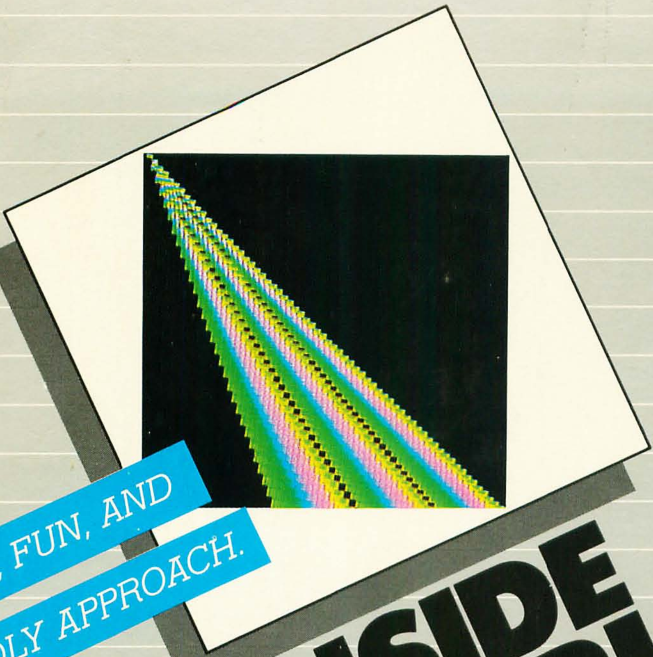


A FAST, FUN, AND FRIENDLY APPROACH.

INSIDE ATARI BASIC

by Bill Carris

ATARI®

Inside ATARI BASIC

Inside ATARI® BASIC

A fast, fun, and friendly approach

WILLIAM CARRIS

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

Library of Congress Cataloging in Publication Data

Carris, William.
Inside ATARI BASIC.

Includes index.
1. Atari computer—Programming. 2. BASIC (Computer program language) I. Title.
QA76.8.A82C37 1982 001.64'24 82-13334
ISBN 0-8359-3083-4
ISBN 0-8359-3082-3 (pbk.)

10 9 8 7 6 5 4 3 2

Printed in the United States of America

This book is dedicated to

NAME$*

for the varying degrees of help you gave me with no STRINGS attached

*NAME$ is further defined on page 42.

Contents

Introduction

The purpose of this book is to introduce you to the key concepts of BASIC programming while inflicting as little pain as possible.

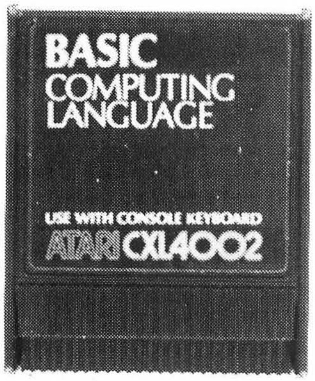
This is a workbook to be used along with an ATARI® Home Computer. When you complete this book you should not only have a beginner's understanding of the BASIC language, but also an introduction to ATARI Computer GRAPHICS and SOUND capabilities as well as other interesting ATARI Computer features.

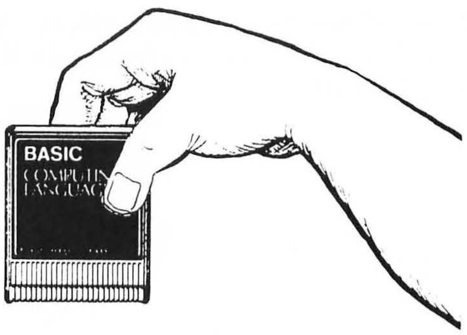
I have intentionally avoided flow charts, many unnecessary details, most exceptions to rules, and other traditional computer education approaches because, while they are necessary evils for the professional programmer, they can also have a tendency to alienate the beginner.

Professional programming requires exacting and tedious attention to detail, but learning how to do elementary programs on a home computer is more of an adventurous and enjoyable exploration of the man-machine relationship. It is simply "No-Big-Deal" if you make a mistake while learning how to operate a home computer. Those who learn from their mistakes and master this book should be chomping at the bit for more advanced information and methods. A few may even develop the dedication to become professionals. But, first things first—Welcome to ATARI BASIC.

This book is designed to be used with either the ATARI 400 Home Computer™ or The ATARI 800 Home Computer™

along with the
ATARI BASIC Cartridge.

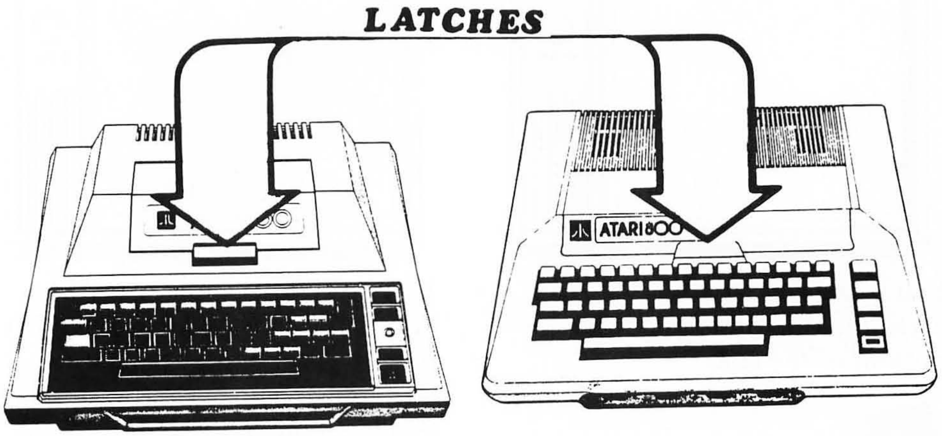




WHAT IS BASIC?
BASIC is a very popular computer language, which is easier to learn than most.

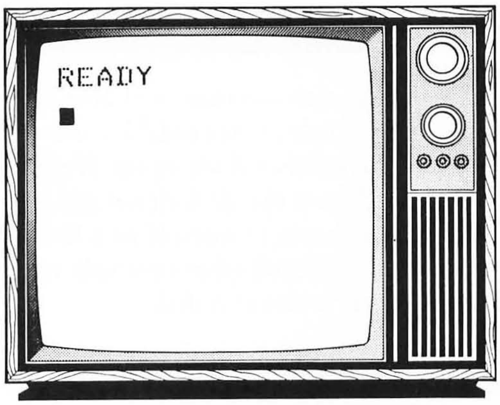
Part 1
Basic Programming

To insert the BASIC cartridge, first open the cartridge door by pulling the latch down and toward you.

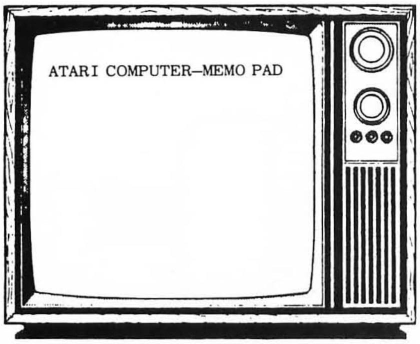


Next, with the label facing you, push the cartridge slowly but firmly down into the cartridge slot.

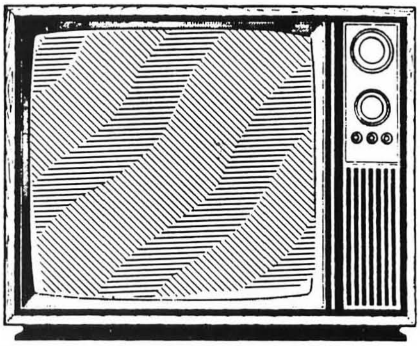
When you close the cartridge door, *turn the computer on* and your TV should look like this...



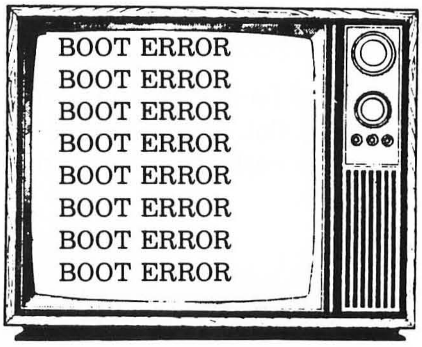
If you get any of these,



You might not have pressed the cartridge down firmly to the bottom of the slot.



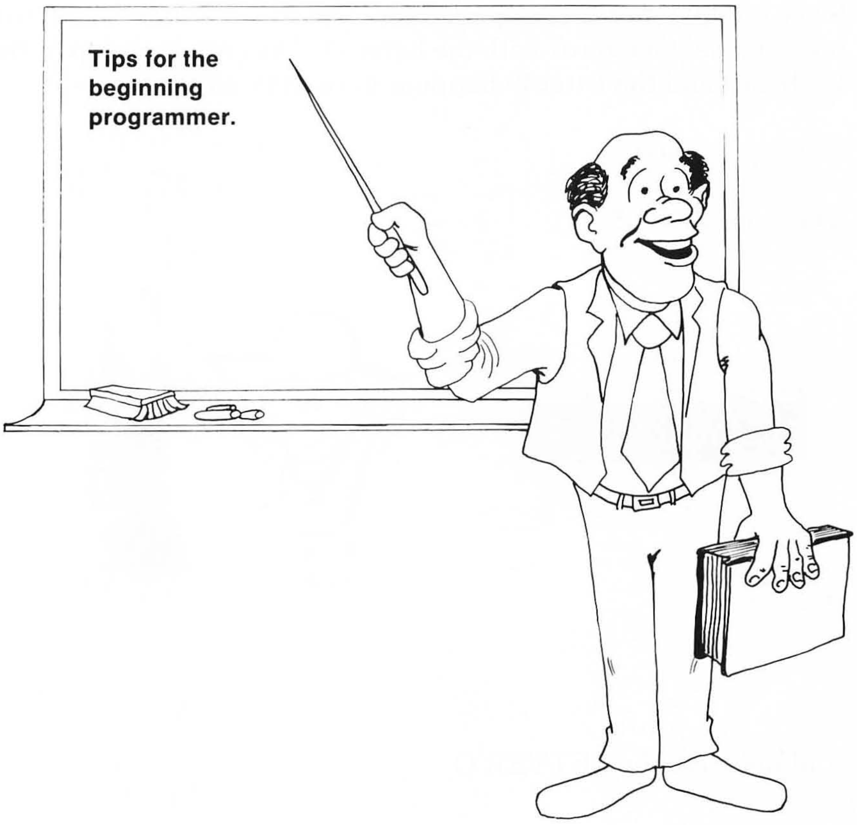
You probably do not have your computer's 2-3 switch and the TV turned to the same channel. Try switching channels and also check your computer switchbox on the back of the TV to make sure it's switched to "computer."



```
BOOT ERROR
BOOT ERROR
BOOT ERROR
BOOT ERROR
BOOT ERROR
BOOT ERROR
BOOT ERROR
BOOT ERROR
```

You probably have a disk drive connected and something elementary is wrong. For example, you could be using the wrong diskette or you might have left the disk drive door open. If you are just learning BASIC, you probably should turn the disk drive off. Turn the computer off, then on again. After introducing yourself to a little BASIC you can use your Disk Drive Manual to familiarize yourself with basic disk instructions. For starters work without a disk.

NOTE: For any other problems or complete system set-up instructions, consult your owner's guide.

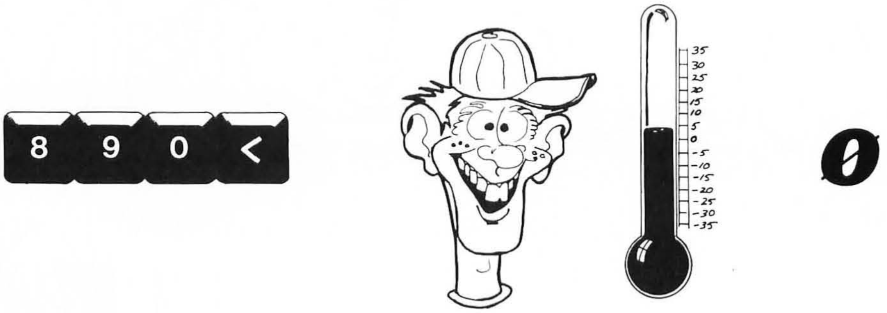


- Fear not! Nothing you can type will damage the computer or your TV.
- Making ERRORS is a natural part of learning to program. Everyone makes errors so you might as well get used to the idea.
- It is normal for your computer to use UPPER-CASE letters. It will not accept instructions written in lower-case letters.
- Also, when you get to the edge of the screen, just keep typing; the computer will automatically "wrap around" to the next line.

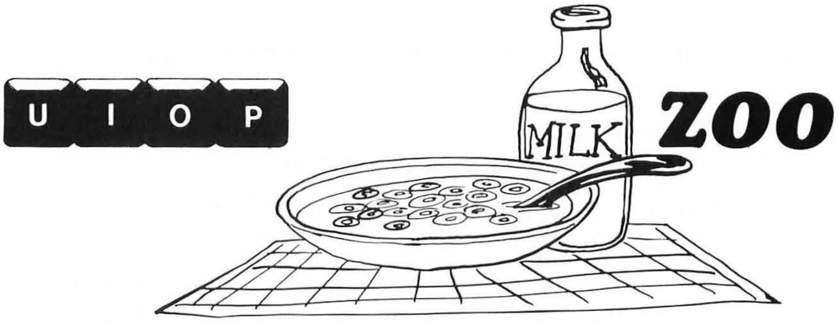
Since computers are very particular creatures, it is important that you do not confuse your zeros with the letter O. The zero is on top of the keyboard and the letter O happens to be very close to it.

So keep in mind...

These are ZEROS



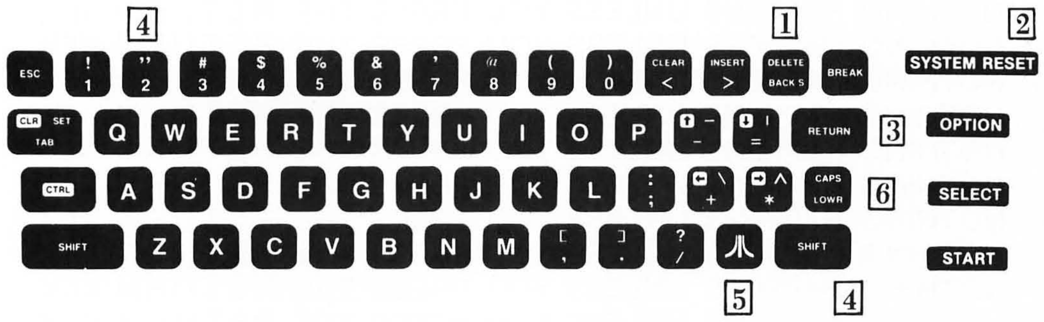
And these are the LETTER O



If you look closely at a zero on the TV screen, you will see that it has an **S** shape in the center and the letter O does not. Most computer printers and many people listing programs cross the zero (Ø) to avoid confusion.

Key Concepts

Although your computer's keyboard is very much like a typewriter's keyboard, it has some special features. You need to learn about some of these features right away and others later.



Here are a few keys you should know about right away.
Try them out...

1 DELETE BACK S This key will rub out any letter you type by accident. Type a few letters then "BACKSPACE" over them to get the idea.

2 SYSTEM RESET This key will help if you get confused with what is happening on the screen. Until you get familiar with the keyboard, you might accidentally create a screen full of error messages or characters you don't understand. To clear up a confusing screen and start from scratch, press SYSTEM RESET.

3 When you press RETURN you are telling the computer that you are entering your information or answer.

Just a brief reminder:

NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY

```
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
NOTHING HAPPENS UNLESS YOU PRESS THE RETURN KEY
```

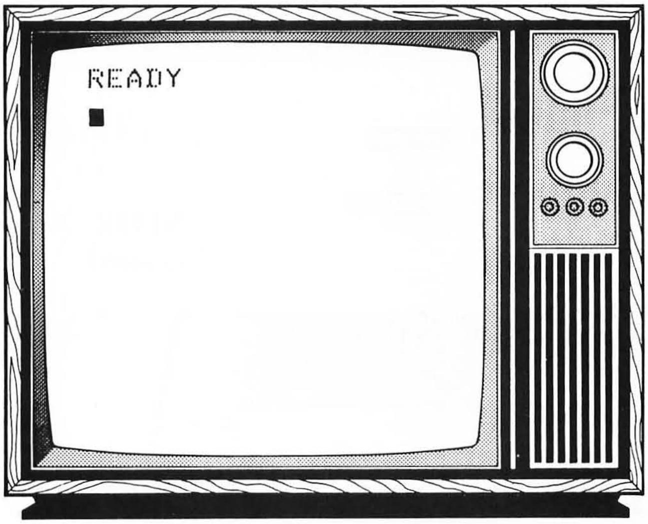
$\boxed{4}$ You will also be introduced to some *two-key combinations* which will require you to hold one key while pressing another. The first is the combination needed to print quotes. HOLD DOWN the

SHIFT key and PRESS the " 2 key at the same time.

$\boxed{5}$ The key puts you into INVERSE VIDEO . Press it and type something. Press it again and you're back to normal.

$\boxed{6}$ The CAPS LOWR key gives you lower case. Press it and type some lower-case letters. *To get out of lower case, you have to hold* SHIFT *and press* CAPS LOWR .

NOTE: Inverse video and lower-case letters will be handy later, but as mentioned before, you should work in standard UPPER CASE when learning.



When your computer says READY...

What your computer is ready to do is take your orders.

You will soon find, in fact, that the trick to writing BASIC programs is learning what orders to give your computer.

One of the most common instructions you will use is obtained by typing the letters...

PRINT

which tells the computer that you want to print something on the screen.

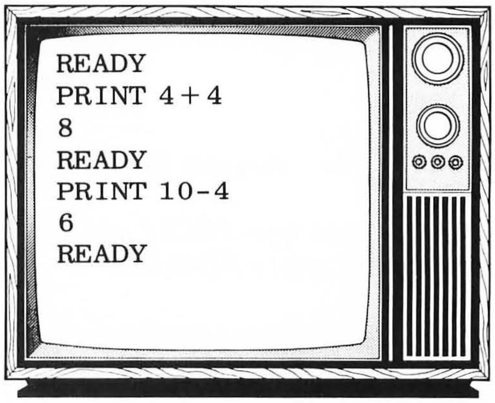
When you type PRINT and press the RETURN *key, you're telling the computer to print a blank line.*

Performing Simple Calculations

```
PRINT 4+4 RETURN

PRINT 10-4 RETURN
```

NOTE: No equal signs are needed.

The screen should look like this.



```
READY
PRINT 4+4
8
READY
PRINT 10-4
6
READY
```

Doing Multiplication and Division

* * * GO FORTH AND MULTIPLY * * *

The symbol for multiplication with your computer is the asterisk * which is made by pressing the ^ * key.

These will not work...

```
PRINT 5 X 7
PRINT (5) (7)
PRINT 5•7
```

This will work...

```
PRINT 5 *7  RETURN
```

will give you an answer of 35.

/ / / DIVIDE AND CONQUER / / /

The symbol for division with the computer is the slash / produced by pressing the ? / key.

```
PRINT 99/3  RETURN
```

will give you an answer of 33.

* * * * * * IS THE SYMBOL FOR MULTIPLICATION

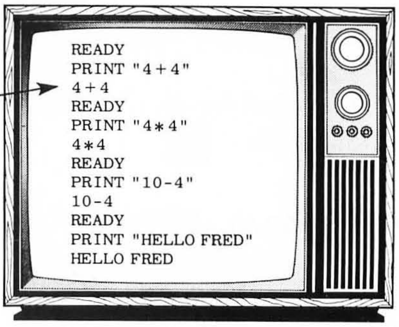
/ / / / / / / / / IS THE SYMBOL FOR DIVISION

Printing Letters, Words, and Characters

```
PRINT "4+ 4" RETURN

PRINT "4*4"  RETURN

PRINT "18-4" RETURN

PRINT "HELLO FRED" RETURN
```

Anything inside of quotes " " will be printed directly.

The result is printed without quotes. Trying to print quotes on the screen is possible, but will give the beginner a mental hernia.



```
READY
PRINT "4+4"
4+4
READY
PRINT "4*4"
4*4
READY
PRINT "10-4"
10-4
READY
PRINT "HELLO FRED"
HELLO FRED
```

NOTE: You can abbreviate PRINT using a ? but beginners probably will understand their program listing better if they spell out PRINT in full.

Direct Mode and Programming Mode

With your computer, you can either DO IT NOW or DO IT LATER.

Direct Mode

Instructions such as PRINT "HELLO" RETURN are executed as soon as RETURN is pressed so they are said to be in the DIRECT or IMMEDIATE mode.

The DIRECT mode is useful for getting quick answers and experimenting to see how the computer will react to a certain instruction.

Programming Mode

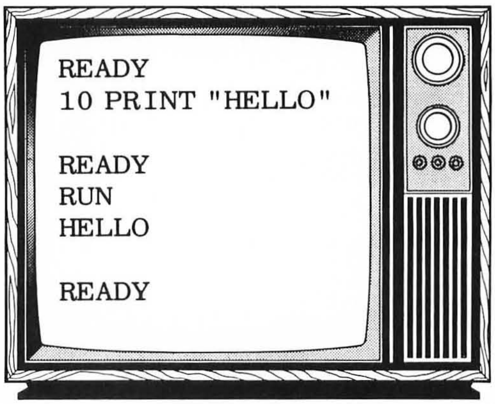
A program is an instruction or a series of instructions which are not executed until you type RUN RETURN .

Every time you type RUN, the program is executed again.

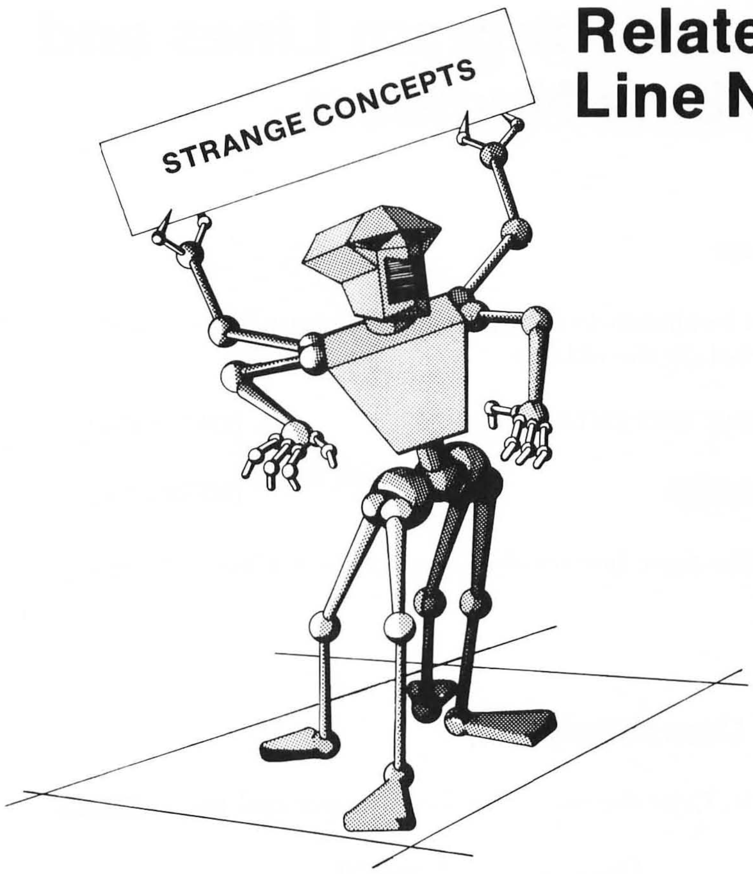
This Is a Program

```
10 PRINT "HELLO"     RETURN
RUN                  RETURN
```

Notice the line number 10 before the print command.



```
READY
10 PRINT "HELLO"

READY
RUN
HELLO

READY
```



Related to Line Numbers

Each group of instructions in a program is preceded by a line number. Unless instructed otherwise, the computer starts executing the program at the lowest line number and works toward the highest.

```
10 PRINT "HELLO"
20
30
```

Instead of numbering their lines 1, 2, 3, etc., most people like to leave room for inserting new lines later so they number them something like 10, 20, 30, etc.

More About Program Lines and Line Numbers

Replacing Lines

The easiest way for beginners to change a line is to type it over again. The new line will replace the old line.

```
10 PRINT "HOW ARE YOU THIS FINE DAY"          (OLD LINE)

10 PRINT "HI" RETURN                          (NEW LINE)
```

As long as you use the same line number, even a shorter line will replace the original one.

Erasing Lines Completely

There's nothing to it. Type the unwanted line number and press RETURN

```
10 RETURN                  Does away with line 10.
```

The 3 Line Maximum

When writing programs, you need a new line number for every 3 lines printed on the screen.

```
10 PRINT "YOU CANNOT PRINT MORE THAN 3
LINES OF INFORMATION ON THE SCREEN WIT
HOUT ADDING A NEW LINE NUMBER"
20 PRINT "THAT IS WHY LINE 20 WAS ADDED
"
```

You've Had it Too Easy so Far

You will find that it soon becomes second nature to press the RETURN key after making entries.

In order to make program listings and instructions less complicated for you to read, from this point forward, RETURN key instructions usually won't be shown.

THIS IS NOT A COLD-TURKEY WITHDRAWAL, HOWEVER. HERE ARE A FEW MORE RETURNS JUST TO TIDE YOU OVER.

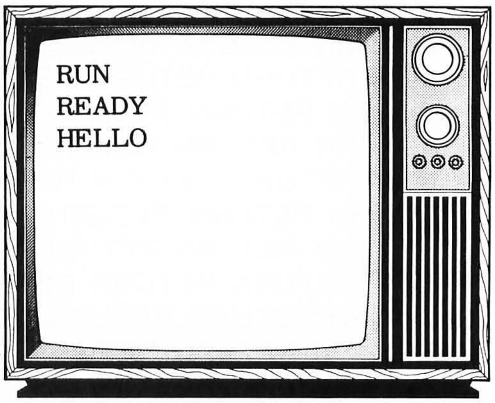
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN
RETURN RETURN RETURN RETURN RETURN RETURN

REMark Statements

THESE HELP YOU AND OTHER PEOPLE UNDERSTAND YOUR PROGRAM.

```
10 PRINT "HELLO"
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
```

If you type a line number and follow it with REMARK or REM, you can then type three lines of information which won't affect the program's operation.

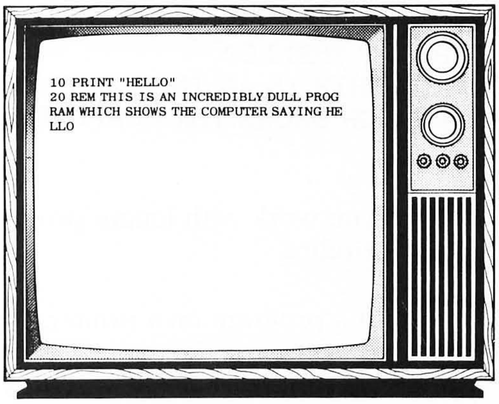


```
RUN
READY
HELLO
```

LIST or L.

```
LIST
```

To list the entire program, simply type LIST.



```
10 PRINT "HELLO"
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
```

Listing Parts of Programs

When you start working with longer programs, there will be times when you want to list just a single line or only a few lines of the program.

To list a single line of a program, type LIST and add the line number before pressing RETURN .

LIST 20 Only line 20 will appear on the screen.

```
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
```

In order to list a range of program lines, type...

LIST 10, 20

The first program line you want goes here.

Don't forget to use a comma.

The last line you want to see goes here.

THIS WILL LIST LINES 10 AND 20 AND ANY LINES THAT WOULD HAVE BEEN BETWEEN THEM.

```
10 PRINT "HELLO"
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
```

Later, as you work with longer programs, you will appreciate this partial listing flexibility.

To list a program on a printer, type...

```
LIST "P:
```

Use LPRINT to Print on Paper

```
10 LPRINT "HELLO"
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
```

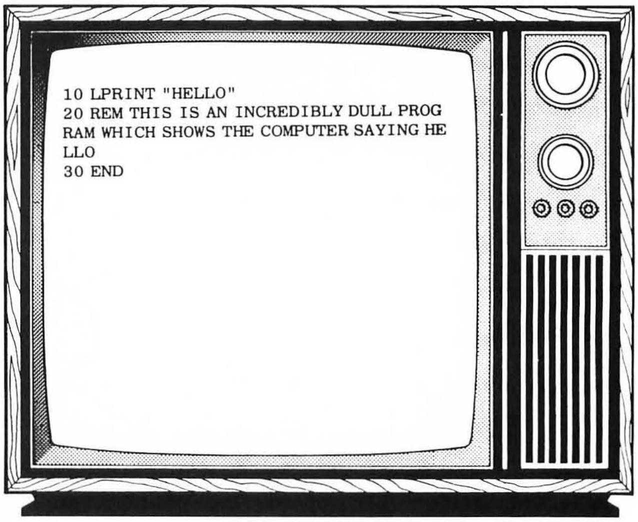
Use an LPRINT (LINE PRINT) command for obtaining a hard copy.

NOTE: You cannot use LPRINT without a printer. If you try to do so you will get an ERROR message #138, which means you printed without a printer. What could be simpler?

END

```
30 END
```

Some types of programs require an END statement to be placed where you want the computer to stop computing. It is not always necessary on the ATARI Computer, but it is good to be aware of it so you can use it later, when you need it.



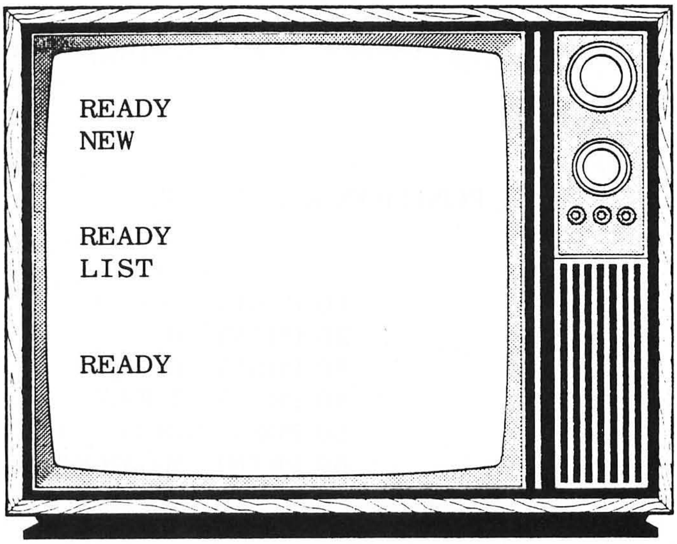
```
10 LPRINT "HELLO"
20 REM THIS IS AN INCREDIBLY DULL PROG
RAM WHICH SHOWS THE COMPUTER SAYING HE
LLO
30 END
```

NEW

```
NEW
```

NEW will erase your program. No second chances.

As you can see, after you type LIST, no program appears.



```
READY
NEW


READY
LIST


READY
```

NOTE: *Turning off your computer or opening the cartridge door will also erase your program.* Turning off the TV will not affect it, however.

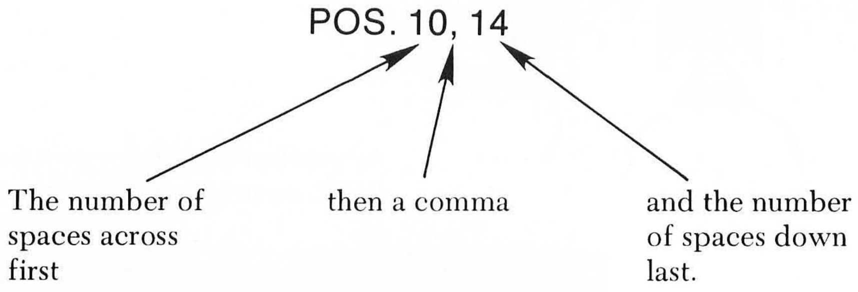
POSITION or POS.

POSITION tells the computer where to start printing on the screen.

Imagine that your screen is divided into 40 blocks across and 24 blocks down.

If you don't use a POSITION statement before a PRINT, the computer automatically indents two spaces and starts printing on the line below the one it just finished.

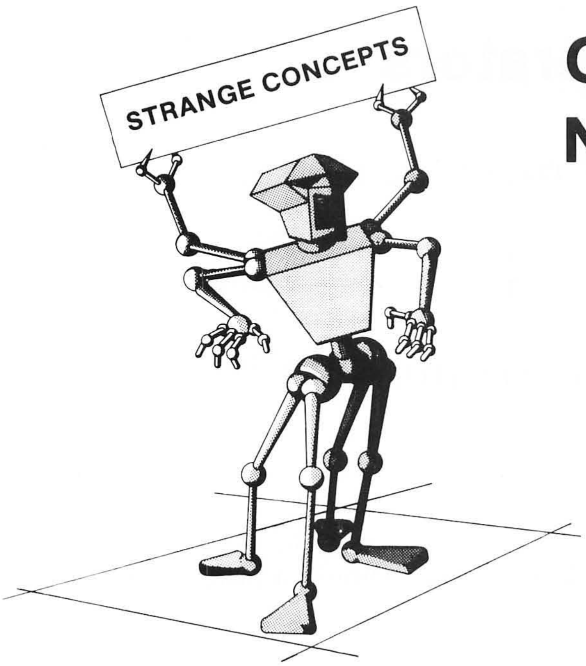
With POSITION, always give


POS. 10, 14
The number of spaces across first
then a comma
and the number of spaces down last.

In a program, POSITION is used as follows...

```
10 POSITION 0,0
20 PRINT "HI"
30 POSITION 35,0
40 PRINT "THERE"
50 POSITION 16,11
60 PRINT "MIDDLE"
```

Before you run this, press SYSTEM RESET to clear the screen. Later you will learn more efficient ways to get unwanted words out of your display.

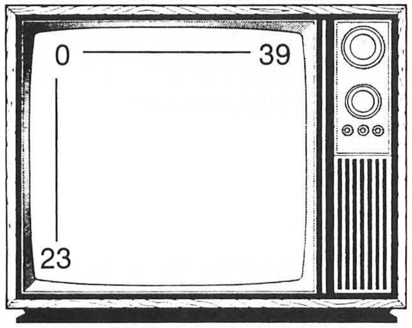


Computer Numbering

For technical reasons, computers' internal numbering systems use 0 as the first number instead of 1.

For example, if a computer had 12 graphics levels, they would be numbered 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11.

As used with POSITION...



40 spaces across are numbered 0 through 39.
24 spaces down are numbered 0 through 23.

Operators

English teachers are all dishonest. They told you that these...

: , ; were punctuation marks!

THEY'RE OPERATORS

The Colon :

You can use a colon to put more than one instruction in a program line. This enables you to:

- Save Time.
- Save Memory.
- Organize Your Program Better.
- Occasionally Screw Things Up Incredibly.

Type in NEW RETURN then the following one-line program. Notice that you have condensed three instructions into one line.

```
10 PRINT "I AM ABOUT TO PRINT IN THE M
IDDLE OF THE SCREEN": POSITION 6,11:PRI
NT "MY DESTINY HAS BEEN FULFILLED"
```

Here's a common pitfall:

```
10 REM THIS IS THE FIRST LINE OF THE S
TOCK PROGRAM: PRINT "STOCK AND BOND ANAL
YSIS"
```

Nothing following a REM statement is recognized. When learning BASIC, keep REMs isolated on separate lines.

The Comma ,

The comma helps you organize your output into columns. When you put commas in between printed items, the computer will put the words or numbers into columns which are ten spaces apart.

```
10 PRINT "ATARI","TEA","AND","ORANGE"
20 PRINT "1","2","3","4"
```

```
ATARI     TEA       AND       ORANGE
1         2         3         4
```

NOTE: Information on how to change the spacing between columns can be found in the section on PEEKS and POKES.

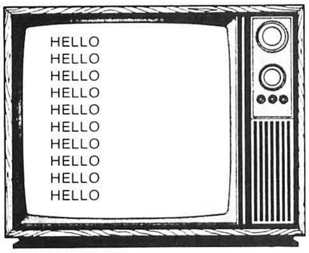
The Semicolon ;

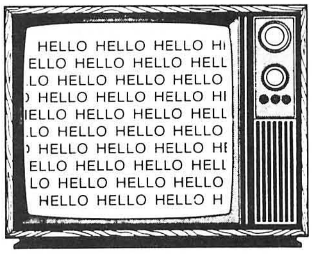
This operator allows us to join things together.

Try this program first without and then with the SEMICOLON.

```
10 PRINT "HELLO ";
20 GOTO 10
```

NOTE: Pressing the BREAK key will stop BASIC programs like this. Typing CONT RETURN will make the program CONTINUE again.





```
10 PRINT "GOLDPRICE= "; "$385.98"
WILL PRODUCE. . .
GOLDPRICE=$385.98
```

Why you do such a thing will become apparent soon.

Error Messages—The Best Friends You'll Ever Hate

ATARI Computers give you two very useful types of ERROR MESSAGES.

The first is displayed when you type a line without proper punctuation, legal operators, or anything else that doesn't make sense to the computer.

Here are some common slips...

PRIMT "HELLO"

produces
ERROR—PRIMT $\boxed{\text{“}}$HELLO"

on clear inspection, you will notice that PRINT is spelled wrong.

When you get this type of error message, the computer puts a marker on top of the first area in question not the error itself. After some practice you'll spot what the computer doesn't like.

PRINT HELLO"

produces ERROR—PRINT HELLO$\boxed{\text{”}}$

PRINT is spelled correctly, but there's no quotation mark before the H in HELLO.

print "hello"

produces ERROR—$\boxed{\text{p}}$rint "hello"

Remember you can't use lower case letters for commands.

POS.2.5 : PRINT “HELLO”	
ERROR—POS.2.5$\boxed{:}$PRINT “HELLO”	How unforgiving, you used a period (.) instead of a comma between the 2 and 5.
POS 2,5:PRINT “HELLO”	
ERROR—POS$\boxed{2}$,5:PRINT “HELLO”	RATS! You got the comma right but left out the period (.) after POS.
10 “PRINT HELLO”	
10 ERROR—$\boxed{“}$PRINT HELLO”	A Freudian slip; you put the quotation mark in front of the P instead of the H.

Program Error Messages

Assuming you have typed in your program, but there's something illogical about it, you will then get another type of ERROR MESSAGE when you try to RUN the program.

This error message will give you an ERROR type specified by a number and then indicate the line at which the computer had trouble executing your program.

When you get an error message with a line number, such as...

```
10 LPRINT "HELLO"
RUN
ERROR 138 AT LINE 10
```

First look at the line specified for obvious mistakes.

Then look up the type of ERROR. The more common errors are listed in the back of this book. If you don't find it there, consult your ATARI BASIC Reference Manual for a full listing.

In the above case, an ERROR 138 indicates a "Device Time Out." Someone used an LPRINT command without turning on their printer.

By the way, this may seem like a lot of trouble, but most computers give you a lot less help finding problems!

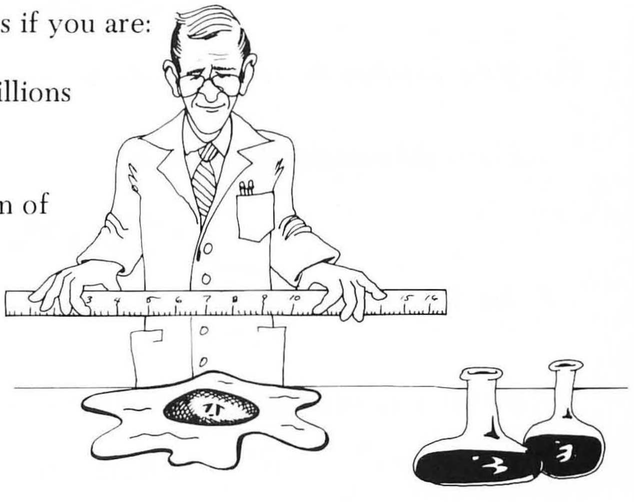
Three Math Tips

Math Tip #1

Computers use abbreviated forms of very large and very small numbers. The easiest way for a beginner to deal with this is to:

Avoid Using **very large** or
very small numbers!

You cannot, of course, do this if you are:



1. An oil-rich sheik with billions to keep track of.
2. A microbiologist with .000000007TH of a micron of cell growth to measure
3. A masochist.

Assuming you are #3, here is a brief summary of how to translate this abbreviated notation into ordinary human notation. *VERY LARGE* abbreviated numbers have:

1. An "E"
2. A "+" sign
3. A number after the plus (+) sign

3.7E + 11

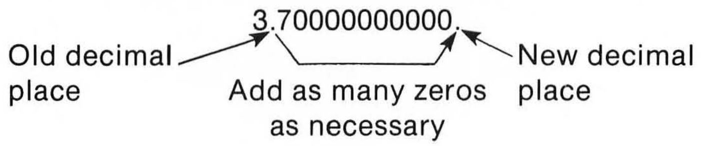
To convert a *VERY LARGE* abbreviated number such as:

3.7E + 11

1. Write the number up to the E

3.7

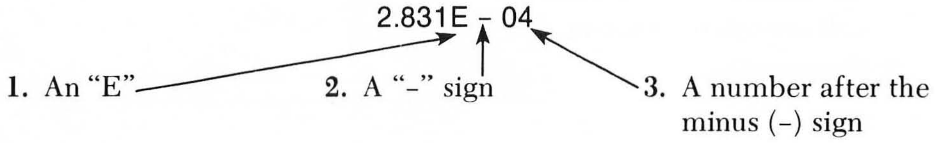
2. Start at the decimal point and count, to the right, the number of places indicated by the number after the plus sign (in this case, "11").



370000000000.

This is the number the computer abbreviated as 3.7E + 11

VERY SMALL abbreviated numbers have:



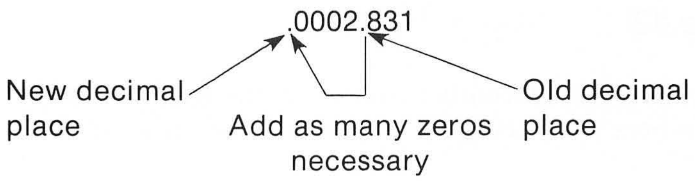
To convert a VERY SMALL abbreviated number such as:

2.831E - 04

1. Write the number up to the E

2.831

2. Start at the decimal place and count, to the *left*, the number of places indicated by the number after the minus sign (in this case, "4").



.0002831

This is the number the computer abbreviated as 2.831E - 04.

Math Tip #2: Order of Execution

The computer calculates mathematical expressions in a certain order. Calculations of the same type or of equal priority are performed from left to right.

	SYMBOL	
1.	( )	—Everything inside *parentheses* comes first, then
2.	^	—*Exponentiation*, or raising a number to a power, such as: 5 ^ 3 which = 5 times 5 times 5 or 5^3 or 125, then
3.	*	—*Multiplication* and — Performed in order of occurrence from left to right
	/	—*Division* then
4.	+	—*Addition* and — performed in order of occurrence from left to right
	-	—*Subtraction*

Examples:

Look what a difference this can make...

```
9 * 4+ 3= 39
```
but...
```
9 * (4+ 3) = 63
```
or....
```
62 - 3 * 8 = 38
```
but...
```
(62 - 3) * 8 = 472
```
When in doubt, put parentheses around what you want done first.

Math Tip #3

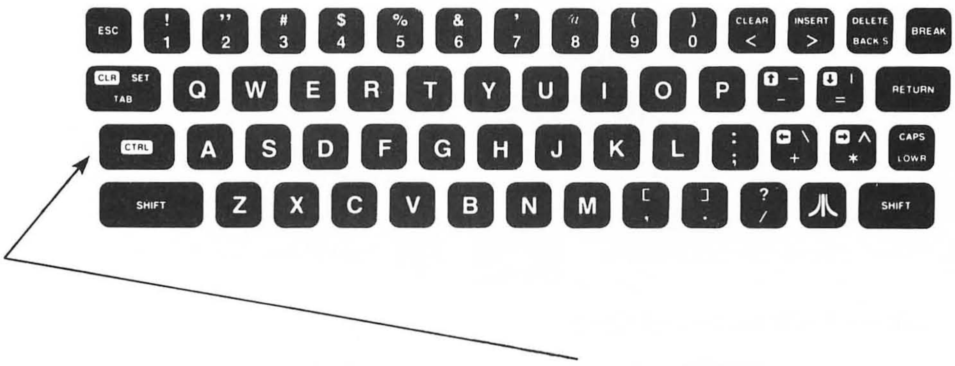
Remember when Mrs. Grundle, your 5TH grade teacher, taught you how to write numbers with a comma after every three places?

1,000,000 = one million (according to Miss Grundle)

WELL, GET WITH IT MRS. GRUNDLE!!!
COMPUTERS DON'T USE COMMAS!!!

1000000 = One million (according
to your computer)

More Key Information


ESC ! 1 " 2 # 3 $ 4 % 5 & 6 ' 7 @ 8 (9) 0 CLEAR < INSERT > DELETE BACK S BREAK
CLR SET TAB Q W E R T Y U I O P - _ | = RETURN
CTRL A S D F G H J K L : ; \ + ^ * CAPS LOWR
SHIFT Z X C V B N M [,] . ? / SHIFT

The Magic of the CONTROL (CTRL) Key

Once you learn how to use the CTRL key you can really get control of your computer display.

The CTRL key is always held down while pressing another key.

For instant gratification, hold down the CTRL key and press the " 2 key.

This section is going to concentrate on using the CTRL key for editing programs and changing your screen display.

Remember, when you see CTRL you should hold the control key down while pressing the other key.

Use the following steps in order to:

A. Move the cursor (screen marker) without changing the text on the screen.

B. Insert and delete words and spaces.

1. Start by typing a program line.

```
10 PRINT "HELLO JOHN"
```

2. Now press the following... CTRL ↑ until you have put the cursor back on top of the 1.

3. Now press CTRL and hold the → down until the cursor is on top of the J in JOHN. If you went too far you can use CTRL ← to back up or keep going with CTRL → and let the cursor wrap around the screen.

   Notice that your program line remains intact. It would have been wiped out if you had not been holding the CTRL key down.

4. Release the CTRL key when making your changes. With the cursor on top of the J, *take your finger off the* CTRL *key* and change JOHN to MARY. Press RETURN and the change is completed.

5. Now, use the CTRL and arrow keys as before only this time stop between HELLO and MARY.

```
10 PRINT "HELLO MARY"
           Stop here
```

6. Press the CTRL and the INSERT > key on the top row of the keyboard. Spread the words out by inserting spaces—even down to the next line if you want. You can hold CTRL and INSERT > down to insert a lot of spaces.
7. Press the space bar once and type the word you want to insert.

```
10 PRINT "HELLO TYPHOID<—        MARY"
```

8. Holding down CTRL and using the DELETE BACK S key will bring MARY back next to TYPHOID.
9. Don't forget to press RETURN or the changes won't be made in the computer's memory.

Using the CTRL and DELETE BACK S keys can be very absorbing. Try this...

Use CTRL and the ↑ - and → * keys to place the cursor on top of the T in line 10.

```
10 PRINT "HELLO [T]YPHOID MARY"
```

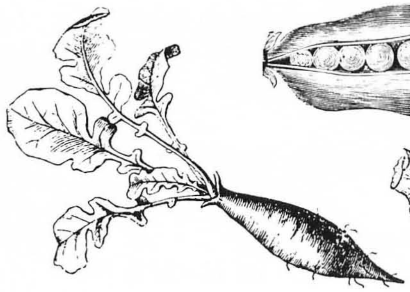
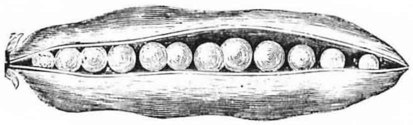
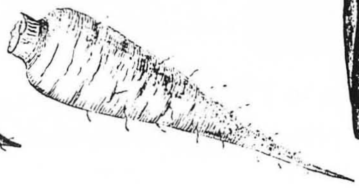
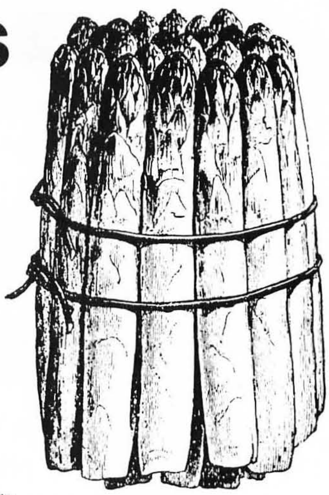
Now press CTRL DELETE BACK S and watch TYPHOID being absorbed into the cursor.

If you don't press RETURN after you have made your changes then only the screen display was changed and not the program line in the computer's memory. If you press RETURN you will get rid of the TYPHOID in MARY forever.

FINAL ADVICE. You're probably going to make a few mistakes as you get control of the CTRL key. For instance, you may accidentally press CTRL CLEAR instead of CTRL INSERT. But after some practice, you will really appreciate these screen editing features.

VARIABLES

These are *vegetables*, not *variables:*

These are *variables*, not *vegetables:*

```
A          SUM        MONTH                  GROUPSIZE
            ANUM                    A$
  NAME$                        KOLORONE

ANSWER4                AGE           SCORE
                     ZONK
```

A *variable* is anything that can change or *vary*. Here are some familiar variables:

A musical note
The color of a TV screen
The price of gold
Your age
The rate of inflation

When using the computer, we usually assign variables names to help us remember what they stand for and to work with them. The price of gold might be called GOLDPRICE or GPRICE or PRICE or GOLD or G or BANANAPRICE. The computer could care less what you call it.

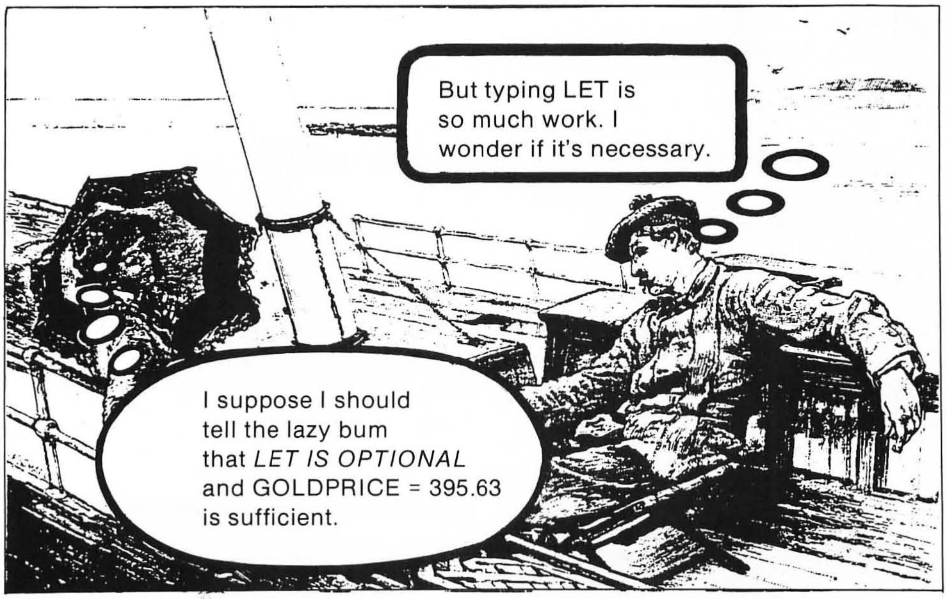
HINT: Don't leave any spaces in the variable name. GOLDPRICE not GOLD PRICE.

After we have a name, we need a value for the variable.

```
LET GOLDPRICE = 395.63
```

This would tell the computer to give the variable called GOLDPRICE a value of 395.63.

Now, tell it to PRINT GOLDPRICE and it will give you a value of 395.63.


But typing LET is so much work. I wonder if it's necessary.
I suppose I should tell the lazy bum that *LET IS OPTIONAL* and GOLDPRICE = 395.63 is sufficient.

ALSO NOTE: The computer starts out giving a value of zero (0) to all variables.

TYPE THIS: PRINT ZONK—It will show you that ZONK is 0 until you tell the computer otherwise.

There are TWO kinds of variables

Numeric variables

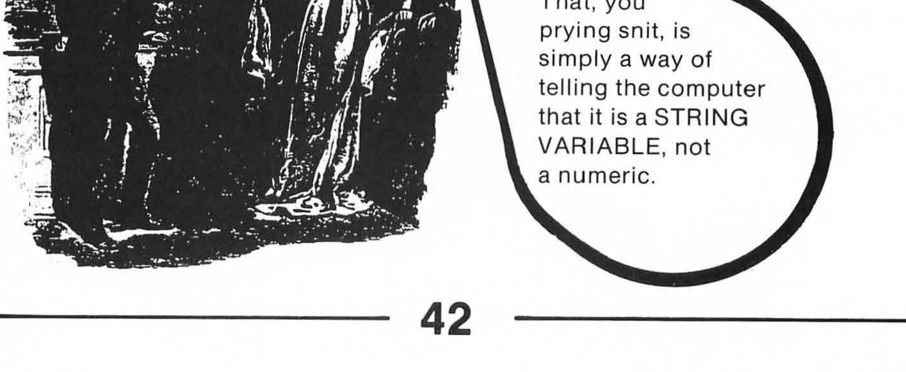
You put numbers in numeric variables:

```
GOLDPRICE = 395.63
AGE = 22
SPEED = 120
```

String variables

You put words or characters in these:

```
NAME$ = "JOHN DOE"
MONTH $ = "MAY"
PLACE$ = "EGYPT"
A$ = "NANCY"
```


Now, suppose you tell me the meaning of the $ used after the names for the string variables!
That, you prying snit, is simply a way of telling the computer that it is a STRING VARIABLE, not a numeric.

Two Important Things About String Variables

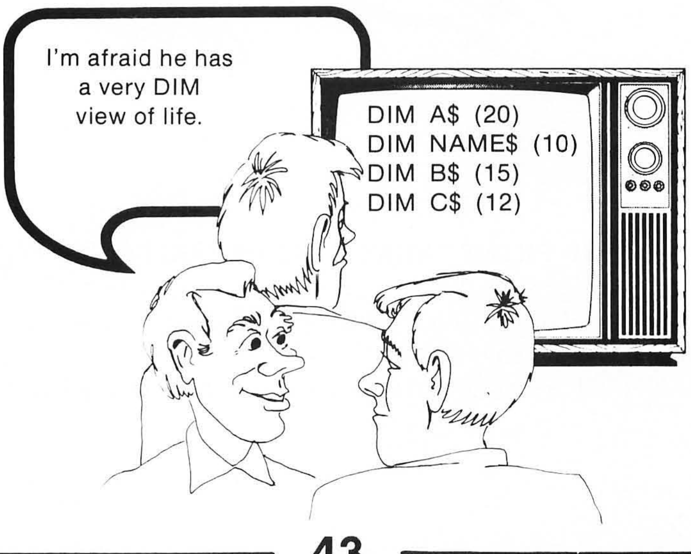
1. YOU MUST first tell the computer how many spaces to reserve for your string variable. That is done with a DIMENSION statement. It goes like this:

```
10 DIM NAME$(8)
```

This means you can have as many as eight characters in your name.
"BOB" would fit.
"BETTY" would be fine.
"FRANZ SCHUBERT" would be unfinished.

2. The other important thing is to remember to put QUOTES around your STRINGS.

```
10 DIM A$(20)<—We're allowing more room
20 LET A$ = "FRANZ SCHUBERT"
              Don't forget to use quotes here
30 PRINT A$
```



INPUT

A statement which gives the computer a great deal of flexibility is INPUT. INPUT is used to make the computer ask you a question and it lets you enter your own answer.

Here is an example of how INPUT is used in a program:

```
10 DIM DOG$(20)
20 PRINT "WHAT KIND OF DOG DO YOU OWN"

30 INPUT DOG$
40 PRINT "A ";DOG$;" IS A GOOD DOG"
```

Things to Remember About Input

1. It is usually used with a prompt which guides the user to type in the correct information.

This is the prompt:

```
          20 PRINT "WHAT KIND OF DOG DO YOU OWN"
```

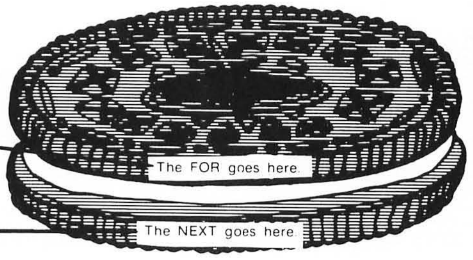
2. You do not need to put a ? at the end of your prompt. The INPUT statement will do that for you.
3. If you INPUT a string variable you need to DIMension it first.

FOR / NEXT Loops

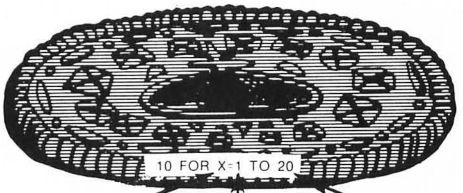
The FOR/NEXT loop is like an oreo cookie.

First,

At the end of the loop



In between them is the cream of the program, or what you want to repeat by using the loop. Here's what is in the FOR part:



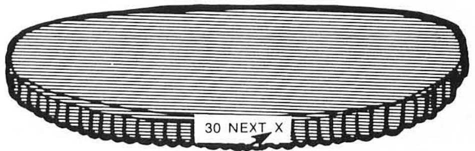
The variable name comes after the FOR

Then comes the lower limit (what X will be equal to when the loop begins).

Then comes the upper limit (what X will be when the loop is completed).

20 PRINT "THIS IS THE CREAM"

Any legal BASIC statement or group of statements can go here.



This has to be the same variable name used in the FOR part of the loop.

Here's the program again:

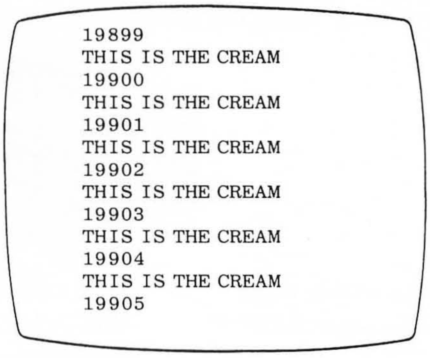
```
10 FOR X=1 TO 20
20 PRINT "THIS IS THE CREAM"
30 NEXT X
```

Run this. Then, add this... `25 PRINT X`

As you can see, the value of X is increased by 1 each time through the loop.

Change line 10 to... `10 FOR X=1 TO 20000`

...and see who gets bored first, you or the computer.



```
19899
THIS IS THE CREAM
19900
THIS IS THE CREAM
19901
THIS IS THE CREAM
19902
THIS IS THE CREAM
19903
THIS IS THE CREAM
19904
THIS IS THE CREAM
19905
```

There is a way to make the computer skip a specific amount of numbers each time through the loop:

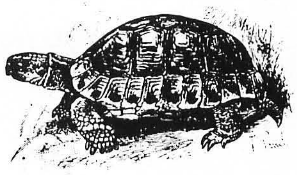
```
10 FOR Z=0 TO 100 STEP 5
20 PRINT Z
30 NEXT Z
```

This tells the computer to count only every 5th number between 0 and 100.

You can also go backwards...

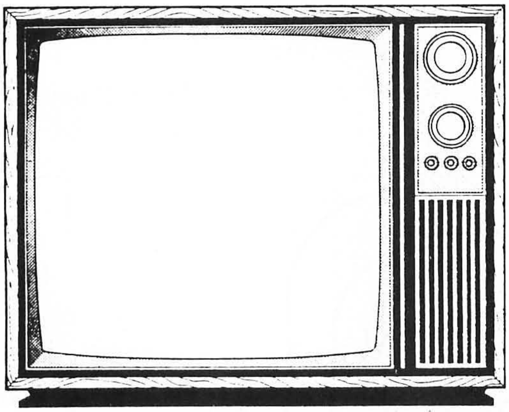
```
10 FOR Z=100 TO 0 STEP -5
20 PRINT Z
30 NEXT Z
```

Don't forget the - (minus) sign when going from high to low values.



Using the FOR/NEXT Loop to Delay the Computer...

Sometimes the computer works too fast!!! Here, for example, is the computer saying HELLO 43 times...



WANT TO SEE IT AGAIN?
To slow down a display, or anything else the computer does, you can use a FOR/NEXT loop with *nothing* between the FOR and the NEXT.

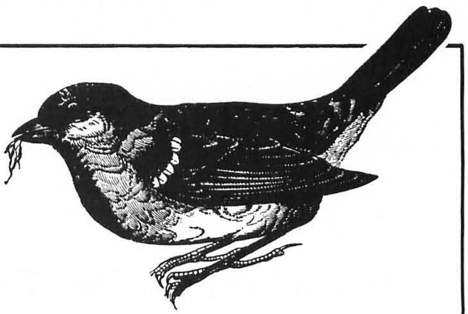
```
10 PRINT "GOOD"
20 FOR DE=1 TO 500
30 NEXT DE
40 PRINT "MORNING"
```

When you use a blank FOR/NEXT loop, you are making the computer go through the loop and do *nothing* each time through.

"Computer—go chase your own tail for a count of 500."

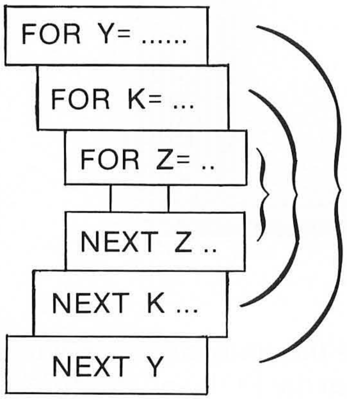
Try experimenting with different values in line 20 and observe the effect.

Nesting FOR/NEXT Loops...

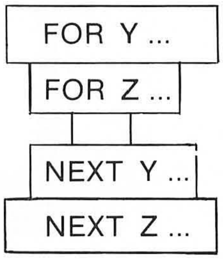


When you *nest* a FOR/NEXT loop, you put one inside of another. You can do this as much as necessary, BUT, you have to build your FORs and NEXTs from the inside out...

This will work



This will never do



This is a final sample:

```
10 FOR Y=1 TO 3
20 FOR K=255 TO 0 STEP -5
30 SOUND 0,K,10,10
40 PRINT K
50 FOR Z=1 TO 30: NEXT Z: REM DELAY
60 NEXT K
70 NEXT Y
```

If you can't hear anything, turn up the volume on your TV.

Even More Key Concepts

Here are two more keyboard control screen editing features which you may find useful:

I. How to temporarily stop many BASIC programs and how to interrupt and observe parts of a long program listing. To stop the program or listing hold CTRL and press !/1. To restart the display, hold CTRL and press !/1 again. Try this with a program such as `10 PRINT "HELLO": GOTO 10`

II. How to duplicate lines. Sometimes you might build on to your program so much that you box yourself in and need to renumber. Almost as frequently, you might need to repeat a line you already have or use a similar line in another part of a program. Instead of retyping the whole line, clone the one you already have and make the necessary changes.

This is all you need to do in order to clone program lines...

```
30 FOR DELAY=1 TO 50: NEXT DELAY
```

Let's suppose you have this perfectly wonderful delay loop and you want one of the exact same loops at line 70 and at line 90.

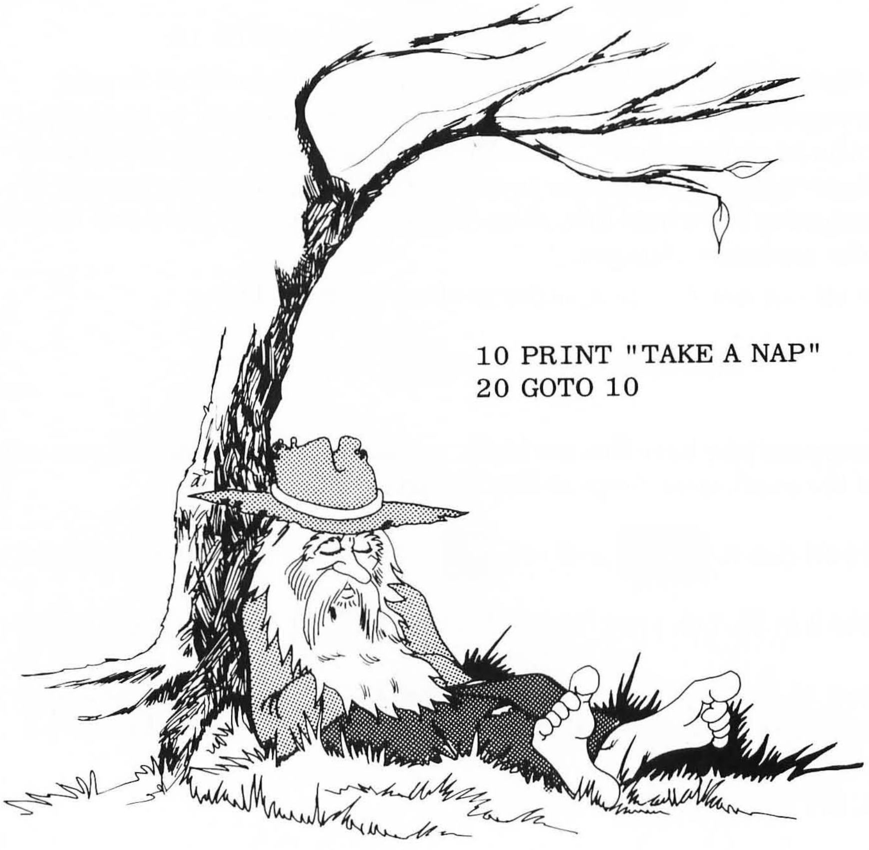
1. Hold down CTRL and use ↑ to position the cursor on top of the 3 in 30. Lift your finger off of the CTRL key and type a 7 on top of the 3 and press RETURN. Nothing else will appear to have changed. Now go back up one more time and change the 7 to a 9 and press RETURN.
2. LIST the program to reveal

```
30 FOR DELAY=1 TO 50: NEXT DELAY
70 FOR DELAY=1 TO 50: NEXT DELAY
90 FOR DELAY=1 TO 50: NEXT DELAY
```

GOTO

GOTO is pretty straightforward (and is probably one of the single greatest computer discoveries you will encounter).

```
10 PRINT "SOMEONE PULL THE PLUG"
20 GOTO 10
```



```
10 PRINT "TAKE A NAP"
20 GOTO 10
```

Rip Van Programmer

Steppers—Counters—Accumulators

STEPPERS change the value of a variable each time through the loop. Used as a COUNTER, they count how many times you've been through the loop.

This is called initializing a variable (giving it an initial value)

This is not a type of algebraic equation.
It really stands for: Let the current value of X be increased by one.

```
10 X=1
20 PRINT X
30 X=X+1

40 GOTO 20
```

ACCUMULATORS keep a running total for you.

```
10 REM UNBALANCED CHECKBOOK PROGRAM
20 REM NUM IS THE CHECK NUMBER
30 REM AMT IS THE CHECK AMOUNT
40 REM TOT IS THE ACCUMULATED TOTAL
50 NUM=1: AMT=0: TOT=0: REM INITIALIZING
VARIABLES
60 PRINT "ENTER AMOUNT OF CHECK #";NU
M; : INPUT AMT
70 NUM=NUM+1: TOT=TOT+AMT
80 PRINT : PRINT "YOU ARE $";TOT; "IN D
EBT"
90 PRINT : GOTO 60
```

A GOTO LOOP that never stops going forms an unconditional branch. The following unconditional branch is fairly limited and produces disastrous results.

10
I'm afraid an unconditional branch caused her to take in too much salt!

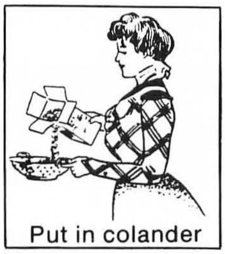
20

30 GOTO 10

By combining the GOTO with the conditional branch produced by the IF/THEN, see what is now possible:

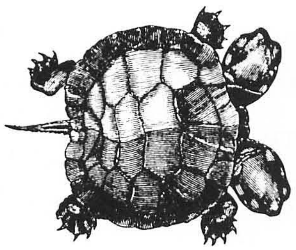
NOTE: This is an example—not a real program.

```
10 SALT THE WATER
20 TASTE THE WATER
30 IF SALT CONTENT = .9 PARTS PER MILLION
THEN PRINT "GOSH! THIS TASTES GOOD.": GOTO 50
40 GOTO 10
50 COOK FOR SEVEN MORE MINUTES
```

60 

ETC.

The IF/THEN decision capability is really the key to the whole business of BASIC programming. Of course, since it's the key, you have to insert it properly.


There are only two ways to look at it. The comparison is true or it is not.

IF A>B THEN
means (IF A IS GREATER THAN B)

IF A<B THEN
(IF A IS LESS THAN B)

IF A=B THEN
(OBVIOUS)

IF A <= B THEN
(IF A IS LESS THAN OR EQUAL TO B)

IF A >= B THEN
(IF A IS GREATER THAN OR EQUAL TO B)

IF A<>B THEN
(IF A DOES NOT EQUAL B)

IF A=X AND B=Y THEN
IF A=X OR B=Y THEN

If the computer finds the comparison true, it will do whatever follows THEN.

```
10 IF A>B THEN PRINT "A IS BIGGER"
```

If the comparison is *not true*, the program drops through to the next line.

```
20 PRINT "THEN A MUST BE <THAN OR=TO B"
```

```
10 PRINT : PRINT "WHAT IS YOUR AGE"
20 INPUT A
30 IF A>95 THEN PRINT "WELL, NOT EXACTL
Y A SPRING CHICKEN" : GOTO 10
40 PRINT "OH, AN INEXPERIENCED YOUTH! "

50 GOTO 10
```

Using the IF/THEN

In the following program, the computer checks to see if you put in the number 25. If you didn't, it will give you another chance.

```
10 REM ONLY 25 WILL WORK
20 PRINT "ENTER A NUMBER"; : INPUT A
30 IF A <0 THEN PRINT "NOT A VALID NUMB
ER" : GOTO 90
40 IF A<1000 THEN PRINT "IT'S WITHIN O
UR RANGE" : PRINT : GOTO 60
50 PRINT "THIS NUMBER IS OUT OF OUR RA
NGE" : GOTO 90
60 PRINT "CHECKING NUMBER" : FOR DE=1 TO
400 : NEXT DE
70 IF A<>25 THEN GOTO 90
80 PRINT : PRINT "RIGHT YOU ARE! " : END
90 PRINT : PRINT "TRY AGAIN" : PRINT
100 GOTO 20
```

Don't forget STRING comparisons:

```
        IF CITY$ = "BUFFALO" THEN PRINT "AT
        LAST, I'VE FOUND A DECENT PLACE TO
        LIVE!"
```

```
10 DIM NAME$(30)
20 PRINT "WHAT IS YOUR FIRST NAME";
30 INPUT NAME$
40 IF NAME$ = "JANET" THEN GOTO 60
50 GOTO 20
60 PRINT : PRINT "WELCOME TO THE SYSTEM
, ";NAME$
```

and, we can combine STRING
and NUMERIC comparisons...

```
10 DIM NAME$(30)
20 PRINT "WHAT IS YOUR FIRST NAME";
30 INPUT NAME$
40 IF NAME$ = "JANET" THEN GOTO 60
50 GOTO 20
60 PRINT : PRINT "WELCOME TO THE SYSTEM
, ";NAME$
70 PRINT "WHAT IS YOUR ACCOUNT #";
80 INPUT ACC
90 IF ACC=472 THEN GOTO 110
100 PRINT "INVALID NUMBER, TRY AGAIN":
GOTO 80
110 PRINT : PRINT "ACCEPTED"
120 PRINT "CHOOSE YOUR TRANSACTION"
```

ON/GOTO

ON/GOTO allows you to do a number of comparisons very easily. Here is the ON/GOTO format:

```
ON   X   GOTO  ________ , ________ , ________ ETC.
```

any numeric variable

the line that would be executed if X=1

the line that would be executed if X=2

if X=3

EXAMPLE:

This is great for multiple-choice type inputs.

```
10 PRINT "1=LINE 100"
20 PRINT "2=LINE 150"
30 PRINT "3=LINE 200"
40 PRINT "4=LINE 250"
50 PRINT : PRINT "PICK YOUR NUMBER AND
PRESS THE RETURN KEY"
60 INPUT X: IF X>4 THEN GOTO 10
70 ON X GOTO 100,150,200,250
100 PRINT "THIS IS LINE 100": GOTO 300
150 PRINT "THIS IS LINE 150": GOTO 300
200 PRINT "THIS IS LINE 200": GOTO 300
250 PRINT "THIS IS LINE 250": GOTO 300
300 PRINT : FOR DELAY=1 TO 200: NEXT DEL
AY: RUN
```

NOTE: When the computer encounters RUN within a program, it will start the program from the beginning.

GOSUB and RETURN

GOSUB and RETURN are close cousins to the GOTO statement, but there is one important difference:

> A GOSUB keeps track of where it left off, and, when it encounters a RETURN, it goes back to the next item on the program agenda and executes it.

For example:

```
10 PRINT "HELLO"
20 GOSUB 100
30 PRINT "GOODBYE"
40 END
100 FOR Z=1 TO 10: PRINT "GLAD TO SEE Y
OU": NEXT Z: RETURN
```

Now, add another line.

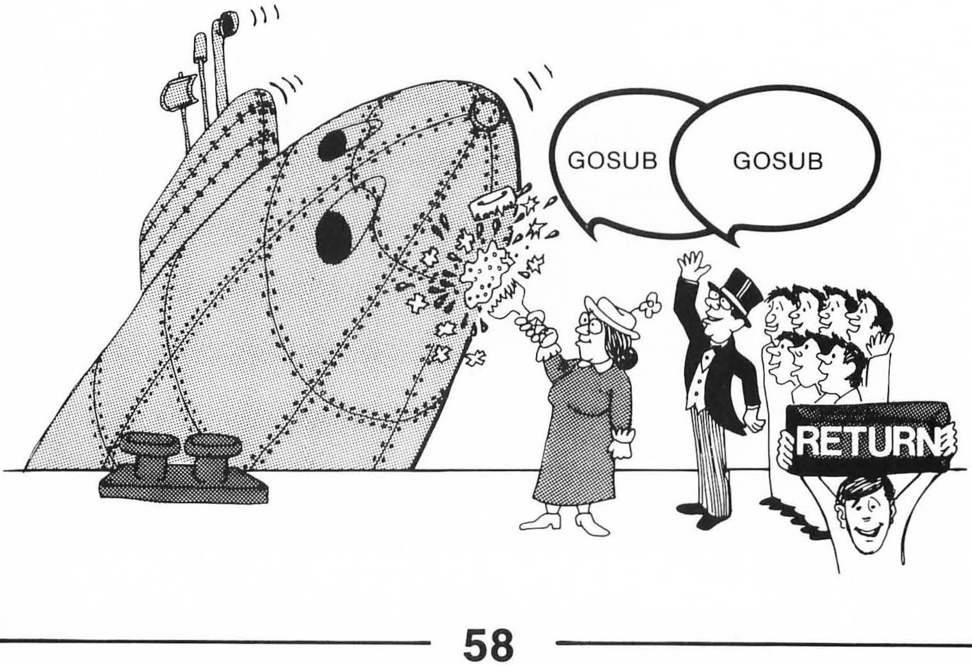
```
5 PRINT "GOOD MORNING": GOSUB 100
```

NOTE: The END in line 40 is necessary to prevent the program from crashing into line 100.

In the following program, which calculates an average, the "SUBROUTINE" adds the new number to the total.

```
10 PRINT "HOW MANY NUMBERS TO BE AVERA
GED";
20 INPUT N
30 FOR Z=1 TO N
40 PRINT "ENTER NUMBER ";Z;
50 INPUT NEWNUM
60 GOSUB 200
70 NEXT Z
80 AVE=TOT/N
90 PRINT : PRINT "THE AVERAGE IS";AVE
100 END
200 TOT=TOT+NEWNUM: RETURN
```

Of course, this could have been done without a SUBROUTINE; but keep in mind, we could have added innumerable calculations between line 200 and the RETURN.



ON/GOSUB

ON/GOSUB is very much like ON/GOTO. However, the ON/GOSUB branch contains a RETURN and the ON/GOTO does not. Here's an example of ON/GOSUB:

```
10 REM ONGOSUB
20 PRINT : PRINT "QUESTION 1"
30 PRINT "WHAT IS THE SQUARE ROOT OF 2
5"
40 INPUT SQ: IF SQ= 5 THEN REFORC= INT (RN
D (0) *3) +1: ON REFORC GOSUB 1000, 2000, 30
00, 4000: GOTO 70
50 PRINT : PRINT "TRY AGAIN": GOTO 30
60 REM ONLY TWO QUESTIONS IN SAMPLE
70 PRINT : PRINT "QUESTION 2"
80 PRINT "HOW FAR IS IT IN MILES TO TH
E SUN"
90 INPUT SD: IF SD= 93000000 THEN REFORC
= INT (RND (0) *3) +1: ON REFORC GOSUB 1000,
2000, 3000, 4000: GOTO 110
100 PRINT : PRINT "TRY AGAIN": GOTO 80
110 PRINT : PRINT "ETC. ": END
1000 PRINT "GREAT WORK! ": RETURN
2000 PRINT "GOOD GOING! ": RETURN
3000 PRINT "ANOTHER EINSTEIN! ": RETURN
4000 PRINT "CORRECT AGAIN": RETURN
```

In this example, the computer selects, at random, one of four reinforcers if a person gets the answer correct.

NOTE: For more information on how to use random numbers, consult the section on FUNCTIONS.

READ/DATA

```
10 READ X
20 DATA 20,7,5 100,40

40 GOTO 10
```

Computers do not read for enjoyment; they must be instructed to read. Fortunately, there's not much of a trick to this; you simply have to tell the computer in a straightforward manner to...

READ X (any numeric or string variable name)

The computer will find the value of X in the *DATA* statement.

```
DATA 5,10,15,20,25
```

Commas are used to separate values in data statements.
Used in a program, it can look like this:

```
10 READ X
20 PRINT X: IF X=25 THEN END
30 GOTO 10
40 DATA 5,10,18,200,25
```

No comma used here

The POINTER is very finicky and does not lose track of which value of X it is reading.

The pointer starts at the first position following DATA.

```
40 DATA 5, 10, 18, 200, 25
```

The second time through the loop, the next value is pointed to

```
40 DATA 5, 10, 18, 200, 25
```

If you run out of room for DATA values, you can create a *new* DATA line.

```
10 READ X
20 PRINT X
30 GOTO 10
40 DATA 5,10,18,200,25
50 DATA 34,44,5,56,2,344,333,5,5,666,2
,2,37,86,5,84,35,66,6,88,57,7777,33,3,
333,9,4,7,34,85,3,44,46,3,6,3,6,9
60 DATA 2,34,55,66,77,-1
```

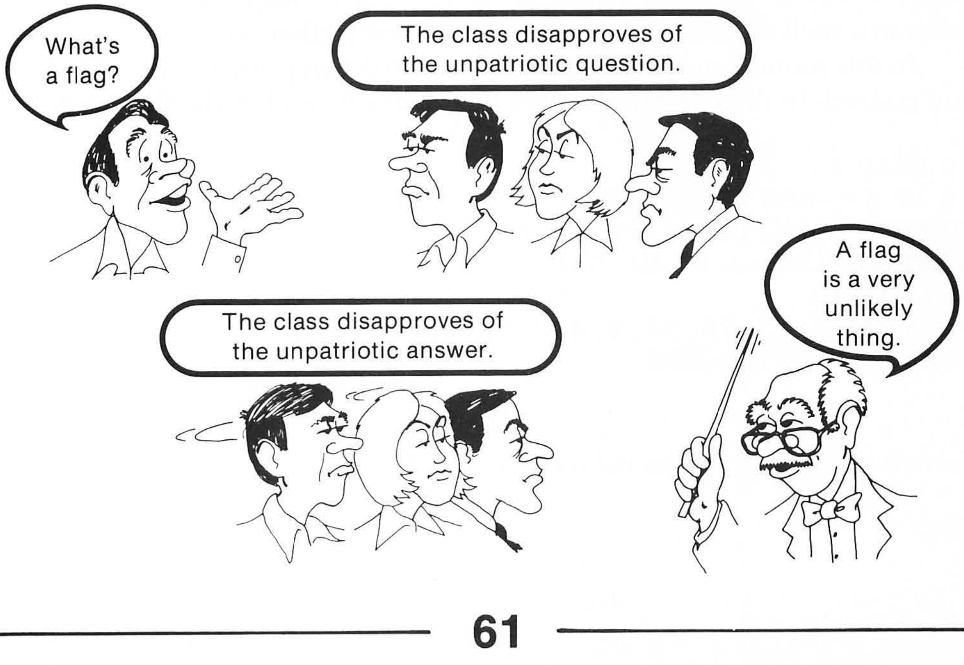
Imagine assigning all of those values using LET statements!

As mentioned earlier, the POINTER is very picky, and if you run out of DATA, it will show its displeasure....

`"ERROR-6 AT LINE 10"` or some such nonsense

will be the result of the POINTER pointing to thin air, instead of a value for the variable.

The standard way to deal with this is to use a piece of DATA called a FLAG in combination with an IF-THEN statement.


What's a flag?
The class disapproves of the unpatriotic question.
A flag is a very unlikely thing.
The class disapproves of the unpatriotic answer.

A FLAG is a variable in a data statement which is *very unlikely* to occur, *unless* it is placed there intentionally. It is put at the end of a DATA statement to let the computer know it has reached the end of the DATA statement.

This is an example of a flag within a program using a DATA statement:

```
10 REM MPG = MILES PER GALLON
20 READ MPG
30 IF MPG=470 THEN END  ←—— Here's the test for the flag.
40 PRINT MPG; " MPG"
50 DATA 10, 20, 15, 60, 40, 470← Unlikely mpg
60 GOTO 20
```

RESTORE

Before leaving READ and DATA, a few more issues must be covered. RESTORE, for example, will set the pointer back to the beginning.

Not only is this often incredibly handy, but it's necessary in programs such as the inventory program listed further on.

In this sound program, line 20 instructs the computer to "RESTORE" and go back to the beginning of the data when it reaches the flag—888.

```
10 READ X
20 IF X=-888 THEN RESTORE : GOTO 10
30 SOUND 0,X,10,10: PRINT X
40 FOR DELAY=1 TO 40: NEXT DELAY
50 GOTO 10
60 DATA 23,55,76,33,6,4,200,165,20,3,4
5,55,76,23,78,-888
```

Remember to turn up the volume on your TV.

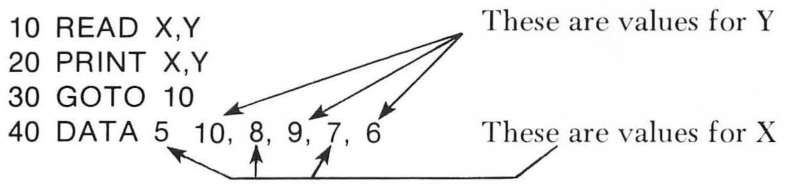
WAIT, THERE'S MORE.........

You can add a specific line number to the RESTORE statement...

RESTORE 3000

This instructs the computer to set the pointer to the first value in the data statement on *line 3000.* Of course, there's even more!

The computer can keep track of multiple variables.



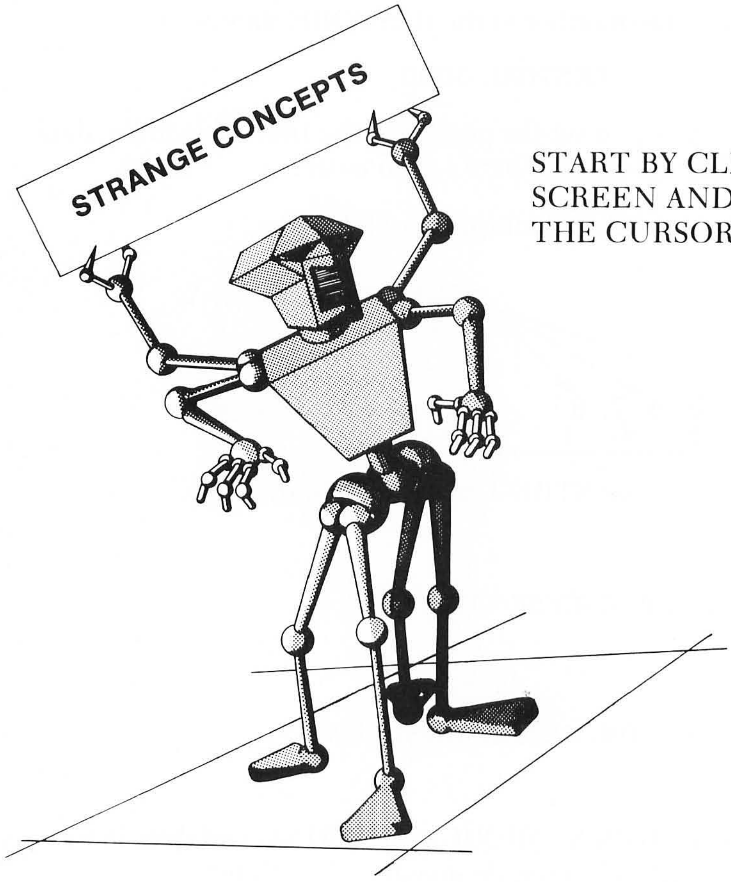
```
10 READ X,Y
20 PRINT X,Y
30 GOTO 10
40 DATA 5  10, 8, 9, 7, 6
```

AND MORE!!! You can use STRINGS in DATA statements.

```
10 DIM NAME$ (20)
20 READ NAME$
30 IF NAME$ = "COTTAGE CHEESE" THEN END

40 PRINT NAME$
50 GOTO 20
60 DATA DIANE, DAVE, TOM, MARIE, COTTAGE C
HEESE
```

Finally, you can use both NUMERIC and STRING variables in the same READ statement. Here's a simple inventory program:

```
10 REM PN=PARTNUMBER AND P$=PARTNAME
20 DIM PART$ (20) , P$ (20)
30 PRINT "WHAT IS THE NAME OF THE PART
": INPUT PART$
40 RESTORE
50 READ P$, PN
60 IF P$ = "FROG" THEN GOTO 90
70 IF P$ =PART$ THEN PRINT "THE PART NU
MBER IS ";PN: PRINT : GOTO 30
80 GOTO 50
90 PRINT "NO SUCH PART": PRINT : GOTO 30

100 DATA WHEEL, 1234, BOLT, 2345, SPOOL, 34
56, LATCH, 5678, COG, 6789, FROG, 7890
```



START BY CLEARING THE SCREEN AND TURNING OFF THE CURSOR.

Many times you want to put words and characters on the screen without having the READY and the cursor left in the display.

To clear the screen in the beginning of a program, start with a rather strange sequence of keystrokes...

```
5 PRINT " ↖"
```

To get this arrow, press ESC once, then hold down CTRL and press CLEAR < . Then close the quote.

This instruction tells the computer to start your program by clearing the screen.

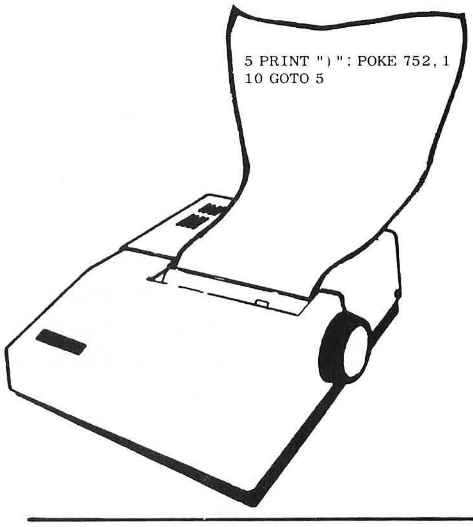
Try...
```
5 PRINT "↰"
10 GOTO 5
RUN
```

Notice that the screen is being cleared, but the cursor keeps popping in. This cursor can easily be eliminated by changing line 5 to

```
5 PRINT "↰": POKE 752,1
```

With the above line at the beginning of your program you can make things a lot neater.

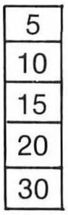
NOTE: When you list a program containing a PRINT "↰" on a printer, the arrow will not show up. Instead, you will see a parenthesis ")" or bracket "}" on the paper.



```
5 PRINT ")": POKE 752,1
10 GOTO 5
```

Hip Hip ARRAYS

ARRAYS can be thought of as boxes.

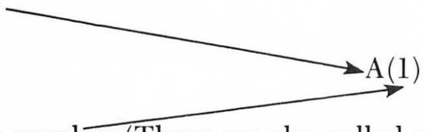


or "elements" in which you store numbers. In ATARI BASIC, they are groups of numeric variables and, like any numeric variable, you can assign values to them, change the values, or do mathematical manipulations with them (by multiplying the value in one box by the value in another, for instance). In this example, you could multiply the top box by the bottom and get 150.

Of course, just like anything else related to computer programming, you need to learn a few rules...

ARRAY Rule #1

Each ARRAY element needs a two-part name. First comes a letter or letters.

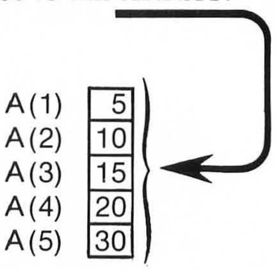


Then a number (These are also called subscripts.)

Let's assume we have assigned *these values* to this ARRAY.

In this case,

A(1) is equal to 5
A(2) is equal to 10
A(3) is equal to 15
etc.



ARRAY Rule #2

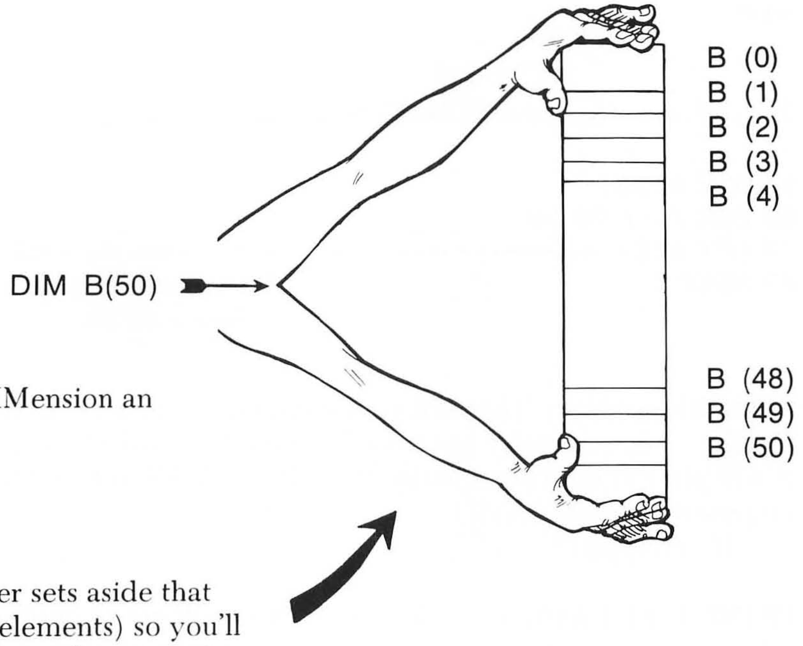
Before working with ARRAYS, *you must DIMension* them. But there is an important difference between DIMensioning string variables and DIMensioning ARRAYS...

When you DIM an ARRAY such as...

```
DIM B(50)
```

you're not saying how many characters are going to be in that ARRAY, as you would be if you were DIMensioning a string. When you DIMension ARRAYS, you're telling the computer how many boxes you are going to store numbers in and what the names of those boxes are going to be.

Here's an example of using an ARRAY with the letter B...



When you DIMension an ARRAY...

...the computer sets aside that many boxes (elements) so you'll be able to put numbers into them. You even get one extra free because B (0) can be used also.

Look, I've had it up to here with these ARRAY rules. Tell me what you do with these things.

First, you can assign values to variables very quickly with ARRAYS. It's easier to assign values to a lot of ARRAY elements than it is to assign values to individual variables.

```
10 A1=1
20 A2=2
30 A3=3
etc.
```

Imagine writing lines like this 50 times!

Instead, you can use an ARRAY to assign values...

```
10 DIM A(50)
20 FOR Z=1 TO 50
30 LET A(Z) =Z
40 NEXT Z
```

This assigns the value of Z to the ARRAY starting with 1 and ending with 50.

RUN this program. Then tell your computer to PRINT A(37) RETURN . The computer has stored 37 into that ARRAY element so it will print 37 until you change the value in A(37) or turn OFF the computer and erase A(37).

If you type...

```
PRINT A(5) +A(9)
```

You will get 14 until you change these values or erase them from memory.

And you can play some music with your ARRAY.

```
10 REM ARRAY SOUND PROGRAM ONE
20 DIM A(50),B(50),C(50),D(50)
30 FOR Z=1 TO 50
40 LET A(Z)=0: LET B(Z)=0: LET C(Z)=0: LE
T D(Z)=0: REM SETS ALL VALUES TO ZERO
50 LET A(Z)=Z*5: LET B(Z)=Z*4: LET C(Z)=
Z*3: LET D(Z)=Z*2
60 NEXT Z
70 FOR N=1 TO 50
80 SOUND 0,A(N),10,10: SOUND 1,B(N),10,
10: SOUND 2,C(N),10,10: SOUND 3,D(N),10,
10
90 NEXT N
100 FOR N=50 TO 1 STEP -1
110 SOUND 0,A(N),10,10: SOUND 1,B(N),10
,10: SOUND 2,C(N),10,10: SOUND 3,D(N),10
,10
120 NEXT N
130 GOTO 70
```

By changing a few lines, you can get a screen display of all of the values in your ARRAYS.

```
10 REM PROGRAM ARRAY SOUND TWO
20 DIM A(50),B(50)
30 FOR Z=1 TO 50
40 LET A(Z)=0: LET B(Z)=0: REM SETS ALL
VALUES TO ZERO
50 LET A(Z)=Z *5: LET B(Z)=Z*4
60 NEXT Z
70 FOR N=1 TO 50
80 SOUND 0,A(N),10,10: SOUND 1,B(N),10,
10
90 NEXT N
100 FOR N=50 TO 1 STEP -1
110 SOUND 0,A(N),10,10: SOUND 1,B(N),10
,10
120 NEXT N
130 SOUND 0,0,0,0: SOUND 1,0,0,0
140 FOR N=1 TO 50: PRINT "A(";N;")=";A(
N): SOUND 1,A(N),10,10: NEXT N
150 FOR N=1 TO 50: PRINT ,"B(";N;")=";B(
N): SOUND 1,B(N),10,10: NEXT N
```

As mentioned earlier, you can also do mathematical manipulations with ARRAY elements.

If you RUN the preceding program and press SYSTEM RESET , the values are still in the ARRAY elements.

So, if you type... `PRINT A(50)` RETURN

...You should get a *250*.

Now type... `PRINT B(50)` RETURN

...And you should get a *200*.

So if you type... `PRINT A(50) +B(50)` RETURN

...You should get a *450*.

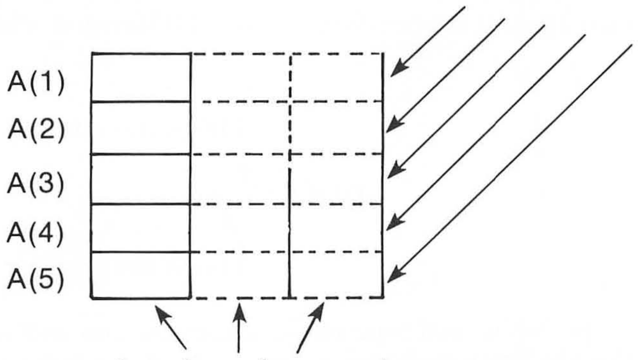
ARRAYS are useful in programs which keep track of and manipulate large groups of numbers

- Instrument readings could be I(1) through I(40).
- Average rainfall in the 100 largest cities in the United States could be R(1) through R(100).
- Daily hamburger production rates for your 175 hamburger joints could be stored in an ARRAY which is labeled H(1) to H(175).

MATRIX Madness

A *more powerful and more complicated* version of an ARRAY is called a MATRIX.

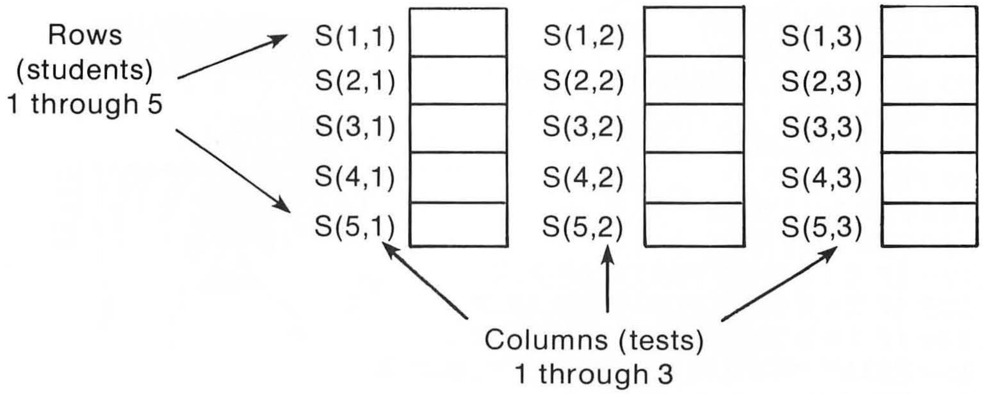
A MATRIX not only consists of rows of variables going down...



...but also columns going across.

For our sample program, you should imagine a class with 5 students. (The others got bored and dropped out.)

Each student will take three tests. The score for the first test will go into column #1; the score for the second will go into column #2; etc. For the sake of impersonalization, we will refer to each student with an "S" and a number. So our student test score matrix would look like this...



The only things we're missing now are the students' scores.

Although it's possible to fill in the scores by assigning values in this manner...

```
LET S(3,1)=78
```

...this program will use INPUT statements.

NOTE: Once again it is necessary to use a DIMension which is executed as follows:

This is the number of rows.

```
DIM S(5,3)
```

This is the number of columns.

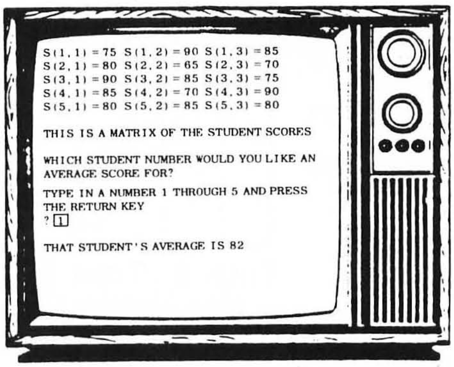
If you type NEW and type in this program, you will see that the beauty of the MATRIX is that you can manipulate all the values in both the rows and columns. Or pass this up and go directly to the next program which is more elaborate.

```
10 DIM S(5,3)
20 FOR COLUMN=1 TO 3
30 FOR ROW=1 TO 5
40 PRINT "TYPE IN THE STUDENT'S SCORE
AND PRESS RETURN"
50 INPUT SCORE
60 LET S(ROW,COLUMN)=SCORE
70 NEXT ROW
80 NEXT COLUMN
90 PRINT "↖"
100 FOR X=1 TO 3
110 FOR Z=1 TO 5
120 IF X=1 THEN POSITION 2,Z
130 IF X=2 THEN POSITION 15,Z
140 IF X=3 THEN POSITION 28,Z
150 PRINT "S(";Z;",";X;")=";S(Z,X)
160 NEXT Z
170 NEXT X
```

NOTE: I promise that this is the longest program I will ever ask you to type into your computer.

```
10 REM STUDENT MATRIX PROGRAM
20 PRINT "↰"
30 REM MATRIX 1
40 DIM S(5,3)
50 FOR COLUMN=1 TO 3
60 FOR ROW=1 TO 5
70 POSITION 2,11
80 PRINT "WHAT IS THE SCORE OF STUDENT
 NUMBER ";ROW;" "
90 IF COLUMN=1 THEN PRINT "ON TEST NUM
BER ONE"
100 IF COLUMN=2 THEN PRINT "ON TEST NU
MBER TWO"
110 IF COLUMN=3 THEN PRINT "ON TEST NU
MBER THREE"
120 INPUT SCORE: PRINT "↰"
130 IF SCORE>100 THEN PRINT "SMART KID
,THE HIGHEST SCORE IS 100.  TRY AGAIN
.";:PRINT:PRINT:GOTO 120
140 LET S(ROW, COLUMN)=SCORE
150 NEXT ROW
160 NEXT COLUMN
170 PRINT "↰"
180 FOR X=1 TO 3
190 FOR Z=1 TO 5
200 IF X=1 THEN POSITION 2,Z
210 IF X=2 THEN POSITION 15,Z
220 IF X=3 THEN POSITION 28,Z
230 PRINT "S(";Z;",";X;")=";S(Z,X)
240 NEXT Z
250 NEXT X
260 POSITION 1,8: PRINT "THIS IS A MATR
IX OF THE STUDENT SCORES"
270 POSITION 1,10: PRINT "WHICH STUDENT
NUMBER WOULD YOU LIKE AN AVERAGE SCOR
E FOR?"
280 PRINT : PRINT "TYPE IN A NUMBER 1 T
HROUGH 5 AND PRESS THE RETURN KEY"
290 INPUT S
300 IF S>5 THEN PRINT "↰":POSITION 1,1
1:PRINT "THERE ARE ONLY 5 STUDENTS":FO
R DE=1 TO 300:NEXT DE:GOTO 170
310 PRINT : PRINT "THAT STUDENT'S AVERA
GE IS ";INT((S(S,1)+S(S,2)+S(S,3))/3)
320 GOTO 270
```

Your final display should look like this.



```
S(1,1)=75 S(1,2)=90 S(1,3)=85
S(2,1)=80 S(2,2)=65 S(2,3)=70
S(3,1)=90 S(3,2)=85 S(3,3)=75
S(4,1)=85 S(4,2)=70 S(4,3)=90
S(5,1)=80 S(5,2)=85 S(5,3)=80

THIS IS A MATRIX OF THE STUDENT SCORES

WHICH STUDENT NUMBER WOULD YOU LIKE AN
AVERAGE SCORE FOR?
TYPE IN A NUMBER 1 THROUGH 5 AND PRESS
THE RETURN KEY
?1

THAT STUDENT'S AVERAGE IS 82
```

Of course the values will be those that you enter.

The computer could also easily average the class scores for each test. Simply add lines that tell it to add all 5 scores for each column and divide by 5.

There are plenty of other applications for MATRICES and there are many examples found in other BASIC books. In general, almost anything that can be put into rows and columns can be built into a MATRIX.

For instance, a Vice President of Sales might create a monthly report which looks something like this...

Monthly Report from the
Desk of K. Schaefer
$ Sales by Product by Region

	Widgets	Gadgets	Sprockets	Sockets	Total dollars for each region
Region #1					
Region #2					
Region #3					
Region #4					
Region #5					
Totals:					
	Total $ for Widgets	Total $ for Gadgets	Total $ for Sprockets	Total $ for Sockets	Total $ Sales

Part 2

Graphics, Color, and Sound

The Fundamentals of ATARI Graphics

ATARI Home Computers have very powerful graphics capabilities. Some of the advanced graphics require more than an introductory knowledge of programming and computer design. This section, however, should get you started with ATARI Computer graphics.

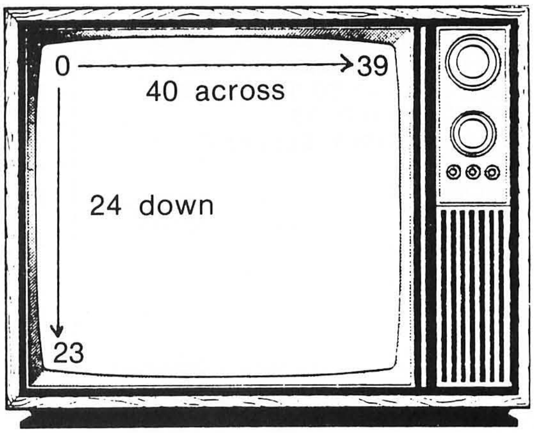
ATARI Computers have 3 *"text" modes for displaying words* on the screen. The first mode is...

GRAPHICS MODE 0
(Normal Blue Screen)

This mode is entered by any one of the following:

- Turning the computer ON
- Pressing SYSTEM RESET
- Typing GR. 0 RETURN

GR. 0 for Normal-Size Characters



In GRAPHICS 0 you can POSITION from 0 to 39 across and from 0 to 23 down. The computer normally indents 2 spaces when printing on the screen but not on the printer.

By using a POSITION in GRAPHICS 0, you can PRINT anywhere on the screen.

Number across **Number down**

```
10 POS. 16, 11
20 PRINT "MIDDLE"
```

You can use POSITION each time you PRINT in order to keep your display on the same part of the screen.

First RUN this program:

```
10 PRINT "↰": POKE 752,1
20 POSITION 19,11
30 PRINT X
40 X=X+1
50 GOTO 20
```

Then change line 50 to....

```
50 GOTO 30
```

and see the difference.

POSITION can be used to emphasize displays by making them appear to move.

Try the following.....

```
10 PRINT "↰": POKE 752,1
20 FOR X=1 TO 16
30 POSITION X,11
40 PRINT "MIDDLE"
50 FOR DE=1 TO 30: NEXT DE
60 FOR Y=0 TO 15
70 POSITION Y,11: PRINT " "
80 NEXT Y
90 NEXT X
```

or a slightly fancier version which makes the characters appear to walk across the screen.

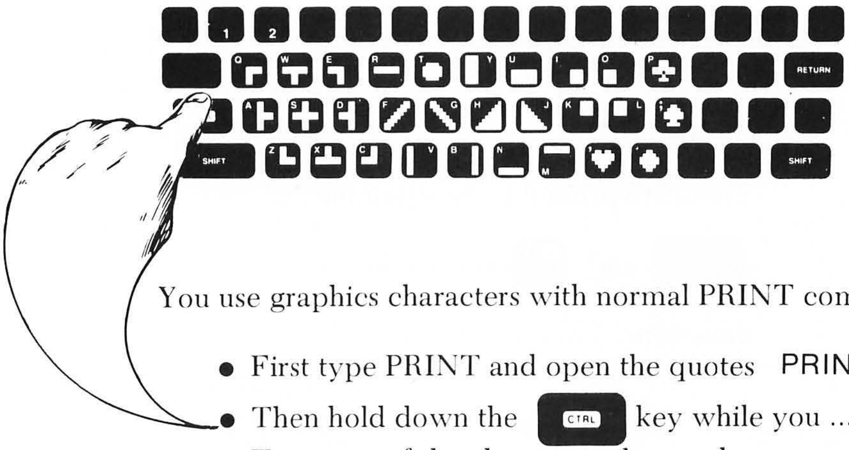
```
10 PRINT "↰": POKE 752,1
20 FOR X=1 TO 16
30 POSITION X,11
40 IF X/2=INT(X/2) THEN POSITION X,11:
PRINT "M/DD/E": GOTO 60
50 IF X<16 THEN POSITION X,11: PRINT "M
/DD/E"
60 IF X=16 THEN POSITION X,11: PRINT "M
IDDLE"
70 FOR DE=1 TO 50: NEXT DE
80 FOR Y=0 TO 15
90 POSITION Y,11: PRINT " "
100 NEXT Y
110 NEXT X
```

Simple but Common Errors Often Made when Learning to Use POSITION Statements

1. Positioning too far
   Example: POS. 40,12
   39 is the maximum across in GRAPHICS 0, and 23 is the maximum down.
2. Abbreviating POS. without the period.
3. Substituting zero for the O in POS.

Graphics Characters Are also Used in GRAPHICS 0

You have probably, by now, either intentionally or accidentally encountered the graphics characters which are also available to you in GRAPHICS 0.



You use graphics characters with normal PRINT commands.

- First type PRINT and open the quotes PRINT "
- Then hold down the CTRL key while you ...
- Type any of the characters shown above.
- Then close the quotes and press the RETURN key.

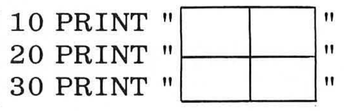
To draw a window, for instance, type this program:

Hold the CTRL key down only when

you're pressing these letters...

```
10 PRINT "QWE"
20 PRINT "ASD"
30 PRINT "ZXC"
```

This will appear like this on the screen.



RUN this program and then add...

```
40 GOTO 10
```

This will produce the windows of a very tall skyscraper.

For drawing boxes with graphics characters, try the following key combinations.

Hold down the CTRL key while

typing these keys:

```
10 PRINT "QRRRRRRE" which looks like this. " ┌──────┐ "
20 PRINT " |       | "
```

These are printed by pressing the SHIFT and ↓ keys *at the same time.* They are the sides of the box.

IMPORTANT..... How to create the sides of boxes.

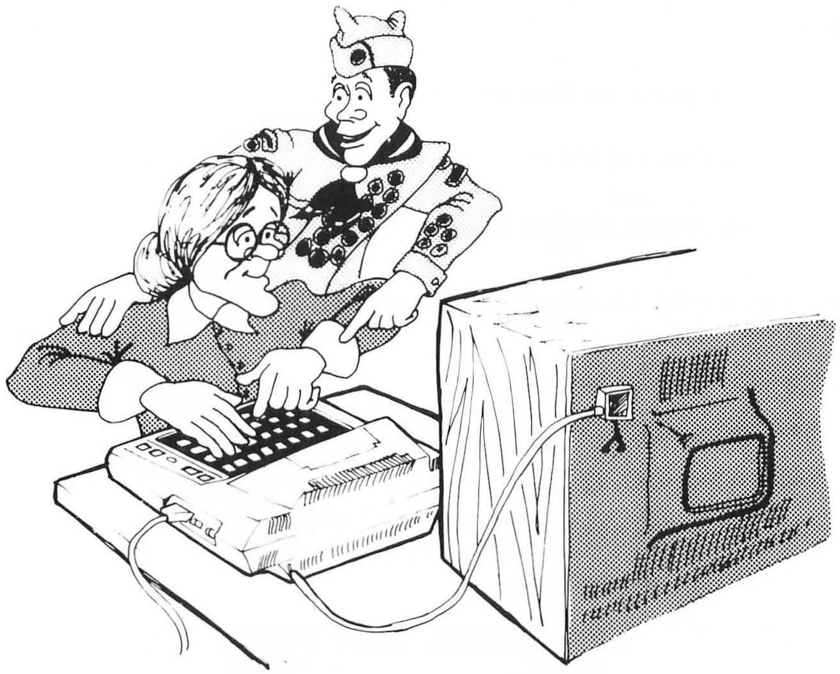
```
30 PRINT "ZRRRRRRC" which looks like this " └──────┘ "
```

Hold the CTRL key while pressing these keys.

RUN this program and try the following one for more flexibility in box size.

```
10 PRINT "WHAT IS THE HEIGHT OF THE BO
X";
20 INPUT H
30 PRINT " ┌──────┐ "
40 FOR BOXHEIGHT=1 TO H
50 PRINT " |       | "
60 NEXT BOXHEIGHT
70 PRINT " └──────┘ "
80 FOR DE=1 TO 200:NEXT DE
90 RUN
```

Helpful Hints for Using Graphics Characters



- A CTRL T makes a good ● in case you want to draw a good dot.
- If you want to experiment with graphics character drawing, you can press the CTRL and CAPS LOWR keys at the same time and it will lock the keyboard into a "graphics character mode." To get the keyboard back to normal, press SYSTEM RESET or SHIFT and CAPS LOWR at the same time.
- Your printer will not print out graphics characters. Graphics characters are for screen displays only.

● Remember, as long as they are in quotes you can also create pictures with mathematical operators, punctuation marks and inverse video characters.

Here's a simple program that uses the

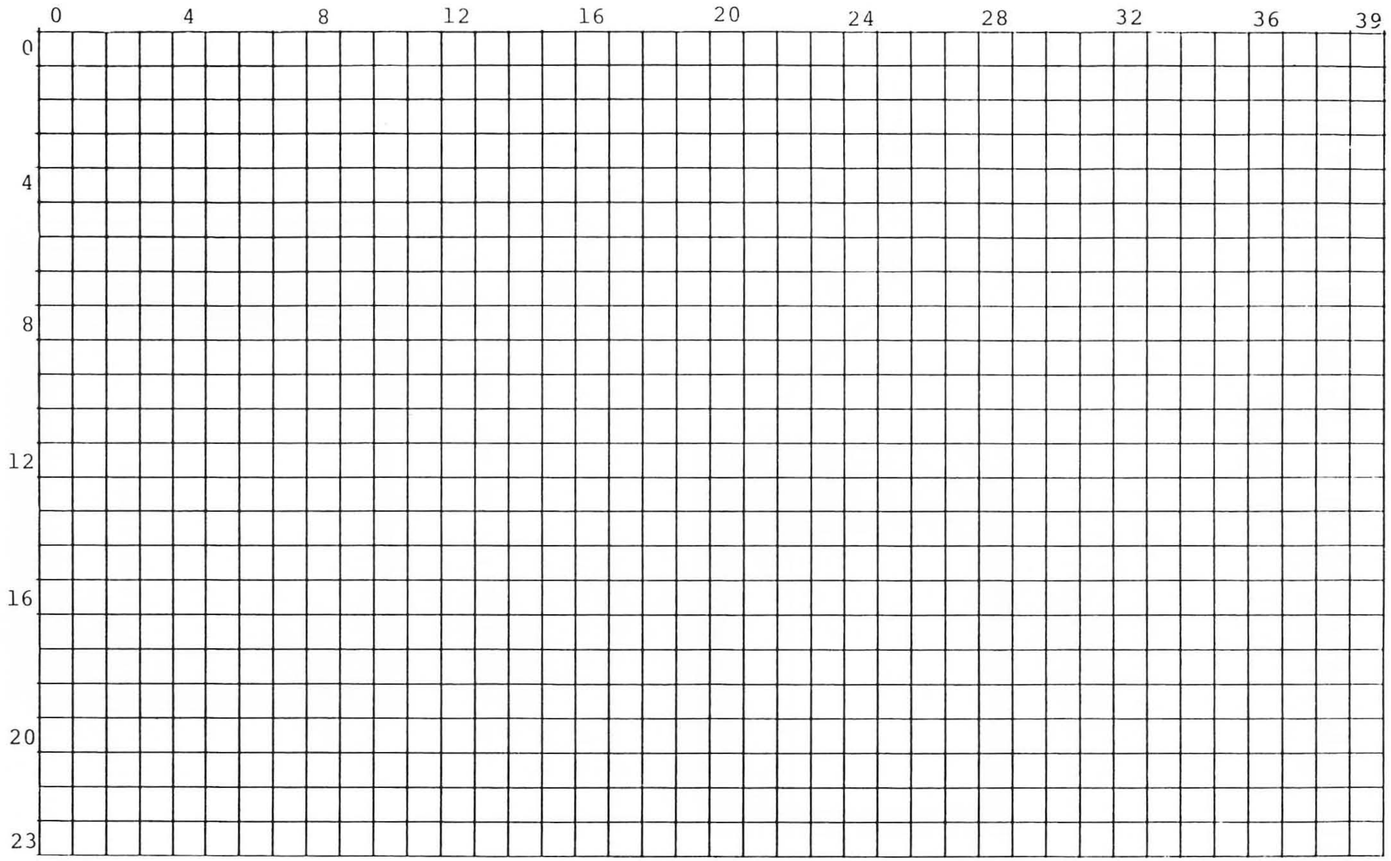
LESS THAN <
and
GREATER THAN >

symbols to give the illusion of movement.

```
5 PRINT "↑": POKE 752,1
10 POSITION 19,12: PRINT "<>"
20 FOR DE=1 TO 50: NEXT DE
30 POSITION 19,12: PRINT "><"
40 FOR DE=1 TO 50: NEXT DE
50 GOTO 10
```

The following chart can be used for laying out displays in the GRAPHICS 0 mode.

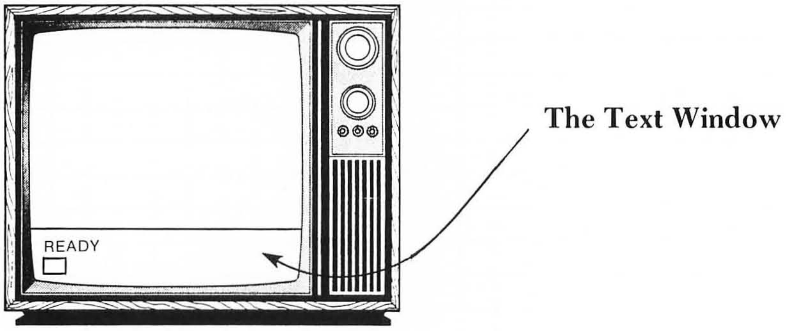
Graphics Mode 0



Notes: ____________________

GRAPHICS 1—For LARGER Text

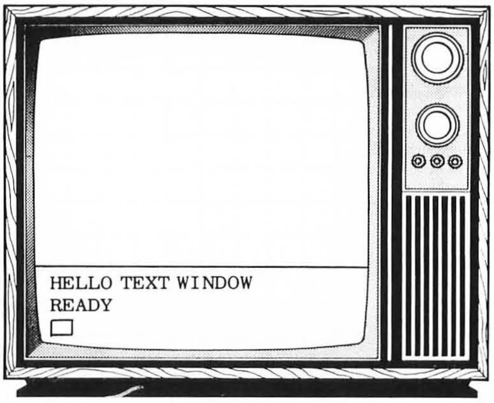
The next text mode is GR. 1. This is *entered* by *typing GR. 1* RETURN . As you can see, you have now dropped down to the bottom of the screen and discovered the "TEXT WINDOW."



The text window is a real pleasure because unlike some of your inlaws, it can be very useful, and you can make it go away quickly and easily!

You can print a line of normal-size text along the top of the text window.

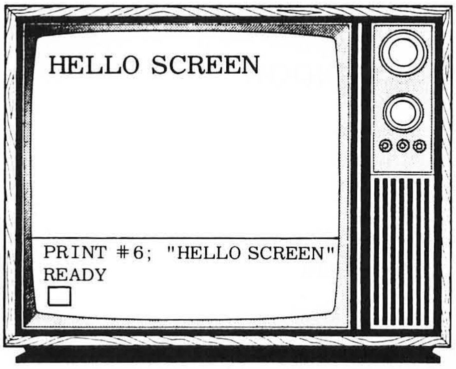
Try... `PRINT "HELLO TEXT WINDOW"`



or you can *use the magic #6;* and see what happens...

```
PRINT #6; "HELLO SCREEN"
```

Your screen should look like this



```
HELLO SCREEN

PRINT #6; "HELLO SCREEN"
READY
```

If it doesn't, you probably missed something... Don't worry; go on to the next text mode and you'll get to the same place quickly.

GRAPHICS 2—For the LARGEST Text

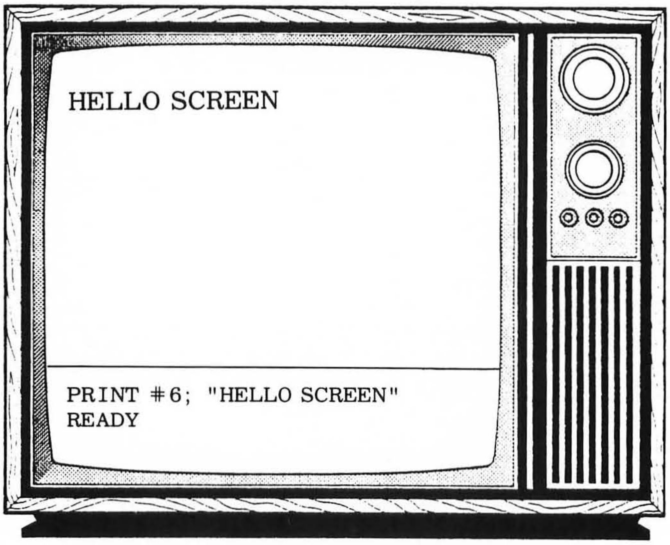
Type...

```
GR. 2 RETURN
PRINT "HELLO TEXT WINDOW"
```

Now try...

```
PRINT #6;"HELLO SCREEN"
```

Your display should look like this...



```
HELLO SCREEN

PRINT #6; "HELLO SCREEN"
READY
```

***** Tricks Ahead *****

Type...

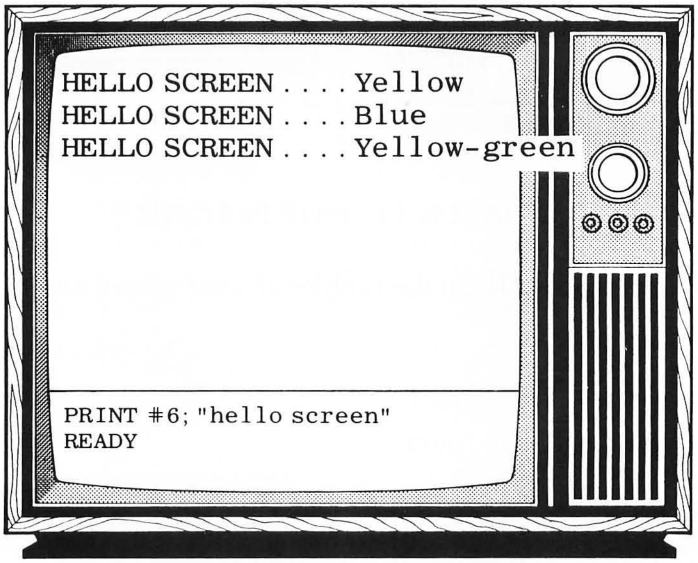
```
PRINT #6;"HELLO SCREEN"
```

(Press the ⏻ key here in order to get in and out of INVERSE VIDEO.)
Now try...

```
PRINT #6;"hello screen"
```

(Press CAPS LOWR here and SHIFT CAPS LOWR here to get in and out of lower lower case.)

If all went well and your color is properly adjusted, your display should look like this...



```
HELLO SCREEN . . . . Yellow
HELLO SCREEN . . . . Blue
HELLO SCREEN . . . . Yellow-green

PRINT #6; "hello screen"
READY
```

If you have nothing like this, then press SYSTEM RESET and go back to where you typed GR.2 RETURN ..

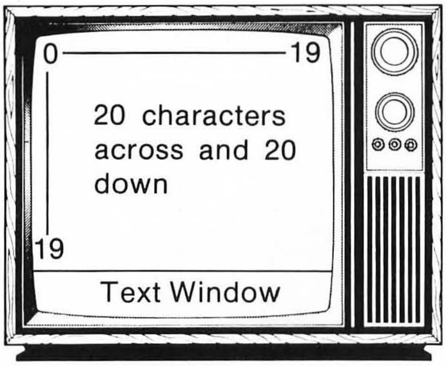
Don't skip over this section without learning it or you'll never amount to anything.

NOTE: You can get one more color by using a combination of inverse video and lower case. The section on COLOR and PEEKS and POKES will explain how to make your large text any color you want.

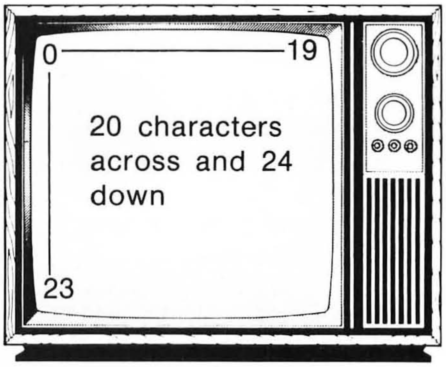
Using POSITION in GRAPHICS Modes 1 and 2

You can also use POSITION in GRAPHICS 1 and 2, but you have to adjust for fewer points across and down. If you recall, there are 40 spaces across and 24 spaces down in GRAPHICS 0.

In GRAPHICS 1, you can place...



or

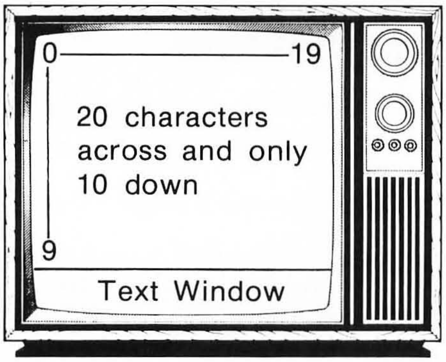


without the text window

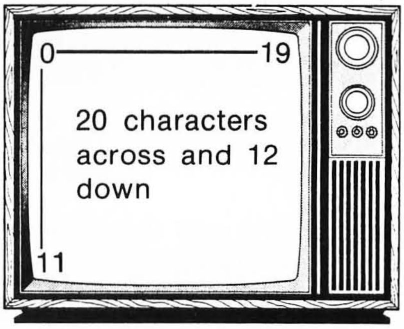
```
GR.1: POS.17,11: PRINT#6; "HI"
```

would place the word "HI" in the middle of your screen and to the extreme right.

In GRAPHICS 2, you have room for...



or



without the text window

Remember, when using GRAPHICS Mode 1 or 2 and you want to display *large text, use* a #6; *between PRINT* and your *first quotation mark.* Without a #6; you will PRINT in the text window at the bottom of the screen.

```
10 GRAPHICS 2
20 PRINT #6; "THIS IS LARGE TEXT"
30 POKE 752,1: REM TURNS CURSOR OFF
40 PRINT "THIS IS SMALL TEXT, BUT IT'S
 O.K. ALSO"
50 FOR DELAY=1 TO 300: NEXT DELAY
60 PRINT
70 FOR DELAY=1 TO 300: NEXT DELAY
80 GOTO 40
```

Getting Rid of the Text Window

** This can only be done in the programming mode. **

To get rid of a text window, add 16 to whatever GRAPHICS mode you are in.

+16 means no text window

Type... NEW RETURN

```
10 GRAPHICS 2+16
20 PRINT #6; "THIS IS LARGE TEXT"
30 PRINT #6; "WITH NO TEXT WINDOW"
40 GOTO 40: REM THIS LINE KEEPS THE DI
SPLAY ON THE SCREEN
```

NOTE: You may see someone's program listings which show something like...

```
10 GR.18
```

This is the same as GR. 2 + 16. Using the standard GRAPHICS mode number and adding 16 seems to be just as easy and is less confusing.

Here's a flashy program which combines GR. 0 and GR. 2 with a few other tricks...

```
10 PRINT "↰": DIM NAME$ (20) : POSITION 2,
7
20 PRINT "TYPE IN YOUR NAME AND PRESS
THE RETURN KEY"
30 POSITION 4,12: PRINT "NAME";: INPUT N
AME$
40 GRAPHICS 2+16
50 POSITION (20-LEN(NAME$))/2,4: REM CE
NTERS THE NAME
60 PRINT #6;NAME$
70 POSITION 1,7: PRINT #6; "IS A FLASHY
PERSON": REM TYPE "IS A FLASHY PERSON" I
N INVERSE VIDEO
80 FOR FLIP=0 TO 20
90 FOR FLASH=0 TO 14
100 SETCOLOR 0,0,FLASH: SOUND 0,FLASH,1
0,10
110 NEXT FLASH
120 FOR DELAY=1 TO 20: NEXT DELAY
130 NEXT FLIP
140 SETCOLOR 0,0,14: SOUND 0,0,0,0
150 FOR DELAY=1 TO 1000: NEXT DELAY
160 SETCOLOR 0,0,0: SETCOLOR 2,0,0
170 FOR DELAY=1 TO 800: NEXT DELAY
180 RUN
```

NOTE: You can save line 50 in order to center any GR.2 input called NAME$

and

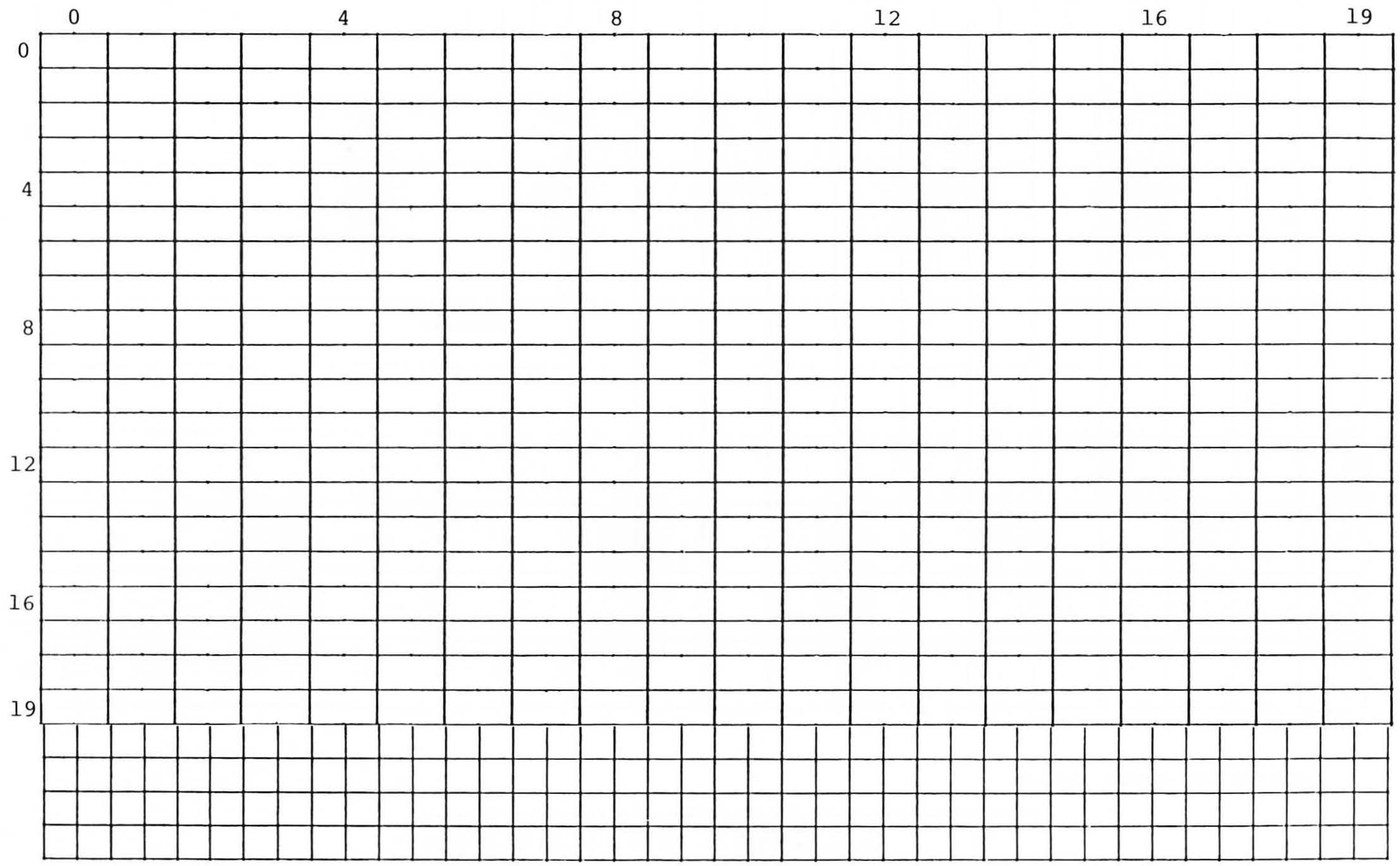
save lines 80 through 130 to make the input flash with sound.

Further details on how this type of program works and what SETCOLOR does can be found in the section on COLOR.

The following four pages will help you arrange GR. 1 and GR. 2 displays with or without text windows. Notice that although the width is the same, Graphics 1 characters are only half as tall as Graphics 2 characters.

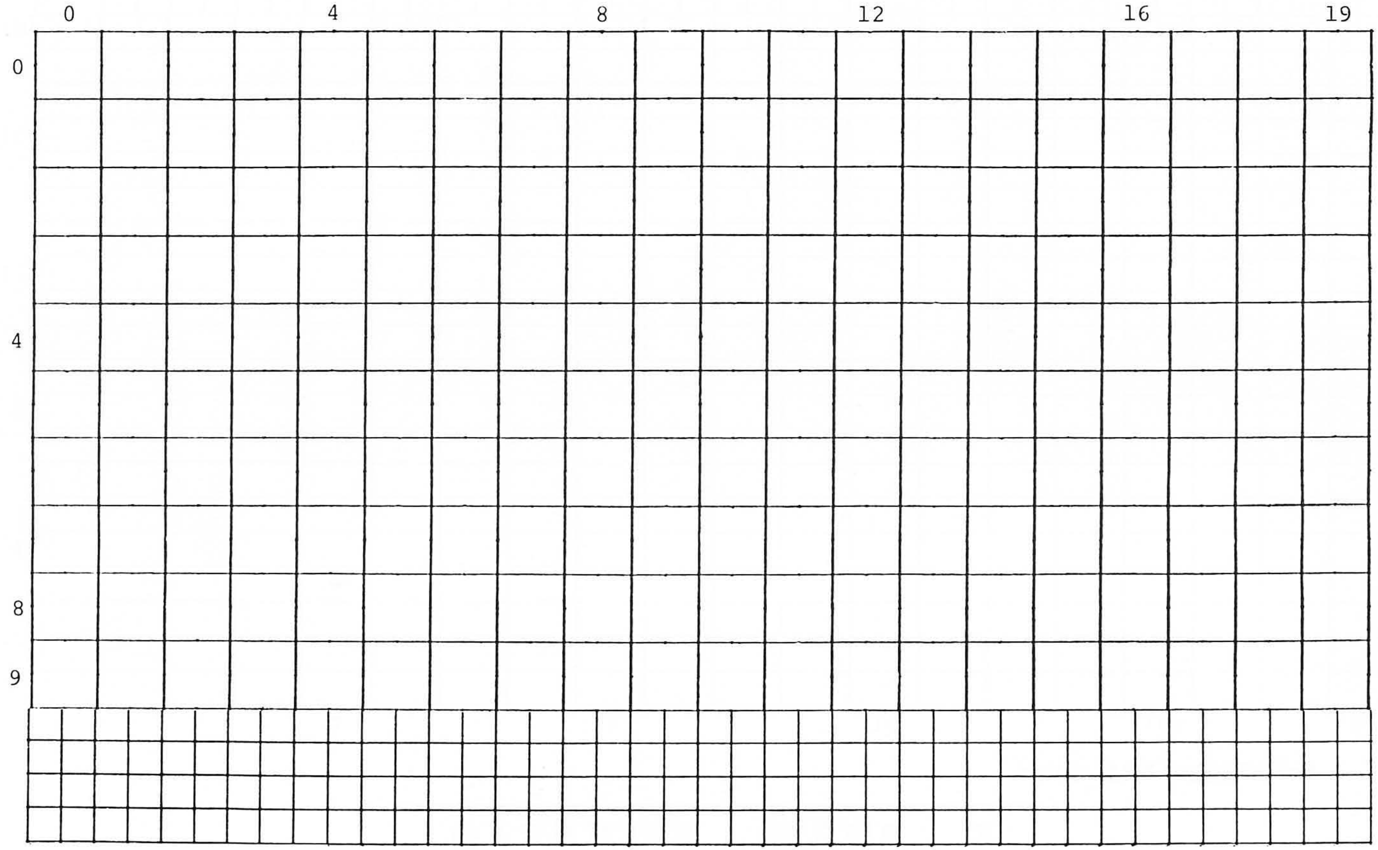
Graphics Mode 1

with Text Window

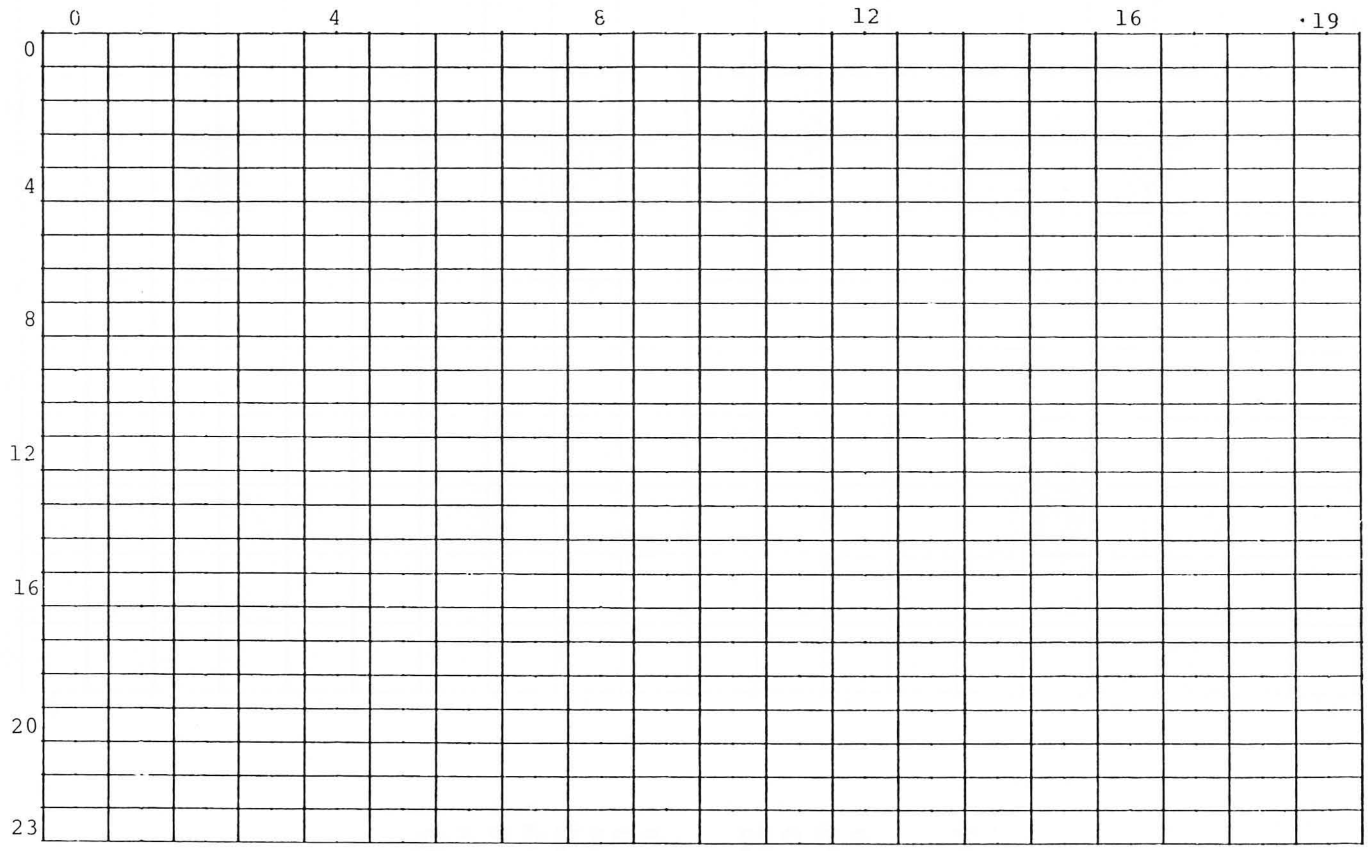


Graphics Mode 2

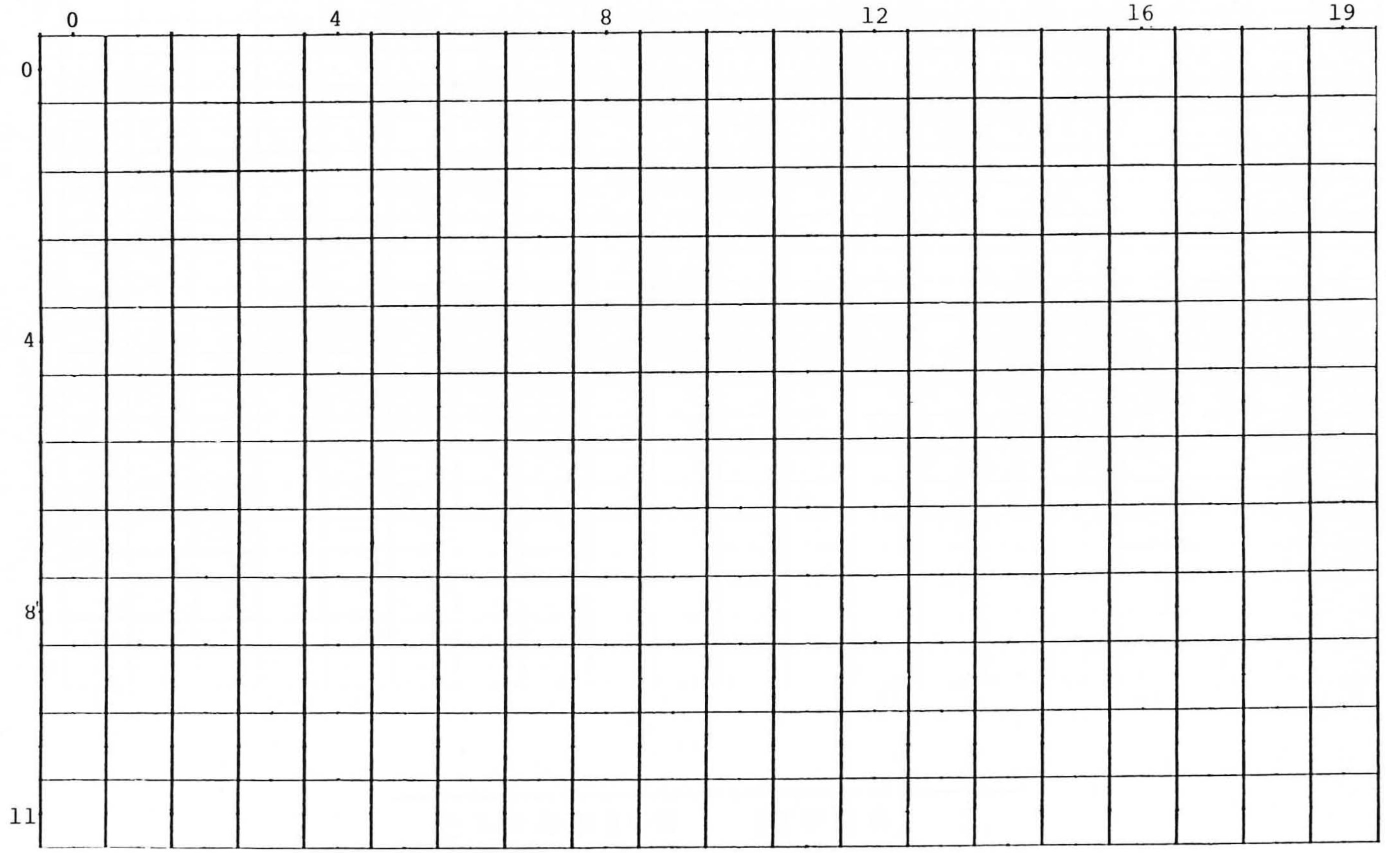
with Text Window



Graphics Mode 1

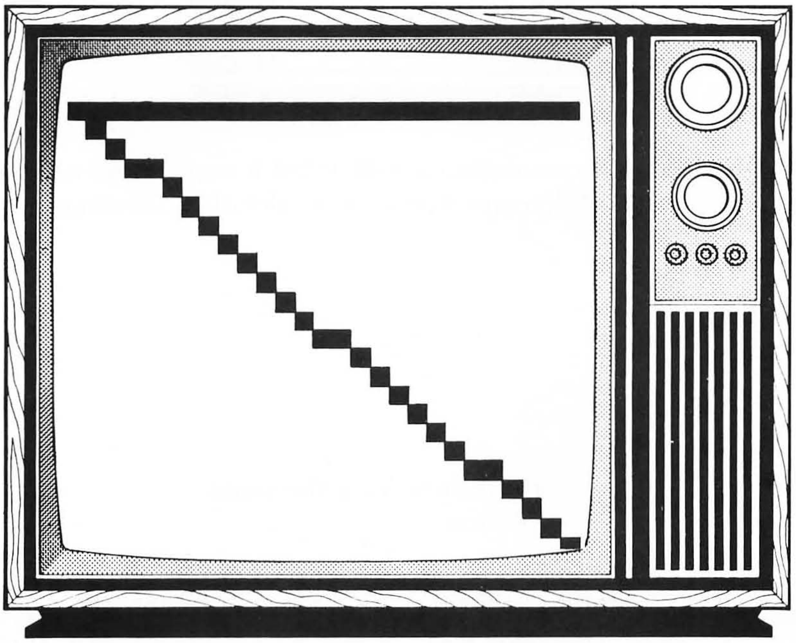


Graphics Mode 2



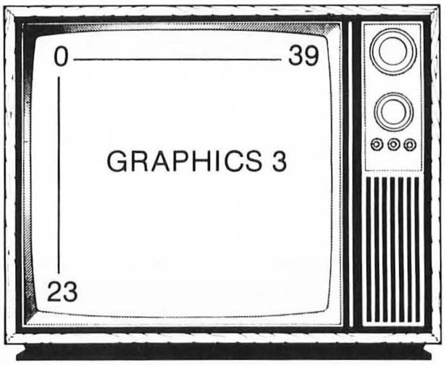
GRAPHICS Modes 3 Through 8

These are the true graphics modes since you usually display blocks of color instead of words or letters.



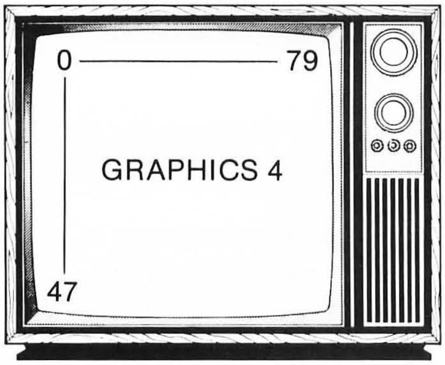
As a general rule, when you go higher in GRAPHICS modes, you get more screen dots (called pixels) and therefore the possibility of a more detailed drawing.

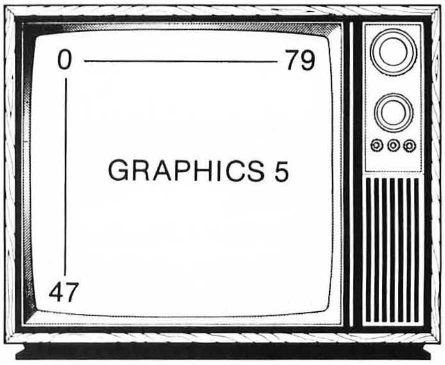
Of course that formula is too simple for computer designers, so here's the basic idea of the trade-offs involved in using each of the graphics modes...



GR.3 has the same resolution as GR. 0 but it uses blocks of color instead of characters. You cannot make very detailed drawings using GR.3.

Both of these have the same resolution

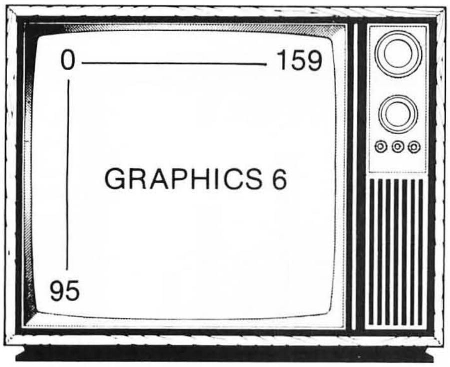




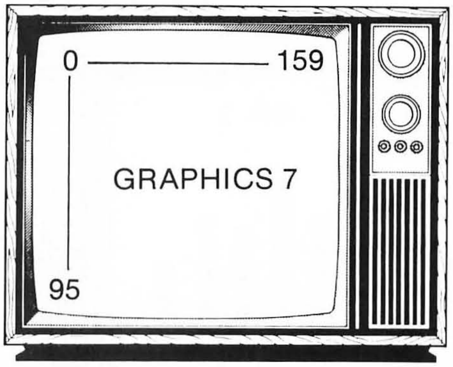
This one uses fewer colors (2) but less memory. Use it when you need to conserve memory.

This one has more colors but uses twice as much memory.

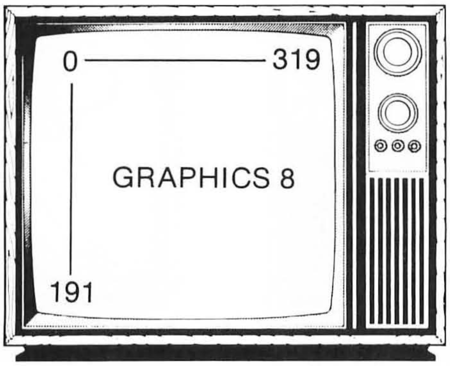
Both of these have the same moderately high resolution.



Again, half the colors and half the memory used.

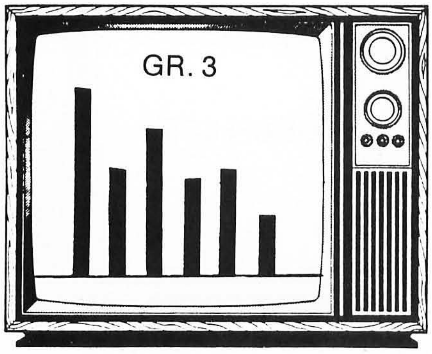


This has pretty good resolution and can use 4 colors. It uses twice as much memory as GRAPHICS 6, however.

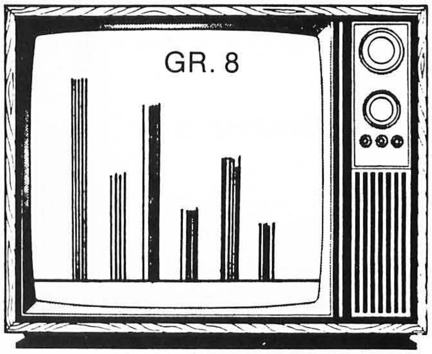


Very high resolution, 2 colors, lots of memory used. Use this where detailed drawings are important.

As a general rule, consider what type of display you need, and make it easy on yourself and the computer. For instance, you should probably use GR. 3 to draw a bar chart instead of GR. 8.

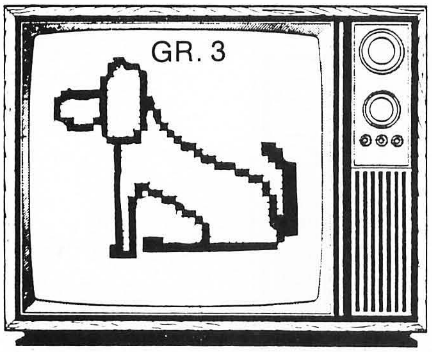


Lots of colors; not much memory used; and not too many points to plot.

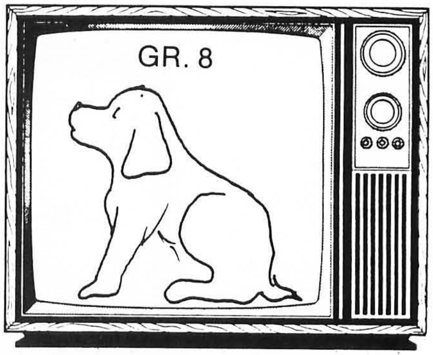


Only one color is used. More lines had to be drawn. More memory was used.

On the other hand, if you wanted to draw a Springer Spaniel...



Would you let this dog fetch your slippers?



A fine specimen, if I do say so.

In summary...If you want to draw an ugly bar chart, use GRAPHICS 8. If you want to draw an ugly dog, use GRAPHICS 3.

Time to Fiddle Around...

You can do this in direct (immediate) mode with no program lines.

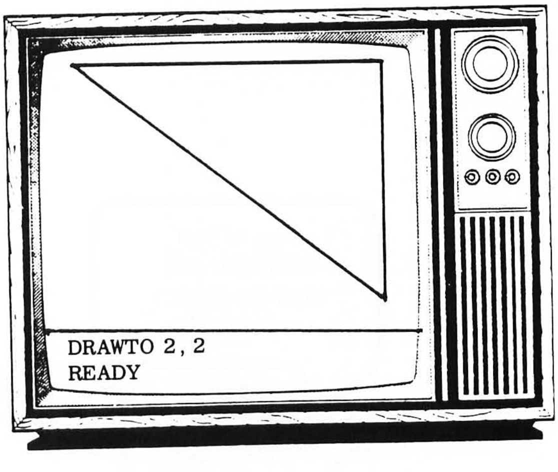
```
GR. 3
COLOR 1
PLOT 2, 2
DRAWTO 37, 2
DRAWTO 37, 18

DRAWTO 2, 2
```

(Selects the preset color)
(PLOT means put a dot of color or start drawing from these points. It is similar to POSITION in the text modes.)
(DRAWTO means start at the last PLOT or DRAWTO point and put a line of color here.)

This is the best diagonal line the computer can draw in GRAPHICS 3.

On to the finer things in life...

```
GR. 7
COLOR 1
PLOT 2, 2
DRAWTO 150, 2
DRAWTO 150, 70
DRAWTO 2, 2
```

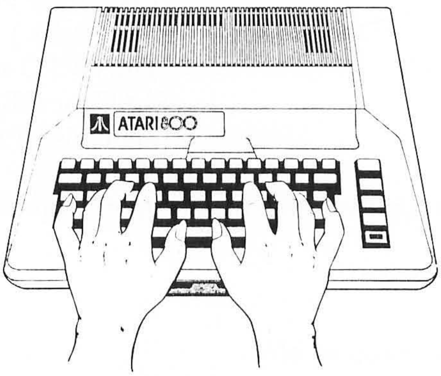


```
DRAWTO 2, 2
READY
```

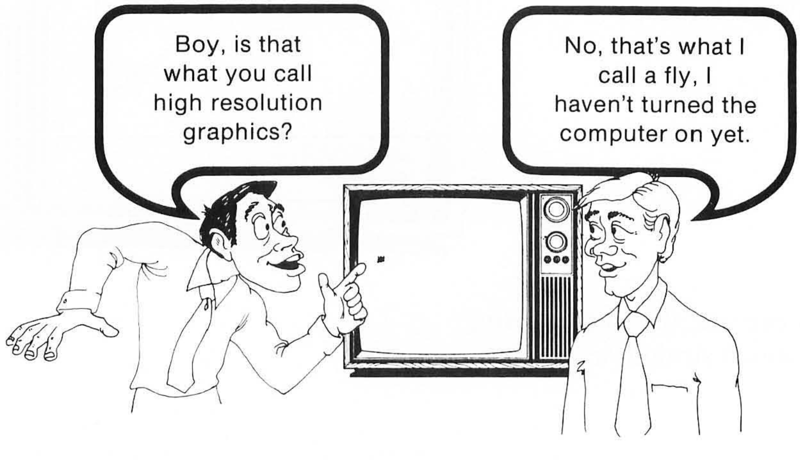
As you can see, the diagonal line is now much straighter.

This is a very fine mode you've gotten us into Ollie!

```
GR. 8
COLOR 1
PLOT 5, 5
DRAWTO 300, 5
DRAWTO 300, 150
DRAWTO 5, 5
```



As you can see, the higher the resolution, the finer the detail that is possible.


Boy, is that what you call high resolution graphics?
No, that's what I call a fly, I haven't turned the computer on yet.

Graphics Tips for Beginners

Even though the Text Window is very handy for experimenting in the immediate mode and for easily displaying words and graphics at the same time, you may often want to eliminate it from your programs.

So remember, in the programming mode you can eliminate the text window by adding 16 to GRAPHICS 1 through 8. For example,

```
10 GR.2+16        OR        10 GR.8+16
```

When using a GRAPHICS mode without a Text Window, you might also forget to instruct the computer to keep the display on the screen and not go back to READY

One way to maintain your display on the screen, is to make the last line an endless loop.....

Try the Following Programs:

`10 GRAPHICS 7`
`20 COLOR 1`
`30 PLOT 2,2: DRAWTO 150,80`
	After the program is run, "READY" appears in the text window.

`10 GRAPHICS 7+16`
`20 COLOR 1`
`30 PLOT 2,2: DRAWTO 150,80`
	By adding 16 to GRAPHICS 7 you eliminate the text window, but the program is executed so fast you can hardly see it.

`10 GRAPHICS 7+16`
`20 COLOR 1`
`30 PLOT 2,2: DRAWTO 150,80`
`40 GOTO 40`
	Line 40 creates an ENDLESS LOOP which prevents the program from ending.

`10 GRAPHICS 7+16`
`20 COLOR 1`
`30 PLOT 2,2: DRAWTO 150,80`
`40 FOR DE=1 TO 500: NEXT DE`
	This new line 40 creates a slight delay before the program ends.

If you just want to put up a display for a little while, then go on to another part of the program—put a delay loop in the program. Try the following program which switches between GR. 7 and GR. Ø.

```
10 PRINT "↖": PRINT "TYPE A NUMBER FROM
 1 TO 800 AND PRESS RETURN";: INPUT HOW
LONG
20 FOR EX=1 TO 3
30 GRAPHICS 7: POKE 710,0
40 COLOR 1
50 PLOT 2,2: DRAWTO 158,2
60 DRAWTO 158,80
70 DRAWTO 2,2: POKE 752,1: PRINT "THIS I
S THE GR. 7 DELAY"
80 FOR DELAY=1 TO HOWLONG: NEXT DELAY
90 GRAPHICS 0: POKE 752,1
100 SETCOLOR 2,8,6: SETCOLOR 1,0,12
110 POSITION 8,11
120 PRINT "THIS IS THE GR. 0 DELAY"
130 FOR DELAY=1 TO HOWLONG: NEXT DELAY
140 NEXT EX
150 RUN
```

The most frequent ERROR message you will encounter using different graphics modes is an ERROR 141. It means in simple colloquial terms..."Now you've gone too far buster (or bustress)." You have positioned, plotted, or drawn to a point beyond the range of the graphics mode that you are working in. BACK OFF!

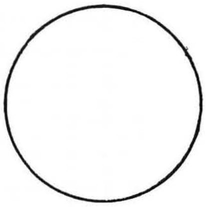
```
GR. 3
COLOR 1
PLOT 45,2
```

Now tell the computer you're sorry and try...

```
PLOT 39,2
```

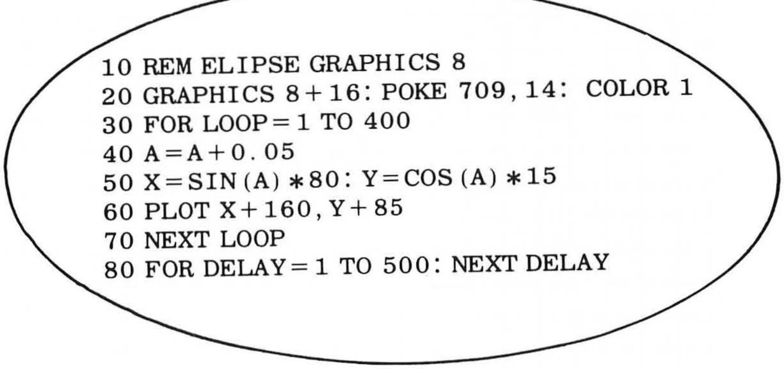
Since figuring out how to draw circles in BASIC sometimes involves more work than the results appear to be worth, I have included a couple of circle routines for you to experiment with. It is probably a good idea to start keeping files of various routines you run across in order to select the most efficient one for each program you are writing.

```
10 REM BIG CIRCLE
20 GRAPHICS 7+16: COLOR 1
30 FOR LOOP=1 TO 400
40 A=A+0.05
50 X=SIN(A)*50: Y=COS(A)*45
60 PLOT X+80,Y+45
70 NEXT LOOP
80 FOR DELAY=1 TO 500: NEXT DELAY
```



```
10 REM BIG CIRCLE GRAPHICS 8
20 GRAPHICS 8+16: POKE 709,14:  COLOR 1
30 FOR LOOP=1 TO 400
40 A=A+0.05
50 X=SIN(A)*50: Y=COS(A)*45
60 PLOT X+160,Y+85
70 NEXT LOOP
80 FOR DELAY=1 TO 500: NEXT DELAY
```

Slight changes in the preceding program produce an ellipse instead of a circle.

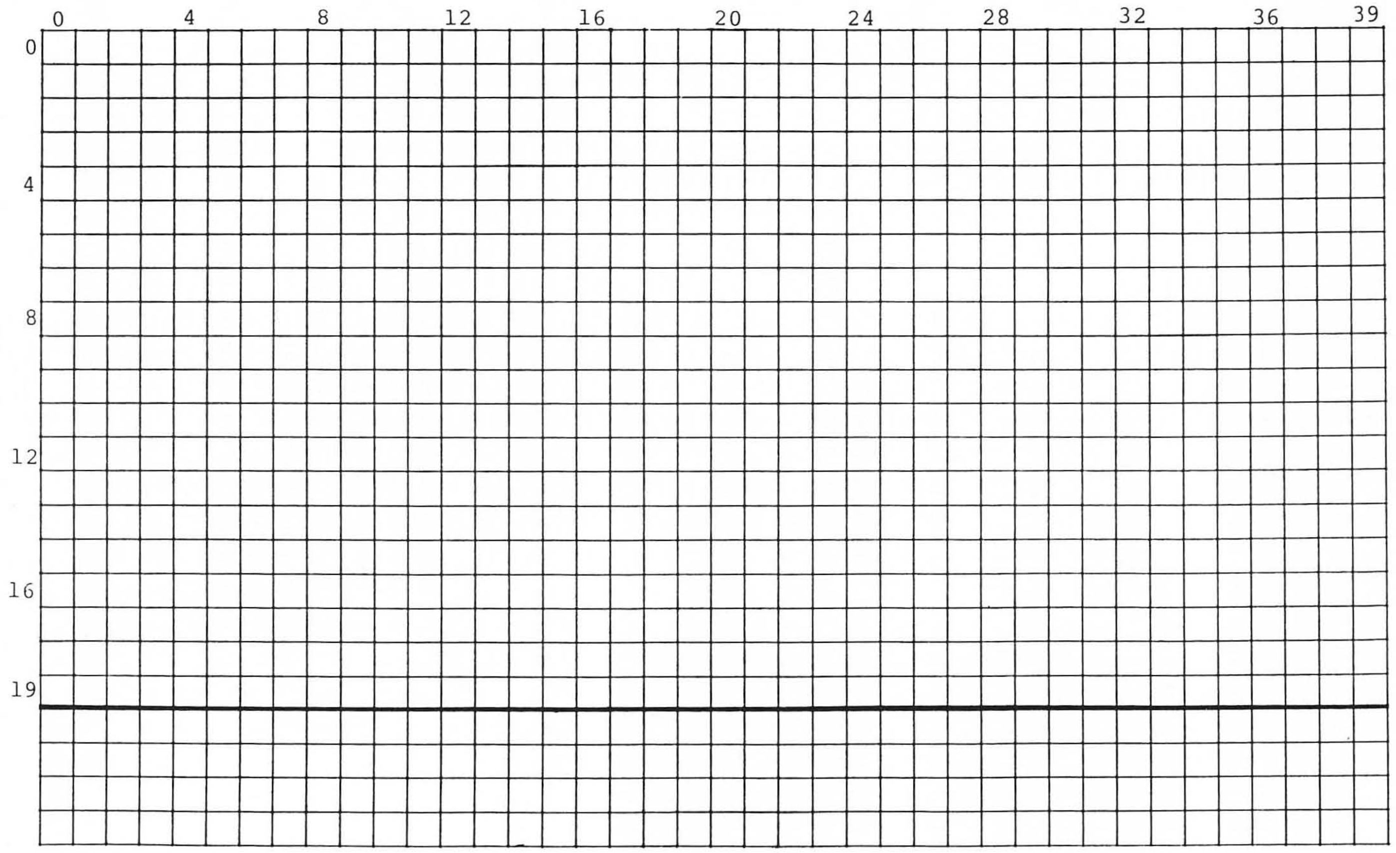


```
10 REM ELIPSE GRAPHICS 8
20 GRAPHICS 8+16: POKE 709,14:  COLOR 1
30 FOR LOOP=1 TO 400
40 A=A+0.05
50 X=SIN(A)*80: Y=COS(A)*15
60 PLOT X+160,Y+85
70 NEXT LOOP
80 FOR DELAY=1 TO 500: NEXT DELAY
```

The following charts will help you design displays in GRAPHICS 3 through 8.

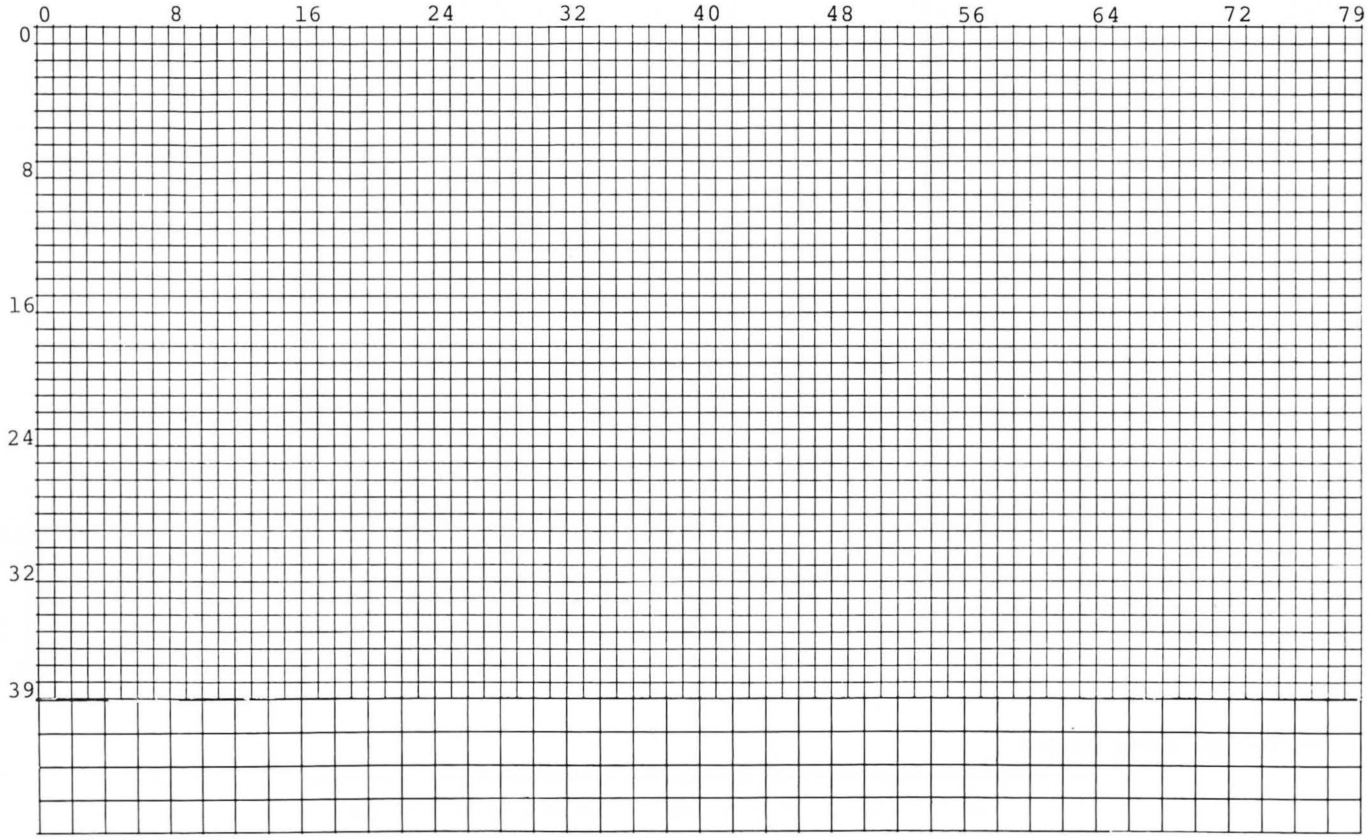
Graphics Mode 3

with Text Window



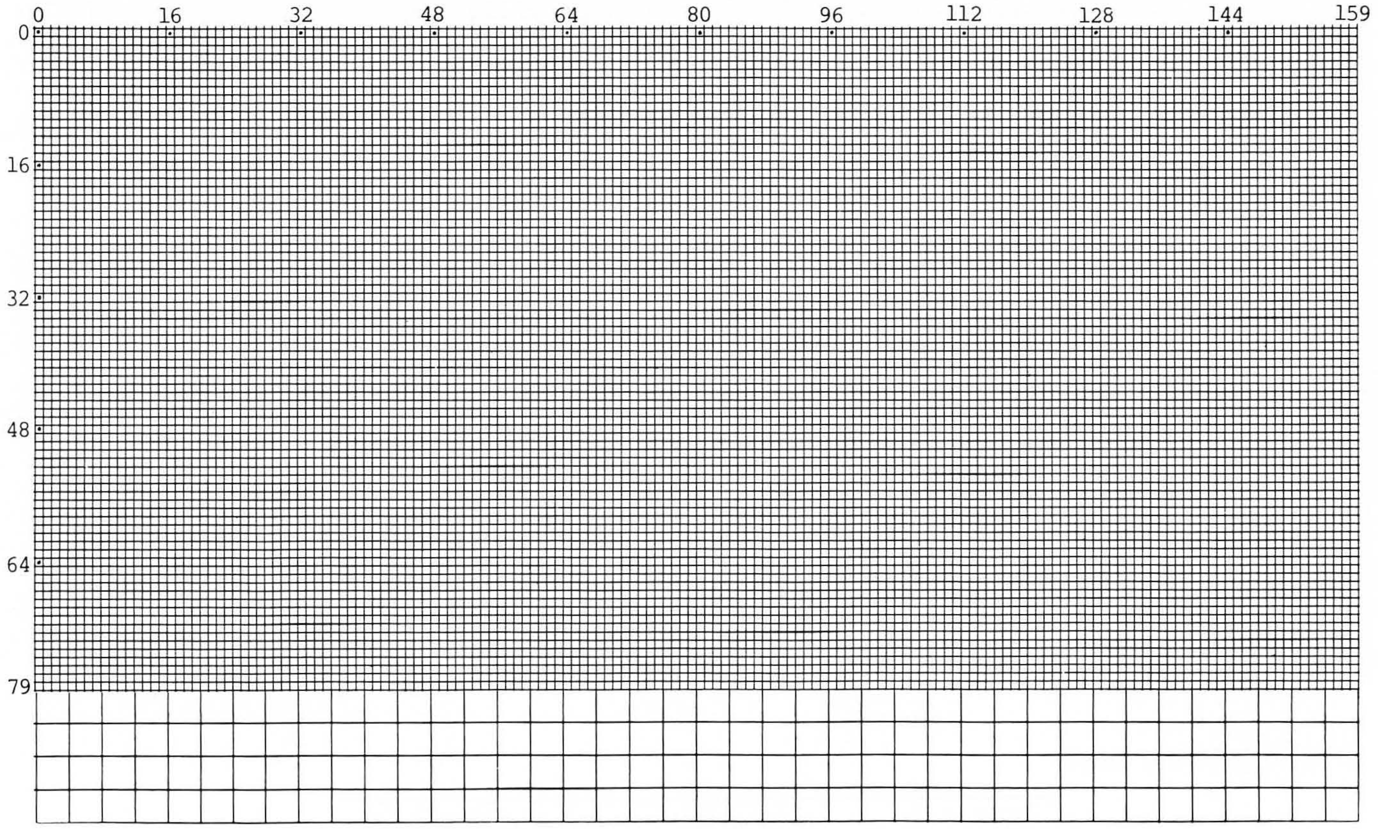
Graphics Mode 4 or 5

with Text Window



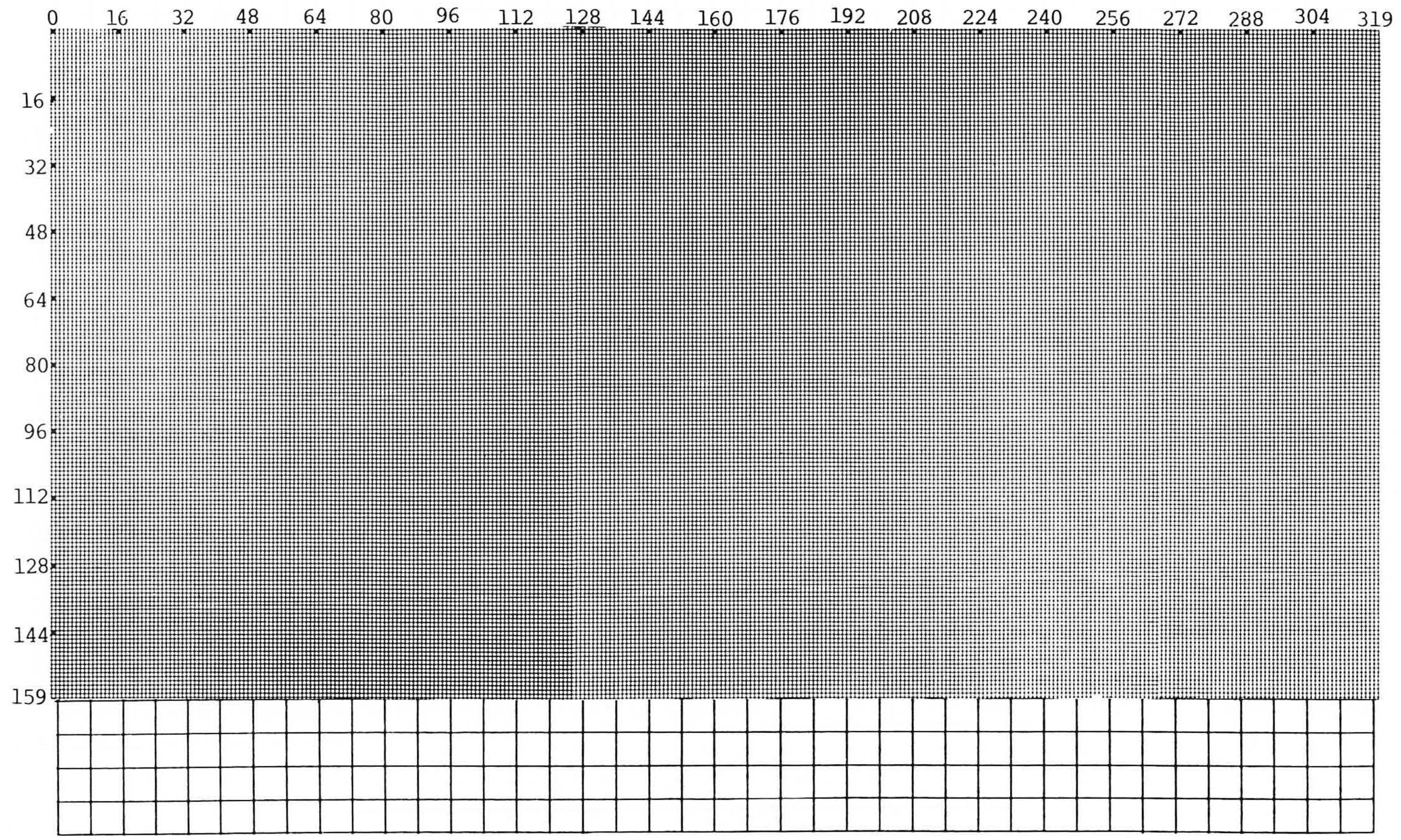
Graphics Mode 6 or 7

with Text Window

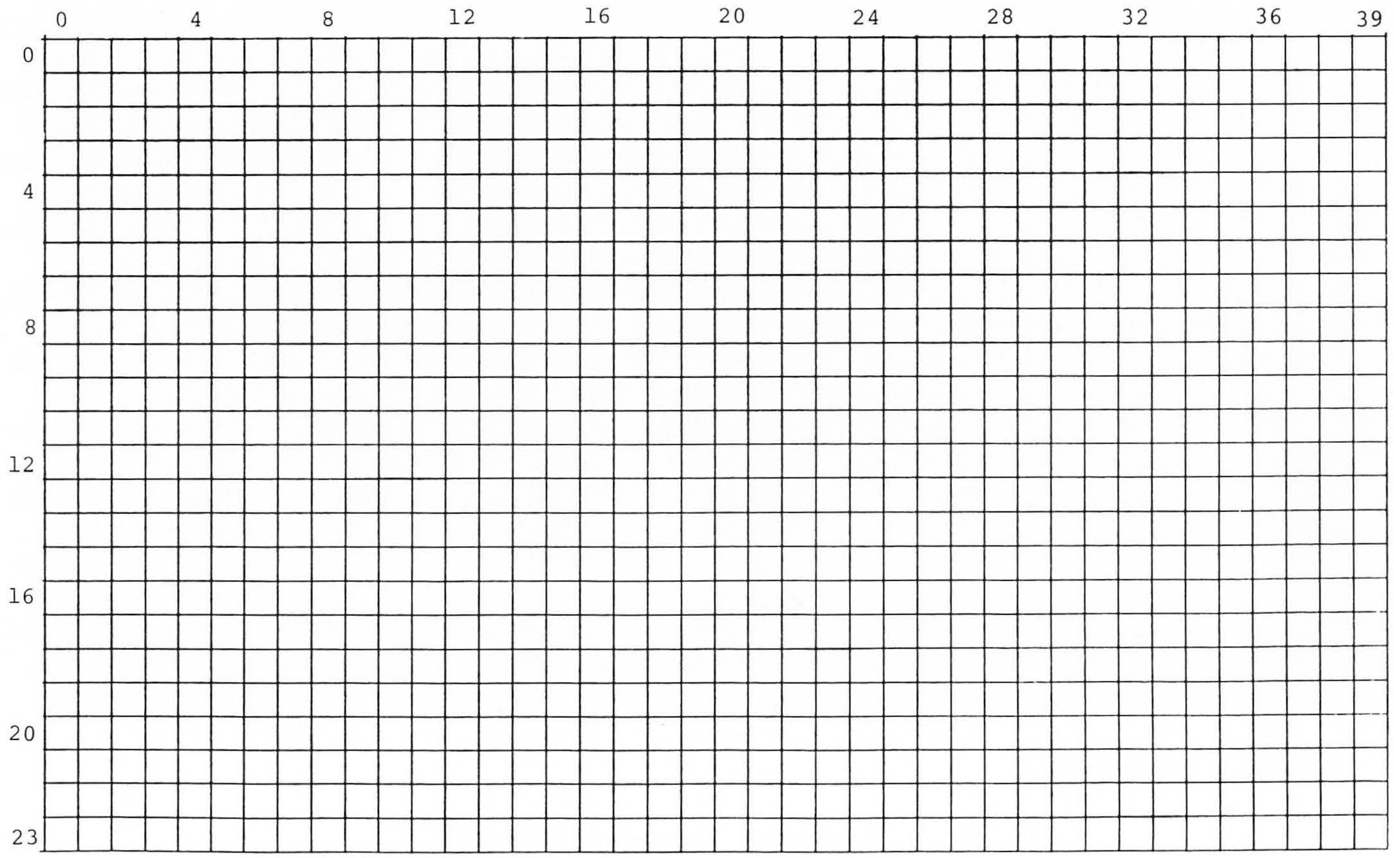


Graphics Mode 8

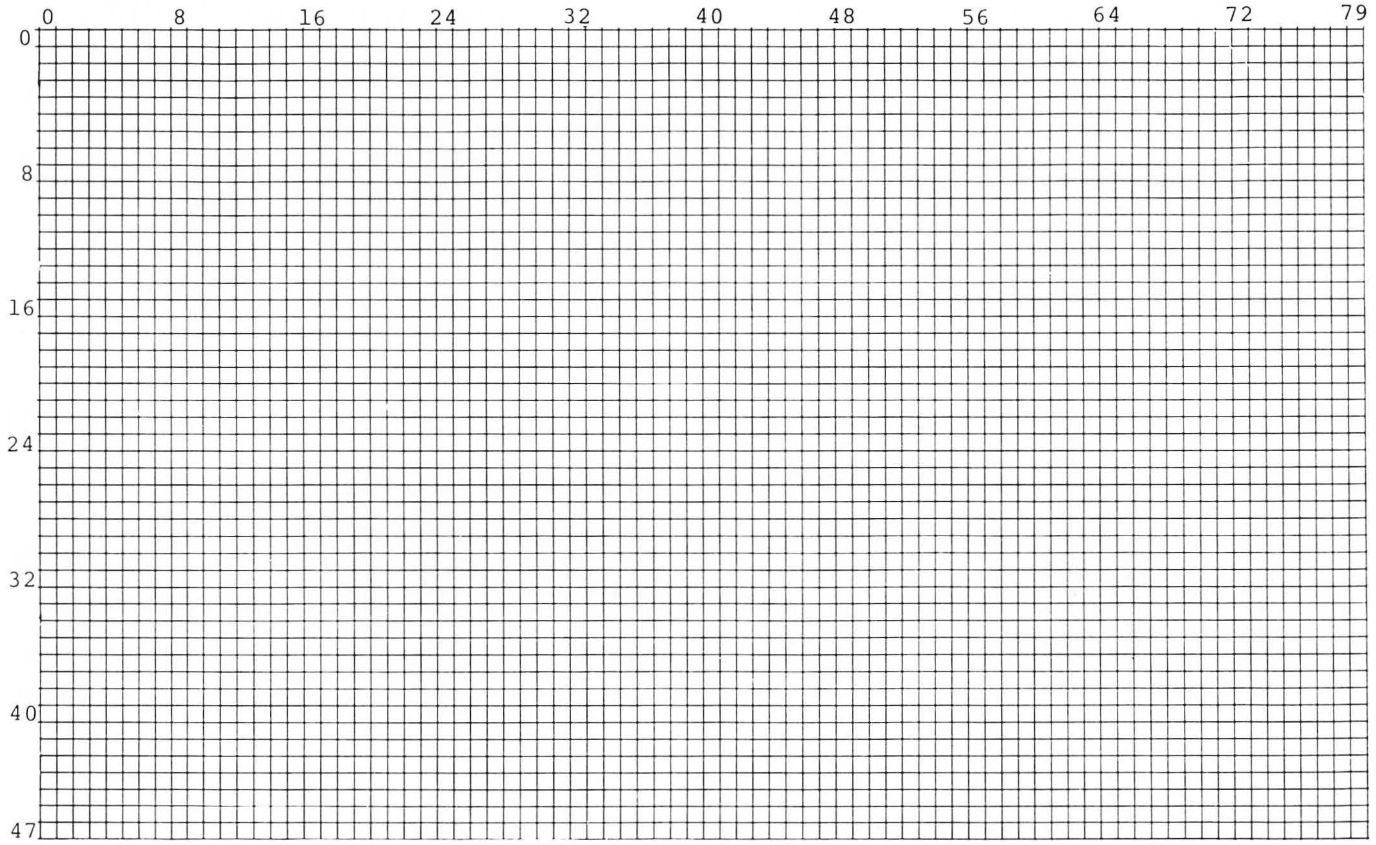
with Text Window



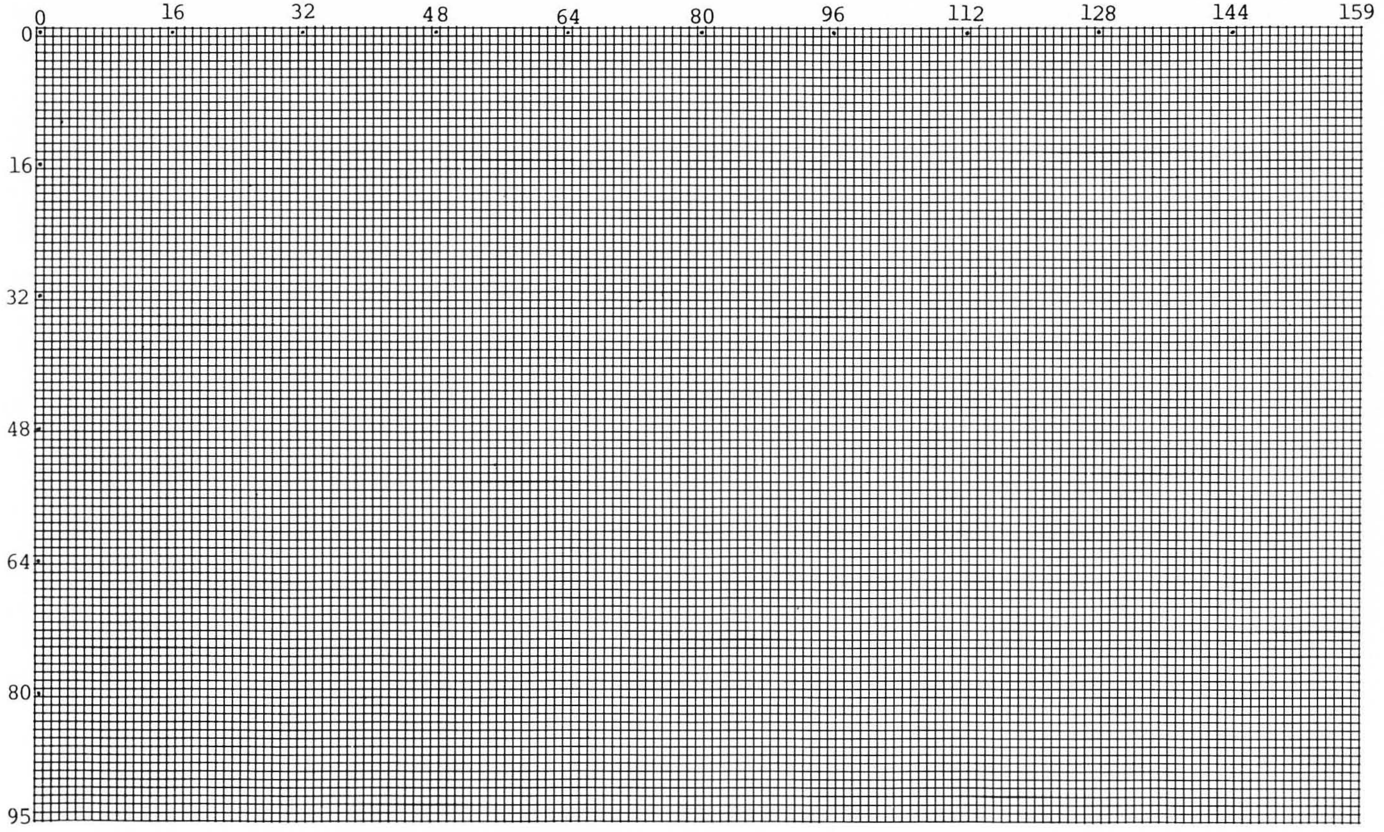
Graphics Mode 3



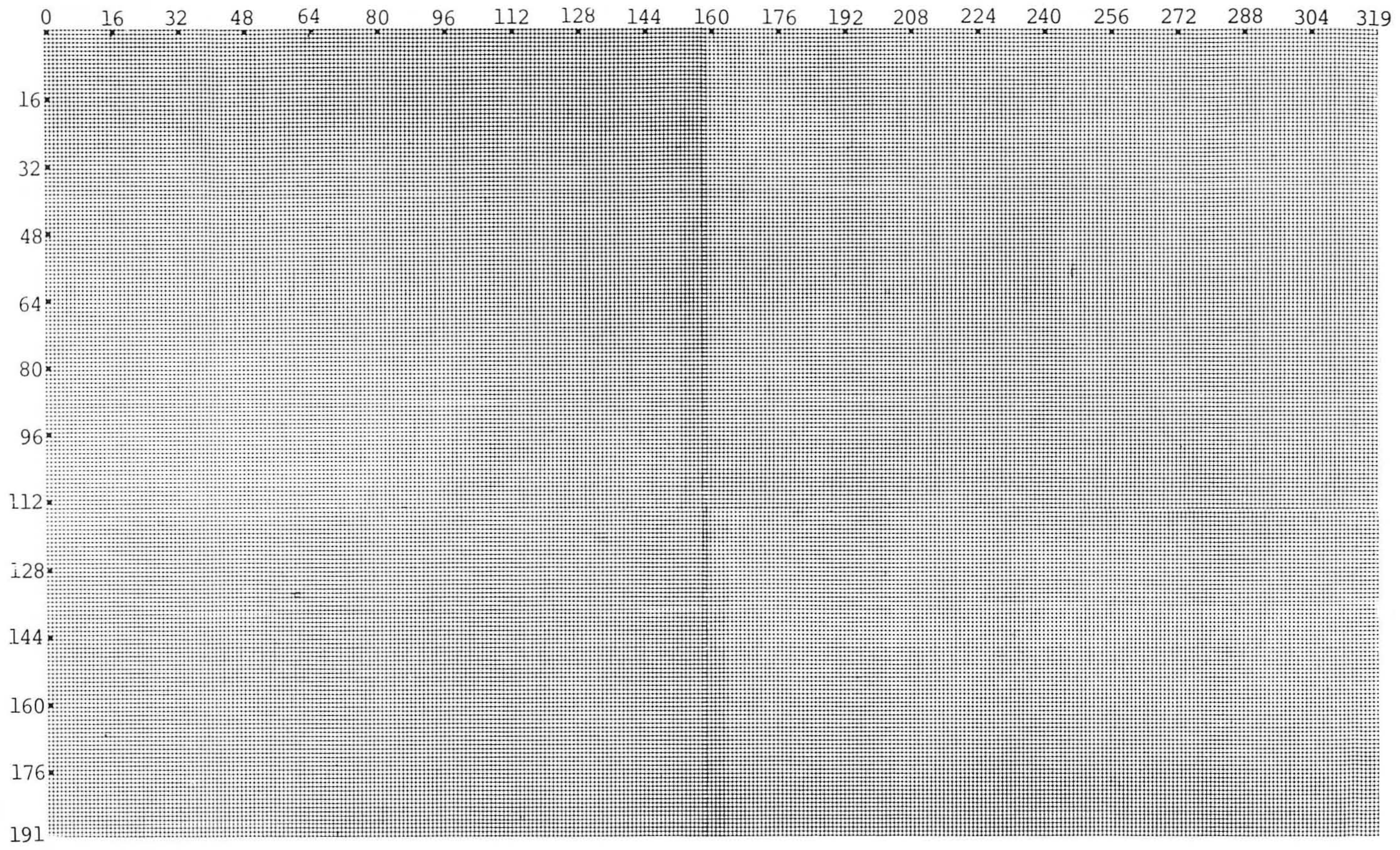
Graphics Mode 4 or 5



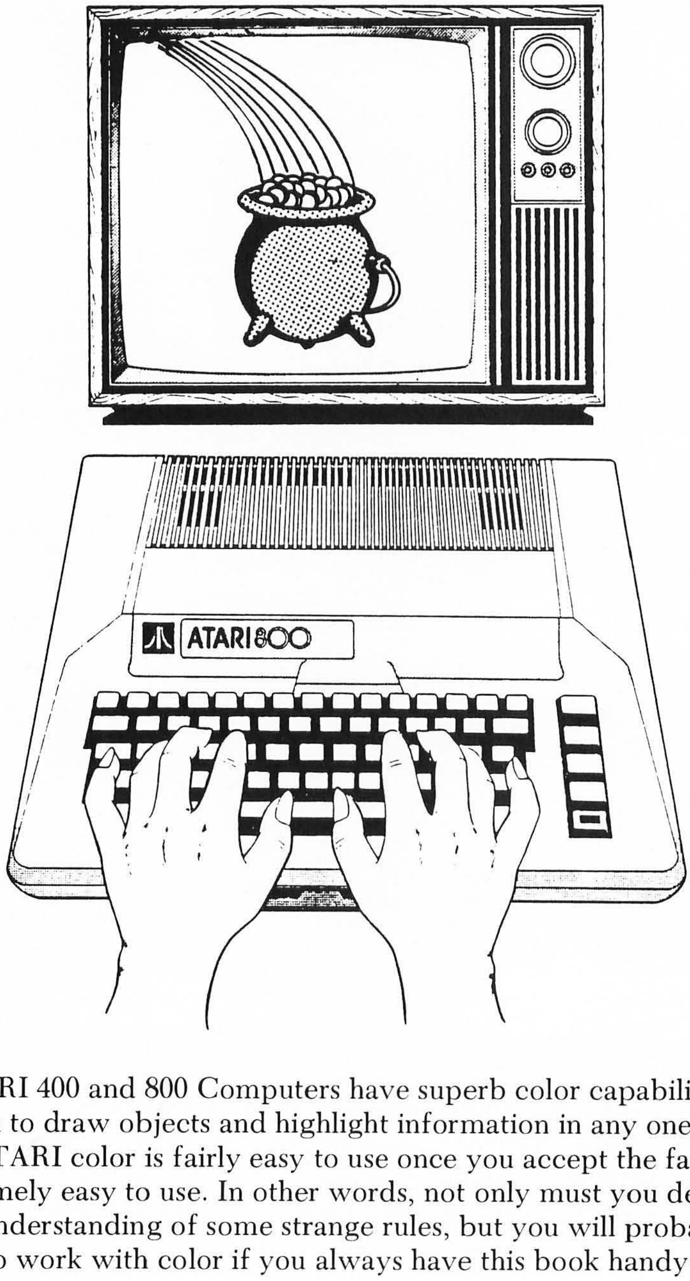
Graphics Mode 6 or 7



Graphics Mode 8



ATARI COLOR



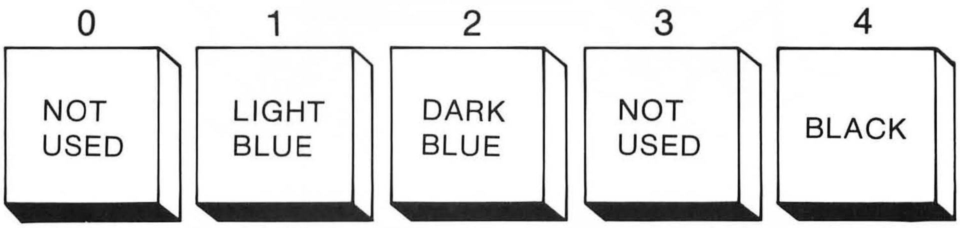
The ATARI 400 and 800 Computers have superb color capabilities which allow you to draw objects and highlight information in any one of 128 colors. ATARI color is fairly easy to use once you accept the fact that it is not extremely easy to use. In other words, not only must you develop a general understanding of some strange rules, but you will probably find it easier to work with color if you always have this book handy.

SETCOLOR

For a more thorough understanding of ATARI color, you need to be familiar with the use of two statements, SETCOLOR and COLOR. Since only SETCOLOR (abbreviated SE.) is used in both the TEXT and GRAPHICS modes, we can look at that first and forget about COLOR for a moment.

Using SETCOLOR in GR. 0

Your computer has color registers for storing colors. The specific colors stored are determined by the TEXT or GRAPHICS mode you are in. Usually 3 to 5 colors are stored in color registers. For example, when you turn on your computer and automatically enter Mode 0, the color registers are filled as follows...



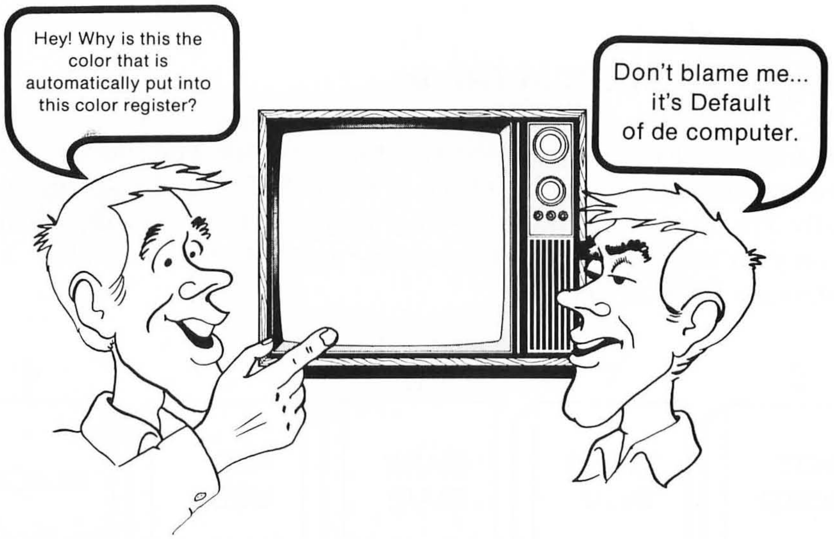
- A light blue is placed in Register 1 to control the brightness or luminance of the characters, such as the word "READY" in the upper left hand corner.
- A dark blue is placed in Register 2, which dictates the color of your screen or "background color."
- A black is placed into Register 4, which controls the color of the "border" around your background.
- Since Registers 0 and 3 are not used in this mode, no colors are placed into these registers.

The colors which are automatically placed in the color registers are called "default colors."

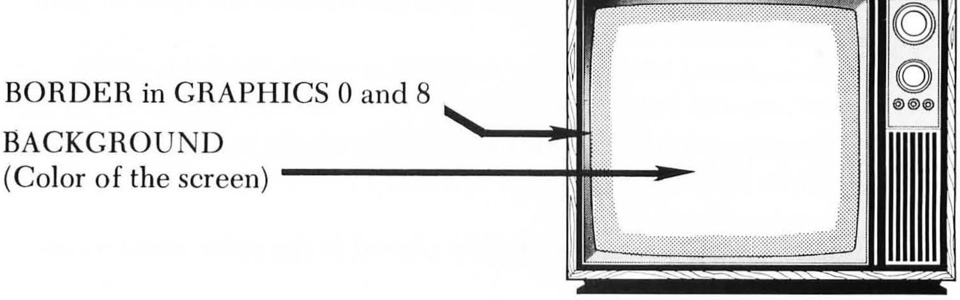
Default Colors

Everytime you turn the computer ON or change GRAPHICS modes, the *computer selects specific colors* to put into the color registers.

You have a choice of using these *"default" colors* or *selecting new ones* by *using* SETCOLOR.


Hey! Why is this the color that is automatically put into this color register?
Don't blame me... it's Default of de computer.

NOTE: While all graphics modes have a background color, GRAPHICS Modes 0 and 8 also have a border around the screen. The default color of the border is always black, but you will soon learn how to change it.



Changing Colors

You have the ability to change the background or the border of this screen to any of 16 basic colors. In addition, you can determine how bright or dark you want that color. Sixteen colors times eight luminances (brightnesses) equal a total of 128 colors to select from.

The 16 basic colors you can choose from are

COLORS

 0 = Gray
 1 = Light orange
 2 = Orange
 3 = Red orange
 4 = Pink
 5 = Purple
 6 = Purple blue
 7 = Blue
 8 = Blue
 9 = Light blue
10 = Turquoise
11 = Green blue
12 = Green
13 = Yellow green
14 = Orange green
15 = Light orange

You can use this program to select these colors on the screen.

```
10 REM COLOR CHOOSE
20 PRINT "↰": REM CLEARS SCREEN
30 POKE 752,1: REM TURNS OFF CURSOR
40 POSITION 3,6: PRINT "TYPE IN A NUMBE
R AND PRESS RETURN"
50 POSITION 6,9: PRINT "WHAT NUMBER DO
YOU WANT";: INPUT KOLOR
60 SETCOLOR 1,X,14: REM MAKES TEXT BRIG
HT
70 POSITION 9,11: PRINT "THE COLOR NUMB
ER IS ";KOLOR
80 SETCOLOR 2,KOLOR,4
90 FOR DELAY=1 TO 1000: NEXT DELAY
100 GOTO 20
```

NOTE: In order to avoid confusing the computer, it's good practice not to use variable names which contain BASIC key words such as COLOR or NEW. For example, line 40 contains the variable KOLOR.

If your colors do not match these numbers very well, it is possible that your TV or monitor is out of adjustment or has controls such as Automatic Fine Tuning (AFT) ON when they should be OFF. If you cannot get close, any authorized ATARI Service Center should be able to check your computer output quickly and easily.

In addition to being able to control the color of your screen and border, you can also control how bright or dark the characters on the screen are. The following information will give you practice in controlling *background color, border color* and *character brightness...*

The first number following the SETCOLOR selects color register (1,2 or 4 in GR. 0). For example, your background color is controlled by a SETCOLOR 2.

SE. 2,4,8

will turn your screen into a fairly gross pink color which is color 4 with a brightness of 8.

The next number controls the actual color.

SE. 2, *0* through *15*, 8

Any of the basic 16 colors listed on the previous page can go here.

For instance, SE. 2,6,8 will turn your screen purple-blue.

The third number controls the brightness of the color you selected.

SE.2,6, *0* through *14*

0 = DARKEST
14 = BRIGHTEST
(Only EVEN numbers have an effect so 3 would be the same as 2 or 11 the same as 10.)

Now, select another color register.
SETCOLOR 1 controls the brightness of the characters.

SE.1, *ANY NUMBER,* *0-14*

No effect, because the color of the characters is always the same as the GR.0 background.

0 = Darkest
14 = Brightest
Even numbers only.

SE. 2,6,8

SE. 1,0,8

IMPORTANT...If these last two numbers are the same, then your characters will disappear because the letter and the background become the same color and brightness.

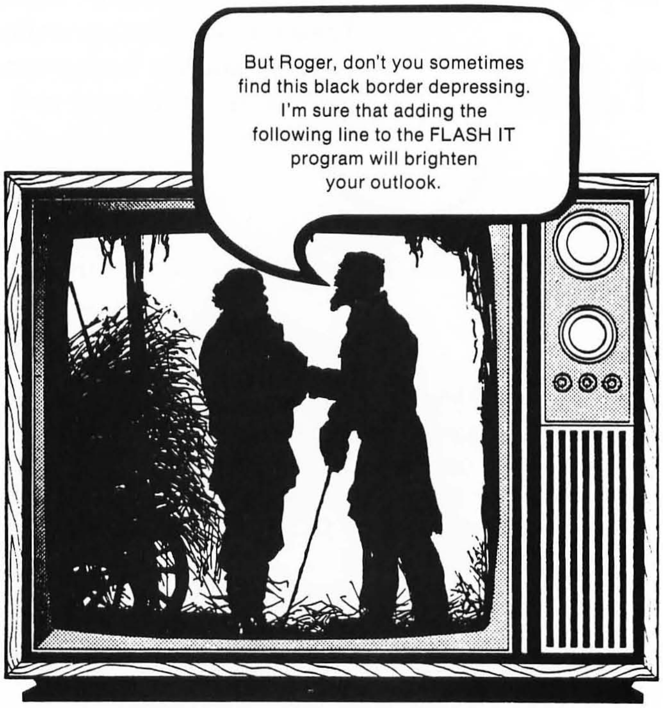
Here's a program that demonstrates how you can use this to make words appear and disappear.

```
10 PRINT "↰": REM CLEARS THE SCREEN
20 POKE 752,1: REM TURNS OFF CURSOR
30 POSITION 12,11
40 PRINT "NOW YOU SEE IT"
50 SETCOLOR 2,15,12
60 SETCOLOR 1,X,2
70 FOR DELAY=1 TO 600: NEXT DELAY
80 SETCOLOR 1,X,12: REM THIS MAKES THE
CHARACTERS THE SAME BRIGHTNESS AS THE
BACKGROUND
90 FOR DELAY=1 TO 600: NEXT DELAY
100 GOTO 50
```

Now erase that program and type in this one which slowly flips through the brightness of the letters and makes them appear to flash.

```
10 PRINT "↖": REM CLEARS THE SCREEN
20 POKE 752,1: REM TURNS OFF CURSOR
30 POSITION 10,11
40 PRINT "NOW YOU FLASH IT"
50 SETCOLOR 2,13,8: REM BACKGROUND
60 FOR REPEAT=1 TO 100
70 FOR FLASH=0 TO 14 STEP 2
80 SETCOLOR 1,X,FLASH
90 FOR DELAY=1 TO 15: NEXT DELAY
100 NEXT FLASH
110 NEXT REPEAT
```

If you recall, SETCOLOR 4, ___, ___ controls the color of the border.


But Roger, don't you sometimes find this black border depressing. I'm sure that adding the following line to the FLASH IT program will brighten your outlook.

```
85 SETCOLOR 4,6,6:REM BORDER COLOR
```

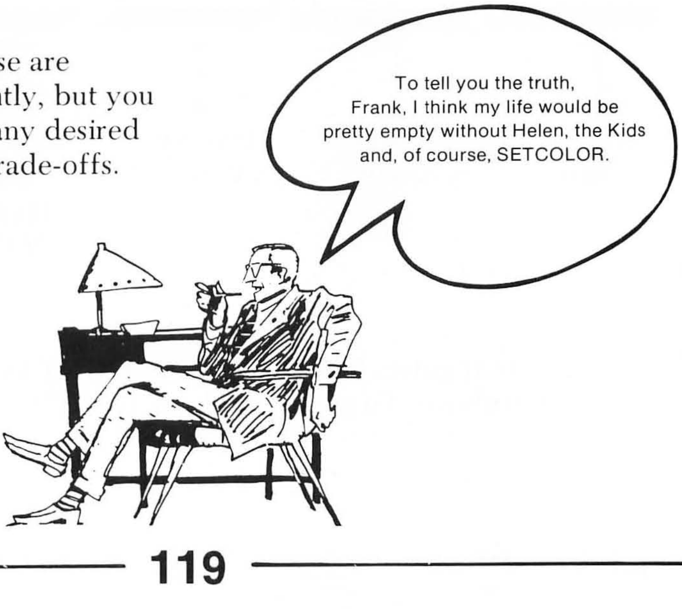
Using SETCOLOR in GRAPHICS 1 and 2

Now that you have mastered the color control of Graphics Mode 0, let us proceed to Graphics Modes 1 and 2. Otherwise you might not get confused!

Graphics 1 and Graphics 2 do not have borders, but they do have text windows. In both of these modes, you can control the following...

- The color of the background
- The color of LARGE WORDS PRINTed after a PRINT #6; in normal upper-case letters
- The color of LARGE WORDS PRINTed after a PRINT #6; in inverse video UPPER-CASE letters
- The color of LARGE WORDS PRINTed after a PRINT #6; in inverse video lower-case letters
- The color of LARGE WORDS PRINTed after a PRINT #6; in lower-case letters
- The color of the text window
- The luminances of the text in the text window

NOTE: Not all of these are controlled independently, but you should be able to get any desired effect with very few trade-offs.


To tell you the truth, Frank, I think my life would be pretty empty without Helen, the Kids and, of course, SETCOLOR.

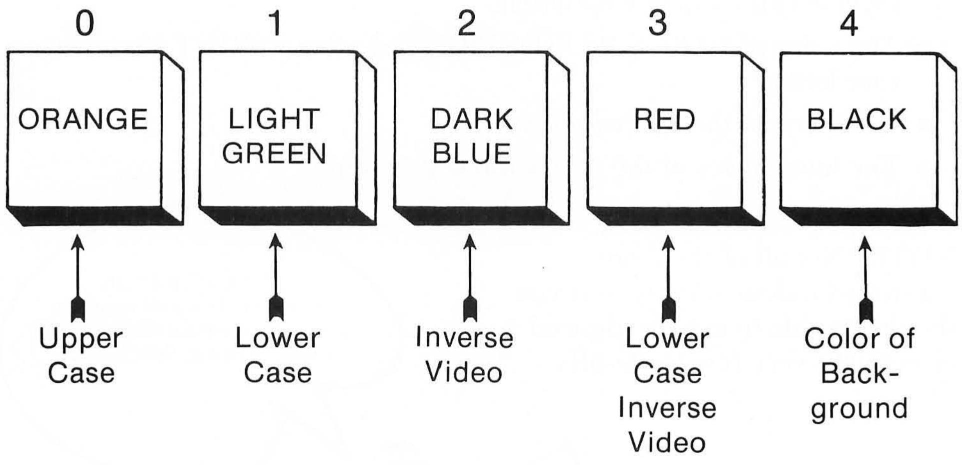
Default Colors in GR. 1 and 2

Since the color registers are the same for both GRAPHICS 1 and 2, these examples will be done in GRAPHICS 2. So the only difference is that the words will be bigger than if they were done in GRAPHICS 1.

Press SYSTEM RESET
Then type...

```
GR.2 RETURN
```

When you did that, the computer placed its GRAPHICS 1 and 2 default colors into the color registers.
It did so as follows...



To get to registers 0 through 3, type PRINT #6; and use the character styles shown above. To get to register 4, use SETCOLOR 4, ___ , ___

A Quick Refresher

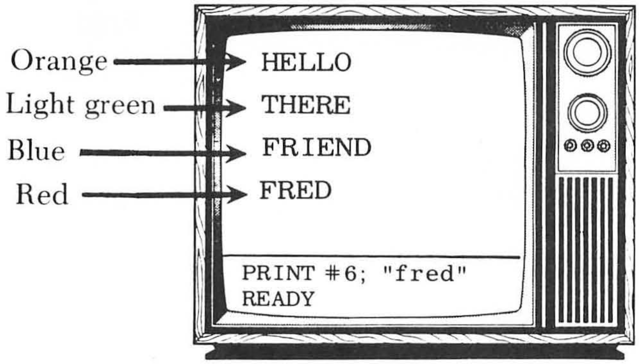
To get into INVERSE VIDEO	Press ⩓ once.
To get out of it...	Press it again.
To get into lower case...	Press CAPS LOWR once.
To get out of it...	Press SHIFT and CAPS LOWR at the same time.

REMEMBER, the computer will not accept instructions such as PRINT in lower case.

Now try this...

PRINT #6; "HELLO"	Print in UPPER CASE.
PRINT #6; "there"	Print in lower case.
PRINT #6: "FRIEND"	Print in INVERSE VIDEO.
PRINT #6: "fred"	Print in lower-case INVERSE VIDEO.

After a few tries your output should look like this...



```
PRINT #6; "fred"
READY
```

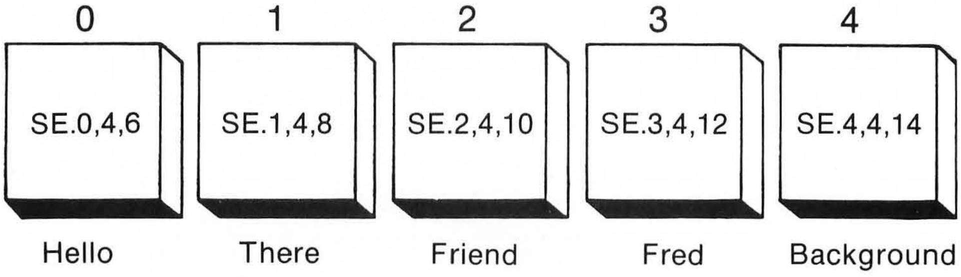
If you are hopelessly confused at this point, review the section on GRAPHICS 2 and come back to this section. Getting in and out of INVERSE VIDEO and lower case can be tricky for the beginner. Don't be afraid to press SYSTEM RESET and type GR. 2 for a fresh start.

Now let's change the colors in those registers using SETCOLOR. We could start by making everything a different shade of that gross pink that I seem to like so much.

SE.0,4,6 ———	makes HELLO almost RED.
SE.1,4,8 ———	makes THERE a true PINK.
SE.2,4,10 ———	makes FRIEND and the text window a color which should only be used for poodle collars.
SE.3,4,12 ———	makes FRED bright PINK.

Finally, let's lighten the background also...

SE.4,4,14

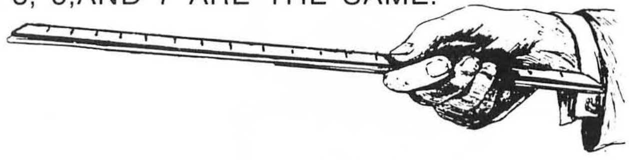


Color for GRAPHICS Modes 3 Through 8

Now that you have an idea of how SETCOLOR works with the Text Modes, you can learn how to use it with the COLOR statements that are used with GRAPHICS MODES 3 through 8.

Here's a general rule.

COLORS FOR GRAPHICS 3, 5,AND 7 ARE THE SAME.



When you enter GRAPHICS Mode 3, 5, or 7, the SETCOLOR registers are filled with the following default colors...

0	1	2	3	4
ORANGE	LIGHT GREEN	DARK BLUE	NOT USED	BLACK

Default color of back-ground.

You can now do one of two things:

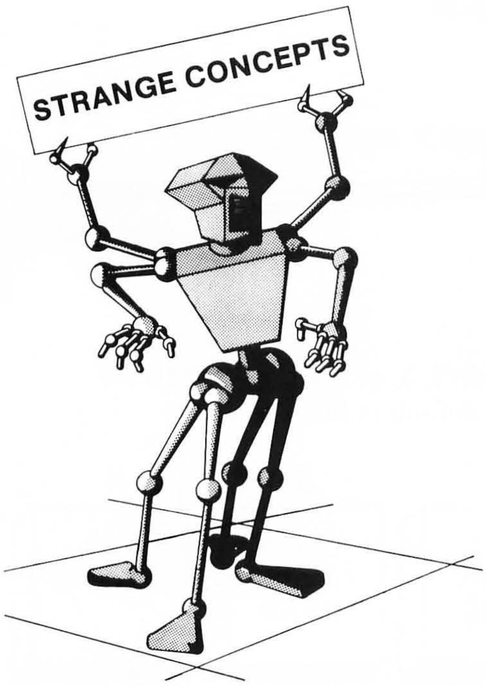
1. Change any of the colors in these registers using *SETCOLOR*.

-OR-

2. Use *COLOR* to pick from registers 0,1,2 or 4. This will determine the color of dots or lines you put on the screen.

Using COLOR

COLOR is used in the true GRAPHICS MODES (3 through 8), in order to select which color register you want. When you want to select or change the color of the next point or line to be drawn, you use COLOR and a number to select the appropriate SETCOLOR register.



You would think that if you wanted to use the color in register 0 you would type COLOR 0, or for register 1 you would use COLOR 1. Well, that would be too easy for you! In order to pack a lot of power into your BASIC cartridge, the designers decided that you would have to do a little translating here. As it turns out, things are a little bit more tricky than we might have wished.

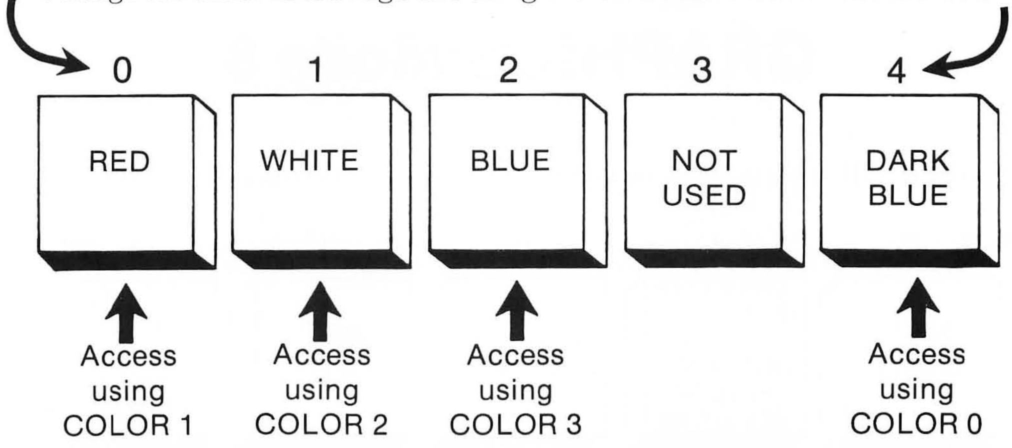
Here's an example that uses the default colors in registers 0,1,2 and 4. Type the following and don't forget to press the RETURN key after each line.

```
GRAPHICS 3
COLOR 1
PLOT 2,2:DRAWTO 38,2
COLOR 2
PLOT 2,4:DRAWTO 38,4
COLOR 3
PLOT 2,6:DRAWTO 38,6
```

Notice that COLOR 1,2 and 3 were used to select registers 0,1 and 2!
Also observe that using SETCOLOR to change the value in these registers will change the colors of the lines you have just drawn.
Type the following and notice the results...

SE.0,4,4 ————	changes the first line to RED.
SE.1,0,14 ————	changes the second line to white and makes the characters in the text window brighter because this register's luminance also controls the brightness of the characters in the text window.
SE.2,8,0 ————	changes the color of the third line and the color of the text window.
SE.3,8,12 ————	HAS NO EFFECT! This register is not used in this GRAPHICS MODE.
SE.4,7,4 ————	changes the background to a different shade of blue

Change the color in the registers using these numbers with SETCOLOR.

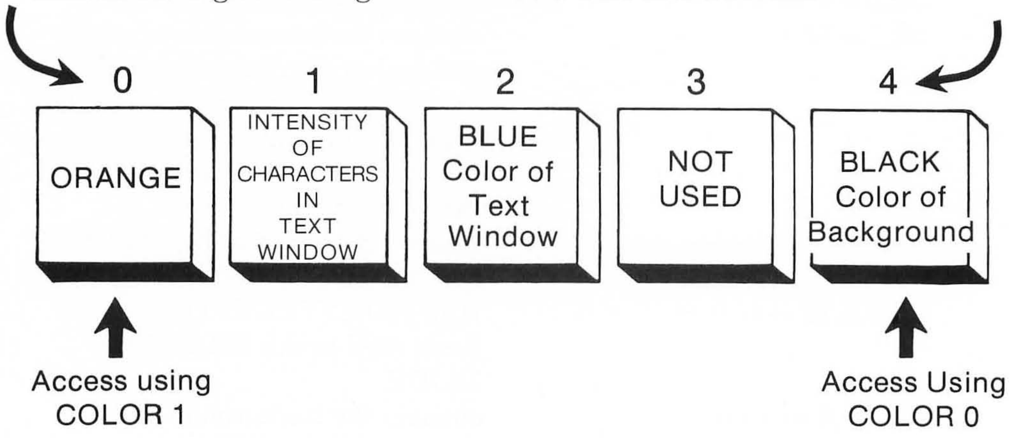


Nothing to it, right?

Wrong! You see, the way this works in the other GRAPHICS Modes is slightly different.

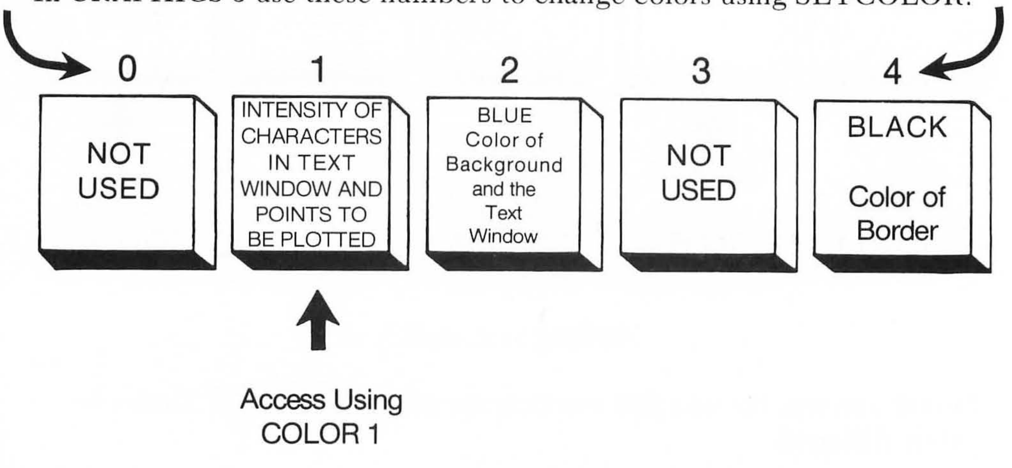
GRAPHICS Modes 4 and 6

For GRAPHICS 4 and 6, only two color statements are used. Change the color in the registers using these numbers with SETCOLOR.


0
1
2
3
4
ORANGE
INTENSITY OF CHARACTERS IN TEXT WINDOW
BLUE Color of Text Window
NOT USED
BLACK Color of Background
Access using COLOR 1
Access Using COLOR 0

GRAPHICS Mode 8

In GRAPHICS 8 use these numbers to change colors using SETCOLOR.


0
1
2
3
4
NOT USED
INTENSITY OF CHARACTERS IN TEXT WINDOW AND POINTS TO BE PLOTTED
BLUE Color of Background and the Text Window
NOT USED
BLACK Color of Border
Access Using COLOR 1

Summary of SETCOLOR and COLOR

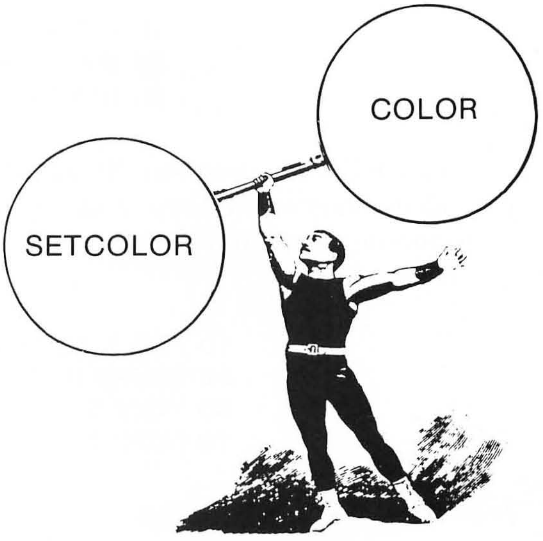
SETCOLOR registers can be used to control a variety of things:

- The color of the border or the background.
- The color of the points you plot and the lines you draw.
- The brightness of the characters on the screen.
- The color of objects used in advanced graphics techniques.

COLOR 0,1,2,3, or 4 are used (starting with GRAPHICS Mode 3) to tell the computer that you want to plot or draw a line with a color stored in a particular color register. You have to be sure that the COLOR number you choose is appropriate for the SETCOLOR register used in the graphics mode you are in.

If you're confused, don't worry; all of these are summarized in the section titled "The Relationship Between Using SETCOLOR and POKEing COLORS." After you read about POKEing COLORS, you will find an easy-to-use summary chart.

With a little review and practice, you will find the use of SETCOLOR and COLOR effortless.



Sound Advice

"I've never met a pitch and distortion value I didn't like."
(W. Rogers, *paraphrased*)

"Always give a distortion register an even value."
(W.C. Fields, *paraphrased*)

Try this program...

```
10 FOR Z=0 TO 255
20 SOUND 0,Z,10,10
30 NEXT Z
```

Now RUN the program. Be sure the TV volume is turned up. This is your elementary sound loop, which would be a monumental programming accomplishment on some other computers.

Now add...

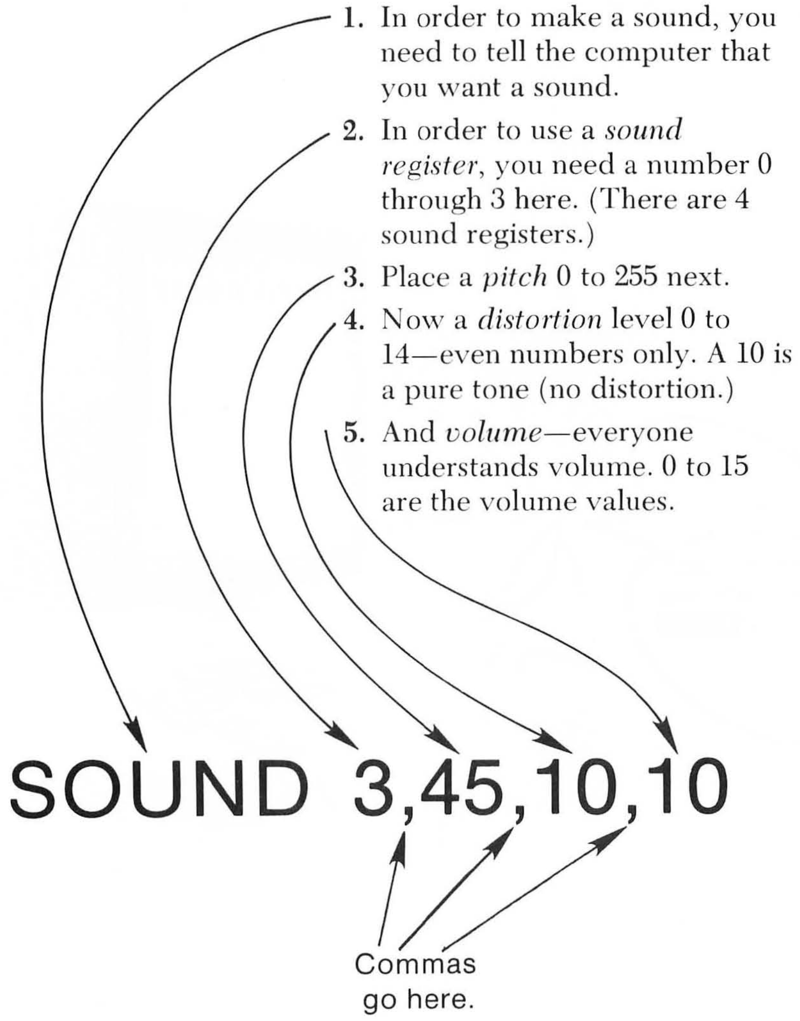
```
40 FOR Z=255 TO 0 STEP -1
50 SOUND 0,Z,10,10
60 NEXT Z
70 GOTO 10
```

Now RUN this new program.

The Essentials of ATARI SOUND

There are many complex and tricky ways to create sounds with ATARI Home Computers, but the essential elements of creating sound in BASIC can be summarized as follows:

1. In order to make a sound, you need to tell the computer that you want a sound.
2. In order to use a *sound register*, you need a number 0 through 3 here. (There are 4 sound registers.)
3. Place a *pitch* 0 to 255 next.
4. Now a *distortion* level 0 to 14—even numbers only. A 10 is a pure tone (no distortion.)
5. And *volume*—everyone understands volume. 0 to 15 are the volume values.



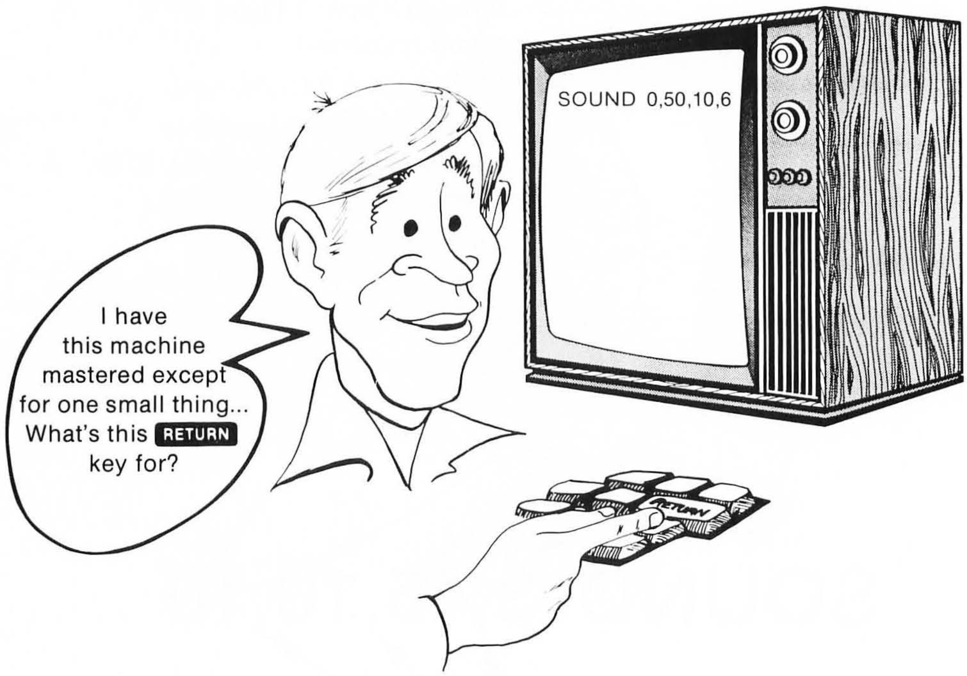
Notice that you can do a totally different sound for each of the four sound registers...

PRESS **SYSTEM RESET**

Now try some harmony in the direct mode. Type...

```
SOUND 0,50,10,6
SOUND 1,100,10,6
SOUND 2,150,10,6
SOUND 3,200,10,6
```

Remember to press the **RETURN** key and turn up the TV volume.


SOUND 0,50,10,6
I have this machine mastered except for one small thing... What's this RETURN key for?
RETURN

To turn off a sound, use a SOUND 0,0,0,0

Be sure to change this number to turn off the correct sound register.

```
10 REM SIMPLE KEYBOARD ORGAN
20 PRINT "TURN UP THE TV VOLUME AND HI
T ANY KEY EXCEPT BREAK"
30 OPEN #1, 4, 0, "K: "
40 GET #1, X
50 SOUND 0, X, 10, 8: POKE 712, X
60 FOR D=1 TO 75: NEXT D: SOUND 0, 0, 0, 0
70 GOTO 40
```

These instructions look at the ASCII code for the key you press. This is a bit advanced and not really relevant to sound. More information on using ASCII coding can be found in the section on FUNCTIONS.

Try this program the way it's listed and then take out the SOUND 0,0,0,0 to see the difference.

Here's a program to play along with *Stormy Weather*...

```
10 GRAPHICS 7+16: SETCOLOR 4, 0, 0: SETCOL
OR 1, 0, 4
20 COLOR 2
30 PLOT 2, 2: DRAWTO 30, 34: DRAWTO 78, 40:
DRAWTO 100, 57: DRAWTO 110, 50: DRAWTO 140
, 76
40 FOR LIGSOU=1 TO 255
50 SOUND 0, LIGSOU, 8, 10
60 IF LIGSOU=8 THEN SETCOLOR 1, 0, 14
70 NEXT LIGSOU
80 SETCOLOR 1, 0, 0
90 FOR DE=1 TO 200: NEXT DE
100 GOTO 10
```

You'll find more examples using SOUND in the section on Paddle Controllers.

POKEing Around

When you turn your ATARI Computer on, specific memory locations are automatically filled with values that influence the way the computer behaves.

For instance, in the standard GRAPHICS mode 0, the number which is placed into *location 710* controls the color of your screen. When you turn your computer ON, a 148 is placed into that location. 148, as it turns out, will make your screen the familiar BLUE. You already know how to change this using SETCOLOR, but here's a new way.

If you type...

```
POKE 710,79
```

...you will notice that the screen is changed to a somewhat unattractive color.

Now type...

```
POKE 712,79
```

...and you will observe that the border has changed to the same color.

From this you can conclude that in GRAPHICS Mode 0, the color of the screen is controlled by location 710, and the color of the border is controlled by location 712. The value 79 is simply the number associated with a particular color.

Try typing in this program...

```
10 FOR X=0 TO 254 STEP 2
20 POKE 710,X
30 NEXT X
```

Now run it...

Then try...

```
10 PRINT "↰"
20 FOR X=0 TO 254 STEP 2
30 PRINT X
40 POKE 710,X
50 FOR D=1 TO 50: NEXT D
60 NEXT X
```

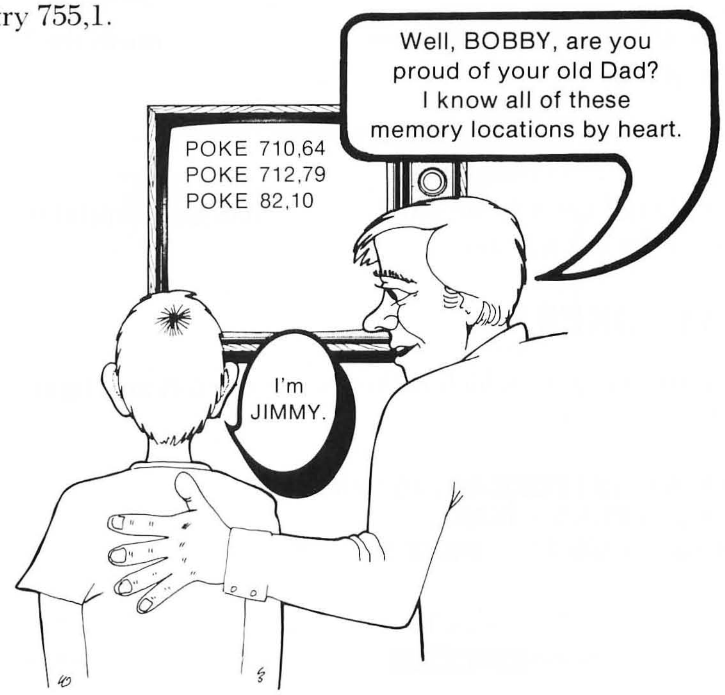
If you want to stop and look at a particular number and color, press CTRL and 1 at the same time. Press them again to continue.

Now add one more line to the program and make the cursor disappear.

```
5 POKE 752,1
```

To turn the cursor back on, POKE a 0 into location 752.

Sometimes a 755,1 is needed to turn the cursor off. If 752,1 doesn't work, then try 755,1.


Well, BOBBY, are you proud of your old Dad? I know all of these memory locations by heart.
POKE 710,64
POKE 712,79
POKE 82,10
I'm JIMMY.

PEEKing not POKEing

It is also possible to look into a location just to see what value is there without changing it. You do this by "PEEKing" that location.

For instance...

Press SYSTEM RESET

Now type...

```
PRINT PEEK(710)
```

Notice parentheses are used around the location when you are PEEKing.

You should get a 148...the number associated with the standard BLUE screen.

Now try PEEKing into location 82, which controls the left margin of your computer.

```
PRINT PEEK(82)
```

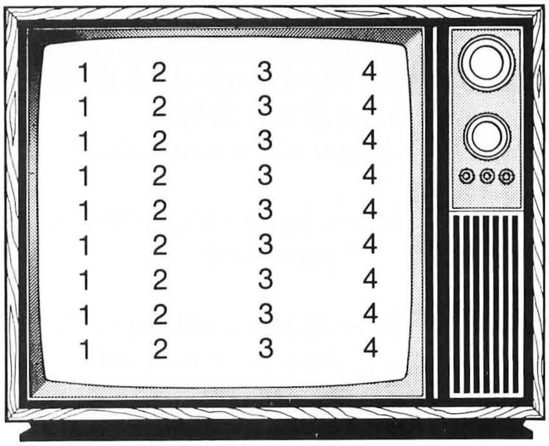
You should get a 2, the number of spaces the computer usually indents for the left margin.

Back to POKES

Try this program which makes both your left and right margins the same value.

```
10 POKE 82,19: POKE 83,19: PRINT
20 PRINT "HELLO THERE"
30 FOR DE=1 TO 350: NEXT DE: GOTO 20
```

...And you see that you now have new margins until you POKE them back to normal, press SYSTEM RESET or turn your computer OFF.



```
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
1    2    3    4
```

POKEing a new tab width...

When you print words or numbers with commas between them, the computer normally allows for 10 spaces for each column. To change this, POKE a new tab width value into location 201.

```
10 POKE 201,2: POKE 82,0
20 PRINT : PRINT A,B,C,D,E,F,G
30 A=A+1: B=B+1: C=C+1: D=D+1: E=E+1: F=F+1
: G=G+1
40 GOTO 20
```

Here's a program that combines POKEing color registers with POKEing new margins:

```
10 POKE 710,0: POKE 752,1
20 FOR Z=1 TO 38: POKE 82,Z
30 PRINT"\"
40 POKE 712,Z*5
50 SOUND 0,Z*8,10,10
60 NEXT Z
70 FOR X=37 TO 1 STEP -1
80 POKE 82,X
90 POKE 712,X*5
100 PRINT "/"
110 SOUND 0,X*8,10,10
120 NEXT X
130 RUN
```

Location 755 has some interesting effects on normal and INVERSE VIDEO text. The normal value for this location is 2, so to get everything back to normal POKE 755,2

Other values in location 755 have the following effects...

POKE 755,0—Changes INVERSE VIDEO to normal text and turns the cursor off.

POKE 755,1—Makes anything in INVERSE VIDEO disappear and turn the cursor off.

POKE 755,2—Returns all to normal.

POKE 755,3—Makes full blocks out of anything in INVERSE VIDEO.

POKE 755,4—Prints characters upside down with the cursor OFF.

POKE 755,6—Prints characters upside down with the cursor still ON.

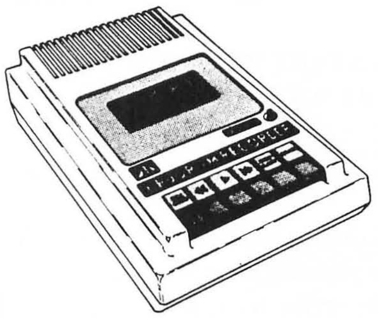
The following program shows the same INVERSE VIDEO word being changed by POKEing different values into location 755.

```
10 PRINT "↖": REM CLEARS THE SCREEN
20 POKE 752,1: REM TURNS CURSOR OFF
30 POSITION 16,12: PRINT "SURPRISE": REM
 PRINT "SURPRISE" IN INVERSE VIDEO
40 FOR Z=0 TO 4
50 POKE 755,Z
60 FOR DE=1 TO 300: NEXT DE
70 NEXT Z
80 GOTO 40
```

Here are a few more interesting POKE locations...

POKE 54018,52—Will start your ATARI 410 Program Recorder. (The PLAY button on the program recorder must be pressed down.)

POKE 54018,60—Will turn the Program Recorder off.



You can play an ordinary music cassette through your TV speaker with a POKE 54018,52. Be sure to rewind the tape and press down PLAY on the 410 Program Recorder before you begin.

Here's a short program which shows you how you can use the computer to start and stop the music.

```
10 PRINT "↖": POKE 752, 1: DIM AN$ (1) , ST$
(1)
20 PRINT "TO HEAR MUSIC,  TYPE Y AND PR
ESS RETURN"
30 INPUT AN$
40 IF AN$= "Y" THEN POKE 54018, 52: GOTO
60
50 GOTO 20
60 PRINT "↖": PRINT "TO STOP MUSIC, TYP
E N AND PRESS RETURN"
70 INPUT ST$
80 IF ST$= "N" THEN POKE 54018, 60: GOTO
20
90 GOTO 60
```

You can have your computer PEEK at location 53279 to see if anyone out there is pressing the START, SELECT or OPTION keys.

A 6 means START is pressed.
A 5 means SELECT is pressed.
A 3 means OPTION is pressed.

Here's an opener for your game programs:

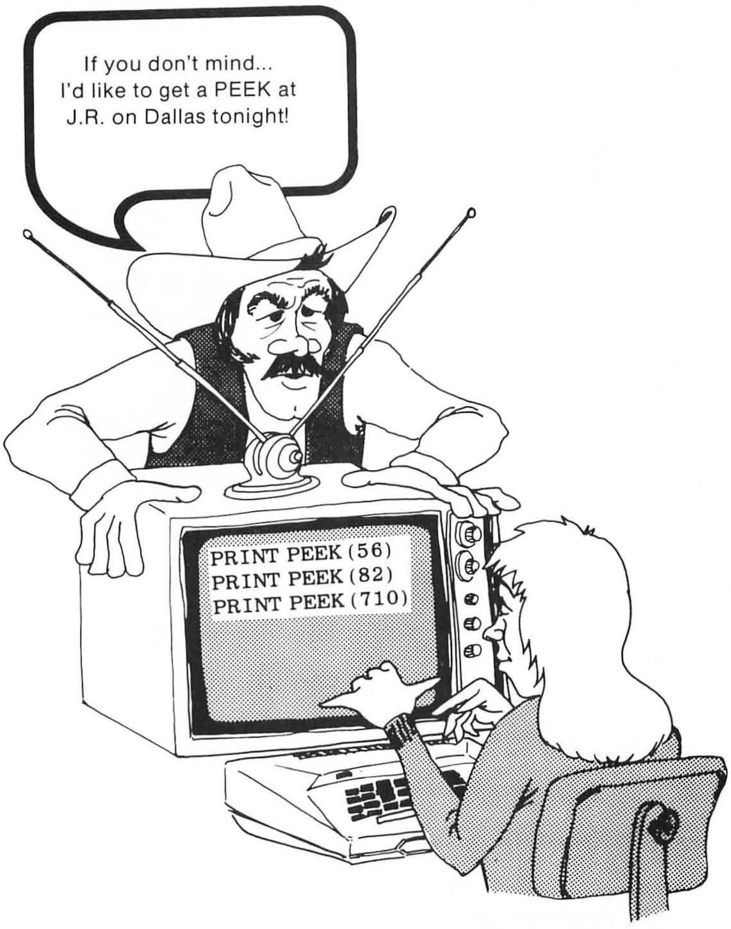
```
10 PRINT "PRESS START TO BEGIN"
20 IF PEEK(53279) =6 THEN GOTO 40
30 GOTO 20
40 PRINT "READY TO BEGIN"
```

There are also locations in the computer which you can use for timers.

The core of this second counter program is line 50. You might experiment with it in order to make computer clocks, internal timers or stop watches. Keep in mind, however, it takes time for your computer to draw fancy displays; so you'll need to adjust your timing to match your displays.

```
10 PRINT "↰": POKE 752,1
20 POSITION 5,11: PRINT "PRESS START FO
R SECOND COUNTER ": IF PEEK(53279) =6 THE
N PRINT "↰": GOTO 40
30 GOTO 20
40 POKE 18,0: POKE 19,0: POKE 20,0
50 SECONDS= INT( (PEEK(18) *65535+PEEK(19
) *256+PEEK(20)) /60)
60 POSITION 19,11: PRINT SECONDS
70 IF SECONDS=60 THEN PRINT "↰": GOTO 4
0
80 GOTO 50
```

A more detailed list of locations you can POKE around in are listed in the section called "memory locations" in the ATARI BASIC Reference Manual.



NOTE: It is very easy for you to "CRASH" the program (temporarily disable it) by POKEing numbers into unknown memory locations. If this happens, you will have to turn the computer OFF, then ON again. Some people recommend that you don't POKE around in unknown memory locations because you might accidentally erase a diskette or something like that. PEEKing memory locations never hurts anything.

The Relationship Between Using SETCOLOR and POKEing Colors

As you have learned, ATARI Computers have 128 colors and they also have 16 colors. Both statements are right, in a way. You do have 128 color variations to choose from.

Start with 16 basic colors...

0=Gray
1=Gold
2=Orange
3=Red-orange
4=Pink
5=Pink-purple
6=Purple-blue
7=Blue
8=Another Blue
9=Light blue
10=Turquoise
11=Green-blue
12=Green
13=Yellow-green
14=Orange-green
15=Light orange

...and each of these colors has 8 levels of brightness or luminance (dark to light)

16 colors × 8 luminances = 128 colors to choose from!

To see all 128 colors, use this familiar program:

```
10 PRINT "↰": POKE 752,1
20 FOR Z=0 TO 254 STEP 2
30 POSITION 19,12: PRINT Z
40 POKE 710,Z
50 FOR DELAY=1 TO 25: NEXT DELAY
60 NEXT Z
```

NOTE: Only even numbers affect the screen color; that's why you use 0 to 254 instead of 0 to 128.

The SETCOLOR ____, ____, ____ format was designed to help you understand and use the color/luminance relationship better. Sometimes SETCOLOR is easy, but sometimes using POKES seems more convenient. Understanding the relationship between both will make your life much simpler.

Here's a quick review of the way SETCOLOR works.

When you use SETCOLOR ...

SETCOLOR ____, ____, ____

This number is used for the color register.

And this is the brightness number. The even values 0 through 14, give you your dark to light value for each of the 16 colors.

This is the color number.

0 = Gray

through

15 = Light Orange

POKEing colors simply inserts equivalent color values from 0 to 254 into the equivalent color registers.

Color Registers	704	Color Values	0
			1
			2
			3
Through		Through	
			252
			253
	712		254

In GRAPHICS MODE 0 for instance...

```
SETCOLOR 2, ____, ____
```

or

```
POKE 710, ____
```

will control the background color of the screen.

Try the following examples and look on your SETCOLOR and POKE Equivalents Chart for a full understanding of the relationship between the two.

```
SE.2,3,6
WILL PRODUCE THE SAME COLOR AS...
POKE 710,54
or
SE.2,15,4
IS EQUIVALENT TO...
POKE 710,244
```

```
SE.4,4,6
WILL PRODUCE THE SAME COLOR AS...
POKE 712,70
or
SE.4,7,12
IS EQUIVALENT TO...
POKE 712,124
```

SETCOLOR and POKE Equivalent Chart

SE. REGISTER #, ____, ____

			EQUIVALENT POKE VALUE
COLOR= 0			
		LUMINANCE= 0	0
		LUMINANCE= 2	2
		LUMINANCE= 4	4
	Gray	LUMINANCE= 6	6
		LUMINANCE= 8	8
		LUMINANCE= 10	10
		LUMINANCE= 12	12
		LUMINANCE= 14	14
COLOR= 1			
		LUMINANCE= 0	16
		LUMINANCE= 2	18
		LUMINANCE= 4	20
		LUMINANCE= 6	22
	Gold	LUMINANCE= 8	24
		LUMINANCE= 10	26
		LUMINANCE= 12	28
		LUMINANCE= 14	30
COLOR= 2			
		LUMINANCE= 0	32
		LUMINANCE= 2	34
		LUMINANCE= 4	36
	Orange	LUMINANCE= 6	38
		LUMINANCE= 8	40
		LUMINANCE= 10	42
		LUMINANCE= 12	44
		LUMINANCE= 14	46
COLOR= 3			
		LUMINANCE= 0	48
		LUMINANCE= 2	50
		LUMINANCE= 4	52
	Red-	LUMINANCE= 6	54
	orange	LUMINANCE= 8	56
		LUMINANCE= 10	58
		LUMINANCE= 12	60
		LUMINANCE= 14	62

			EQUIVALENT POKE VALUE
COLOR= 4			
		LUMINANCE= 0	64
		LUMINANCE= 2	66
		LUMINANCE= 4	68
	Pink	LUMINANCE= 6	70
		LUMINANCE= 8	72
		LUMINANCE= 10	74
		LUMINANCE= 12	76
		LUMINANCE= 14	78
COLOR= 5			
		LUMINANCE= 0	80
		LUMINANCE= 2	82
		LUMINANCE= 4	84
	Pink-	LUMINANCE= 6	86
	purple	LUMINANCE= 8	88
		LUMINANCE= 10	90
		LUMINANCE= 12	92
		LUMINANCE= 14	94
COLOR= 6			
		LUMINANCE= 0	96
		LUMINANCE= 2	98
		LUMINANCE= 4	100
	Purple-	LUMINANCE= 6	102
	blue	LUMINANCE= 8	104
		LUMINANCE= 10	106
		LUMINANCE= 12	108
		LUMINANCE= 14	110
COLOR= 7			
		LUMINANCE= 0	112
		LUMINANCE= 2	114
		LUMINANCE= 4	116
	Blue	LUMINANCE= 6	118
		LUMINANCE= 8	120
		LUMINANCE= 10	122
		LUMINANCE= 12	124
		LUMINANCE= 14	126

		EQUIVALENT POKE VALUE
COLOR= 8		
	LUMINANCE= 0	128
	LUMINANCE= 2	130
	LUMINANCE= 4	132
Blue	LUMINANCE= 6	134
	LUMINANCE= 8	136
	LUMINANCE= 10	138
	LUMINANCE= 12	140
	LUMINANCE= 14	142
COLOR= 9		
	LUMINANCE= 0	144
	LUMINANCE= 2	146
	LUMINANCE= 4	148
Light	LUMINANCE= 6	150
blue	LUMINANCE= 8	152
	LUMINANCE= 10	154
	LUMINANCE= 12	156
	LUMINANCE= 14	158
COLOR= 10		
	LUMINANCE= 0	160
	LUMINANCE= 2	162
	LUMINANCE= 4	164
Turquoise	LUMINANCE= 6	166
	LUMINANCE= 8	168
	LUMINANCE= 10	170
	LUMINANCE= 12	172
	LUMINANCE= 14	174
COLOR= 11		
	LUMINANCE= 0	176
	LUMINANCE= 2	178
	LUMINANCE= 4	180
Green-	LUMINANCE= 6	182
blue	LUMINANCE= 8	184
	LUMINANCE= 10	186
	LUMINANCE= 12	188
	LUMINANCE= 14	190

		EQUIVALENT POKE VALUE
COLOR= 12		
	LUMINANCE= 0	192
	LUMINANCE= 2	194
	LUMINANCE= 4	196
Green	LUMINANCE= 6	198
	LUMINANCE= 8	200
	LUMINANCE= 10	202
	LUMINANCE= 12	204
	LUMINANCE= 14	206
COLOR= 13		
	LUMINANCE= 0	208
	LUMINANCE= 2	210
	LUMINANCE= 4	212
Yellow-	LUMINANCE= 6	214
green	LUMINANCE= 8	216
	LUMINANCE= 10	218
	LUMINANCE= 12	220
	LUMINANCE= 14	222
COLOR= 14		
	LUMINANCE= 0	224
	LUMINANCE= 2	226
	LUMINANCE= 4	228
Orange-	LUMINANCE= 6	230
green	LUMINANCE= 8	232
	LUMINANCE= 10	234
	LUMINANCE= 12	236
	LUMINANCE= 14	238
COLOR= 15		
	LUMINANCE= 0	240
	LUMINANCE= 2	242
	LUMINANCE= 4	244
Light	LUMINANCE= 6	246
orange	LUMINANCE= 8	248
	LUMINANCE= 10	250
	LUMINANCE= 12	252
	LUMINANCE= 14	254

Sometimes POKEing 0 to 254 into color registers 704 and 712 can be easy and interesting to experiment with. You will find most of your results using locations 708 to 712. Later you will learn more about locations 704 through 707.

Example:

```
10 FOR Z=0 TO 254 STEP 2
20 POKE 710,Z
30 FOR DELAY=1 TO 35: NEXT DELAY
40 NEXT Z
```

Will produce the same result as this SETCOLOR version...

```
10 FOR K=0 TO 15: REM LOOP FOR 16 COLOR
S
20 FOR L=0 TO 14 STEP 2: REM LOOPS THRO
UGH THE 8 LUMINANCES OF EACH
30 SETCOLOR 2,K,L
40 FOR DELAY=1 TO 35: NEXT DELAY
50 NEXT L
60 NEXT K
```

However, let's say you wanted to flip from a dark red screen through to a light red screen, over and over. For that, all you would have to remember (or find out) is that red is number 4 which is placed second in the SETCOLOR statement, and you could just flip through the luminance values (0 to 14).

This is much easier than remembering the POKE value equivalents

```
10 FOR N=0 TO 14 STEP 2
20 SETCOLOR 2,4,N
30 FOR DELAY=1 TO 35: NEXT DELAY
40 NEXT N
50 GOTO 10
```

```
10 FOR N=64 TO 78 STEP 2
20 POKE 710,N
30 FOR DELAY=1 TO 35: NEXT DELAY
40 NEXT N
50 GOTO 10
```

In conclusion, sometimes SETCOLOR is the approach to use and sometimes you have to POKE your way through color graphics. As you learn about new graphics modes, you will find that both SETCOLOR and color POKES are necessary. Experimenting with both SETCOLOR and POKES will get you further down the color brick road.

SETCOLOR Register and POKE Location Equivalents

The following registers are essentially equivalent. It is up to you to determine whether to use a SETCOLOR (*Register, Color, Luminance*) or a POKE *(0 to 254)*.

Graphics Mode 0

Use This or This

SETCOLOR 2, ___, ___	Screen Color	POKE 710, ___
SETCOLOR 4, ___, ___	Border Color	POKE 712, ___
SETCOLOR 1, ___, ___	Intensity of Characters	POKE 709, ___

GRAPHICS Modes 1 and 2

SETCOLOR 2, ___, ___	Text Window	POKE 710, ___
SETCOLOR 1, ___, ___	Intensity of Text in Text Window	POKE 709, ___
SETCOLOR 4, ___, ___	Background	POKE 712, ___
SETCOLOR 0, ___, ___	Color of Large (PRINT #6;) Text Entered in Normal Upper-Case Letters	POKE 708, ___
SETCOLOR 2, ___, ___	Color of Large (PRINT #6;) Text Entered in INVERSE VIDEO. *Note*: This will be the same color as the Text Window.	POKE 710, ___
SETCOLOR 3, ___, ___	The Color of Large (PRINT #6;) Text Entered in Lower Case INVERSE VIDEO	POKE 711, ___
SETCOLOR 1, ___, ___	The Color of Large (PRINT #6;) Text Entered in Lower Case. *Note*: This will also control the intensity of normal-size characters in the Text Window and could make them disappear if their intensity matches the background intensity.	POKE 709, ___

GRAPHICS Modes 3, 5 and 7

SETCOLOR 4, ___, ___	Background	POKE 712, ___
SETCOLOR 2, ___, ___	Text Window	POKE 710, ___
SETCOLOR 1, ___, ___	Intensity of Characters in Text Window	POKE 709, ___
SETCOLOR 0, ___, ___	COLOR 1	POKE 708
SETCOLOR 1, ___, ___	COLOR 2 (Also controls intensity of Characters in Text Window).	POKE 709, ___
SETCOLOR 2, ___, ___	COLOR 3 (Also Color of Text Window)	POKE 710, ___

GRAPHICS Modes 4 and 6

SETCOLOR 4, ___, ___	Background Color	POKE 712, ___
SETCOLOR 0, ___, ___	COLOR 1	POKE 708, ___
SETCOLOR 1, ___, ___	Intensity of Characters in Text Window	POKE 709, ___
SETCOLOR 2, ___, ___	Text Window	POKE 710, ___

GRAPHICS Mode 8

NOTE: The color of the text window is the same color as the background.

SETCOLOR 2, __, __	Background	POKE 710, __
SETCOLOR 1, __, __	COLOR 1 Luminance Control Only. Color is the Same Color as Background.	POKE 709, __
SETCOLOR 4, __, __	Border	POKE 712, __

GTIA

GTIA GRAPHICS Modes 9, 10, and 11

(Covered in the section on GTIA)

All The Way With GTIA

In a continuing effort to improve product performance, ATARI's Home Computer Division has put a new part, called the GTIA chip, into its Home Computers.

This chip will allow you to write advanced graphics programs in BASIC because you have access to more shades and colors in additional graphics modes. You actually have 256 colors to choose from in one of the GTIA modes!

GTIA GRAPHICS 9

GRAPHICS 9 allows you to simultaneously use 16 luminances of a single color so you can create images with subtle degrees of shading on the screen.

GTIA GRAPHICS 10

GRAPHICS 10 allows you to put eight different colors of any luminance on the T.V. screen at one time.

```
10 GRAPHICS 10
20 FOR KOLASIGN=705 TO 712: POKE KOLASI
GN, INT(RND(0)*255): NEXT KOLASIGN
30 COLOR INT(RND(0)*7)+1: PLOT INT (RND(
0)*79), INT(RND(0)*191): DRAWTO 39,95
40 GOTO 20
```

NOTE: The use of random numbers (RND) is explained in detail in the section on FUNCTIONS.

GTIA GRAPHICS 11

GRAPHICS 11 gives you the ability to put 16 colors on the screen at one time! You can also easily vary the luminance of the 16 colors.

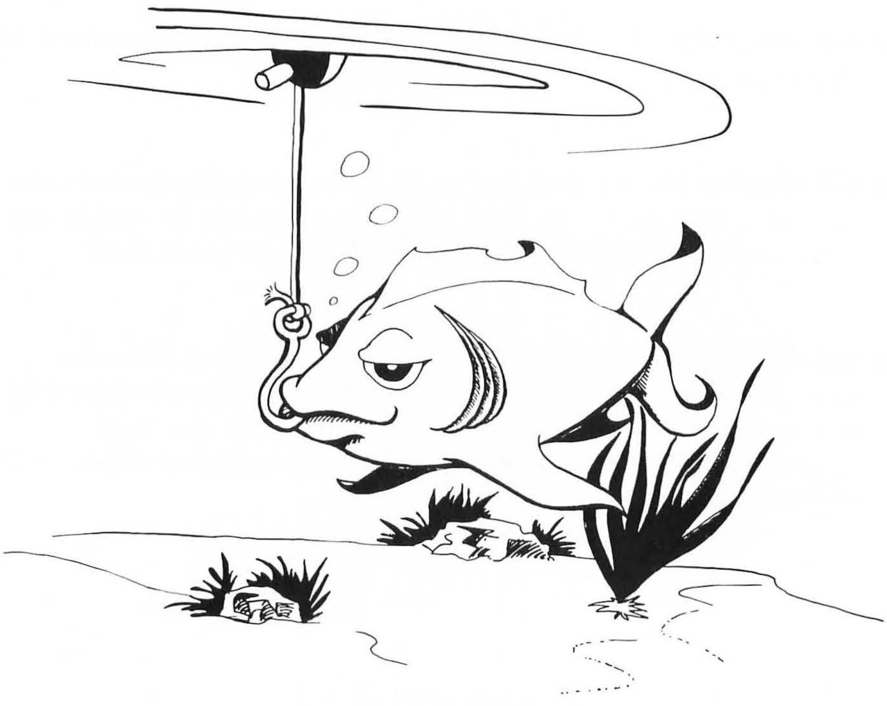
To summarize the three additional modes of the GTIA chip...

- **GRAPHICS 9**—You choose 1 of 16 colors and then you can use 16 shades of that color.

- **GRAPHICS 10**—8 colors, but no limitations regarding what color is used or the luminance. Dark and light colors can be used in any combination. The background color can also be controlled.

- **GRAPHICS 11**—16 different colors on the screen at the same time. You are given a full spectrum and you choose all or some of the 16 colors. You can vary how light or dark this spectrum appears. But, the same lightness or darkness applies to all colors at the same time.

The GTIA chip opens up a world of color and computer design to a beginning BASIC programmer. Some of these capabilities were previously known only to heavy-duty programmers and the people with access to expensive graphics computers.



There is one catch of course!

You may not have the GTIA chip in your computer. It is possible that you, instead, have the original CTIA chip.

The GTIA Test

An easy way to find out which chip you have in your computer is to RUN this simple program...

```
10 GRAPHICS 11
20 FOR Z=0 TO 15
30 COLOR Z
40 PLOT 0,Z:DRAWTO 79,Z
50 NEXT Z
60 GOTO 60
```

If you get any sort of ERROR message when you run the above program, check to see if you copied the program accurately. If you have a nice rainbow-like color bar display across the top of the screen, then you have a GTIA chip in your computer. If you don't, then you either have a Black-and-White television or a CTIA chip.

If you do not have a GTIA chip, but would really like to explore its graphics capabilities, then...

1. Bring your computer to an ATARI Factory Authorized Service Center for a GTIA upgrade.

—OR—

2. Sell your computer to a friend who is color blind and buy a new one.

In order to see the difference between Graphics 11 (16 colors) and Graphics 9 (16 shades of 1 color), simply change the first line of the previous program to read GR. 9 instead of GR. 11 and notice the difference when you RUN the program.

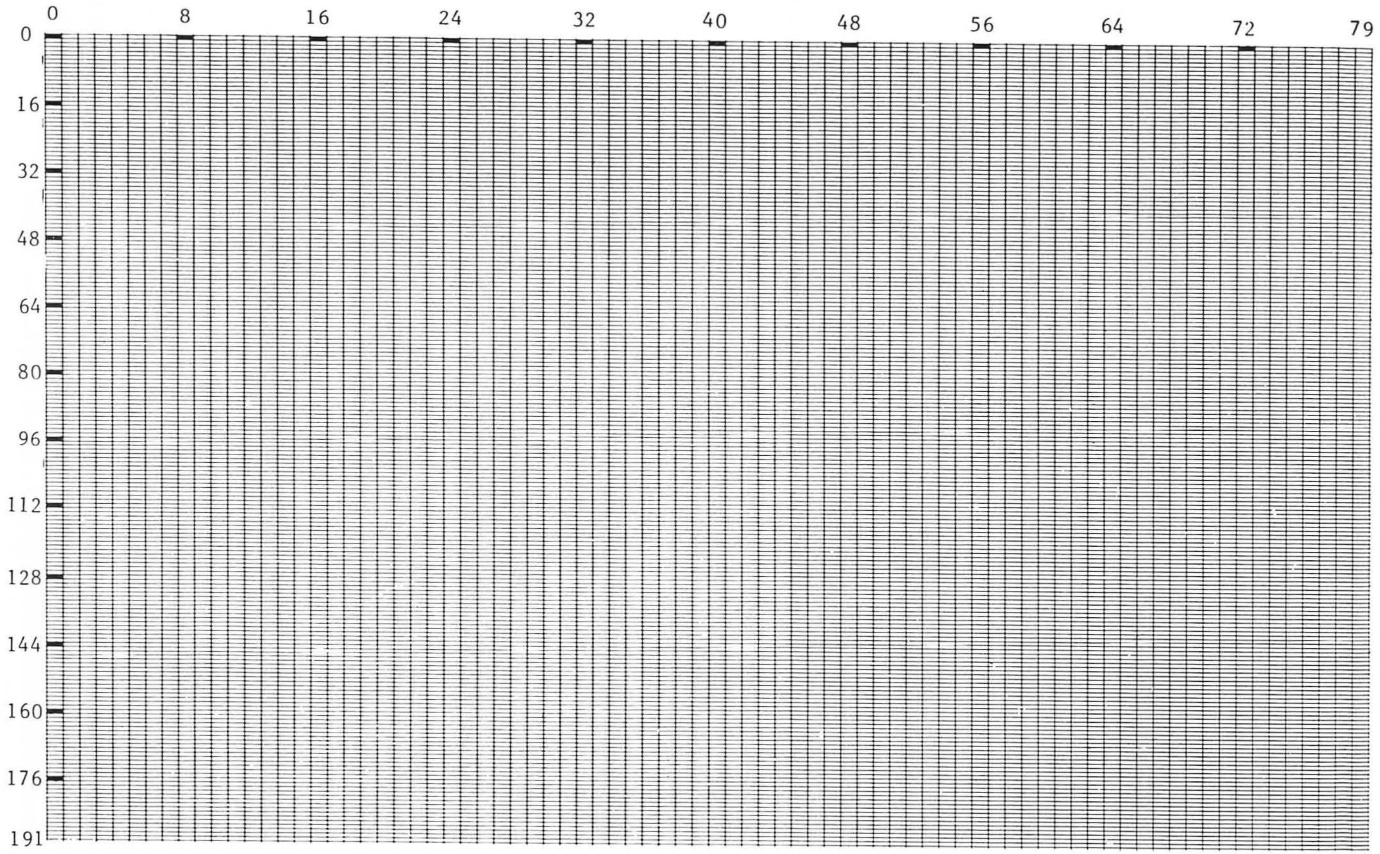
How To Program the GTIA Modes 9, 10, or 11 in BASIC

Item #1: There are no text windows in GR. 9, 10, or 11 (hence no immediate mode), so you must write programs in order to use the GTIA modes. These are graphics modes only, so you do not have to add a " + 16 " to get rid of the text window.

Item #2: The resolution of Graphics Modes 9, 10, and 11 are all the same: 80 × 192. This is unusual since there are fewer points across than down.

Use the following chart for plotting out your GTIA programs.

Graphics Mode 9, 10, or 11



GTIA

Programming with GRAPHICS 9

In order to select your basic color use SETCOLOR as follows.

```
SETCOLOR 4,____,0
```

Always use a 4 here.

A zero goes here to give you a full range of shades.

A number (0 through 15) goes here. A 0, for instance, would give you shades of Gray and 4 would give you shades of Red. Refer to the basic 16 colors listed in the section on Color.

To select how bright a dot or line will be, use *COLOR* followed by a number, *0 through 15*. (0 is the darkest and 15 is the lightest.)

NOTE: If you're real quick you may have noticed that GRAPHICS 9 is a special mode that allows for the selection of 16 colors plus 16 shades so in this mode there are actually 256 colors to choose from!

Here's an example that produces lines that are two different shades of red on a dark red background.

```
10 GRAPHICS 9
20 SETCOLOR 4,4,0
30 COLOR 12: REM LIGHT RED LINE
40 PLOT 39,2: DRAWTO 39,191
50 PLOT 40,2: DRAWTO 40,191
60 COLOR 4: REM DARK RED LINE
70 PLOT 41,2: DRAWTO 41,191
80 PLOT 42,2: DRAWTO 42,191
90 GOTO 90
```

The following program draws a smoothly shaded bar across the screen, flips through all 16 colors by alternating the middle value (the variable FL) in the SETCOLOR statement in line 100.

```
5 REM GTIA BAR GR. 9
10 GRAPHICS 9
20 FOR Z=64 TO 80
30 COLOR Z
40 PLOT 2,Z: DRAWTO 79,Z
50 NEXT Z
60 FOR K=95 TO 80 STEP -1
70 READ X: COLOR X
80 PLOT 2,K: DRAWTO 79,K
90 NEXT K
100 FOR FL=0 TO 15: SETCOLOR 4,FL,0: FOR
 DE=1 TO 300: NEXT DE: NEXT FL: GOTO 100
110 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12,
13,14,15
```

GRAPHICS 10

Like GRAPHICS 9 and 11, the resolution of GRAPHICS 10 is 0 to 79 across and 0 to 191 down. The advantage of GR.10 over all the other modes available is that you cannot only choose 8 colors and also the luminance of each of those colors, as well as one additional color for the background. The dues you pay for this flexibility is that GR.10 is the most awkward mode to assign color and luminance values to. You need COLOR, POKEs, and SETCOLOR to use GR.10 in BASIC.

The easy thing to get used to is the fact that you *use COLOR 0 through 8* to determine the color of the different color points or lines you draw.

The hard part is getting the specific colors you want into the right registers. The following chart should help immensely:

GRAPHICS 10		DEFAULT POKE VALUE	POKE LOCATION
COLOR 0	NO SETCOLOR	0	704 —Background
COLOR 1	NO SETCOLOR	0	705
COLOR 2	NO SETCOLOR	0	706
COLOR 3	NO SETCOLOR	0	707
COLOR 4	SETCOLOR 0,_,_	40	708
COLOR 5	SETCOLOR 1,_,_	202	709
COLOR 6	SETCOLOR 2,_,_	148	710
COLOR 7	SETCOLOR 3,_,_	70	711
COLOR 8	SETCOLOR 4,_,_	0	712

The following program first draws, then changes, the color bars. This shows how you can get 9 different colors on the screen at the same time.

```
10 GRAPHICS 10
20 FOR X=15 TO 65
30 N=INT(RND(0)*8)+1
40 COLOR N: REM GIVES RANDOM COLOR(1 TO
 8) TO LINE DRAWN
50 PLOT X,2: DRAWTO X,191
60 NEXT X
70 REM THIS SECTION PUTS RANDOM COLORS
 INTO POKE LOCATIONS 704 TO 712 OVER A
ND OVER
80 FOR COLPOKE =704 TO 712
90 K=INT(RND(0)*255)
100 POKE COLPOKE,K
110 NEXT COLPOKE
120 FOR DE=1 TO 100: NEXT DE
130 GOTO 80
```

GRAPHICS 11

In GRAPHICS 11 you can put 16 colors on the screen at one time. In addition, you can control the luminance of the colors you put on the screen.

In other words, GR. 11 is like having a rainbow available, and being able to turn a knob and make that rainbow as light or as dark as you want.

The color of the line or point you draw to is determined by your use of COLOR 0 through 15. These colors represent the standard 16 colors with a default luminance value of 6.

```
10 GRAPHICS 11
20 FOR Z=0 TO 15
30 COLOR Z
40 PLOT 0,Z:DRAWTO 79,Z
50 NEXT Z
60 GOTO 60
```

This gives you your basic color program and the following change will give you some interesting results.

```
60 FOR LUMLOOP=0 TO 14:SETCOLOR 4,0,LU
MLOOP:FOR DE=1 TO 35:NEXT DE:NEXT LUML
OOP:GOTO 60
```

To change the brightness of the colors in GR. 11, change the luminance in SETCOLOR Register 4.

```
SETCOLOR 4,0,LUMINANCE
```

Put an even number from 0 to 14 here (0 is dark).

Here are the GRAPHICS 11 rules again:

- You do not have a text window at the bottom of the screen.
- The resolution is 80 (0 to 79) across and 192 (0 to 191) down.
- You can put any of 16 general spectrum colors on the screen at one time, but you don't have to use all 16 colors if you don't need them.
- Colors are selected by specifying a COLOR 0 through 15 before you do a PLOT or DRAWTO.
- You can make all 16 colors bright or dark by changing this value,

SETCOLOR 4,0,____

but you can't make some colors bright and some dark. For that, you need GRAPHICS 10.

```
5 GRAPHICS 11
10 FOR Z=0 TO 79
20 PLOT 0,Z
30 COLOR Z
40 DRAWTO Z,79
50 NEXT Z
60 FOR Y=79 TO 0 STEP -1
70 PLOT Y,Z
80 COLOR Y
90 DRAWTO Y,79
100 NEXT Y
110 GOTO 10
```

```
10 GRAPHICS 11
20 LL=191
30 RL=79
40 FOR ZEBRA=0 TO 48
50 REM GTIA HIWAY*FAST LANE*
60 COLOR ZEBRA
70 PLOT RL,LL
80 DRAWTO 0,0
90 RL=RL-1
100 NEXT ZEBRA
110 GOTO 110
```

```
5 GRAPHICS 11
10 FOR Z=1 TO 191
15 COLOR Z/12
20 PLOT 0,Z
40 DRAWTO 79,Z
50 NEXT Z
60 FOR FL=0 TO 14 STEP 2: SETCOLOR 4,0,
FL: FOR DE=1 TO 50: NEXT DE: NEXT FL
70 FOR FLBACK=14 TO 0 STEP -2: SETCOLOR
 4 ,0,FLBACK: FOR DE=1 TO 50: NEXT DE: NEX
T FLBACK: GOTO 60
```

Using ATARI Joysticks and Paddle Controllers

The ATARI Joystick and Paddle Controllers are the same ones that are used with the ATARI Video Game System. Not only are they a necessary part of your computer system if you want to play games, but they are fun to experiment with and great for educational programs. To help you use these controllers, ATARI put Joystick and Paddle programming aids into its BASIC language.

The Joy of Joysticking

The Joystick is most useful when you want movement that is up and down, or left and right.

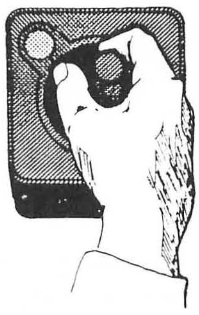
During its spare time, the ATARI computer checks to see if anyone out there is fooling around with a Joystick or Paddle Controller. It does this 60 times each second!

The computer interprets the position of the Joystick as certain numbers. Keep in mind, the numbers are predetermined. You don't program the numbers, you just have the computer do something if it finds a particular number.

Try this example...

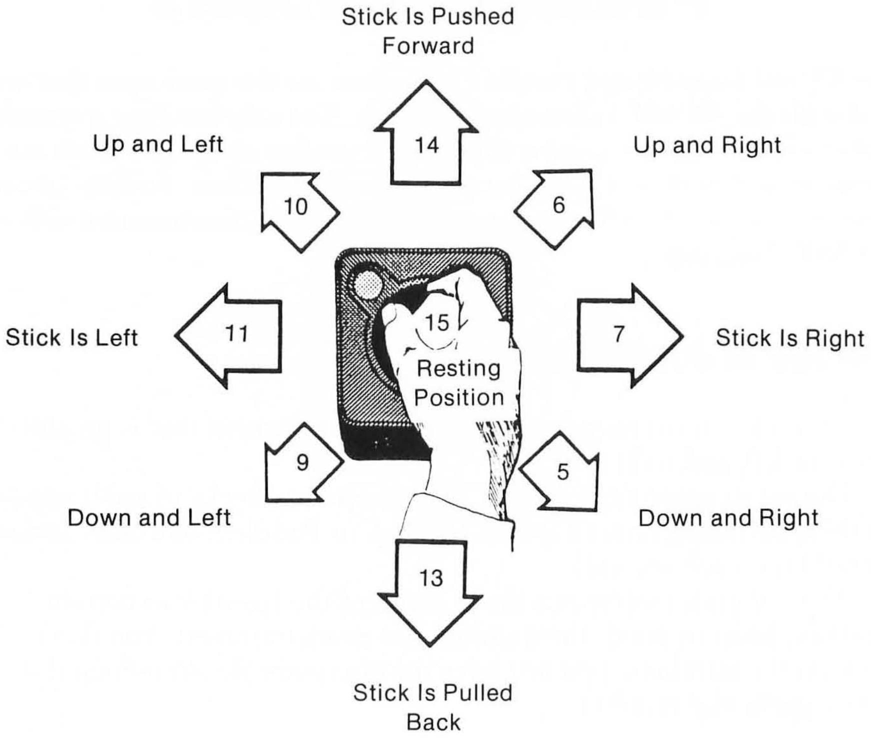
```
10 IF STICK(0) =14 THEN PRINT "HOW FORW
ARD OF YOU! "
20 GOTO 10
```

Hold the Joystick with the button in the upper left-hand corner.



Now RUN the program and see what happens when you push the Joystick forward.

Notice that the computer reads a 14 when it senses that the Joystick is pushed forward. The other 8 numbers it interprets are as follows...



Try this program:

```
10 PRINT STICK(0) : GOTO 10
```

RUN the program and move the stick around or change it to this...

```
10 X=STICK(0)
20 PRINT X
30 SOUND 0,X*10,10,10
40 GOTO 10
```

NOTE: It is sometimes easier to work with Joysticks when you assign them variable names such as...X=STICK(0).

Here's a program you can use to teach someone the points of a compass.

```
10 PRINT "}": POKE 752,1: X=STICK(0)
20 IF X=14 THEN POSITION 17,2: PRINT "N
ORTH"
30 IF X=13 THEN POSITION 17,22: PRINT "
SOUTH"
40 IF X=11 THEN POSITION 2,12: PRINT "W
EST"
50 IF X=7 THEN POSITION 34,12: PRINT "E
AST"
60 IF X=6 THEN POSITION 29,4: PRINT "NO
RTH EAST"
70 IF X=5 THEN POSITION 29,20: PRINT "S
OUTH EAST"
80 IF X=10 THEN POSITION 2,4: PRINT "NO
RTH WEST"
90 IF X=9 THEN POSITION 2,20: PRINT "SO
UTH WEST"
110 IF X=15 THEN POSITION 0,12: PRINT "
YOU WILL HAVE NO DIRECTION IN LIFE UNT
IL         YOU MOVE YOUR JOYSTICK"
120 FOR DELAY=1 TO 500: NEXT DELAY
130 GOTO 10
```

STRIG

STRIG stands for STICK TRIGger. The computer is also checking to see if the red button or trigger on the Joystick is being pushed. When the button is depressed, the computer gets the number 0 instead of a number 1.

RUN the following program and press the joystick button to get the idea.

```
10 PRINT STRIG(0)
20 GOTO 10
```

Or to get a simple video game type of shot fired, type and run this program:

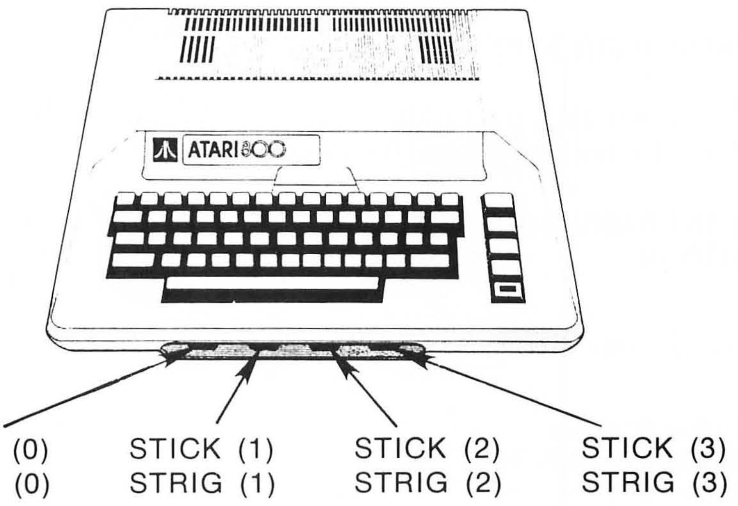
```
10 REM STIKFIRE
20 GRAPHICS 3+16
30 SETCOLOR 0,2,14: REM BRIGHTENS DOT
40 IF STRIG(0)=0 THEN GOTO 60
50 GOTO 10
60 FOR T=0 TO 38: SOUND 0,T,8,10
70 COLOR 1: PLOT T,12: COLOR 0: PLOT T,12
: NEXT T
80 SOUND 0,0,0,0
90 GOTO 40
```

A final Joystick program for you to try is this Video Sketchpad Program:

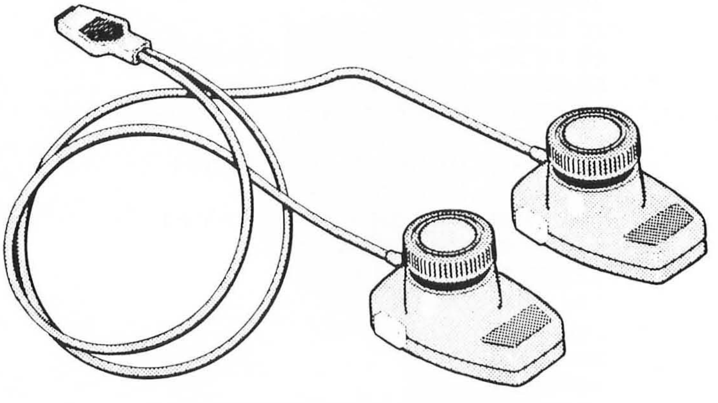
```
10 GRAPHICS 7+16: X=80: Y=40
20 ST=STICK(0)
30 IF ST=7 THEN X=X+1
40 IF ST=6 THEN X=X+1: Y=Y-1
50 IF ST=14 THEN Y=Y-1
60 IF ST=5 THEN X=X+1: Y=Y+1
70 IF ST=11 THEN X=X-1
80 IF ST=10 THEN X=X-1: Y=Y-1
90 IF ST=13 THEN Y=Y+1
100 IF ST=9 THEN X=X-1: Y=Y+1
110 IF X>158 THEN X=159
120 IF Y>94 THEN Y=95
130 IF X<1 THEN X=0
140 IF Y<1 THEN Y=0
150 COLOR 1
160 PLOT X, Y
170 FOR DELAY=1 TO 5: NEXT DELAY
180 COLOR 0
190 IF STRIG(0) =0 THEN COLOR 1
200 PLOT X, Y
210 GOTO 20
```

NOTE: You will only draw when the Joystick button is pressed down, and you can also erase lines by retracing them without holding the button down.

Keep in mind that you can write programs using more than one Joystick as long as you refer to them properly.



Using the Paddle Controllers



Very Important Issue—Why are these things called Paddle Controllers? They are called Paddle Controllers simply because they are similar to the controllers used on the first Video Game "PONG"; hence (Ping Pong)—Paddle Controllers. You will find, however, that these have many uses besides game control.

Programming with the Paddle Controllers is similar to programming with the Joystick except instead of interpreting 9 positions from the Joystick the computer will recognize Paddle position from 1 to 228. To see this in action, RUN this program and move the Paddle around.

```
10 PRINT PADDLE(0) : GOTO 10
```

As with the Joystick, the Paddle trigger only returns one of two numbers...1 = not pressed and 0 = pressed.

```
10 PRINT PADDLE(0),PTRIG(0)
20 GOTO 10
```

Let's make some music now:

```
10 X=PADDLE(0)
20 SOUND 0,X,10,10
30 GOTO 10
```

And how about a little bit of color and graphics...

```
10 SETCOLOR 2,A,2: COLOR 1
20 LET A=PADDLE(0)/6.2
30 POKE 82,A: PRINT "**"
40 SOUND 0,A,10,10
50 IF PTRIG(0)=0 THEN SOUND 0,A,6,A
60 GOTO 10
```

A FINAL NOTE: Don't get caught up programmer's creek without a PADDLE (0).

FUNCTIONS

Functions are things that you wish the computer would do automatically...and it does!

Four function calculators (called "four bangers") perform addition, subtraction, multiplication, and division.

More complex calculators have more automatic functions such as square root and percentage. Small computers, however, are easily capable of creating almost any function or subroutine as it is needed. Only some of the more complex and frequently used functions are included in the ATARI BASIC Instruction Set.

All of the ATARI BASIC functions, including trigonometric functions are described in the ATARI BASIC Reference Manual, and you should consult that before you attempt any sophisticated mathematical programs.

What is presented here is a general introduction to some of the functions you might use when first learning BASIC.

Some functions or automatic routines you may find useful or interesting are...

INT, RND, SQR, ASC, CHR$, and LEN

INT

INT stands for INTEGER. This function gives you a whole number instead of a fraction or decimal number. It always rounds the number down to the nearest whole number. You use INT as follows...

(The number goes inside the parentheses.)

`PRINT INT(5.2340)`	would give you....5
`PRINT INT(9/2)`	would give you a 4 instead of a 4.5

You might be wondering why you would want to round your numbers off since it's not much extra work for the computer to keep them exact, here's a few reasons:

The average house has 23.212
windows. Your car gets 29.17934
MPG. And, of course, the old
standard...The average American
has 2.5 children.

This is the type of information people are used to getting from computers and although it may be accurate from a mathematical standpoint, it can make you and your computer sometimes sound silly. INT helps keep simple issues simple.

Also, you can check to see if a number is odd or even by using INT. Consider the following...

```
10 INPUT X
20 IF X/2=INT(X/2) THEN PRINT "YOUR NU
MBER IS EVEN"
```

Let's say X=9. In that case, X/2 would = 4.5 but INT(X/2) would = 4 because 4.5 is rounded down to 4 and the computer would not print "YOUR NUMBER IS EVEN."

Don't believe me, ask your computer.

```
10 PRINT "↖": PRINT "TYPE IN A NUMBER A
ND PRESS RETURN"
20 INPUT X
30 IF X/2=INT(X/2) THEN PRINT "YOUR NU
MBER IS EVEN": FOR DE=1 TO 400: NEXT DE:
RUN
40 PRINT "THAT'S ODD...TRY AGAIN": FOR
DE=1 TO 400: NEXT DE: RUN
```

You will find additional uses for the INT function including using it in conjunction with the RND function.

RND

RND stands for "RANDOM NUMBER". A number which is random has the same chance of coming up, or being picked, as any other number. After a number is selected, it gets put back into the hopper. It could get selected immediately or not for a long time. It is truly the luck of the draw.

There are two ways to get random numbers with the ATARI computer—one easy way is to PEEK a location which produces random numbers between 0 and 255, and the other is to use the RND function to create your own.

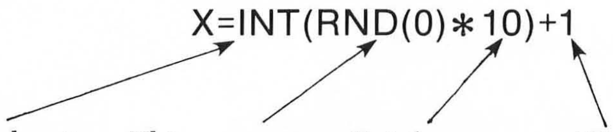
First use the PEEK method.

```
10 X=PEEK(53770)
20 PRINT X: GOTO 10
```

Run this to see the results.
Now hit it maestro...add

```
15 SOUND 0,X,10,10
```

The random numbers you PEEK are easy to use, but they are not as flexible as those you can get by using the RND function.



INT makes it a round number.

This never changes.

Put the highest random number you want here.

This gets rid of the zero and makes a 10 possible because the INT function would round a number such as 9.89984372 down to a nine.

Try this...

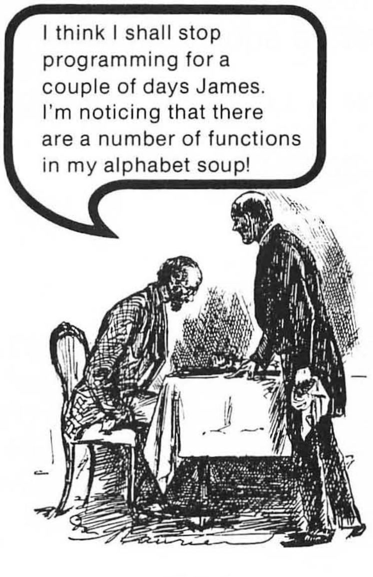
```
10 X=INT(RND(0)*10)+1
20 PRINT X
30 GOTO 10
```

Now change line 10 to ...

```
10 X=INT(RND(0)*1000)+1
```

In this simple game, the computer picks a number between 1 and 25 and you try to guess it.

```
10 PRINT "↰": POKE 752,1
20 X=INT(RND(0)*25)+1
30 PRINT "I'M THINKING OF A NUMBER BET
WEEN 1 AND 25..GUESS THE NUMBER AND PR
ESS RETURN"
40 POSITION 18,11: INPUT G
50 IF X=G THEN GOSUB 80: POSITION 16,7:
PRINT "CORRECT! ": FOR Z=0 TO 254 STEP 2
: POKE 712,Z: SOUND 0,Z,10,10: NEXT Z: RUN

60 IF X>G THEN POSITION 10,7: PRINT "TO
O LOW...TRY AGAIN! ": POSITION 18,11: PR
INT "      ": GOTO 40
70 IF X<G THEN POSITION 10,7: PRINT "TO
O HIGH...TRY AGAIN! ": POSITION 18,11: PR
INT "      ": GOTO 40
80 POSITION 10,7: PRINT "
          ": RETURN
```


I think I shall stop programming for a couple of days James. I'm noticing that there are a number of functions in my alphabet soup!

SQR

The SQR function gives you the SQUARE ROOT of a number...
It's used like this...

```
PRINT SQR(100)          would give you 10
PRINT SQR(25)           would give you 5
```

Remember, you don't use SQR to square a number. Instead you use the following:

```
PRINT 10^2              This gives you 99.99999998
                                   or
PRINT 5^2               which gives you 24.9999993
```

NOTE: All computers use different formulas for squaring numbers. If you need to have an exact SQUARE, you can increase the squared number by adding .5 so it is raised up to the next highest whole number. Then, use the INT function to round it down to a whole squared number.

```
10 PRINT "↰": POKE 752,1: PRINT "TYPE IN
 THE NUMBER YOU WANT TO SQUARE AND PRE
SS RETURN"
20 INPUT X: POSITION 7,11: PRINT "THE SQ
UARE OF ";X;"IS...";INT((X^2)+0.5)
30 FOR DE=1 TO 700: NEXT DE: RUN
```

ASC and CHR$

For every letter, number, and character on your ATARI Computer keyboard, there is a code number. For the standard numbers and keys, the code is a computer industry standard ASCII. (Pronounced Ass kē)

For the special characters found on the ATARI Computer, an extended version of ASCII code is used. This is referred to as ATASCII which is simply an abbreviation for ATARI-ASCII.

To see the code for the letter A, type...

```
PRINT ASC("A")
```

You should get a 65 when you press RETURN. And if you type...

```
PRINT CHR$(65)
```

You should get the letter A.

(This instruction tells the computer to PRINT the "character string" for the ASCII code #65.)

PRINTER NOTE: Since some of the codes are special to the ATARI Home Computer, it is usually not possible to print the entire list of CHR$ equivalents on most computer printers.

On your computer screen, however

(Any letter or character goes here.)

PRINT ASC (" ") will give you the code number

and

(Any number from 0 to 255 goes here.)

PRINT CHR$ ( ) will give you the character represented by the code #.

To see (and hear) all of the code numbers and characters available, use the following program.

```
10 FOR Z=0 TO 255
20 PRINT CHR$(Z)
30 NEXT Z
```

Did you notice that one of the code #'s rang the bell of the computer...Namely CHR$(253)

Try this maddening program...

```
10 PRINT CHR$(253) : GOTO 10
```

In order to type on the keyboard and get the code # first and then display the CHR$, use the following program...

```
10 GRAPHICS 2+16: PRINT #6; "SIMPLE TOUC
H TYPING TUTOR"
20 OPEN #1, 4, 0, "K: "
30 GET #1, X
40 POSITION 9, 5
50 PRINT #6; CHR$ (X)
60 SOUND 0, 30, 10, 4: FOR DE=1 TO 75: NEXT
  DE: POSITION 9, 5: PRINT #6; " ": SOUND 0,
0, 0, 0
70 GOTO 30
```

NOTE: PRINT CHR$(34) will print quotes on the screen
and
ASCII codes can be used to alphabetize words. The upper-case (capital) A is equal to 65, B=66, etc.

LEN

LEN will tell you how many characters there are in a string variable...PRINT LEN (A$)

```
10 DIM NAME$ (20) : PRINT "WHAT IS YOUR F
IRST NAME? ": INPUT NAME$
20 PRINT "THERE ARE "; LEN (NAME$) ; " LET
TERS IN YOUR FIRST NAME"
```

or you can use LEN to center outputs...

```
5 PRINT "↖": POKE 752, 1
10 DIM NAME$ (30) : PRINT "WHAT IS YOUR F
IRST AND LAST NAME? ": INPUT NAME$
20 PRINT "↖": POSITION (40-LEN(NAME$) ) /
2, 11: PRINT NAME$
30 PRINT : POSITION 15, 13: PRINT "IS CEN
TERED"
40 GOTO 40
```

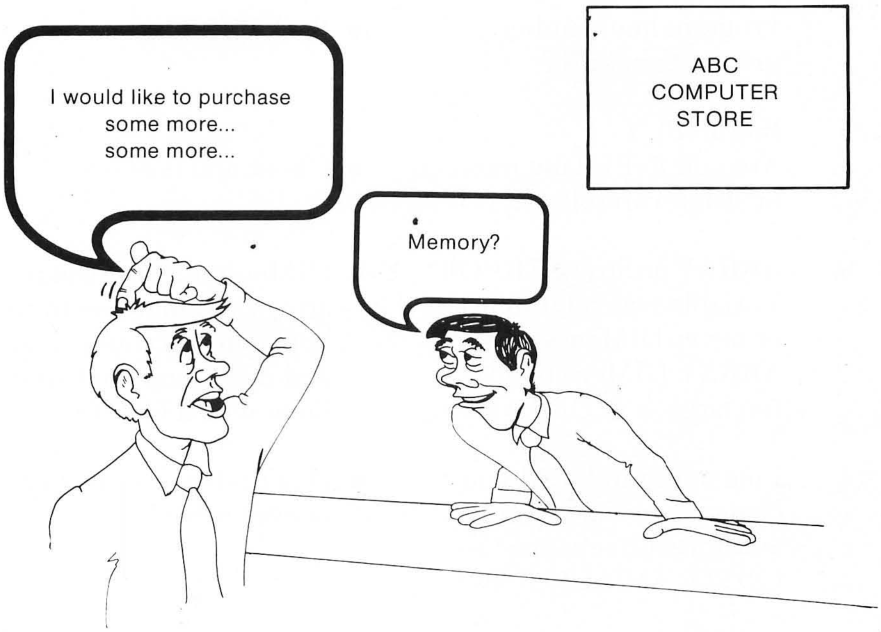
NOTE: For more information on ATARI Computer functions, be sure to check the ATARI BASIC Reference Manual.

ERROR Messages

This is a limited view of the most common ERROR messages. Consult The ATARI BASIC Reference Manual and the ATARI DOS II Manual for complete details.

Common ERROR Messages

Error #	Probable Causes	Some Solutions
2	Not enough Ram memory.	● Get more RAM ● Learn how to program more efficiently ● Learn assembly language


I would like to purchase some more... some more... some more...
Memory?
ABC COMPUTER STORE

Error #	Probable Causes	Some Solutions
3	Value ERROR. You tried to do something like put a negative number into a sound statement or divide by zero.	• Don't do that.
4	Too many variables, you are limited to 128.	• Cut down on variables. • Use ARRAYS.
6	Out of data ERROR. You did not have a proper flag in your data statement. Or, if you were given an ERROR 6 without a line #, you pressed RETURN when the cursor was on top of the READY on the screen.	• In case of the former, create a flag and check for it. • In case of the latter, learn to not care—it doesn't hurt anything.
7	Program line number greater than 32767	• Back off!
8	Bad INPUT. Are you INPUTing numeric or string variables?	• Check and correct.
9	ARRAY or String ERROR. Variable was reDIMensioned or never DIMensioned. Or ARRAY DIMension was too large, e.g. DIM A(5780)	• DIMension your strings, arrays and matrices in the beginning of your program and don't loop back over them with a GOTO.
12	Line not found. Did you change your line numbers and forget to change the GOTOs AND GOSUBs?	• Don't refer to a line that doesn't exist.

Error #	Possible Causes	Some Solutions
13	No matching FOR even though a NEXT was encountered.	• Look for the FOR statement or an unnested loop.
16	RETURN ERROR	• You might have deleted a GOSUB and left in the RETURN.
130	Device specified is non-existent or not hooked up.	• Check to see the appropriate peripherals are properly connected to the computer.
138	DEVICE TIME OUT. This can occur for a number of reasons and can occasionally be very frustrating.	• Connect any unconnected cables. Turn "ON" any "OFF" devices. With cassette programs, rewind cassette and try again.
139	Device NAK. (Device nonacknowledgement)	• Un-NAK the Device. Look at your ERROR 138 instructions and try again.
141	Cursor out of range. You have cursed too much or drawn, or positioned out of range of your present graphics mode, e.g., in Graphics 3, you cannot PLOT 45,20 and in Graphics 2, you cannot POS 2,15.	• This one will get you every time. Keep checking and you'll probably find you have overextended yourself.
143	Checksum ERROR. (Something-did-not-work= ERROR.)	• With the cassette, rewind cassette and try again.

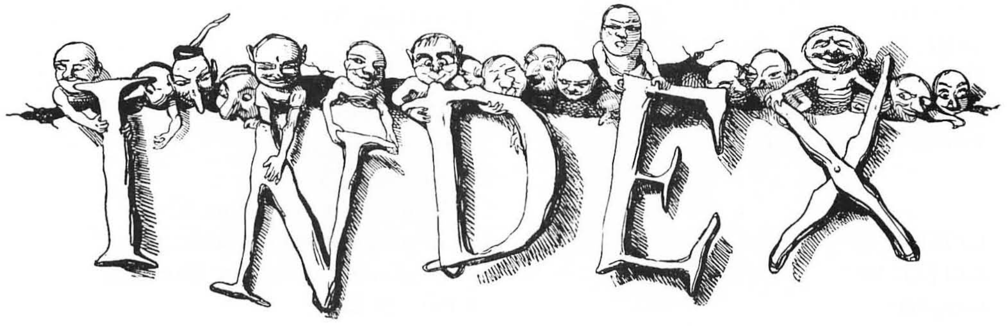
Error #	Possible Causes	Some Solutions
144	Problems reading or writing on diskette.	● Check to see that you don't have a write-protect tab on your diskette. Try RUNning the program again; your drive might be slightly incompatible with your diskette.
147	Not enough RAM for your graphics mode.	● Don't use this graphics mode. ● See ERROR #2.
170	File not found. Did you spell it right? Or... did you have a (·) between the file name (program name) and the extender? Or the wrong diskette?	● If you have a disk, type DOS RETURN A RETURN RETURN (Check spelling of file.) Type B RETURN ● Try RUNning the program again.

Conclusion

I have tried with this book to include only the necessary, fun or interesting. Because of the nature of the subject, however, I suspect I have left out some of the former and know I have left out heaps of the latter. Hopefully, I have whetted your appetite by helping you nibble on some of the capabilities of the ATARI Home Computer.

There are many fine books and articles currently being published on such ATARI Computer features as display list interrupts, fine scrolling and player-missile graphics. These are not easy subjects to master, but if you were patient enough to successfully master what was presented here, dive in!

These "Earthlings" were somewhat advanced for their time. Observe here, for instance, at least a rudimentary understanding of Player Missile Graphics.


INDEX

A VALUABLE WORKBOOK FOR THREE DISTINCT GROUPS OF PEOPLE.

First and foremost, for those who have never programmed any type of computer before, this book assumes no prior knowledge of programming or computer operation. It also assumes that you do not wish to spend the next ten years learning how computers work.

Next, there are some of you who have used computers before and may even be BASIC wizards, but are new to ATARI BASIC. Each computer is different and this book allows you to skip over what is familiar and concentrate on the specifics such as screen editing, graphics and color.

Finally, there are significant parts of this book which will prove useful to seasoned programmers who are familiar with ATARI Home Computers but need quick references to color, GTIA, and other graphics modes.

Inside ATARI BASIC takes the mystery and confusion out of learning to operate your home computer. You'll find it intentionally avoids flow charts, unnecessary technical details, and much of the "computerese" used in other books. Plain language and painless learning, with an entertaining writing style, makes this book the perfect companion for your ATARI Home Computer!

See page 160 for the listing of the program shown on the cover.

ATARI®

A Warner Communications Company

Atari, Inc.
Home Computer Division
P.O. Box 61657
Sunnyvale, CA 94086

0-8359-3081-5

C060992 REV. A.