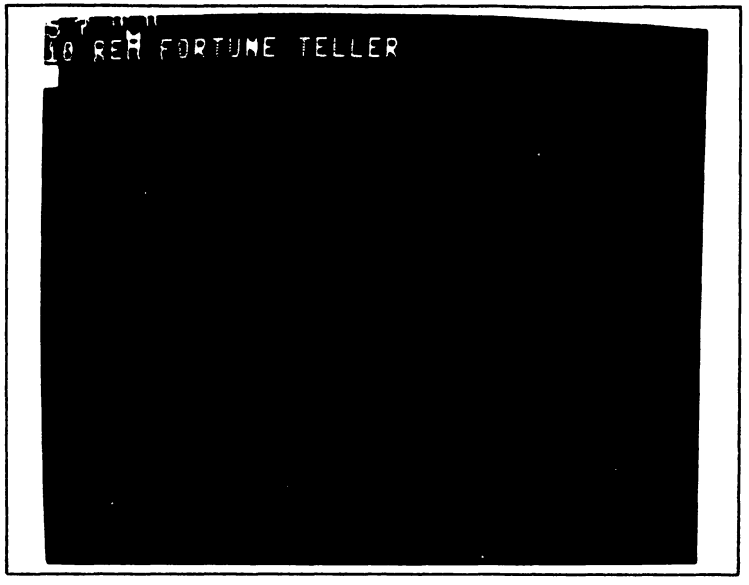


160. Type in: 60 ? "NEITHER ONE OF THEM EQUALS ONE" (press RETURN), 20 LET A=10 (press RETURN), 30 LET B=10 (press RETURN).



161. RUN the program. Your screen will now tell you that neither A nor B equals 1 because the computer found that the conditions called for on lines 40 and 50 weren't met by the values of A and B. So it did the only thing it could; it continued to line 60, which had no conditions, and executed its PRINT command.

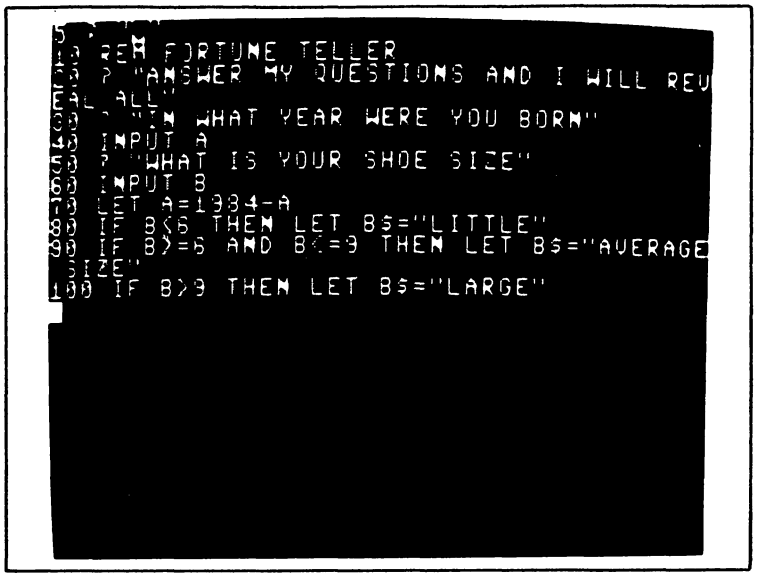




WHEN YOUR COMPUTER ASKS YOU QUESTIONS

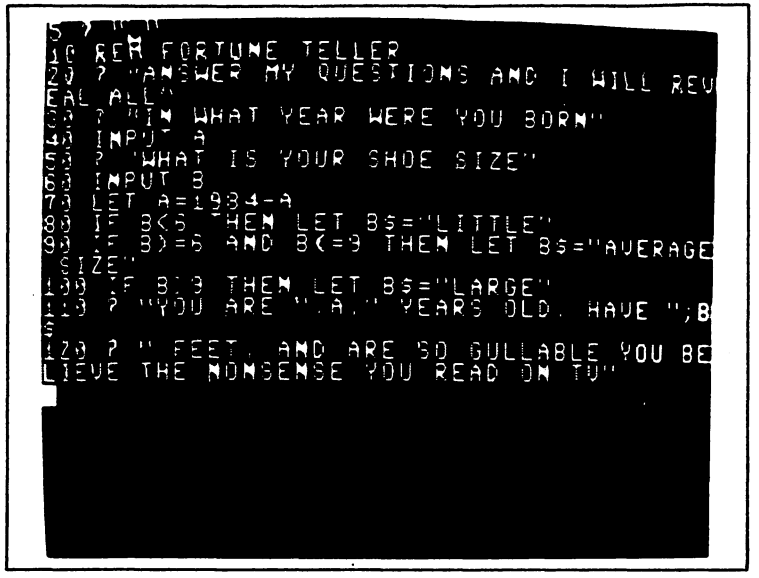
162. Let's put everything we have done so far to use to illustrate a new command called INPUT. When you use INPUT in a program, the computer will stop executing the program and wait until you give it something the program needs before it goes on. We'll start by clearing the last program out of memory by typing: NEW (and pressing RETURN).

163. Clear the screen, then type in: 5 ? "♥" (press RETURN), 10 REM FORTUNE TELLER (press RETURN). Your programs won't execute REM statements. They are nothing more than REMarks or reminders.



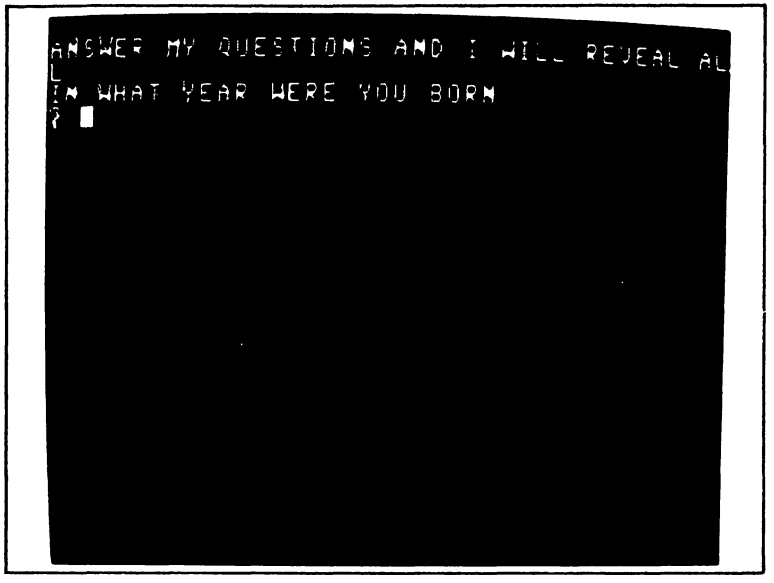
164. Now type in the following program lines and press RETURN after each one to enter it into memory: 20 ? "ANSWER MY QUESTIONS AND I WILL REVEAL ALL" 30 ? "IN WHAT YEAR WERE YOU BORN" 40 INPUT A 50 ? "WHAT IS YOUR SHOE SIZE" 60 INPUT B 70 LET A=1984−A 80 IF B<6 THEN LET B$="LITTLE" 90 IF B>=6 AND B<=9 THEN LET B$= "AVERAGE SIZE" 100 IF B>9 THEN LET B$="LARGE."

NOTE: If you are reading this in a year other than 1984, use the current year in line 70.



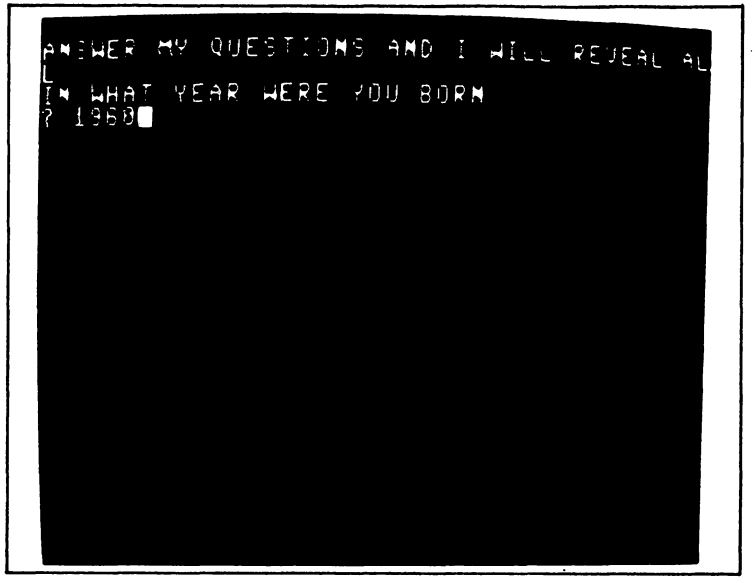
165. To finish the program, type in: 110 ? "YOU ARE ";A;" YEARS OLD, HAVE ";B$; (press RETURN), 120 ? " FEET, AND ARE SO GULLABLE YOU BELIEVE THE NONSENSE YOU READ ON TV" (press RETURN).

NOTE: The C64 limits the length of program lines. You can't put more than two screen lines into a single program line. Since I knew that the length of this PRINT statement would exceed that limit I chose to break the statement in the middle, at a convenient place, where a string variable was being printed and the line required a semicolon. Ending a PRINT statement with a semicolon causes the next PRINT statement to begin at the semicolon. If that's less than per-

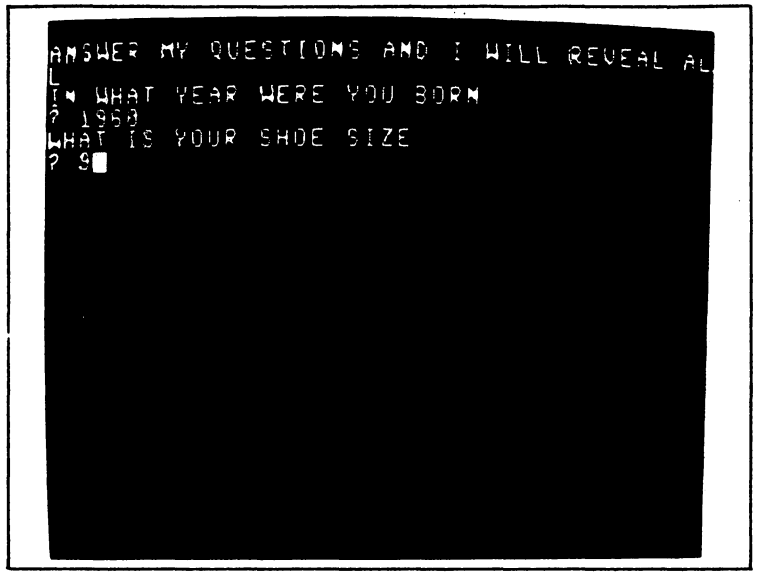


fectly clear, don't worry about it. I'll demonstrate this feature later in the book, in a much simpler manner.

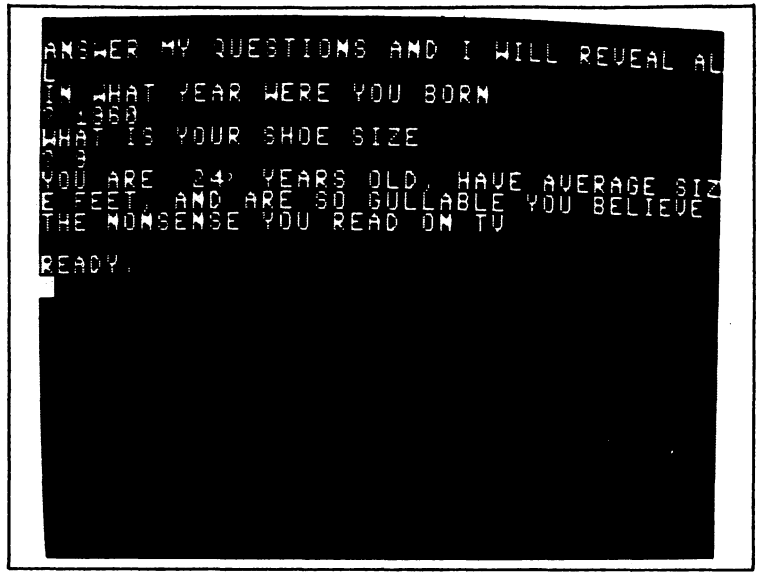
166. Now RUN the program. Notice that the program is executed, but only to a point. Line 5 cleared the screen and brought the cursor back to HOME. The PRINT statements in lines 20 and 30 were executed. But what's this question mark followed by a blinking cursor doing up on the screen? It is the result of line 40 being executed. An INPUT command causes a question mark to be printed on the screen and the cursor to blink next to it, to remind you that the computer has asked you a question.



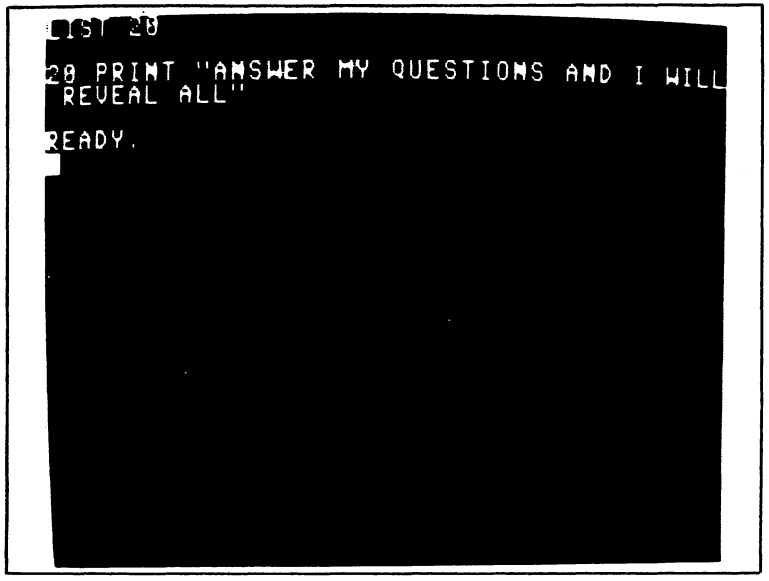
167. Now you'll have to answer the question before the program can go any further. Let's say, for the sake of this example, that you were born in 1960. Type in: 1960.



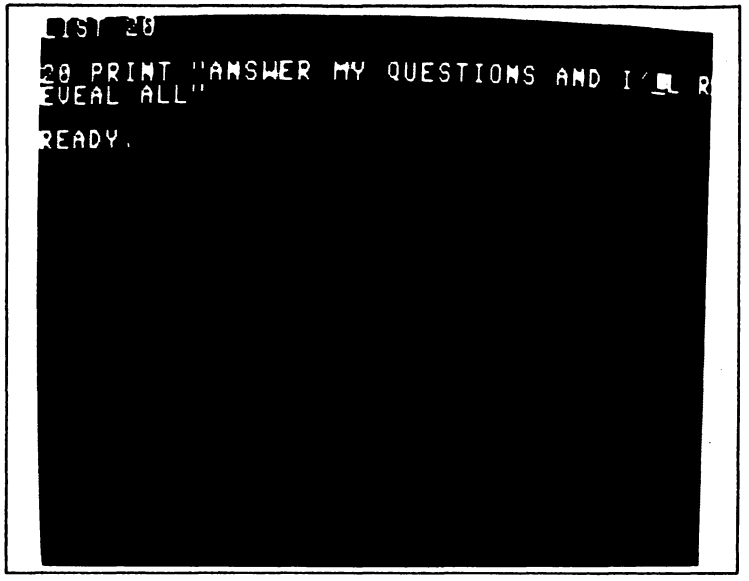
168. As soon as you press RETURN, you are asked (by line 50) to supply your shoe size. Again, just for the heck of it, let's say you wear a size 9 shoe. Type in: 9.



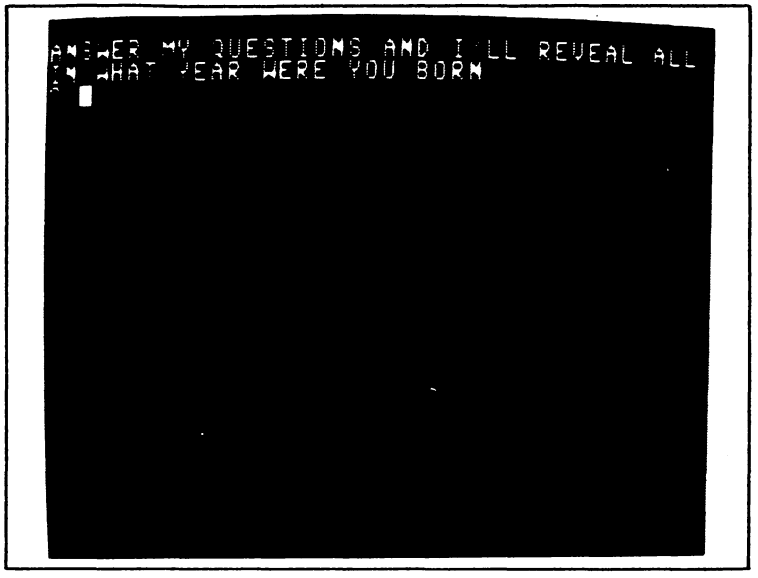
169. Now when you press RETURN the computer goes on to inform you of your age, the relative size of your feet, and your gullability. Having done all it was asked to do, it also lets you know that it is ready for more.



170. There is a lot that can be done to improve this silly little program. To begin with, look at illustration 169 and notice, in the opening sentence, that the word "ALL" is broken and wrapped around onto the next line. That's easy to fix. Clear the screen, then type: LIST 20 (and press RETURN).

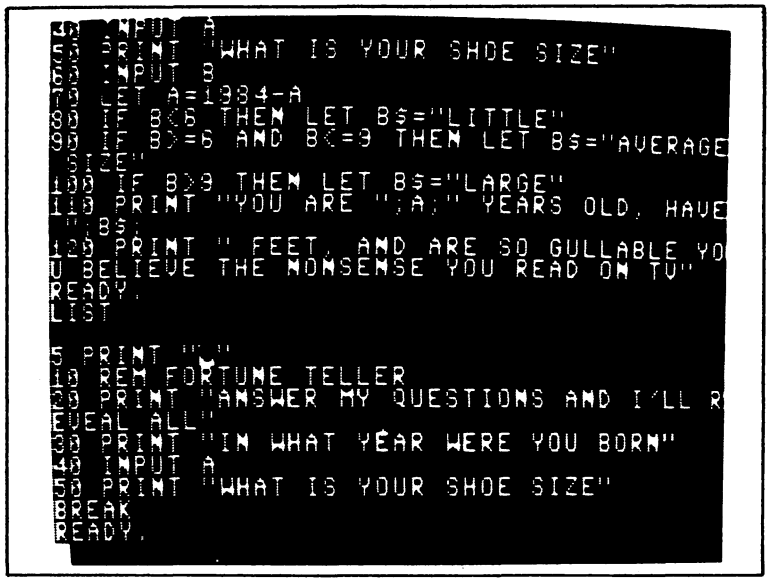


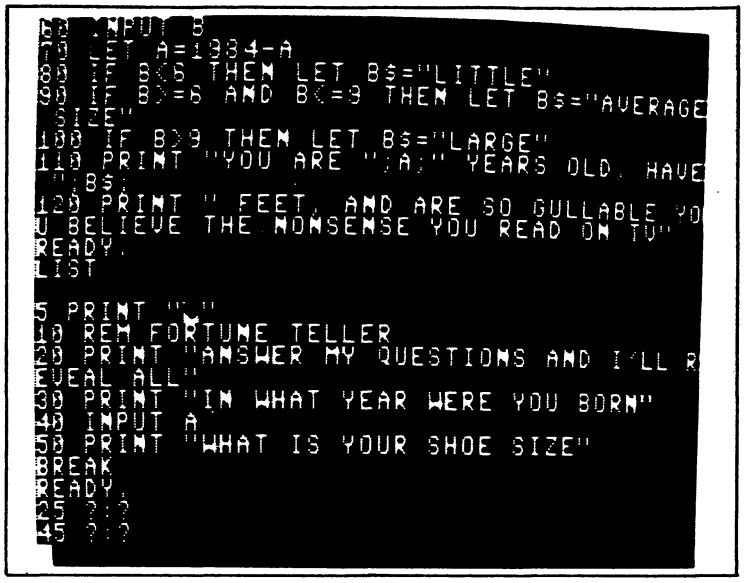
171. Use the SHIFT and CRSR keys to move the cursor to the first L in the word WILL. Press the INST/DEL key twice to get rid of the letters W and I. Then use the SHIFT and horizontal CRSR keys to move the cursor to the space between I and the remaining two Ls. Press the SHIFT and 7 keys to put in an apostrophe (') and to form the contraction, I'LL.



172. Press RETURN to enter the modified line 20 into memory, clear the screen, then RUN the program. That's a bit better, isn't it? But the screen is still rather confusing because the opening statement is not separated from the first question.

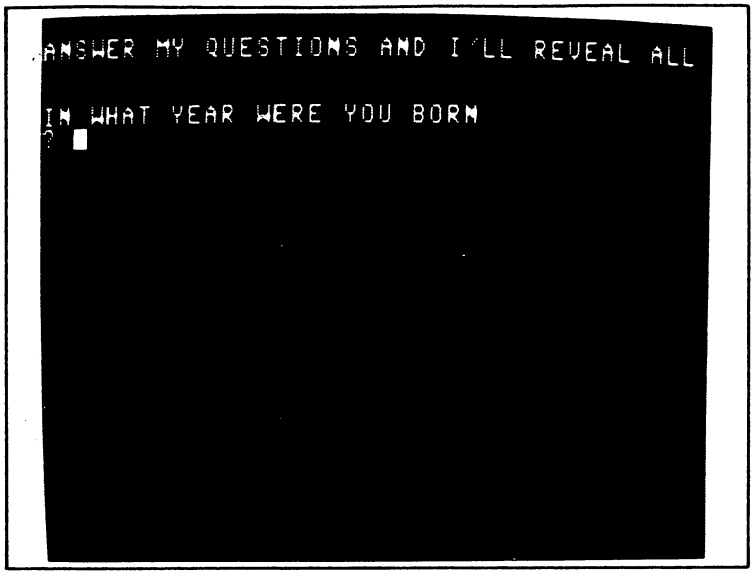
173. Press RUN/STOP while pressing RESTORE to get out of the program.



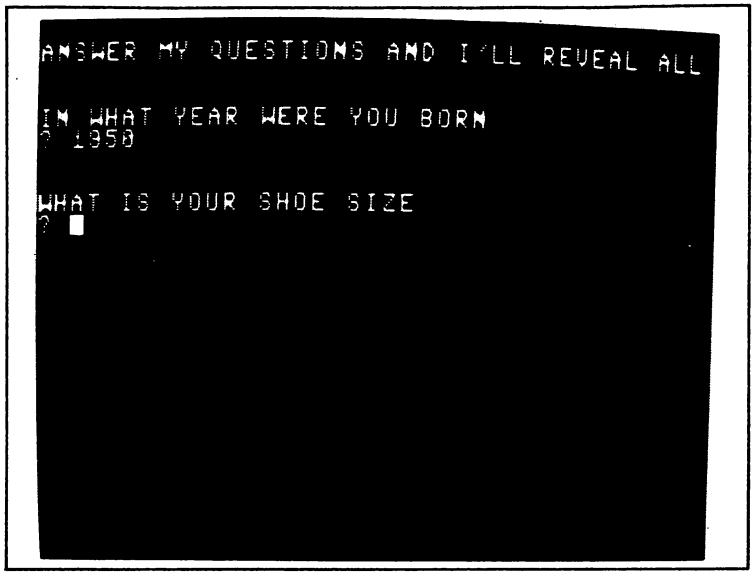


174. Then type: LIST (and press RETURN). Please pardon a momentary digression here, but I'd like to show you a useful trick. Type: LIST (and press RETURN) again. The moment you lift your finger from the RETURN key, press the CTRL key. Doing so will cause the program to LIST slowly, one line at a time. The moment you see line 50 or 60 appear, release the CTRL key and immediately press the RUN/STOP key. Doing so will cause the program lines to stop scrolling up onto the screen, and will also cause the words "BREAK" and "READY" and the cursor to appear. That procedure probably sounds awfully complicated, but it really isn't. Try it several times, until you get the hang of it. It is a very useful procedure for any programmer, as it allows you to examine your program lines slowly, and to stop them for correction when you find a problem. Notice that the procedure only begins to work after the first 25 screen lines have been filled. That's why I had you initially LIST the program twice.

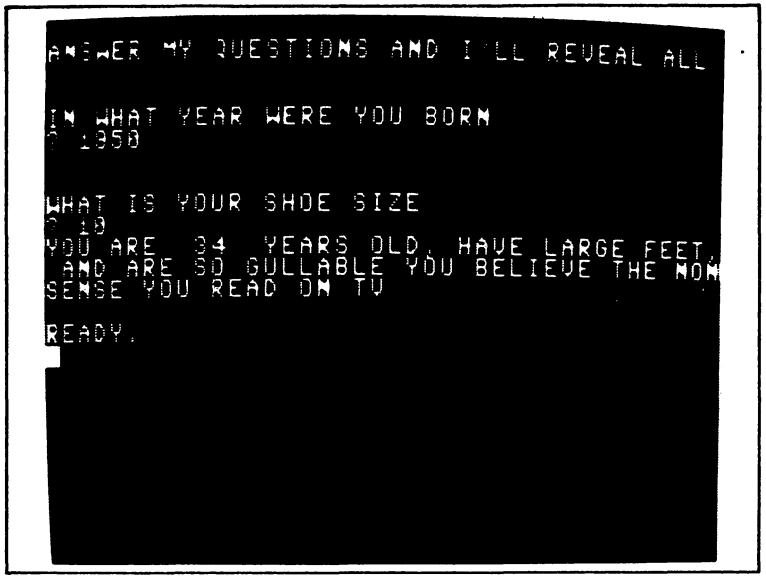
175. Now you can see that if you add a couple of PRINTs between 20 and 30, and between 40 and 50, the screen will look less confusing when the program is RUN. Type in 25 ?:? (press RETURN), 45 ?:? (press RETURN).



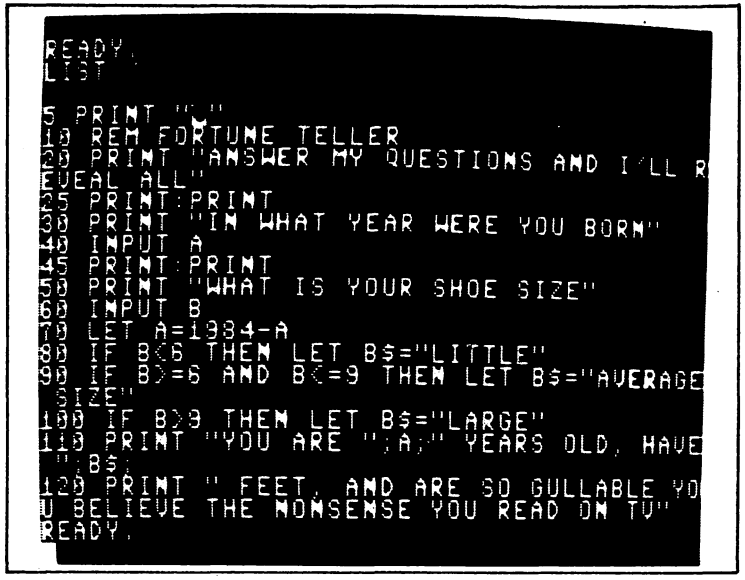
176. Clear the screen, then RUN the program. There now, that's beginning to look better, isn't it?



177. Type in 1950 (and press RETURN) to see if the rest of the spacing you just added works well. It certainly does.



178. Tell it 10 (and press RETURN). Hmm . . . the presentation of the punch line can use some improvement as well.



179. Get the program LISTed slowly, and stop it when you see the place you want to make a change. Actually, that turns out to be close enough to the end so that you can just type LIST (and press RETURN), but I thought the review might be good for you.

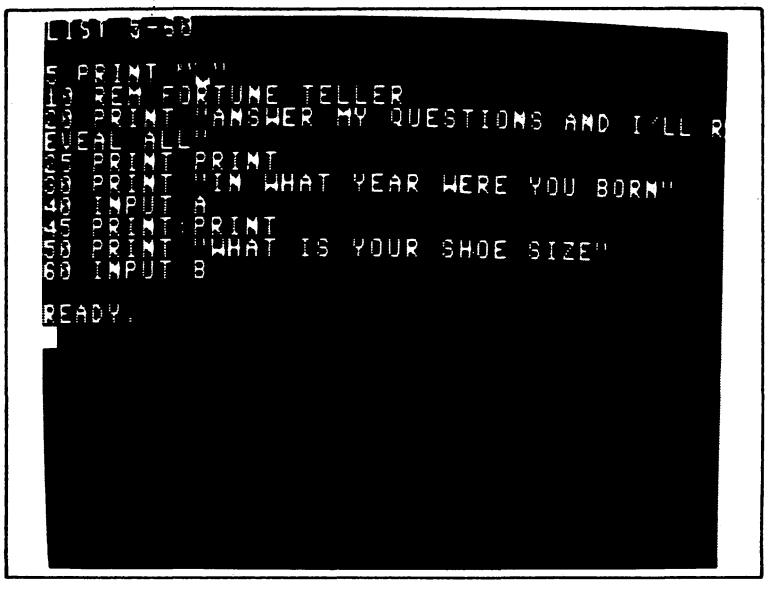
```
READY.
LIST

 5 PRINT "■"
 10 REM FORTUNE TELLER
 20 PRINT "ANSWER MY QUESTIONS AND I'LL R
EVEAL ALL"
 25 PRINT:PRINT
 30 PRINT "IN WHAT YEAR WERE YOU BORN"
 40 INPUT A
 45 PRINT:PRINT
 50 PRINT "WHAT IS YOUR SHOE SIZE"
 60 INPUT B
 70 LET A=1984-A
 80 IF B<6 THEN LET B$="LITTLE"
 90 IF B>=6 AND B<=9 THEN LET B$="AVERAGE
 SIZE"
 100 IF B>9 THEN LET B$="LARGE"
 110 PRINT "YOU ARE ";A;" YEARS OLD, HAVE
 ";B$;
 120 PRINT " FEET, AND ARE SO GULLABLE YO
U BELIEVE THE NONSENSE YOU READ ON TV"
READY.
105 ? "■"
```

180. Perhaps the most interesting way to present the punch line would be to clear the screen before the delivery. So let's add: 105 ? "■" (and press RETURN). Remember to use the SHIFT and CLR/HOME keys.

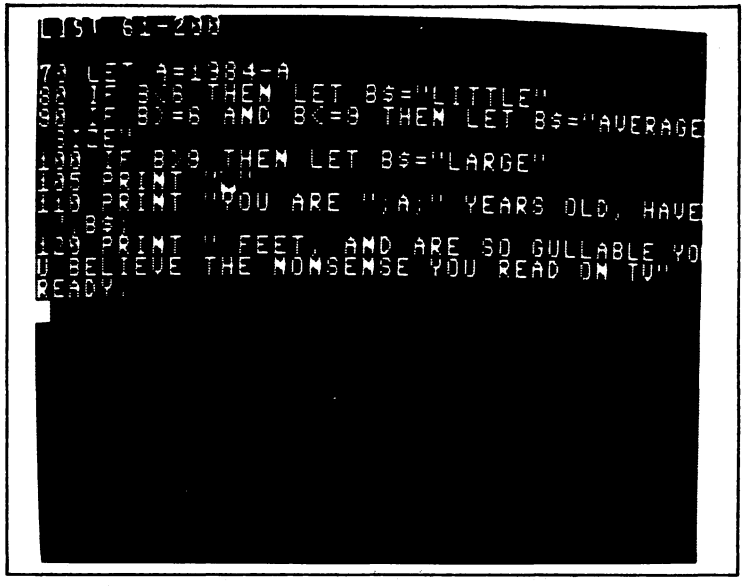
```
YOU ARE  34  YEARS OLD, HAVE LARGE FEET,
 AND ARE SO GULLABLE YOU BELIEVE THE NON
SENSE YOU READ ON TV

READY.
```

181. Now RUN the program again. Give it any answers you wish. There now, that's a lot better, isn't it?

NOTE: I am well aware that, depending on your age and the size of your feet, several of the words in the punch line may be broken and wrapped around the screen. Correcting the program, however, to avoid this is not straightforward because B$ varies in length, depending on the shoe size. (Of course, A can also vary if you let kids under ten and seniors over 100 play with the program.) It is left as an exercise for you, dear reader, to improve upon this presentation.

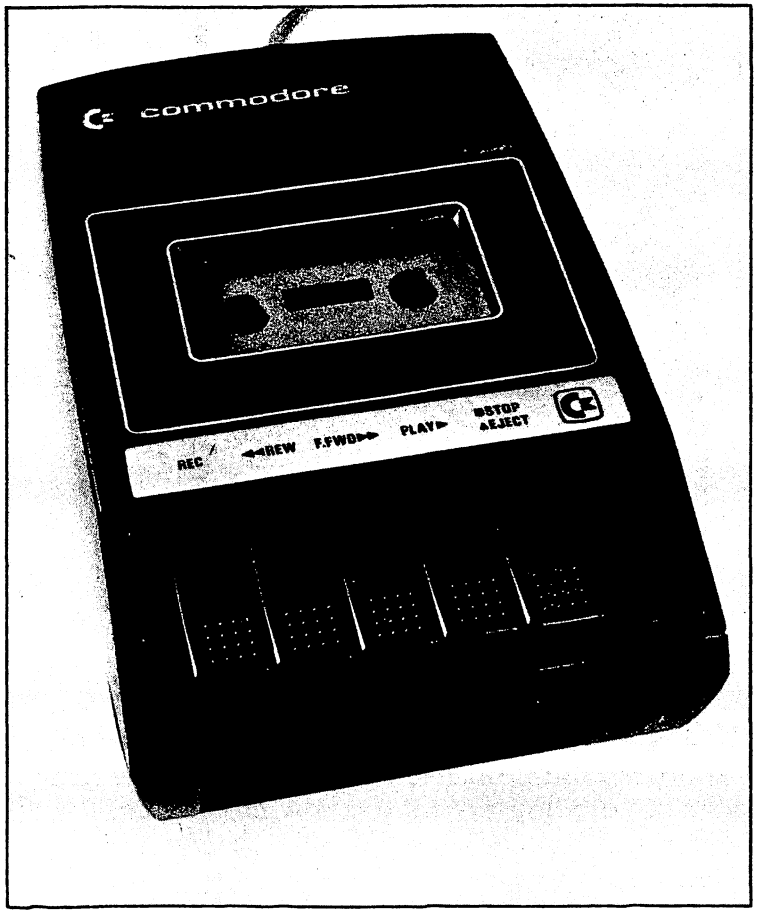


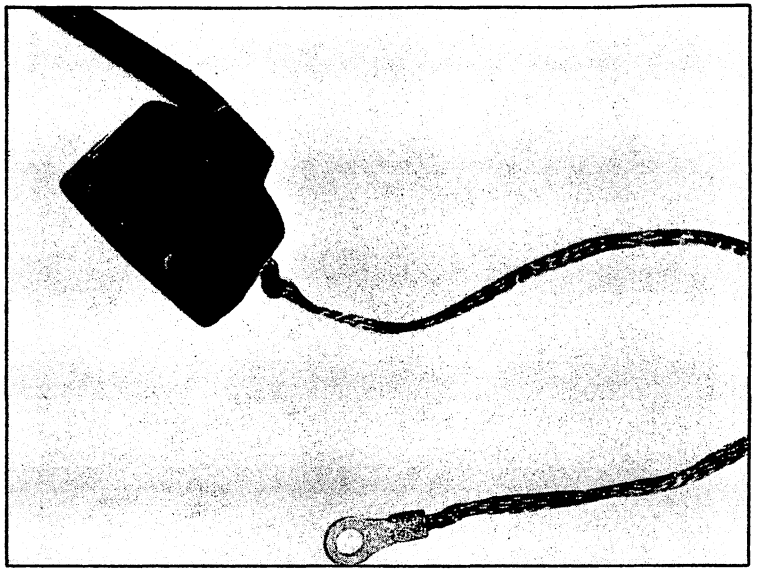
182. You may wish to make use of this program again, so I have cleared the screen and shown it, in manageable sections, in this and the next illustration. Notice that you are able to get any reasonable amount of any program listed, just the way you want it, by typing LIST, followed by the number of the first program line you want shown, a hyphen (-), and the number of the last program line you want shown.



183. If you're in doubt, pick a pair of numbers you think will bracket the program section you're after.

NOTE: If possible, try not to turn your C64 off or type the command NEW into the machine until you have completed the following section. It will show you how to save this silly program on tape, if you wish to.



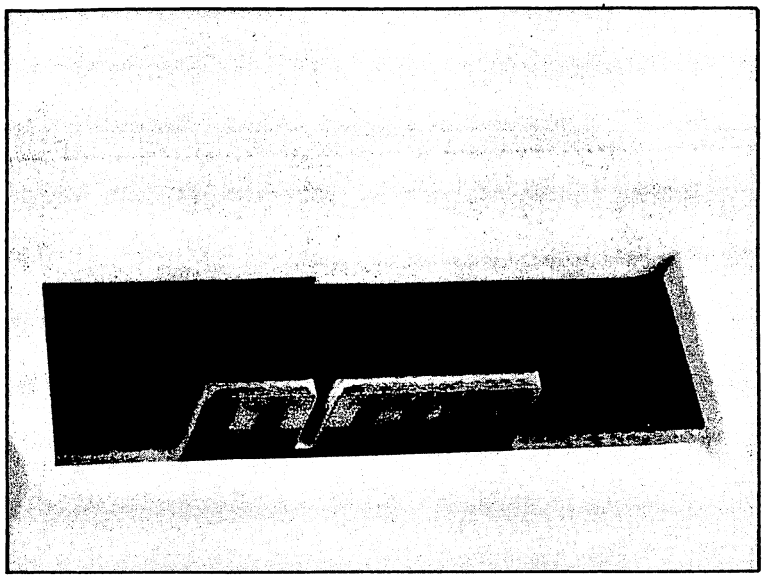


THE TAPE RECORDER

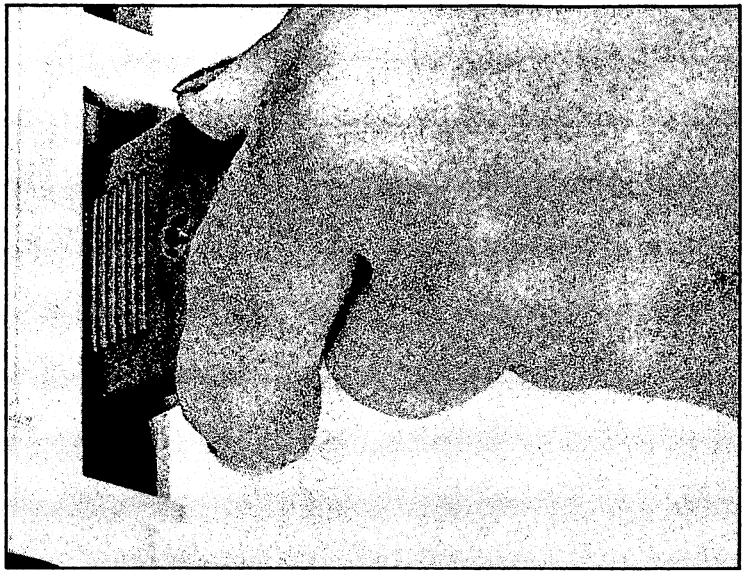
Saving Your Programs

184. Datassettes come in several different models—black ones, white ones, long ones, short ones, some with tape counters, some without, some with lights, some without. Regardless of which model you have, the instructions offered here will work.

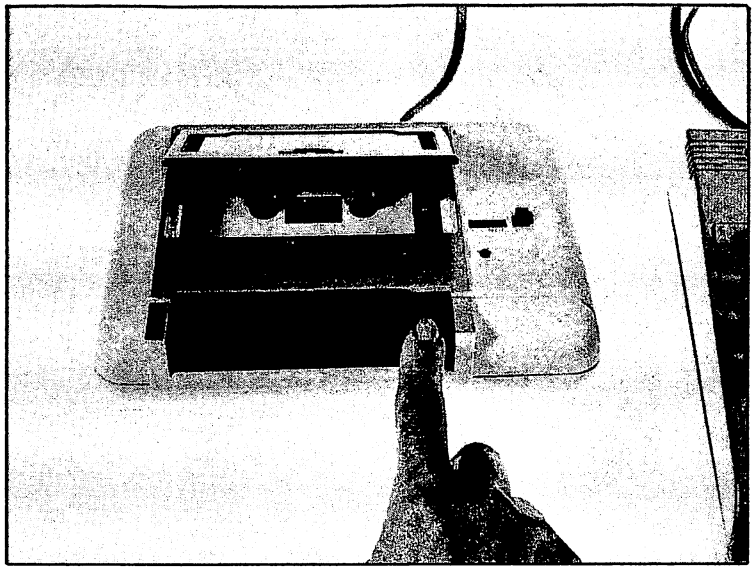
185. Saving your programs on tape is quite simple, after you have done it successfully a few times, but until then it can be a somewhat confusing experience. Begin by looking at the plug on the end of your Datassette cable. Note that the braided metal ground strap, coming from the side of the plug, is not used with the C64. Simply tape it to the cable, out of the way. Notice that the plug itself has a plastic separator between the two rightmost metal connectors and the four to their left.



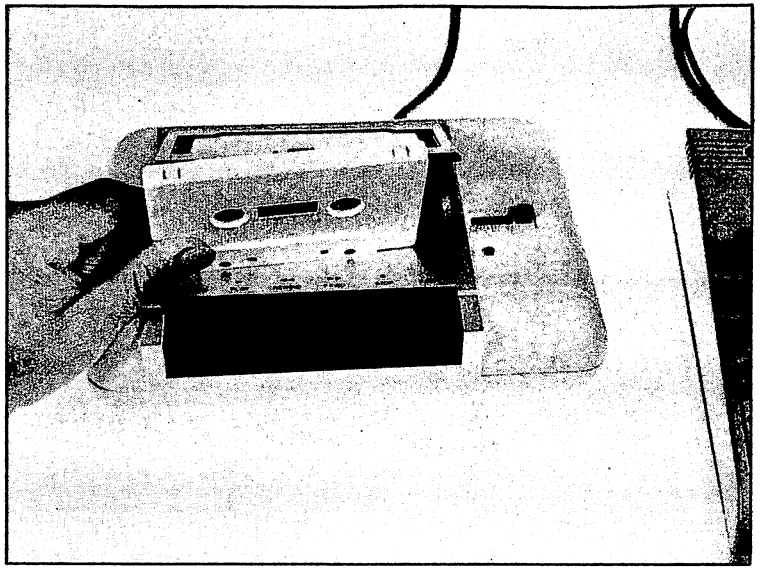
186. Now look at the PC-board edge connector in the small slot just to the left of center, at the rear of the C64. Notice that it has a slot cut in it.



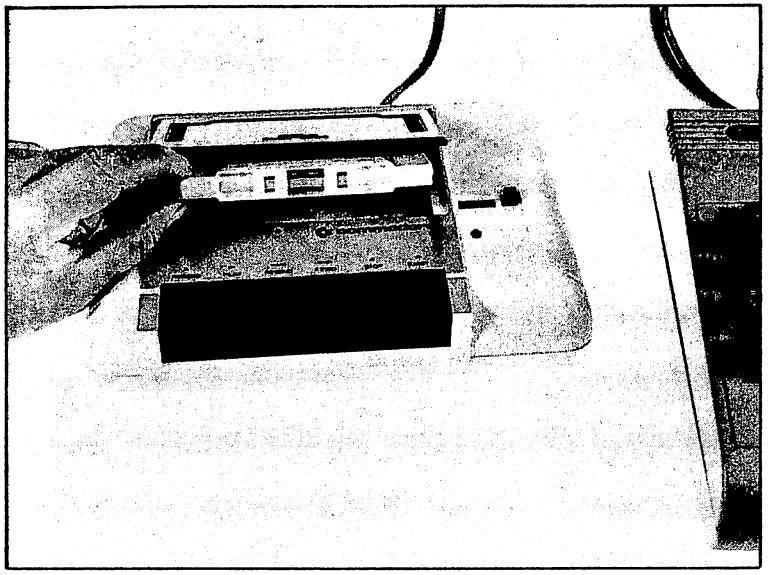
187. Hold the Datassette plug so that the plastic separator will slide into the slot in the PC board. Push the plug gently but firmly onto the board.



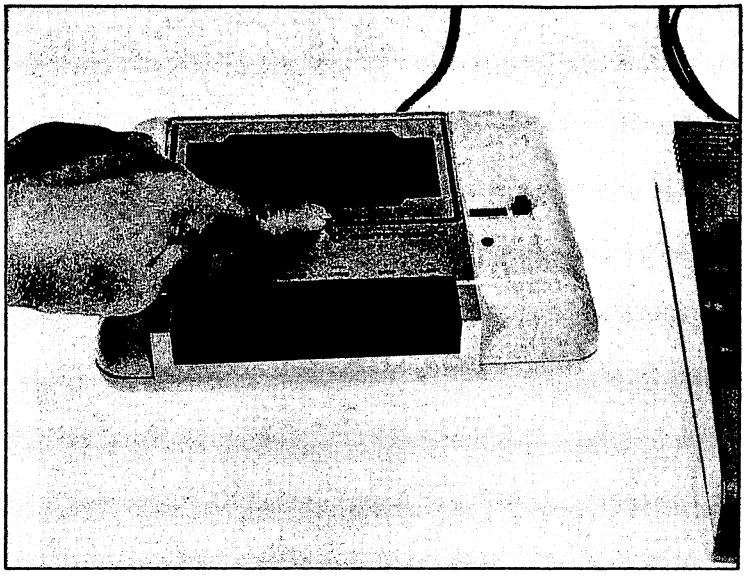
188. Press the EJECT lever on the Datassette to open the tape compartment door.



189. Select a blank C-5, C-10, or C-20 tape cassette. Be sure the cassette's record tabs are intact and that it is rewound (tape on the left hub) to the end of the leader.

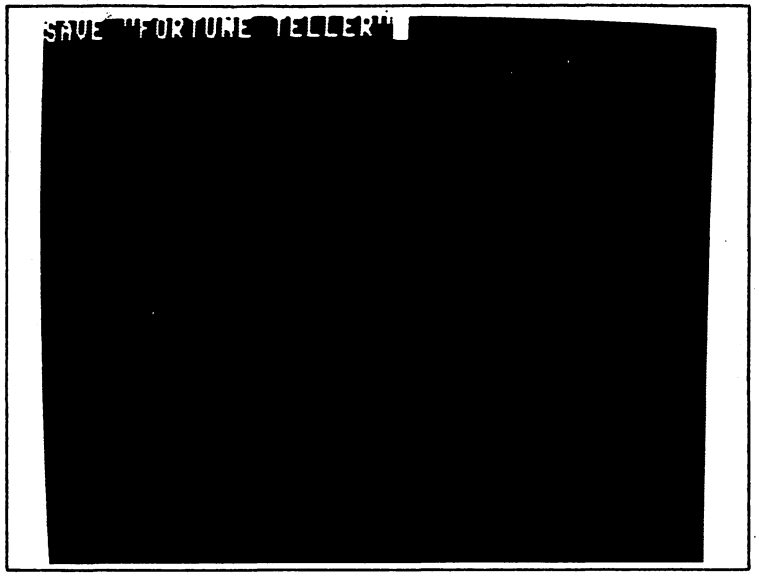


190. Insert the cassette into the holder on the door.

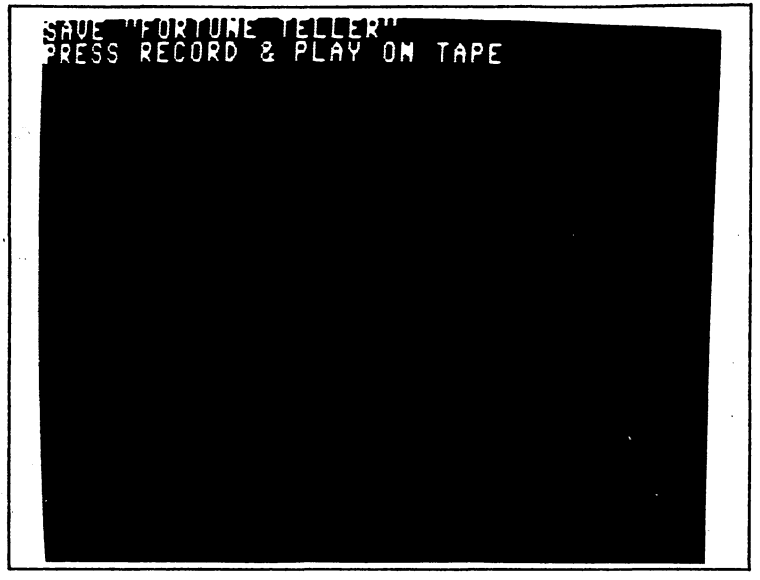


191. Close the door by pressing gently but firmly downward until it latches, with an audible click, into place.

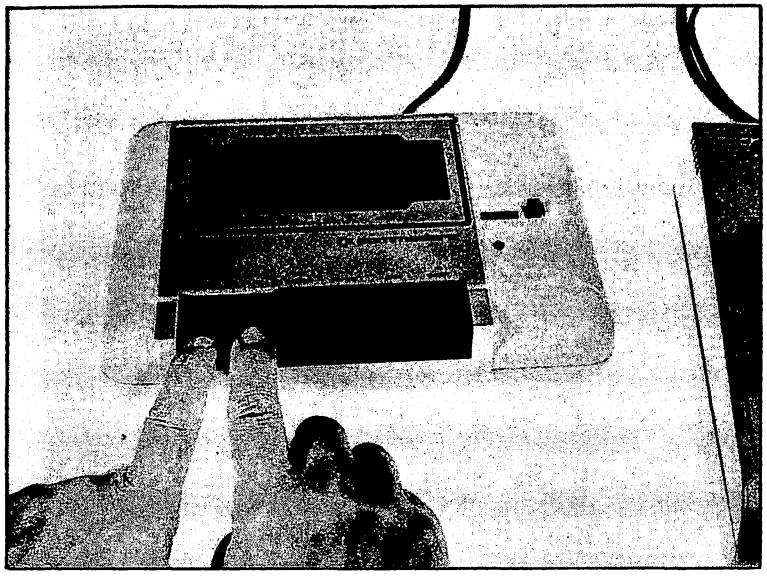
NOTE: If your Datassette is equipped with a tape counter, now is the time to push the button and reset it to 000.



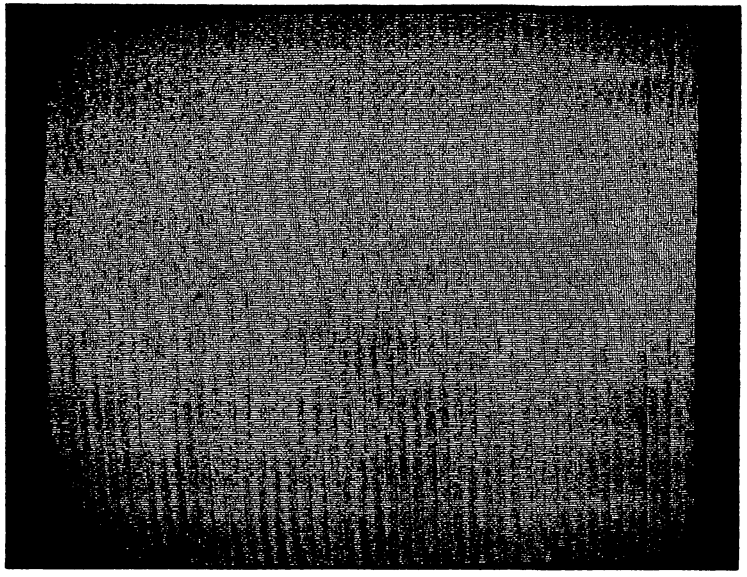
192. The FORTUNE TELLER program should still be loaded in memory. If it isn't, check illustrations 182 and 183 and type it in. Clear the screen, then type in: SAVE "FORTUNE TELLER."



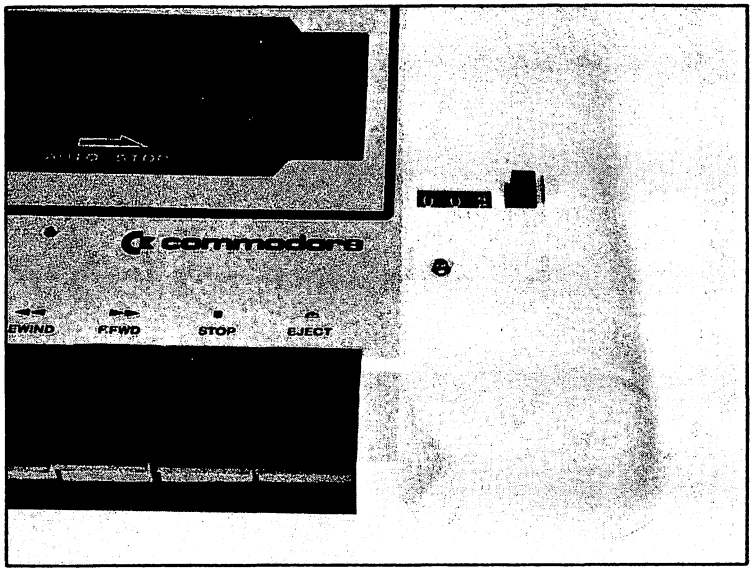
193. Press RETURN and the computer will give you the instructions: PRESS RECORD & PLAY ON TAPE.



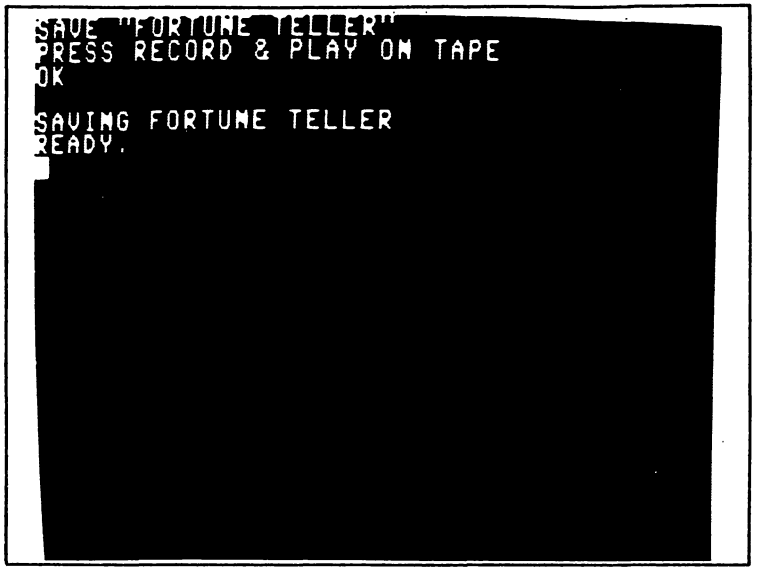
194. Do just what it tells you to do.



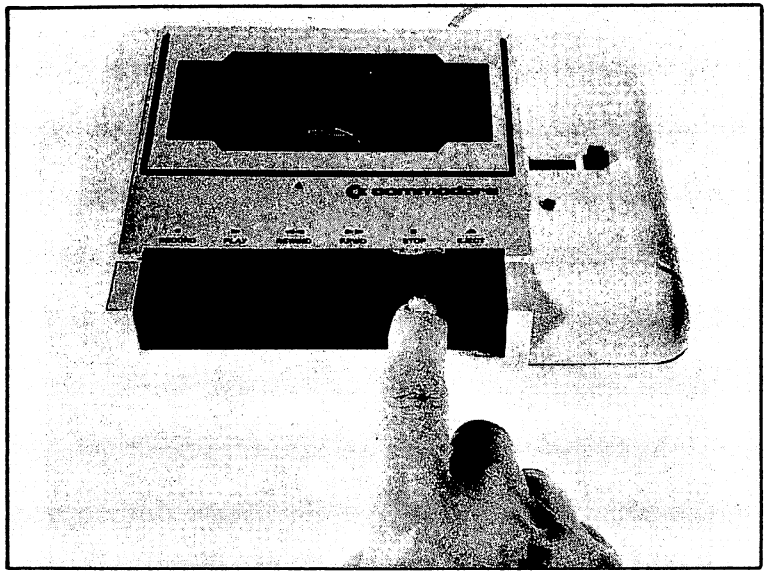
195. As soon as you do, several things will happen. The screen will go completely blank.



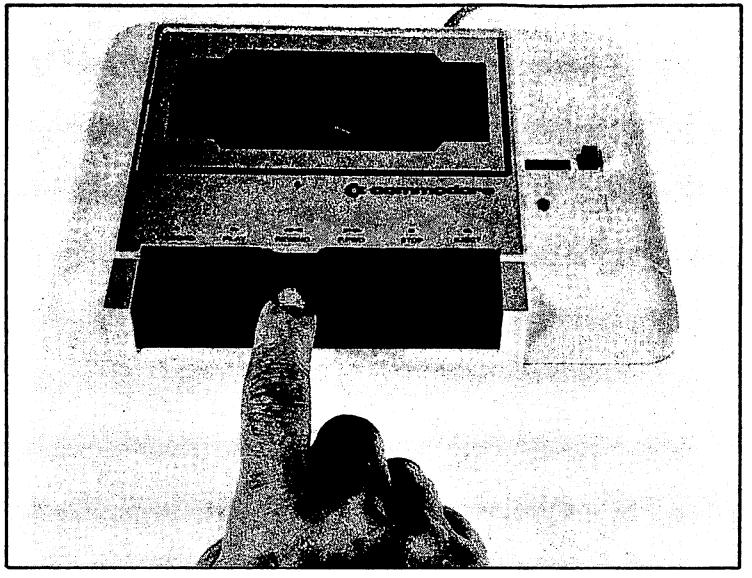
196. If your Datassette is equipped with one, the SAVE-indicator light will go on. Simultaneously, you will hear the motor begin to run, the tape hubs will begin to turn, and the counter dials (if there are any) will begin to move.



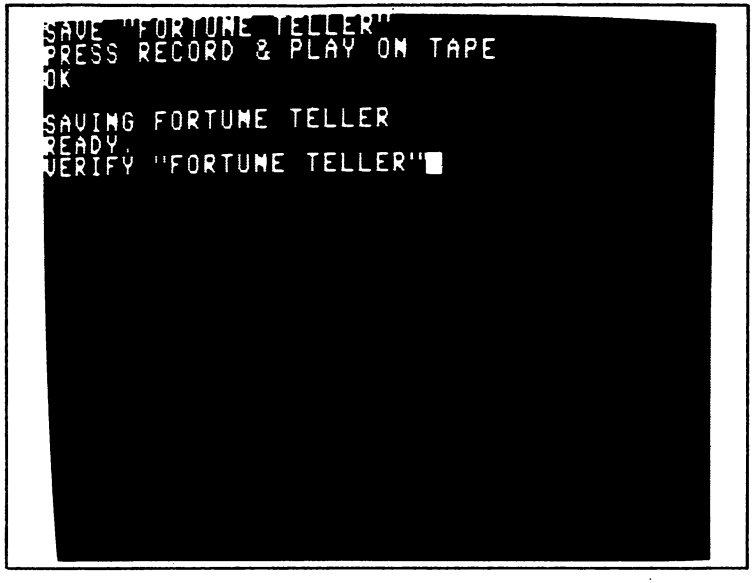
197. After about 25 seconds the Datassette SAVE light will go out, the motor will turn off, the tape hubs will stop turning, and the screen will suddenly reappear, this time bearing the messages, OK, SAVING FORTUNE TELLER, and READY, followed by a blinking cursor.



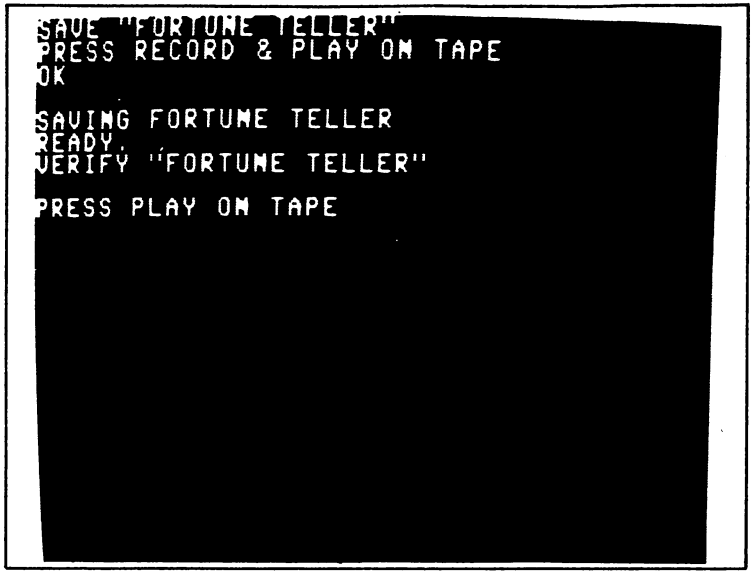
198. Now for the moment of truth. It's time to discover if the program was properly recorded. Begin by pressing the STOP lever on the Datassette.



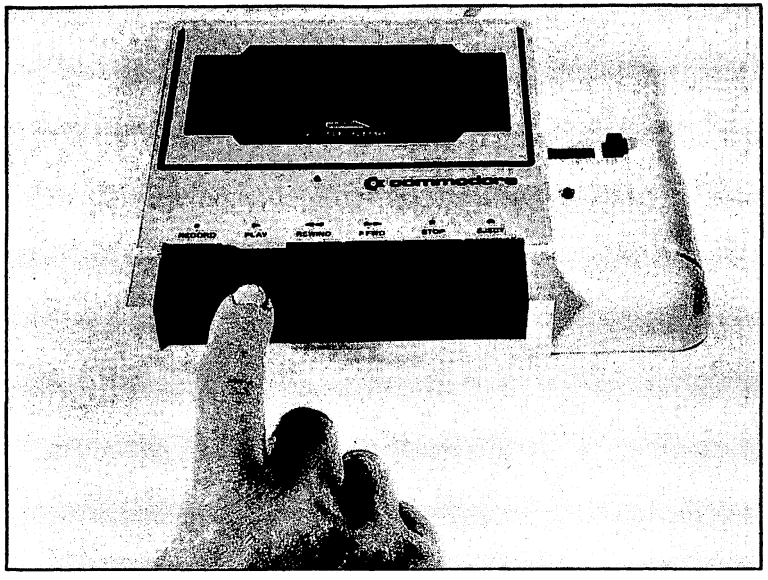
199. Then press the REWIND lever to rewind the tape to the beginning.



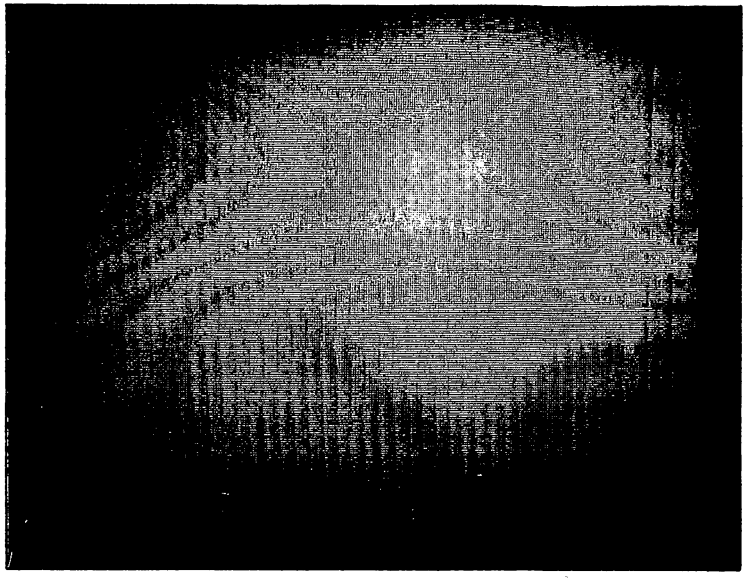
200. Press STOP to release the REWIND lever and type into the computer: VERIFY "FORTUNE TELLER."



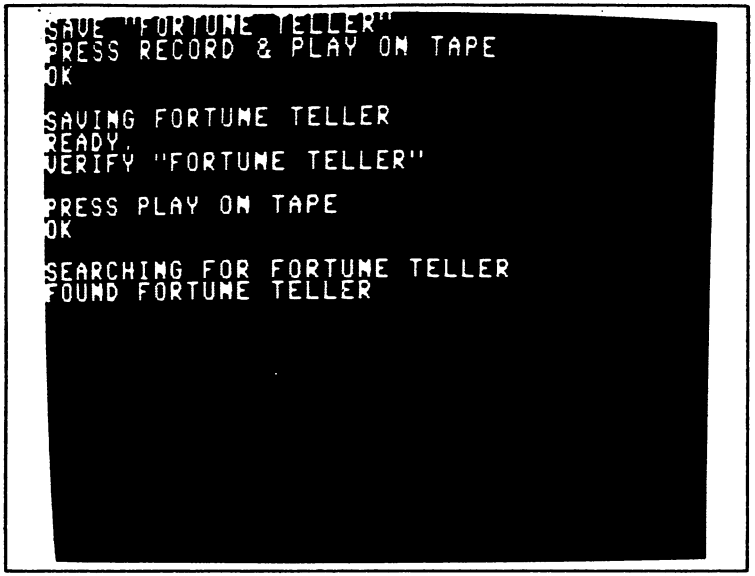
201. Press RETURN and again the C64 will give you an instruction: this time to PRESS PLAY ON TAPE.



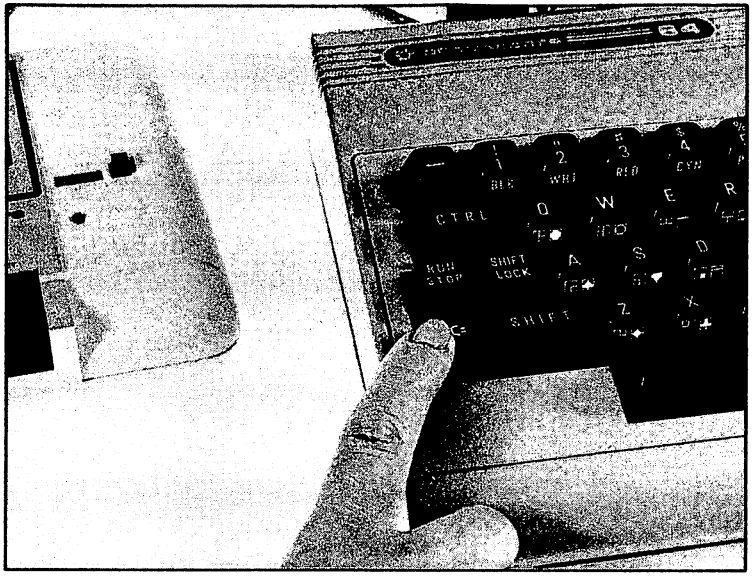
202. Do just as it asks.



203. And again things will begin to happen. First the screen will go blank and the tape hubs will begin to turn.

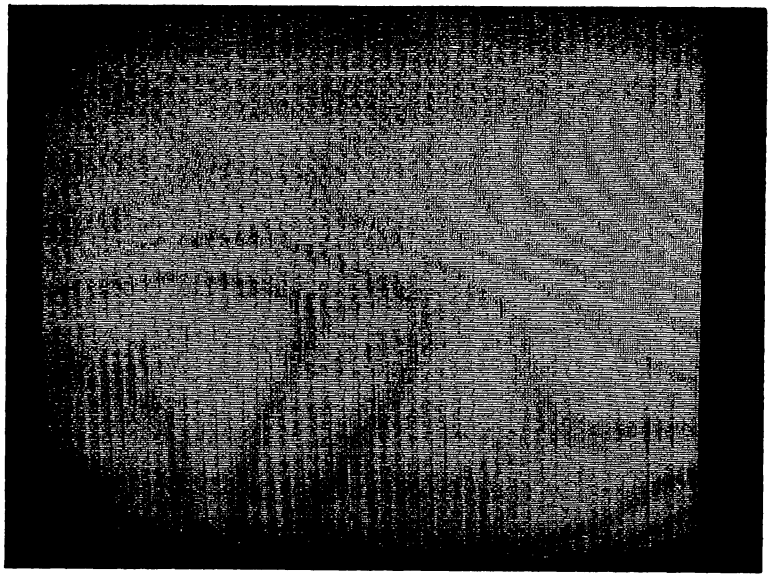


204. Then, after a few seconds, the computer will give you the message: OK, SEARCHING FOR FORTUNE TELLER, FOUND FORTUNE TELLER.

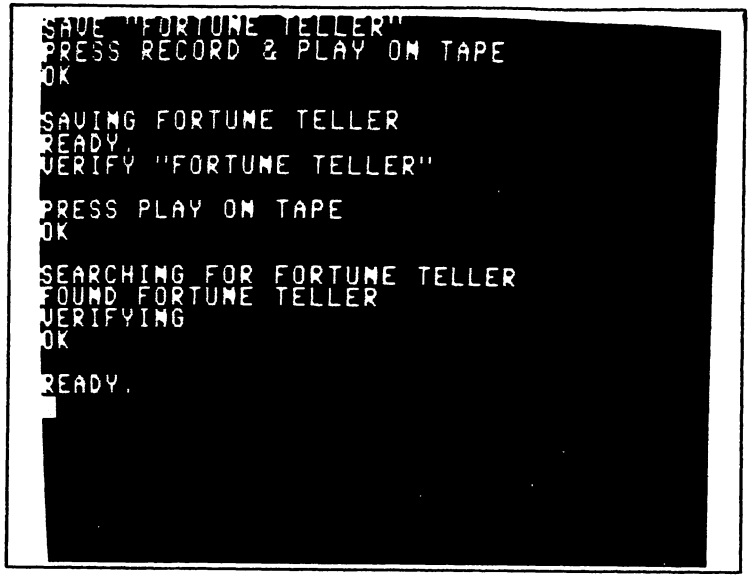


205. That's your cue to press the COMMODORE key.

NOTE: If you don't press the COMMODORE key, the computer will wait a few seconds, assume you dozed off, and will proceed to verify the program all by itself.

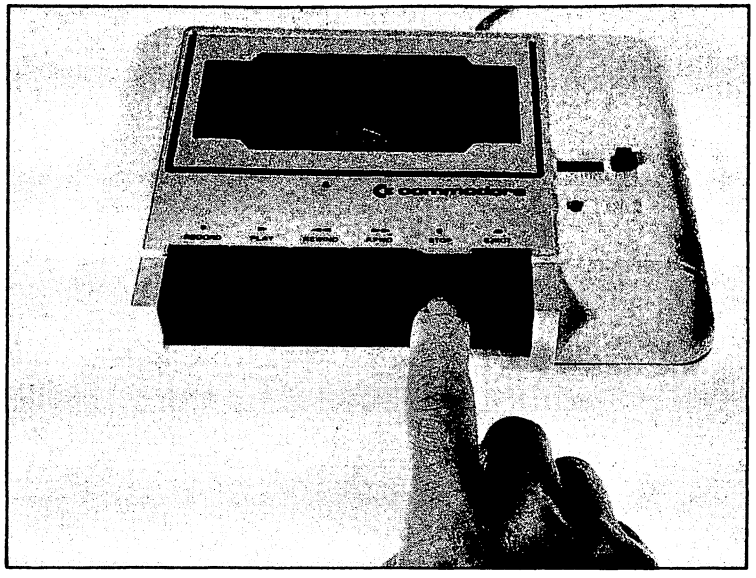


206. Again the screen will go blank and the tape hubs will begin to turn.



207. Then, after a few seconds, if the tape recording of the program is found to be flawless, you will get the message, VERIFYING, OK, READY, and a blinking cursor.

```
SAVE "FORTUNE TELLER"
PRESS RECORD & PLAY ON TAPE
OK

SAVING FORTUNE TELLER
READY.
VERIFY "FORTUNE TELLER"

PRESS PLAY ON TAPE
OK

SEARCHING FOR FORTUNE TELLER
FOUND FORTUNE TELLER
VERIFYING
?VERIFY  ERROR
READY.
```

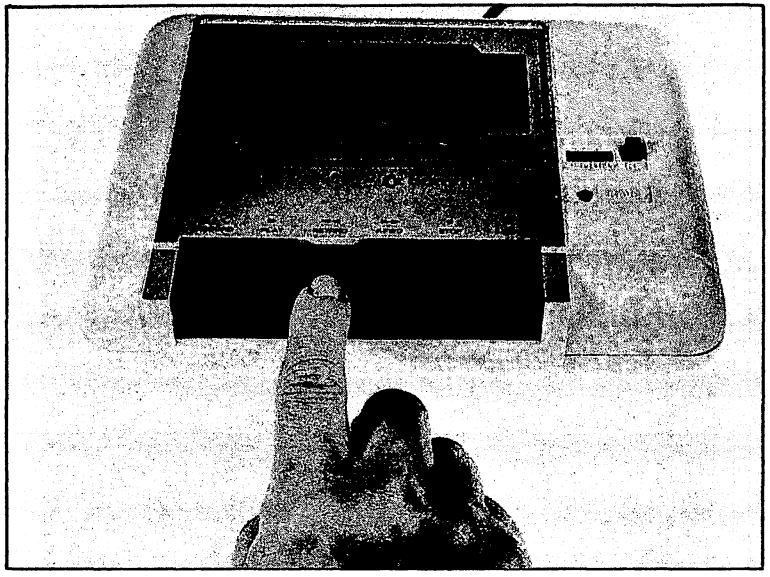


208. If, however, there is an error on the tape, the C64 will give you the awful message, VERIFYING, ?VERIFY ERROR, READY, and blink its cursor at you. If it does, don't panic. Just rewind the tape, go back to illustration 192, and start all over again.

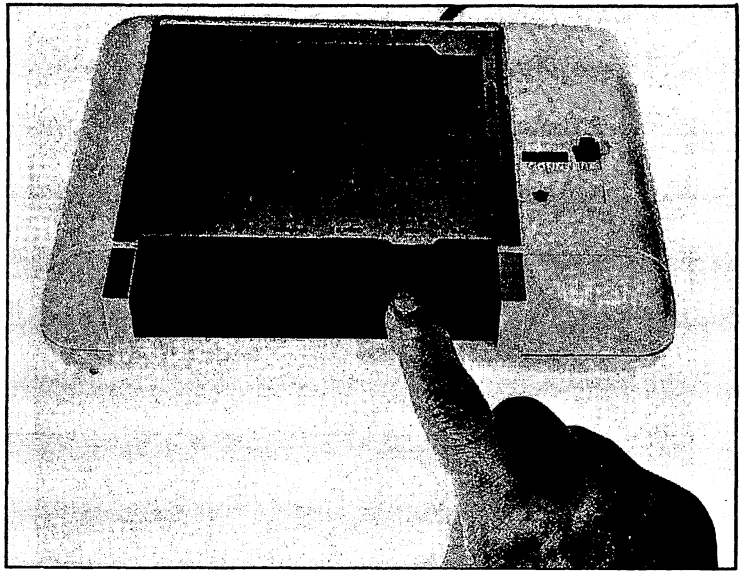
NOTE: The length of time it takes to SAVE, VERIFY, or LOAD a tape-recorded program depends on the length of the program. Short ones like FORTUNE TELLER take only a few seconds. Longer ones may take ten minutes or more.

Loading Your Saved Programs

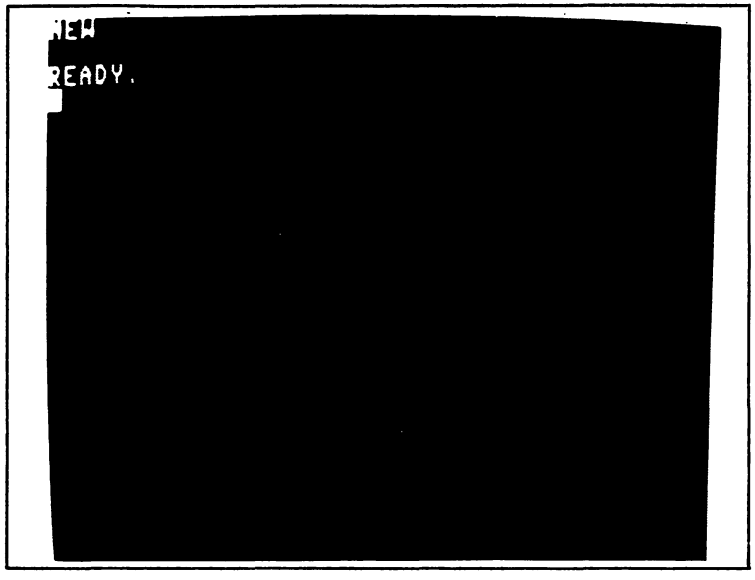
209. Once you have successfully verified that you have saved your program on tape, you can begin to experiment with loading it back into memory. The first step is to press the STOP lever on the Datassette.



210. Then press the REWIND lever to rewind the tape.

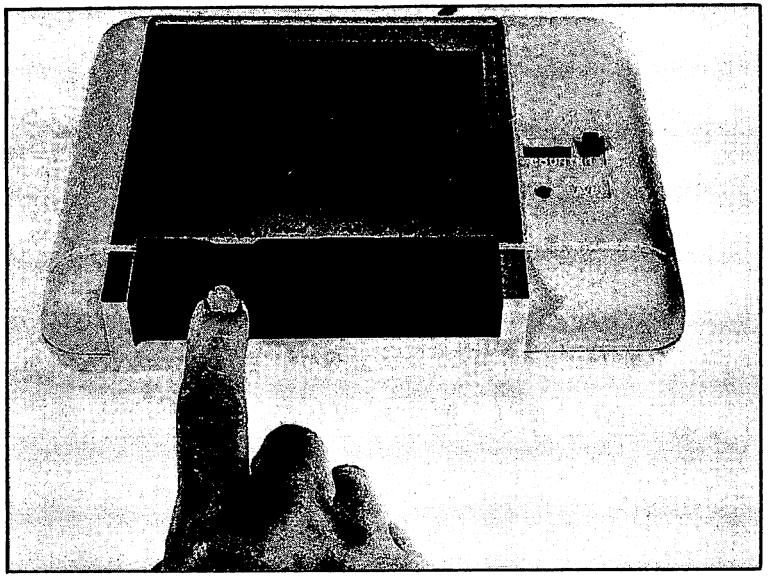


211. When it is rewound, press STOP.

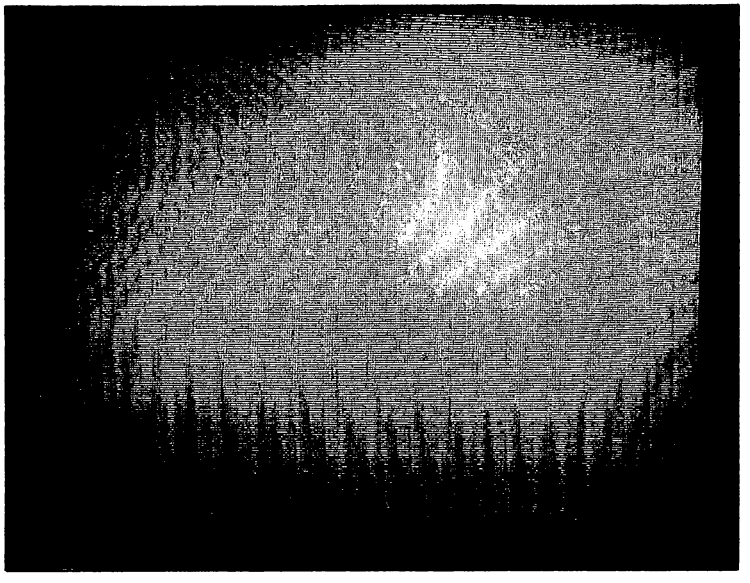


212. Now clear the screen, and type in: NEW (press RETURN) to clear memory.

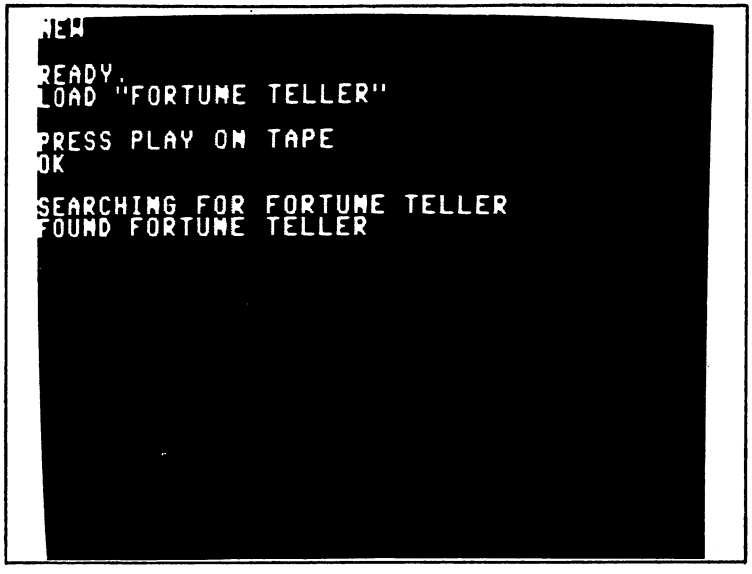
213. Type: LOAD "FORTUNE TELLER" (press RETURN), and the computer will tell you to PRESS PLAY ON TAPE.



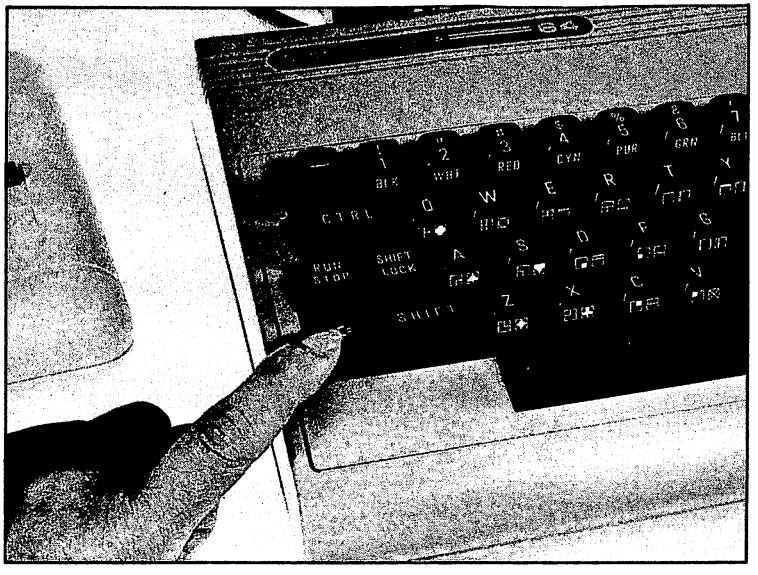
214. Do just what the nice machine asks. Press the PLAY lever.



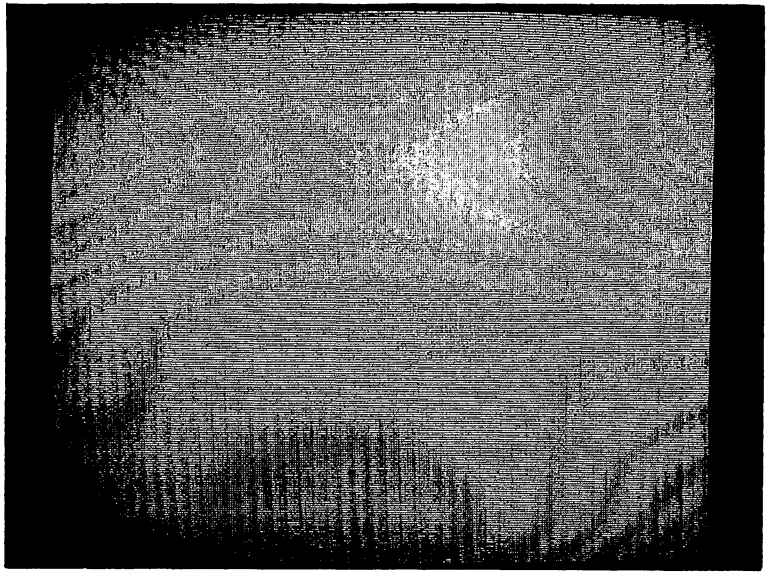
215. As soon as you do, the screen will go blank and you will see the tape hubs begin to turn.



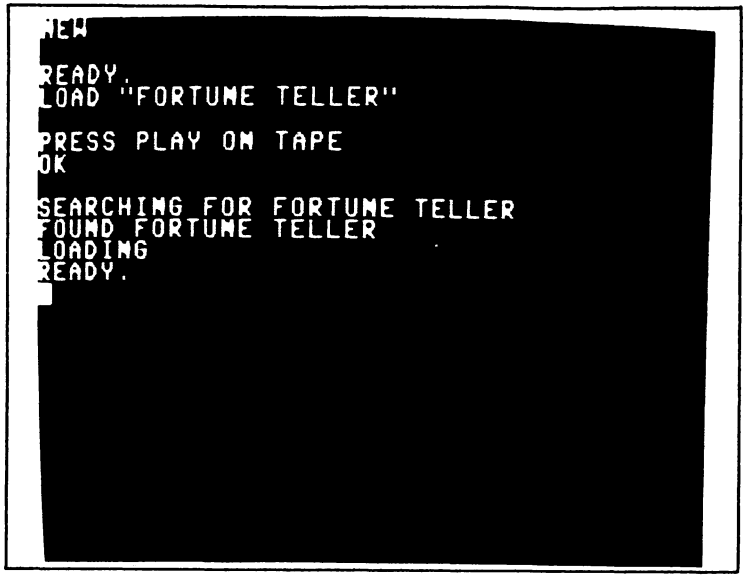
216. Then, when the tape has advanced to the correct position, the computer will let you know that it is SEARCHING FOR FORTUNE TELLER, and that it has FOUND FORTUNE TELLER.



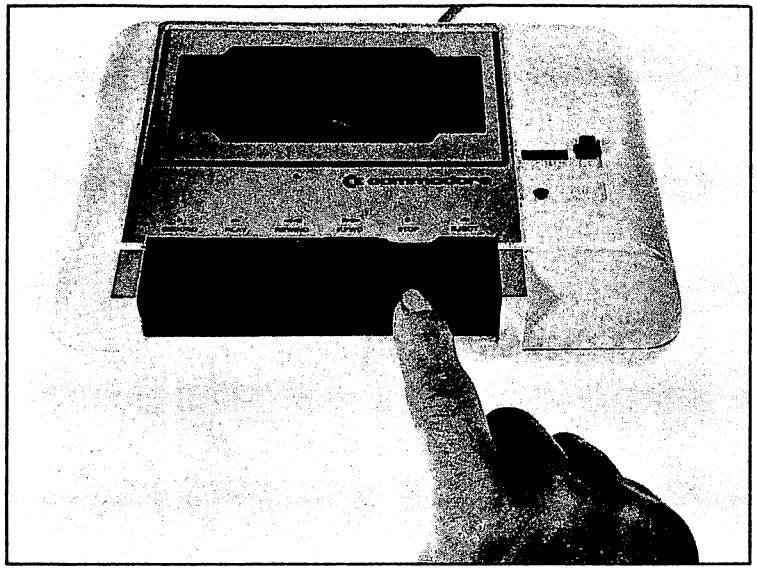
217. Press the COMMODORE key to load the program into the computer's memory.



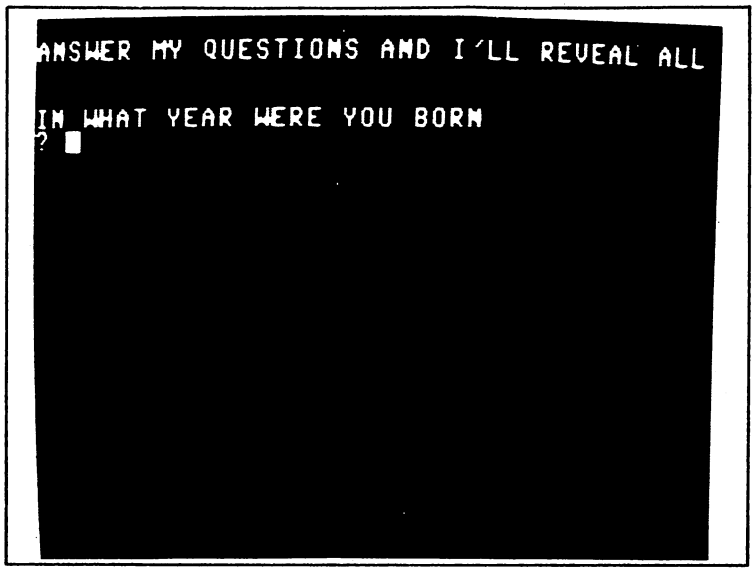
218. The screen will go blank.



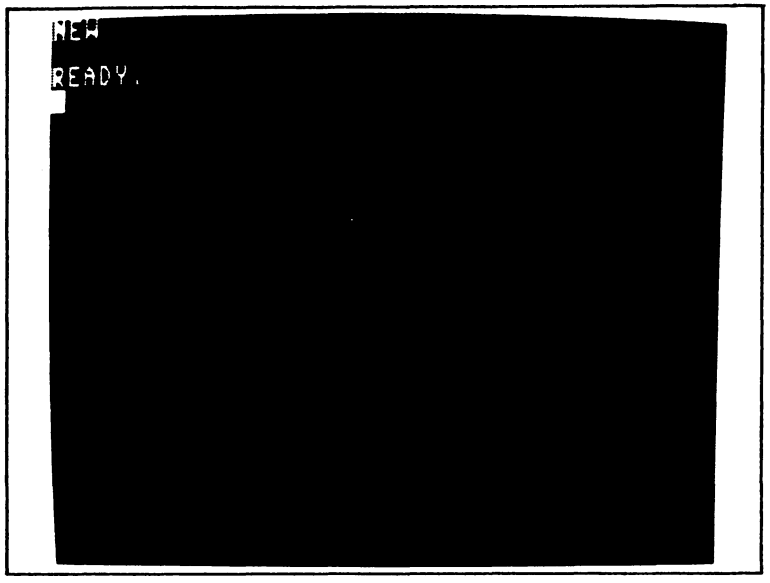
219. When the program has been successfully loaded, the screen will reappear with the message, LOADING, READY, and a blinking cursor.

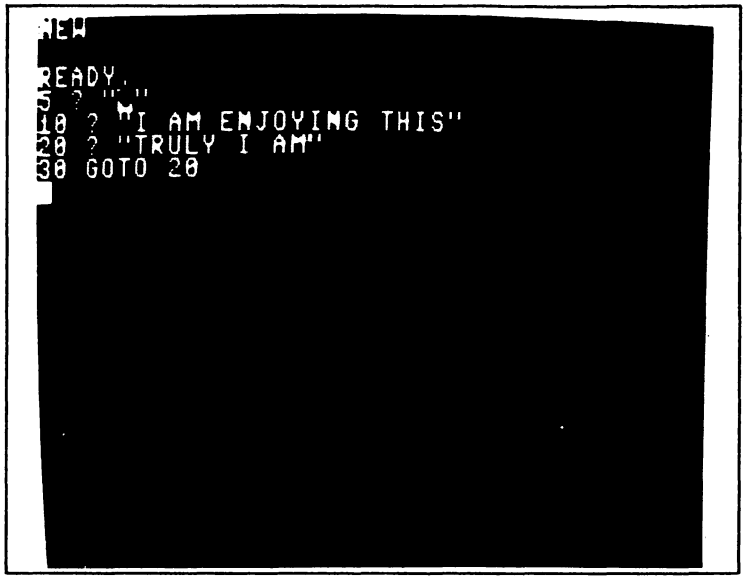


220. Press the STOP lever on the Datassette.



221. Just to be sure that the computer has done what it says it has, type in: RUN (and press RETURN). Your screen should look like this. Go through the program one more time to convince yourself that it is the program you saved.



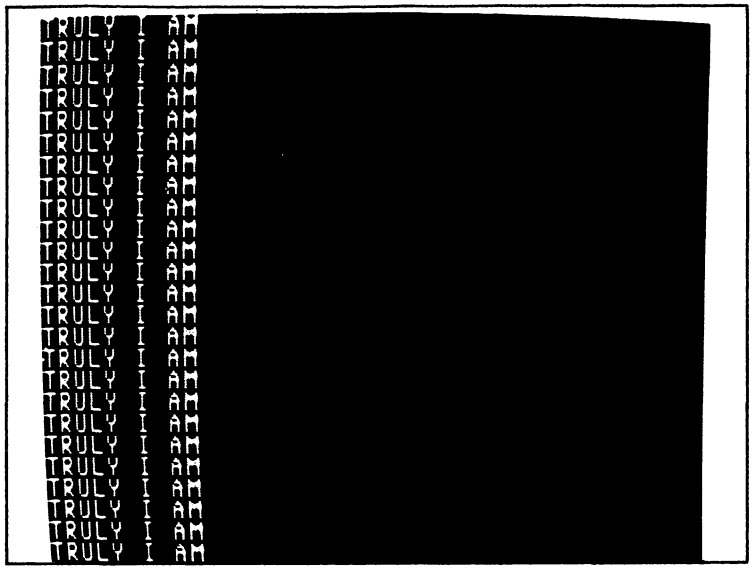


SOME USEFUL PROGRAM EXAMPLES

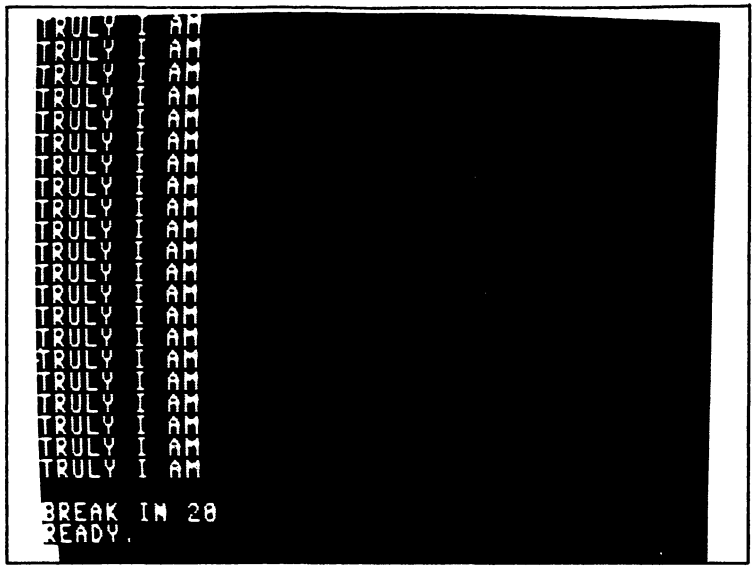
Back to BASIC Basics

222. Now it's vocabulary-building time. BASIC is not the richest language in the world, but it is the one the C64 understands, and it's time to learn more about it. Let's start off by pressing SHIFT and CLR/HOME to clear the screen. Then type in: NEW (and press RETURN) to clear THE FORTUNE TELLER out of memory.

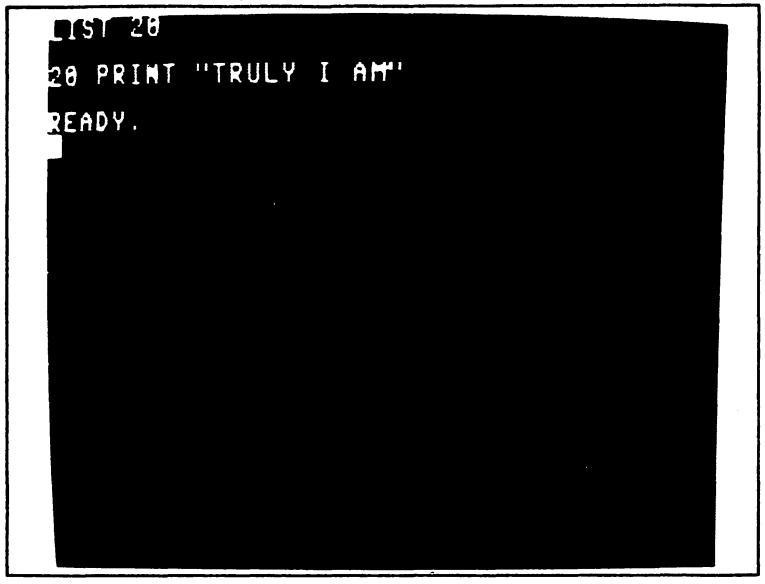
223. Type in: 5 ? "♥" (press RETURN), 10 ? "I AM ENJOYING THIS" (press RETURN), 20 ? "TRULY I AM" (press RETURN), 30 GOTO 20 (press RETURN).



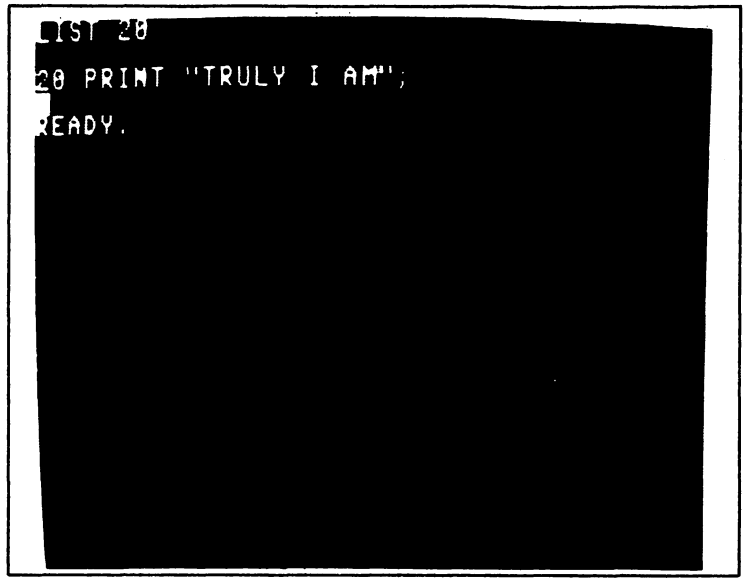
224. RUN the program, and your screen will look like this. The line on the bottom of the screen will keep flickering as the program keeps executing line 30, only to receive the command to GOTO line 20 again, where it is commanded to PRINT TRULY I AM.



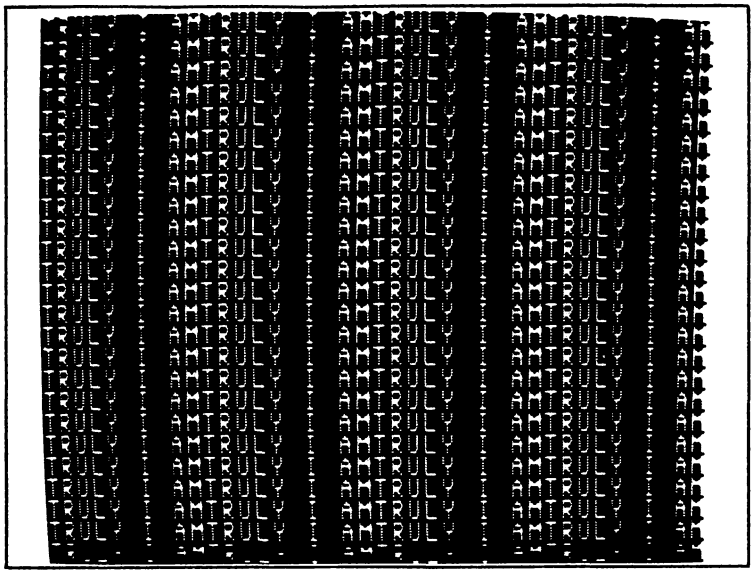
225. The way to get the program out of the loop it's in is to press the RUN/STOP key. That will get you the message, BREAK IN 20 and leave the computer ready for more work.



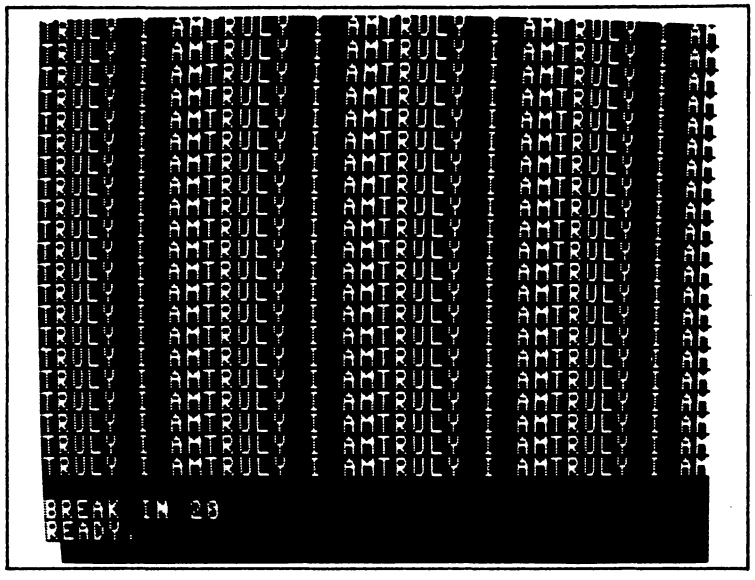
226. Now clear the screen, type: LIST 20 (and press RETURN).



227. Just to illustrate what a little syntax can do, move the cursor to the end of line 20, type in a semicolon (;), and press RETURN.

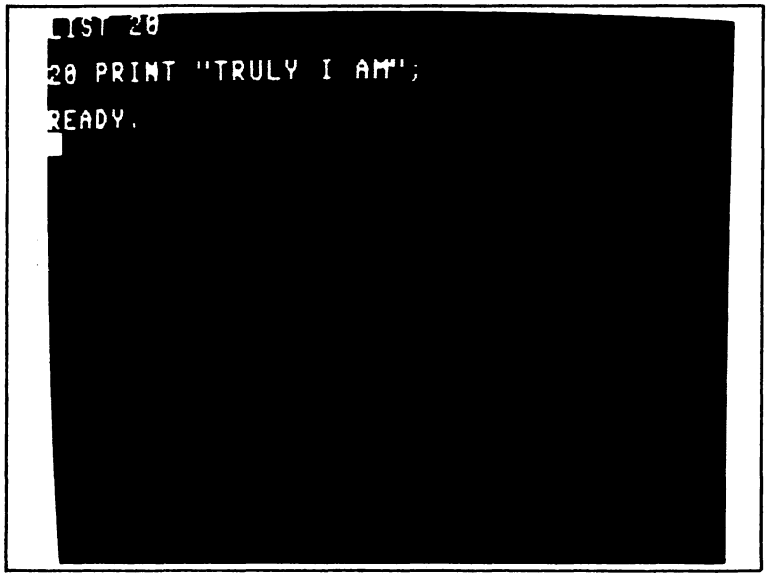


228. RUN the program. That's quite a change, isn't it? The program is in the same loop—line 20, line 30, line 20, and so on—but this time, because there is a semicolon at the end of the PRINT command on line 20, the next PRINT statement gets PRINTed where the semicolon appears. (Remember how the semicolon was used in FORTUNE TELLER.)

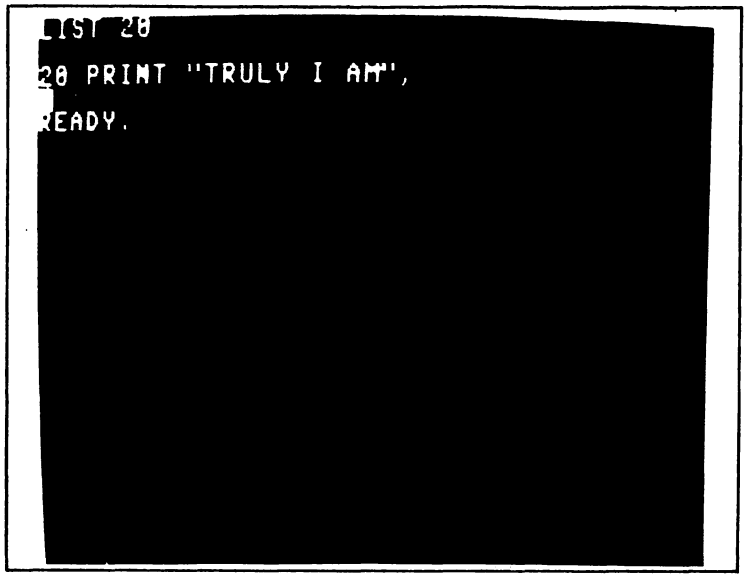


229. Break the program loop by pressing the RUN/STOP key. Notice that each time line 20's PRINT command is executed, the letter T in TRULY is placed right after the last M in AM.

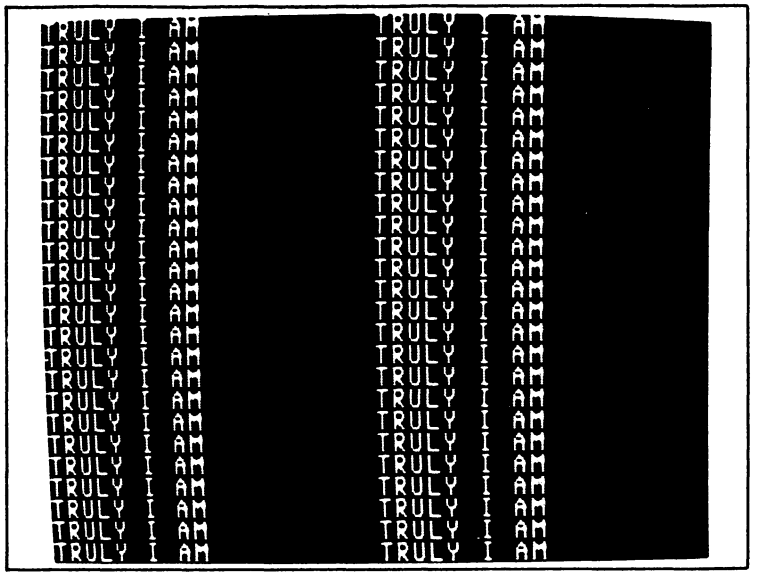
NOTE: If you would like to see these peculiar lines continue to keep scrolling up and off the screen, you need only type in the command: CONT (and press RETURN). The program will CONTinue right where it left off and RUN until you press the RUN/STOP key again.



230. Clear the screen. Type in: LIST 20 (and press RETURN).

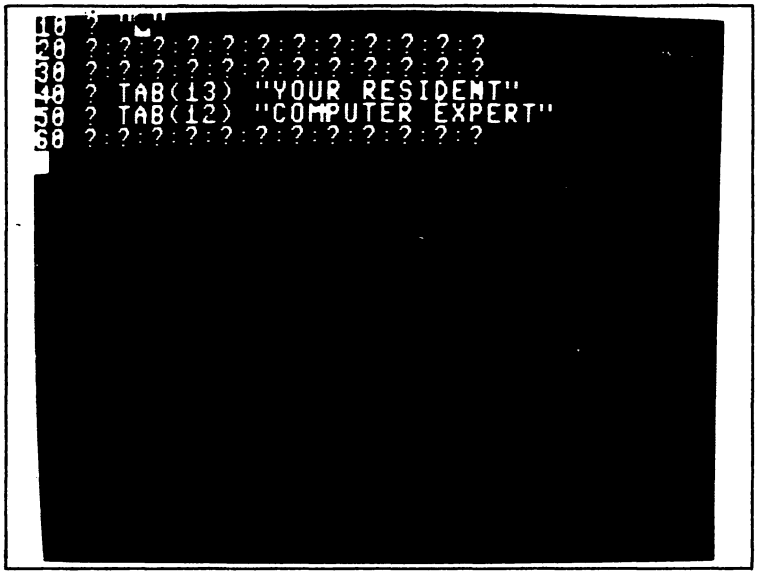


231. Position the cursor over the semicolon at the end of program line 20, type a comma (,), and press RETURN.



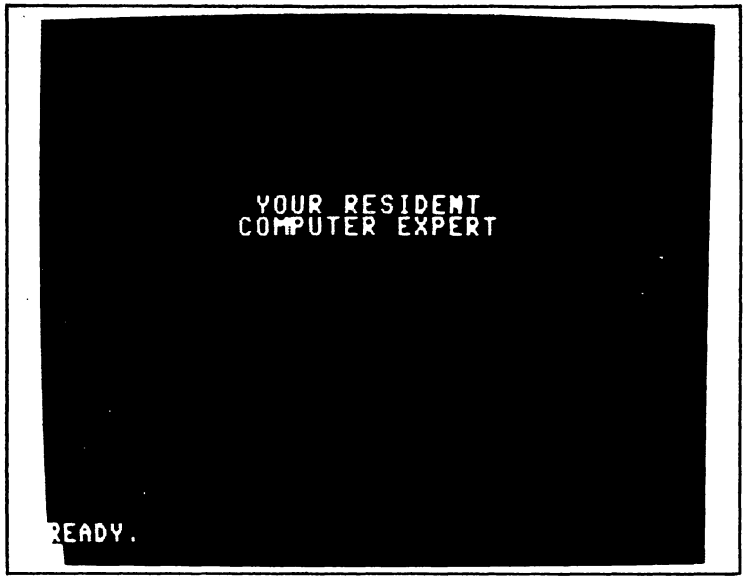
232. RUN the program. Now line 20 is being PRINTed in two neat columns.

NOTE: Ending a PRINT statement consisting of more than 10 and less than 20 characters with a comma results in a double-column presentation. If the PRINT statement has more than 1 and less than 10 characters, four neat columns will result.



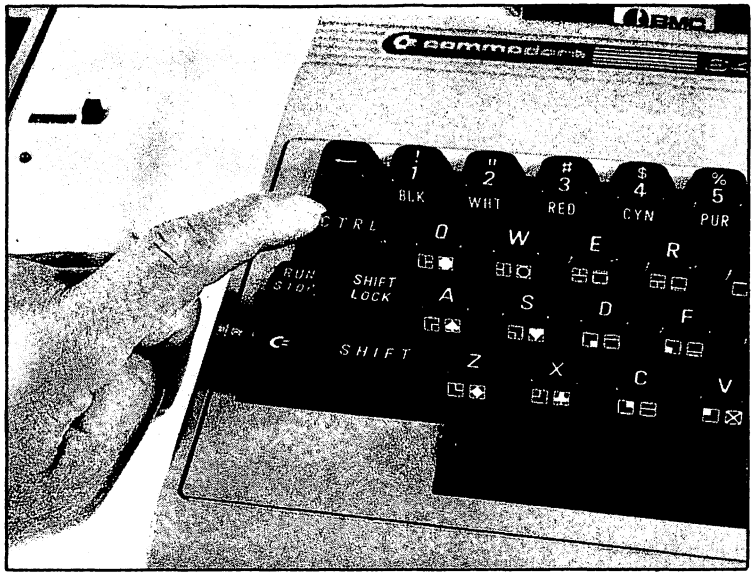
Loops, Scrolling, and Subroutines

233. Let's begin this formidable sounding section the way that Cecil B. DeMille might have—with an impressive set of titles rolling up onto the screen. If the program in illustration 232 is still running, break it with the RUN/STOP key, then clear both memory (NEW) and the screen, and type in the following program lines: 10 ? "♥" (press RETURN), 20 ?:?:?:?:?:?:?:?:?:?:?:? (press RETURN), 30 ?:?:?:?:?:?:?:?:?:?:?:? (press RETURN), 40 ? TAB(13) "YOUR RESIDENT" (press RETURN), 50 ? TAB(12) "COMPUTER EXPERT" (press RETURN), 60 ?:?:?:?:?:?:?:?:?:?:?:? (press RETURN).



SUGGESTION: You can save yourself a lot of work when typing the program called for in this illustration by using line 20 to give you lines 30 and 60. All you have to do is type in line 20 (it contains 12 question marks, by the way) and press RETURN. Then move the cursor up to the 2 in line 20, type a 3, and press RETURN. That establishes line 30 in memory and leaves line 20 intact in memory. Now move the cursor to the 3 you just typed, type a 6, and press RETURN. That will establish line 60 in memory, leaving lines 20 and 30 intact. If all that just confuses you, simply type each line in as shown.

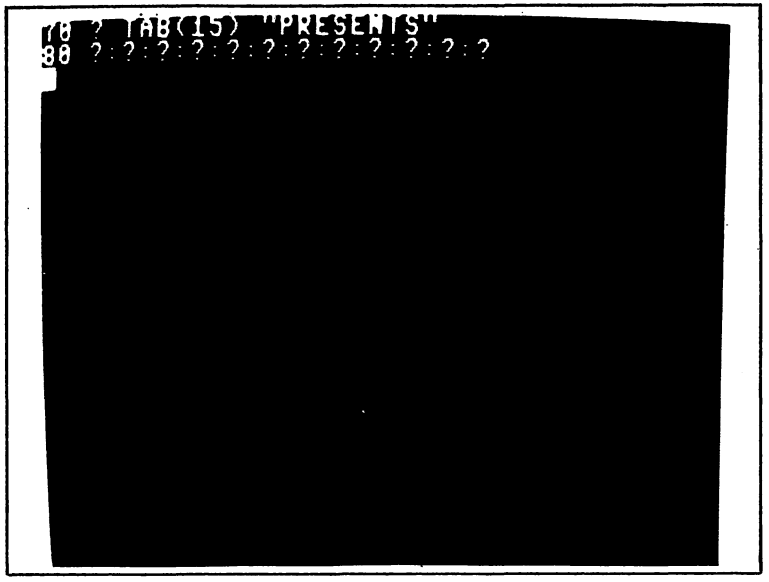
234. RUN the program a few times to see what it does. Notice that lines 40 and 50 are first printed at the bottom of the screen, then they move up (scroll) to the center of the screen.



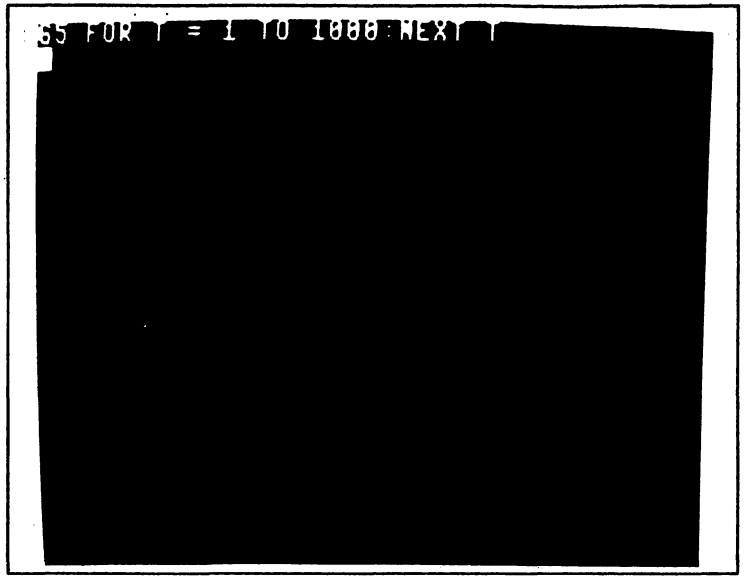

```
LIST
10 PRINT "{CLR}"
20 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:P
RINT:PRINT:PRINT:PRINT:PRINT:PRINT
30 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:P
RINT:PRINT:PRINT:PRINT:PRINT:PRINT
40 PRINT TAB(13) "YOUR RESIDENT"
50 PRINT TAB(12) "COMPUTER EXPERT"
60 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:P
RINT:PRINT:PRINT:PRINT:PRINT:PRINT
READY.
```

235. It may help to see what's happening if you press the CTRL key immediately after you press RETURN to get the program to RUN. Holding the ConTRoL key down causes the program to RUN slowly, one command at a time.

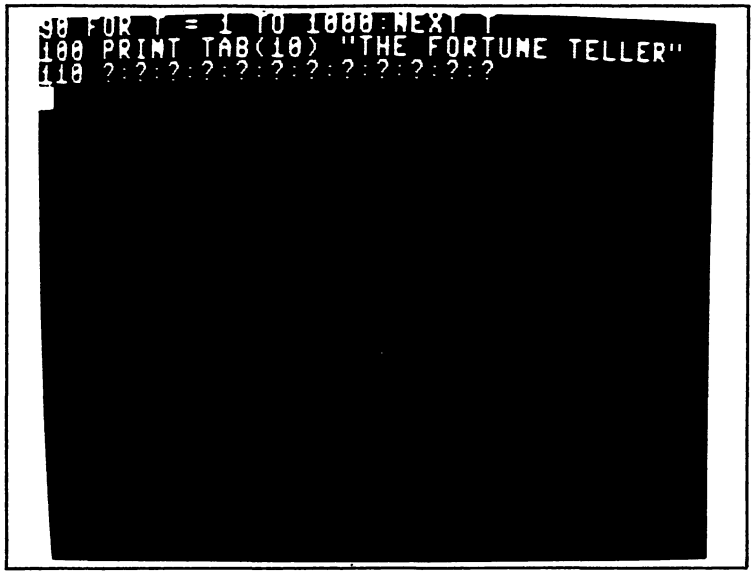
236. Clear the screen and LIST the program. A bit of explanation is in order here. Line 10 simply clears the screen and leaves the cursor at row zero on the screen (the rows, from top to bottom, are numbered from 0 to 24). Line 20 moves the cursor down twelve rows to row 12. Line 30 moves the cursor from row 12 to row 24 near the bottom of the screen. Line 40 commands the C64 to begin PRINTing, on row 24, at column 13, by means of the TAB command. The columns, from left to right, are numbered from 0 to 39. The number within the TAB command's parentheses tells the computer at which column to begin PRINTing. Line 50 commands the machine to begin PRINTing at column 12 on row 24, automatically moving the line just printed by program line 40 up one row in the process. This computer offers only horizontal TABbing. There is no command word provided for vertical TABbing, which is why lines 20 and 30 were necessary. Line 60, with twelve PRINT commands, moves the two printed lines up twelve rows to the center of the screen.

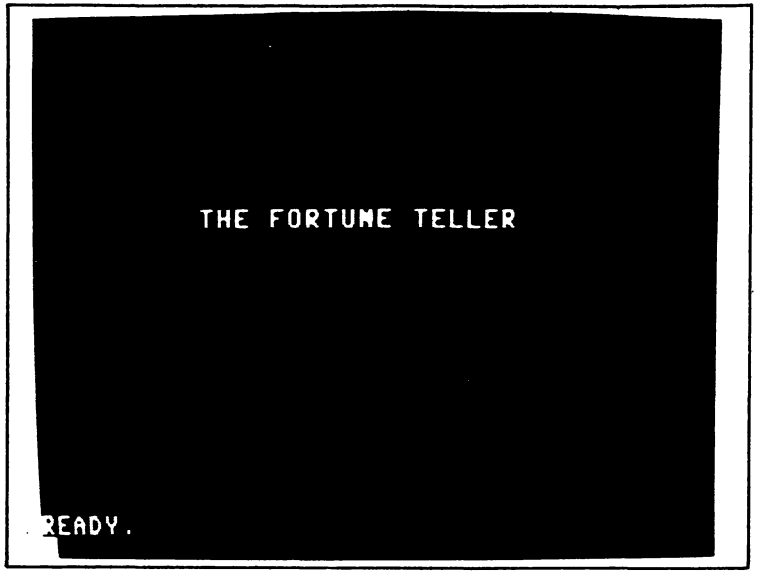


237. Clear the screen to get rid of all the confusing clutter, then add the following program lines: 70 ? TAB(15) "PRESENTS" (press RETURN), 80 ?:?:?:?:?:?:?:?:?:?:?:? (press RETURN).



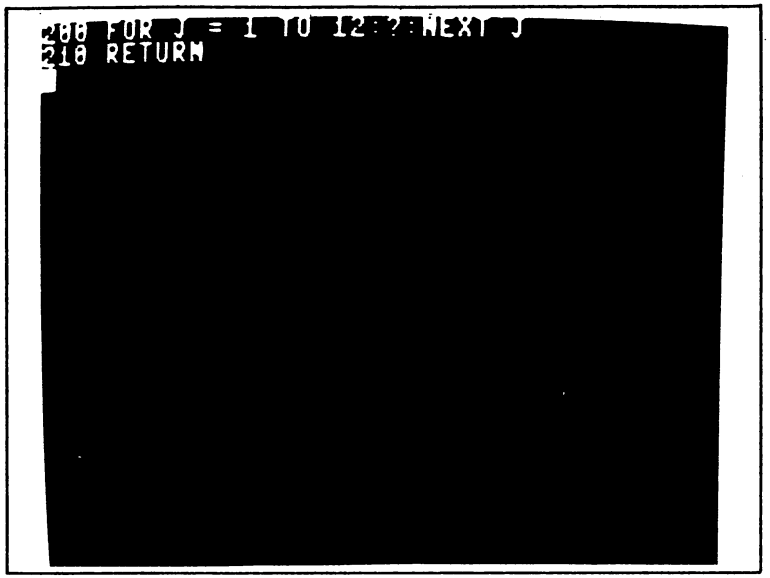
238. Now when you RUN the program, the first two lines sort of whiz by unless you use the CTRL key to slow things down. However, it isn't necessary to use the CTRL key. We can do something similar with a single program line containing two commands. The line we will use is called a delay loop. Clear the screen, then type in: 65 FOR T = 1 TO 1000:NEXT T (and press RETURN).

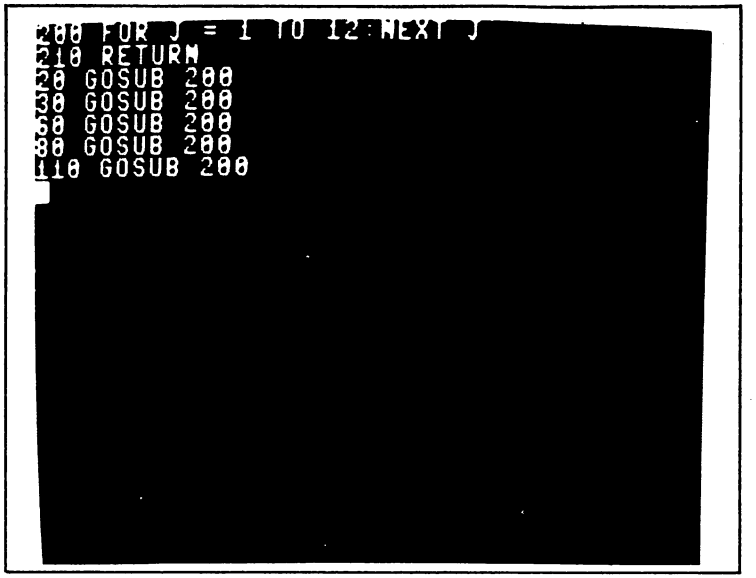




239. RUN the program. That's a lot better, isn't it? You can lengthen the time that the initial two lines remain at center screen by increasing the value following the word TO (raising 1000 to 1500, for example), or conversely, you can shorten the lines' time at center screen by decreasing the TO value (lowering the 1000 to 500, for example). After you try that out to your satisfaction, clear the screen. Now type in: 90 FOR T=1 TO 1000:NEXT T (press RETURN), 100 PRINT TAB(10) "THE FORTUNE TELLER" (press RETURN), 110 ?:?:?:?:?:?:?:?:?:?:?:? (press RETURN).

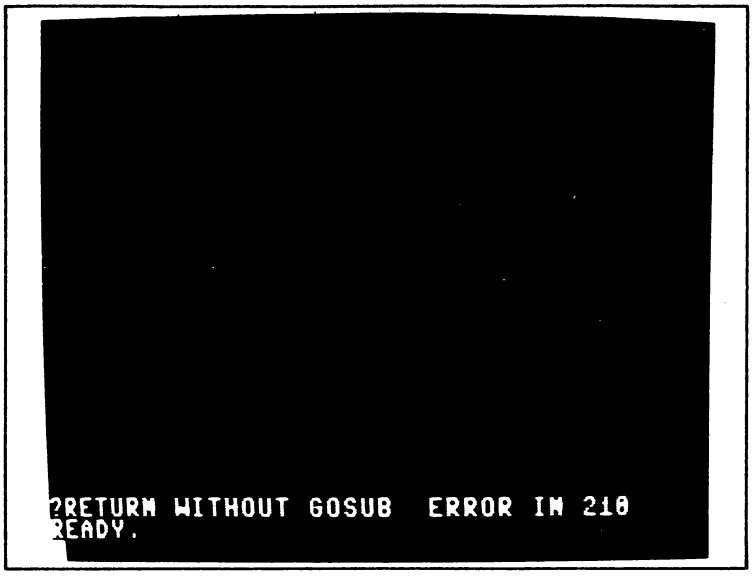
240. Now RUN the program and you will see lines 40 and 50 appear at the bottom of the screen, scroll to mid-screen, hold briefly, then scroll up and off the screen, being replaced by line 70, which starts at the bottom of the screen and scrolls up to mid-screen. Line 70 then scrolls up and off, being replaced by line 100, which appears at the bottom of the screen and scrolls to mid-screen. As soon as all this action is over, the computer runs out of things to do and announces that it is ready for more.

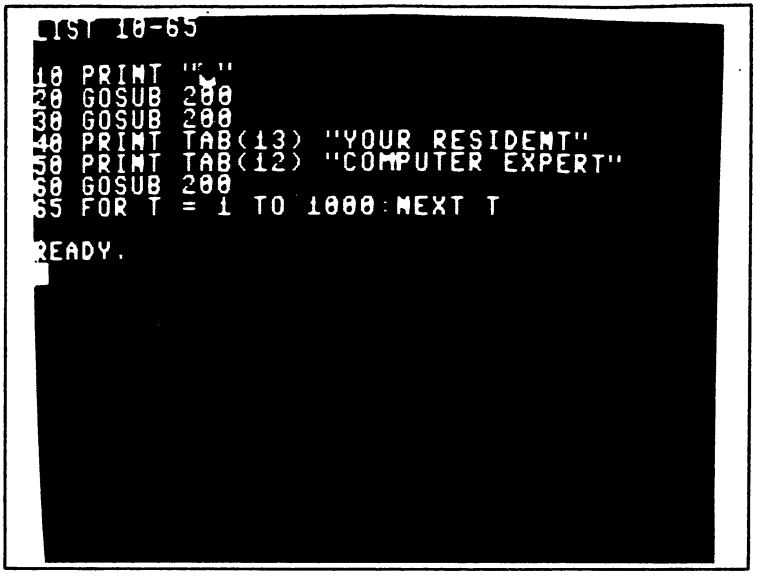




241. I got the scrolling titles I was after, but frankly, the program I used to get them is a mess. We can simplify things by getting rid of all those ridiculous lines containing 12 consecutive PRINT commands. Before we do, however, let's set up a subroutine which will do exactly what each one of those lines did. Clear the screen, then type in: 200 FOR J = 1 TO 12:?:NEXT J (press RETURN), 210 RETURN (press RETURN). Program line 200 contains three commands, separated by colons, which perform a loop. The single line causes the PRINT command to be executed 12 times. The C64 keeps track by counting how many trips around the loop have been completed, and when they equal 12, it goes on to execute the next program line.

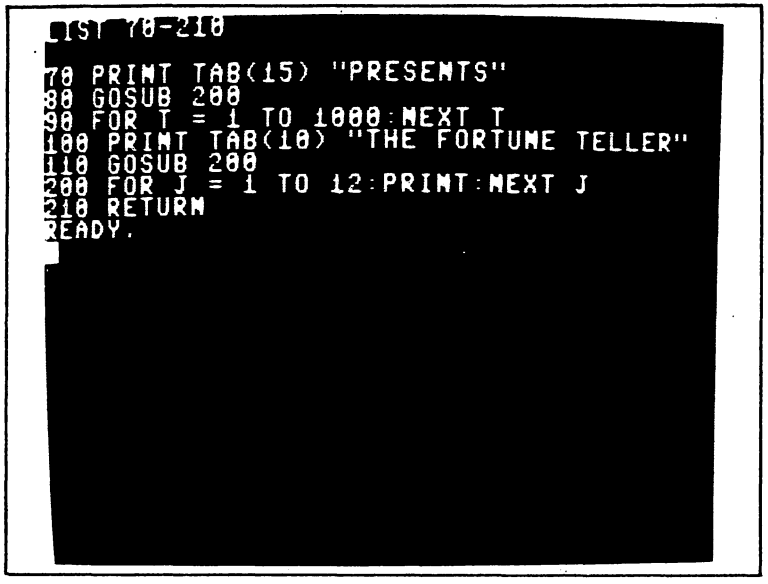
242. Now we can get rid of those old multiple-print lines (20, 30, 60, 80, and 110) by typing in the following program lines which will automatically replace them in the computer's memory: 20 GOSUB 200 (press RETURN), 30 GOSUB 200 (press RETURN), 60 GOSUB 200 (press RETURN), 80 GOSUB 200 (press RETURN), 110 GOSUB 200 (press RETURN).



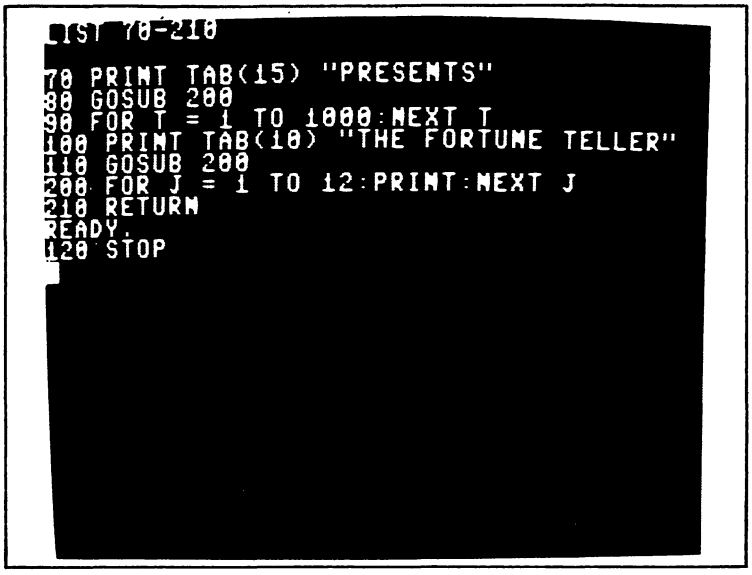


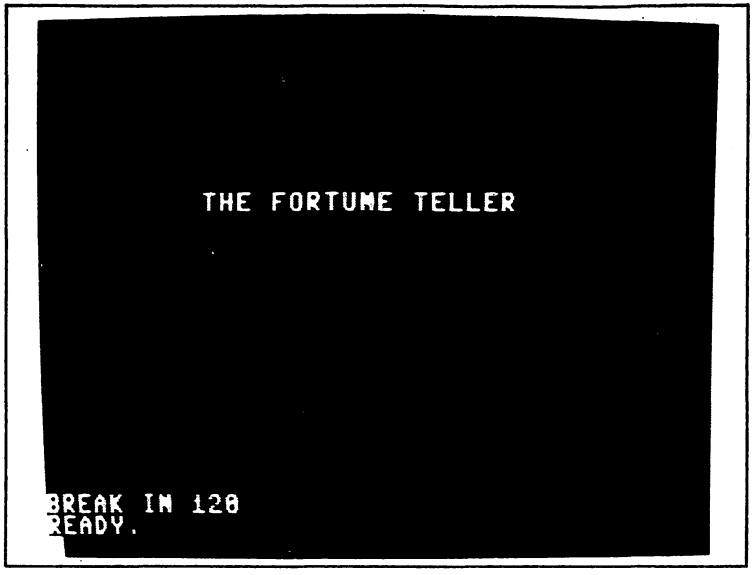
243. RUN the cleaned up program Whoops! Everything went smoothly until line 100 simply scrolled up off the screen and disappeared. Then the computer automatically gave you the error message shown on the screen, to explain what went wrong.

244. Clear the screen, then LIST program lines 10 through 65 and let's see what went wrong. Line 10 just cleared the screen. Line 20 told the computer to GO to the SUBroutine at line 200. Line 200 moved the cursor down 12 rows. Line 210 told the computer to RETURN to the line following the GOSUB command that got it to line 200. The C64 does what it is told to do, so it RETURNed to line 30, where it was told to GO to the SUBroutine at line 200 again. The subroutine at line 200 moved the cursor down to the bottom of the screen. Then line 210 RETURNed the program to line 40 which PRINTed at the thirteenth position on the bottom row of the screen. Next line 50 PRINTed on the bottom row, moving line 40 up one row. Then line 60 brought us back to the subroutine at line 200, which moved the two PRINTed lines up 12 rows to mid-screen. Line 210 RETURNed us to line 65, which did nothing more than cause the computer to count on its fingers to 1000. During the period of time the computer was counting, the two PRINTed lines remained at mid-screen.



245. Clear the screen, then LIST lines 70 through 210. After the C64 counted to 1000, the program moved along to line 70, and its PRINT command was carried out, at the cursor's position on the bottom row of the screen. Then the computer moved on to line 80, which took it back to the subroutine at line 200 for another 12 PRINT commands. Those moved the first two PRINTed lines up and off the top of the screen and caused the PRINTed material on the bottom row to move up 12 rows to mid-screen. The computer then RETURNed to line 90 and had to count to 1000 again. That kept the word PRESENTS at mid-screen for a while. Next the program moved along to line 100. Its PRINT command was executed at the cursor's position on the bottom row of the screen. Then line 110 sent the C64 back to the subroutine at line 200 again, which moved PRESENTS off the screen and moved THE FORTUNE TELLER up 12 lines to mid-screen. Line 210 sent the computer back up to the line following the one on which the GOSUB command appeared. Lo and behold! That line turns out to be line 200. So the computer expected 12 PRINT commands, sending THE FORTUNE TELLER up and right off the top of the screen. And when line 210 tried to find the GOSUB command that got the program to this subroutine, it was unable to. That's why the C64 gave us the error message and indicated that the error was in line 210. At last!

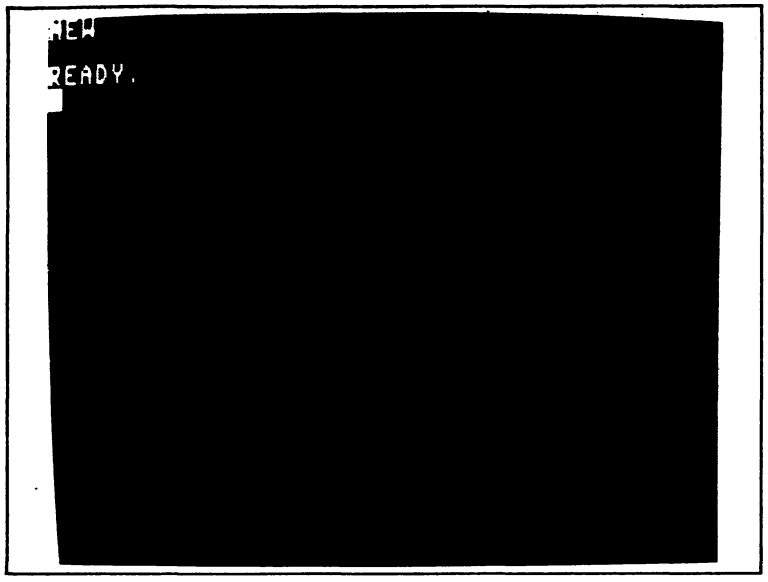


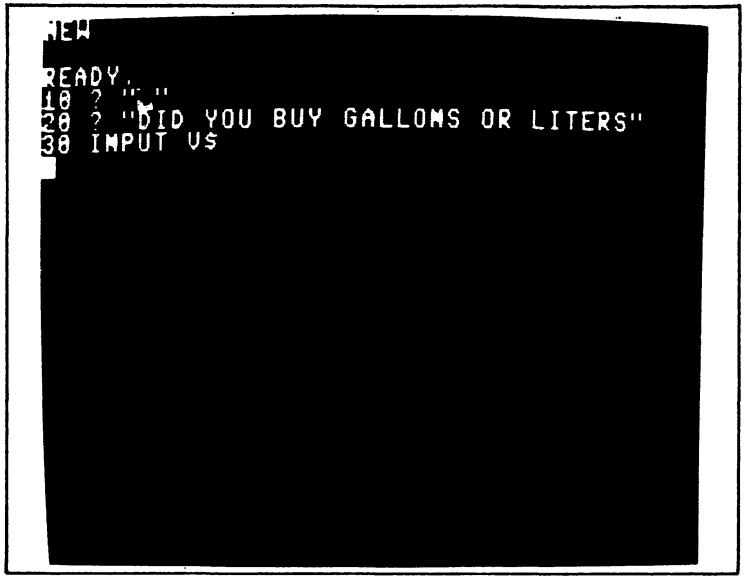


246. All we have to do to fix things is add one command; type in: 120 STOP (press RETURN).

247. RUN the program. That did it. It was a struggle, but we won!

SUGGESTED EXERCISE: Take a look at illustrations 182 and 183. They show the FORTUNE TELLER program you wrote earlier. Why don't you add it to what is already on the screen and then record the entire thing so that you will have both the scrolling introductions and THE FORTUNE TELLER as a single program. You can begin by using the same delay loop, as the one used in line 90, to replace the STOP command you just added as line 120. Then type in: 130 GOSUB 200 (press RETURN), 140 GOTO 250 (press RETURN). Now all you have to do is renumber all the lines in THE FORTUNE TELLER program to begin with 250, and increase numerically from there. When you are through, RUN the combined programs, debug them (get the errors out) as necessary, and SAVE the combined programs on tape as a single program.

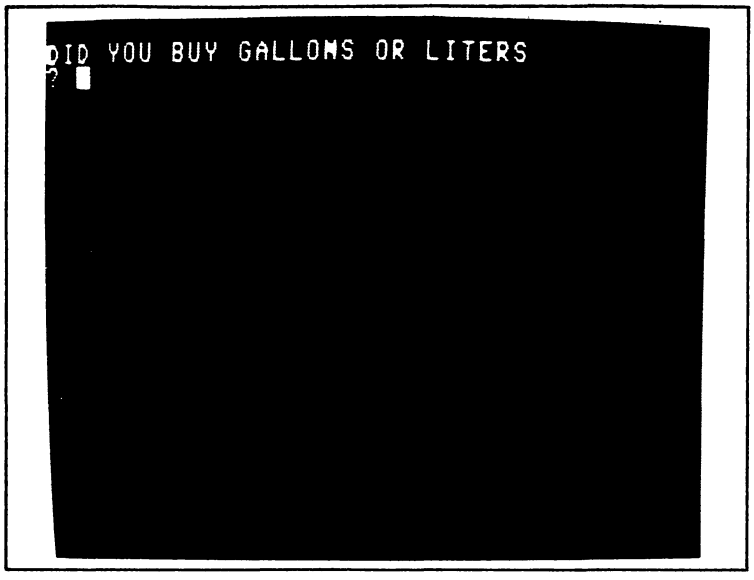




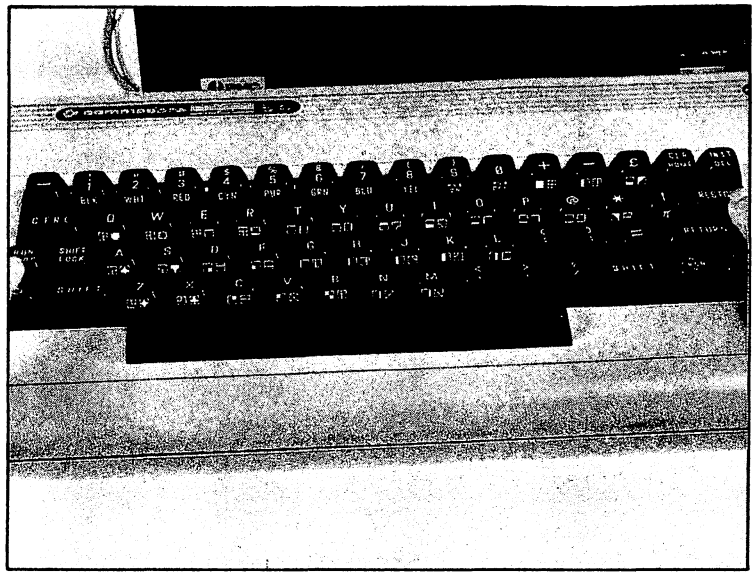
Prompts

248. Always try to make your programs as easy to use as possible. When you ask the user for information, tell him or her exactly how to answer. The most effective way to do this is with a prompt. An effective prompt is to underline a key word or key letter. Clear the screen, then type in: NEW (and press RETURN) to clear the computer's memory.

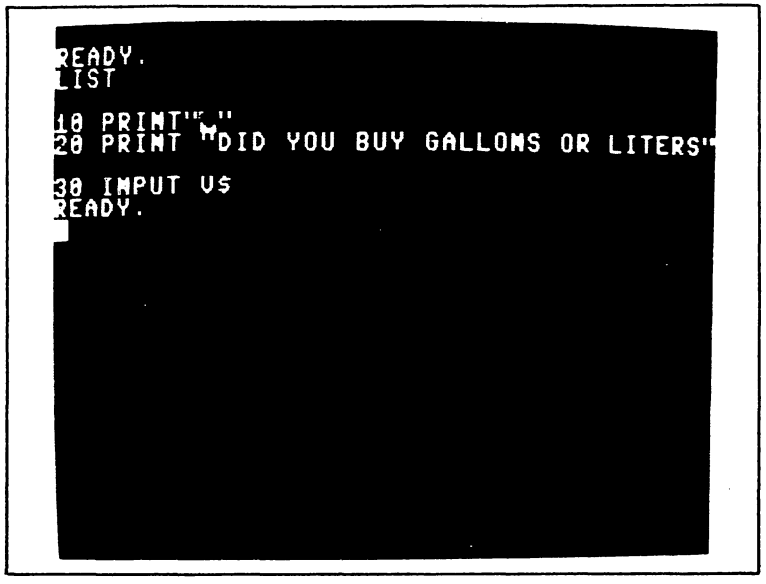
249. Type in the following program lines: 10 ? "♥" (press RETURN), 20 ? "DID YOU BUY GALLONS OR LITERS" (press RETURN), 30 INPUT V$ (press RETURN).



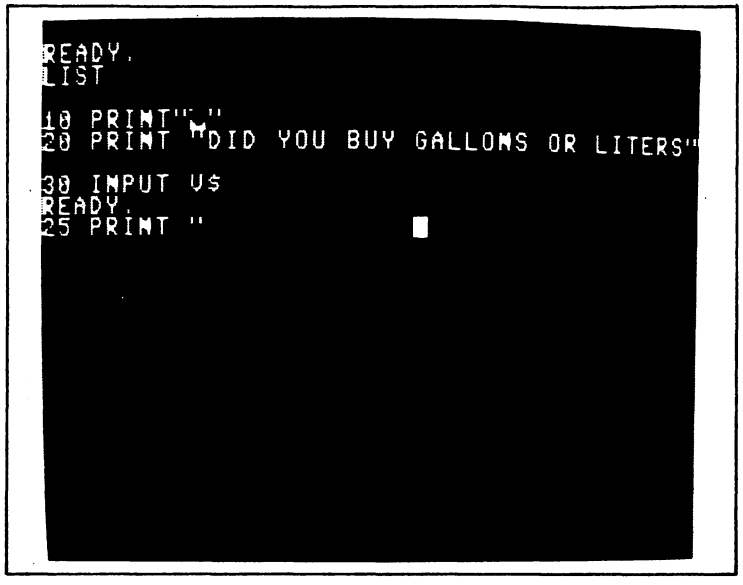
250. RUN the program and you are asked a simple question. To answer it you will have to type in either six or seven letters.



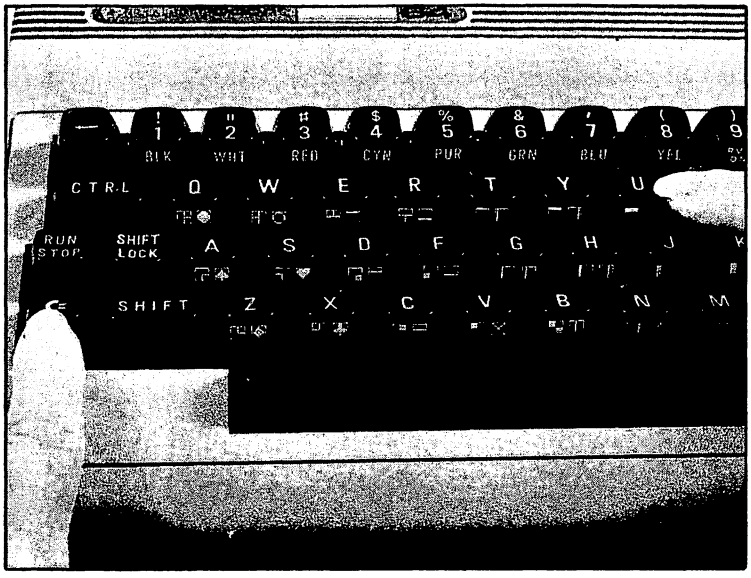
251. The same question can be answered just as effectively with a single key press. Press RUN/STOP and RESTORE to break out of the program (which is waiting for your INPUT).



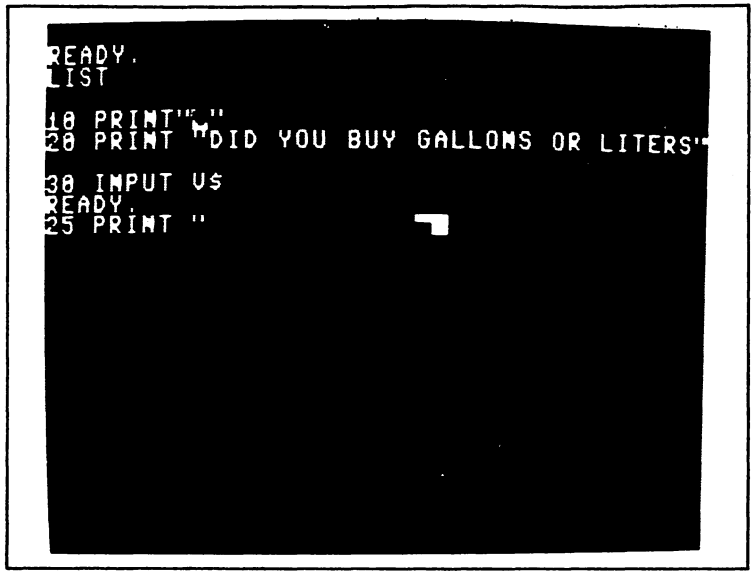
252. Type LIST (and press RETURN) to get your program listed on the screen.



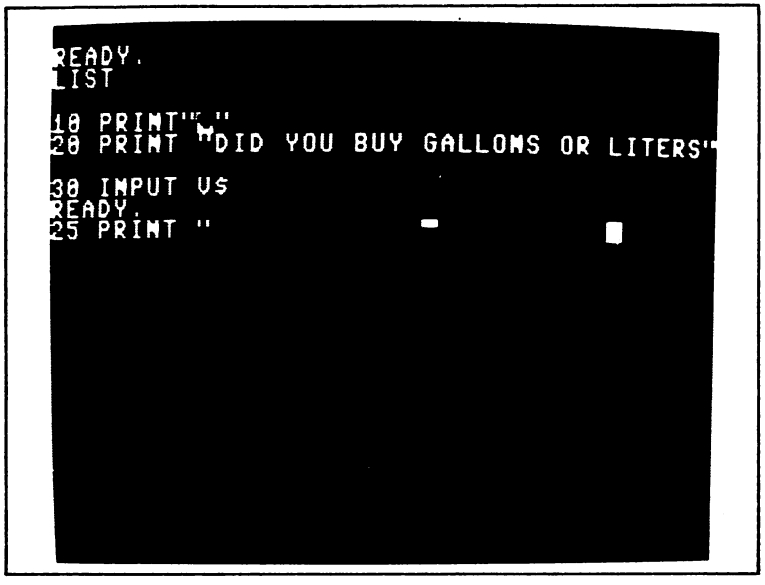
253. Type in: 25 PRINT " and use the SPACE bar to move the cursor under the letter G in the word GALLON in line 20.



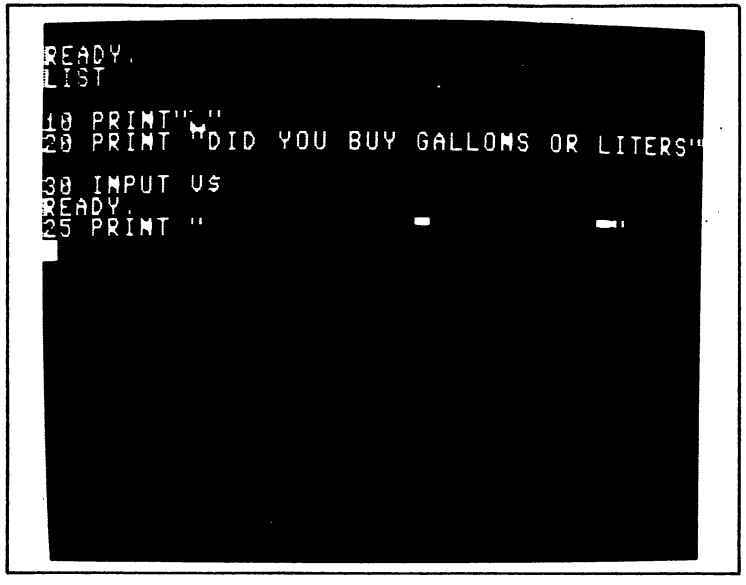
254. Press the COMMODORE and U keys simultaneously.



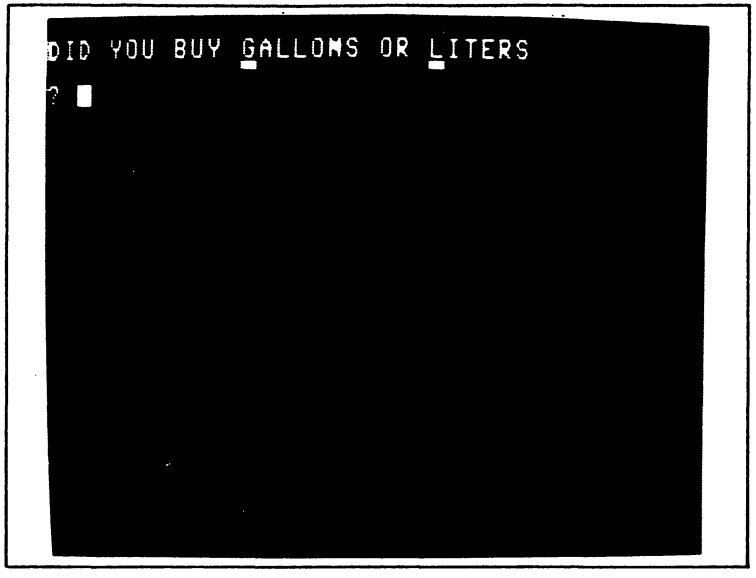
255. That puts a heavy underscore on the screen (see the graphic depiction on the left side of the front of the U key).

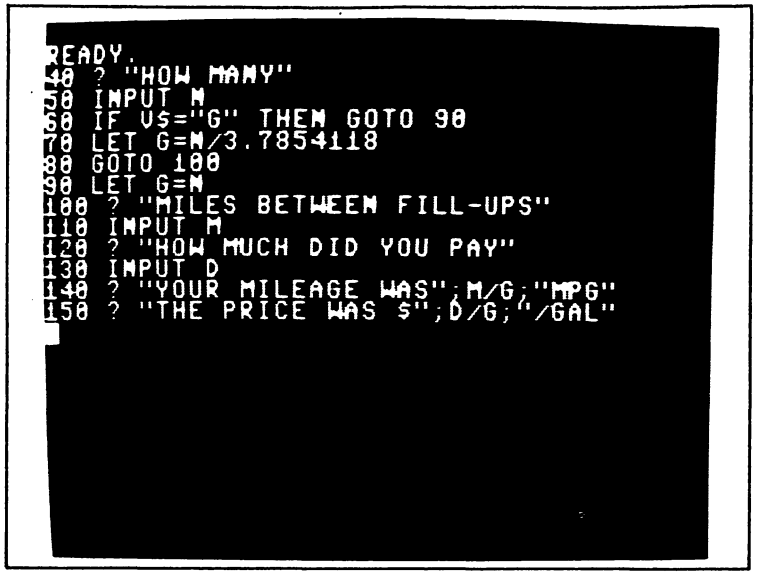


256. Use the SPACE bar to move the cursor under the letter L in the word LITERS in line 20.



257. Press the COMMODORE and U keys for another heavy underscore, then type the final quotation mark (") (and press RETURN) to enter line 25 into memory.

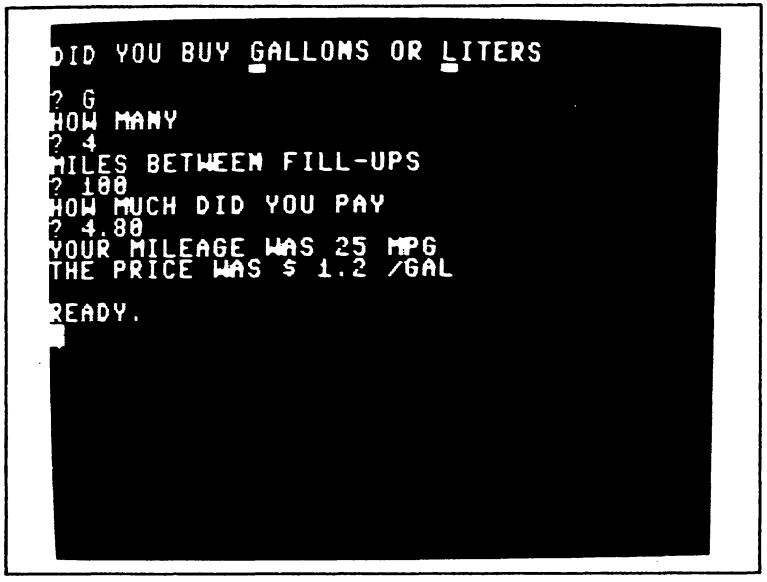




258. RUN the program. Wouldn't you be inclined to answer with a single press of the G or the L key now? Most people would be. If you wish, you can present entire short prompt words heavily underscored in this manner.

NOTE: Other methods of presenting prompt letters or prompt words include placing the letters or words between greater-than and less-than brackets, or presenting the letters and words in reverse video (see illustrations 269 through 277 to learn how this is done). Try these alternatives to determine your own preference.

259. Press the RUN/STOP and RESTORE keys together to break out of the program, then add the following program lines and press RETURN after each one: 40 ? "HOW MANY" 50 INPUT N 60 IF V$="G" THEN GOTO 90 70 LET G=N/3.7854118 80 GOTO 100 90 LET G=N 100 ? "MILES BETWEEN FILL-UPS" 110 INPUT M 120 ? "HOW MUCH DID YOU PAY" 130 INPUT D 140 ? "YOUR MILEAGE WAS";M/G;"MPG" 150 ? "THE PRICE WAS $";D/G;"/GAL"



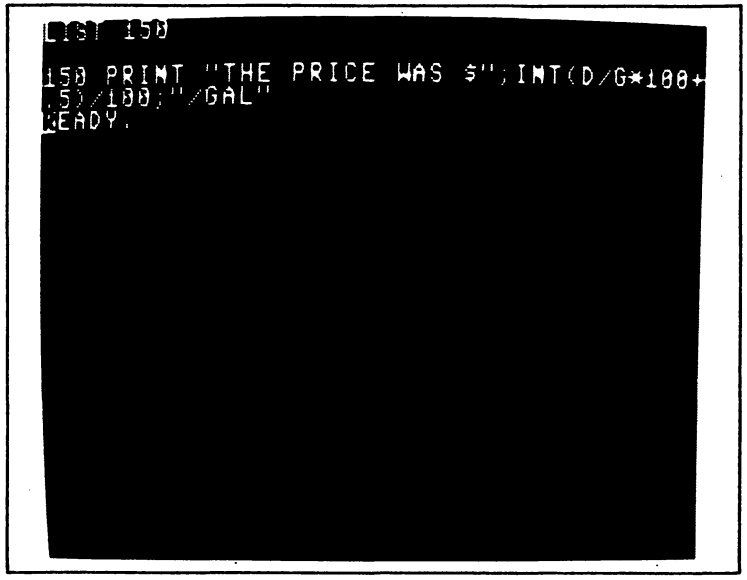
260. The program provides you with the miles per gallon your car is getting, and the cost per gallon of the gas or diesel fuel you are using. It is just as happy to work with metric liters as it is with plain old U.S. gallons. The only thing the program assumes of you is that you will truthfully tell it the number of miles between the time you last had the tank filled to the top and the most recent top-off. RUN the program. Type in the following answers to see how it works: G (press RETURN), 4 (press RETURN), 100 (press RETURN), 4.80 (press RETURN). The computer will immediately let you know that you are getting 25 miles per gallon and that your fuel cost you a dollar twenty per gallon. The C64 suppresses (eliminates) zeroes that aren't needed, so it tells you $1.2 instead of $1.20. However, I'm sure that won't confuse you.

```
LIST 60-100

60 IF V$="G" THEN GOTO 90
70 LET G=N/3.7854118
80 GOTO 100
90 LET G=N
100 PRINT "MILES BETWEEN FILL-UPS"

READY.
```

```
DID YOU BUY GALLONS OR LITERS
? L
HOW MANY
? 11
MILES BETWEEN FILL-UPS
? 121
HOW MUCH DID YOU PAY
? 4.67
YOUR MILEAGE WAS 41.6395298 MPG
THE PRICE WAS $ 1.60707937 /GAL

READY.
```

261. Clear the screen, then type LIST 60-100 and press RETURN to get the portion of the program we are interested in LISTed. It is very straightforward, but you may want to look at lines 60, 70, and 80 to see how the program recognizes if either gallons or liters are being entered. If it's gallons, line 60 simply directs the computer to skip the conversion process and go directly to line 90. If, however, you purchase your fuel in liters, you will have pressed L in answer to the first question, and V$ will then equal L. So when the computer gets to line 60 and finds that V$ does not equal G, it will proceed to convert the number of liters purchased (N) to gallons on line 70 and then, at line 80, be told to skip over line 90 and go right on to line 100 to complete the program.

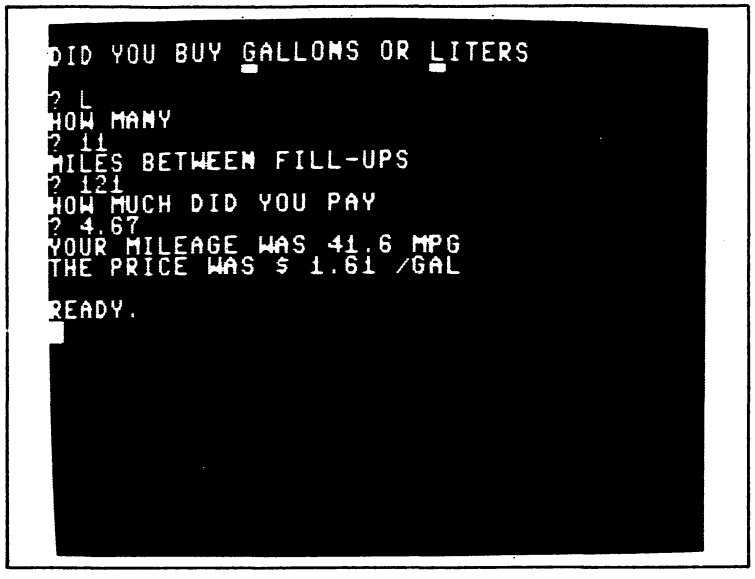
262. Now, just to illustrate how annoying decimal places can be, RUN the program again. This time give it the following answers: L, 11, 121, and 4.67 (entering each one in turn by pressing RETURN). Your screen will now look like this.



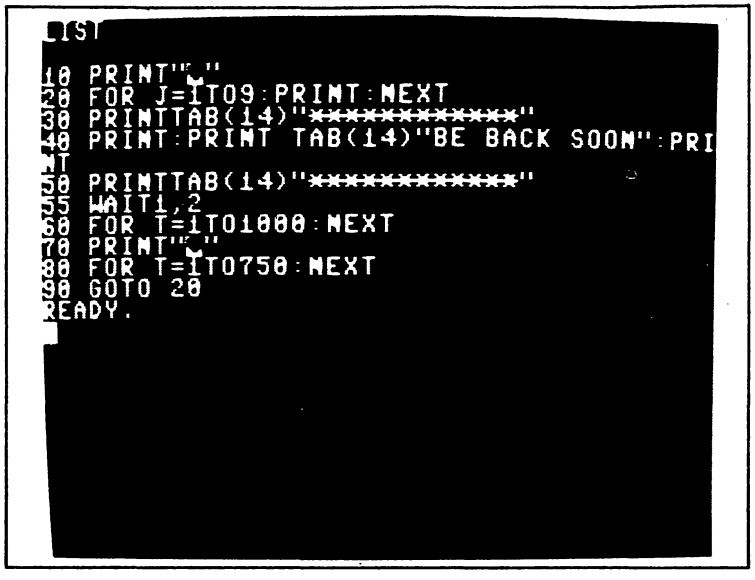
Rounding and Limiting Decimal Places

263. Numbers with lots of decimal places are fine when you need extreme accuracy, but they are really out of place in this little mileage program. It is quite simple, however, to suppress decimals. If you want to eliminate all decimal places and round a number N up if its decimal value equals or exceeds 0.5, or down if it is less than .5, use the following: N=INT(N+.5). If you want to limit a number N to one decimal place use N=INT (N*10+.5)/10. If you want to limit a number N to two decimal places use N=INT(N*100+.5)/100. To limit the miles-per-gallon figure to a single decimal place, clear the screen, LIST line 140, edit it to the line shown here, and press RETURN.

264. To limit the price-per-gallon figure to two decimal places (rounded to the nearest penny), clear the screen, LIST line 150, and edit it to the line shown here.



265. Clear the screen, then RUN the program and use the same values you did in illustration 262. Note that the answers provided are rounded to the nearest tenth of a mile and to the nearest penny.

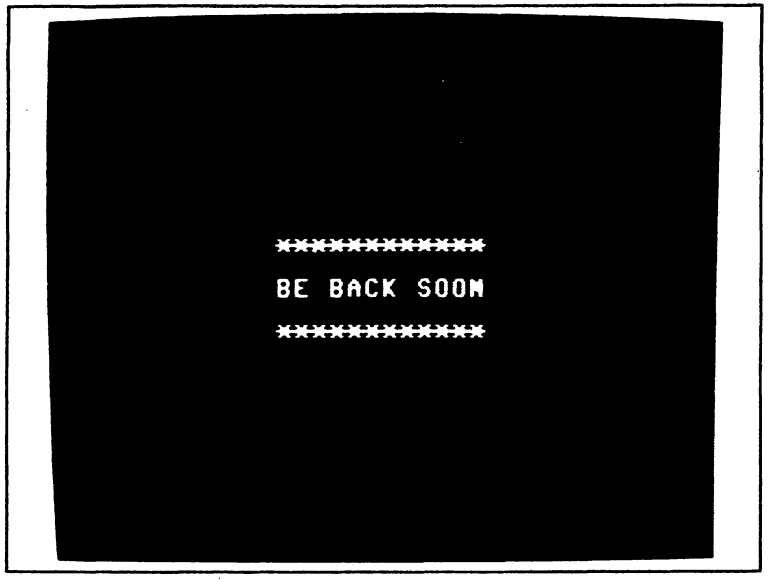


FANCY ROUTINES

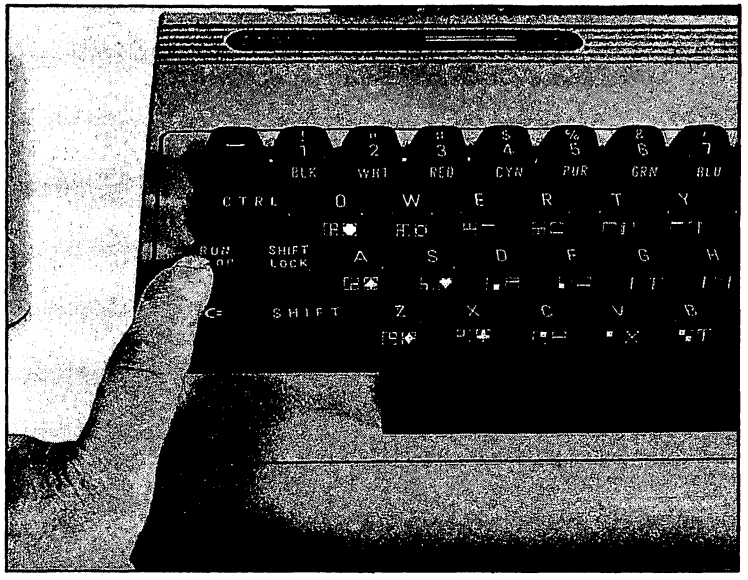
A Blinking Bulletin Board

266. The little program shown LISTed here will cause the message, BE BACK SOON, embellished with two asterisk bars, to flash on and off at a rate controlled by the delay loops on lines 60 and 80. Clear memory and clear the screen, then type in the program, entering each line to memory by pressing RETURN.

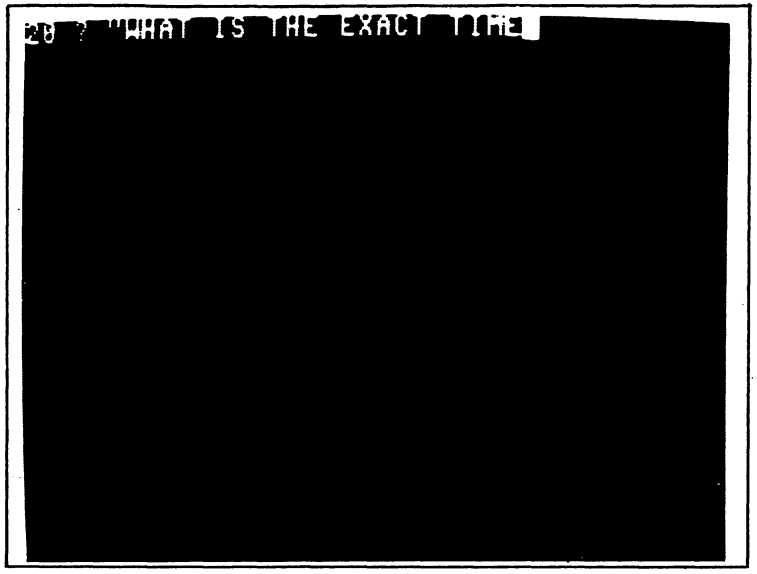
NOTE: In a loop, following NEXT with the variable name is optional. I have not done so here, nor will I do so in some of the programs which follow.



267. RUN the program. If you don't like the flashing rate, change the delay loop lines to suit you.

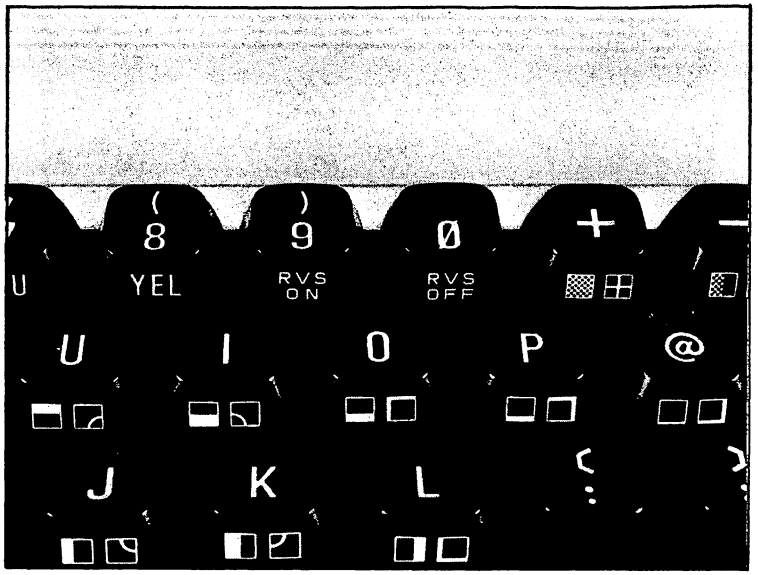


268. It is only necessary to press the RUN/STOP key to interrupt the main program loop and end the program.

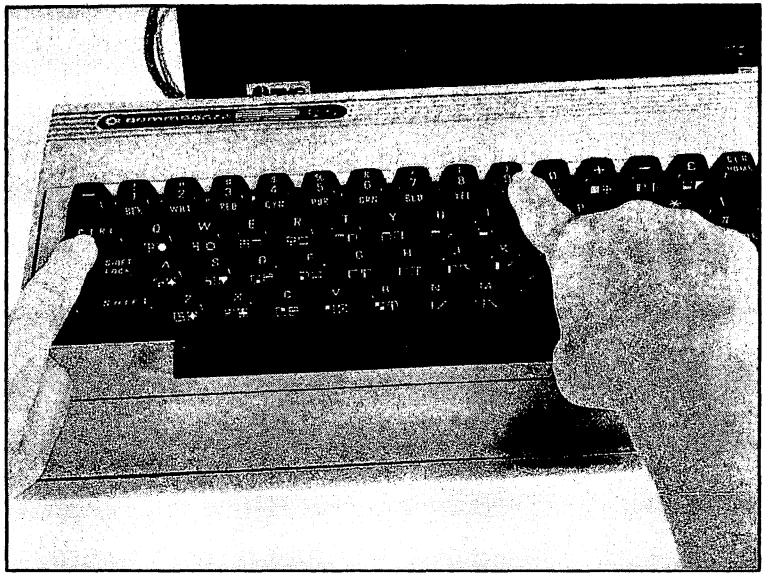


A 24-Hour Clock

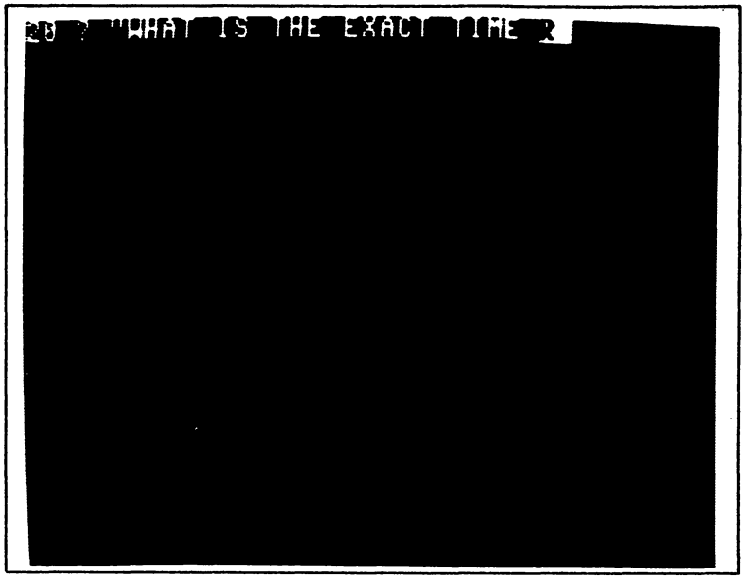
269. Your computer has a built-in clock. You can turn your C64 and its monitor into a very expensive digital clock with a simple program, only one line of which contains anything new to you. Clear memory and clear the screen, then let's begin with that line, which will contain a reverse-video prompt. Type in: 20 ? "WHAT IS THE EXACT TIME.



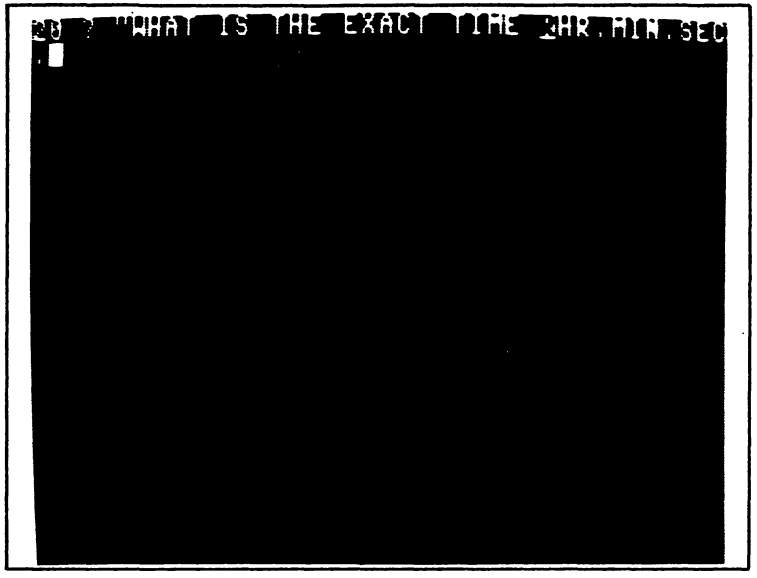
270. Notice the abbreviations RVS/ON and RVS/OFF on the fronts of the 9 and 0 keys, respectively.



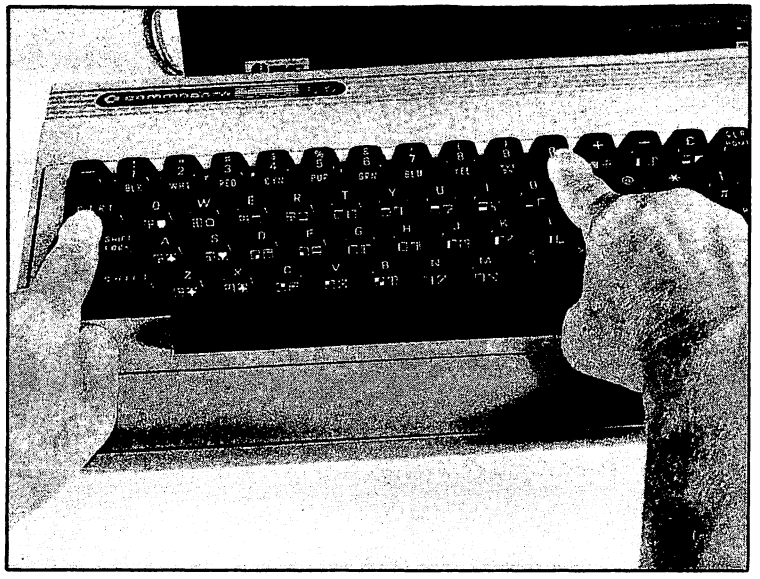
271. Move the cursor one space to the right and press the CTRL and 9 keys at the same time.



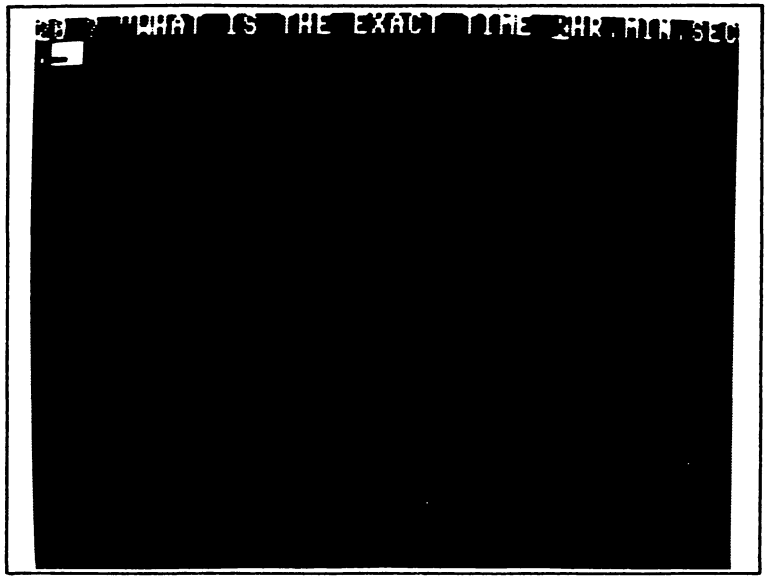
272. That will leave a reverse-video (black on white) letter R on the screen.



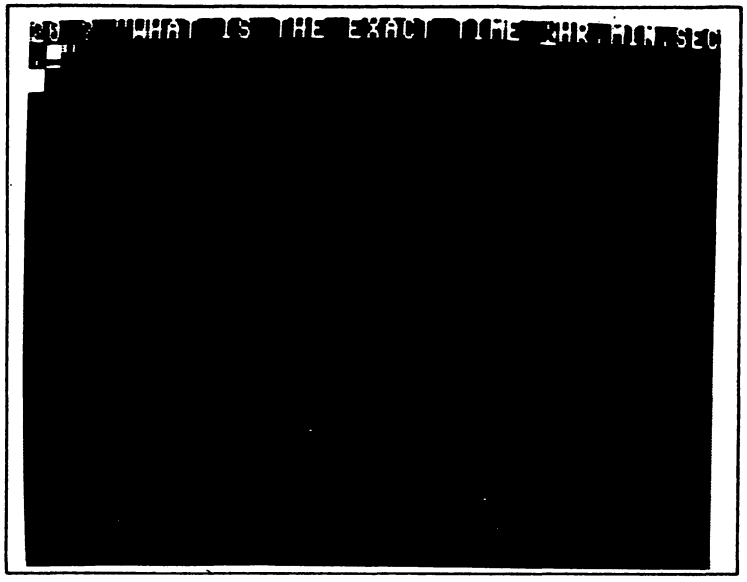
273. Type: HR.MIN.SEC.



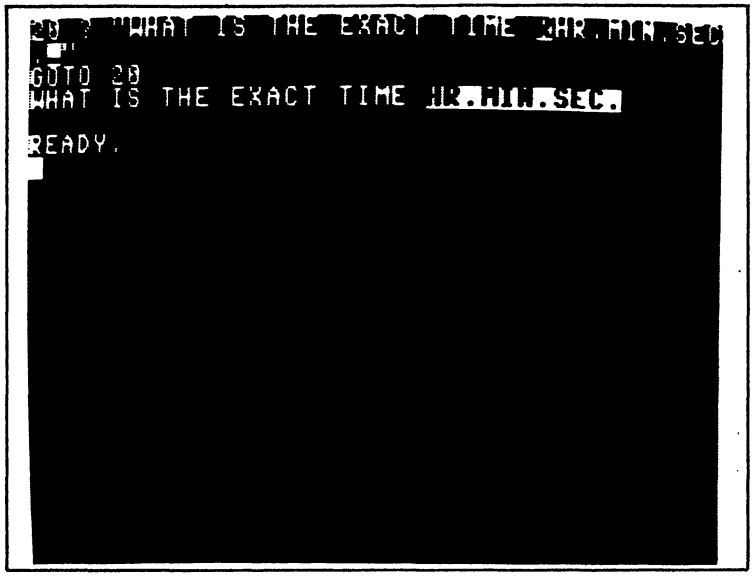
274. Then press the CTRL and 0 keys.



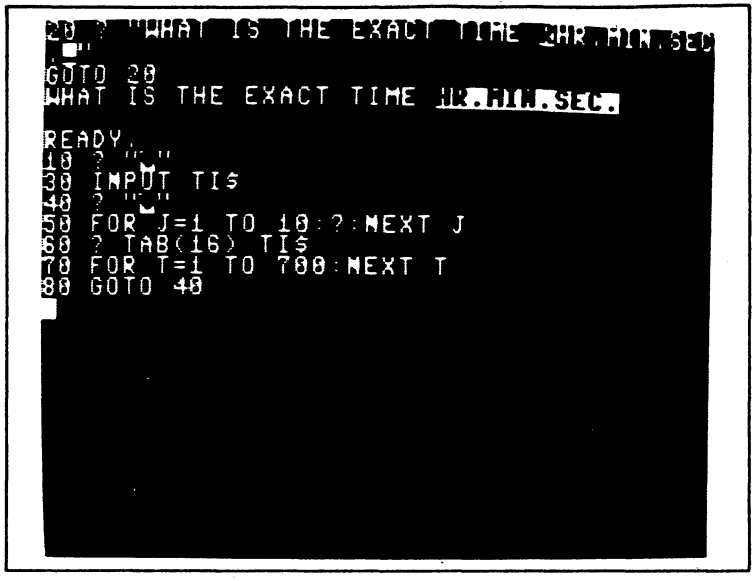
275. That will leave a reverse-video underscore (_) on the screen.



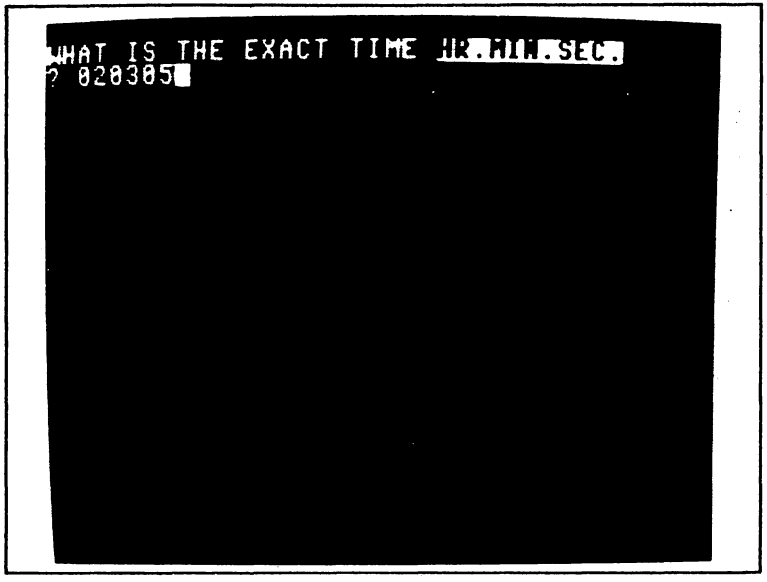
276. Type in the final quotation mark " (and press ENTER) to place the line in memory.



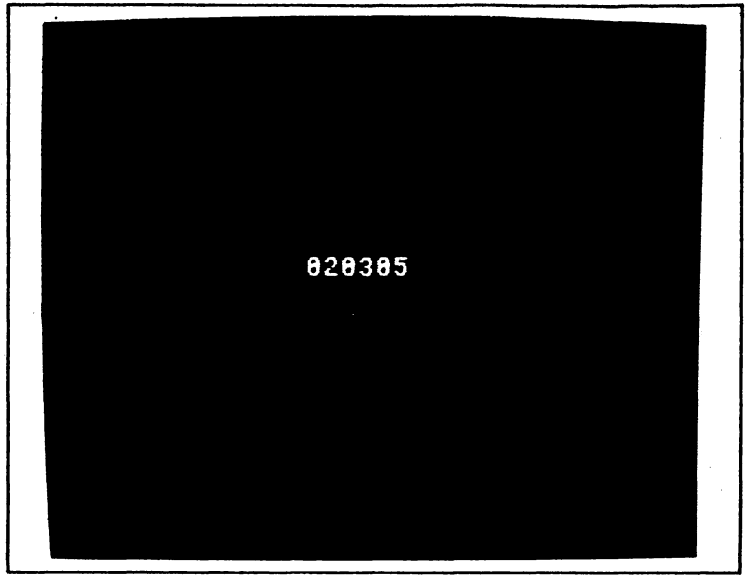
277. The reverse-video letter (R) and underscore you see in the program line on the screen will not appear when the program is RUN, *but everything between them will appear in reverse video.* To see this type in: GOTO 20 (and press RETURN).



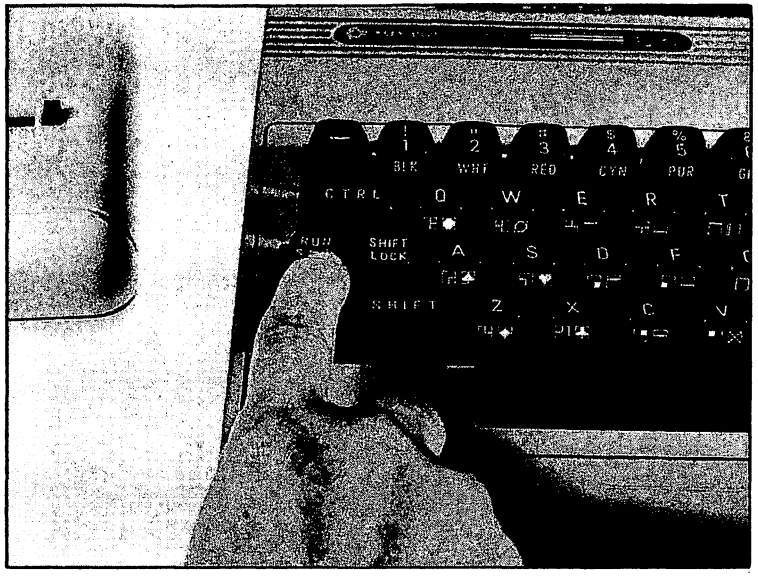
278. Now type in and enter the rest of the program lines exactly as they are shown here.



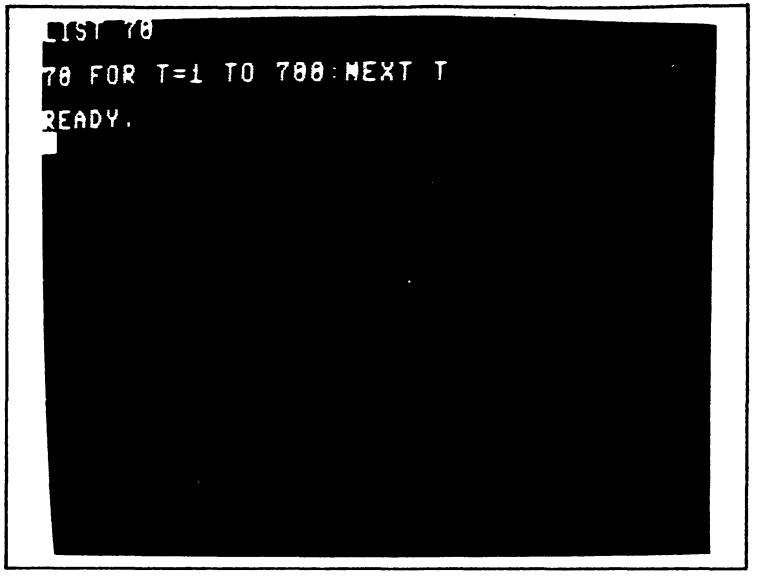
279. RUN the program. Line 20 will ask you to tell the computer the exact time in hours, minutes, and seconds. The C64 keeps universal, 24-hour time (universal everywhere but, alas, the U.S.), so if it's 8:46 and six seconds PM, you type in 204606. If it's three minutes and five seconds after two AM, be sure to type in 020305, and so forth.



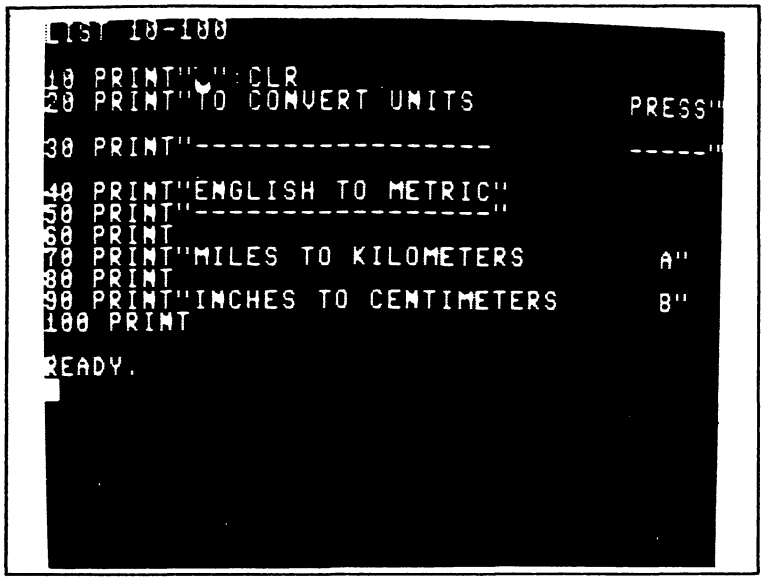
280. Press RETURN and the machine will continuously PRINT the time at the center of the screen, correcting its display once each second.

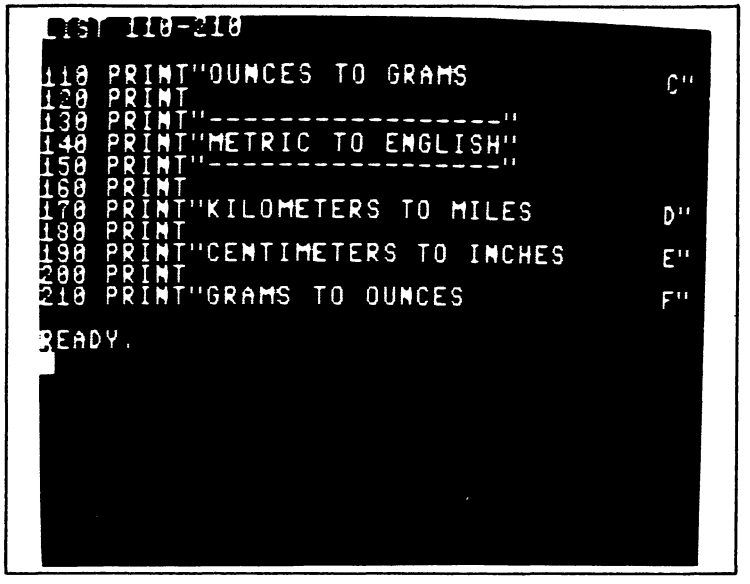


281. A single press of the RUN/STOP button will interrupt the loop and stop the program.



282. Line 70 was arrived at by trial and error. It was included to keep the same number of seconds from appearing on the screen for two consecutive changes.

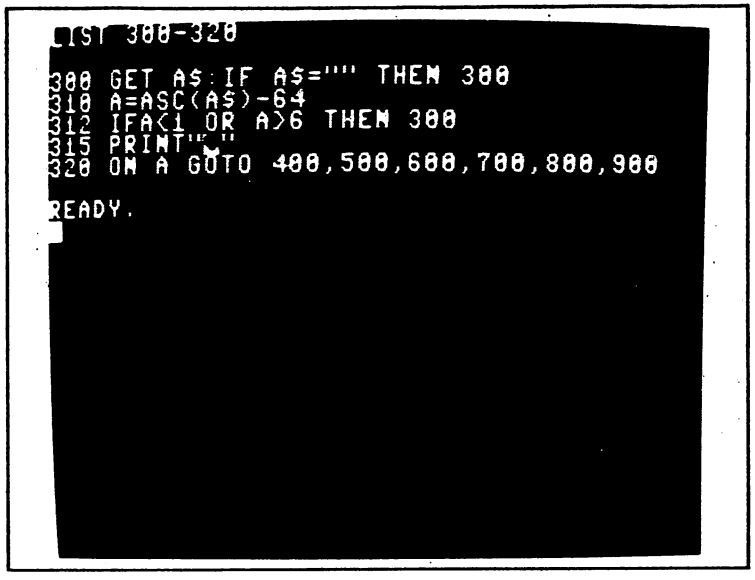




Menus

283. The program that is LISTed in this, and the next five illustrations is designed to show you the most convenient method of presenting choices to the users of your programs. You can handle the problem by offering the user a menu. Most of the lines LISTed here—20 through 100—contain nothing you haven't already dealt with. They are simple PRINT statements used to get the menu on the screen. There is a new command on line 10. It is the CLR command following the colon. It is used to CLeaR old variable values out of memory, just as the first command on line 10 clears the screen. Whenever a RUN command is executed, the variable values are automatically cleared from memory. As you will soon see, however, it is not necessary to use RUN for this program, so you must include a command that will clear memory of old variable values each time the program is used. Before you start to copy the program, be sure to type in NEW (and press RETURN) to clear any old material out of memory, and to clear the screen. Then type in each line and enter them into memory by pressing RETURN at the end of each line.

284. All the lines LISTed here—110 through 210—are simple PRINT statements used to get the menu on the screen. None of them contains anything new to you. Type the lines and enter them into memory.



285. Now, in the LISTed lines 300 through 320 we have reached two special new commands that make selections from a menu very simple. They are GET and ON. The GET command appears in line 300. It tells the computer to GET the character of any key pressed, without the RETURN key having to be pressed, and make the string variable A$ equal to that character. The second command on line 300 tells the computer to wait until a key has been pressed (making A$ greater than an empty string) before going on to the next program line. After a key has been pressed, line 310 establishes a numerical variable A with the value of the ASCII code (a code you will learn more about as you gain programming experience) of A$ minus 64. (The command ASC(A$) provides the ASCII value of the string variable A$.) This is done for simplicity's sake so that the A key will have a value of one, the B key a value of two, and so on. Notice that the command word LET has been omitted from line 310. Its use is optional with the C64. Line 312 tells the computer to ignore any keys other than A through F, those used in the menu. Note that the command GOTO has been omitted from line 312. Its use after the conditional word THEN is optional. Line 315 simply clears the menu away. Line 320 commands the machine that on receiving a value between one and six of the variable A, it is to GOTO any of six different program lines, each separated by a comma. This command is quite ingenious. It will automatically determine the appropriate line for any of the six values of A. On receiving a value of 1 for A, the computer automatically goes to the first of the series of line numbers in the line. When A equals 2, the second line number in the series is gone to, and so on. If you still have the strength, or the interest, after that long-winded explanation, type the lines in, exactly as you see them here, and enter them into memory by pressing RETURN after each one. Stick with it, even if you were completely confused by the verbiage above. The eventual result will be worth the effort.

```
LIST 400-900

400 Z$="MILES":Y=1.6093:W$="KILOMETERS":
GOTO1000
500 Z$="INCHES":Y=2.54:W$="CENTIMETERS":
GOTO1000
600 Z$="OUNCES":Y=28.3495:W$="GRAMS":GOTO
1000
700 Z$="KILOMETERS":Y=.6214:W$="MILES":G
OTO1000
800 Z$="CENTIMETERS":Y=.3937:W$="INCHES"
:GOTO1000
900 Z$="GRAMS":Y=.0353:W$="OUNCES"

READY.
```

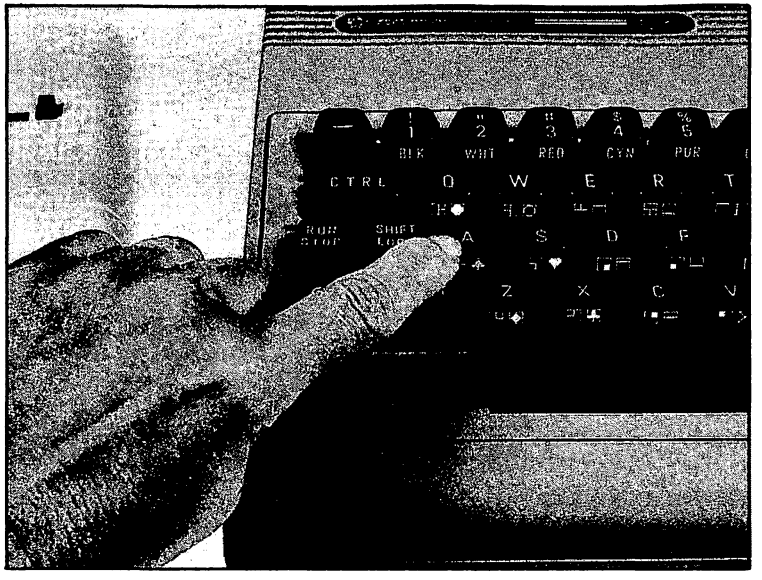
```
LIST 1000-1060

1000 PRINT"HOW MANY ";Z$
1010 INPUT Z
1020 X=Y*Z
1030 PRINT W$;"=";X
1035 PRINT:PRINT
1040 PRINT"TO RETURN TO MENU PRESS  M"
1050 GET B$:IF B$=""THEN 1050
1060 IF B$="M" THEN 10
READY.
```

286. Each of these LISTed lines—400 through 900—are the ones called by line 320. Each one contains values for three different variables: two string variables named Z$ and W$, and a numeric variable named Y. Each line ends with the same command, which is GOTO line 1000. That is where the variables will be used. Copy each of the lines exactly as shown here. Be careful to separate each variable value with a colon (:) and to place a colon before the GOTO command on each line. Enter each line into memory by pressing RETURN after it is completely typed in.

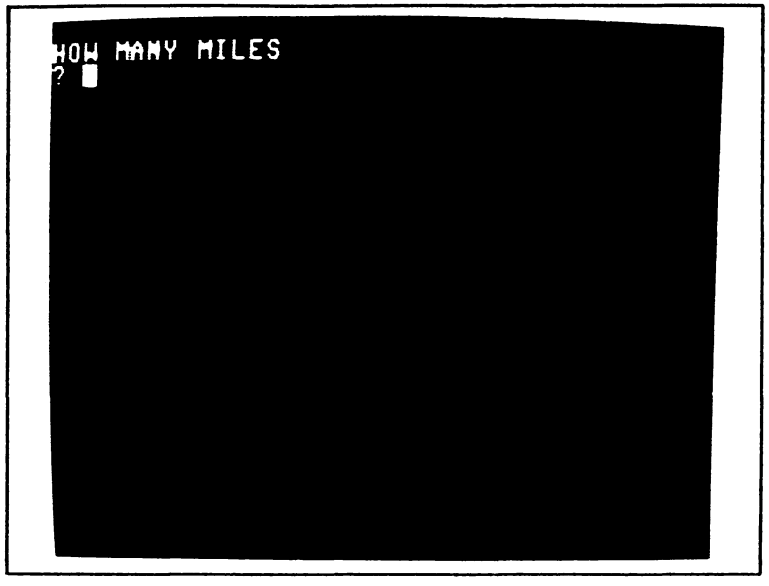
287. LISTed lines 1000 through 1060 contain nothing new. Simply type them in and enter them to memory. I should call your attention to lines 1040 through 1060, however. They offer the user of this little program the ability to get back to the original menu without having to rerun the program. That ability is the reason the CLR command had to be used in program line 10. I probably ought to point out, as well, that if any key other than M is pressed when the program offers the option on line 1040, the program will just run out of instructions, come to an end, and the word READY will appear on screen along with a flashing cursor, announcing that the machine is through with the program and ready to go on to other things.

```
TO CONVERT UNITS          PRESS
-----------------         -----
ENGLISH TO METRIC
-----------------

MILES TO KILOMETERS         A
INCHES TO CENTIMETERS       B
OUNCES TO GRAMS             C

-----------------
METRIC TO ENGLISH
-----------------

KILOMETERS TO MILES         D
CENTIMETERS TO INCHES       E
GRAMS TO OUNCES             F
```

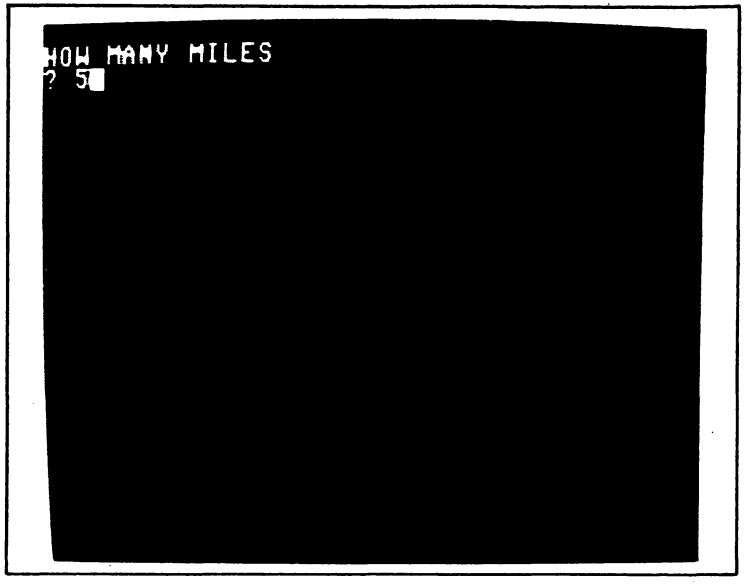
288. Now, at last, RUN the program. If you managed to type in all those lines perfectly, this is what you should see. The program offers to convert miles to kilometers if you press A; inches to centimeters if you press B; ounces to grams if you press C; kilometers to miles if you press D; centimeters to inches if you press E; or grams to ounces if you press F.



289. Let's start at the beginning, and press the A key.



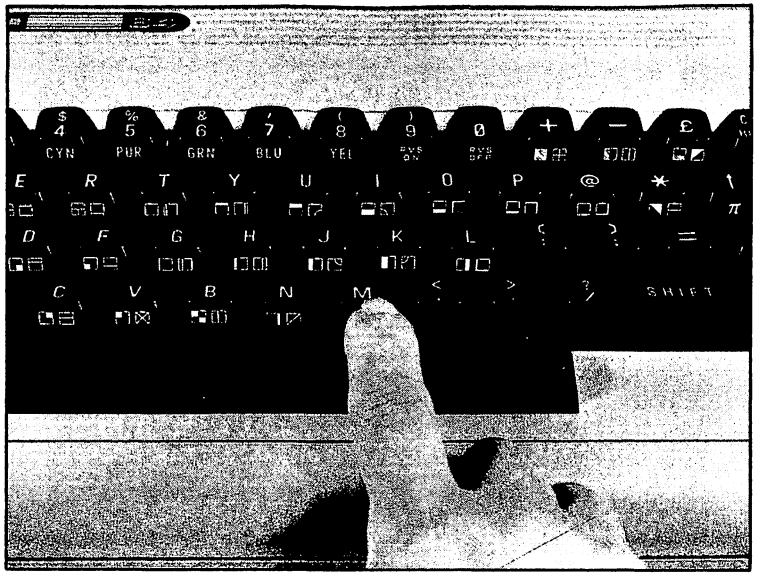
290. Notice that it is not necessary to press RETURN. As soon as you press the A key, the screen changes to this.



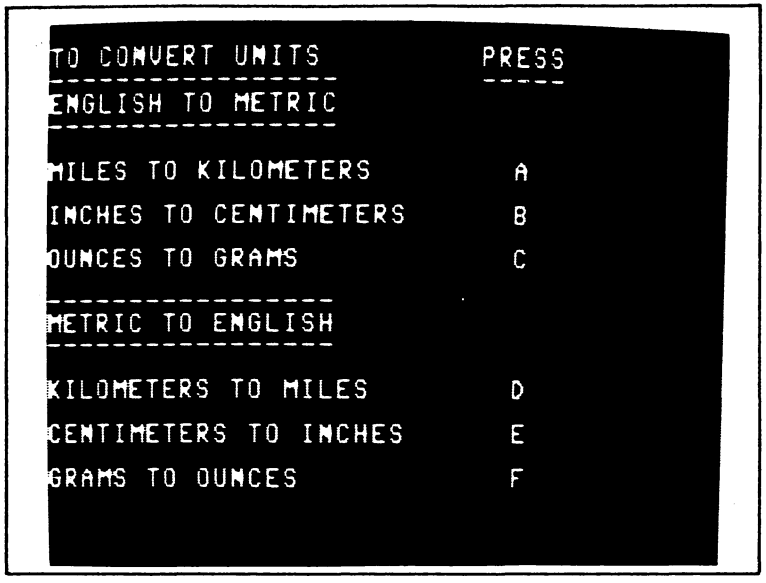
291. Whenever you see a question mark followed by a blinking cursor, you know that you will have to enter your answer to the question by first typing in the answer and then pressing RETURN. That being the case, type in a 5.

```
HOW MANY MILES
? 5
KILOMETERS= 8.0465

TO RETURN TO MENU PRESS  M
```

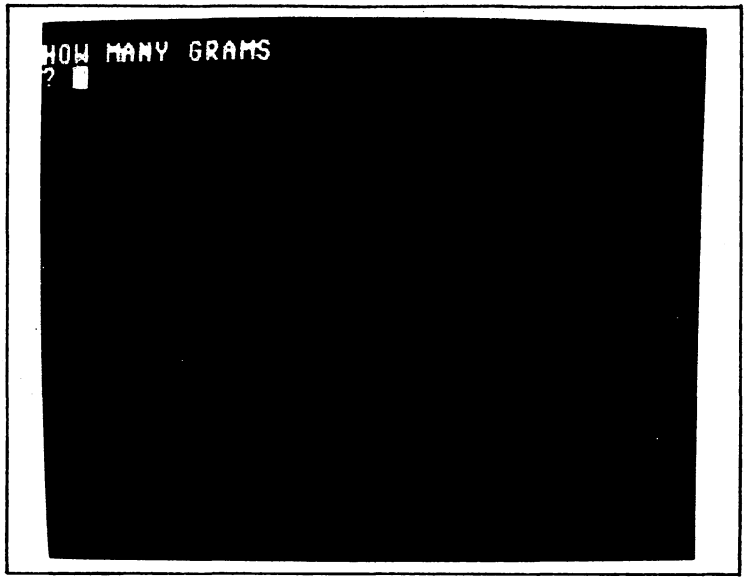
292. Press RETURN and you are immediately provided with the number of kilometers equivalent to five miles, and an invitation to try the program again by pressing the M key and returning to the menu.



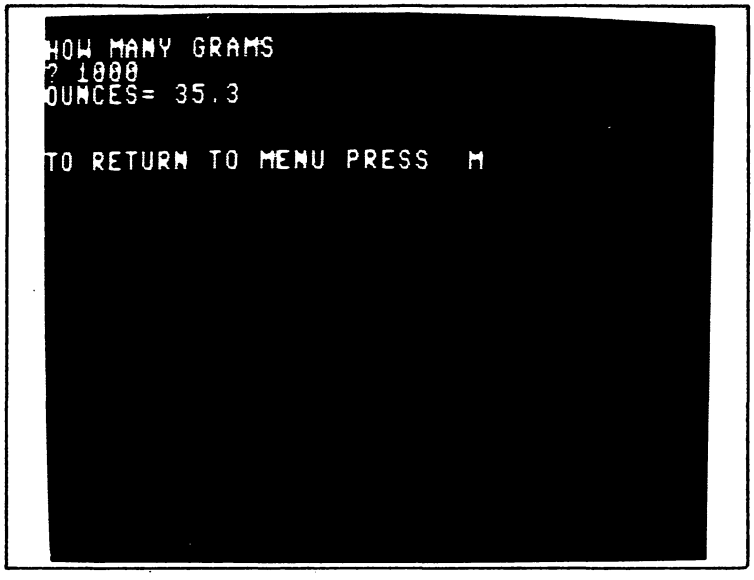
293. Let's give it another try. Press the M key . . .



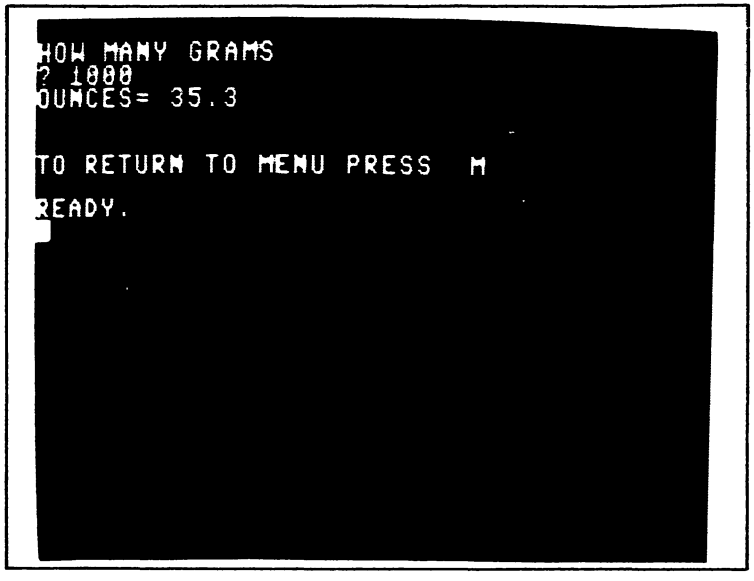
294. . . . and lo and behold! We are instantly given back the menu.



295. One more check ought to convince you that it really works. This time press the F key, and you will be asked the number of grams you want to convert to ounces.

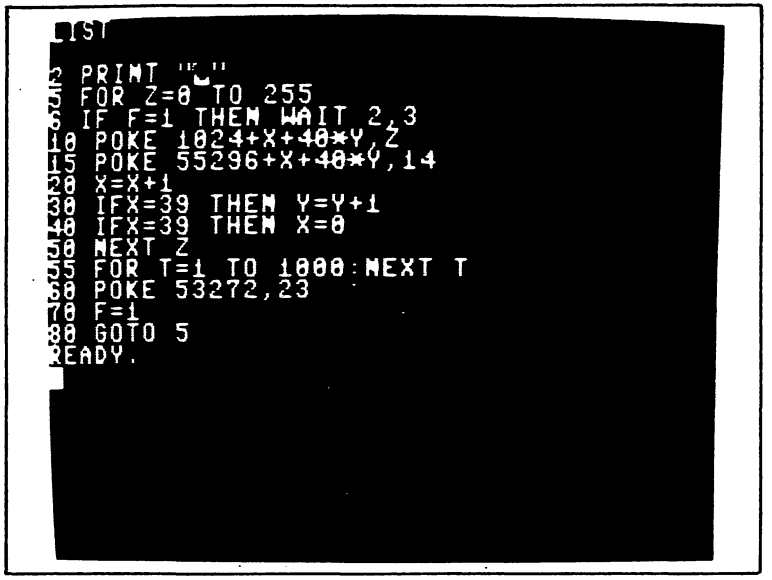


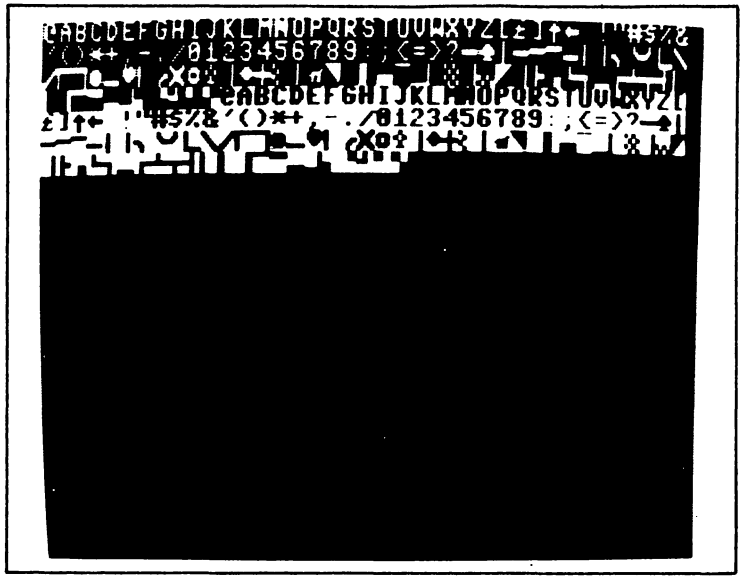
296. Type in 1000 (press RETURN), and you will get your answer: 1000 grams equals 35.3 ounces. You will also be asked if you want to return to the menu.



297. This time press any key but the M, and you will exit the program, as indicated by READY and the blinking cursor the C64 offers.

NOTE: Now that you have had a chance to use the program, I'd suggest going back to illustrations 283 through 287 to see how the program is put together in essentially three modular components—the menu (program lines 10 through 260), the data (program lines 400 through 900), and the calculations (program lines 1000 through 1060).

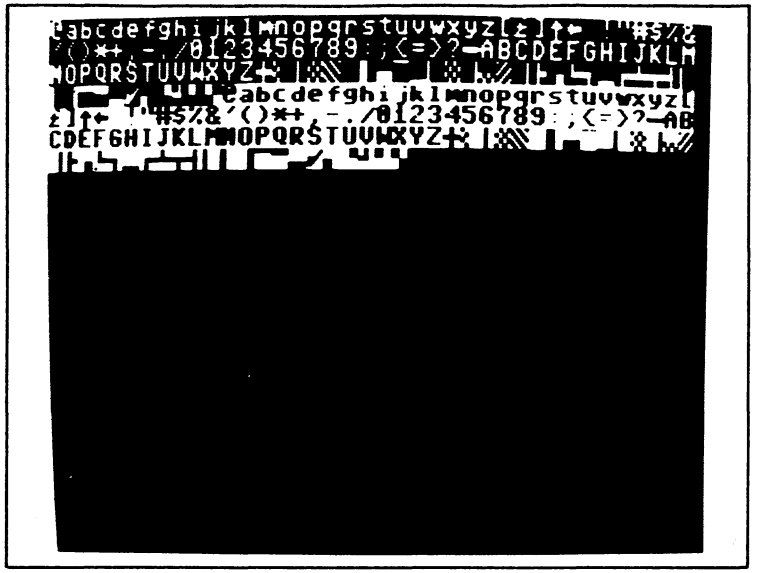




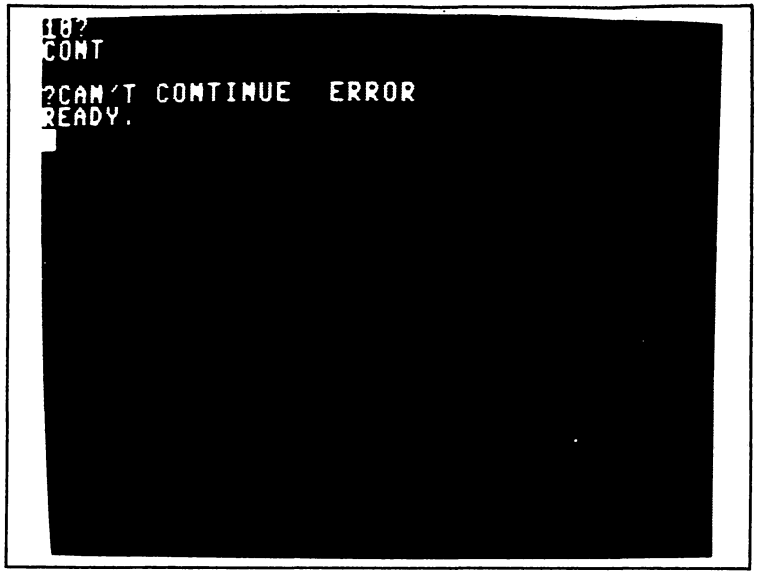
THE CHARACTER SET

298. There are 256 different characters, symbols, graphic designs, and punctuation marks that the C64 can put on the screen (although it can't put all of them on the screen at the same time). To see them, first type in NEW (and press RETURN) to clear memory, then clear the screen and copy the program shown LISTed in this illustration, entering each line into memory by pressing RETURN at the completion of the line. (I won't try to explain how most of the program works, but I suggest that you return to it after you have finished the section on graphics. It should make sense then. Line 60 does nothing but switch to lower case.)

299. RUN the program. The screen will slowly fill with the upper-case character set, until it appears like this.

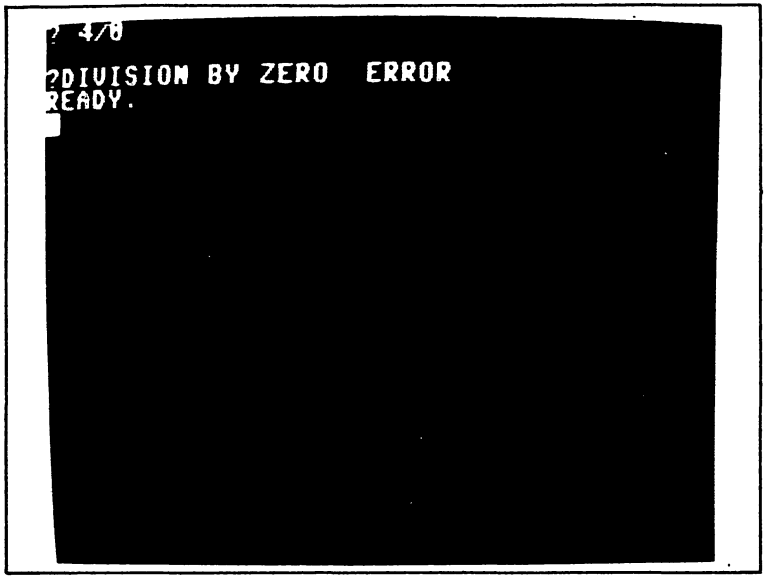


300. Then, after a short pause, it will abruptly change to show the lower-case character set, and end up looking like this. There is some redundancy and overlap but you will see everything at least once. Use the RUN/STOP and RESTORE keys together to get the READY and the blinking cursor.

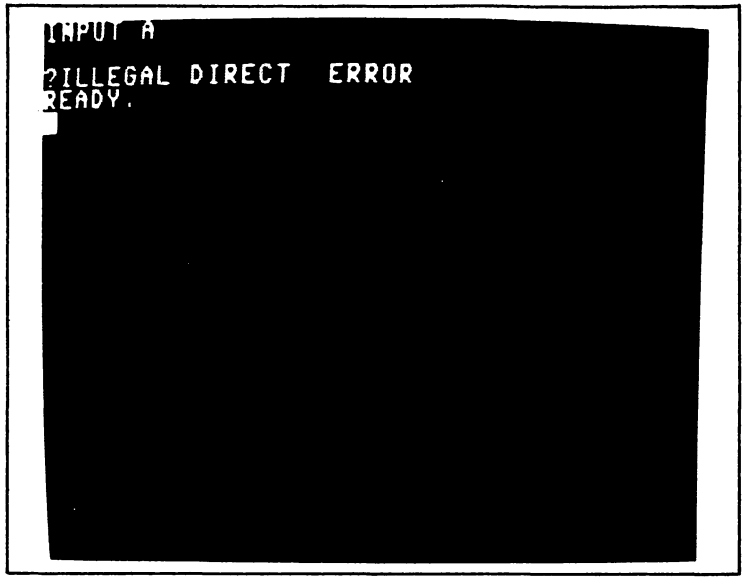


ERROR MESSAGES

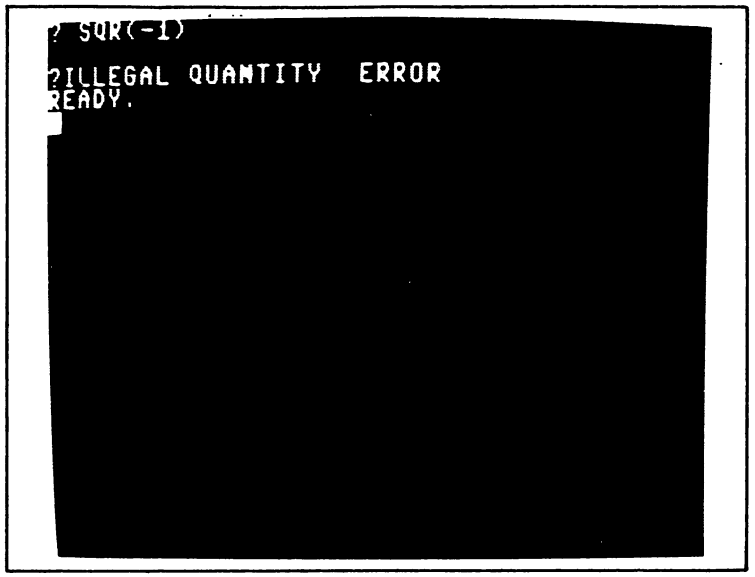
301. Your computer constantly checks up on how you are doing. When you foul up, it will tell you how you fouled up, and even where. To see an example of this, type NEW (and press RETURN) to clear memory. Clear the screen. Type in: 10? (and press RETURN). Type CONT (and press RETURN). The computer will let you know that it can't CONTinue because the program was never RUN.



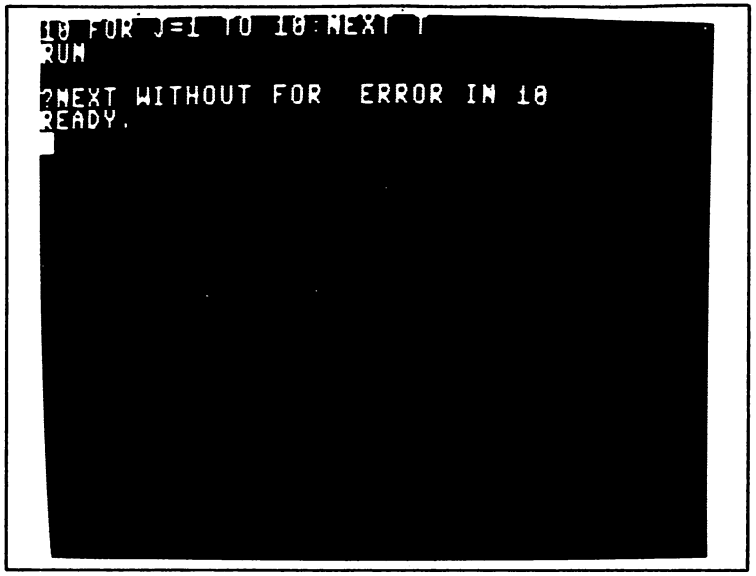
302. If you try to divide by zero, the computer scolds you. To see this, clear the screen, then type in: ? 4/0 (and press RETURN).



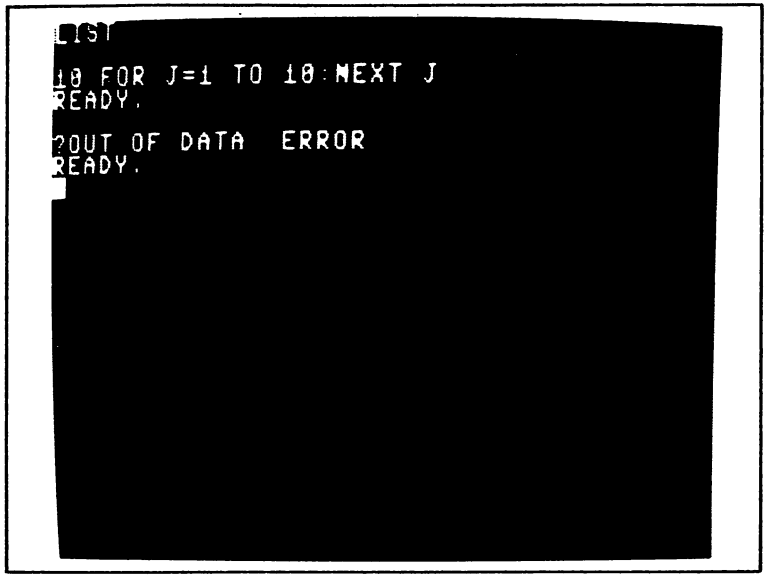
303. You can't use the INPUT command in the immediate mode. If you do, the computer gets mad. To check this, type in: INPUT A (and press RETURN).



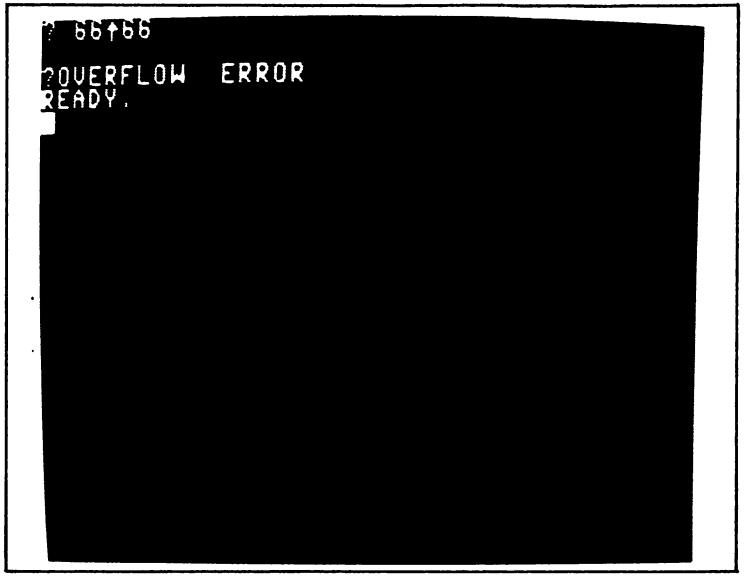
304. You can't ask the C64 for a function of a number that is out of allowable range. To see this, clear the screen, then type in: ? SQR(−1) (and press RETURN).



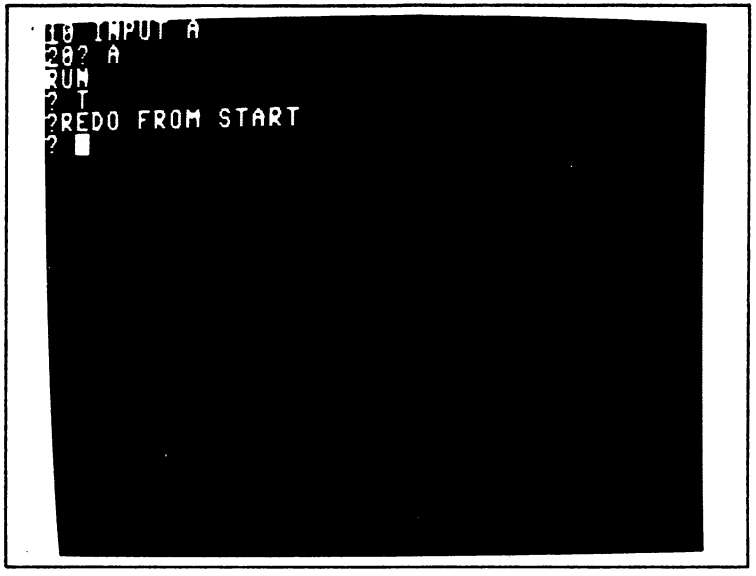
305. If you mess up a FOR-NEXT loop, the machine will let you know. For example, first clear the screen, then type in: 10 FOR J=1 TO 10:NEXT T (and press RETURN). Now RUN it. Notice that in this example, in the delayed mode, the computer named the program line on which the error occurred.



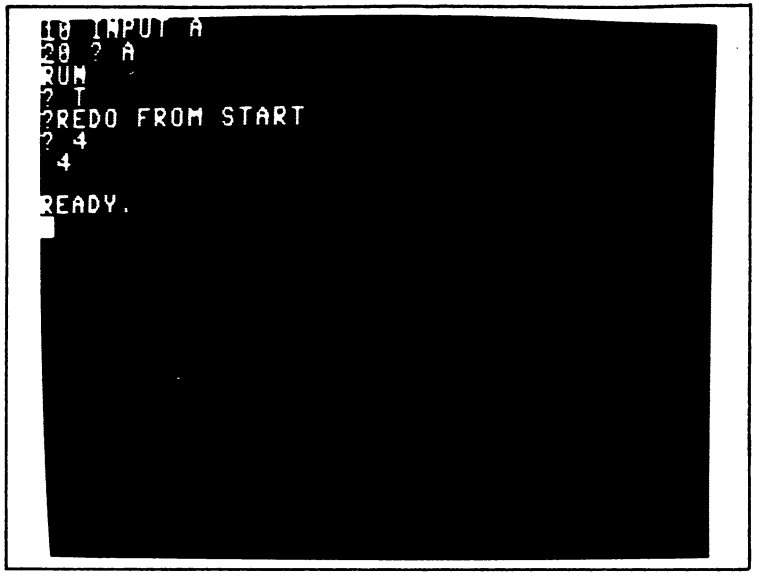
306. The OUT OF DATA error message is really meant to advise of a problem with two commands (READ and DATA) not covered in this book. You will, however, also encounter it when editing program lines. To see this, clear the screen, then type in: LIST (and press RETURN). The program line used in illustration 305 will appear on the screen. Edit the line to change the final T to a J, then press RETURN. That will place the cursor over the word READY that showed up on the screen when you listed line 10. Now press RETURN again, and you will get an OUT OF DATA error report.



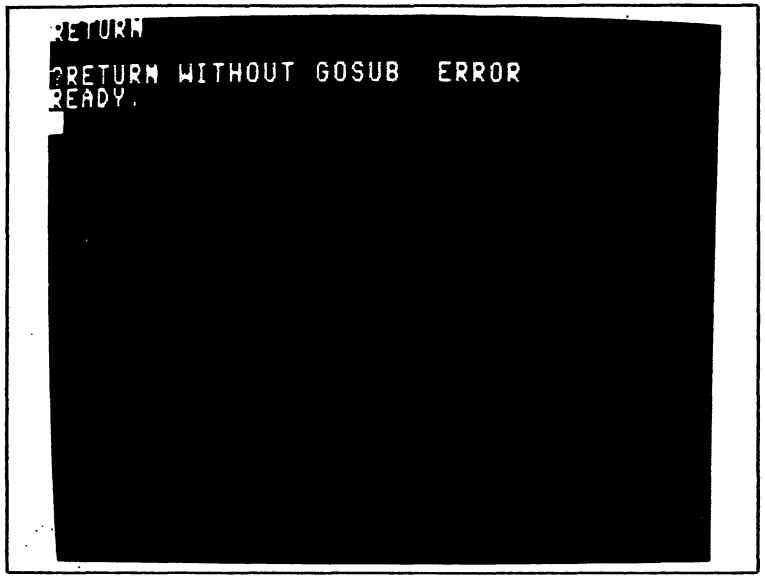
307. If you try to compute a number which is greater than the numerical limit of the C64 (1.70141183E+38, which equals the number to the left of the E with the decimal place moved 38 places to the right) you will be so informed. To see this, clear the screen, then type in: ? 66↑66 (and press RETURN).



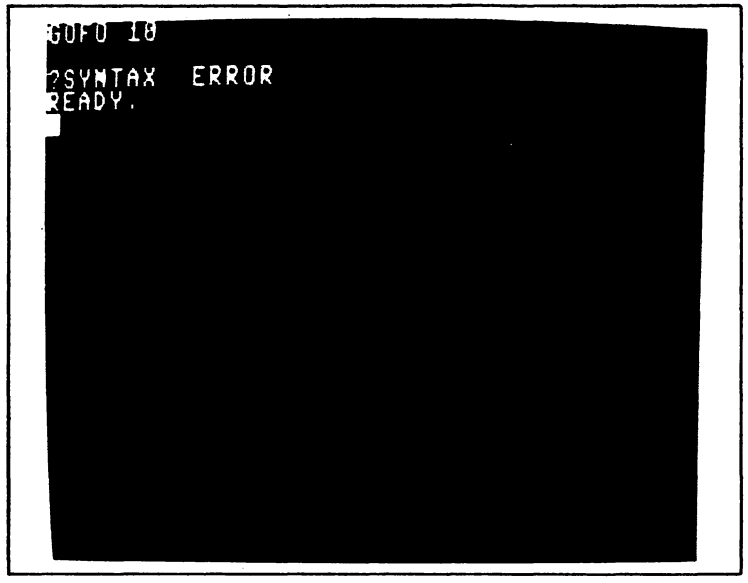
308. If the computer is expecting numerical data and it gets characters instead, it won't accept them. It will tell you to get your act together and give it what it expects, and it will give you the chance to do so. Check this by clearing the screen, then typing in: 10 INPUT A (press RETURN), 20? A (press RETURN). Then RUN the program and you will get a question mark, followed by a blinking cursor. Type the character T (actually, any character will do) and press RETURN. The computer will tell you REDO FROM START, which is C64 language for, "Give me a number, dummy!"



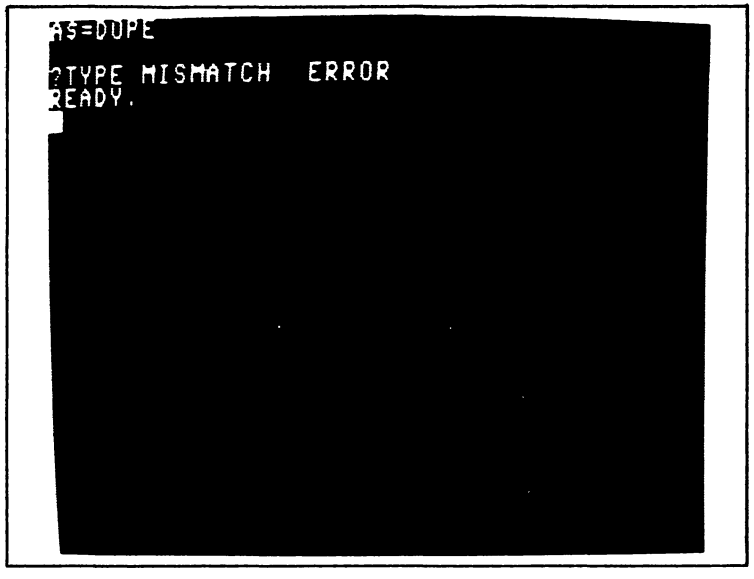
309. And since there is a second question mark and a blinking cursor provided, the machine is inviting you to correct your earlier mistake. Type in: 4 (and press RETURN). Now the computer is happy and goes on to line 20, which commands it to print the 4 you just gave it. It does.



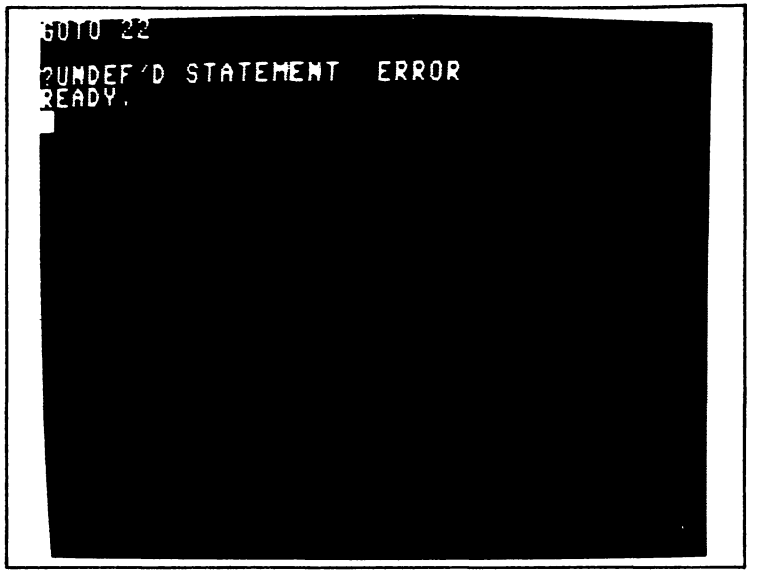
310. If you manage to get a subroutine which ends with a RETURN command without using the command GOSUB, the computer will let you know that you blew it (as it did in our earlier example; see illustrations 244 and 245). To see this, clear the screen, then type in: RETURN (and press RETURN).



311. If you misspell a command, or omit a parenthesis, the computer will chastise you for your bad syntax. To see it do so, clear the screen, then type in: GOFO 10 (and press RETURN). What would you expect to happen when you spell GOTO with an F?

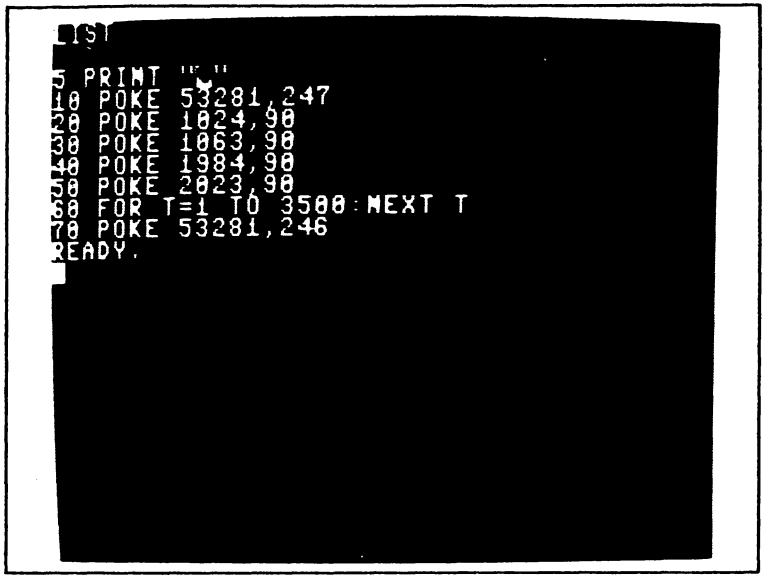


312. If the computer is expecting a string value and you forget to precede that value with a quotation mark, the machine will let you know of your transgression. To see this, clear the screen, then type in: A$=DOPE (and press RETURN).



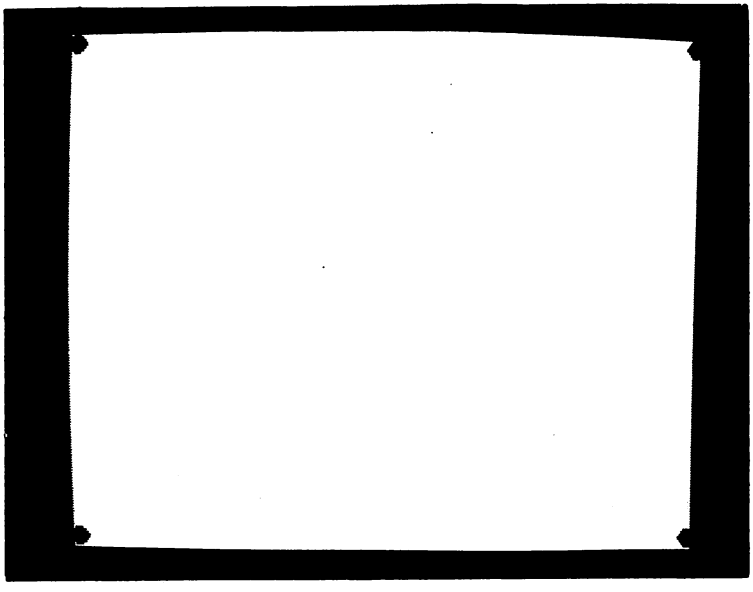
313. If you give the computer a command to GOTO, GOSUB, or RUN a line number that doesn't exist, it will tell you so. Check this by clearing the screen, then typing in: GOTO 22 (and pressing RETURN).

NOTE: There are several other error messages, in addition to those illustrated here. They are errors connected with accessories or commands not covered by this book. You can be sure you will encounter them as you go on to more advanced programming and add accessories to your C64.



GRAPHICS

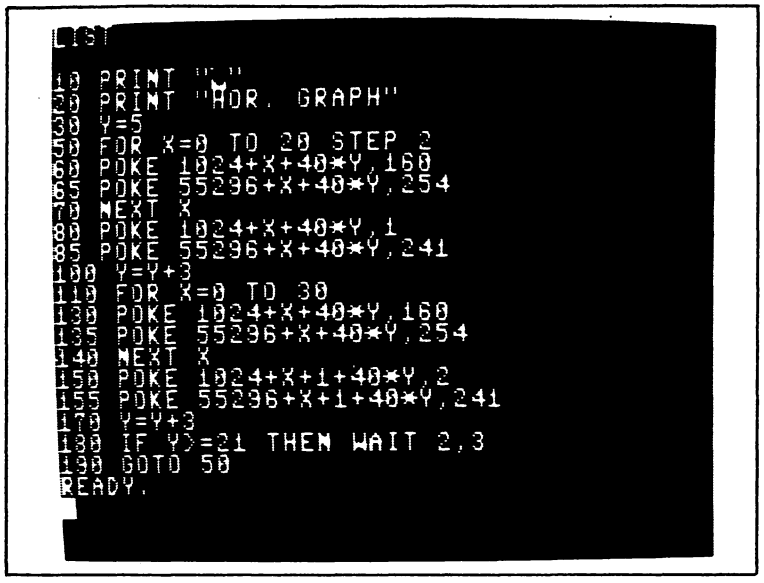
314. The graphics capabilities of the C64 enable the machine to be used for anything from video games to graphing to development of special character sets. However, this presentation will be kept as simple and uncomplicated as possible for the sake of clarity. The BASIC command used in graphics is POKE. It enables you to place a numerical value, corresponding to a character or symbol, and also to a color, at specific locations in the computer's memory. You may recall that the screen is 40 columns wide by 25 rows high. Each intersection of a column and row is referenced by two specific locations in memory—one for characters and symbols, and one for color. The locations for characters or symbols begin with position number 1024, (row 0, column 0), progressing to 1063 (row 0, column 39), progressing to 1984 (row 25, column 0), progressing to 2023 (row 25, column 39). You can put a wide choice of characters or symbols on the screen in any locations you wish if you POKE them into the places you select. (Any screen position for characters or symbols can be readily calculated with the general formula: POKE 1024+X+40*Y, where X = the column number and Y = the row number.) But there is a catch. If you simply POKE the symbol into the desired location, it will indeed go just where you put it, but it will be the same color, blue, as the screen, and so it will be invisible. Therefore, before you POKE any symbols or characters, you must either tell the computer to change the background color to one which will contrast with the color of the symbols or characters (as we will in this example), or you can POKE a specific color into each location at which you have placed a symbol or character. Clear memory and clear the screen, then type and enter into memory the program lines shown LISTed in this illustration.



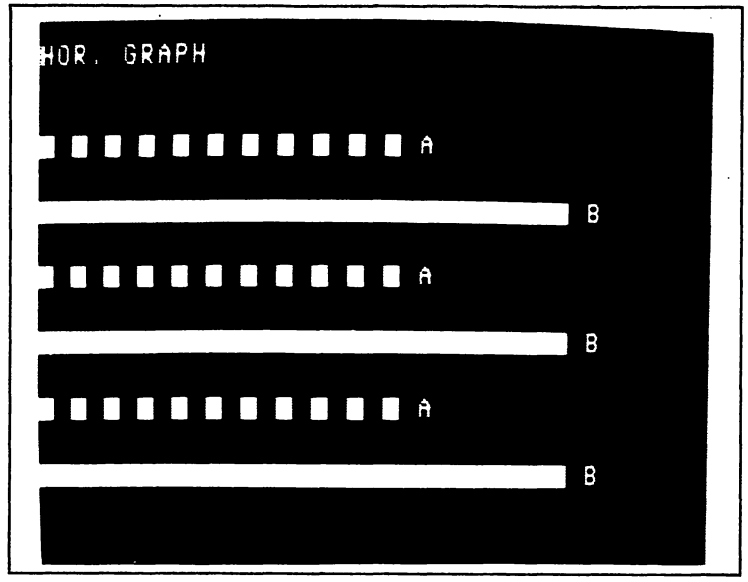
315. RUN the program. Note that the center of the screen turns light and the border turns dark (on color TVs or monitors the screen will be yellow with light-blue borders). A diamond will appear in each of the corners, then will disappear as the screen resumes its normal appearance (gray or black on black-and-white TVs, blue on color TVs) as the program comes to an end. READY and a cursor will then reappear. All this occurs very rapidly. Refer to illustration 314, and we will see how this program works. Line 5 simply clears the screen. Line 10 controls the screen background color. If you POKE a 247 into memory location 53281, it causes the background color to change to yellow (I'll explain that neat trick in the Color section which follows). The proper format for POKE commands is POKE, followed by the memory location to be POKEd, followed by a comma, followed by the numerical value. Lines 20 through 50 POKE a diamond shape (90) into each of the corners of the screen. (For other characters and symbols that can be POKEd see Appendix E of the instruction manual packed with your C64.) They are visible because they are POKEd onto the yellow screen corners in the contrasting color blue, the normal screen background color. Line 60 keeps the diamonds in place for a moment. Line 70 restores the computer to its normal operation (it POKEs the value 246 into memory location 53281 which was the value there before line 10 did its thing).

NOTE: If a diamond is not clear or visible in each corner of the screen, your TV may need adjusting.

SUGGESTION: You can determine the contents of any memory location with the command PEEK. For example, to check on my statement above, the next time you turn the computer on, type: ? PEEK(53281), press RETURN, and you should get the value 246.

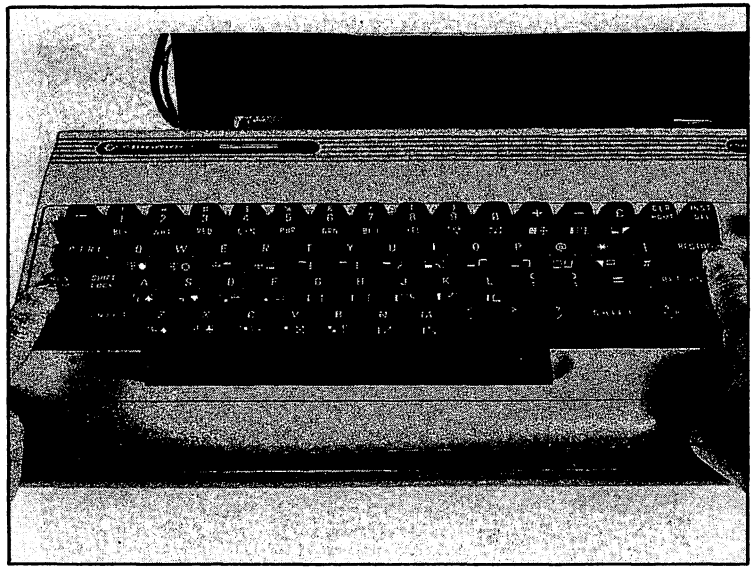


316. Type NEW (and press RETURN) to clear memory. Clear the screen, then type and enter into memory each of the program lines shown LISTed here.



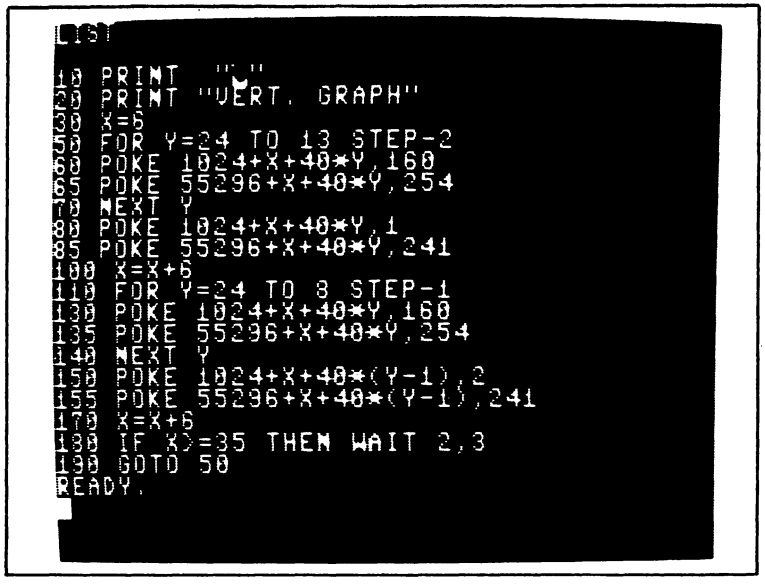
317. RUN the program. Watch it generate the horizontal graph shown here. When you recover from the awe it has inspired, press the RUN/STOP and RESTORE keys. Then refer to illustration 316 to follow this explanation. Lines 10 through 30 are quite straightforward. Lines 50 and 70 constitute a FOR-NEXT loop that caused line 60 to POKE a bar, consisting of six number 160 graphic characters (which happen to be blank spaces), each one separated by a blank column, onto the screen along the fifth row. The blank columns were caused by the command STEP 2 on line 50. It caused the value of X to be incremented from 0 to 2, to 4, and so forth. Line 65 POKEd the color light blue into

each of the blank spaces POKEd in by line 60. Note that the memory locations for color values begin at 55296, and that the same general formula for determining screen locations was used (POKE 55296+X+40*Y). Line 80 POKEd a graphics letter A at the twenty-second column, and line 85 colored it white. Line 100 increased the value of numeric variable Y to 8. Lines 110 and 140 form the FOR-NEXT loop that caused line 130 to generate a bar of 30 graphic characters (character number 160—a blank space) on row eight, and line 135 to fill each of the blank spaces with the color light blue. The bar is solid because, lacking a STEP command, line 110 increased the value of X by 1 each time around the loop. Lines 150 and 155 POKEd the white B label at the end of the bar. Note that X had to be increased by 1 in the general formula on each of these lines, to provide a space between the end of the bar and the label character B. Line 170 increased Y to 11. Line 180 checked to see what row had just been POKEd. Line 190 sent the computer up to the top to start all over again. The program kept looping until Y had been increased to 21, then line 180 told the C64 to WAIT.

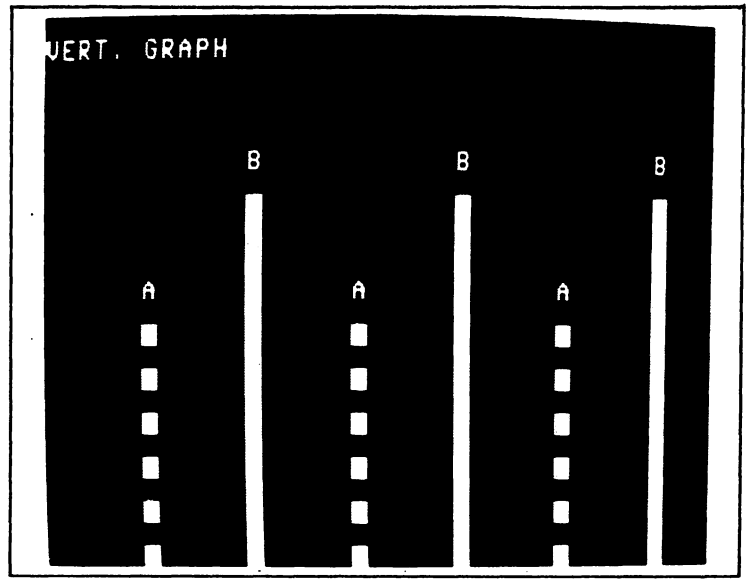


318. With only minor program changes, we can turn the horizontal graph around ninety degrees and convert it into a vertical graph. Press RUN/STOP and RESTORE to clear the horizontal graph off the screen and get the computer ready for more.

SUGGESTION: You may want to record the Horizontal Graph program for future reference. If so, now's the time to do it, as we will modify the program in the next few illustrations.

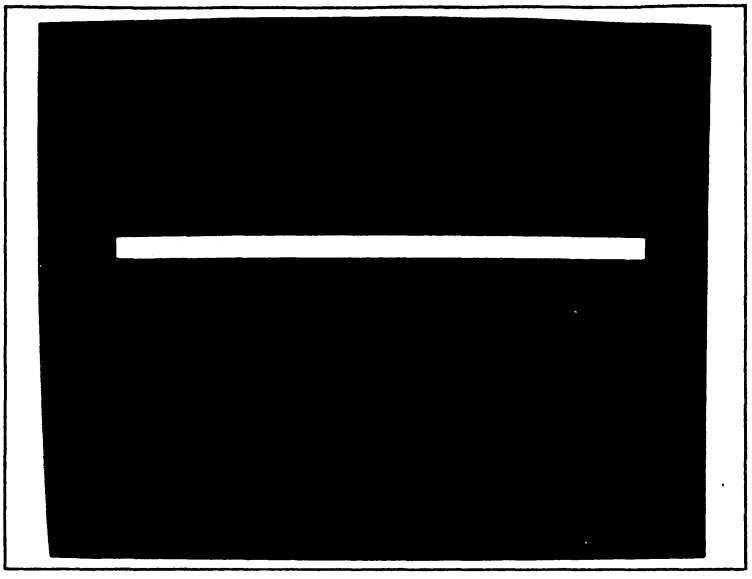


319. If you have gotten good at editing, LIST the Horizontal Graph program lines 10 through 190 and edit them to those shown LISTed here. If your editing is not up to it, clear memory and the screen, then type in and enter into memory the lines LISTed here.



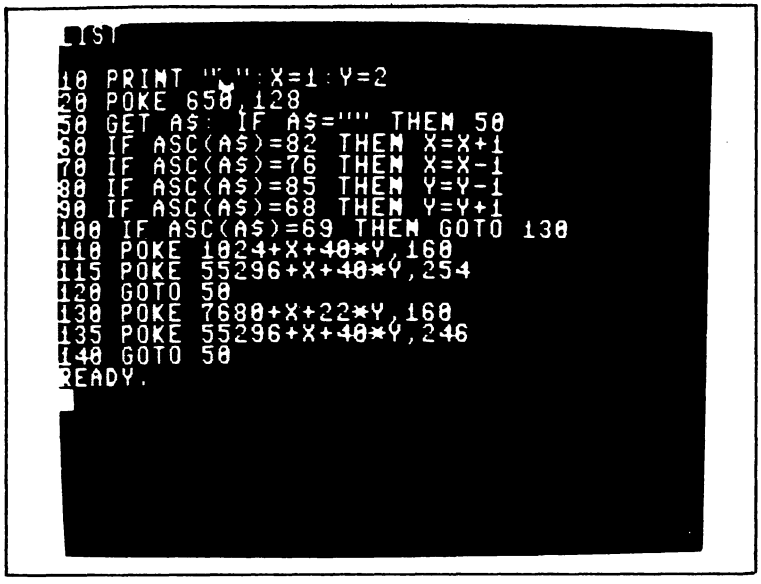
320. RUN the program. Ain't programming grand! Note that the structure of the programs for both horizontal and vertical graphs is identical. Only minor changes were required to convert one to the other. Note also that lines 50 and 110 started Y at its highest possible value and counted backward with STEP commands calling for negative quantities.

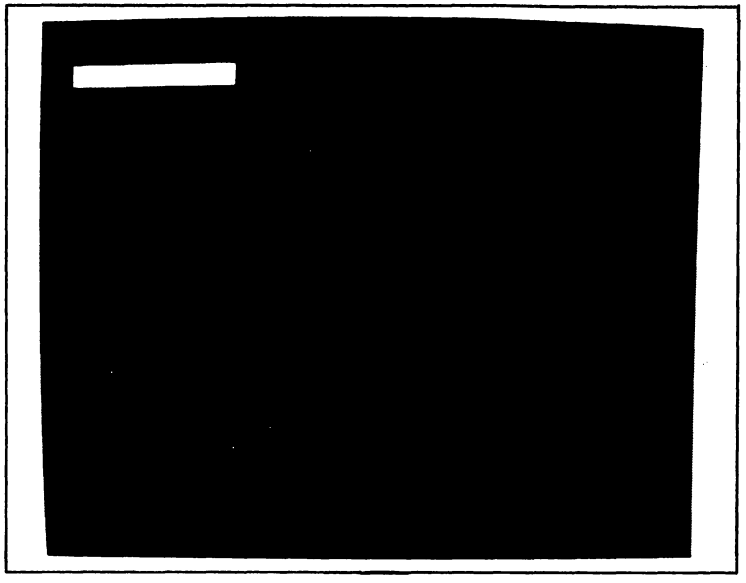
```
LIST

10 PRINT "■"
30 Y=10
40 FOR X=5 TO 35
50 POKE 1024+X+40*Y,160
55 POKE 55296+X+40*Y,254
60 FOR T=1 TO 50:NEXT T
70 NEXT X
80 FOR T=1 TO 1000:NEXT T
90 FOR X=35 TO 5 STEP -1
100 POKE 1024+X+40*Y,160
105 POKE 55296+X+40*Y,246
110 FOR T=1 TO 50:NEXT T
120 NEXT X
130 FOR T=1 TO 500:NEXT T: GOTO 30
READY.
```



321. What gets POKEd can get unPOKEd. To illustrate this, press the RUN/STOP and RESTORE keys, type NEW, and press RETURN to clear the computer's memory. Clear the screen, then type in and enter the program lines shown LISTed here.

322. RUN the program. A horizontal bar will be generated slowly at mid-screen, then it will just as slowly disappear. This nonsense will be repeated for as long as your patience lasts. When it runs out, press the RUN/STOP key and LIST the program, or refer to illustration 321 to discover how the bar got unPOKEd. Lines 40 through 70 generated the bar, consisting of 30 blank spaces (character 160), colored light blue by line 55. Line 60 did nothing but slow the process down. Line 80 kept the bar on the screen for a moment, then the loop formed by lines 90 through 120 took over. Line 100 began to POKE blank spaces (character 160) where the bar of character 160 just ended. Line 105 colored the new blank spaces blue (the normal screen background color), making them invisible, and because of the −1 value of the STEPs in line 90, it continued to POKE those empty blue spaces onto the screen until the bar was gone. Then, after the delay called for by line 130, that line's final GOTO command took the computer back to line 30, and the whole silly process began again.

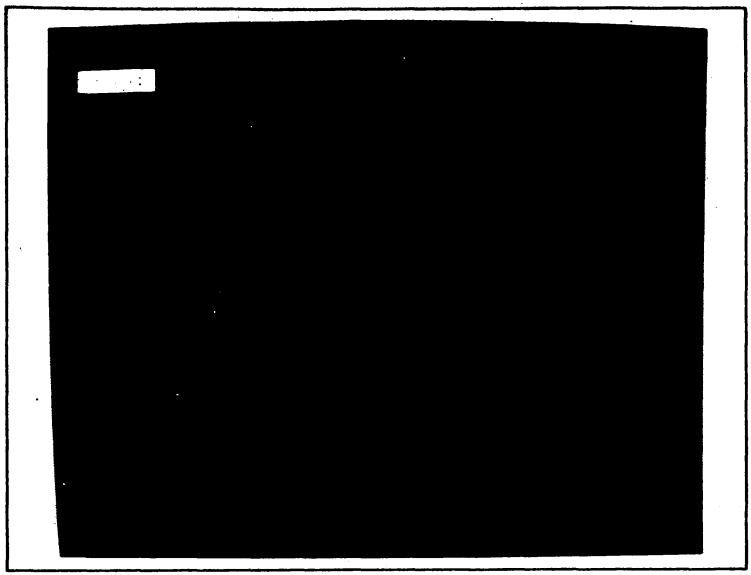




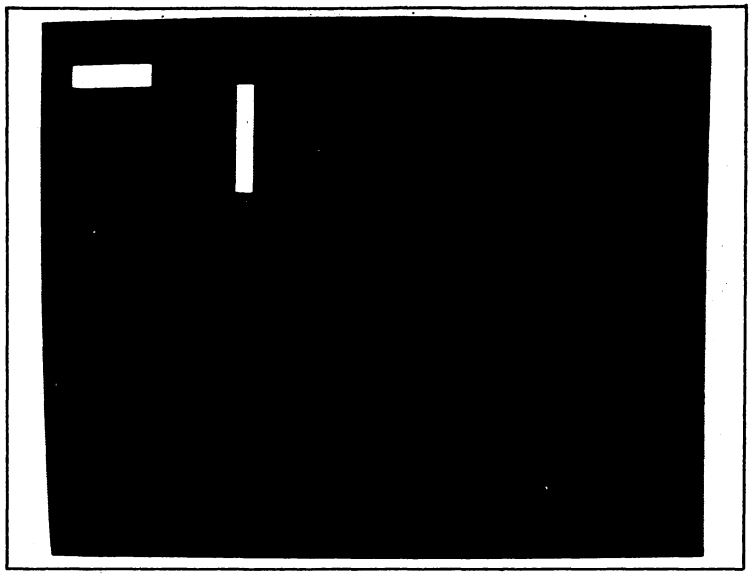
323. If you'd like to draw directly on the TV screen, using the U, D, L, R, and E keys to make your line go up, down, left, right, or be erased, clear memory and the screen, then type in and enter the program shown LISTed in this illustration.

NOTE: Program line 20 makes all the keys on the keyboard into repeat keys. You can use the same POKE, in the immediate mode, any time you power up.

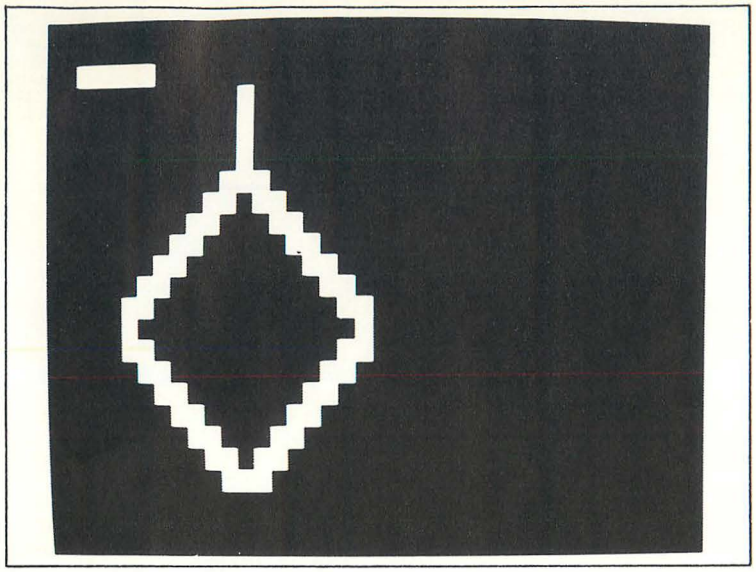
324. When you RUN the program, the screen will go blank and stay that way until you press any of the aforementioned keys except E. Since line 10 places the first character 160 you draw in the second column of the second row, perhaps you ought to start by pressing the R key. The moment you do, a light-blue space (character 160) will appear. Keep the key depressed for a moment and more light-blue spaces will appear, each one a column to the right of the last. For this illustration I have put a line consisting of ten light-blue character 160s on the screen.

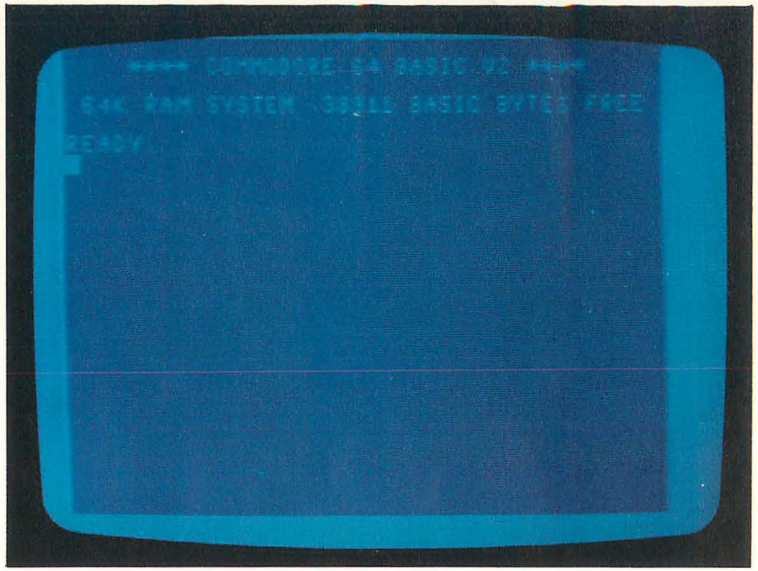


325. Erasing is a little tricky until you get used to it. To remove five of the spaces I just put on the screen I had to first press the E key, then the L key, then the E key, then the L key . . . and so on, until half of the line was erased.



326. To leave a gap of a five-column width and then generate a line vertically downward, five rows long, the procedure is to alternately press the R key and the E key five times each, then press the D key and hold it until five vertical light-blue spaces are generated.



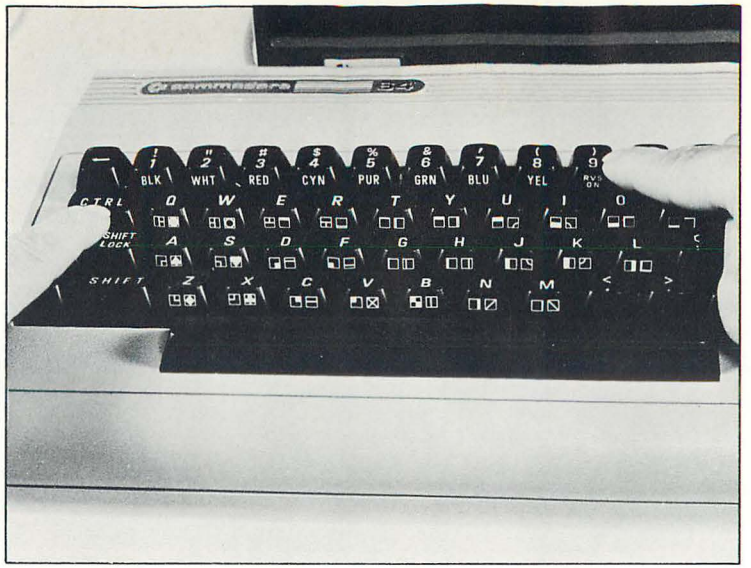


327. To generate diagonals, just alternate presses on the two keys needed to go the way you want to go, as illustrated by this screen.

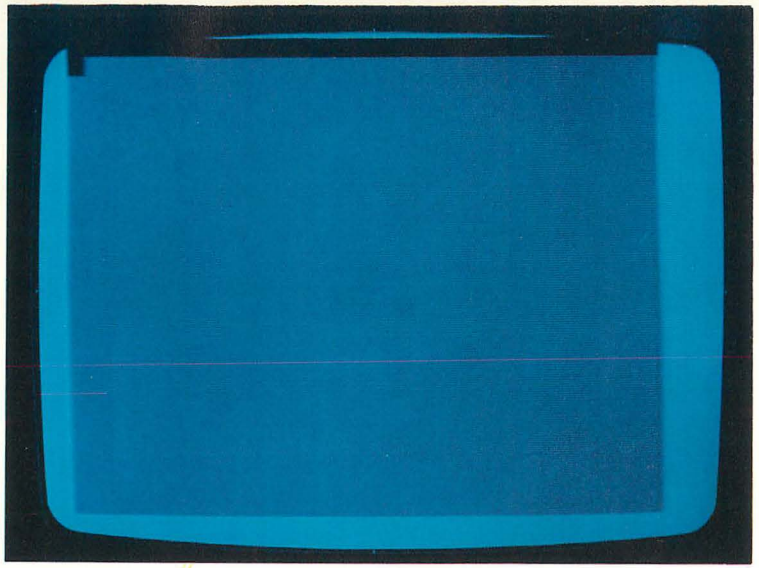
COLOR

328. The first problems you may have to overcome to enjoy the C64's color capabilities are those connected with adjusting your TV set properly. Chances are that when you initially turn on your set it may display a black-and-white picture, if it displays any picture at all. Be sure your set is tuned to either Channel 4 or 3 (depending on which is vacant in your area) and that the RF Modulator switch is set for the same channel. Then you may find that it's necessary to turn off any automatic-tuning or automatic-color-regulating switch and make adjustments to the fine-tuning control of the channel-selector. When you do get a color display, the color of the border, the cursor, and the characters should be light blue, the center of the screen a darker blue. If the colors you have are different, try to adjust the color, tint, brightness, and picture controls (each manufacturer labels their sets' controls differently, so don't be surprised if those labels don't match the ones on your set). If you're patient enough, you should be able to get your screen to look about like this one.

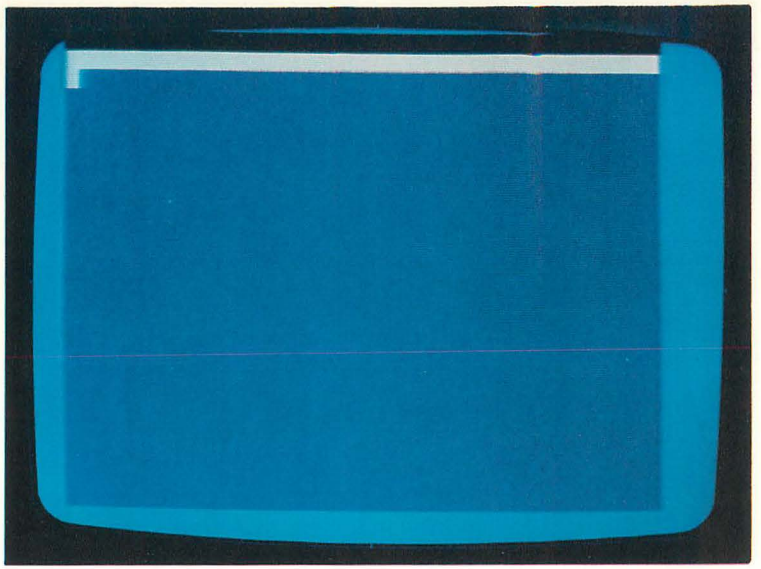
329. That's just the beginning of your set-tuning experience. Now the fun really starts. Clear the screen, then press the CTRL key while you press the 1 key. Note that the 1 key is labeled BLK on its front surface. Your cursor should now be BLacK.



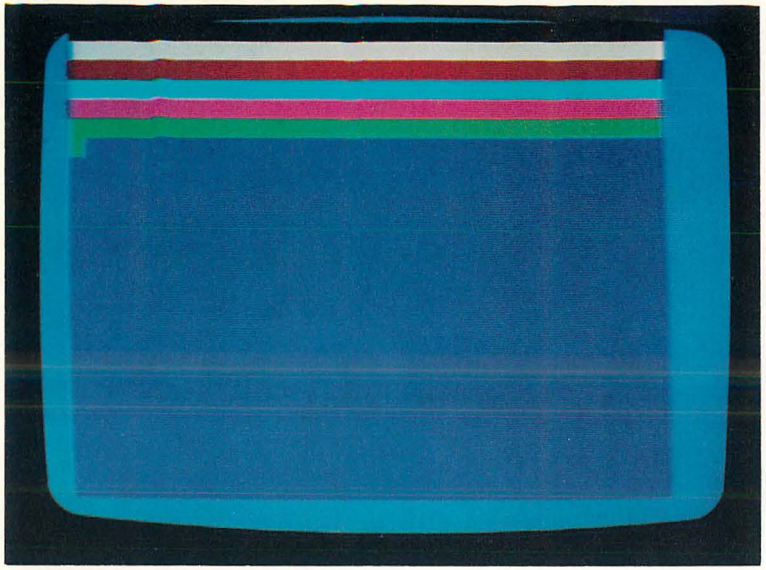
330. Now press the CTRL and 9 (RVS ON) keys.



331. Then press and hold the SPACE bar down until a black bar is generated all the way across the screen and to the first column of the second row, as shown here.

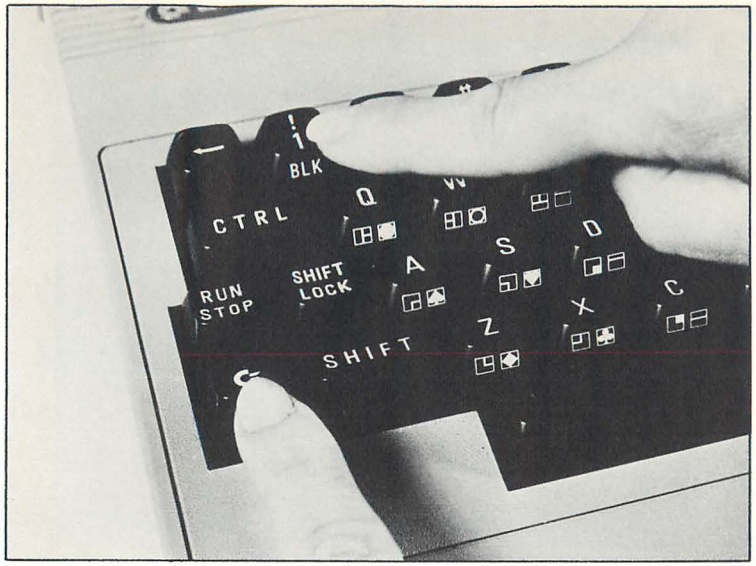


332. Use the CTRL and 2 keys together and you will see the cursor change from black to white. Press and hold the SPACE bar until a white bar is generated completely across the screen and the cursor arrives at the column 0, row 3 position, as shown here.

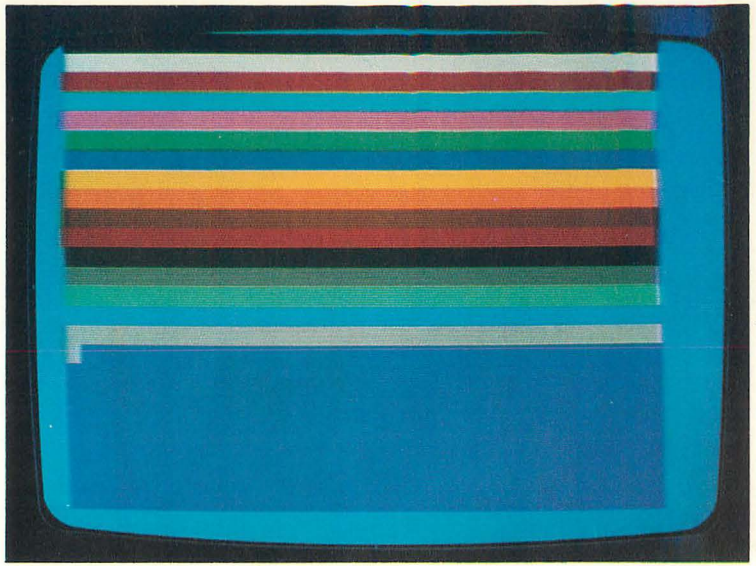


333. Repeat the processes described above (but pressing the 3, 4, 5, and 6 keys with the CTRL key) to generate red, cyan, purple, and green bars, as shown here.

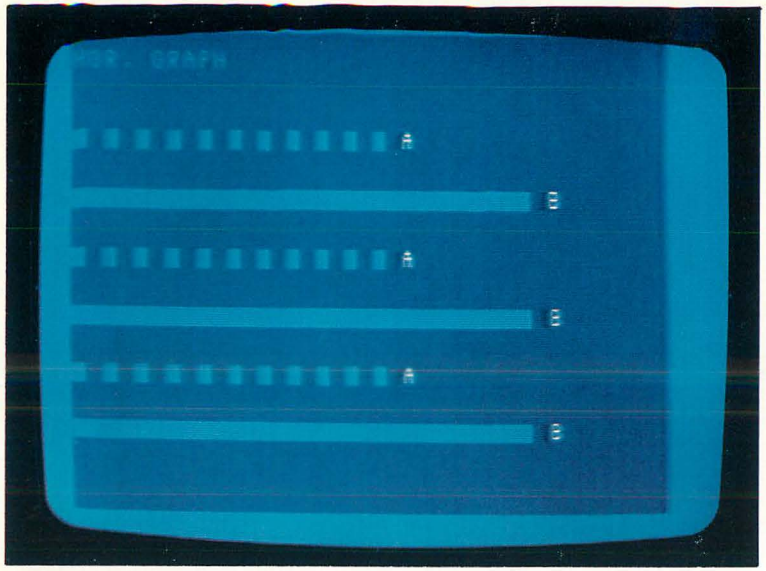
334. Now, try to generate a blue bar with the same procedure (press CTRL along with 7), and note what happens. The cursor becomes the same color (blue) as the screen background and simply disappears. There's not much profit in pursuing an invisible line, so press CTRL and 8 to get a yellow cursor. Move it back to column 1 and down one row to generate a yellow bar across the screen. Now again, adjust your TV controls until the yellow bar looks yellow, rather than orange or some other color, and the rest of the colors look at least somewhat familiar.



335. Press the COMMODORE and the 1 keys simultaneously.



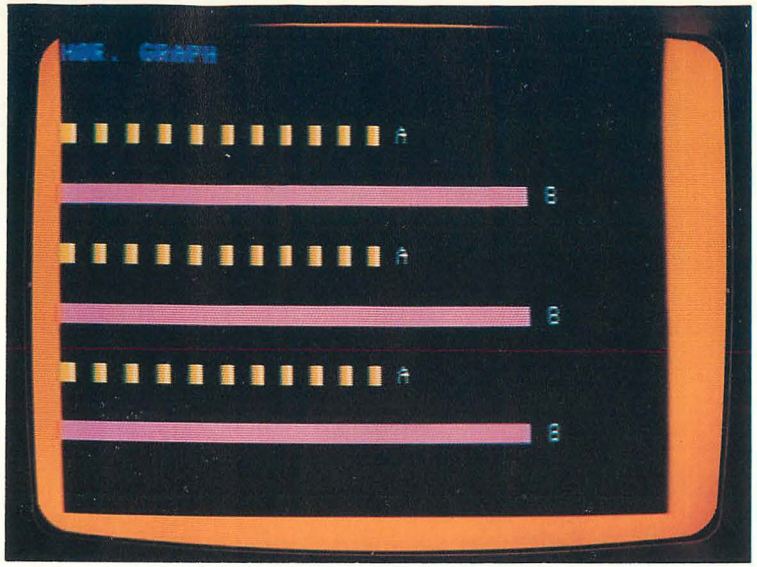
336. That will change the cursor color to orange. Use the SPACE bar to generate an orange bar. Then add all the rest of the keyboard-available colors to the screen with the same procedure. COMMODORE 2 yields brown; COMMODORE 3 provides light red; COMMODORE 4 gives gray 1; COMMODORE 5 offers gray 2; COMMODORE 6 yields light green; COMMODORE 7 provides light blue; COMMODORE 8 gives gray 3. When you get all through, your screen should, hopefully, look like this.

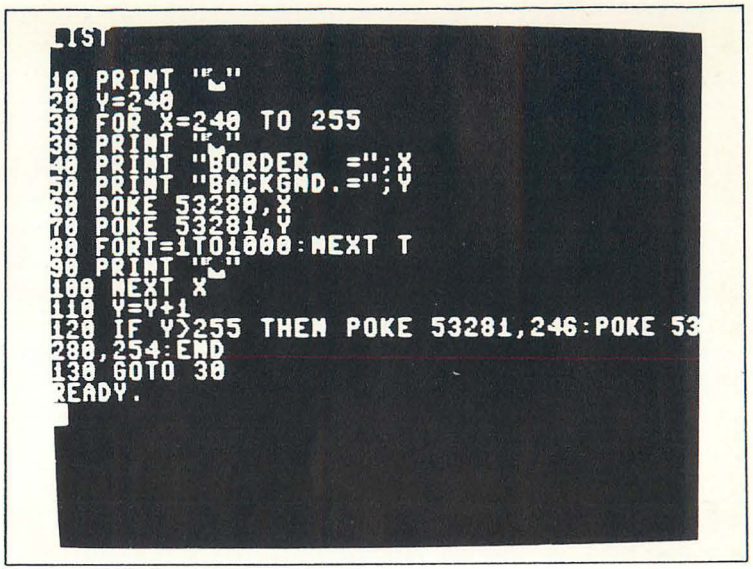


337. Now, with your properly adjusted color TV set, it's time to see how color can be put to work in programming. LOAD the Horizontal Graph program from tape, or type it in again (see illustration 316). RUN the program and note that the bars are all light blue.

```
LIST

10 PRINT "{CLR}"
12 POKE5 3280,248
14 POKE 53281,240
20 PRINT "HOR. GRAPH"
30 Y=5
50 FOR X=0 TO 20 STEP 2
60 POKE 1024+X+40*Y,160
65 POKE 55296+X+40*Y,247
70 NEXT X
80 POKE 1024+X+40*Y,1
85 POKE 55296+X+40*Y,241
100 Y=Y+3
110 FOR X=0 TO 30
130 POKE 1024+X+40*Y,160
135 POKE 55296+X+40*Y,244
140 NEXT X
150 POKE 1024+X+1+40*Y,2
155 POKE 55296+X+1+40*Y,241
170 Y=Y+3
180 IF Y>=21 THEN WAIT 2,3
190 GOTO 50
READY.
```

338. Press RUN/STOP and RESTORE. Clear the screen, then LIST the program and add the following lines: 12 POKE 53280,248 (press RETURN), 14 POKE 53281,240 (press RETURN). Line 12 changes the border area to orange and line 14 changes the screen background to black. Now, edit line 65 so that the number following the comma in the POKE command is 247 (to get a yellow broken bar), and edit line 135 so that the number following the comma in the POKE command is 244 (to get a purple bar).



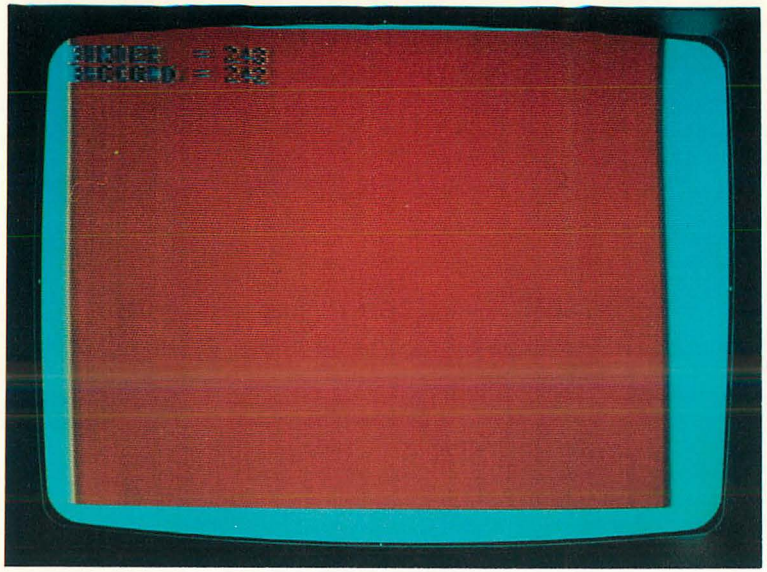


```
LIST

10 PRINT "■"
20 Y=240
30 FOR X=240 TO 255
36 PRINT "■"
40 PRINT "BORDER  =";X
50 PRINT "BACKGND.=";Y
60 POKE 53280,X
70 POKE 53281,Y
80 FORT=1TO1000:NEXT T
90 PRINT "■"
100 NEXT X
110 Y=Y+1
120 IF Y>255 THEN POKE 53281,246:POKE 53
280,254:END
130 GOTO 30
READY.
```

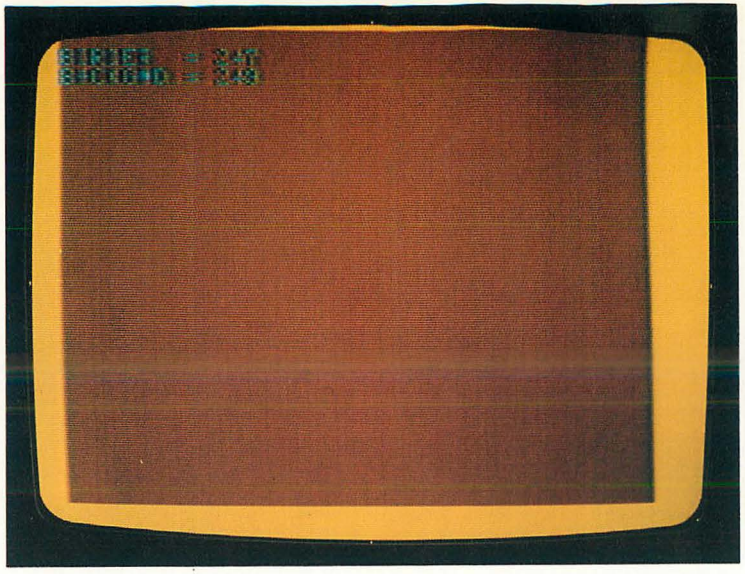
339. Now run the program and your screen should look like this. Now you are getting the idea. Color is available and it can convey a huge amount of information if used correctly. With the yellow and purple graph bars shown there was really no need to use a broken line to differentiate between the bars labeled A and those labeled B.

NOTE: When you POKE color into characters or symbols, follow the comma with 239 plus the number shown on the key top of the color you want displayed. For example, to get yellow bars I used 247 (239 plus the 8 at the top of the YEL key). Other colors available to you are 239 plus 9 for orange; 10 for brown; 11 for light red; 12 for gray 1; 13 for gray 2; 14 for light green; 15 for light blue (the default color of the cursor); and 16 for gray 3.

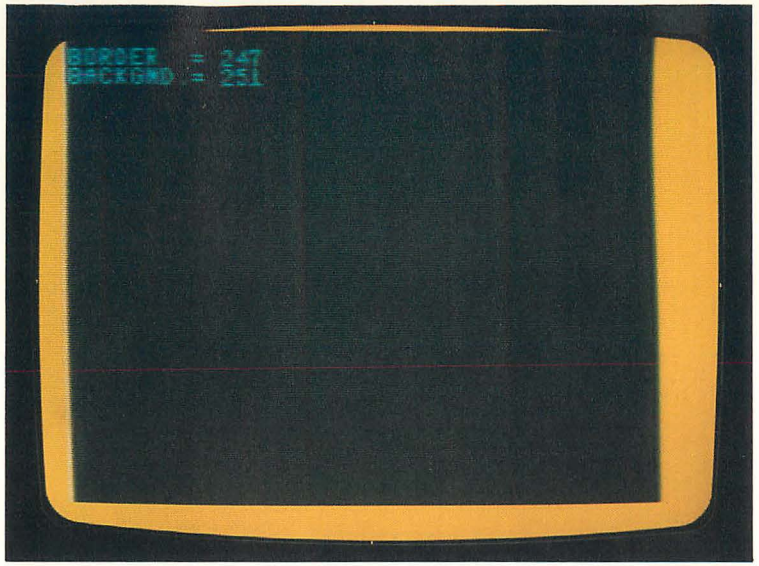
340. To see an incredible display of the border-and-screen color combinations the C64 can provide, type in and enter the program shown here.



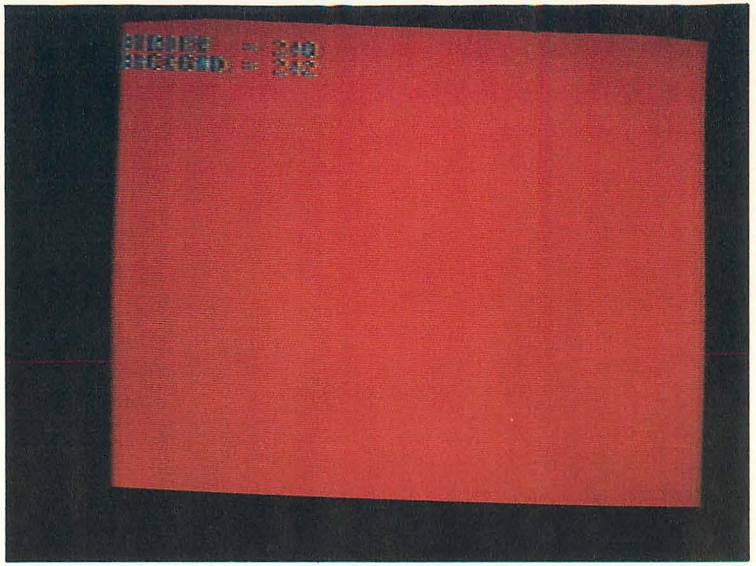
341. RUN the program. Lines 60 and 70 POKE in the border and background colors, respectively. You can press the RUN/STOP key to keep any combination you like on the screen for a while, then when you are ready to go on type in: CONT (and press RETURN). With a 243 border and a 242 background the screen looks like this.



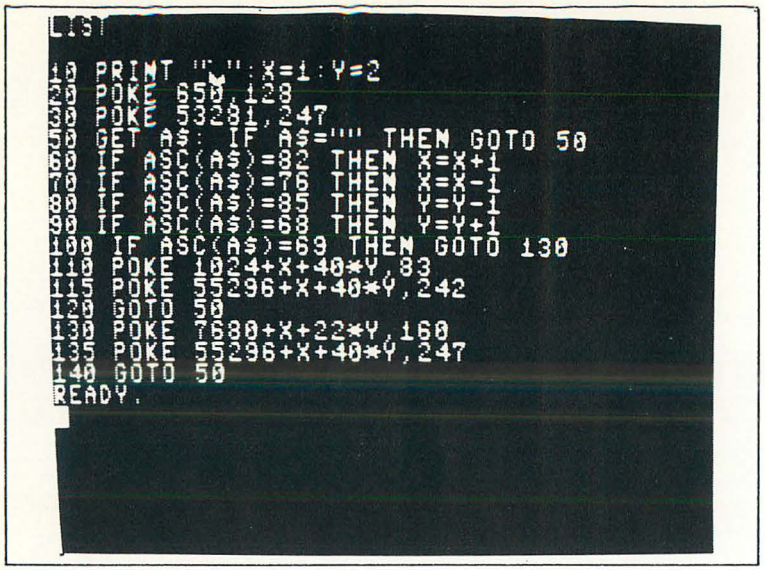
342. And this is what a 247 border and a 249 background looks like.



343. The 247 border and 251 background combination shown here gives more clarity to the text display. Gray and black backgrounds show text best.

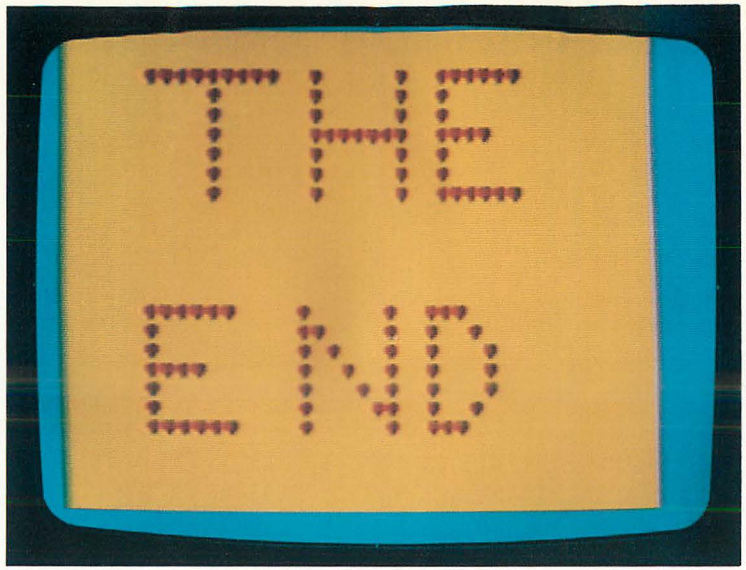


344. You can minimize the border by using black for its color, as with the 240/242 combination shown here.



```
LIST

10 PRINT "■";X=1:Y=2
20 POKE 650,128
30 POKE 53281,247
50 GET A$: IF A$="" THEN GOTO 50
60 IF ASC(A$)=82 THEN X=X+1
70 IF ASC(A$)=76 THEN X=X-1
80 IF ASC(A$)=85 THEN Y=Y-1
90 IF ASC(A$)=68 THEN Y=Y+1
100 IF ASC(A$)=69 THEN GOTO 130
110 POKE 1024+X+40*Y,83
115 POKE 55296+X+40*Y,242
120 GOTO 50
130 POKE 7680+X+22*Y,160
135 POKE 55296+X+40*Y,247
140 GOTO 50
READY.
```

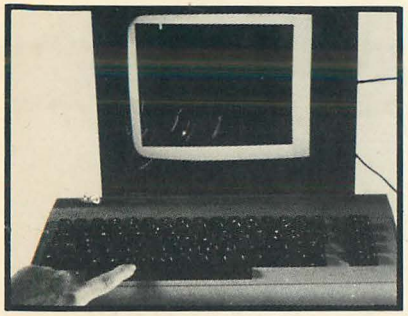
345. To draw the final screen, I added one line and changed two others in the program shown in illustration 323. I added: 30 POKE 53281,247 to change the background to yellow. I followed the comma in line 110 with an 83 to get a heart shape. I followed the comma in line 115 with 242 to color the hearts red, and the comma in line 135 with 247 to color the erasing blank spaces yellow (the same as the background color).

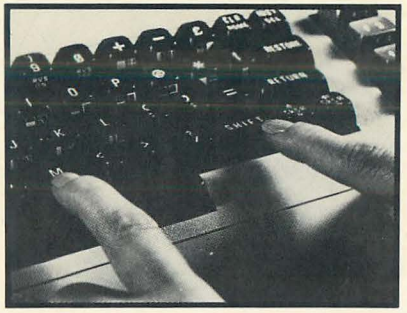


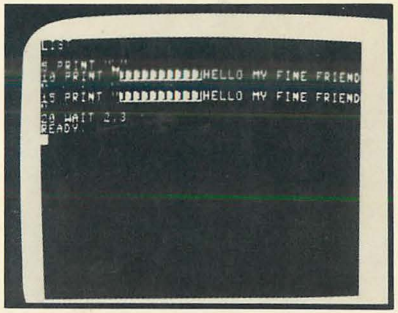
346. Then I used the same R, L, U, D, and E keys to say . . .

The Commodore 64™ ILLUSTRATED

Bob Nadler







When you get your Commodore 64 home, your first question is bound to be "Now what?" Here is the answer. This illustrated introduction is a truly unique approach. It explains every operating step from the start—unpacking the machine and setting up the hardware.

Easy programming exercises help you get a feel for the Commodore 64, and here's where the unique illustration format comes in: each programming exercise is illustrated with photos of the screen, so you can check your results against correct screen presentation. With this visual guidance, you can quickly progress to more advanced techniques and routines.

HAYDEN BOOK COMPANY
a division of Hayden Publishing Company, Inc.
Hasbrouck Heights, New Jersey

ISBN 0-8104-6453-5