

Programmer's Reference Guide to the

# COLOR COMPUTER

Clear, easy-to-use explanations of  
BASIC and Extended BASIC commands  
for the Radio Shack Color Computer,  
plus more than 20 programs  
you can type in and run.

C. Regena

A **COMPUTE!** Books Publication



# A BASIC Glossary

## An Applications Book

Good news for Color Computer users! C. Regena, author of the popular *Programmer's Reference Guide to the TI-99/4A* and well-known columnist for *COMPUTE!* Magazine, has brought her expertise to the TRS-80 Color Computer. Her clear explanations and example programs show you how to use every command and function in Color Computer BASIC and Extended BASIC.

For the beginning programmer, exploring the BASIC language and learning the *ins* and *outs* of each command, the BASIC glossary is an invaluable tool. Each command has a clear explanation of its use, as well as a short program showing that command in action. As you type in and try these programs, you immediately see the command at work and are reminded of its particular abilities.

For the more experienced programmer, the glossary serves as a valuable reference and accompanies more than 20 full-fledged programs, ranging from data sorters to graphic demonstrations. Studying these programs will help you write fully developed ones. As you experiment, you'll also come up with many programming ideas of your own.

You will use this book constantly, referring to it again and again as you learn and improve your BASIC programming skills.

Programmer's Reference Guide to the

# COLOR COMPUTER

C. Regena

**COMPUTE!** Publications, Inc. 

A Subsidiary Of American Broadcasting Companies, Inc.

Greensboro, North Carolina

Radio Shack Color Computer is a trademark of Tandy, Inc.

Copyright 1984, COMPUTE! Publications, Inc. All rights reserved.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the United States Copyright Act without the permission of the copyright owner is unlawful.

Printed in the United States of America

ISBN 0-942386-19-1

10 9 8 7 6 5 4 3 2 1

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is a subsidiary of American Broadcasting Companies, Inc., and is not associated with any manufacturer of personal computers. Radio Shack Color Computer is a trademark of Tandy, Inc.



# Contents

<b>Foreword</b> .....	v
<b>Preface</b> .....	vii
<b>Part One: BASIC Glossary</b> .....	1
<b>Part Two: Programming Tips and Hints</b> .....	93
Typing and Editing .....	95
POKE Commands .....	96
Conserving Memory .....	96
Programming Ideas .....	97
<b>Part Three: Programs</b> .....	101
Utility Programs .....	103
Carriage Return — Line Feed .....	103
POKEs for Teletype .....	104
Letters .....	104
Sorts .....	106
Graphics .....	108
Flag .....	108
William Shakespeare .....	110
Circles .....	112
Lines .....	112
Boxes .....	113
Dice .....	114
Mathematics .....	115
Homework Helper — Division .....	116
Find All Factors .....	120
Prime Factors .....	120
Greatest Common Factor .....	121
Least Common Multiple .....	123
Homework Helper — Fractions .....	124
Simultaneous Equations .....	135
Computer-Aided Instruction .....	137
Typing Unit I .....	138
Typing Trainer .....	146

Buying Items .....	154
Earning Money .....	156
Learn the Teeth .....	159
New England States .....	164
<b>Part Four: Appendix .....</b>	<b>169</b>
Cassette Recorder .....	171
Color Codes .....	171
Color Set Chart .....	171
ASCII Character Codes .....	172
Charts	
Graphics Screen Worksheet .....	174
PRINT@/Screen Locations .....	175
Set/Reset Worksheet .....	176
<b>Index .....</b>	<b>177</b>

# Foreword

Welcome to the *Programmer's Reference Guide to the Color Computer*, COMPUTE!'s first book solely for the TRS-80 Color Computer. C. Regena, a columnist already well-known for her expertise on the TI-99/4A, Commodore VIC, and Commodore 64, has applied her knowledge of BASIC programming to yet another personal computer, the TRS-80 Color Computer.

You'll find the glossary section invaluable whether you're learning to program the Color Computer or just refreshing your memory of little-used commands. From ABS to VAL, each command and function has been outlined by the author with a clear description of its use, one-line program examples, and even short programs illustrating its abilities.

In addition to the explanations, C. Regena also shows you Color Computer BASIC in action, with many fully developed programs and utilities you can study — or just type in and use. These programs demonstrate the full capabilities of the Color Computer, as well as provide ideas for programs of your own.

As with all COMPUTE! Books, the clear, concise writing and easy-to-follow format make it simple to use this book as both a valuable reference guide and a source of creative and useful programs.

Our thanks to the members of our editorial and production staffs who assisted in the development of this guide.

# Preface

This *Reference Guide* has been designed as a handbook for programmers who may need a quick reminder of how to use a BASIC command. The first section of this book is a glossary of BASIC programming words listed alphabetically. The listings for each word show the syntax, explain parameters, give a general description of use, and present a sample program.

The second section of the book is a "random sampling" of programming tips and hints — tidbits which may help you in your programming.

The third section contains programs illustrating different programming techniques on the Color Computer — in a variety of applications. I hope you can use these programs and then glean some of the techniques to use in your own programs.

The Appendix contains charts and tables of codes or numbers that are often needed in programming.

This Guide is limited to the BASIC programming language for the TRS-80 Color Computers. Machine language commands are not included.

My goal in writing this book was to offer you an easy way to look up commands as you are programming — to find out where you might need commas or what the number limits are. I also wanted to present some ideas to help you in your own programming efforts. I would like you to *enjoy* BASIC programming on your TRS-80 Color Computer!

C. Regena

**One**

---

**BASIC  
Glossary**



## **ABS**

ABS(*n*) computes the absolute value of the numeric expression *n*. The expression *n* is evaluated and must be a number from  $-10^{38}$  to  $10^{38}$ . If the number is positive or zero, the value returned is equal to that number. If the number is negative, the value returned is the number part of *n* without the negative sign.

### **Valid statements**

```
PRINT ABS(10)
300 Y=ABS(SC)
500 X=X+ABS(D+3)
```

### **Sample program**

```
100 CLS
110 FOR I=1 TO 5
120 READ A
130 PRINT "A = ";A,"ABS(A) =";ABS(A)
140 PRINT
150 NEXT I
160 DATA 3,-3,10,4,-123
170 END
```

### **Sample run**

```
A = 3      ABS(A) = 3
A = -3     ABS(A) = 3
A = 10     ABS(A) = 10
A = 4      ABS(A) = 4
A = -123   ABS(A) = 123
```

## **AND**

AND is used in IF-THEN statements to combine relational expressions for a conditional branch. AND indicates that both relations must be true for the condition listed after the word THEN to be executed. AND may also be used to show a true or false condition as numerical results. -1 is true; 0 is false.

# AND

---

## Valid statements

```
400 IF A$ = "YES" AND B$ = "YES" THEN 500
600 IF T > 100 AND L = 2 THEN PRINT "YOU WIN"
```

This program demonstrates both uses of AND. Notice that both statements are true. You can easily change the values to provide a false statement.

## Sample program

```
100 A=0: B=0
110 IF A=0 AND B=0 THEN PRINT "TRUE"
120 C=(A=0) AND (B=0):PRINT C
130 END
```

Also see NOT, OR

# ASC

ASC(s) returns the ASCII code for the first character of string s. ASCII is the American Standard Code for Information Interchange. The Appendix has a table of ASCII codes.

## Valid statements

```
PRINT ASC("K")
100 IF ASC(A$) = 32 THEN END
300 B = ASC(B$) - 64
```

This sample program will provide the ASCII code for each key pressed.

## Sample program

```
100 PRINT "KEY PRESSED", "ASC(K$)"
110 PRINT
120 K$=INKEY$
130 IF K$="" THEN 120
140 IF ASC(K$)=8 THEN PRINT " ",ASC(K$):GOTO
    120
150 PRINT K$,ASC(K$)
160 GOTO 120
170 END
```

Also see CHR\$

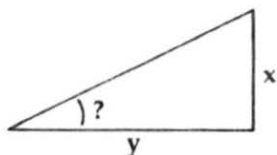
# ATN

## Extended BASIC only

ATN(n) returns the arc tangent of the numeric expression n. Arc



*tangent* means "the angle whose tangent is" and will be expressed in radians. To get the angle in degrees, multiply by  $180/\pi$  or  $(180/(4*ATN(1)))$  or 57.29577951.



$$? = ATN(x/y)$$

### Valid statements

```
PRINT ATN(1/2)
PRINT ATN(3)
100 R = ATN(X)
200 D = ATN(X)*(180/(4*ATN(1)))
```

Using this sample program, you can arrive at the arc tangent expressed in both radians and degrees.

### Sample program

```
100 CLS
110 PRINT "ENTER A NUMBER"
120 INPUT N
130 PRINT "ATN(N) = "; ATN(N); "RADIANS"
140 D = ATN(N) * (180 / (4 * ATN(1)))
150 PRINT TAB(8); D; "DEGREES"
160 PRINT
170 GOTO 110
180 END
```

Also see TAN

## AUDIO

AUDIO ON connects the sound from the cassette recorder to the television or monitor speaker when the MOTOR ON command has turned on the recorder. You may use this command to have sound during programs.

AUDIO OFF turns off the sound.

### Valid statements

```
200 AUDIO ON
500 AUDIO OFF
```

# AUDIO

## Sample program

First prepare a cassette tape with your recorded voice, music, or sound effects by recording in the usual way without the computer connected. Connect the computer to the recorder with the three-pronged cable. Now try this sample program.

```
100 CLS
110 PRINT "MAKE SURE CASSETTE IS READY."
120 PRINT "PRESS 'PLAY' ON THE RECORDER."
130 PRINT
140 PRINT "PRESS THE SPACE BAR TO START."
150 A$=INKEY$:IF A$="" THEN 150
160 IF ASC(A$)<>32 THEN 150
170 MOTOR ON
180 AUDIO ON
190 PRINT
200 PRINT "PRESS ANY KEY TO STOP."
210 A$=INKEY$:IF A$="" THEN 210
220 AUDIO OFF
230 MOTOR OFF
240 END
```

Also see MOTOR

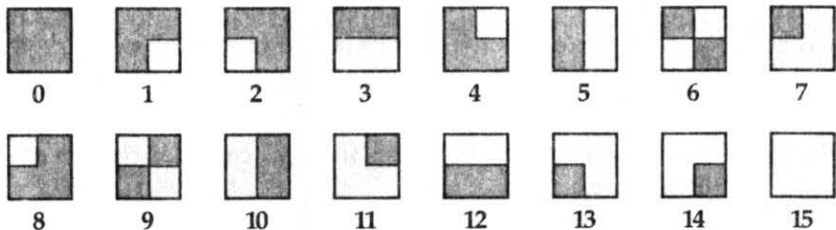
## CHR\$(n)

CHR\$(*n*) returns the character corresponding to the ASCII code number *n*. Refer to the ASCII code table in the Appendix.

Code numbers 128 to 191 are used for predefined graphics characters. To get different colors and shapes, use this formula:

$$\text{code} = 128 + 16 * (\text{color}-1) + \text{pattern}$$

where *color* is one of the color numbers from 1 to 8 and *pattern* is one of the following pattern numbers.



## Valid statements

```
PRINT CHR$(128)
100 A$="A "+CHR$(X)
120 IF CHR$(N)="Y" THEN 300
```

**Sample programs**

```

100 CLS
110 PRINT "  N{4 SPACES}CHR$(N) "
120 FOR N=65 TO 77
130 PRINT N;TAB(10);CHR$(N)
140 NEXT N
150 END

```

Changing the values in line 120 will, of course, provide you with other characters matching your new numbers.

This short program demonstrates the graphics-building abilities of the CHR\$ command.

```

100 CLS
110 A$=CHR$(191)+CHR$(191)+CHR$(191)
120 B$=CHR$(128)+CHR$(128)
130 C$=B$+CHR$(128)
140 PRINT @138,C$+C$+C$
150 PRINT TAB(10);CHR$(128)+CHR$(185)+B$+C$+
CHR$(182)+CHR$(128)
160 PRINT TAB(10);B$+CHR$(185)+C$+CHR$(182)+
B$
170 PRINT TAB(10);C$+A$+C$
180 PRINT TAB(10);C$+CHR$(190)+CHR$(191)+CHR
$(189)+C$
190 PRINT TAB(10);C$+A$+C$
200 PRINT TAB(10);C$+CHR$(187)+CHR$(179)+CHR
$(183)+C$
210 PRINT TAB(10);C$+C$+C$
220 END

```

Also see ASC

**CIRCLE****Extended BASIC only**

CIRCLE (*x,y*), *r,c,h,s,e* draws a circle on the screen with the center at location (*x,y*). The parameters are:

- |          |            |   |
|----------|------------|---|
| <i>x</i> | 0 to 255   | <i>x</i> coordinate of center point.  |
| <i>y</i> | 0 to 191   | <i>y</i> coordinate of center point.  |
| <i>r</i> | 0 to 65535 | Radius. You may specify a radius that puts only part of the circle on the screen. |
| <i>c</i> | 0 to 8     | Color number. Optional. Default value is foreground color.                        |
| <i>h</i> | 0 to 255   | Height to width ratio. Optional. Default value is 1.                              |

## CIRCLE

---

- s* 0 to 1 Starting point of circle. Optional. Default value is 0.
- e* 0 to 1 Ending point of circle. Optional. Default value is 1.

If you omit *c*, *h*, or *s*, you must still use commas if you use a later parameter.

To use the start and end options, you must use the ratio  $h=1$ . If you imagine a clock face, the start at 0 would be at 3:00. A value of .25 would be at 6:00, a value of .5 would be at 9:00, and .75 would be at 12:00.

### Valid statements

```
200 CIRCLE(127,95),10
300 CIRCLE(X,Y),90,3
400 CIRCLE(X+3,60),10,,5
500 CIRCLE(120,60),30,,1,.25,.75
```

Note in line 160 that the starting and ending points are not equal. What effect will that have on line 160's circle?

### Sample program

```
100 PMODE 3,1
110 SCREEN 1,1
120 PCLS
130 CIRCLE(30,30),15
140 CIRCLE(60,40),20,3
150 CIRCLE(127,95),30,2,2
160 CIRCLE(200,150),25,3,1,.1,.8
170 GOTO 170
180 END
```

## CLEAR

CLEAR sets all numeric variables to zero and string variables to null.

CLEAR *n* clears space for strings. The computer automatically reserves about 200 characters. To clear more, use a statement such as 100 CLEAR 1000.

CLEAR *n,m* clears *n* space for strings, and *m* sets the memory protection address. CLEAR 500,12000 clears string space of 500 bytes and reserves memory addresses from 12001 to the end of RAM so you can store a machine language program in that area.

**Valid statements**

```
100 CLEAR
100 CLEAR 500
100 CLEAR 600,4050
```

**Sample programs**

```
100 X=X+4
110 A$=STR$(X)
120 PRINT X,A$
130 GOTO 100
140 END
```

RUN and see the results. Press BREAK, add line 125 CLEAR, and RUN to see the difference. X and A\$ are returned to 0 and null in line 125.

```
100 PRINT MEM
110 CLEAR 200
120 PRINT MEM
130 CLEAR 500
140 PRINT MEM
150 CLEAR 1000
160 PRINT MEM
170 END
```

PRINT MEM prints the amount of memory available. NEW and RESET do not return the CLEARED space to normal. CLEAR 200 returns the computer to original conditions.

*Also see MEM*

**CLOAD**

CLOAD loads (reads in) a program from cassette tape to the computer.

1. Position the cassette tape.
2. Type NEW to clear the previous program.
3. Press PLAY on the cassette recorder.
4. Type CLOAD and press ENTER.

The computer will clear the screen and print S (for searching) in the upper-left corner. When a program is found, the message will change to F (found) and blink while the program is loading. If the program has been named, the message will be F NAME, where NAME is the name of the program. The message OK appears when the program has been loaded.

The CLOAD command will load the first program the computer

# CLOAD

---

finds on the tape. If you prefer to load a certain program and there are several programs on the tape, use CLOAD "NAME" for the NAME of the program. The computer will search for the correct program and skip all other programs.

## Valid commands

CLOAD

CLOAD "NOTES"

CLOAD "FRACTION"

*Also see* CSAVE, LOAD, SAVE

# CLOSE

CLOSE closes a file by ending communication with a particular device.

If you do not CLOSE a file, the communication channel remains open even if you don't use it. You may try to open the file later and will get an error. If a file has been OPENED for OUTPUT, you cannot INPUT items to the file and vice versa.

The devices are:

# 0            screen  
#-1           cassette recorder  
#-2           printer  
# 1 - #15    disk drive

## Valid statements

500 CLOSE #-1 (for cassette)

520 CLOSE #-2 (for printer)

540 CLOSE #5 (for disk drive)

## Sample program

```
100 CLS
110 PRINT "PRESS RECORD AND PLAY"
120 PRINT "ON CASSETTE,"
130 PRINT "THEN PRESS ENTER."
140 A$=INKEY$: IF A$="" THEN 140
150 IF ASC(A$)<>13 THEN 140
160 OPEN "0", #-1, "NAMES"
170 CLS
180 PRINT "TYPE A NAME"
190 PRINT "THEN PRESS ENTER."
200 PRINT:PRINT "TO END THE LIST,"
210 PRINT "TYPE ZZZ":PRINT
220 INPUT "NAME ";N$
```

```
230 IF N$="ZZZ" THEN 260
240 PRINT #-1,N$
250 GOTO 200
260 CLOSE #-1
270 END
```

Also see INPUT #, OPEN, PRINT #

## CLS

The command CLS clears the screen (erases everything) and starts back at the top-left corner for the next item to be printed (unless otherwise specified in the next PRINT statement). The screen will be green.

You may specify a screen color with a number  $n$  from 0 through 8 by using CLS( $n$ ). The printing will still be black with a green background. If you want the printing only (and not the rest of the line) to be black on green, use the semicolon after the print list.

### Valid statements

```
CLS
CLS 3
CLS(7)
```

### Sample program

```
100 CLS
110 PRINT "THE SCREEN WAS CLEARED."
120 FOR D=1 TO 1000
130 NEXT D
140 CLS(3)
150 PRINT "HELLO"
160 PRINT @170,"TITLE";
170 FOR D=1 TO 1000
180 NEXT D
190 END
```

Changing the values in line 120 will shorten or lengthen the delay between the screen clearing and the messages appearing. A subroutine such as this would be useful in the opening of a game program.

## COLOR

### Extended BASIC only

COLOR  $f,b$  sets the foreground  $f$  and background  $b$  colors used

by other graphics commands, where  $f$  and  $b$  are the color numbers 0 through 8. Keep in mind that you are limited to the two color sets.

If you do not use a COLOR statement, the default values are the highest-numbered available color code for the foreground and the lowest for the background.

## Valid statements

COLOR 2,3

COLOR 7,5

## Sample program

```
100 PMODE 3,1
110 PCLS
120 SCREEN 1,0
130 COLOR 3,4
140 CIRCLE (19,96),20
150 COLOR 2,4
160 LINE (0,0)-(191,96),PSET
170 COLOR 0,4
180 LINE -(30,50),PSET,B
190 COLOR 3,4
200 LINE -(50,70),PSET,BF
210 GOTO 210
220 END
```

See CIRCLE and LINE for further explanation of those commands' abilities and parameters. Combining those commands with COLOR allows you to better use the graphic capabilities of the computer.

## CONT

CONT continues a program after you have pressed the BREAK key or the program has encountered a STOP statement. RUN will restart the program from the beginning and clear all variables, but CONT will continue the program with all variables at the values they had before the breakpoint.

## Valid command

CONT

## Sample programs

```
100 X=X+4
110 PRINT X
120 GOTO 100
```



RUN this program for a while, then press the BREAK key. Then type CONT and press ENTER. The program continues.

```
100 Y=Y+2
110 PRINT Y
120 STOP
130 GOTO 100
```

RUN this program. It will print a value of Y then print BREAK IN 120. Type CONT and press ENTER. The program will print the next value of Y and stop. Press RUN again. Notice the value of Y is the same as the first time you ran the program. You can now see the difference between the CONT command and the RUN command.

## COS

### Extended BASIC only

COS(*n*) returns the value of the cosine of an angle *n* where *n* is expressed in radians. If the angle is in degrees, first multiply by  $\pi/180$  or  $(4*ATN(1))/180$  or 0.0174532925.

If the number *n* is greater than about 5.9 E +09, the value returned is zero.

### Valid statements

```
PRINT COS(2)
100 X=COS(A)
110 X=COS(A+B)+Y
```

### Sample program

```
100 CLS
110 PRINT "X", "COS(X)"
120 FOR X=0 TO 3.2 STEP .25
130 PRINT X, COS(X)
140 NEXT X
150 END
```

Notice that changing the value of the STEP command in line 120 will print different numbers under the X column. Refer to the sample program under ATN (arc tangent) to see how to write your own program for the COS command, which will show results in both radians *and* degrees.

## CSAVE

CSAVE is used to save a program on cassette tape.

1. Insert the cassette, making sure the tape is positioned

beyond the leader or past any previously saved programs or at a particular counter position.

2. Press the RECORD and PLAY buttons on the recorder at the same time until they lock.
3. Type CSAVE and press ENTER.

When OK appears, the program has been recorded.

You may name the program by using CSAVE "TITLE" where the name of the program may be up to 8 letters long.

To save in ASCII format, use the letter A after the title:

CSAVE "TITLE",A

In everyday use, you would probably use CSAVE. The ASCII option may be used if the program later would be edited by a utility or word processing program.

## Valid commands

CSAVE

CSAVE "FLAG"

CSAVE "GEORGE"

CSAVE "MATH1",A

*Also see* CLOAD, LOAD, SAVE

## DATA

DATA statements store data in a program. The data list may be numbers or strings, and the items must be separated by commas. If the string contains leading or trailing spaces, or embedded commas, it must be in quotes.

A DATA statement may be placed anywhere in the program. As the program is RUN, DATA statements are ignored until a READ statement, which is used to assign the data to variables, is executed. The first READ statement starts to read items at the beginning of the first DATA statement. There must be enough DATA to satisfy the READ statement or you will get an out-of-data error. The DATA type, either string or numeric, must correspond to the READ statement. Data items are read in order by the READ statement unless a RESTORE statement is used, which starts reading items from the beginning of the first DATA statement.

## Valid statements

300 DATA 7,5,3,2

400 DATA JOHN,MARY,DICK

```
500 DATA 3,SMITH,JIM,8403
600 DATA 2," PAPER"
```

### Sample program

```
100 CLS
110 FOR I=1 TO 10
120 READ ROOM,NAME$
130 IF ROOM<>3 THEN 150
140 PRINT NAME$
150 NEXT I
160 DATA 3,JOHN,4,JACK,1,JIM,3,JOE
170 DATA 2,JERRY,3,KENT,3,LARRY
180 DATA 3,RICK,2,BOB,1,ANDY
190 END
```

What effect will occur on the data printed if line 110 is changed to FOR I=1 TO 8?

Also see READ, RESTORE

## DEF FN

### Extended BASIC only

DEF FN allows you to define your own numeric function to use later in the program. It is helpful if you need to use the same function several times in a program. A function must be defined before it is used. Functions may be redefined anywhere in the program, nested functions are allowed, and a function may consist of only one variable. The form is:

DEF FN *name(argument)* = *definition expression*

The *name* may be any valid numeric variable. The *argument* can contain any valid variable, constant, or expression. The *definition expression* may be one line only (no colons), cannot contain verbs or other commands, but can call another function.

### Valid statements

```
100 DEF FN A(X)=X+3
120 DEF FN B(X)=SIN(X)+.5*SIN(2*X)
300 PRINT FN A(X)
400 ON FN J(2) GOTO 100,500,700
600 Y=FN B(J)+FN C(J)
```

### Sample programs

```
100 CLS
110 DEF FN PI(X)=4*ATN(1)
```

## DEF FN

---

```
120 FOR R=1 TO 4
130 PRINT "RADIUS = ";R,
140 PRINT "DIAMETER = ";2*R
150 PRINT "CIRCUMFERENCE = ";2*R*FN PI(X)
160 PRINT "AREA = ";R*R*FN PI(X)
170 NEXT R
180 END
```

Altering the values in line 120 will provide different results.

```
100 PMODE 4,1
110 SCREEN 1,1
120 PCLS
130 DEF FN S(X)=SIN(X)
140 LINE(0,96)-(255,96),PSET
150 FOR X=0 TO 25 STEP .3
160 Y=96-(40*FN S(X))
170 LINE(X*10,Y)-(X*10,96),PSET
180 NEXT X
190 GOTO 190
200 END
```

Note the slight difference in the results if you change SIN(X) to COS(X) in line 130.

## DEL

### Extended BASIC only

DEL deletes complete lines as you are editing a program. Although you can delete (get rid of) a line in your program by typing the number of the line, then by pressing ENTER, if you have more than one line it is easier to use DEL.

The forms are:

DEL 200	Deletes line 200.
DEL 300-500	Deletes all lines from 300 to 500, inclusive.
DEL -400	Deletes lines from the beginning of the program to line 400.
DEL 700-	Deletes from line 700 to end of program.
DEL-	Deletes the entire program from memory.

### Sample program

```
100 CLS
110 PRINT "HELLO"
120 PRINT "HERE ARE INSTRUCTIONS."
130 PRINT "USE ARROW KEYS."
140 PRINT "PRESS ENTER TO FIRE."
```

```
150 REM START GAME HERE
160 END
```

Try typing DEL 110 (press ENTER) then LIST. Line 110 will be missing.

Try DEL 120-140 and LIST.

Try DEL- and LIST.

## DIM

DIM is for DIMension and reserves space for arrays (numbered variables). The first subscript is zero. If you use an array without a DIMension statement, the computer automatically reserves space for eleven elements, ten plus the zero element, as if DIM ARRAY(10) had been specified. You may, however, save memory by DIMensioning arrays that use fewer elements. A DIMension statement is required for subscripts greater than ten. The number of dimensions you use depends on how large the arrays are. More than one dimension may be specified.

### Valid statements

```
100 DIM A(14)
100 DIM A$(15),P(15)
120 DIM B(5)
150 DIM A(3,5,5)
```

### Sample program

```
100 CLS
110 DIM NAME$(14),SCORE(14),PCT(14)
120 FOR I=1 TO 14
130 READ NAME$(I),SCORE(I)
140 PCT(I)=INT(SCORE(I)*100/27)
150 PRINT NAME$(I),SCORE(I);" ";PCT(I);"%
160 NEXT I
170 DATA ALAN,17,BILL,23,CHAD,26
180 DATA DAN,10,ELLA,17,FRED,21
190 DATA GINA,15,HAL,16,INEZ,22
200 DATA JOHN,21,KEN,18,LES,18
210 DATA MIKE,22,NED,21
220 REM PROGRAM WOULD CONTINUE
230 END
```

Note that the space reserved by the DIM command in line 110 matches the number of DATA elements in lines 170-210.

A DIMension statement is necessary before you use GET and PUT statements in graphics. You must create a two-dimensional

# DIM

array that matches the size of the rectangle of graphics you use. Since arrays have a zero element, the array may be one less than the dimension of the rectangle.

## Sample program

```
100 PMODE 4,1
110 PCLS:CLS
120 SCREEN 1,1
130 DIM S(26,16)
140 LINE(40,8)-(32,24),PSET
150 LINE-(50,12),PSET
160 LINE-(30,12),PSET
170 LINE-(46,24),PSET
180 LINE-(40,8),PSET
190 GET(28,8)-(52,24),S,G
200 J=28:K=8
210 FOR I=1 TO 7
220 PCLS
230 J=J+28:K=K+10
240 PUT(J,K)-(J+24,K+16),S,PSET
250 FOR D=1 TO 250:NEXT D
260 NEXT I
270 END
```

Also see GET, PUT

# DRAW

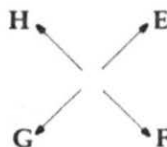
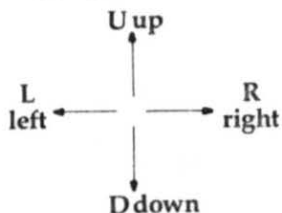
## Extended BASIC only

DRAW *l* draws a line where *l* is a string expression describing the line. There are several options:

- Mx,y Move to a certain point position x,y
- BMx,y Move without drawing (Blank Move) to a certain position x,y, where x and y are the coordinates on the graphics screen and x is a numeric expression from 0 to 255 and y a numeric expression from 0 to 191.

UDRL Directions to draw, with distance, such as U25.

EFGH



N	No update in starting position.
Cc	Color. <i>c</i> is a color number from 0 to 8. Default value is the foreground color.
Aa	Angle. <i>a</i> is the angle displacement and may be from 0 to 3 with the following angles: A0 rotate clockwise zero degrees A1 rotate clockwise 90 degrees A2 rotate clockwise 180 degrees A3 rotate clockwise 270 degrees Default value is A0.
Ss	Scale. <i>s</i> is the number of factors of 1/4 and can be from 1 to 62. Default value is S4.
B	Blank. The next line (only) is blank or invisible.
X	Execute a substring.

### Valid statements and sample program

Type this in before continuing.

```
100 PMODE 3,1
110 PCLS
120 SCREEN 1,1
240 GOTO 240
```

Add the following lines one at a time and RUN the program to see how the DRAW options work. Press the BREAK key to stop the program.

```
130 DRAW "BM125,96;U25;L15;D30;R10"
```

The line starts at location 125,96 and draws Up 25, Left 15, Down 30, and Right 10, where the numbers are units on the graphics screen. The comma between the *x,y* coordinates is required, as are the quotation marks, but the semicolons are optional. If you leave out any of the numbers after a direction, the computer will assume a value of 1.

```
140 DRAW "BM+10,-5;E8F8"
```

You may move a relative position from the previous position. The first number after BM must have a + or - sign to indicate relative position. (A + sign indicates Right, or Down; a - sign indicates Left, or Up.) If the second direction is positive, its sign is optional. This line will move ten units to the right and up five units from the end of the previous line.

```
150 DRAW "BM32,32;NG10;NF10;D40"
```

The N option means No update. This line will start at *x,y*

# DRAW

---

position 32,32. The first command is to draw in the G direction, but go back to 32,32. Next, the computer draws ten units in the F direction but returns to 32,32. D40 draws down 40, and the position remains at the end of the line.

```
160 DRAW "BM200,128;C3;G10F10E10H10"
```

The color option Cc may be placed anywhere in the DRAW statement. All actions which follow will be in the color c specified.

```
170 DRAW "BM50,50;S2;NG10;NF10;D40;BM+20,0;S8;NG10;NF10;D40;S4"
```

This line is drawn with a scale factor. The number after S indicates how many multiples of 1/4. For example, S2 means a scale factor of 2/4 or one-half. After the Ss command, all motions will be scaled until the next Ss command. Line 170 draws an arrow half-size, an arrow double-size, then returns the scale to normal, S4.

```
180 DRAW "BM+0,10;A1;NG10;NF10;D40"
```

Your drawing may be rotated using the A option. The statement above draws an arrow rotated 90 degrees in the same directions used previously.

```
190 DRAW "BM+0,15;R20;BR10;R20;BR10;R20"
```

B is the blank option. The next line only is drawn blank or invisible. Line 190 produces a series of dashes.

There are several ways to use string variables with the DRAW command. Following are two methods:

```
200 A$ = "BM+6,0;U8E4F4D4NL8D4"
```

```
210 DRAW A$
```

Above, we used the arrow several times. It would be easier to use a string variable for the instructions to draw an arrow, then execute the variable whenever we want. The procedure requires the X option. There must be a semicolon after the dollar sign of the variable name.

```
220 B$ = "NG10;NF10;D40"
```

```
230 DRAW "BM150,10;XB$;BM+20,0;XB$;BM+20,-40;S2;XB$;BM+20,0;C4;S4;A3;XB$;A0;BM-20,10;XB$;"
```

The program should now look like this

```
100 PMODE 3,1
```

```
110 PCLS
```



```

120 SCREEN 1,1
130 DRAW "BM125,96;U25;L15;D30;R10"
140 DRAW "BM+10,-5;E8F8"
150 DRAW "BM32,32;NG10;NF10;D40"
160 DRAW "BM 200,128;C3;G10F10E10H10"
170 DRAW "BM50,50;S2;NG10;NF10;D40;BM+20,0;S
      8;NG10;NF10;D40;S4"
180 DRAW "BM+0,10;A1;NG10;NF10;D40;A0"
190 DRAW "BM+0,15;R20;BR10;R20;BR10;R20"
200 A$="BM+6,0;UBE4F4D4NL8D4"
210 DRAW A$
220 B$="NG10;NF10;D40"
230 DRAW "BM150,10;XB$;BM+20,0;XB$;BM+20,-40
      ;S2;XB$;BM+20,0;C4;S4;A3;XB$;A0;BM-20,10
      ;XB$;"
240 GOTO 240

```

Another sample program

```

100 PMODE 3,1
110 PCLS
120 SCREEN 1,0
130 A$="UBE4F4D4NL8D4"
140 B$="U12R6F2D2G2NL6F2D2G2L6"
150 C$="H2UBE2R4F2;BM+0,8;G2L4"
160 LINE (60,60)-(188,130),PSET,BF
170 DRAW "S6BM80,120;C6;XA$;BM+12,0;XB$;BM+2
      0,0;XC$;S4;"
180 DRAW "BM204,120;A3;C7;XA$;BM+12,0;XB$;BM
      +20,0;XC$;"
190 GOTO 190

```

Also see the Mathematics and Computer-Aided Instruction sections for other samples of programs using the DRAW statement.

## EDIT

### Extended BASIC only

EDIT *l* allows you to change a program line *l* without retyping the line. Type EDIT, then the line number, then press ENTER. The line will appear as it is in the program, then the line number will appear at the left margin. You may use the following procedures to edit the line:

- SPACE Moves forward on the line without making changes.
- n*SPACE Moves forward *n* characters on the line without making changes.
- ← Moves to the left on the line without making changes.

- $n$  ← Moves to the left  $n$  characters without making changes.
- C Change. Put the cursor on top of the character to be changed; press C, then the new character.
- $n$ C Number of changes. Put the cursor on top of the first character to be changed. Press the number of characters to be changed, then C, then type that number of new characters. Note: Be careful with this editing procedure. Once you press a number, you have to type that number of new characters.
- D Delete. Put the cursor on top of the character to be deleted and press D. The character will be deleted.
- $n$ D Number of deletes. Put the cursor on top of the first character to be deleted. Press the number of characters to be deleted, then D.
- I Insert. Put the cursor on top of the character. The insertion will come before that character. Press I and insert the characters desired. You will be in insert mode until you press ENTER, which finishes editing that line, or SHIFT ↑, which stops insertion but keeps the cursor on the line being edited.
- L Lists the line again so you can continue editing.
- H Hacks off the rest of the line, then allows you to insert letters.
- X Extends the line. Press X to get to the end of the line, then add characters to the line. You can also back-space, which deletes those characters.
- K Kills or deletes the rest of the line. You can then press ENTER to complete the editing, or I or H to insert or add characters to the line.
- $n$ K $c$  Kills or deletes the line up to the  $n$ th occurrence of the character  $c$ .
- S $c$  Search. Press S, then the character  $c$ . The cursor will go to that character.
- $n$ S $c$  Search. Searches for the  $n$ th occurrence of the character  $c$ .
- E Exit without changing more.
- ENTER Completes editing of a line.

## ELSE

ELSE is used in an IF-THEN statement to give a command if the

condition is not true. ELSE may be followed by a statement number or a command. The IF-THEN statement does not have to include ELSE. The ELSE statement would be used if you do not want program flow to go directly to the line following the IF-THEN statement.

**Valid statements**

```
300 IF C=1 THEN 400 ELSE 500
300 IF X=Y+3 THEN PRINT "YES" ELSE PRINT "NO"
300 IF A$="YES" AND B$="YES" THEN 700 ELSE A=X+Y
```

**Sample program**

```
100 CLS
110 PRINT "PRESS A LETTER."
120 A$=INKEY$
130 IF A$="" THEN 120
140 PRINT A$;" - ";
150 IF ASC(A$)>64 AND ASC(A$)<91 THEN PRINT
    "LETTER" ELSE PRINT "NOT A LETTER"
160 PRINT:GOTO 110
170 END
```

*Also see IF, THEN*

**END**

END stops the program (and prints the message OK). It is good programming practice to use END as the last statement in your program even though it is optional.

You can arrange your program so the end is not at the last line. The END statement will prevent the computer from going to any other lines. OK will be printed as the message. If you use a STOP command instead, the message printed will be BREAK IN *lll*, where *lll* is the line number.

**Valid statement**

```
990 END
```

**Sample program**

```
100 CLS
110 FOR I=1 TO 10
120 PRINT I
130 NEXT I
140 PRINT
150 END
```

## EOF

EOF checks for the end of file, or last data item, when reading in items from a cassette or disk file.

- IF EOF(-1) means you have reached the end of data in the file.
- IF EOF(0) means you have not reached the end of file; there is another data item.

### Valid statement

```
150 IF EOF(-1) THEN 990
```

### Sample program

```
100 CLS
110 PRINT "PREPARE CASSETTE FILE"
120 OPEN "I", #-1, "NAMES"
130 IF EOF(-1) THEN 170
140 INPUT #-1, N$
150 PRINT N$
160 GOTO 130
170 CLOSE #-1
180 REM PROGRAM CONTINUES
```

## EXP

### Extended BASIC only

EXP( $n$ ) is the exponential function, which returns the value of  $e^n$  where  $e = 2.718281828$  and  $n$  is a numeric expression which is evaluated.  $n$  must be a numeric expression less than or equal to 88.

EXP is the inverse of the natural logarithm function:  $N = \text{EXP}(\text{LOG}(N))$ .

### Valid statements

```
PRINT EXP(9)
230 PRINT EXP(B*3)
300 A=EXP(C*2)+Y
```

### Sample program

```
100 A=4.32
110 PRINT "A = "; A
120 PRINT "EXP(A) = "; EXP(A)
130 PRINT "EXP(A*2) = "; EXP(A*2)
140 B=A+EXP(2)
150 PRINT "B = "; B
160 END
```

Changing the value of A will show new results.

Also see LOG

## FIX

### Extended BASIC only

FIX(*n*) returns the whole number part of a numeric expression *n* (the number to the left of the decimal). You may also think of the FIX function as a truncating function.

```
FIX(3.546) = 3
FIX(0)     = 0
FIX(-2.64) = -2
```

### Valid statements

```
PRINT FIX(N)
300 X=X+FIX(3.04*Y)
```

### Sample program

```
100 CLS
110 PRINT "ENTER A DECIMAL NUMBER"
120 INPUT N
130 PRINT "FIX(N) = ";FIX(N)
140 PRINT
150 GOTO 110
160 END
```

Also see INT

## FOR

FOR combined with NEXT creates a FOR-NEXT loop. Each FOR statement must have a corresponding NEXT statement, and each NEXT statement must have a preceding FOR statement. When you perform a procedure a certain number of times, you can use a FOR-NEXT loop. The form is

```
FOR i = a TO b
(loop statements)
NEXT i
```

where *i* is an index or counter and *a* is the first number assigned to *i*. The computer executes the statements after the FOR statement down to the NEXT statement with *i*=*a*. *i* is incremented by 1 each time, and the loop is performed repeatedly until *b* is exceeded.

# FOR

---

If you wish to increment by a number other than 1, use STEP, and the increment may be positive, negative, or a fraction:

```
10 FOR I=A TO B STEP C
20 NEXT I
```

The NEXT statement may leave off the index:

```
10 FOR D=1 TO 20
20 NEXT
```

With no statements between the FOR and NEXT statements, the computer just seems to pause while it is actually counting.

Loops may be nested.

```
100 FOR A=1 TO 3
110 FOR B=1 TO 10
120 PRINT B*A
130 NEXT B
140 NEXT A
```

Lines 130 and 140 may be combined as 130 NEXT B,A

## Valid statements

```
150 FOR I=X TO Y
220 FOR A=1 TO 500
300 FOR X=20 TO 10 STEP -1
400 FOR B=2 TO 20 STEP 2
```

## Sample program

```
100 CLS
110 FOR I=1 TO 10
120 PRINT I
130 NEXT I
140 FOR D=1 TO 500:NEXT
150 FOR A=1 TO 3
160 FOR B=1 TO 5
170 PRINT A*B
180 NEXT B,A
190 FOR D=1 TO 500:NEXT D
200 FOR C=10 TO 1 STEP -.5
210 PRINT C
220 NEXT
230 END
```

Note that the C loop which begins in line 200 counts backward from 10 to 1 in steps of  $-.5$ .

Also see NEXT, STEP

## GET

### Extended BASIC only

GET  $(x1,y1)-(x2,y2),t,G$  reads the graphic contents of a rectangle with opposite corners at locations  $(x1,y1)$  and  $(x2,y2)$  into an array  $t$ . Later in the program the PUT statement can place the graphics in a different position. Action graphics can be simulated by using GET and PUT statements.

A DIMENSION statement is required for the size of the rectangle:

```
DIM t(x2-x1,y2-y1)
```

The array  $t$  contains the rectangle's contents. The points in the GET statement are the  $x1,y1$  coordinates of the upper-left corner of the rectangle around the graphics and the  $x2,y2$  coordinates of the lower-right corner. G tells the computer to store the contents with full graphic detail. G is an option. If you use G, you must use the action options with PUT (PSET, PRESET, AND, OR, NOT). The options are used in PMODE 4 or 2 and are optional in PMODE 0, 1, or 3. You must be in the same PMODE for GET as you are for PUT.

### Valid statements

```
250 GET (30,55)-(60,75),A
```

```
300 GET (40,60)-(60,70),B,G,
```

### Sample program

```
100 PMODE 3:PCLS
110 SCREEN 1,0
120 DIM T(24,16)
130 COLOR 3,1
140 LINE(20,80)-(28,92),PSET,BF
150 PAINT(22,82),3,3
160 LINE(28,88)-(44,92),PSET,BF
170 PAINT(30,90),3,3
180 DRAW "C4;BM24,88;R4F2D4G2L4H4U4E2"
190 DRAW "BM40,92;R2FD2GL2HU2E"
200 COLOR 2,1
210 LINE(22,82)-(26,86),PSET,BF
220 GET(20,80)-(44,96),T,G
230 X1=20:X2=44
240 S=10
250 FOR I=1 TO 20
260 X1=X1+S:X2=X2+S
270 PCLS
280 PUT(X1,80)-(X2,96),T,PSET
```

```
290 NEXT I
300 S=-S
310 GOTO 250
320 END
```

Watch for the interesting results when you alter the FOR statement in line 250 to "1 TO 1", or "1 TO 3".

*Also see* DIM, PUT

## GOSUB

GOSUB *l* tells the computer to GO to a SUBroutine starting at line *l* then return when it finds the command RETURN. The next line executed will be the line directly following the GOSUB statement. GOSUB is used when a procedure is required several places in a program. You can have a GOSUB within another subroutine. With some computers it is faster to have subroutines at the beginning of the program with lower line numbers.

### Valid statement

```
300 GOSUB 570
```

### Sample program

```
100 GOTO 170
110 PRINT
120 PRINT "PRESS ENTER TO CONTINUE"
130 A$=INKEY$: IF A$="" THEN 130
140 IF ASC(A$)<>13 THEN 130
150 CLS
160 RETURN
170 CLS
180 PRINT "HAVE INSTRUCTIONS HERE"
190 GOSUB 110
200 PRINT "FIRST PROBLEM INTRODUCED"
210 GOSUB 110
220 PRINT "ANSWER PRESENTED"
230 GOSUB 110
240 END
```

This subroutine could be useful in a program introducing and answering a math problem, for example. You could insert additional lines between lines 180 and 190, between lines 200 and 210, and between lines 220 and 230.

*Also see* RETURN



## GOTO

GOTO *l* transfers program execution to the specified line number *l* rather than going to the very next statement in numerical order.

### Valid statements

```
100 GOTO 500  Transfers down to line 500
700 GOTO 400  Transfers back to line 400
900 GOTO 900  Program stays at line 900 until you press BREAK
```

### Sample program

```
100 CLS
110 GOTO 140
120 PRINT "THIS IS SECOND"
130 GOTO 160
140 PRINT "THIS IS FIRST"
150 GOTO 120
160 PRINT "THIS IS LAST"
170 PRINT
180 PRINT "PRESS <BREAK> TO STOP."
190 GOTO 190
200 END
```

## HEX\$

### Extended BASIC only

HEX\$(*n*) returns a string representing the hexadecimal (base 16) equivalent of the decimal number (base 10) or variable *n*, where *n* may be from 0 to 65535. When you work with machine language programs, you may want to use hex numbers. Hex numbers are preceded by &H and consist of the numerals 0-9 and letters A-F, which represent numerals. Hex constants must be in the range of 0 to FFFF.

### Valid statements

```
PRINT HEX$(125)
250 PRINT HEX$(D)
```

### Sample program

```
100 CLS
110 PRINT "ENTER A DECIMAL NUMBER"
120 INPUT "TO BE CONVERTED"; D
130 PRINT
140 PRINT "THE EQUIVALENT HEXADECIMAL"
150 PRINT "VALUE IS "; HEX$(D)
```

# HEXS

---

```
160 PRINT:PRINT
170 GOTO 110
180 END
```

Use this sample program to convert decimal numbers to hexadecimal equivalents.

## IF

IF starts a conditional branching command. There are several forms:

```
IF test THEN line
IF test THEN action
IF test THEN line1 ELSE line2
IF test THEN line1 ELSE action2
IF test THEN action1 ELSE line2
IF test THEN action1 ELSE action2
```

where *test* is a relational or numeric expression; *line*, *line1*, and *line2* are line numbers; and *action*, *action1*, and *action2* are valid commands. The *actions* may be a series of commands separated by colons.

For the IF-THEN statements, IF the *test* expression is true, THEN the program branches to the line number specified or performs the specified action. If the *test* expression is not true, the program goes immediately to the next statement in numerical order.

For the IF-THEN-ELSE statements, IF the *test* expression is true, THEN the program transfers to the *line1* specified or performs *action1*. If the *test* expression is not true, the ELSE command transfers the program to *line2* or performs *action2*.

The *test* expression may contain the logical words AND, OR, or NOT.

## Valid statements

```
250 IF A THEN 500
300 IF B=1 THEN 400 ELSE B=B+1
```

## Sample program

```
100 CLS
110 A=4:B=8
120 IF B=2*A THEN 150
130 PRINT "B<>2*A"
140 GOTO 160
```

```

150 PRINT "B=2*A"
160 IF B>A THEN PRINT "B>A" ELSE PRINT "B<=A"
"
170 PRINT "TRY AGAIN? (Y/N)"
180 A$=INKEY$
190 IF A$="N" THEN END
200 IF A$<>"Y" THEN 180
210 PRINT
220 INPUT "A = ";A
230 INPUT "B = ";B
240 GOTO 120
250 END

```

Can you create additional lines that will demonstrate the forms of the IF command which have *not* been used in the above sample program? Refer to the list of IF forms.

*Also see ELSE, THEN*

## INKEY\$

INKEY\$ detects if you have pressed a key on the keyboard. This command is useful if you want your user to respond with just a one-key answer, such as yes or no (Y/N), a multiple-choice response, or a number or letter. The INKEY\$ method has less chance of user error than INPUT. INKEY\$ does not wait for a key to be pressed. If no key is pressed when the INKEY\$ is checked, then INKEY\$ has the value of a *null string*.

### Valid statements

```

250 A$=INKEY$
300 R$=INKEY$

```

### Sample program

```

100 CLS
110 PRINT
120 PRINT "CHOOSE: ":PRINT
130 PRINT "1 FIRST OPTION"
140 PRINT "2 SECOND OPTION"
150 PRINT "3 THIRD OPTION"
160 PRINT "4 END PROGRAM":PRINT
170 C$=INKEY$:IF C$="" THEN 170
180 C=ASC(C$)
190 IF C<49 OR C>52 THEN 170
200 PRINT
210 ON C-48 GOTO 220,240,260,280
220 PRINT "## FIRST OPTION CHOSEN"

```

## INKEY\$

---

```
230 GOTO 110
240 PRINT "*** SECOND OPTION CHOSEN"
250 GOTO 110
260 PRINT "*** THIRD OPTION CHOSEN"
270 GOTO 110
280 CLS:END
```

Notice how many other programs in this reference guide use the INKEY\$ command.

## INPUT

INPUT allows the user to enter something as the program is being run. The computer will print a question mark, then receive user input until the ENTER key is pressed. Valid input can be up to 125 characters and must not contain a colon or a comma. If the variable name for the input value is numeric and the user enters nonnumeric characters, the computer will print REDO and wait for more input.

### Valid statements

100 INPUT A	Receives number from user.
200 INPUT B\$	Receives string from user.
300 INPUT "NAME:";N\$	Prints prompt in quotes then receives string.

### Sample program

```
100 CLS
110 PRINT "WHAT IS YOUR NAME?"
120 INPUT N$
130 PRINT
140 PRINT "HELLO, ";N$
150 PRINT
160 INPUT "ENTER A NUMBER";N
170 PRINT "NUMBER TIMES 2 =";N*2
180 PRINT
190 END
```

Also see LINE INPUT

## INPUT #

INPUT # accepts data from external devices. INPUT #-1 reads data from a previously prepared cassette file. The cassette is device #-1. The INPUT #-1 statement must be preceded by a

statement to OPEN device #-1 for input.

INPUT #1 - INPUT #15 read data from previously prepared disk files. The INPUT # statement must be preceded by a matching OPEN # statement.

### Valid statements

```
200 OPEN "I", #-1, "REGIONS"
210 INPUT #-1, R$
220 OPEN "I", #2, "REGIONS/DAT"
230 INPUT #2, R$
```

### Sample program

```
100 CLS
110 PRINT "PREPARE CASSETTE"
120 PRINT "THEN PRESS <ENTER>."
130 A$=INKEY$: IF A$="" THEN 130
140 IF ASC(A$)<>13 THEN 130
150 PRINT
160 OPEN "I", #-1, "NAMES"
170 IF EOF(-1) THEN 210
180 INPUT #-1, N$
190 PRINT N$
200 GOTO 170
210 CLOSE #-1
220 END
```

Also see CLOSE, OPEN

## INSTR

### Extended BASIC only

INSTR(*s1*,*s2*) is a function used with strings. It returns the position of the first occurrence of string *s2* in string *s1*. The value returned is a number that tells at which character position *s2* starts. If *s2* is not found in *s1*, the value returned is zero.

### Valid statements

```
140 P=INSTR(A$,B$)
200 PRINT INSTR(LIST$,N$)
```

### Sample program

D\$ is a string listing the abbreviations for the months. You enter the name of a month, M\$. Line 140 changes M\$ to only the first three letters of your input value. Line 150 searches D\$ for the first occurrence of the month name and will return the month number.

# INSTR

---

If you enter only two letters, MA, the month could be MAR or MAY, but the first occurrence is MAR for the third month.

```
100 CLS
110 D$="JANFEBMARAPR MAYJUNJULAUGSEPOCTNOVDEC
"
120 PRINT
130 INPUT "ENTER A MONTH ";M$
140 M$=LEFT$(M$,3)
150 M=(INSTR(D$,M$)-1)/3+1
160 IF M>=1 THEN 190
170 PRINT "NOT IN LIST"
180 GOTO 120
190 PRINT "MONTH NUMBER =";M
200 GOTO 120
210 END
```

## INT

$\text{INT}(n)$  returns the integer value of the numeric expression  $n$ . An integer function considers the integer to be the greatest whole number less than the given value. If you think of it as a number line, the integer value is the first whole number to the left of the number  $n$ . If the number is positive, the value returned is the whole number with the decimal portion truncated. If the number is negative, the value returned is the whole number less than the given  $n$ .

```
INT(5.479) = 5
INT(0)     = 0
INT(-3.6)  = -4
```

### Valid statements

```
PRINT INT(3.14159)
200 A=INT(A)
300 SC=SC+INT(2.5*B)
```

### Sample program

```
100 CLS
110 PRINT "ENTER A DECIMAL NUMBER."
120 INPUT N
130 PRINT "FIX(N)=";FIX(N),"INT(N)=";INT(N)
140 PRINT
150 GOTO 110
160 END
```

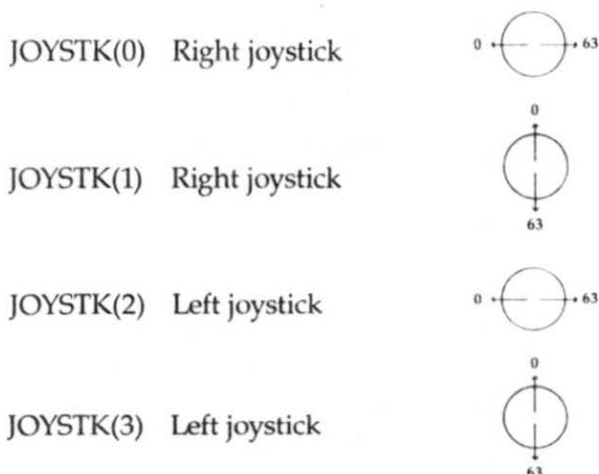
After typing in this sample program, enter negative fractions to

see the difference between the FIX and INT commands.

Also see FIX

## JOYSTK

JOYSTK(*n*) returns the horizontal or vertical position of the joystick and is a value from 0 to 63. *n* defines the joystick used and can be 0, 1, 2, or 3 as follows:



Note: Anytime you use JOYSTK(1), JOYSTK(2), or JOYSTK(3), you must first read JOYSTK(0).

To detect when the red button of any joystick is pressed, PEEK(65280). Ordinarily the value of  $P = \text{PEEK}(65280)$  is 255 or 127. If the right joystick button is pressed, then  $P$  will be 126 or 254. If the left joystick button is pressed, then  $P$  is either 125 or 253.

### Valid statements

```
100 M = JOYSTK(0)
200 IF JOYSTK(1) > 31 THEN 1000
300 X = 30 + JOYSTK(3)
190 P = PEEK(65280)
200 IF P = 126 THEN 590
210 IF P = 254 THEN 590
```

### Sample program

This program illustrates the use of the right joystick only.

# JOYSTK

---

```
100 CLS
110 PRINT "USE THE RIGHT JOYSTICK."
120 X=240:C=159
130 PRINT @X,CHR$(C);
140 P=PEEK(65280)
150 IF P<>126 AND P<>254 THEN 170
160 C=C+16:IF C>255 THEN C=159
170 IF JOYSTK(0)<15 THEN X=X-1
180 IF JOYSTK(0)>45 THEN X=X+1
190 IF JOYSTK(1)>15 THEN 220
200 X=X-32
210 IF X<32 THEN X=X+32
220 IF JOYSTK(1)<45 THEN 250
230 X=X+32
240 IF X>479 THEN X=X-32
250 IF X<32 THEN X=32
260 IF X>479 THEN X=479
270 GOTO 130
280 END
```

Also see PEEK

## LEFT\$

LEFT\$(s,n) is a string function that returns the left *n* characters of the string *s*, or the first *n* characters. *n* must be a number zero or greater. If the number *n* includes a decimal fraction, only the integer portion of *n* is considered. If the number *n* is greater than the length of the string *s*, only the string *s* is returned (no added blanks). An example is LEFT\$("HI THERE",2) returns HI.

### Valid statements

```
PRINT LEFT$(A$,5)
300 L$=LEFT$(P$,N)
310 N$=LEFT$(N$,X)+" SMITH"
```

### Sample program

```
100 CLS
110 PRINT "ENTER A WORD."
120 INPUT W$
130 PRINT "LIST HOW MANY LETTERS?"
140 INPUT N
150 IF N>=0 THEN 170
160 PRINT "MUST BE GREATER THAN ZERO.":GOTO
130
170 L$=LEFT$(W$,N)
180 PRINT L$
```



```
190 PRINT
200 GOTO 110
```

Notice that if you respond with a number less than the length of the word, only *n* number of letters will print.

*Also see* MID\$, RIGHT\$

## **LEN**

LEN(*s*) returns the length of string *s*, or the number of characters in *s*. For example, the LENGTH of a string "HELLO" is 5.

A string may be up to 255 characters long (you will need to CLEAR memory if you work with long strings). An LS error indicates that the string is longer than 255 characters.

### **Valid statements**

```
PRINT LEN(B$)
230 C=LEN(C$)
300 FOR L=1 TO LEN(W$)
PRINT LEN("THIS IS A TEST")
```

### **Sample program**

```
100 CLS
110 PRINT "ENTER A WORD OR PHRASE."
120 INPUT S$
130 PRINT "THE LENGTH OF YOUR INPUT IS"
140 PRINT LEN(S$); "CHARACTERS."
150 PRINT
160 GOTO 110
170 END
```

This command could be useful, for example, to check that the correct number of characters was entered in a string, or to return a count of characters entered.

## **LET**

LET is used to assign values in a program. The word LET is optional with this computer and can be omitted. LET A=6 assigns the value of 6 to the variable name A. Another way to write the command is A=6.

### **Valid statements**

```
100 LET B=54
```

```
110 LET N$="BOB"  
200 LET A=A+8
```

### Sample program

```
100 CLS  
110 LET A=10  
120 LET B=25  
130 PRINT "B IS";B  
140 PRINT "B/A =" ;B/A  
150 PRINT "A+B =" ;A+B  
160 LET B=30  
170 PRINT "B NOW IS";B  
180 END
```

## LINE

### Extended BASIC only

LINE( $x_1,y_1$ )-( $x_2,y_2$ ),PSET draws a line from the point specified by location ( $x_1,y_1$ ) to the point specified by ( $x_2,y_2$ ) where  $x_1$  and  $x_2$  are the horizontal distances from 0 to 255, and  $y_1$  and  $y_2$  are the vertical distances from 0 to 191. PSET turns the line on (Point SET) to the foreground color or the prespecified color.

LINE( $x_1,y_1$ )-( $x_2,y_2$ ),PRESET erases the line or Point RESETs the line to the background color.

You may specify only the second point if you wish, and the line will be drawn from (128, 96) or the latest previous position. Be sure to include the hyphen: LINE -( $x_2,y_2$ ),PSET.

Boxing in the line is optional:

LINE( $x_1,y_1$ )-( $x_2,y_2$ ),PSET,B

This command will outline a box or rectangle with one corner at ( $x_1,y_1$ ) and the opposite corner at ( $x_2,y_2$ ).

You may color the box with the foreground color by using Box-Filled:

LINE( $x_1,y_1$ )-( $x_2,y_2$ ),PSET,BF

The lines drawn will be in the foreground color. If you wish to draw in a different color, first use the statement COLOR  $f,b$  where  $f$  and  $b$  are the foreground and background colors.

### Valid statements

```
350 LINE(0,0)-(255,191),PSET  
360 LINE(0,0)-(128,96),PRESET  
400 LINE -(90,125),PSET
```

```
500 LINE(10,10)-(20,50),PSET,B
550 LINE(100,40)-(120,80),PSET,BF
```

### Sample program

```
100 PMODE 3,1
110 SCREEN 1,0
120 PCLS
130 LINE(0,0)-(255,191),PSET
140 LINE(100,10)-(130,30),PSET,B
150 LINE(145,10)-(170,20),PSET,BF
160 COLOR 3,1
170 LINE(20,100)-(50,85),PSET
180 LINE(70,100),PSET
190 LINE(90,85),PSET,B
200 LINE(100,100)-(120,120),PSET,BF
210 GOTO 210
220 END
```

Try typing in this line and rerunning the program to see the effect of the PRESET command.

```
135 LINE(0,0)-(255,191),PRESET
```

*Also see* PRESET, PSET

## LINE INPUT

### Extended BASIC only

LINE INPUT allows any ASCII character (except ENTER) that is on the keyboard to be input to a string variable. You may include a prompt in quotes.

On a regular INPUT statement, if you enter a colon or a comma, you will get the message EXTRA IGNORED, and the string you entered will contain only the characters up to the colon or comma. LINE INPUT accepts the colon or comma as part of the input string.

Leading blanks on INPUT are ignored, but on LINE INPUT they are considered part of the string.

INPUT prints a question mark, then blinks the cursor, but LINE INPUT just blinks the cursor.

INPUT may list more than one variable, and the user separates items with commas. LINE INPUT allows only one variable to be input because the commas are part of the string.

### Valid statements

```
100 LINE INPUT A$
120 LINE INPUT "ENTER NAME";P$
```

## LINE INPUT

---

Note the slight difference between lines 190 and 210. What will be the result when you run this sample program?

### Sample program

```
100 CLS
110 PRINT "WHAT IS YOUR NAME?"
120 LINE INPUT N$
130 PRINT "HELLO, ";N$
140 PRINT "ENTER ANY CHARACTERS."
150 LINE INPUT "YOUR PHRASE: ";P$
160 PRINT
170 PRINT "LENGTH IS ";LEN(P$)
180 PRINT:PRINT "NOW TRY THESE:"
190 INPUT "ENTER A NAME ";N1$
200 PRINT "NAME IS ";N1$
210 LINE INPUT "ENTER A NAME ";N2$
220 PRINT "NAME IS ";N2$
230 END
```

Also see INPUT

## LIST

LIST lists your program on the screen with statements in numerical order. There are several forms of the command:

LIST	Lists complete program.
LIST <i>l</i>	Lists line number <i>l</i> only.
LIST - <i>l</i>	Lists all lines up to and including line number <i>l</i> .
LIST <i>l1</i> - <i>l2</i>	Lists program from line <i>l1</i> to <i>l2</i> inclusive.
LIST <i>l</i> -	Lists all lines from line <i>l</i> to the end of the program.

While you are LISTing the program, if you wish to stop the display, press SHIFT and @ together. Press any key to continue.

### Valid commands

```
LIST
LIST 300
LIST -150
LIST 400-470
LIST 900-
```

Also see LLIST

## LLIST

LLIST will print the listing of a program or portions of the program on a printer. The forms of the command are similar to LIST:

LLIST	Lists the complete program on printer.
LLIST <i>l</i>	Prints line number <i>l</i> on printer.
LLIST <i>-l</i>	Prints program listing up to line <i>l</i> on printer.
LLIST <i>l1-l2</i>	Prints all lines from line <i>l1</i> to line <i>l2</i> inclusive on printer.
LLIST <i>l-</i>	Prints program listing from line <i>l</i> to end of program on printer.

### Valid commands

LLIST  
 LLIST 750  
 LLIST -300  
 LLIST 400-700  
 LLIST 800-

*Also see* LIST

## LOAD

### Disk BASIC only

LOAD "TITLE" loads (reads in) a specified program from disk to the computer. The equivalent command when loading programs from cassette is CLOAD.

Unlike CLOAD, the program name is not optional when LOADING from disk. If a program with the specified title is found on the disk, the computer loads the program and prints OK on the screen. If the file is not found, an error message is printed. If no *extension* is specified with the program name, then the extension /BAS is assumed.

If an R is added to the LOAD statement, the program will begin to run as soon as it finishes loading.

### Valid commands

LOAD "GAME"  
 LOAD "HOMEWORK/SCH"  
 LOAD "DEMON/PRG",R

*Also see* CLOAD, CSAVE, SAVE

## LOG

### Extended BASIC only

LOG(*n*) returns the natural logarithm of a number *n*, where *n* must be greater than zero and less than about  $10^{38}$ . The natural logarithm is the exponent to which the base *e* (2.718281828) must

be raised for the resultant value of  $n$ .

LOG is the inverse of EXP, or  $N = \text{LOG}(\text{EXP}(N))$ .

To find the logarithm of a number  $N$  with another base  $B$ , specified as  $X$ , use the formula  $X = \text{LOG}(N)/\text{LOG}(B)$ .

## Valid statements

```
PRINT LOG(10)
230 A=3*LOG(X)
280 IF LOG(N)<0 THEN 700
```

## Sample program

```
100 CLS
110 N=2
120 PRINT "N =";N;" {3 SPACES}LOG(N) =";LOG(N)
130 N=N+1
140 GOTO 120
150 END
```

Use this sample program to arrive at the natural logarithm of any number  $N$ .

*Also see EXP*

## MEM

Enter PRINT MEM to find out how much free memory (RAM, Random Access Memory) is available in the computer. The amount of memory you have depends on the length of your program and any statements or commands that have reserved different amounts of memory (PCLEAR, CLEAR, DIM). The more memory your program has used up, the less free memory you have left.

## Valid statements

```
PRINT MEM
150 PRINT MEM
```

## Sample program

```
100 CLEAR 200
110 PRINT MEM
120 CLEAR 1000
130 PRINT MEM
140 DIM A(20)
150 PRINT MEM
160 DIM B(40)
```

```

170 PRINT MEM
180 CLEAR 200
190 PRINT MEM
200 END

```

*Also see CLEAR*

## MID\$

MID\$(*s, n, m*) is a string function that returns a segment of string *s* starting with character number *n* and *m* letters long. For example, MID\$("THIS IS AN EXAMPLE",6,5) would return IS AN.

### Valid statements

```

PRINT MID$(S$,4,8)
300 N$=N$+MID$(A$,X,5)

```

### Sample program

```

100 A$="THIS IS A LONG STRING."
110 PRINT A$
120 PRINT MID$(A$,1,4)
130 PRINT MID$(A$,7,12)
140 PRINT MID$(A$,LEN(A$)-4,5)
150 END

```

*Also see LEFT\$, RIGHT\$*

## MOTOR

MOTOR ON turns the cassette recorder on through the remote switch if you have the PLAY button pressed. AUDIO ON will then connect the recorder's sound to your monitor or television speaker. AUDIO OFF will turn the sound off, and MOTOR OFF will turn the cassette recorder off. Pushing the RESET button will also turn the recorder off.

### Valid statements

```

200 MOTOR ON
500 MOTOR OFF

```

### Sample program

Prepare a cassette with your voice, music, or sound effects that go with your program. Then type in and run this program.

```

100 CLS
110 PRINT "PREPARE CASSETTE."
120 PRINT "PRESS 'PLAY' ON RECORDER"

```

# MOTOR

---

```
130 PRINT "THEN PRESS <ENTER>."
140 A%=INKEY$:IF A%="" THEN 140
150 IF ASC(A%)<>13 THEN 140
160 MOTOR ON
170 AUDIO ON
180 PRINT
190 PRINT "YOU SHOULD HEAR THE RECORDING."
200 PRINT
210 PRINT "PRESS <ENTER> TO STOP."
220 A%=INKEY$:IF A%=""THEN 220
230 IF ASC(A%)<>13 THEN 220
240 AUDIO OFF
250 MOTOR OFF
260 END
```

*Also see* AUDIO

## NEW

NEW erases the BASIC program currently in memory and allows you to start or load a new program. There will be no more numbered lines stored in the computer. All numeric variables return to zero. Any values POKEd into memory, however, will still be there.

### Valid statements

```
NEW
900 NEW
```

## NEXT

NEXT is the last statement in a FOR-NEXT loop. NEXT increments the loop counter or index. If the index is greater than the limit in the FOR statement, program control goes to the statement following NEXT; otherwise, the loop is performed again. The index variable on the NEXT statement is optional unless loops are nested.

### Valid statements

```
200 NEXT
250 NEXT I
300 NEXT J,I
```

### Sample program

```
100 CLS
110 FOR I=1 TO 5
120 PRINT I
```



```

130 NEXT I
140 PRINT
150 FOR J=1 TO 200:NEXT
160 FOR I=1 TO 5
170 FOR J=1 TO 3
180 FOR K=10 TO 20 STEP 2
190 PRINT I*J*K
200 FOR L=1 TO 100:NEXT
210 NEXT K, J, I
220 END

```

Also see FOR, STEP

## NOT

NOT is one of the logical operators in Boolean algebra.

NOT 0 = -1

NOT -1 = 0

NOT (any number) = -number - 1

NOT reverses the state of a test in an IF statement.

### Valid statements

```

PRINT NOT 0
230 IF NOT A THEN 350
500 PRINT NOT X

```

### Sample program

```

100 A=1
110 PRINT A, NOT A
120 A=0
130 PRINT A, NOT A
140 A=-1
150 PRINT A, NOT A
160 A=-56
170 PRINT A, NOT A
180 A=34
190 PRINT A, NOT A
200 END

```

Note the results, especially from lines 150 and 170. Were they expected?

Also see AND, OR

## ON...GOSUB

ON *n* GOSUB *l1, l2, l3* ... tells the computer to evaluate the numeric expression *n*, then branch to a subroutine, depending

on the value of  $n$ . If  $n$  is 1, the program will go to the subroutine starting at line 11. If  $n$  is 2, GOSUB 12; if  $n$  is 3, GOSUB 13; etc. Program control will then return to the line following the ON ... GOSUB statement.

If  $n$  does not have a corresponding line number listed, the program goes to the next line.

## Valid statements

```
250 ON CH GOSUB 100,300,500,700,800
390 ON X-3 GOSUB 450,790,790,450,930,670
```

## Sample program

```
100 CLS
110 PRINT
120 PRINT "CHOOSE:"
130 PRINT "1 GAME ONE"
140 PRINT "2 GAME TWO"
150 PRINT "3 GAME THREE"
160 PRINT "4 END PROGRAM"
170 PRINT
180 C$=INKEY$: IF C$="" THEN 180
190 C=ASC(C$)
200 IF C<49 OR C>52 THEN 180
210 ON C-48 GOSUB 1000,2000,3000,4000
220 FOR D=1 TO 1000:NEXT
230 GOTO 100
1000 PRINT "YOU CHOSE GAME ONE"
1010 REM GAME ONE WOULD BE HERE
1020 RETURN
2000 PRINT "YOU CHOSE GAME TWO"
2010 REM GAME TWO WOULD BE HERE
2020 RETURN
3000 PRINT "YOU CHOSE GAME THREE"
3010 REM GAME THREE WOULD BE HERE"
3020 RETURN
4000 PRINT "END PROGRAM"
4010 PRINT
4020 END
```

The ON ... GOSUB command can also be used to show the ability of truth expressions for multiple branching to subroutines. In the following sample program, random numbers are generated in line 110 and then compared by the formula in line 120. If the numbers are equal, the ON ... GOSUB command branches to line 140; if  $A > B$ , to line 150; and if  $A < B$ , to line 160. Run the program several times to see the results change.

**Sample program**

```

100 CLS
110 A=RND(6):B=RND(6):PRINT A,B
120 X=-(A=B)-2*(A>B)-3*(A<B):PRINT "X=";X:ON
    X GOSUB 140,150,160
130 END
140 PRINT "A=B":RETURN
150 PRINT "A>B":RETURN
160 PRINT "A<B":RETURN

```

*Also see GOSUB*

**ON...GOTO**

ON *n* GOTO *l1,l2,l3* ... evaluates the numeric expression *n*, then branches according to the value of *n*. IF *n* is 1 the program goes to line *l1*; if *n* is 2 the program goes to line *l2*; if *n* is 3 the program goes to line *l3*, etc.

**Valid statements**

```

300 ON N GOTO 400,500,100
790 ON X-48 GOTO 1000,1050,1070,1090

```

**Sample program**

```

100 CLS
110 PRINT "CHOOSE A NUMBER"
120 PRINT "1 2 3 4 5"
130 PRINT
140 A%=INKEY%:IF A%="" THEN 140
150 A=ASC(A%)
160 IF A<49 THEN B=6:GOTO 180
170 IF A>53 THEN B=6 ELSE B=A-48
180 ON B GOTO 230,250,270,290,320,340
190 REM UNLESS YOU TRANSFER TO
200 REM THESE LINES, THE PROGRAM
210 REM WILL NOT GET HERE.
220 STOP
230 PRINT "YOU CHOSE ONE"
240 GOTO 350
250 PRINT "YOU CHOSE TWO"
260 GOTO 350
270 PRINT "YOU CHOSE THREE"
280 GOTO 350
290 PRINT "YOU CHOSE FOUR"
300 GOTO 350
310 GOTO 350
320 PRINT "YOU CHOSE FIVE"

```

# ON...GOTO

---

```
330 GOTO 350
340 PRINT "NOT ONE OF THESE NUMBERS"
350 FOR D=1 TO 1000:NEXT D
360 GOTO 100
370 END
```

As you can see, the ON ... GOTO command is similar to the ON ... GOSUB command.

Also see GOTO

## OPEN

OPEN "O",#-1,"title" opens device #-1, the cassette recorder, for output so you can save files with the PRINT #-1 statement. The name of the file, *title*, is included in quotes and can be up to eight characters.

OPEN "I",#-1,"title" opens device #-1, the cassette recorder, to input or load items back from tape. The file of information is named *title*. Use INPUT #-1,s where s is a variable name to read items from tape.

OPEN "O",#n,"title", where n is a number between 1-15 inclusive, opens a file with the name *title* to the disk drive. Data can then be output to the disk using the PRINT #n statement.

OPEN "I",#n,"title" opens file n, where n is between 1-15, for input from disk. INPUT #n can be used to retrieve data from OPENed disk files.

Use EOF before the INPUT statement to detect the end of file.

The devices are:

(default value)	#0	screen (keyboard)
	#-1	cassette recorder
	#-2	printer
	#1 - #15	disk drive

You do not need to OPEN devices #0 and #-2.

### Valid statements

```
150 OPEN "O",#-1,"DATA"
250 OPEN "I",#-1,"PEOPLE"
350 OPEN "O",#2, "DSKFILE/BAS"
450 OPEN "I",#4, "ADDRESS/DAT"
```

### Sample program

```
100 CLS
110 PRINT "GET NEW CASSETTE READY."
```

```

120 PRINT "PREPARE RECORDER TO 'RECORD',"
130 PRINT "THEN PRESS <ENTER>."
140 A$=INKEY$: IF A$="" THEN 140
150 IF ASC(A$)<>13 THEN 140
160 OPEN "O", #-1, "TEST"
170 PRINT:PRINT "SAVING DATA":PRINT
180 FOR I=1 TO 5
190 READ D$
200 PRINT #-1, D$
210 NEXT I
220 CLOSE #-1
230 PRINT "NOW REWIND TAPE."
240 PRINT:PRINT "PRESS 'PLAY' ON RECORDER"
250 PRINT "THEN PRESS <ENTER>."
260 A$=INKEY$: IF A$="" THEN 260
270 IF ASC(A$)<>13 THEN 260
280 OPEN "I", #-1, "TEST"
290 PRINT:PRINT "READING DATA"
300 IF EOF(-1) THEN 340
310 INPUT #-1, D$
320 PRINT D$
330 GOTO 300
340 CLOSE #-1
350 DATA HELLO, HI, THIS IS A TEST
360 DATA SAMPLE OF OPEN, USING CASSETTE
370 END

```

*Also see* CLOSE, INPUT #, PRINT #

## OR

Logical OR is used in IF statements to combine conditions. One condition or the other or both must be true for the THEN action to take place. More than one OR may be used.

### Valid statements

```

300 IF X=3 OR X=6 THEN 700
400 IF ANS$="YES" OR SCORE=100 THEN 800

```

### Sample program

```

100 CLS
110 INPUT "ENTER A NUMBER "; N1
120 INPUT "ENTER ANOTHER NUMBER "; N2
130 IF N1<N2 OR N1<N2*2 THEN 160
140 PRINT "N1>=N2 OR N1>=N2*2"
150 GOTO 170
160 PRINT "N1<N2 OR N1<N2*2"
170 PRINT

```

# OR

---

```
180 PRINT "TRY AGAIN? (Y/N)"
190 INPUT A$
200 IF A$="Y" OR A$="YES" OR A$="YEP" THEN 1
    00
210 IF A$="N" OR A$="NO" OR A$="NOPE" THEN 2
    40
220 PRINT "SORRY, DON'T UNDERSTAND."
230 GOTO 170
240 END
```

*Also see AND, IF*

## PAINT

### Extended BASIC only

PAINT ( $x,y$ ), $c$ , $b$  paints color  $c$  from the point ( $x,y$ ) to the border color  $b$ .

$x$  may be 0 to 255     $x$  coordinate of starting location  
 $y$  may be 0 to 191     $y$  coordinate of starting location  
 $c$  may be 0 to 8        color of paint  
 $b$  may be 0 to 8        border color to stop painting

You may paint only with colors available in the PMODE and color set specified in the SCREEN statement. All lines or drawings in colors other than  $b$  will be ignored — in other words, painted over. When painting objects, make sure that the  $x,y$  position is within the desired area and that there are no "leaks," or paint may flow into other areas. You also need to pay attention to the order in which you draw things. For example, if you first draw a blue line, then draw a red circle over the line and PAINT within the circle until it reaches red, the PAINT color will be contained in the circle. The blue will be ignored. However, if you draw the red circle first, then the blue line on top of it, the paint will leak out the points where the blue cut an opening in the red circle.

### Valid statements

```
300 PAINT(X,Y),C,B
400 PAINT(20,50),7,8
```

### Sample program

```
100 PMODE 3,1
110 SCREEN 1,0
120 PCLS
130 CIRCLE (100,100),20
140 PAINT (100,100),3,4
```

```

150 LINE (200,20)-(200,80),PSET
160 LINE (150,30)-(250,60),PSET,B
170 PAINT (152,32),2,4
180 COLOR 3,1
190 LINE (20,100)-(50,130),PSET,B
200 CIRCLE(35,115),10
210 PAINT(22,102),4,3
220 COLOR 2,1
230 LINE(150,180)-(255,120),PSET
240 COLOR 3,1
250 LINE(150,120)-(255,180),PSET
260 CIRCLE(200,150),30
270 PAINT(200,140),4,3
280 GOTO 280
290 END

```

## PCLEAR

### Extended BASIC only

PCLEAR *n* will reserve *n* pages of video Random Access Memory for graphics. *n* may be a number from 1 to 8. Each page contains 1536 memory locations. If you do not specify PCLEAR, the default value of four pages (6K) will be reserved. Greater detail and color in your program will require more memory. A guideline is:

Graphics mode desired	Reserve	Where <i>n</i> may be
PMODE 4, <i>n</i>	PCLEAR 4+( <i>n</i> -1)	1 to 5
PMODE 3, <i>n</i>	PCLEAR 4+( <i>n</i> -1)	1 to 5
PMODE 2, <i>n</i>	PCLEAR 2+( <i>n</i> -1)	1 to 7
PMODE 1, <i>n</i>	PCLEAR 2+( <i>n</i> -1)	1 to 7
PMODE 0, <i>n</i>	PCLEAR 1+( <i>n</i> -1)	1 to 8

Ordinarily, if you turn on the computer and PRINT MEM, you have 8487 bytes free on the 16K Extended BASIC Color Computer. If you do not want to use a lot of graphics, PCLEAR 1 to clear one page of memory. PRINT MEM and you will see there are now 13,095 bytes, so you can write a longer program which requires more memory.

You cannot use the command PCLEAR 0, but one way to clear zero pages and gain even more memory is to type POKE 25,6:NEW and press ENTER. On the 16K Color Computer you will have 14,631 bytes free.

# PCLEAR

---

## Valid statements

```
100 PCLEAR 8
100 PCLEAR 6
```

## Sample programs

```
100 CLS
110 PRINT MEM
120 PCLEAR 8
130 PRINT MEM
140 PCLEAR 1
150 PRINT MEM
160 PCLEAR 2
170 PRINT MEM
180 PCLEAR 3
190 PRINT MEM
200 PCLEAR 4
210 PRINT MEM
220 END
```

```
100 PCLEAR 8
110 FOR P=1 TO 8
120 PMODE 0,P
130 PCLS
140 LINE (P*5,P*5)-(P*20,P*20),PSET,BF
150 NEXT P
160 L1=1:L2=8:S=1
170 FOR P=L1 TO L2 STEP S
180 PMODE 0,P
190 SCREEN 1,0
200 FOR D=1 TO 20:NEXT D
210 NEXT P
220 S=-S:LL=L1:L1=L2:L2=LL
230 GOTO 170
```

*Also see* PMODE

## PCLS

### Extended BASIC only

PCLS clears the graphics screen. PCLS *c* clears the screen and sets the screen color *c* (or clears the screen with color *c*), where *c* is the color number from 0 to 8. If *c* is omitted, the current background color is used.

## Valid statements

```
110 PCLS
350 PCLS 4
```



**Sample program**

```

100 PMODE 3,1
110 SCREEN 1,0
120 GOSUB 260
130 PCLS
140 CIRCLE(100,100),35
150 GOSUB 260
160 PCLS 2
170 LINE(20,20)-(100,100),PSET,BF
180 GOSUB 260
190 PCLS 3
200 GOSUB 260
210 PCLS 4
220 GOSUB 260
230 DRAW "C2;BM 50,50;U10E20F20D10G20H20"
240 GOSUB 260
250 END
260 FOR D=1 TO 500:NEXT
270 RETURN
280 END

```

**PCOPY****Extended BASIC only**

PCOPY *s* TO *d* copies the graphics contents of one memory page *s* to another page *d* (source TO destination). You must have previously reserved enough pages. This command is useful if you want to transfer a detailed graphics drawing to another page.

You must be careful about the order in which your program statements are executed. If you go to a different page with PMODE, then PCLS, then PCOPY, the results are different from those when you do not use PCLS.

In the following example, the graphics only in the top fourth of the screen are PCOPYed to the other pages. The following pages print the graphics lower each time. Experiment with changing the order of the lines containing the PCOPY commands to see what happens. Also try drawing in a different location, using PCOPY to copy the contents to another page.

**Valid statements**

```

450 PCOPY 4 TO 3
700 PCOPY 1 TO 3

```

**Sample program**

```

100 PCLEAR 8
110 PMODE 3,1

```

## PCOPY

---

```
120 PCLS
130 SCREEN 1,0
140 CIRCLE (126,22),20
150 LINE (70,0)-(180,45),PSET,B
160 PAINT (72,2),3,4
170 DRAW "BM120,25;E8R5F8D5G8L5H8U5"
180 PAINT (126,25),4,4
190 GOSUB 260
200 PCOPY 1 TO 2
210 GOSUB 260
220 PCOPY 1 TO 3
230 GOSUB 260
240 PCOPY 1 TO 4
250 GOTO 250
260 FOR D=1 TO 1000:NEXT
270 RETURN
280 END
```

## PEEK

PEEK(*n*) allows you to examine the contents of memory location *n*. An application of PEEK in a BASIC program is programming with joysticks. PEEK(65280) will tell you if the red button of the joystick has been pressed. If PEEK(65280) returns 255 or 127, the button has not been pressed.

### Valid statements

```
100 P=PEEK(65314)
320 S=PEEK(D+B)
500 IF PEEK(65280)=255 OR PEEK(65280)=127 THEN 300
```

### Sample use

In command mode (not a program), enter the following commands:

```
PRINT PEEK(149)
PRINT PEEK(150)
```

The values returned should be 0 and 87. Now enter:

```
POKE 149,1
POKE 150,255
```

The first two values are for printing at the rate of 600 baud. To change for a printer requiring a data transfer rate of only 110 baud, the values 1 and 255 should be POKEd into memory. You may see the results by:

PRINT PEEK(149),PEEK(150)

To return to 600 baud,

POKE 149,0

POKE 150,87

*Also see* JOYSTK, POKE

## PLAY

### Extended BASIC only

PLAY *s* plays music on your computer with the description contained in string *s*. *s* specifies the note or notes to be played. The tone is specified either by note name (ABCDEFGG or A# for A-sharp and B- for B-flat) or by numbers from 1 to 12. The numbers can be prefaced by the letter N for Note. If you do not use N, the numbers need to be separated by semicolons. Semicolons can separate all the options, but are not required. Several options are available:

- O*o* **Octave.** *o* is a number from 1 to 5; 1 is the lowest octave and 5 is the highest. The octaves start with note C. Default value is O2 or the previously set octave.
- L*l* **Length.** *l* is a number from 1 to 255. The number *l* represents  $1/l$  of the whole note. L1 is a whole note; L8 represents an eighth note; L200 represents a 1/200th note. Default value is L4 or the previously set length.  
**Dotted note.** The dot after the length represents a dotted note (the value of the note plus half the value of the note). For example, L8. represents a dotted eighth note. A dotted eighth note is held as long as an eighth note plus a sixteenth note.
- T*t* **Tempo.** *t* is a number from 1 to 255. T1 is the slowest, and T255 is the fastest. Default value is T2.
- V*v* **Volume.** *v* is a number from 1 to 31. V1 is the softest and V31 is the loudest. The default value is V15.
- P*p* **Pause.** *p* is a number from 1 to 255. The pause length corresponds to the *l* length of the note. P4 would pause the length of a quarter note. P does not allow the dotted function; you could just combine pause lengths, such as P4P8.
- X **Execute a substring.** The substring must be followed by a semicolon, even if it is the last item in the list. If

# PLAY

---

a substring A\$ has been defined as "ABC", the command would be "XA\$;" to play the music.

- + **Use with O, V, T, or L.** Increments the present value by 1 each time the line is played. For example, O+ will play octave 3 the first time, octave 4 the second time, and octave 5 the third time.
- **Use with O, V, T, or L.** Decreases the present value by 1 each time the line is played.
- > **Use with O, V, T, or L.** Multiplies the current value by 2.
- < **Use with O, V, T, or L.** Divides the current value by 2.

## Valid statements

```
120 PLAY M$
150 PLAY "ABO3C;C#;D"
200 PLAY "O3;XB$;O4;XB$;"
```

Note the C# sign used in the second example statement. It refers to the note C-sharp.

## Sample program

```
100 GOSUB 470
110 PRINT "CDEFGAB"
120 PLAY "CDEFGAB"
130 GOSUB 470
140 PRINT "5;3;1;3;5"
150 PLAY "5;3;1;3;5"
160 GOSUB 470
170 PRINT "O1;C;O2;C;O3;C;O4;C;O5;C"
180 PLAY "O1;C;O2;C;O3;C;O4;C;O5;C"
190 GOSUB 470
200 S$="L2;O2;C;L8;E6;L1;O3;C"
210 PRINT S$
220 PLAY S$
230 GOSUB 470
240 S$="L4.G;L8;A;L4;F"
250 PRINT S$
260 PLAY S$
270 GOSUB 470
280 S$="T4;O2;CDEGGAGL2E;T16;CDEGGAGL2E"
290 PRINT S$
300 PLAY S$
310 GOSUB 470
320 PRINT "VARYING VOLUME 1-31"
330 FOR V1=1 TO 31
340 PLAY "LBV"+STR$(V1)+"CD"
350 NEXT V1
360 GOSUB 470
```

```

370 S$="CDP4;EDEG;P2;GEC"
380 PRINT S$
390 PLAY S$
400 GOSUB 470
410 A$="L4CCGGAAL2G"
420 PRINT "A$=";A$
430 S$="XA$;O3;XA$;O2;XA$;"
440 PRINT S$
450 PLAY S$
460 CLS:GOTO 510
470 FOR D=1 TO 500:NEXT
480 CLS
490 PLAY "O2;L4;T2;V15"
500 RETURN
510 END

```

By altering the numbers in the various PLAY commands, you can experiment with different tones or volumes.




Also see SOUND

## PMODE

### Extended BASIC only



PMODE *m,p* sets up the resolution mode *m* to be used and the starting page *p* of screen memory. Mode *m* is a numeric expression from 0 to 4. The default value is 2 or the current value. *p* is the start-page from 1 to 8. This number is optional; if it is omitted, the previously set page is used. If you have not used PMODE since power-up, the default value is PMODE 2,1. The modes selected require a certain amount of memory which must be cleared with a PCLEAR statement (see PCLEAR for chart).

To see the page you are working on in a program, you will need the SCREEN statement which specifies text or graphics page plus the color set.

PMODE	Resolution	Color Set SCREEN	Color Combination
4	256 x 192		0 Black/Green 1 Black/Buf
3	128 x 192		0 Green/Yellow/Blue/Red 1 Buff/Cyan/Magenta/Orange
2	128 x 192		0 Black/Green 1 Black/Buf

## PMODE

---

1	128 x 96		0	Green/Yellow/Blue/Red
			1	Buff/Cyan/Magenta/Orange
0	128 x 96		0	Black/Green
			1	Black/Buff

### Valid statements

```
100 PMODE 3,1
120 PMODE 4,X
130 PMODE A,1
```

### Sample program

```
100 FOR M=0 TO 4
110 PMODE M,1
120 PCLS
130 SCREEN 1,1
140 CIRCLE(128,90),40
150 LINE(0,0)-(255,191),PSET
160 FOR D=1 TO 1000:NEXT D
170 NEXT M
180 END
```

Also see PCLEAR, SCREEN

## POINT

POINT( $x,y$ ) tells whether a point at location  $x,y$  is lit up (SET) or not.  $x$  is the horizontal position from 0 to 63, and  $y$  is the vertical position from 0 to 31. If the point has not been set, the value of POINT( $x,y$ ) will be 0 if it is black, or -1 if it is the background color. The point is black if it is next to a point that has been set in the same block. If the point has been set, the value will be the color number of the point.

### Valid statements

```
PRINT POINT(13,15)
40 IF POINT(H,V)=4 THEN 600
250 S=POINT(32,Y)
```

### Sample program

```
100 CLS
110 FOR X=0 TO 63
120 SET (X,0,4)
130 SET (X,1,4)
140 SET (X,2,3)
```

```

150 SET (X,3,3)
160 SET (X,28,3)
170 SET (X,29,3)
180 SET (X,30,4)
190 SET (X,31,4)
200 NEXT X
210 FOR Y=4 TO 27
220 SET (0,Y,3)
230 SET (1,Y,3)
240 SET (2,Y,2)
250 SET (3,Y,2)
260 SET (60,Y,2)
270 SET (61,Y,2)
280 SET (62,Y,3)
290 SET (63,Y,3)
300 NEXT Y
310 X=RND(63):Y=RND(31)
320 P=POINT(X,Y)
330 PRINT @260,P;
340 IF P<2 THEN SET (X,Y,3):GOTO 310
350 SOUND RND(100)+155,2
360 SET (X,Y,1)
370 GOTO 310
380 END

```

Change the values in line 310 for different results.

## POKE

POKE *a,v* allows you to change the value of contents of memory. The new value *v* is stored in the specified address *a*. The values and addresses vary with different brands and models of computers.

### Valid statements

```

POKE 151,64
100 POKE 65495,0
300 POKE A,D

```

### Sample application

```

POKE 25,6:NEW (same as PCLEAR 0)
POKE 65495,0 (speeds execution — may not work on all
              Color Computers)
POKE 65494,0 (return to normal speed)

```

Also see PEEK

# POS

---

## POS

### Extended BASIC only

POS(0) returns the POSITION of the cursor on device #0, the screen. POS(-2) returns the POSITION of the printhead on device #-2, the printer. The number returned is the column number.

### Valid statements

```
PRINT POS(-2)
150 PRINT POS(0)
230 P=POS(0)
```

### Sample programs

```
100 CLS
110 PRINT "START TYPING."
120 K$=INKEY$: IF K$="" THEN 120
130 IF POS(0)>14 THEN PRINT CHR$(13);
140 PRINT K$;
150 GOTO 120
160 END
```

In effect, this is an automatic ENTER, or carriage return.

```
100 PRINT #-2, "HELLO";
110 PRINT POS(-2)
120 END
```

## PPOINT

### Extended BASIC only

PPOINT(*x,y*) returns the color value of the specified graphics point at location *x,y*, where *x* is the horizontal distance from 0 to 255 and *y* is the vertical distance from 0 to 191. For accuracy, be sure to use PPOINT in the same PMODE you PSET the graphics points.

### Valid statements

```
PRINT PPOINT(126,96)
200 IF PPOINT(A,B)=4 THEN 350
300 ON PPOINT(X,Y) GOTO 1000,350,560,890
```

### Sample program

```
100 PMODE 3,1
110 PCLS
120 SCREEN 1,0
```



```
130 CIRCLE(128,96),40
140 PAINT(128,96),4,4
150 COLOR 3,1
160 LINE(20,20)-(100,50),PSET,BF
170 DRAW "C2BM20,180;E30F30L60"
180 PAINT(24,178),2,2
190 X=RND(255):Y=RND(191)
200 P=PPOINT(X,Y)
210 IF P=4 THEN SOUND 200,3:PSET(X,Y,1):GOTO
    190
220 IF P=3 THEN SOUND 100,3:PSET(X,Y,1):GOTO
    190
230 IF P=2 THEN SOUND 50,3:PSET(X,Y,1):GOTO
    190
240 PSET(X,Y,4)
250 GOTO 190
260 END
```

What will happen if you let this program run?

*Also see* PSET

## PRESET

### Extended BASIC only

PRESET(*x,y*) may be thought of as Point RESET. The dot at horizontal location *x* and vertical location *y* is turned off, or RESET to the background screen color.

### Valid statements

```
350 PRESET(200,J)
400 PRESET(X,Y)
800 PUT (100,100)-(120,120),Z,PRESET
```

### Sample program

```
100 PMODE 3,1
110 PCLS
120 SCREEN 1,0
130 CIRCLE(128,96),50
140 PAINT(128,96),4,4
150 FOR X=100 TO 156
160 FOR Y=90 TO 102
170 PRESET(X,Y)
180 NEXT Y,X
190 GOTO 190
200 END
```

*Also see* PUT

# PRINT

---

## PRINT

The PRINT statement allows you to print numbers and strings on the screen. There are several forms of the PRINT command, which are discussed on the following pages.

### Valid statements

PRINT Prints a blank line.  
PRINT N Prints a number N.  
PRINT S\$ Prints a string variable S\$.  
PRINT "HI" Prints the message in quotes.

The print separators are the semicolon and the comma. The semicolon prints the next expression listed right after the preceding expression. The comma starts the next expression in the next print field — starting in column 0 or column 16. A separate PRINT statement will start in the next row.

### Valid statements

```
PRINT "A=";A  
PRINT NAME$,PHONE$  
350 PRINT "HELLO";N$  
400 PRINT X,Y;Z
```

### Sample program

```
100 CLS  
110 PRINT "THIS IS AN EXAMPLE."  
120 PRINT  
130 N$="BOB"  
140 M$="RANDY"  
150 PRINT N$;M$  
160 PRINT N$,M$  
170 A=7:B=8  
180 PRINT  
190 PRINT "A=";A  
200 PRINT "A+B =",A+B  
210 END
```

Also see PRINT @, PRINT #, PRINT USING, PRINT TAB, TAB, USING

## PRINT @

There are 512 positions on the text screen, numbered from 0 to 511. The PRINT @ statement may specify where you want an item to start printing. The following chart gives the positions. Add the row number to the column number for the PRINT @ position. Example: third row, third column is PRINT @ 66.

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

PRINT @ Screen Locations

# PRINT @

---

## Valid statements

150 PRINT @226,"HELLO"

160 PRINT @X+4,M\$

350 PRINT @165,

(The next item to be printed will start in position 165 — this does not work in Extended BASIC.)

## Sample program

```
100 CLS
110 PRINT "HELLO THERE"
120 PRINT @45,"POSITION @45"
130 PRINT @92,"POSITION 60"
140 N$="KELLY"
150 PRINT @215,N$
160 A=24:B=25
170 PRINT @256,A*B
180 R$(1)="ED":R$(2)="BILL"
190 R$(3)="JOHN":R$(4)="JIM"
200 I=RND(4)
210 PRINT @335+I,R$(I)
220 END
```

## PRINT #

PRINT #-1,*d* prints a data item *d* as an output item to device #-1, the cassette, for data files. PRINT #-1 statements must be preceded by an OPEN "O",#-1 statement to OPEN the output file.

PRINT #-2,*i* prints item *i* on the printer if it is connected to the serial I/O jack at the back of the computer. The OPEN command is not necessary.

If your printer has lowercase capability, any letters which appear in reversed colors (green with black background) on the screen will be printed in lowercase on the printer. Press SHIFT and 0 (zero) simultaneously to get into reverse mode. To then get a capital letter, press SHIFT and the letter. To return all letters to normal mode, again press SHIFT and 0 together.

PRINT #*n*,*d*, where *n* is a number from 1 to 15 inclusive, writes item *d* to a disk file. The file *n* must have been previously OPENed and specified as an input file. In most cases, the WRITE #*n* command is easier to use than PRINT #*n*. See the *Disk System Owners Manual & Programming Guide* for more details.

## Valid statements

230 PRINT #-1,N

```

450 PRINT #-1,NA$
PRINT #-2,"HELLO"
120 PRINT #-2,A
240 PRINT #-2,M$
320 PRINT #-2,TAB(9);A$
400 PRINT #-2, USING "$#.##";M
PRINT #1,"NAME"
PRINT #3,M$
PRINT #5,A,B,Z

```

## Sample program

Creating data files on tape.

```

100 CLS
110 PRINT "CREATE A FILE"
120 PRINT "PREPARE CASSETTE TO RECORD,"
130 PRINT "THEN PRESS ANY KEY."
140 A$=INKEY$: IF A$="" THEN 140
150 OPEN "0", #-1, "SCORES"
160 PRINT:PRINT "ENTER SCORES."
170 PRINT "ENTER ZZZ TO END."
180 PRINT
190 INPUT "NAME ";N$
200 IF N$="ZZZ" THEN 250
210 PRINT #-1,N$
220 INPUT "SCORE = ";S
230 PRINT #-1,S
240 GOTO 180
250 CLOSE #-1
260 END

```

Try this on your printer.

## Sample program

```

100 CLS
110 PRINT "PREPARE PRINTER"
120 PRINT "THEN PRESS <ENTER>"
130 PRINT
140 A$=INKEY$: IF A$="" THEN 140
150 IF ASC(A$)<>13 THEN 140
160 PRINT #-2,"THIS IS A SAMPLE."
170 M$="MESSAGE"
180 PRINT #-2,M$
190 A=25
200 PRINT #-2,"A = ";A
210 PRINT #-2,TAB(15);A
220 END

```

Also see CLOSE, OPEN, PRINT USING

# PRINT TAB

---

## PRINT TAB

PRINT TAB(*t*) tabulates over *t* columns before printing. Note: Printing will start in the *next* column. PRINT TAB(2)"HI" will start the *H* in the third column. You may prefer to think of the screen as starting with column 0.

You may use either a semicolon or no punctuation between the parenthesis and the item to be printed.

Keep in mind that positive numbers print one blank space, then the number.

If the item to be printed is longer than the rest of the line, printing will simply "wrap" to the next line.

### Valid statements

```
PRINT TAB(5)"HELLO"  
350 PRINT TAB(T);T$  
400 PRINT TAB(N);N  
510 PRINT #-2,TAB(10);A$
```

### Sample program

```
100 CLS  
110 PRINT "01234567890123456789012345678901"  
120 PRINT TAB(5);"HELLO"  
130 PRINT TAB(7)"TESTING TAB(7)"  
140 X=4:Y=5:Z=-6  
150 PRINT TAB(X)Y  
160 PRINT TAB(Y);Z  
170 PRINT TAB(X+Y)3*Y  
180 A$="LEWIS":B$="MELISSA"  
190 PRINT TAB(15)A$  
200 PRINT TAB(28)B$  
210 PRINT  
220 END
```

Alter the numbers in line 140 for different results. Try printing columns of words using this TAB command.

Also see PRINT #-2, TAB

## PRINT USING

### Extended BASIC only

PRINT USING *format;item* uses a specified *format* to print *items*. USING lets you use dollar signs and format columns of numbers, for example. You might right-justify a list of numbers and left-justify a list of names. USING can replace many lines of logic

to achieve the same printing result. The following field specifiers may be used.

#	Indicates position of digits in numeric expressions.
####	Positions four digits by right-justifying (ones column lined up, tens column lined up, etc.).
##.###	Allows three decimal places. For numbers with more than three decimal places, the printed number will be rounded. The computer will fill in with zeros if the number does not have three decimal places.
\$\$\$.	Allows printing money using the dollar sign and rounding to the second decimal place.
###,###,###	Places commas in large numbers. A comma anywhere between the first # sign and a decimal will indicate automatic comma placing.
**####	Places leading asterisks in field up to the number. Use ** plus the correct number of # signs.
**\$	Places leading asterisks before a dollar amount.
↑↑↑↑	Four up-arrows after a number specification indicates exponential form.
+####	Places a + sign before positive numbers and a - sign before negative numbers.
##+	Places a + sign after positive numbers and a - sign after negative numbers.
##-	Places a - sign after negative numbers and a space after positive numbers.
\$\$\$\$	Floating dollar sign; places dollar sign immediately before the number with right justification.
!	Prints first string character only in <i>item</i> .
% %	Designates field for strings. The length of the field is 2 plus the number of spaces between the % symbols.

## Valid statements

PRINT USING \$\$\$.;10.358

150 PRINT USING ##,##.##;N

200 F\$="! % %"

210 PRINT USING F\$;N\$,D\$

250 PRINT USING "##.##↑↑↑↑";82858 (no space between # and !)

The PRINT USING statement can also be used to format output to the printer or disk drive.

# PRINT USING

---

```
100 PRINT #-2, USING "$#.##";D
200 OPEN "O",#1,"MONEY/DAT"
210 PRINT #1, USING "$#.##";M
```

## Sample program

```
100 CLS
110 PRINT USING "####.##";123.456
120 PRINT USING "####.##";25.789
130 PRINT USING "####.##";6
140 PRINT
150 FOR I=1 TO 5
160 READ N
170 PRINT USING "#,###.#";N
180 NEXT I
190 DATA 1234.568,536.888,34.105,3.99,2256
200 GOSUB 560
210 A$="%{5 SPACES}%"
220 FOR I=1 TO 5
230 READ N$
240 PRINT USING A$;N$
250 NEXT I
260 DATA SHEILA,DOUGLAS,ROGER,CHAN,BOB
270 PRINT
280 PRINT USING "##.# ";18.9,22.32,2,32.15
290 PRINT USING "###,###,#";9876543
300 PRINT USING "##### ";123,25,4
310 PRINT USING "##.##↑↑↑↑";35543
320 GOSUB 560
330 RESTORE
340 PRINT
350 FOR I=1 TO 5
360 READ N
370 PRINT USING "#####,##.##";N
380 NEXT I
390 RESTORE
400 FOR I=1 TO 5
410 READ N
420 PRINT USING "$#####.##";N
430 NEXT I
440 GOSUB 560
450 FOR I=1 TO 5
460 READ N$
470 PRINT USING "!";N$
480 NEXT I
490 FOR I=1 TO 5
500 READ N
510 PRINT USING "+###";N
520 NEXT I
```



```

530 DATA 234,-122,73,-55,-2
540 PRINT
550 END
560 PRINT
570 PRINT "PRESS <ENTER>"
580 A$=INKEY$:IF A$="" THEN 580
590 IF ASC(A$)<>13 THEN 580
600 CLS:RETURN
610 END

```

This program demonstrates the various uses of the PRINT USING command.

*Also see* USING

## PSET

### Extended BASIC only

PSET( $x,y,c$ ) places a point of color  $c$  at a location of horizontal location  $x$  and vertical location  $y$ , where  $c$  is one of the color numbers from 0 to 8,  $x$  may be from 0 to 255, and  $y$  may be from 0 to 191.

All three numbers may be numeric expressions.

PSET is also used in commands to set points to the foreground color or color specified in a COLOR statement. Examples are the LINE and PUT statements.

### Valid statements

```

250 PSET(200,100,4)
270 PSET(I,J,3)
350 LINE(20,30)-(70,50),PSET
540 PUT (A,B)-(A+20,B+10),I,PSET

```

### Sample program

```

100 PMODE 1,1
110 PCLS
120 SCREEN 1,1
130 LINE(0,0)-(90,50),PSET
150 C=RND(3)+5
160 X=RND(255):Y=RND(191)
170 PSET(X,Y,C)
180 GOTO 150
190 END

```

*Also see* GET, LINE, PPOINT, PUT

## PUT

### Extended BASIC only

After you GET a graphic display in a particular rectangle  $t$ , you may PUT the graphics back on the screen in a different place or later in the program. GET and PUT can move objects more quickly than DRAWing the graphics each time. The form is

`PUT(x1,y1)-(x2,y2),t,a`

$(x1,y1)$  is the location of the upper-left corner of the rectangle containing the graphics, and  $(x2,y2)$  is the location of the lower-right corner, where  $x1$  and  $x2$  may be from 0 to 255, and  $y1$  and  $y2$  may be from 0 to 191.  $t$  is the variable name of the array, which must have been previously DIMensioned.  $a$  is the action which determines how the new display will be shown on the screen. PSET sets each point that was in the GET rectangle. PRESET resets each point that was set in the GET rectangle. AND, OR, and NOT are logical operators that may be used as actions. AND compares points set in the original rectangle with the destination rectangle. If both points are set, the screen point will be set; otherwise, the point is reset. OR compares the points, and if either is SET, the point will be SET. NOT reverses the state of each point in the destination rectangle regardless of the PUT array's contents.

You must PUT in the same PMODE as the GET. The array  $t$  must be DIMensioned using the size of the rectangle.

### Valid statements

`250 PUT(100,100)-(120,110),A,PSET`

`350 PUT(I,J)-(K,L),T,PSET`

### Sample program

```
100 PMODE 4,1
110 PCLS
120 SCREEN 1,1
130 DIM A(16,16)
140 DRAW"BM0,191;E16NL8NDB"
150 GET(0,175)-(16,191),A,G
160 X=0:Y=175
170 PCLS
180 PUT(X,Y)-(X+16,Y+16),A,PSET
190 X=X+16:Y=Y-16
200 IF X+16>255 OR Y-16<0 THEN 160 ELSE 170
210 END
```

*Also see* DIM, GET, PRESET, PSET

## READ

READ *v* reads an item from the DATA list and assigns it to the variable *v*, which may be either numeric or string. You can read several items in one READ statement. The type of DATA must correspond to the type of variable name (numeric or string). There must also be enough DATA to supply the READ statements.

The first READ statement reads the first item in the first DATA statement. The computer keeps track of which data items have been used, while each READ statement takes the next data item in turn. A RESTORE statement starts the DATA list over with the first item when the next READ statement is executed.

### Valid statements

```
250 READ N
350 READ A$
420 READ A$(I)
500 READ SCORE(N),NAME$(N)
```

### Sample program

```
100 CLS
110 READ A
120 READ B
130 PRINT A*B
140 READ C,D
150 PRINT C-D
160 FOR I=1 TO 7
170 READ N$
180 PRINT N$
190 NEXT I
200 DATA 3,5,12,5
210 DATA CHERY,RICHARD,CINDY,BOB,RANDY,GEORG
    E,SUSAN
220 END
```

Change the values in line 160 to 1 TO 5. How many names from line 210 will print on the screen?

*Also see* DATA, RESTORE

## REM

REM indicates a REMark statement, which is ignored by the computer. The abbreviation is ' (apostrophe). You may place REM statements anywhere in the program to create visual spaces between lines or to document the program.

# REM

---

## Valid statements

```
100 REM TITLE
110 ' CALCULATE FACTORS
120 PRINT: PRINTS BLANK LINE
450 REM
```

Before you RUN this sample program, which lines will PRINT? If you enter RUN again, will the screen clear?

## Sample program

```
100 REM SAMPLE PROGRAM
110 REM PRINT MESSAGE
120 REM CLS
130 PRINT "GRANT IS MY FRIEND."
140 A=5
150 REM A=9
160 PRINT "A = ";A
170 REM STATEMENT IGNORED
180 REM
190 END
```

# RENUM

## Extended BASIC only

RENUM allows you to renumber the lines in your program, including all lines referenced by other lines (except in REMARK statements). The command RENUM will renumber all lines. The first line will become line 10, and the lines will increment by 10.

RENUM  $n$  will renumber all lines, but the first line will start with the number  $n$  you specify. The increment will be 10.

RENUM  $n,s,i$  will renumber your program starting with line  $s$ , assigning the old line  $s$  with the new line number  $n$ , and incrementing by  $i$ .  $n$  is optional; the default value makes the first line number 10.  $s$  is optional; if  $s$  is omitted, the whole program will be renumbered.  $i$  is optional; the default value is 10.

You cannot specify  $n$  and  $s$  in such a way that  $s$  would change the order of existing lines. The lines must stay in the same order.

It *does* take less memory to use smaller line numbers. You may wish to develop a program with different segments starting at 100, 1000, 2000, 3000, etc.; then when you have completed the program you may RENUM 1,1,1 to use the least required amount of memory.

Many programmers like to increment lines by 10 so that they may later add lines between 20 and 30, for example. If you have more than ten lines to put between two existing program lines which are only ten apart, first RENUM 100,50 to get more room between statements.

## Valid commands

```
RENUM  
RENUM 100  
RENUM 1,1,1  
RENUM 100,10,5
```

## Sample application

Type in the following program, then LIST.

```
100 CLS  
110 A=8:B=9  
120 PRINT A,B  
130 IF A<7 THEN 110  
140 ON A-7 GOTO 150,170,190  
150 PRINT "SAMPLE"  
160 GOTO 200  
170 PRINT "A-7=2":GOTO 200  
180 REM YOU SHOULD NOT GET HERE  
190 PRINT "A-7=3"  
200 END
```

Type RENUM and press ENTER. Then LIST. Notice the lines start with 10 and increment by 10. Notice that all line numbers after THEN, GOTO, and ON ... GOTO are appropriately changed.

Type RENUM ,,5 and press ENTER. LIST.

Type RENUM 1,1,1 and press ENTER. LIST.

Type RENUM 100,8,10 and press ENTER. LIST. This command indicates to start renumbering, with line 8 changed to 100, and increment by 10.

Try your own combinations of numbers.

## RESET

RESET( $x,y$ ) erases a dot (RESETs the point) that was previously SET at location  $x,y$ .  $x$  is a number from 0 to 63, and  $y$  is a number from 0 to 31. By using a combination of SET and RESET, you may "blink" dots or create moving characters.

# RESET

---

## Valid statements

```
300 RESET(33,20)
400 RESET(X+4,Y)
```

## Sample program

```
100 CLS
110 FOR C=10 TO 53
120 FOR R=10 TO 21
130 SET (C,R,4)
140 NEXT R,C
150 FOR R=17 TO 14 STEP -1
160 FOR C=20 TO 43
170 RESET (C,R)
180 NEXT C,R
190 C=RND(23)+20
200 R=RND(4)+13
210 SET (C,R,3)
220 RESET (C,R)
230 GOTO 190
240 END
```

*Also see SET*

## RESTORE

RESTORE is used in conjunction with DATA and READ statements. Ordinarily, the READ statements read the DATA items in exact order. The computer keeps track of which DATA item has been read and sets a pointer. The next READ statement uses the next DATA item, no matter where the DATA statements are placed in the program. RESTORE restores the pointer to the first DATA item for the next READ statement so that the READ statement following the RESTORE will start again with the first DATA item, even if there were later DATA items that had not been READ.

## Valid statement

```
300 RESTORE
```

## Sample program

```
100 CLS
110 FOR I=1 TO 5
120 READ N
130 PRINT N
140 NEXT I
150 RESTORE
160 PRINT
```

```
170 TN=1
180 FOR I=1 TO 5
190 READ N
200 TN=TN*N
210 PRINT TN;
220 NEXT I
230 PRINT:PRINT
240 RESTORE
250 TN=0
260 FOR I=1 TO 5
270 READ N
280 TN=TN+N
290 PRINT TN
300 NEXT I
310 DATA 5,8,9,2,4
320 PRINT
330 END
```

Note how the same DATA items in line 310 are used several times, for different purposes, through the RESTORE command.

*Also see* DATA, PRINT USING, READ

## RETURN

RETURN is the last statement of a subroutine and RETURNS the program execution to the command just after the GOSUB command. Subroutines can be placed anywhere in the program but must end with a RETURN statement. You also need to place a statement before the subroutines so program flow will not enter the subroutine; otherwise, when the computer hits the RETURN statement, you will get an error. One subroutine may include several RETURN statements if there are branching statements that cause a RETURN with different conditions.

### Valid statement

```
590 RETURN
```

### Sample program

```
100 CLS
110 PRINT "START PROGRAM"
120 GOSUB 190
130 PRINT "NOW ON LINE 130"
140 GOSUB 230
150 PRINT "NOW ON LINE 150"
160 GOSUB 190
170 PRINT "NOW ON LINE 170"
```

# RETURN

---

```
180 GOTO 260
190 PRINT
200 PRINT "SUBROUTINE 500"
210 PRINT
220 RETURN
230 PRINT
240 PRINT "SUBROUTINE 600"
250 PRINT:RETURN
260 END
```

Note how both GOSUB commands on lines 120 and 160 are RETURNed by the command in line 220.

*Also see GOSUB*

## RIGHT\$

RIGHT\$(s,n) is a string function that returns the last (or right) *n* characters in string *s*. *n* must be a number zero or greater. If the number *n* is greater than the length of the string *s*, only the original string is returned (no added spaces).

### Valid statements

```
250 R$=RIGHT$(S$,3)
PRINT RIGHT$(NAME$,5)
560 IF RIGHT$(X$,2)="ON" THEN 780
```

### Sample program

```
100 CLS
110 A$="SAMPLE OF RIGHT$"
120 PRINT A$
130 PRINT
140 PRINT "RIGHT$(A$, )", "STRING"
150 PRINT
160 PRINT TAB(9);4,RIGHT$(A$,4)
170 PRINT TAB(9);12,RIGHT$(A$,12)
180 PRINT TAB(9);0,RIGHT$(A$,0)
190 PRINT TAB(9);20,RIGHT$(A$,20)
200 X=8
210 PRINT TAB(10)"X",RIGHT$(A$,X)
220 PRINT
230 END
```

*Also see LEFT\$, MID\$*



## RND

RND( $n$ ) where  $n$  is a number greater than zero chooses a random integer (whole number) from 1 to  $n$ . For example, RND(6) could be the number of dots showing on dice — 1, 2, 3, 4, 5, or 6.

RND( $n$ ) where  $n$  is less than or equal to zero (a negative number) returns a decimal fraction.

On start-up, the computer always gives the same sequence of random numbers. If you have Extended BASIC and want your game to be more randomized, add a statement such as  $X = \text{RND}(\text{TIMER})$  or  $X = \text{RND}(-\text{TIMER})$  at the beginning of the program.

### Valid statements

```
100 X = RND(-TIMER)
230 D = RND(6) + RND(6)
510 Y = RND(40) + 100
```

### Sample program

```
100 CLS
110 FOR I=1 TO 8
120 PRINT RND(10)
130 NEXT I
140 PRINT RND(0)
150 FOR C=1 TO 4
160 PRINT RND(20) + 100
170 NEXT C
180 END
```

Notice what line 140 produces. Create a FOR-NEXT loop for this demonstration of the RND command.

## RUN

RUN is the command that starts the program RUNNING or executes the program from its beginning. You can also specify a starting line number with RUN  $l$  where  $l$  is one of the program line numbers.

There are several ways to stop a program while it is running. Press the BREAK key on the keyboard. You can then print any variable values if you wish, then CONTINUE the program by typing CONT and pressing ENTER, or you can EDIT or LIST or change your program.

If you do not want BREAKPOINT to occur, press SHIFT and the @ key at the same time. To continue the program, press any key.

# RUN

---

A STOP command in the program acts just like the BREAK key. The program will also stop at an END statement or with syntax or other errors.

## Valid commands

RUN           Executes from beginning of program.  
RUN 320       Starts executing at line 320.

## Sample program

```
100 CLS
110 PRINT " 110"
120 PRINT " 120"
130 PRINT " 130"
140 PRINT " 140"
150 PRINT " 150"
160 PRINT " 160"
170 END
```

RUN the program. Try using RUN with different line numbers.

If you have a disk system, you can load and run a program stored on disk with the command:

RUN "filename"

where "filename" is the name of the program on disk.

## SAVE

### Disk BASIC only

SAVE "TITLE" is used to store a program on disk. The equivalent command for cassette is CSAVE.

Unlike CSAVE, the program name is not optional when storing programs on disk. The name can be up to eight characters long. Also, a three-character *extension* should be added to the title, separated by a slash (/). If no extension is specified, the computer will add the extension /BAS.

To save a program as an ASCII text file instead of as a tokenized program file, add the letter A after the title.

## Valid commands

SAVE "PROGRAM1"  
SAVE "DEMO/PRG"  
SAVE "LETTER/DAT",A

Also see CLOAD, CSAVE, LOAD

## SCREEN

### Extended BASIC only

SCREEN  $t,c$  determines the screen type  $t$ , text or graphics, and the color set  $c$ . If you designate a PMODE for graphics for drawing, you will not actually see the graphics on the screen until you use the SCREEN statement.

The screen type  $t$  is 0 (zero) for text screen and 1 for graphics screen. The color set  $c$  is either 0 or 1. 0 is used for black/green in the two-color mode and green/yellow/blue/red in the four-color mode. 1 is used for black/buff in the two-color mode and buff/cyan/magenta/orange in the four-color mode.

### Valid statements

```
120 SCREEN 1,1
240 SCREEN X,1
520 SCREEN 0,0
```

### Sample program

```
100 FOR D=1 TO 1000:NEXT
110 SCREEN 0,1
120 FOR D=1 TO 1000:NEXT
130 SCREEN 0,0
140 FOR D=1 TO 1000:NEXT
150 FOR P=0 TO 4
160 PMODE P,1
170 PCLS
180 SCREEN 1,0
190 CIRCLE(80,80),50
200 FOR D=1 TO 1000:NEXT D
210 SCREEN 1,1
220 FOR D=1 TO 1000:NEXT D
230 NEXT P
240 END
```

## SET

SET( $x,y,c$ ) places a dot of color  $c$  on the screen in a position with  $x$  horizontal location and  $y$  vertical location.  $x$  is a number from 0 to 63, and  $y$  is a number from 0 to 31. Color  $c$  is a number from 0 to 8. Any of the numbers may be numeric expressions which are evaluated.

Within blocks of four dots on the grid chart, all four dots must be SET either the same color or one color and black. In other words, within a four-dot block you cannot have a red dot

## SET

---

and a blue dot. Also, when you set one of the four dots, the others go black.

### Valid statements

```
250 SET(X,Y,C)
350 SET(X+1,12,4)
```

### Sample program

```
100 CLS
110 C=4
120 GOSUB 180
130 I=RND(30)*2
140 J=RND(14)*2
150 SET(I,J,3):SET(I+1,J,3)
160 SET(I,J+1,3):SET(I+1,J+1,3)
170 GOTO 130
180 FOR I=0 TO 63
190 SET(I,0,C)
200 SET(I,1,C)
210 SET(I,30,C)
220 SET(I,31,C)
230 NEXT I
240 FOR I=2 TO 29
250 SET(0,I,C)
260 SET(1,I,C)
270 SET(62,I,C)
280 SET(63,I,C)
290 NEXT I
300 RETURN
310 END
```

The subroutine beginning at line 180 draws a border of color C.

Also see RESET

## SGN

$\text{SGN}(n)$  returns the sign of the numeric expression  $n$ . The expression is first evaluated and must result in a number between  $-10^{38}$  and  $10^{38}$ . If the number evaluated is positive,  $\text{SGN}(n)$  will be 1. If the number is negative,  $\text{SGN}(n)$  will be  $-1$ . If the number is zero,  $\text{SGN}(n)$  will be 0.

### Valid statements

```
PRINT SGN(X)
A=SGN(T+5)
B=C*SGN(R)
```

In the following sample program, pressing the left arrow key will produce a negative value, and, using the GOSUB command in line 150, will SHIFT the program to line 170. Pressing the right arrow key will produce a positive value, and SHIFT to line 210. Any other key will move the program to line 190.

### Sample program

```

100 CLS
110 PRINT "PRESS LEFT OR RIGHT ARROW"
120 A$=INKEY$: IF A$="" THEN 120
130 IF ASC(A$)=8 THEN X=-1:GOTO 150
140 IF ASC(A$)=9 THEN X=+1 ELSE X=0
150 ON SGN(X)+2 GOSUB 170,190,210
160 GOTO 120
170 PRINT "- ";
180 RETURN
190 PRINT "0 ";
200 RETURN
210 PRINT "+ ";
220 RETURN
230 END

```

The SGN command is easily used with the GOTO or GOSUB command to provide multiple branching to other sections of the program, or to subroutines.

## SIN

$SIN(n)$  is a function that evaluates the sine of angle  $n$  where  $n$  is in radians and is a numeric expression from  $-10^{38}$  to  $10^{38}$ . If  $n$  is about  $4E+09$  the value returned is zero.

If the angle is in degrees, first multiply by  $\pi/180$  or  $(4*ATN(1))/180$  or  $0.0174532925$  to convert the angle to radians.

### Valid statements

```

PRINT SIN(1)
230 PRINT SIN(A)
280 F= SIN(A)+SIN(B)

```

### Sample program

```

100 PMODE 4,1
110 PCLS
120 SCREEN 1,0
130 LINE(0,96)-(255,96),PSET
140 FOR S=0 TO 25 STEP .4

```

```
150 G=96-(40*SIN(S))
160 LINE(S*10,G)-(S*10,96),PSET
170 NEXT S
180 GOTO 180
190 END
```

Refer to the sample program for the ATN command to see how to write a short program of your own which will result in both the radians and degrees expressed for the SIN function.

## SKIPF

SKIPF or SKIPF "*name*" may be used to skip past a program which has been recorded on the cassette tape to position the tape to save data or another program.

### Valid commands

SKIPF                Skips the next program.  
SKIPF "*name*"       Searches for the specified title then skips to the *end* of that program.

### Sample application

You may wish to keep copies of several programs on one tape. Store cassette tapes in the rewind position, so if something happens to the bare tape it won't be on a valuable program. If you want to add a program to that tape, use SKIPF or SKIPF "TITLE" to skip past the last previously saved program. Fast forward slightly to leave a little space, then record your next program.

## SOUND

SOUND *n,t* creates music on the computer by playing a specified tone *n* for a specified length of time *t*. Both numbers may be from 1 to 255. For the tones, 1 is the lowest and 255 is the highest. For the time, 1 is the shortest (about .06 second) and 255 is the longest (about 15.30 seconds).

### Valid statements

```
350 SOUND 5,10
420 SOUND 89,2
```

The following chart specifies the numbers *n* which may be used for tones on the musical staff.

89 99 108 117 125 133 140 147 153

159 165 170 176 180 185 189 193 197

5 19 32 45 58 69 78

### Sample program

```

100 T=5
110 FOR I=1 TO 7
120 READ N
130 SOUND N,T
140 NEXT I
150 DATA 89,89,108,125,89,125,108
160 SOUND 32,T
170 SOUND 89,T
180 SOUND 89,T
190 SOUND 108,T
200 SOUND 125,T
210 SOUND 89,T*2
220 SOUND 78,T
230 END

```

Change line 100 to T=3, RUN, and notice the difference. You can change the whole song's tempo with one line.

Also see PLAY

## SQR

### Extended BASIC only

SQR(*n*) is a function which returns the square root of a numeric expression *n*, where *n* is zero or positive. The square root means

that a number times itself will result in the number  $n$ .

## Valid statements

```
PRINT SQR(144)
300 PRINT SQR(A + 36)
540 C = SQR(A*A + B*B)
```

## Sample program

```
100 CLS
110 FOR N=0 TO 100
120 PRINT SQR(N)
130 NEXT N
140 END
```

## STEP

STEP is an optional word in FOR-NEXT loops. STEP  $s$  indicates the increment size for the loop index.  $s$  is positive, negative, or a fraction. In the statement FOR I=L1 TO L2 STEP S, the index I will start at L1 to perform the loop. When the word NEXT is executed, I is incremented by S. If the new I is greater than L2, program control goes to the statement immediately after NEXT; otherwise, the loop is performed again, starting at the statement after FOR. If STEP is omitted, the increment size is one.

## Valid statements

```
130 FOR I= 1 TO 10 STEP 3
250 FOR J= 2 TO 12 STEP 2
300 FOR K= 25 TO 21 STEP -1
400 FOR L= 10 TO 20 STEP .5
```

## Sample program

```
100 CLS
110 FOR S=1 TO 10
120 FOR I=S TO 50 STEP S
130 FOR D=1 TO 100:NEXT D
140 PRINT I
150 NEXT I
160 NEXT S
170 PRINT
180 FOR I=10 TO 0 STEP -1
190 PRINT I
200 FOR D=1 TO 100:NEXT D
210 NEXT I
220 END
```



Alter the STEP value in line 180. What changes occur?

Also see FOR, NEXT

## STOP

STOP is a command that will stop the program from executing any more statements. The message will be

BREAK IN *l*

where *l* is the line number. You can then PRINT any variables to analyze your program. You can continue the program with all variables as is with the very next statement by entering CONT. You can RUN to restart the program from the beginning, or you can EDIT or LIST.

### Valid statement

```
990 STOP
```

### Sample program

```
100 CLS
110 PRINT "USE OF STOP"
120 X=1
130 Y=X*2+1
140 PRINT X,Y
150 X=X+1
160 IF X/10=INT(X/10) THEN 180
170 GOTO 130
180 PRINT "TYPE PRINT X <ENTER>"
190 PRINT "THEN TYPE CONT <ENTER>"
200 STOP
210 GOTO 130
220 END
```

## STRINGS

### Extended BASIC only

STRING\$(*n,c*) is a string function that creates a string of length *n* of the character *c*. If *c* is a numeric expression, the string will consist of characters with the ASCII code of *c*. This provides a simple method of creating long strings.

### Valid statements

```
210 P$=STRING$(12,"+")
365 B$=STRING$(25,32)
400 PRINT STRING$(128," ")
```

# STRINGS

---

## Sample program

```
100 CLS
110 B$=STRING$(25,32)
120 L$=STRING$(31,"=")
130 T$="PRESS ANY KEY TO CONTINUE"
140 PRINT @128,L$,TAB(10)"INSTRUCTIONS",L$
150 PRINT @419,T$
155 FOR I=1 TO 200:NEXT
160 IF INKEY$<>" " THEN 200
170 PRINT @419,B$
180 FOR I=1 TO 200:NEXT
190 GOTO 150
200 END
```

## STR\$(n)

STR\$(*n*) is a string function that converts a number *n* from a numeric expression or variable to a string expression. All characters in *n* must be numeric. You may wish to convert a number to a string to combine it with other strings for string manipulation.

### Valid statements

```
250 A$=STR$(A)
540 N$=STR$(3.14159)
620 B$=B$+STR$(AD)
```

## Sample program

```
100 CLS
110 FOR I=1 TO 4
120 READ N$,A
130 PRINT N$,A
140 X$(I)=STR$(A)+"", "+N$"
150 NEXT I
160 PRINT
170 FOR I=1 TO 4
180 PRINT X$(I)
190 NEXT I
200 DATA BRIAN,13,JARED,11,SHELLEY,8,SETH,4
210 END
```

## TAB

The TAB function is similar to the TAB key on a typewriter. You can specify a column in TAB(*c*) and the cursor will tabulate or skip over to that column before printing the next item. TAB (3);"ITEM"

will start printing ITEM in the fourth column. You may want to think of the screen with columns starting with 0 and going to 31; the message will start in the column *c* specified. Keep in mind that numbers which are positive (greater than zero) print a blank space before the number. The semicolon between the right parenthesis and the item to be printed is optional.

### Valid statements

```
100 PRINT TAB(10);"NAME"
120 PRINT TAB(X)A$
130 PRINT TAB(5);Z
150 PRINT TAB(3);A$;TAB(15);B$
200 PRINT #-2,TAB(7);N$
```

### Sample program

```
100 CLS
110 PRINT "01234567890123456789012345678901"
120 FOR T=1 TO 12
130 PRINT TAB(T); "HELLO"
140 NEXT T
150 END
```

Also see PRINT TAB, PRINT #-2

## TAN

### Extended BASIC only

TAN(*n*) is a function that returns the tangent of the angle *n* where *n* is expressed in radians and may be a number from  $-10^{38}$  to  $10^{38}$ . Larger numbers may return zero or a bad argument.

If the angle is in degrees, you can convert to radians by multiplying by  $\pi/180$  or  $(4*ATN(1))/180$  or 0.01745329251994.

### Valid statements

```
PRINT TAN(2)
150 PRINT TAN(N)
320 A = TAN(X) + TAN(Y)
```

### Sample program

```
100 CLS
110 ANGLE=30
120 C=0.01745329251944
130 RAD=ANGLE*C
140 PRINT TAN(RAD)
150 RAD=45*C
160 PRINT TAN(RAD)
```

# TAN

---

```
170 PRINT TAN(60*C)
180 END
```

*Also see ATN*

## THEN

THEN is a word in the IF-THEN conditional branching statement. THEN can be followed by an action or a line number. If the test condition is true, the action following THEN will be executed; if a line number is listed, program control will go to that line. The IF statement can also contain the word ELSE.

### Valid statements

```
250 IF A = 10 THEN 700
300 IF A > 0 THEN PRINT A
```

### Sample program

```
100 CLS
110 INPUT "ENTER NAME ";N$
120 IF LEN(N$)<5 THEN PRINT "LEN < 5":GOTO 140
130 PRINT "LEN >= 5"
140 IF ASC(N$)<78 THEN 190
150 PRINT "NAME IS IN LAST PART OF ALPHABET"
160 GOTO 200
170 PRINT TAN(60*C)
180 END
190 PRINT "NAME STARTS WITH LETTER BEFORE N"
200 PRINT
210 END
```

*Also see ELSE, IF*

## TIMER

### Extended BASIC only

TIMER either returns the contents of the timing function or allows setting for timing purposes. TIMER will return a value from 0 to 65535. When the computer is first turned on, TIMER starts incrementing. After the number 65535, TIMER starts over at zero. TIMER increments at a speed of about 60 counts per second. To calculate a certain amount of time during processing, you may use the following general idea:

```
200 A = TIMER    reads timer value
```

## TROFF and TRON

---

210 - 290            executes the program  
300 B = TIMER       reads timer value  
320 T = (B - A) / 60   T seconds

Another way you can use the TIMER function is to set TIMER to a certain value, such as 400 TIMER = 0, then later read the TIMER value.

### Valid statements

```
PRINT TIMER
200 A = TIMER
500 TIMER = 0
600 X = RND(-TIMER)
```

### Sample program

```
100 CLS
110 PRINT TIMER
120 PRINT
130 TIMER = 0
140 PRINT TIMER
150 FOR I = 1 TO 100
160 NEXT I
170 PRINT TIMER
180 PRINT
190 PRINT "TYPE YOUR NAME THEN PRESS ENTER."
200 TIMER = 0
210 INPUT N$
220 T = TIMER
230 PRINT "IT TOOK"; T; "COUNTS"
240 PRINT "OR APPROXIMATELY"; T / 60, "SECONDS"
250 PRINT
260 END
```

Refer to the "Typing Trainer" program in the last section of the book for an obvious application of the TIMER command.

## TROFF and TRON

### Extended BASIC only

TROFF and TRON are used to TRACE the flow of a program. TRON turns the tracer on. TROFF turns the tracer off. Either statement can be entered as a direct command or as a statement in the program. When the tracer is ON, program lines are listed as the program is RUN. You can see exactly which lines the computer is executing and in which order. The TRACING function is useful in debugging a program.

# TROFF and TRON

---

## Valid statements

```
TRON
TROFF
250 TRON
280 TROFF
```

## Sample program

```
100 CLS
110 TRON
120 FOR I=1 TO 10
130 PRINT I
140 NEXT I
150 TROFF
160 END
```

## USING

### Extended BASIC only

USING is a word used with PRINT to give field specifications for formatting output. USING allows right- or left-justification of columns of items, rounding to a certain decimal place, printing dollars and cents, printing in exponential form, and various other options. See PRINT USING for a complete list of field specifications.

### Valid statements

```
PRINT USING "$###.##";M
200 PRINT USING "***###";N
300 PRINT USING F$;N$
```

### Sample program

```
100 CLS
110 PRINT USING "$###.##";23.5688
120 PRINT USING "#,###,###";9876543
130 A$="##(3 SPACES)%<8 SPACES)%"
160 PRINT USING A$;12,"SHERRI"
170 PRINT USING A$;11,"KRISTI"
180 PRINT USING A$;9,"MICHAEL"
190 PRINT USING A$;2,"ANDY"
200 END
```

*Also see* PRINT #, PRINT USING

## VAL

VAL(s) returns the numeric value of string s. If s is a number, VAL(s) will return the number. If s is a character (or contains a character) other than a number, VAL(s) will be 0 (zero). If you are working with a string that contains several items, you can take the VAL of a portion of the string so you can calculate with the number portion.

### Valid statements

```
PRINT VAL(N$)
250 PRINT VAL(A$)
360 A = VAL(MID$(M$,5,3))
```

### Sample program

```
100 CLS
110 PRINT "ENTER A WHOLE NUMBER"
120 INPUT N$
130 FOR I=1 TO LEN(N$)
140 A=ASC(MID$(N$,I,1))
150 IF A>47 AND A<58 THEN 190
160 PRINT "NOT A WHOLE NUMBER."
170 PRINT
180 GOTO 110
190 NEXT I
200 N=VAL(N$)
210 PRINT "N SQUARED =" ; N*N
220 PRINT
230 END
```





**Two**

---

**Program-  
ming Tips  
and Hints**



### Typing and Editing

- ←** Backs up one position as you are typing and erases the last typed character.
- SHIFT** **←** Erases the line you are presently typing, and starts again at the beginning of the line.
- BREAK** Stops a listing.  
Stops a program as it is running. You may type CONT to continue the program, or RUN to start it over.
- SHIFT** **@** Pauses while a program is running or stops scrolling while you are listing a program. Press any key to continue.
- CLEAR** Clears screen, places cursor at upper-left corner of screen.
- RESET** Button at the back of the computer. Stops whatever you are doing, clears the screen, and places the cursor at the top of the screen. The program stays in memory if you are programming or running a program. If you have a Program Pak inserted, the program will start over.
- SHIFT** **0** Zero key. Reverses the characters from black on green to green on black, or from reversed characters to normal. Commands will not be accepted if they are typed with reversed characters. If you use a printer, the regular characters are capitals, and the reversed characters are lowercase.
- ↑** Prints ↑. Used for exponentiation, such as 2 ↑ 5.
- SHIFT** **↑** Prints ←.
- SHIFT** **↓** Prints left bracket, [.
- SHIFT** **→** Prints right bracket, ].  
Apostrophe, the abbreviation or token for REM in REMark statements. Example: 100 ' GAME.

- ? Abbreviation or token for PRINT. Example:  
100 ? X + Y.  
To delete a line as you are programming, type the line number, then press ENTER.

### POKE Commands

- POKE 65495,0** Speeds up the execution of a program. This may not work on all Color Computers. Do not use this higher speed during input/output (saving, loading, or printing to a printer), nor during SOUND and PLAY statements.
- POKE 65494,0** Returns to normal speed after the above command.
- POKE 65313,4** A substitute for the MOTOR ON command to turn the cassette recorder on.
- POKE 65313,52** A substitute for the MOTOR OFF command to turn the cassette recorder off.
- POKE 25,6:NEW** Clears more memory; equivalent to PCLEAR 0, which is not an acceptable BASIC command.
- POKE 113,0:EXEC 40999** Restarts the computer as if you had turned the power off then back on.
- EXEC 44539** Equivalent to the procedure:  
300 E\$ = INKEY\$:IF E\$ = "" THEN 300  
which pauses until a key is pressed.

### Conserving Memory

You may put more than one command per line number by separating commands with colons. Example:

```
100 CLS:PCLS:PMODE 4,1:SCREEN 1,1
```

To save memory, use lower line numbers. When you have completed a program, RENUM 1,1 to renumber the program starting with line 1 and incrementing by 1 to reduce the total size of the program.

If you have 16K Extended BASIC and PRINT MEM, you get 8487 bytes free. Type in PCLEAR 3, PCLEAR 2, or PCLEAR 1 to

get more memory (the default value on power-up is PCLEAR 4). PCLEAR 1 will yield 13,095 bytes free. The command PCLEAR 0 is not accepted, but you can accomplish the same thing by typing POKE 25,6:NEW (ENTER). You'll have 14,631 bytes to work with.

The computer automatically reserves about 200 characters for working with strings. To reserve more or less space, use the CLEAR *nnnn* command. Examples:

```
100 CLEAR 500 (for more strings)
100 CLEAR 100 (for fewer strings, more memory)
```

To save memory, you may leave out spaces in your commands. For example:

```
100 FOR A = 1 TO 10  may be changed to
100FORA = 1TO10
```

A space is required between a variable name and a following BASIC word. Example:

```
100 FOR A = B TO C  may be changed to
100FORA = B TOC
```

Keep in mind that subscripts in an array start with the *zero* element. DIM A(5) actually reserves six elements for A: A(0), A(1), A(2), A(3), A(4), and A(5).

If you use a subscripted variable or an array without a DIMENSION statement, the computer automatically reserves 11 elements — up to A(10). If you require fewer elements, use the DIMENSION statement to reserve only the necessary memory. Example:

```
100 DIM B(3)
```

To save memory, keep variable names short. Although the computer accepts longer names, it recognizes only the first two letters. In a critical memory program, use one-letter variable names.

## Programming Ideas

As you are developing a program, increment your line numbers by at least 10 so you can easily insert lines. With Extended BASIC, you may renumber the lines later with the RENUM command.

You may use a variable name longer than two characters, but only the first two characters are recognized. The variable names BLACK and BLUE would both be recognized as the variable BL.

## Programming Hints and Tips

---

When you first turn on the computer, all numeric variables are zero. Each time you RUN a program, however, the variables start with the value which is there. NEW returns all numeric variables to zero, but not all the values previously POKEd into memory. NEW does not reset PMODE, PCLEAR, or CLEAR values set by a previous program. Default (power-up) values are:

```
PMODE 2,1
PCLEAR 4
CLEAR 200 or
CLEAR 200,16383 for 16K
CLEAR 200,32767 for 32K
```

A string must have fewer than 256 characters.

To really randomize numbers when you use the RND function, you should first have the statement  $X = \text{RND}(-\text{TIMER})$ .

Example:

```
100 R = RND(-TIMER)
110 R = RND(6)
```

In the DRAW statement, any diagonal distance specified is actually the distance of the diagonal of a square whose sides are the specified distance — the distance is  $\sqrt{2}$  times the specified distance.

To use variables in a DRAW or PLAY statement, put an equal sign before the variable name and a semicolon after the variable name. The variable name may be subscripted. The variable must represent a positive number. Examples:

```
100 PMODE 3:PCLS:SCREEN 1,0
110 FOR X=20 TO 220 STEP 40
120 FOR Y=20 TO 190 STEP 30
130 DRAW "BM=X;,=Y;U1OR15F10G10L15U10"
140 NEXT Y
150 NEXT X
160 GOTO 160

100 FOR X=4 TO 50 STEP 2
110 PLAY "L=X;CDE"
120 NEXT X

100 FOR X=31 TO 1 STEP -1
110 PLAY "L100;V=X;A"
120 NEXT X
```

To get interesting sound effects, use very short durations. In a PLAY statement, the shortest duration is L255. Try varying the volume or the tone in a FOR-NEXT statement. Examples:

### **varying volume**

```
110 PLAY "L255;O3"  
110 FOR V = 31 TO 1 STEP -1  
120 A$ = "V" + STR$(V)  
130 PLAY "XA$;D"  
140 NEXT V
```

### **100 random tones in lowest octave**

```
100 PLAY "L255;O1"  
110 FOR I = 1 TO 100  
120 A$ = STR$(RND(12))  
130 PLAY A$  
140 NEXT I
```

### **varying tone — played five times**

```
100 PLAY "L255;O3"  
110 FOR I = 1 TO 5  
120 FOR N = 1 TO 12  
130 A$ = STR$(N)  
140 PLAY A$  
150 NEXT N  
160 NEXT I
```

With this sample program, try different octaves by changing the O3 in line 100.

Try changing line 120 to FOR N = 12 TO 1 STEP -1.





**Three**

---

**Programs**



## Utility Programs

These programs can assist you in your own programming. "Carriage Return — Line Feed" and "POKEs for Teletype" may be useful if you have a printer other than Radio Shack. Since the TRS-80 Color Computer has a standard RS-232 interface built in, you can connect a different type of printer to your computer. However, some of the printing features are built into the computer to work with a Radio Shack printer.

Carriage Return — Line Feed will assist you in LISTing a program — after each return at the end of the line, you need a line feed to move the paper upward. POKEs for Teletype is used to print at a baud rate of 110. The built-in rate is 600.

"Letters" and "Sorts" are programs you may wish to use within your own programs. Letters offer a way to *draw* the letters on a graphics screen. The sort routines are used to alphabetize lists or arrange data in numeric sequence.

### Carriage Return — Line Feed

The Color Computer does *not* automatically issue a line feed command with a carriage return when you are using a printer. If you use a Radio Shack printer, there will be no problem. If you use another type of printer, all your printing will be on one line. The following program from Radio Shack will POKE values into memory locations so a line feed will occur with each carriage return.

```
100 CLS
110 DATA 52,20,214,111,193,254
120 DATA 38,11,129,13,38,7,190
130 DATA 160,2,173,3,134,10,53
140 DATA 20,57
150 FOR D=1000 TO 1021
160 READ E
170 POKE D,E
180 NEXT D
```

## Programs

---

```
190 POKE 1021,PEEK(359)
200 POKE 1022,PEEK(360)
210 POKE 1023,PEEK(361)
220 POKE 359,126
230 POKE 360,3
240 POKE 361,232
250 END
```

Type the program in (or CLOAD it from a previously saved tape). RUN. Be sure your next command is NEW to get rid of this program. Do not RUN it again. You also do not want segments of it to RUN again.

This program may cause machine lock-up problems on certain functions. You cannot BREAK or RESET. You must turn off the computer and start over. Problems may occur with PRINT LOG(1), PRINT USING, and evaluating any functions with very large numbers.

## POKEs for Teletype

To print at a baud rate of 110 (such as with a teletype) instead of the standard baud rate of 600, use the following procedure:

```
POKE 149,1
POKE 150,255
POKE 151,64
POKE 152,0
```

## Letters

If you are drawing graphics on medium or high-resolution graphics screens, you cannot simply PRINT messages, too. There are several ways to DRAW the letters on the screen. This program segment offers one way to define some letters to be drawn.

Line Numbers	Explanation
100	DIMensions L\$ array for 26 letters plus a space. If you want to include more symbols, adjust the 26. To save memory, include only the letters you will actually use in the program.
110-130	READ from DATA first the symbol or letter, then the drawing instruction.
400-600	A sample program segment showing how you would call the subroutine. PMODE 4 is the high-resolution graphics screen. DRAW "S8" indicates

- the scale — a small letter. After you run this program as is, try DRAW "S4". A message or word is defined in M\$. The next statement specifies the starting position of the lower-left corner of the message.
- 1000-1060 Subroutine to draw the message. For 1 to the length of the message or word, the computer compares the L\$ array elements to each letter of the message and draws the appropriate letter.
- 2000-2260 DATA with the drawing instructions. You may wish to change the design of the letters or put more space between letters.

### Program 1. Letters

```

100 DIM L$(26,1)
110 FOR I=0 TO 26
120 READ L$(I,0),L$(I,1)
130 NEXT I
400 PMODE 4,1:PCLS:SCREEN 1,0
450 DRAW "SB"
500 M$="ABCDEFGH IJKLM"
510 DRAW "BM10,30;"
520 GOSUB 1000
530 M$="NOPQR STUVWXYZ"
540 DRAW "BM30,70;"
550 GOSUB 1000
600 GOTO 600
1000 FOR L=1 TO LEN(M$)
1010 FOR I=0 TO 26
1020 IF L$(I,0)=MID$(M$,L,1) THEN 1040
1030 NEXT I
1040 DRAW L$(I,1)
1050 NEXT L
1060 RETURN
2000 DATA " ", "BM+7,0"
2010 DATA A, "U4E2F2D2NL4D2;BM+4,0"
2020 DATA B, "U6R3FDGNL3FDGL3;BM+7,0"
2030 DATA C, "BM+2,0;H1U4E1R2F1;BM+0,+4;G1L2;
BM+6,0"
2040 DATA D, "U6R3F1D4G1L3;BM+7,0"
2050 DATA E, "U6NR3D3NR2D3R3;BM+3,0"
2060 DATA F, "U3NR2U3R4;BM+3,6"
2070 DATA G, "BM+1,0;H1U4E1R2F1;BM+0,2;NL1D2G
1L2;BM+6,0"
2080 DATA H, "U3NU3R4NU3D3;BM+3,0"
2090 DATA I, "BM+3,0;NU6;BM+4,0"
2100 DATA J, "BM+0,-1;F1R1E1NU5;BM+4,1"

```

```
2110 DATA K, "U3NU3R1NE3F3; BM+3, 0"
2120 DATA L, "NU6R4; BM+3, 0"
2130 DATA M, "U6F2ND1E2D6; BM+3, 0"
2140 DATA N, "U6F1D1F2D1F1NU6; BM+3, 0"
2150 DATA O, "BM+2, 0; H1U4E1R2F1D4G1L2; BM+6, 0"
2160 DATA P, "U6R3F1D1G1L3; BM+7, 3"
2170 DATA Q, "BM+2, 0; H1U4E1R2F1D3G1NH1NF1G1L1
    ; BM+6, 0"
2180 DATA R, "U6R3F1D1G1L3R1F3; BM+3, 0"
2190 DATA S, "BM+0, -1; F1R2E1U1H1L2H1U1E1R2F1;
    BM+3, +5"
2200 DATA T, "BM+3, 0; U6NL3R3; BM+3, 6"
2210 DATA U, "BM+0, -1; NU5F1R2E1NU5; BM+3, 1"
2220 DATA V, "BM+0, -6; D2F1D1F1ND1E1U1E1U2; BM+
    3, 6"
2230 DATA W, "NU6E2NU1F2U6; BM+3, 6"
2240 DATA X, "U1E4U1; BL4D1F4D1; BM+3, 0"
2250 DATA Y, "BM+3, 0; U3H2U1D1F2E2U1; BM+3, 6"
2260 DATA Z, "NR4U1E4U1NL4; BM+3, 6"
2270 END
```

## Sorts

One of the functions of a computer is to organize data. There are many kinds of sort routines or algorithms to arrange your data. You may want to arrange scores in numerical order, sort names by birth date, or alphabetize names. Here are four different sort routines written in BASIC. Ordinarily you would have a list of data. To demonstrate these programs, lines 100-120 randomly choose 50 integers from 1 to 100. Lines 200-300 are the sort routine. Lines 500 to the end print the results in sorted order.

To alphabetize, change the variable names such as A(I) to A\$(I), string variables. These sorts are for ascending order. To change to descending order, change all less than (<) signs in the sort routine to greater than (>) signs.

"Inter Sort" is a simple interchange. Each item is compared to the next one. If they are out of order, they are interchanged. This routine slows down with more numbers or with mixed-up numbers.

"Shell Sort" is considerably faster than Inter Sort because the number of comparisons that need to be made is reduced.

"Min Sort" and "Min/Max Sort" require a certain number of "passes" no matter how mixed up the numbers are. Min Sort finds the minimum number with each pass and puts it at the end. Min/Max Sort finds both the maximum and minimum with each pass and places them at the endpoints in order.

### Program 2. Inter Sort

```

100 REM INTERCHANGE SORT
110 DIM A(50)
120 FOR I=1 TO 50:A(I)=RND(100):PRINTA(I);:N
    EXT:PRINT:PRINT
200 L=49
210 S=0:FOR I=1 TO L:IF A(I)<=A(I+1)THEN 230
220 AA=A(I):A(I)=A(I+1):A(I+1)=AA:S=1:L=I
230 NEXT
240 IF S=1 THEN 210
500 FOR I=1 TO 50:PRINTA(I);:NEXT
510 END

```

### Program 3. Shell Sort

```

100 REM SHELL SORT
110 DIM A(50)
120 FOR I=1 TO 50:A(I)=RND(100):PRINTA(I);:N
    EXT:PRINT:PRINT
200 B=1
210 B=2*B:IF B<=50 THEN 210
220 B=INT(B/2):IF B=0 THEN 500
230 FOR I=1 TO 50-B:C=I
240 D=C+B:IF A(C)<=A(D)THEN 260
250 AA=A(C):A(C)=A(D):A(D)=AA:C=C-B:IF C>0 T
    HEN 240
260 NEXT:GOTO 220
500 FOR I=1 TO 50:PRINTA(I);:NEXT
510 END

```

### Program 4. Min Sort

```

100 REM SORT C
110 DIM A(50):N=50
120 FOR I=1 TO 50:A(I)=RND(100):PRINTA(I);:N
    EXT:PRINT:PRINT
200 M=A(1):IM=1
210 FOR I=2 TO N
220 IF A(I)>M THEN M=A(I):IM=I
230 NEXT
240 AA=A(N):A(N)=A(IM):A(IM)=AA:N=N-1:IF N>1
    THEN 200
500 FOR I=1 TO 50:PRINTA(I);:NEXT
510 END

```

### Program 5. Min/Max Sort

```

100 REM SORT D
110 DIM A(50):N=50:S=1
120 FOR I=1 TO 50:A(I)=RND(100):PRINTA(I);:N
    EXT:PRINT:PRINT

```

```
200 MN=A(S):IM=S:MX=MN:IX=S
210 FOR I=S TO N
220 IF A(I)>MX THEN MX=A(I):IX=I
230 IF A(I)<MN THEN MN=A(I):IM=I
240 NEXT
250 IF IM=N THEN IM=IX
260 AA=A(N):A(N)=A(IX):A(IX)=AA:N=N-1
270 AA=A(S):A(S)=A(IM):A(IM)=AA:S=S+1
280 IF N>S THEN 200
500 FOR I=1 TO 50:PRINTA(I);:NEXT
510 END
```

## Graphics

Graphics (drawing) and music are what make a home computer *fun*. Many people learn programming by first trying to draw pictures on the screen. The programs in this section illustrate some of the graphics possible on the Color Computer. "Flag" and "William Shakespeare" also have music. William Shakespeare shows that you can make a rather detailed drawing on the screen. Take a look at it — then draw your own!

"Circles," "Lines," and "Boxes" are some short programs using graphics. Try changing some of the variables or limits in the programs for slightly different effects.

"Dice" illustrates one way you can draw dice, where the dice numbers have been chosen randomly by the computer. Use this concept to design your own dice game.

### Flag

This program illustrates a quick way to draw stripes and then a blue field for a flag by using the Box Filled option of the LINE command in Extended BASIC. Lines 150-220 draw the alternating stripes by changing colors with the COLOR command. Lines 230-240 draw the blue field.

A star shape is drawn with the string A\$ defined in line 260. Lines 270-390 place the star at several locations.

Lines 400-450 play music. V changes the volume level of the music.

Line 460 holds the picture on the screen. Press BREAK to stop the program.



**Program 6. Flag**

```

100 REM FLAG
110 REM
120 PMODE 3,1
130 PCLS
140 SCREEN 1,0
150 FOR I=0 TO 150 STEP 30
160 COLOR 4,1
170 LINE(0,I)-(255,I+14),PSET,BF
180 COLOR 2,1
190 LINE(0,I+15)-(255,I+28),PSET,BF
200 NEXT I
210 COLOR 4,1
220 LINE(0,180)-(255,191),PSET,BF
230 COLOR 3,1
240 LINE(0,0)-(104,89),PSET,BF
250 COLOR 2,1
260 A$="D3L1R3L1D1G2E2F2"
270 DRAW"BM52,4;XA$;"
280 DRAW"BM74,10;XA$;"
290 DRAW"BM86,26;XA$;"
300 DRAW"BM90,42;XA$;"
310 DRAW"BM86,58;XA$;"
320 DRAW"BM76,69;XA$;"
330 DRAW"BM62,77;XA$;"
340 DRAW"BM44,77;XA$;"
350 DRAW"BM30,69;XA$;"
360 DRAW"BM20,58;XA$;"
370 DRAW"BM14,42;XA$;"
380 DRAW"BM20,26;XA$;"
390 DRAW"BM32,10;XA$;"
400 PLAY "V12;L8;02;FD;L4;01;B-;02;DF;L2;B-;
L8;03;DC;L4;02;B-DE;L2;F"
410 PLAY "V16;L8;FF;L4.;03;D;L8;C;L4;02;B-;L
2;A;L8;GA;L4;B-B-FD;01;B-;L8;02;FD;L4;01
;B-;02;DF;L2;B-;L8;03;DC;L4;02;B-DE;L2;F
"
420 PLAY "V24;L8;FF;L4.;03;D;L8;C;L4;02;B-;L
2;A;L8;GA;L4;B-B-FD;01;B-"
430 PLAY "V10;L8;03;DD;L4;DE-F;L2;F;L8;E-D;L
4;CDE-;L2;E-;L4;E-;L4.;D;L8;C;L4;02;B-;L
2;A;L8;GA;L4;B-DE;L2;F"
440 PLAY "V15;L4;FB-B-;L8;B-A;L4;GGG;03;C;L8
;E-DC;02;B-;L4;B-L4.;A"
450 PLAY "V25;L8;FF;L3;B-;L8;03;CDE-;L2;F;L8
;02;B-;03;C;L2;D;L8;E-;L4;C;L1;02;B-"
460 GOTO 460
470 END

```

### William Shakespeare

This is a high-resolution, graphics-demonstration program which uses repetitive lines and partial circles. The CIRCLE command first specifies the location of the center point of the circle in x and y coordinates of the high-resolution screen. The next parameter is the radius, or how big the circle will be. Other parameters are optional. In order, they are color, height-width ratio, starting point, and ending point. The height-width ratio can make an oval shape. The starting and ending points are specified in fractions.

#### Line Numbers Explanation

130-140	Specify high-resolution graphics, clear the screen with buff, and specify black lines.
150	Draw the head.
160-190	Draw eyes.
200-310	Draw other facial features.
320-440	Draw hair.
450-480	Draw collar.
490-790	Draw jacket.
800-880	Play music.
890	Hold picture on screen; press BREAK to stop program.
900	End.

#### Program 7. William Shakespeare

```
100 REM WILLIAM SHAKESPEARE
110 REM EXCERPT FROM ROMEO AND JULIET
120 REM
130 PMODE 4,1:PCLS 5:SCREEN1,1
140 COLOR 0
150 CIRCLE(128,64),30,,1.5
160 FOR I=110 TO 142 STEP 32
170 CIRCLE(I,60),8,0,.5
180 CIRCLE(I+4,60),3,0
190 PAINT(I+4,60),0,0:NEXT I
200 CIRCLE(110,56),8,0,.5,.6,1
210 CIRCLE(142,56),8,2,.5,.5,1
220 COLOR 0,1:LINE(124,60)-(116,76),PSET
230 CIRCLE(118,78),4,0,1,.1,.6
240 CIRCLE(124,80),3,0,1,.5,1
250 CIRCLE(130,78),3,0,1,.75,.5
260 DRAW "BM126,90;E4R4F2R4"
270 DRAW "BM126,90;H4L46L26L2"
```

```
280 CIRCLE(126,90),8,0,.5,1,.6
290 FOR I=132 TO 135
300 CIRCLE(I,78),8,0,1,.25,.5
310 CIRCLE(I-20,78),8,0,1,.05,.25:NEXT I
320 A$="68D368D8F4R3E3"
330 DRAW "BM98,50;N;XA$;"
340 DRAW "BM98,46;N;XA$;"
350 DRAW "BM100,42;N;XA$;"
360 A$="F8D3F8D6G4L2H2"
370 DRAW "BM158,56;NXA$;"
380 DRAW "S3BM157,60;NXA$;"
390 DRAW "BM156,63;NXA$;"
400 DRAW "S2;BM155,65;NXA$;"
410 DRAW "S4;BM158,52;NXA$;"
420 DRAW "BM158,48;NXA$;"
430 DRAW"BM156,44;F4;NXA$;"
440 DRAW "BM154,40;F8;NXA$;"
450 LINE(98,78)-(40,140),PSET
460 CIRCLE(128,172),120,0,.4,.62,.84
470 CIRCLE(170,100),30,2,1,.75,.08
480 LINE(197,110)-(184,134),PSET
490 LINE(196,108)-(255,120),PSET
500 CIRCLE(255,191),35,0,2,.5,.75
510 LINE(68,110)-(0,120),PSET
520 LINE-(15,191),PSET
530 LINE(116,126)-(102,191),PSET
540 LINE(136,126)-(122,191),PSET
550 J=126
560 FOR I=134 TO 184 STEP 8
570 CIRCLE(J,I),2
580 J=J-2:NEXT I
590 FOR I=255 TO 246 STEP -3
600 CIRCLE(I,191),35,0,2,.5,.75
610 NEXT I
620 J=110:K=60
630 FOR I=196 TO 192 STEP -1
640 LINE(I,J)-(I+K,J+10),PSET
650 J=J+2:K=K-5
660 NEXT I
670 J=112:K=68:L=18
680 FOR I=68 TO 60 STEP -2
690 LINE(I,J)-(I-K,J+10),PSET
700 LINE -(L,191),PSET
710 K=K-5:J=J+2:L=L+2
720 NEXT I
730 LINE(90,128)-(110,146),PSET
740 LINE(162,128)-(134,148),PSET
750 LINE(114,126)-(100,191),PSET
760 LINE(138,126)-(124,191),PSET
```

## Programs

---

```
770 LINE(112,128)-(98,191),PSET
780 LINE(140,128)-(126,191),PSET
790 LINE(0,0)-(255,191),PSET,B
800 A$="02G#03DP6402G#03DP6402G#03FEDP16"
810 B$="L403C02BP6403C02BP6403C02L2GP16"
820 C$="L203F#L2.GP32L402B03C02AGL103CP16"
830 D$="L402ABL2G#03L2.EP32L402GAA#EFL1G#L2G
    L1CP16"
840 PLAY C$
850 PLAY D$
860 PLAY "L4;XA$;XA$;XB$;XB$;L16GAB03CDE"
870 PLAY C$
880 PLAY D$
890 GOTO 890
900 END
```

## Circles

Concentric circles are drawn in high resolution on a black screen. With varying radii, different color patterns develop. First, the circles are drawn close together, then farther and farther apart, then closer and closer together. The pattern repeats. Press BREAK to stop the program.

### Program 8. Circles

```
100 REM CIRCLES
110 REM
120 PMODE 4,1
130 N=1:A=2:B=6
140 FOR J=A TO B STEP N
150 PCLS
160 SCREEN 1,1
170 FOR I=2 TO 95 STEP J
180 CIRCLE(128,96),I
190 NEXT I
200 NEXT J
210 N=-N:D=A:A=B:B=D
220 GOTO 140
230 END
```

## Lines

Here is a program that draws a pattern of lines in a random fashion. This is a high-resolution screen. Notice the color patterns that develop depending on the distance between the lines.

Two points are picked randomly, then a line is drawn between the points. Four random distances are chosen (S1, S2, S3, S4). The end points of the line are changed by the four S factors and a new line is drawn. Lines are drawn continuously until a line hits a border, then the direction is changed. After 200 lines are drawn, the screen clears and another random pattern develops.

### Program 9. Lines

```

100 REM LINES
110 REM
120 PMODE 4,1
130 PCLS
140 SCREEN 1,1
150 X=RND(-TIMER)
160 PCLS
170 S1=RND(10):S2=RND(5)
180 S3=RND(10):S4=RND(5)
190 X1=RND(255)
200 Y1=RND(191)
210 X2=RND(255)
220 IF X2=X1 THEN 210
230 Y2=RND(191)
240 IF Y2=Y1 THEN 230
250 LINE(X1,Y1)-(X2,Y2),PSET
260 T=T+1:IF T=200 THEN T=0:GOTO 160
270 X1=X1+S1:Y1=Y1+S2
280 X2=X2+S3:Y2=Y2+S4
290 IF X1>255 THEN X1=255:S1=-S1
300 IF X1<0 THEN X1=0:S1=-S1
310 IF Y1>191 THEN Y1=191:S2=-S2
320 IF Y1<0 THEN Y1=0:S2=-S2
330 IF X2>255 THEN X2=255:S3=-S3
340 IF X2<0 THEN X2=0:S3=-S3
350 IF Y2>191 THEN Y2=191:S4=-S4
360 IF Y2<0 THEN Y2=0:S4=-S4
370 GOTO 250
380 END

```

### Boxes

"Boxes" illustrates the Box Filled option of the LINE command by painting random boxes on the screen. Press BREAK to stop the program.

### Program 10. Boxes

```

100 REM BOXES

```

```
110 REM
120 X=RND(-TIMER)
130 PMODE 3,1
140 PCLS
150 SCREEN 1,0
160 COLOR RND(4)
170 LINE (RND(255),RND(191))-(RND(255),RND(191)),PSET,BF
180 GOTO 160
190 END
```

### Dice

"Dice" is a program segment that shows how dice can be drawn in low-resolution graphics using the predefined graphics characters. For this example, three dice are drawn. The number of dots is chosen randomly (line 190). Depending on the number of dots, a particular subroutine will draw the dots (line 200). RUN the program several times to see different dice drawn.

### Program 11. Dice

```
100 REM DICE
110 CLS
120 A$=CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+"
    {3 SPACES}"
130 FOR A=35 TO 131 STEP 32
140 PRINT @A,A$+A$+A$
150 NEXT A
160 A$=CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+"
    {3 SPACES}"
170 PRINT @A,A$+A$+A$
180 FOR I=1 TO 3
190 D=RND(6)
200 ON D GOSUB 230,270,340,370,440,470
210 NEXT I
220 STOP
230 X=12+(I-1)*20
240 Y=6
250 GOSUB 610
260 RETURN
270 X=8+(I-1)*20
280 Y=4
290 GOSUB 610
300 X=X+8
310 Y=8
```

```
320 GOSUB 610
330 RETURN
340 GOSUB 230
350 GOSUB 270
360 RETURN
370 GOSUB 270
380 Y=4
390 GOSUB 610
400 X=X-8
410 Y=8
420 GOSUB 610
430 RETURN
440 GOSUB 370
450 GOSUB 230
460 RETURN
470 X=9+(I-1)*20
480 Y=4
490 GOSUB 610
500 X=X+6
510 GOSUB 610
520 Y=6
530 GOSUB 610
540 Y=8
550 GOSUB 610
560 X=X-6
570 GOSUB 610
580 Y=6
590 GOSUB 610
600 RETURN
610 SET(X,Y,2)
620 SET(X+1,Y,2)
630 RETURN
640 END
```

## Mathematics

One of the main functions of a computer is to take the drudgery (and time) out of calculations. These programs involve mathematical principles. You input the numbers and the computer delivers the answers. The computer can go through an algorithm very quickly to get the solution.

Another application is to put any mathematical formula into an interactive program where the user needs to enter some of the numbers. The computer can give the answer quickly. Some examples are: distance = rate multiplied by time; conversion from

Fahrenheit degrees to Centigrade or Celsius degrees; calculation of interest for savings or loans; solving economics formulas; or designing electric circuits.

### Homework Helper — Division

“Homework Helper — Division” is designed to quickly give answers to students checking their homework assignments. “Division” gives the answers to three types of homework problems an elementary-school student may encounter: division with a remainder, division with a decimal in the quotient, and division in fraction form of a numerator divided by the denominator.

Only the answers are given, not the step-by-step process of long division. The student is encouraged to do his homework, writing each step in the division process, and then to use this program to check his answers. Music and graphics are used to enhance the program.

1. Division with Remainder. Most math problems can simply be corrected with a calculator. If there is a remainder, however, a calculator converts it to a decimal equivalent. When students first learn division, they learn to divide and specify a remainder if the number does not evenly divide. This program keeps the answer in quotient plus remainder form. The student enters the divisor and dividend, and the quotient and remainder are printed.

2. Division with Decimal. Usually after the student masters the idea of a remainder, he is taught how to place a decimal and keep dividing. In this section the student enters the divisor and dividend, and the quotient with a decimal fraction is printed.

3. Convert Fraction to Decimal. A fraction is converted to a decimal by dividing the numerator by the denominator. The student enters the numerator, then the denominator, and the equivalent decimal fraction is returned.

After each problem, the student may enter another problem of the same type. If he has no more problems of the same kind or wishes to end, he enters zero and the menu screen will return.

#### Line Numbers Explanation

120	Specify mode and screen for graphics, clear screen.
130-400	Draw colored graphics.
410-510	Draw “DIVISION” title.
520	Play music.
530-580	Print menu screen of choices.



590-610	Wait for student to press number; branch appropriately.
620-670	Print instructions and graphics for division with remainder.
680-730	Receive input for student's problem.
740-750	Print answer.
760-800	Wait for student to press a key before going to next problem.
810-860	Print instructions and graphics for division with decimal.
870-920	Receive input for student's problem.
930	Print answer.
940-990	Wait for student to press a key before going to next problem.
1000-1050	Print instructions and graphics for converting a fraction to a decimal number.
1060-1110	Receive input for student's problem.
1120	Print answer.
1130-1170	Wait for student to press a key before going to next problem.
1180	End.

### Program 12. Homework Helper — Division

```

100 REM HOMEWORK HELPER--DIVISION
110 REM
120 PMODE 3,1:SCREEN 1,1:PCLS
130 CIRCLE(56,108),16,4
140 PAINT (56,108),4,4
150 J=106
160 FOR I=140 TO 122 STEP -2
170 PSET(I,J,4):PSET(I,J+1,4)
180 J=J+2:NEXT I
190 FOR J=116 TO 120
200 LINE(132,J)-(148,J+16),PSET
210 NEXT J
220 COLOR 3,1
230 DRAW "BM130,115;H44D9F34R10"
240 LINE(160,100)-(166,101),PSET,BF
250 LINE(160,104)-(166,105),PSET,BF
260 FOR J=90 TO 126 STEP 12
270 FOR I=182 TO 222 STEP 8
280 CIRCLE(I,J),2,4
290 NEXT I,J
300 FOR I=182 TO 206 STEP 8
310 CIRCLE(I,138),2,4

```

## Programs

---

```
320 NEXT I
330 LINE(80,28)-(94,39),PSET,BF
340 A$="D4F2D4F2D1262D462D4"
350 DRAW "BM112,20;" + A$
360 DRAW "BM110,20;" + A$
370 DRAW "BM112,19;R64"
380 DRAW "BM112,18;R64"
390 LINE(132,28)-(170,39),PSET,BF
400 LINE(132,4)-(150,11),PSET,BF
410 DRAW "BM38,160;D27R13E4U20H4L13"
420 LINE(64,160)-(64,187),PSET
430 LINE(78,160)-(90,187),PSET
440 LINE(90,187)-(102,160),PSET
450 LINE(114,160)-(114,187),PSET
460 DRAW "BM140,164;U1H4L8G4D5F4R8F4D7G4L8H4
U2"
470 LINE(152,160)-(152,187),PSET
480 DRAW "BM178,160;L10G6D16F6R10E6U16H6"
490 LINE(192,160)-(192,187),PSET
500 LINE(192,160)-(210,187),PSET
510 LINE(210,160)-(210,187),PSET
520 PLAY "L8;O2;CFG A;O3;C;O2;AGFG;L4.;A;L8O3
;CL4.F;L8C;L4.D;L8DL4.FL8DL4CL202A"
530 SCREEN 0,0:CLS
540 PRINT @65,"CHOOSE:"
550 PRINT @161,"1 DIVISION WITH REMAINDER"
560 PRINT @225,"2 DIVISION WITH DECIMAL"
570 PRINT @289,"3 CONVERT FRACTION TO DECIMA
L"
580 PRINT @353,"4 END PROGRAM"
590 A$=INKEY$:IF A$="" THEN 590
600 IF ASC(A$)<49 OR ASC(A$)>52 THEN 590
610 CLS:ON VAL(A$) GOTO 620,810,1000,1180
620 PRINT @164,"DIVISOR "+"CHR$(175)+" DEVI
DEND"
630 PRINT @141,CHR$(175):PRINT @205,CHR$(175
)
640 FOR I=24 TO 55:FOR J=6 TO 7
650 SET(I,J,3):NEXT J,I
660 PRINT @48,"QUOTIENT"
670 PRINT @256,"ENTER '0' TO STOP."
680 PRINT @320,"DIVISOR ";
690 INPUT D
700 IF D=0 THEN 530
710 PRINT @352,"DIVIDEND ";
720 INPUT N
730 IF N=0 THEN 530
740 C=INT(N/D):R=N-C*D
750 PRINT @416,"QUOTIENT = ";C;" R";R
```

```

760 PRINT @480,"PRESS ANY KEY.";
770 A$=INKEY$:IF A$="" THEN 770
780 PRINT @320,"":PRINT @352,"":PRINT @384,"
":PRINT @416,""
790 PRINT @448,"":PRINT @480,"{(14 SPACES)}";
800 GOTO 680
810 PRINT @164,"DIVISOR "+CHR$(191)+" DIVID
DEND"
820 PRINT @141,CHR$(191):PRINT @205,CHR$(191
)
830 FOR I=24 TO 55:FOR J=6 TO 7
840 SET(I,J,4):NEXT J,I
850 PRINT @48,"QUOTIENT . ###"
860 PRINT @256,"ENTER '0' TO STOP."
870 PRINT @320,"DIVISOR ";
880 INPUT D
890 IF D=0 THEN 530
900 PRINT @352,"DIVIDEND ";
910 INPUT N
920 IF N=0 THEN 530
930 PRINT @416,"QUOTIENT = ";N/D
940 PRINT @480,"PRESS ANY KEY.";
950 A$=INKEY$:IF A$="" THEN 950
960 PRINT @320,"":PRINT @352,"":PRINT @384,"
"
970 PRINT @416,"":PRINT @448,""
980 PRINT @480,"{(14 SPACES)}";
990 GOTO 870
1000 PRINT @68,"NUMERATOR"
1010 FOR I=4 TO 29:FOR J=6 TO 7
1020 SET(I,J,3):NEXT J,I
1030 PRINT @131,"DENOMINATOR"
1040 PRINT @113,"={3 SPACES}#.###"
1050 PRINT @192,"ENTER '0' TO STOP."
1060 PRINT @288,"NUMERATOR ";
1070 INPUT N
1080 IF N=0 THEN 530
1090 PRINT @320,"DENOMINATOR";
1100 INPUT D
1110 IF D=0 THEN 530
1120 PRINT @384,"DECIMAL EQUIVALENT =";N/D
1130 PRINT @480,"PRESS ANY KEY.";
1140 A$=INKEY$:IF A$="" THEN 1140
1150 PRINT @288,"":PRINT @320,"":PRINT @352,
"":PRINT @384,""
1160 PRINT @416,"":PRINT @448,"":PRINT @480,
" {(14 SPACES)}";
1170 GOTO 1050
1180 END

```

### Find All Factors

This program will find all the factors of a number that you enter. For example, the number 12 has the factors 12, 6, 4, 3, 2, and 1. You must enter a number greater than 1. If you enter a very large number, the computer may take a long time. To stop the program, enter zero.

#### Program 13. Find All Factors

```
100 REM FIND ALL FACTORS
110 REM
120 CLS:PRINT @35,"FINDING ALL THE FACTORS"
130 A$=CHR$(175)+CHR$(175)+CHR$(175)+CHR$(175)
140 B$=CHR$(133)+CHR$(133)+CHR$(133)+CHR$(133)
150 PRINT @135,A$+" = "+B$
160 A$=A$+"(5 SPACES)+"B$
170 PRINT @103,A$
180 PRINT @167,A$
190 PRINT @199,A$
200 PRINT @288,"ENTER '0' TO STOP."
210 PRINT @352,"WHAT IS THE NUMBER TO FACTOR
?"
220 INPUT A:IF A=0 THEN 370
230 IF A>1 THEN 250
240 PRINT @416,"PLEASE ENTER A NUMBER > 1.";
:GOTO 220
250 PRINT "FACTORS OF ";A;"ARE ":PRINT A;
260 B=INT(A/2+1)
270 FOR C=2 TO B
280 IF A/C<>INT(A/C) THEN 320
290 B=A/C:PRINT B;
300 IF B=1 THEN 340
310 IF B=2 THEN 330
320 NEXT C
330 PRINT " 1"
340 PRINT:PRINT "PRESS ANY KEY TO CONTINUE."
350 G$=INKEY$:IF G$="" THEN 350
360 GOTO 120
370 CLS:END
```

### Prime Factors

"Prime Factors" returns a list of the prime factors of a number you enter. Another term for this process is *complete factorization*. All prime factors multiplied together will yield the original number. For example, the prime factors of the number 12 are 2, 2, 3.

You must enter a number greater than 1. Enter 0 to stop the program.

### Program 14. Prime Factors

```

100 REM PRIME FACTORS
110 REM
120 CLS:PRINT @8,"PRIME FACTORS OR"
130 PRINT @69,"COMPLETE FACTORIZATION"
140 FOR A=12 TO 23
150 FOR B=8 TO 13
160 SET(A,B,4)
170 NEXT B,A
180 PRINT @174,"= "+CHR$(175)+" * "+CHR$(17
5)+CHR$(175)+" * "+CHR$(175)+CHR$(175)
190 PRINT @256,"ENTER '0' TO STOP."
200 PRINT @320,"WHAT IS THE NUMBER TO FACTOR
?"
210 INPUT F:IF F=0 THEN 390
220 IF F>1 THEN 250
230 PRINT "PLEASE ENTER A NUMBER > 1."
240 GOTO 210
250 PRINT:PRINT "THE PRIME FACTORS ARE: "
260 G=INT(F/2)
270 FOR I=2 TO G
280 IF F/I<>INT(F/I) THEN 320
290 F=F/I:G=F
300 PRINT I;
310 GOTO 270
320 NEXT I
330 IF F=1 THEN 350
340 PRINT F
350 PRINT:PRINT
360 PRINT "PRESS ANY KEY TO CONTINUE.";
370 G$=INKEY$:IF G$="" THEN 370
380 GOTO 120
390 CLS:END

```

### Greatest Common Factor

You may enter two numbers, and this program will find their greatest common factor. This mathematical concept is usually introduced prior to simplifying fractions. For example, if the two numbers entered are 12 and 18, the greatest common factor is 6. Both numbers can be evenly divided by 6. Both can also be divided by 3 or 2, but 6 is the largest factor.

### Program 15. Greatest Common Factor

```
100 REM GREATEST COMMON FACTOR
110 REM
120 CLS:PRINT @2,"GREATEST COMMON FACTOR"
130 PRINT @34,"OF TWO NUMBERS"
140 G$=CHR$(175)+CHR$(175)+CHR$(175)
150 H$=G$+CHR$(175)+CHR$(159)+" "+G$
160 G$=CHR$(159)+CHR$(159)+CHR$(159)
170 PRINT @99,H$
180 PRINT @131,H$
190 PRINT @138,G$+" ... "+G$
200 PRINT @163,H$
210 PRINT @224,"ENTER '0' TO STOP."
220 PRINT @288,"FIRST NUMBER ";
230 INPUT M
240 IF M=0 THEN 550
250 IF M>1 THEN 280
260 PRINT "SORRY, ENTER NUMBERS > 1."
270 GOTO 230
280 IF M<10000 THEN 310
290 PRINT "SORRY, MUST BE LESS THAN 10000."
300 GOTO 230
310 PRINT:PRINT "SECOND NUMBER";
320 INPUT N
330 IF N=0 THEN 550
340 IF N>1 THEN 370
350 PRINT "SORRY, ENTER NUMBER > 1."
360 GOTO 320
370 IF N<10000 THEN 400
380 PRINT "SORRY, MUST BE LESS THAN 10000."
390 GOTO 320
400 PRINT:PRINT "GREATEST COMMON FACTOR =";
410 IF M=N THEN G=M:GOTO 510
420 IF M<N THEN 440
430 MM=M:M=N:N=MM
440 FOR I=1 TO M
450 IF (M/I)<>INT(M/I) THEN 490
460 J=M/I
470 IF N/J<>INT(N/J) THEN 490
480 G=J:GOTO 510
490 NEXT I
500 G=1
510 PRINT G
520 PRINT:PRINT "PRESS ANY KEY TO CONTINUE."
;
530 A$=INKEY$:IF A$="" THEN 530
540 GOTO 120
550 CLS:END
```

## Least Common Multiple

"Least Common Multiple" finds the least common multiple of two or three numbers. First, press 0, 2, or 3 for the number of given numbers. Zero will stop the program. Next, enter the numbers. The program will return the least common multiple.

Example: Enter three numbers: 12, 6, and 18. The least common multiple is 36; it is the smallest number that can be divided evenly by 12, 6, and 18.

This mathematical concept is usually introduced to students before they learn about adding and subtracting fractions with unlike denominators. The smallest common denominator of several fractions is the "least common multiple" of the denominators.

### Program 16. Least Common Multiple

```

100 REM LEAST COMMON MULTIPLE
110 REM
120 CLS:PRINT @4,"LEAST COMMON MULTIPLE"
130 PRINT @36,"OF 2 OR 3 GIVEN NUMBERS"
140 G$=CHR$(175)+CHR$(175)
150 H$=G$+"{3 SPACES}" +G$+G$+"{4 SPACES}" +G$
    +G$+G$+G$
160 PRINT @101,H$
170 PRINT @133,G$+"{3 SPACES}" +G$+G$+" >> " +
    G$+G$+G$+G$
180 PRINT @165,H$
190 PRINT @202,G$+G$+"{4 SPACES}" +G$+G$+G$+G$
    $
200 PRINT @288,"HOW MANY NUMBERS--0, 2, OR 3
    ?";
210 I$=INKEY$:IF I$=""THEN 210
220 IF I$="0" THEN 660
230 IF ASC(I$)<50 OR ASC(I$)>51 THEN 210
240 PRINT @318,I$
250 FOR C=1 TO VAL(I$)
260 PRINT "NUMBER";C;
270 INPUT N(C)
280 IF N(C)>1 THEN 310
290 PRINT "SORRY, NUMBER MUST BE > 1."
300 GOTO 270
310 IF N(C)<1000 THEN 340
320 PRINT "SORRY, NUMBER MUST BE < 1000."
330 GOTO 270
340 NEXT C
350 C=VAL(I$):IF C=3 THEN 450
360 IF N(1)<>N(2) THEN 380
370 L=N(1):GOTO 620

```

## Programs

---

```
380 IF N(1)<N(2) THEN 400
390 NN=N(1):N(1)=N(2):N(2)=NN
400 FOR C=1 TO N(1)
410 IF C*N(2)/N(1)=INT(C*N(2)/N(1)) THEN L=C
    *N(2):GOTO 620
420 NEXT C
430 L=N(1)*N(2)
440 GOTO 620
450 IF N(1)=N(2) AND N(2)=N(3) THEN 370
460 S=0
470 FOR C=1 TO 2
480 IF C(C)<=N(C+1) THEN 500
490 NN=N(C):N(C)=N(C+1):N(C+1)=NN:S=1
500 NEXT C
510 IF S=1 THEN 460
520 FOR C=1 TO N(2)
530 F=C*N(3)
540 IF (F/N(1)=INT(F/N(1))) AND (F/N(2)=INT(
    F/N(2))) THEN L=F:GOTO 620
550 NEXT C
560 M=N(2)*N(3)
570 FOR C=1 TO N(1)
580 F=C*M
590 IF F/N(1)=INT(F/N(1)) THEN L=F:GOTO 620
600 NEXT C
610 L=M*N(1)
620 PRINT:PRINT "LEAST COMMON MULTIPLE IS";L
630 PRINT:PRINT "PRESS ANY KEY TO CONTINUE."
    ;
640 I$=INKEY$:IF I$="" THEN 640
650 GOTO 120
660 CLS:END
```

## Homework Helper — Fractions

"Homework Helper — Fractions" quickly gives answers to problems involving fractions. Students are encouraged to do their class assignments on paper in the usual way, writing the problem down and working the problem step by step. This program can then be used to correct assignments. There are seven sections, each introduced with a simple color representation of what it is doing with fractions.

This program is written in 16K Extended BASIC. If you have the 4K Color Computer, each section can be a separate program. For non-Extended BASIC, leave out all the graphics commands.



**1. Equivalence.** Two fractions are of the form  $\frac{a}{b} = \frac{c}{d}$ . Any one of the four positions can be the unknown. The user designates the unknown and inputs the three given values. The computer finds the unknown and prints the equivalent fractions. This section may also be used for finding equivalent ratios. The student may enter 0 to return to the menu screen.

**2. Simplification.** The student inputs a numerator and a denominator. The computer simplifies (reduces) the fraction or tells if it cannot be simplified. The student may enter 0 to return to the menu screen.

**3. Multiplication.** The student designates the number of fractions to be multiplied, then enters the numerator and denominator for each one. The computer multiplies them and simplifies the final fraction. The student may enter 0 at any time to return to the menu screen.

**4. Division.** Two fractions are entered, and the first is divided by the second. The answer is in simplified (reduced) form. The student may enter 0 at any time to return to the menu screen.

**5. Addition — like denominators.** The student specifies the number of fractions to be added and the common denominator. He then enters the numerators. The computer adds the numbers and simplifies the result. The student may press 0 number of fractions or enter 0 as the denominator to return to the menu screen.

**6. Addition — unlike denominators.** This section may be used to add fractions with like or unlike denominators. The student specifies the number of fractions (up to five, which should be sufficient for elementary school mathematics) and then inputs the numerator and denominator of each. The computer adds the fractions and simplifies the results. The student may press 0 number of fractions or enter 0 for any denominator to return to the menu screen.

For subtraction problems, either Section 5 or Section 6 may be used (depending on the denominators), and a negative numerator can be entered.

**7. Comparisons.** Up to nine fractions may be compared on a number line. The student presses the number of fractions and then enters the numerator and denominator of each. The fractions are then arranged from smallest to largest and printed. If two or more of the fractions are equal, the earliest-entered fraction will be listed the appropriate number of times. The student may press 0 number of fractions or enter 0 for any denominator to return to the menu screen.

This program should be adequate for fractions used by students in the fourth, fifth and sixth grades.

### **Program Efficiency Techniques**

**Simplifying Fractions.** One basic technique of simplifying fractions is to start with the numerator as the first factor and see if it can be divided evenly into the denominator. If it can, both numerator and denominator are divided by that factor to immediately yield the simplified fraction. If the denominator cannot be evenly divided, the factor is reduced by one, and the numerator and denominator are tested to see if they are divisible by the new factor.

In each successive test the factor is reduced by one. When both numerator and denominator can be evenly divided by the factor, that factor is the greatest common factor. The numerator and denominator are then divided by this factor to yield the reduced fraction.

For larger numbers the technique can take a lot of time. In this program, the algorithm has been made more efficient by first checking to see which is smaller, the numerator or the denominator. In improper fractions the denominator will be smaller. The starting factor, PL or L, is set equal to the smaller number (line 270, line 1450).

Another efficiency technique is to eliminate testing of all even factors if either numerator or denominator is an odd number. This technique cuts the search time in half. In lines 1460-1470 the step size, S, is set equal to  $-2$  if either the numerator or the denominator is odd, and S is set equal to  $-1$  if both numerator and denominator are even numbers.

The simplifying algorithm is implemented with a FOR-NEXT loop. The starting trial factor is reduced by the step size, S, to a lower limit of 2: 1480 FOR P=L TO 2 STEP S. Within the loop, let  $A = N/P$  where N is the numerator and  $B = D/P$  where D is the denominator. Check to see if  $A = \text{INT}(A)$ . If they are equal, then check if  $B = \text{INT}(B)$ . If both statements are true, the simplified fraction is A/B. Otherwise, P is incremented by S and the loop continues. If the lower limit is reached without finding a successful factor, the fraction cannot be simplified, and the student is notified.

When you're combining several fractions in multiplication or addition, another efficiency technique is to set the starting factor equal to the largest denominator of the original fractions. The common denominator may be much larger than the original

denominators, but the largest factor will always be the largest original denominator.

**Comparisons.** The schoolroom technique for comparing fractions is to find the common denominator and then compare the adjusted numerators. This technique is far too slow, especially for comparing many fractions and/or fractions with large numbers. A very fast technique which achieves the same result is to compute and compare the decimal equivalents of the fractions.

As the fractions are read in, the numerator NN(I) is divided by the denominator DD(I) and stored as a decimal fraction in two identical arrays, FC(I) and FD(I) (lines 2340-2360). The first array FC is sorted from smallest to largest by a standard sort routine. The subscripts are changed as the decimal fractions are arranged in order.

The first element of the first array, FC, is compared with each element of the second array, FD. When a match is made, the subscript value J is used to retrieve the numerator and denominator of the corresponding fraction for printing. The process is repeated for each element in the FC array in order.

Line Numbers	Explanation
120	DIMENSION variables for the maximum number of fractions, nine.
130	Branch to title screen.
140-150	Print list of given fractions.
160-200	Simplify final numerator and denominator if possible by dividing by original denominators.
210-250	Sort routine to arrange denominators in order from smallest to largest.
260	Set limit of largest denominator for simplifying routine.
270	Set limit to smaller of numerator or denominator.
280-330	Simplify fraction and return.
340-360	If fraction is improper, change it to mixed fraction.
370-680	Draw title screen and play music.
690-850	Print main menu screen.
860-880	Wait for student to press a number indicating choice and then branch appropriately.
890-950	Print title and graphics for equivalence.
960-1020	Ask for unknown and then branch appropriately.

## Programs

---

1030-1210	Student enters three given quantities and unknown is calculated.
1220-1270	Print equivalent fractions; wait for student to press a key to continue.
1280-1410	Print title and graphics for simplifying fractions.
1420-1440	Student enters numerator and denominator.
1450-1550	Calculate and print reduced or simplified fraction; wait for student to press a key to continue.
1560-1600	Print title and graphics for multiplying fractions.
1610-1660	Ask for number of fractions and print number.
1670-1710	Student enters numerator and denominator of each fraction.
1720-1750	Clear screen; reprint fractions; simplify and print product; wait for student to press a key.
1760-1840	Print title, graphics, and instructions for dividing fractions.
1850-1880	Student enters fractions.
1890-1950	Clear screen; print problem; print simplified answer; wait for student to press a key.
1960-2010	Print title and instructions for adding fractions; student presses number of fractions.
2020-2040	Student enters denominator, then numerators.
2050-2070	Clear screen; reprint fractions with total; wait for student to press a key.
2080-2140	Print title and instructions for adding fractions; student presses number of fractions.
2150-2200	Student enters fractions.
2210-2240	Calculate answer; reprint problem with total; wait for student to press a key.
2250-2270	Print title and instructions for comparisons.
2280-2320	Ask for number of fractions.
2330-2360	Student enters fractions.
2370-2470	Arrange fractions in order from smallest to largest; wait for student to press a key.
2480	End.

### **Program 17. Homework Helper — Fractions**

```
100 REM HOMEWORK HELPER-FRACTIONS
110 REM
120 DIM NN(9), DD(9), FC(9), FD(9)
130 GOTO 370
140 FOR I=1 TO F:PRINT NN(I);"/";DD(I):NEXT
    I
```

```

530 REM INVERT MATRIX A
540 FOR I=1 TO N
550 IF W(I,I)=0 THEN GOSUB 140
560 W(I,I)=1/W(I,I)
570 FOR J=1 TO N
580 IF J-I=0 THEN 640
590 W(J,I)=W(J,I)*W(I,I)
600 FOR K=1 TO N
610 IF K-I=0 THEN 630
620 W(J,K)=W(J,K)-W(J,I)*W(I,K)
630 NEXT K
640 NEXT J
650 FOR K=1 TO N
660 IF K-I=0 THEN 680
670 W(I,K)=-W(I,I)*W(I,K)
680 NEXT K
690 NEXT I
700 PRINT:PRINT:PRINT "SOLUTION VECTOR X:":P
    RINT
710 FOR I=1 TO N
720 X(I)=0
730 FOR J=1 TO N
740 X(I)=X(I)+W(I,J)*B(J)
750 NEXT J
760 PRINT "X("+RIGHT$(STR$(I),1)+") = ";X(I)
770 NEXT I
780 PRINT
790 END

```

## Computer-Aided Instruction

The computer is a very helpful learning tool in just about any educational subject. The computer offers drill and practice, or actually serves as a tutor, teaching concepts at the student's own pace. The programs in this section illustrate only a few of the ways in which educational programs can be written.

"Buying Items" and "Earning Money" are drill-and-practice programs. They illustrate how you can get the computer to make up "word problems." "Typing Trainer," another drill-and-practice program, uses graphics to liven things up a bit. A built-in time clock is also used to motivate students to try harder.

"Learn the Teeth" and "New England States" are drills using graphics, which first illustrate the concept to be learned. "Typing

Unit 1" is a tutorial; it is a self-contained program that teaches a person the home position in touch-typing.

### Typing Unit 1

"Typing Unit 1" is the first of a series of units written to teach touch-typing on the TRS-80 Color Computer (16K Extended BASIC). Color graphics and music are used to enhance the learning experience. Unit 1 starts teaching the keyboard layout of a typewriter by presenting the "home position" — fingers on A S D F J K L; and thumbs on the space bar.

First, the home position is shown graphically. Two hands are shown on the screen with the letters of the keys in the home position printed above the fingers used to strike those keys. The fingers are numbered for later reference.

For the first exercise a letter is printed above the corresponding finger. The student types the letter shown. The letter must be typed correctly for the program to continue. When the letter is typed correctly, a short tone sounds and the next letter is displayed. First the letters are presented in order from left to right, then they are chosen randomly a total of 20 times.

The next exercise shows a keyboard with the home position keys printed. A short phrase using only home-position letters is printed on the screen. The student types and enters it. If the phrase has been typed correctly, a five-note scale is played; if it is incorrect, a low beep is sounded. If the student takes too much time, a reminder not to look at fingers is printed. The phrase is still tested for correctness.

The phrases are chosen randomly, and five different phrases must be typed correctly to complete the exercise. If the phrase has been typed correctly, it is not used again; but if an error was made, that phrase may be used again.

After this exercise the student has the option of practicing again or ending the program. Only the home-position keys are taught in Unit 1.

Line Numbers	Explanation
110	DIMension the array for typing phrases.
120	Draw the title screen with a picture of a typewriter; print the "Unit 1" title screen.
130	Print first and second instruction screens; draw screen with hands and letters above fingers.
140-150	Delay and play a tune.

160	Branch to first exercise.
170-210	Subroutine for pressing ENTER after instruction screens.
220-420	Print instruction screens.
430-470	Subroutine — draw A; wait for student to type A; sound a tone, and then erase the letter.
480-820	Subroutine for the other letters to be typed.
830-860	Print instruction screen.
870-1000	Print the keyboard; erase all but the home keys.
1010-1030	Define typing phrases.
1040-1060	Randomly choose a phrase.
1070	Print the phrase; sound a beginning tone.
1080	Initialize student's input and timer.
1090-1150	Print student's input until ENTER is pressed or 20 characters have been printed.
1160-1170	Stop the timer. If the student has taken longer than five seconds, he is reminded not to look at fingers.
1180-1210	Compare given phrase with typed phrase. If it is incorrect, sound a beep and delay; if it is correct, play scale.
1220-1240	Clear the printing.
1250-1280	Ask if student wants more practice; branch appropriately.
1290-1340	Print ending screen; play music.
1350	Draw hands.
1360-1370	Erase letters above hands.
1380-1400	Print the letters in order from left to right and wait for student to type each letter.
1410-1430	For 20 times, randomly choose letters to be typed.
1440-1930	Subroutine to draw the typewriter.
1940-2190	Subroutine to draw the keyboard.
2200-2270	Subroutine for the "Unit 1" title screen; includes drawing the keyboard and playing a tune.
2280-2310	Subroutine to outline instructions in color.
2320-2630	Subroutine to draw the hands, draw the letters above the fingers, number the fingers, and play a tune.
2640-2710	Subroutines used in drawing hands.
2720	End.

### Program 19. Typing Unit 1

```
100 'TYPING UNIT 1
110 DIM A$(6)
120 GOSUB 1440: GOSUB 2200
130 GOSUB 220: GOSUB 2320
140 FOR D=1 TO 1000:NEXT D
150 PLAY "L8;03;D;L16;DD;L8;E;L16;EEFFEG;L8;
    EDD;L16;DD;L8;E;L16;EEFGEF;L2;D"
160 GOTO 380
170 PRINT @ 483,"<ENTER>";
180 E$=INKEY$
190 IF E$="" THEN 180
200 IF ASC(E$)<>13 THEN 180
210 RETURN
220 SCREEN 0,0:CLS
230 PRINT @ 99,"AFTER READING EACH"
240 PRINT @ 195,"INSTRUCTION SCREEN"
250 PRINT @ 291,"PRESS <ENTER> TO CONTINUE."
260 PRINT @ 419,"(USE RIGHT LITTLE FINGER.)"
270 GOSUB 180:CLS
280 PRINT @ 66,"PLACE YOUR FINGERS IN THE"
290 PRINT @ 98,"HOME" POSITION."
300 PRINT @ 162,"THUMBS GO ON THE SPACEBAR."
310 PRINT @ 194,"EACH FINGER GETS A KEY."
320 PRINT @ 290,"YOU WILL BE ABLE TO TYPE"
330 PRINT @ 322,"MORE ACCURATELY AND QUICKLY"
    "
340 PRINT @ 354,"IF YOU DO NOT LOOK"
350 PRINT @ 386,"AT YOUR FINGERS."
360 COL=4:GOSUB 2280
370 GOSUB 170:RETURN
380 CLS:PRINT @ 66,"AFTER THE MUSIC PLAYS,":
    PRINT @ 98,"TYPE EACH LETTER AS IT"
390 PRINT @ 130,"APPEARS ON THE SCREEN.":PRI
    NT @ 226,"USE THE CORRECT FINGERS!"
400 PRINT @ 322,"YOU MUST TYPE THE LETTERS":
    PRINT @ 354,"CORRECTLY TO CONTINUE."
410 COL=2: GOSUB 2280
420 GOSUB 170:GOTO 1350
430 L1=12: L2=64: GOSUB 2640
440 H$=INKEY$
450 IF H$<>"A" THEN 440
460 SOUND 159,5: COLOR 5,5
470 LINE(2,64)-(18,78),PSET,BF:COLOR 7,5:RET
    URN
480 DRAW "BM44,61;U1L5D5R5D5L5"
490 H$=INKEY$
500 IF H$<>"S" THEN 490
```



```

510 SOUND 159,5: COLOR 5,5
520 LINE (38,60)-(50,76),PSET,BF:COLOR 7,5:RE
TURN
530 DRAW"BM64,60;R5F2D6G2L5U9"
540 H$=INKEY$
550 IF H$<>"D" THEN 540
560 SOUND 159,5: COLOR 5,5
570 LINE (64,60)-(70,72),PSET,BF:COLOR 7,5:RE
TURN
580 DRAW "BM88,61;R5D1L5D3R4D1L4D6"
590 H$=INKEY$
600 IF H$<>"F" THEN 590
610 SOUND 159,5: COLOR 5,5
620 LINE (88,61)-(94,72),PSET,BF:COLOR 7,5:RE
TURN
630 DRAW "BM158,61;D11L5U2"
640 H$=INKEY$
650 IF H$<>"J" THEN 640
660 SOUND 159,5: COLOR 5,5
670 LINE (152,61)-(158,76),PSET,BF:COLOR 7,5:
RETURN
680 L1=178:L2=60:GOSUB 2680
690 H$=INKEY$
700 IF H$<>"K" THEN 690
710 SOUND 159,5:COLOR 5,5
720 LINE (178,60)-(186,72),PSET,BF:COLOR 7,5:
RETURN
730 DRAW "BM206,61;D11R5"
740 H$=INKEY$
750 IF H$<>"L" THEN 740
760 SOUND 159,5:COLOR 5,5
770 LINE (206,61)-(212,72),PSET,BF:COLOR 7,5:
RETURN
780 DRAW "BM240,64;D2R2U2":DRAW"BM240,74;U2R
2D4"
790 H$=INKEY$
800 IF H$<>";" THEN 790
810 SOUND 159,5:COLOR 5,5
820 LINE (240,64)-(242,76),PSET,BF:COLOR 7,5:
RETURN
830 CLS:PRINT @ 65,"NOW LET'S TYPE WORDS."
840 PRINT @ 161,"KEEP YOUR EYES ON THE SCREE
N.":PRINT @ 257,"TYPE THE PHRASE THAT IS
SHOWN."
850 PRINT @ 289,"PRESS <ENTER> AFTER TYPING"
:PRINT @ 321,"EACH PHRASE."
860 COL=3:GOSUB 2280:GOSUB 170
870 CLS:GOSUB 1940
880 FOR I=4 TO 52 STEP 6

```

## Programs

---

```
890 FOR J=2 TO 3
900 SET(I,J,3):SET(I+1,J,3)
910 SET(I+4,J+8,3):SET(I+5,J+8,3)
920 SET(I+5,J+8,3)
930 NEXT J:NEXT I
940 SET(58,2,3):SET(59,2,3)
950 SET(58,3,3):SET(59,3,3)
960 SET(59,3,3)
970 FOR I=30 TO 36 STEP 6
980 FOR J=6 TO 7
990 SET(I,J,3):SET(I+1,J,3)
1000 NEXT J: NEXT I
1010 A$(1)="A SAD LAD;":A$(2)="A FALL FAD"
1020 A$(3)="ASK ALL LADS":A$(4)="A SAD FALL;
"
1030 A$(5)="A LAD ASKS DAD":A$(6)="ASK DAD; "
1040 FOR I=1 TO 5
1050 J=RND(6)
1060 IF A$(J)="" THEN 1050
1070 PRINT @ 326,A$(J):SOUND 159,5
1080 B$="":TIMER=0
1090 PRINT @ 352,"{6 SPACES}";
1100 T$=INKEY$
1110 IF T$="" THEN 1100
1120 IF ASC(T$)=13 THEN 1160
1130 PRINT T$;
1140 B$=B$+T$
1150 IF LEN(B$)<20 THEN 1100
1160 NT=TIMER:IF NT<300 THEN 1180
1170 PRINT @ 418,"PLEASE DO NOT LOOK AT FING
ERS."
1180 IF B$=A$(J) THEN 1210
1190 SOUND 5,5:FOR D=1 TO 1000:NEXT D
1200 I=I-1:GOTO 1220
1210 PLAY "L16;03;CDEFG":A$(J)=""
1220 PRINT @ 418,"{32 SPACES}"
1230 PRINT @ 326,"{17 SPACES}"
1240 PRINT @ 358,"{32 SPACES}"
1250 NEXT I:PRINT @ 356,"WANT MORE PRACTICE?
Y/N"
1260 CH$=INKEY$
1270 IF CH$="Y" THEN 870
1280 IF CH$<>"N" THEN 1260
1290 CLS:PRINT @ 138,"GREAT!!"
1300 PRINT @ 226,"NOW YOU KNOW THE HOME KEYS
"
1310 PRINT @ 325,"NOW YOU NEED UNIT 2."
1320 COL=3:GOSUB 2280
```

```

1330 PLAY "L8;04;C;L16;EC;L8;03;GG;04;C;L16;
      EC;L8;03;G;04;G;L16;FEDC;03;B;04;C;03;B
      ;04;CDC;03;BA;L2;G"
1340 END
1350 GOSUB 2320:COLOR 5,5
1360 LINE(0,60)-(244,75),PSET,BF
1370 LINE(242,74)-(244,76),PSET,BF:COLOR 7,5
1380 FOR I=1 TO 2
1390 GOSUB 430:GOSUB 480:GOSUB 530:GOSUB 580
1400 GOSUB 630:GOSUB 680:GOSUB 730:GOSUB 780
      :NEXT I
1410 FOR I=1 TO 20
1420 J=RND(8):ON J GOSUB 430,480,530,580,630
      ,680,730,780
1430 NEXT I:GOTO 830
1440 PMODE 3,1:PCLS:SCREEN 1,0:COLOR 3,1
1450 DRAW "BM72,4;R12D1L12R6D14"
1460 DRAW "BM92,4;F8D7U7E8"
1470 DRAW "BM114,19;U15R7D1L7R7F1D5G1L7D1R7"
1480 DRAW "BM130,4;D15":DRAW "BM138,4;D15":D
      RAW "BM148,4;D15"
1490 LINE(138,4)-(148,19),PSET
1500 DRAW "BM158,4;R6F1L7G1D11F1R6D1L6U1R6E1
      U6L2D1"
1510 COLOR 4,1
1520 DRAW "BM76,36;D13F1R6D1L5U1R5E1U13"
1530 DRAW "BM94,36;D15"
1540 DRAW "BM104,36;D15"
1550 LINE(94,36)-(104,51),PSET
1560 DRAW "BM112,36;D15"
1570 DRAW "BM118,36;R12D1L12R6D14"
1580 DRAW "BM164,39;U1E2D15"
1590 LINE(56,96)-(194,107),PSET,BF
1600 DRAW "BM54,98;D7"
1610 DRAW "BM196,98;D7"
1620 DRAW "BM76,132;R100F11L122E11"
1630 PAINT(84,142),4,4
1640 COLOR 3,1
1650 LINE(68,108)-(64,143),PSET
1660 DRAW "BM64,143;E12R100F11"
1670 LINE(182,108)-(184,143),PSET
1680 DRAW "BM68,108;R36D8F4R36E4U8R36"
1690 PAINT(80,120),3,3
1700 LINE(72,84)-(68,95),PSET
1710 LINE(178,84)-(182,95),PSET
1720 LINE(72,84)-(178,84),PSET
1730 LINE(68,95)-(182,95),PSET
1740 PAINT(76,88),3,3
1750 LINE(64,144)-(186,163),PSET,BF

```

## Programs

---

```
1760 LINE(66,164)-(184,175),PSET,BF
1770 LINE(68,176)-(182,187),PSET,BF
1780 LINE(72,188)-(178,191),PSET,BF
1790 COLOR 2,3
1800 FOR L1=84 TO 164 STEP 8
1810 LINE(L1,149)-(L1+1,151),PSET,BF
1820 LINE(L1-2,156)-(L1-1,159),PSET,BF
1830 LINE(L1,164)-(L1+1,167),PSET,BF
1840 LINE(L1+2,172)-(L1+3,175),PSET,BF
1850 NEXT L1
1860 DRAW "BM170,156;D3R1U3"
1870 DRAW"BM172,172;D3R2U3R2D3
1880 LINE(76,172)-(80,175),PSET,BF
1890 DRAW "BM74,164;D3R2U3"
1900 LINE(172,162)-(178,169),PSET,BF
1910 LINE(92,180)-(158,183),PSET,BF
1920 PLAY "L16;O3;CD;L8;EEGF;L16;FGFE;L8;DDF
E;L16;EFED;L8;CEGF;L16;FGFE;L8;D;L16;DE
FD;L2;C"
1930 SCREEN 0,1:RETURN
1940 PRINT @ 34,"Q W E R T Y U I O
P"
1950 PRINT @ 99,"A S D F G H J K L
;"
1960 PRINT @ 164,"Z X C V B N M , ."
1970 FOR I=0 TO 63
1980 SET(I,0,4):SET(I,1,4)
1990 SET(I,12,4):SET(I,13,4)
2000 SET(I,4,3):SET(I,5,3)
2010 SET(I,8,3):SET(I,9,3)
2020 NEXT I
2030 FOR J=2 TO 11
2040 FOR I=0 TO 3
2050 SET(I,J,4)
2060 NEXT I
2070 SET(62,J,4):SET(63,J,4)
2080 NEXT J
2090 FOR I=6 TO 54 STEP 6
2100 FOR J=I TO I+3
2110 SET(J,2,3):SET(J,3,3)
2120 SET(J+2,6,3):SET(J+2,7,3)
2130 SET(J+4,10,3):SET(J+4,11,3)
2140 NEXT J:NEXT I
2150 FOR J=0 TO 1:FOR I=0 TO 1
2160 SET(60+I,2+J,3):SET(4+I,6+J,3)
2170 SET(4+I,10+J,3)
2180 SET(6+I,10+J,3)
2190 NEXT I:NEXT J:RETURN
2200 'UNIT 1 TITLE
```

```

2210 SCREEN 0,0:CLS
2220 PRINT @ 300,"UNIT 1"
2230 PRINT @ 357,"LEARNING THE KEYBOARD"
2240 PRINT @ 425,"HOME POSITION"
2250 GOSUB 1940
2260 PLAY "L16;O3;CE;L8;DD;L16;DF;L8;EE;L16;
EGFEDEFD;L8;EE;L16;CE;L8;DD;L16;DF;L8;E
E;L16;EG;L8;FEF;L2;G"
2270 RETURN
2280 FOR I=0 TO 63
2290 SET(I,0,COL):SET(I,1,COL)
2300 SET(I,30,COL):SET(I,31,COL)
2310 NEXT I:RETURN
2320 PMODE 3,1:PCLS:SCREEN 1,1:COLOR 7,6
2330 DRAW"BM4,191;U11E2U19H2U14H2U26E2U3EU3E
2U1E2R2F4D28F2D6F2D7F2D7F2R2"
2340 DRAW "BM26,161;U13H2U22E2U10E2U6E2U2E2U
4E2U1E2U2E4R3F3D12G2D6G2D6G2D14F2D22R4"
2350 DRAW"BM48,155;U39E2U6E2U6E2U6E2U6E2U3E4
R2F4D4F2D6G2D14G2D43R4"
2360 DRAW "BM72,153;U21E2U6E2U14E2U6E2U2E2U2
E2U2R4F2D2F2D21G2D6G2D18G2D22G2D18R2"
2370 DRAW "BM86,189;E4U1E2U4E2U4E6U3E6R3E2R3
D7G2D6G2D2G2D15"
2380 DRAW "BM138,191;U15H2U2H2U6H2U7R3F3R3F3
D2F2D2F6D4F2D4F6R1"
2390 DRAW "BM160,187;U19H2U22H2U18H2U6H2U22E
2U2R6D3F2D2F2D2F2D6F2D14F2D6F2D20R4E1"
2400 DRAW "BM178,152;U44H2U14H2U6E2U2E4R2F6D
2F2D6F2D6F2D6F2D6F2D34R4"
2410 DRAW "BM204,154;U23E2U14H2U6H2U6H2U10E4
R2F6D2F2D2F2D2F2D2F2D6F2D10F2D22F2D10F2
R2"
2420 DRAW "BM230,160;F2U7E2U6E2U6E2U26E6R2F6
D4F2D26G2D14G2D18F2D10"
2430 PAINT (52,192),8,7
2440 PAINT(200,192),8,7
2450 LINE(100,140)-(146,151),PSET,BF
2460 L1=12:L2=64:GOSUB 2640
2470 DRAW "BM44,61;U1L5D5R5D5L5"
2480 DRAW "BM64,60;R5F2D6G2L5U9"(11 SPACES)
2490 DRAW "BM88,61;R5D1L5D3R4D1L4D6"
2500 DRAW "BM78,156;D7"
2510 DRAW "BM60,157;U1R3D4G4R3"
2520 DRAW "BM36,160;R3D3L2R2D4L3"
2530 DRAW "BM12,166;D6R4L1U1D5"
2540 DRAW "BM158,61;D11L5U2"
2550 L1=178:L2=60:GOSUB 2680
2560 DRAW "BM206,61;D11R5"

```

## Programs

---

```
2570 DRAW "BM240,64;D2R2U2": DRAW "BM240,74;  
U2R2D4"  
2580 DRAW "BM168,156;D7"  
2590 DRAW "BM188,157;U1R3D4G4R3"  
2600 DRAW "BM212,160;R3D3L2R2D4L3"  
2610 DRAW "BM236,166;D6R4L1U1D5"  
2620 PLAY "L8;O3;E;L16;EE;L8;G;L16;GECCDD;L8  
;ECE;L16;EE;L8;G;L16;GECCDE;L2;C"  
2630 RETURN  
2640 LINE (L1,L2)-(L1-6,L2+11),PSET  
2650 LINE (L1,L2)-(L1+6,L2+11),PSET  
2660 LINE (L1-4,L2+7)-(L1+4,L2+7),PSET  
2670 RETURN  
2680 LINE (L1,L2)-(L1,L2+11),PSET  
2690 LINE (L1+8,L2)-(L1,L2+5),PSET  
2700 LINE (L1+8,L2+11)-(L1+1,L2+4),PSET  
2710 RETURN  
2720 END
```

## Typing Trainer

This program was written for the TRS-80 Color Computer (16K Extended BASIC) and uses color graphics and sound to help a student practice typing sentences for speed and accuracy. The TIMER function is used to calculate a rate of typing speed.

There are 40 different 30-stroke sentences that are chosen randomly for the drills. Each drill consists of ten sentences. A sentence is shown on the screen; the student types and enters it. If it is incorrect, a beep sounds and a wrong score is posted. The student has time to review the sentence before continuing. If the typed sentence is correct, a right score is posted, the engine at the top of the screen moves forward, and a tune is played. The running total is displayed on the screen after each sentence. After each sentence is typed, the rate in words per minute is calculated and displayed. The rate is calculated from the number of strokes the student typed, divided by five strokes per word. The computer's timer is used for the elapsed time.

After ten sentences the overall words per minute for all ten sentences is calculated and displayed, and another tune is played. After each drill the student may choose whether to try again or not. If the student enters N for no, the program ends. If Y for yes is entered instead, the drill is repeated with different sentences. Each drill chooses the sentences randomly, and the drill may be performed four times without sentences being repeated. After that,

the sentences are reloaded as data for more drills. This process continues as long as the student wishes to continue.

Line Numbers	Explanation
120-180	Subroutine for PRESS < ENTER > TO CONTINUE. Wait for user to press the ENTER key, then erase this message and return.
190-230	DIMension and load array A\$ with 40 sentences from data. The data is RESTORED after the drill has been completed four times. Sentences are used only once every four drills.
240	Skip over the title screen and instructions if the drill has been completed previously.
250	Print title screen.
260	Print instructions if the user desires.
270-280	Initialize variables for each drill.
290	Clear the screen; draw the engine; print the scoreboard.
300	Play introductory tune.
310	Do procedure ten times.
320	Initialize check flag for each sentence.
330-350	Choose a random number from 1 to 40. If the sentence has been used previously, choose another random number; print the sentence.
360	Start the built-in timer.
370	Receive the user's typed sentence.
380	Stop the timer.
390-460	Compare user's typed sentence to given sentence. If the sentence is incorrect, sound a beep, increment the "wrong" score, set the check flag equal to 1; and print the score. If the sentence was correct, increment and print the "right" score.
470	Reset A\$(J) to a null string so the sentence cannot be used again.
480-500	Find the length of the user's sentence and calculate words per minute (wpm) for that number of strokes. A "word" is five strokes; the timer value is the number of 1/60 seconds. The wpm is rounded to the nearest integer.
510-520	Increment variables for total number of strokes and total time.

530-560	If the sentence was correct, move the engine forward and play a tune.
570	If the sentence was incorrect, pause so the student can review the error; the student must press ENTER to continue.
580-600	Erase the sentences and go to the next sentence.
610-640	When ten sentences have been completed, compute and print the overall total words per minute and play an ending tune.
650-690	Ask WANT TO TRY AGAIN? and then branch appropriately.
700-720	Depending on how many times the drill has been performed, enter the beginning of the drill.
730-750	If the user wants to stop, clear the screen, print BYE, and end.
760-950	DATA statements containing typing sentences.
960-1050	Coordinates for drawing the title screen entered as data.
1060-1440	Subroutine for drawing the title screen.
1060-1100	Set mode, screen, and color, and clear the screen.
1110-1140	Print "TYPING".
1150-1180	Print "TRAINER".
1190-1250	Draw the red engine and coal car.
1260-1410	Draw the blue portions of the engine and coal car.
1420-1440	Play a tune, then return to text screen.
1450-1610	Subroutine for printing instructions.
1450-1500	Clear the screen and ask if instructions are needed; wait for response and branch accordingly.
1510-1610	Print the instructions, then return.
1620-1930	Subroutine for drawing the main screen for the drill.
1620-1890	Clear the screen and draw the engine.
1900-1930	Print the scoring titles and return.
1940-2340	Subroutine for moving the engine forward.
2350	End.

### Program 20. Typing Trainer

```
100 ' TYPING TRAINER
110 GOTO 190
120 PRINT @ 448, " PRESS <ENTER> TO CONTINUE"
130 Z$=INKEY$
140 IF Z$="" THEN 130
150 AZ=ASC(Z$)
```



```

160 IF AZ<>13 THEN 130
170 PRINT @ 449,"(30 SPACES)"
180 RETURN
190 DIM A$(40)
200 RESTORE
210 FOR I=1 TO 40
220 READ A$(I)
230 NEXT I
240 IF FLAG>4 THEN 270
250 GOSUB 1060
260 GOSUB 1450
270 FLAG=1
280 R=0: TL=0: W=0: T2=0: MX=0
290 GOSUB 1620
300 PLAY "L16;03;CDCDCDCDCDEF;L8;G6;L2;F"
310 FOR I=1 TO 10
320 CH=0
330 J=RND(40)
340 IF A$(J)=" " THEN 330
350 PRINT @ 384,A$(J)
360 TIMER=0
370 LINE INPUT B$
380 TT=TIMER(5 SPACES)
390 IF B$=A$(J) THEN 450
400 SOUND 5,5
410 W=W+1
420 CH=1
430 PRINT @ 305,"WRONG: ";W
440 GOTO 470
450 R=R+1
460 PRINT @ 273,"RIGHT: ";R
470 A$(J)=" "
480 LB=LEN(B$)
490 WPM=INT((LB/5)/TT*3600+.5)
500 PRINT @ 343,WPM
510 TL=TL+LB
520 T2=T2+TT
530 IF CH=1 THEN 570
540 GOSUB 1940
550 PLAY "L16;03;CDCCEF;L8;G;L16;E;L4;G"
560 GOTO 580
570 GOSUB 120
580 PRINT @ 384,"(32 SPACES)"
590 PRINT @ 416,"(31 SPACES)"
600 NEXT I
610 PRINT @ 333,"TOTAL WPM:"
620 TW=INT((TL/5)/T2*3600+.5)
630 PRINT @ 345,TW
640 PLAY "L16;03;GE;L8;CC;L16;GE;L8;CC;L16;E
G;L8;F;L16;GFED;L2;C"

```

## Programs

---

```
650 PRINT @ 449,"WANT TO TRY AGAIN?(Y/N)"
660 Z$=INKEY$
670 IF Z$="Y" THEN 700
680 IF Z$="N" THEN 730
690 GOTO 660
700 FLAG=FLAG+1
710 IF FLAG>4 THEN 200
720 GOTO 280
730 CLS
740 PRINT @ 268,"BYE"
750 END
760 DATA "HE FEELS SHE HAS A SAFE LEASE.", "S
HE IS STILL AT THE LAKE SITE."
770 DATA "JUST SOME OF US HAVE TO DO IT.", "J
ANE STARTS HER TALK AT THREE."
780 DATA "HE DID SEEK AID FOR THE TRUCK.", "I
T IS THIS DESK FILE HE SEEKS."
790 DATA "WE WOULD GIVE HIM A GOOD WAGE.", "I
HOPE THAT TAX DOES NOT PASS."
800 DATA "IT IS UP TO THEM TO WORK HARD.", "H
AVE A GOAL; WORK TO REACH IT."
810 DATA "IT IS HOW WE WORK THAT COUNTS.", "T
OM WAS QUICK TO SEND THE BOX."
820 DATA "REX WILL HAVE MUCH MORE TO DO.", "I
WILL GO TO TOWN TO GET THEM."
830 DATA "HE CAN LEND A HAND TO THE BOY.", "I
PAID THE MEN FOR THEIR WORK."
840 DATA "THE WORKER SAID HE STRUCK OIL.", "S
HE SAID WE NEED A NEW CAMPER."
850 DATA "I BOUGHT THE BIG BOX OF BOOKS.", "W
E SHOULD SET A GOAL FOR THEM."
860 DATA "TRY TO TYPE ALL THE BIG WORDS.", "W
E MAY QUIT THIS WORK AT FIVE."
870 DATA "YOU HAVE TO WORK FOR TWO DAYS.", "T
RY TO GET ONE OR TWO OF THEM."
880 DATA "YOUR BEST MEN WILL HELP DO IT.", "H
AVE THE BOYS DO THE WORK NOW."
890 DATA "LET HIM PROVE THE RIGHT THING.", "T
HEY SHOULD READ MY GOOD BOOK."
900 DATA "SHE CAN DO A BIG JOB THE BEST.", "D
AVE MADE A CAGE FOR HIS PETS."
910 DATA "ALL GLAD DADS HAD A GLASS JAR.", "P
UT A LITTLE MORE EFFORT HERE."
920 DATA "GREG BROUGHT IN A LARGE CHECK.", "B
RING ALL BOOKS TO THE TABLES."
930 DATA "HE KNOWS HE MUST KEEP WORKING.", "C
HECK THE PAPER FOR ANY MARKS."
940 DATA "ANDY MUST GIVE MY BAND A HAND.", "T
HERE IS A QUICK QUIZ FOR HIM."
```

```

950 DATA "TWO OF THE GIRLS ARE HERE NOW.," "T
RY NOT TO LOOK AT YOUR HANDS."
960 DATA 60,32,74,32,60,33,74,33,68,33,68,47
,84,32,88,40,94,32,88,40,88,40,88,47,102
,32,102,47,104,32,110,32,104,33,110,33
970 DATA 112,34,112,39,104,40,110,40,104,41,
110,41,120,32,120,47,128,32,128,47,140,3
2,140,47,128,32,140,47
980 DATA 150,32,158,32,150,33,158,33,160,33,
160,34,148,34,148,45,150,46,158,46,150,4
7,158,47
990 DATA 156,42,160,42,156,43,160,43,160,43,
160,45
1000 DATA 56,60,70,60,56,61,70,61,64,62,64,7
5,76,60,76,75,78,60,84,60,78,61,84,61,8
6,62,86,67,78,68,84,68
1010 DATA 78,69,84,69,82,70,86,75,100,60,92,
75,100,60,108,75,94,71,104,71,94,70,104
,70,112,60,112,75,120,60,120,75
1020 DATA 132,60,132,75,120,60,132,75,140,60
,140,75,142,60,150,60,142,61,150,61,142
,66,146,66,142,67,147,67
1030 DATA 142,74,150,74,142,75,150,75,156,60
,156,75,158,60,164,60,158,61,164,61,166
,62,166,67,158,68,164,68,158,69,164,69
1040 DATA 162,70,166,75,76,100,126,107,88,10
8,126,115,88,116,214,151,152,104,166,11
5,156,100,162,103
1050 DATA 194,104,200,115,192,92,202,103,190
,88,204,91,188,84,206,87,0,116,50,155
1060 PMODE 3,1
1070 ' TITLE SCREEN
1080 PCLS
1090 SCREEN 1,0
1100 COLOR 3,1
1110 FOR I=1 TO 22
1120 READ L1,L2,L3,L4
1130 LINE (L1,L2)-(L3,L4),PSET
1140 NEXT I
1150 FOR I=1 TO 35
1160 READ L1,L2,L3,L4
1170 LINE (L1,L2)-(L3,L4),PSET
1180 NEXT I
1190 COLOR 4,1
1200 FOR I=1 TO 10
1210 READ L1,L2,L3,L4
1220 LINE (L1,L2)-(L3,L4),PSET,BF
1230 NEXT I
1240 PSET (158,98,4)

```

## Programs

---

```
1250 PSET (158,99,4)
1260 COLOR 3,1
1270 CIRCLE (132,156),14,3
1280 LINE (132,156)-(132,157),PSET
1290 CIRCLE (168,156),14,3
1300 LINE (168,156)-(168,157),PSET
1310 LINE (184,152)-(208,159),PSET,BF
1320 LINE (206,156)-(222,167),PSET,BF
1330 LINE (224,160)-(224,169),PSET
1340 LINE (226,164)-(226,169),PSET
1350 LINE (228,166)-(228,167),PSET
1360 LINE (52,148)-(86,151),PSET,BF
1370 LINE (96,108)-(118,119),PSET,B
1380 CIRCLE(40,162),8,3
1390 CIRCLE(20,162),8,3
1400 LINE (40,162)-(40,163),PSET
1410 LINE (20,162)-(20,163),PSET
1420 PLAY "L2;03;CC;L16;B;04;C;03;BABA;L2;G;
L16;AB;04;C;03;BABA;L2;G;L16;B;L2;04;D"
1430 SCREEN 0,0
1440 RETURN
1450 CLS
1460 PRINT @ 224,"DO YOU NEED INSTRUCTIONS?(
Y/N)"
1470 Z$=INKEY$
1480 IF Z$="Y" THEN 1510
1490 IF Z$="N" THEN 1610
1500 GOTO 1470
1510 CLS
1520 PRINT @ 0,"YOU WILL SEE A SENTENCE"
1530 PRINT @ 32,"ON THE SCREEN."
1540 PRINT @ 96,"TYPE AND ENTER IT."
1550 PRINT @ 160,"YOU WILL BE TOLD YOUR WORD
S"
1560 PRINT @ 192,"PER MINUTE (WPM) FOR THAT"
1570 PRINT @ 224,"SENTENCE (6 WORDS). "
1580 PRINT @ 288,"AFTER TEN SENTENCES YOUR F
INAL"
1590 PRINT @ 320,"SCORE AND TOTAL WPM ARE SH
OWN. "
1600 GOSUB 120
1610 RETURN
1620 CLS
1630 'DRAW ENGINE
1640 FOR X=2 TO 15
1650 FOR Y=2 TO 3
1660 SET(X,Y,4)
1670 NEXT Y
1680 NEXT X
```

```

1690 FOR X=4 TO 5
1700 FOR Y=4 TO 5
1710 SET(X,Y,4)
1720 SET(X+10,Y,4)
1730 SET(X+24,Y,4)
1740 SET(X+24,Y-2,4)
1750 NEXT Y
1760 NEXT X
1770 FOR X=4 TO 33
1780 FOR Y=6 TO 11
1790 SET(X,Y,4)
1800 NEXT Y
1810 NEXT X
1820 FOR X=10 TO 15
1830 FOR Y=10 TO 13
1840 SET(X,Y,3)
1850 SET(X+12,Y,3)
1860 NEXT Y
1870 NEXT X
1880 SET(34,12,3)
1890 SET(35,13,3)
1900 PRINT @ 273, "RIGHT:"
1910 PRINT @ 305, "WRONG:"
1920 PRINT @ 337, "WPM:"
1930 RETURN
1940 MX=MX+2
1950 SET(34+MX,12,3)
1960 SET(35+MX,13,3)
1970 SET(34+MX-2,12,1)
1980 SET(35+MX-2,12,1)
1990 SET(35+MX-2,13,1)
2000 SET(34+MX-2,13,1)
2010 FOR X=MX TO MX+1
2020 FOR Y=6 TO 11
2030 SET(32+X,Y,4)
2040 NEXT Y
2050 FOR Y=2 TO 5
2060 SET(28+X,Y,4)
2070 SET(28+X-2,Y,1)
2080 NEXT Y
2090 FOR Y=10 TO 13
2100 SET(26+X,Y,3)
2110 SET(22+X-2,Y,1)
2120 SET(14+X,Y,3)
2130 SET(10+X-2,Y,1)
2140 NEXT Y
2150 FOR Y=2 TO 5
2160 SET(14+X,Y,4)
2170 NEXT Y

```

## Programs

---

```
2180 SET (2+X-2, 2, 1)
2190 SET (2+X-2, 3, 1)
2200 FOR Y=4 TO 11
2210 SET (4+X-2, Y, 1)
2220 NEXT Y
2230 NEXT X
2240 FOR X=2 TO 1 STEP -1
2250 FOR Y=10 TO 11
2260 SET (10+MX-X, Y, 4)
2270 SET (22+MX-X, Y, 4)
2280 NEXT Y
2290 FOR Y=4 TO 5
2300 SET (14+MX-X, Y, 1)
2310 SET (6+MX-X, Y, 4)
2320 NEXT Y
2330 NEXT X
2340 RETURN
2350 END
```

## Buying Items

Math competency tests often have story problems or word problems. This program presents a quiz of buying items on a list. A list of items is printed with their costs, which are random numbers within certain limits. One question asks how much it would cost to buy everything on the list. The second question asks, given a certain amount of money, which items could be purchased. The second question is multiple-choice.

The DATA statements in lines 770-800 consist of people's names, items to be purchased, and minimum and maximum costs of the items. The subroutine in lines 190-240 allows printing of a cost with a dollar sign. If you have Extended BASIC, you could change this to a PRINT USING routine. Lines 580-670 randomly choose the multiple-choice items and place the correct answer in one of the choices.

### Program 21. Buying Items

```
10 CLS
20 PRINT @72, "MATH COMPETENCY"
30 PRINT @169, "BUYING ITEMS"
40 PRINT
50 DIM I$(3, 5), I(3, 5, 2), N$(6), J(5), H$(3), S$(4)
60 FOR C=1 TO 6: READ N$(C): NEXT C
70 FOR A=1 TO 3
```

```

80 FOR C=1 TO 5:READ I$(A,C),I(A,C,1),I(A,C,
  2):NEXT C
90 NEXT A
100 H$(1)="PENCIL AND ERASER":H$(2)="BALL AN
  D TRUCK":H$(3)="CANDY AND FRUIT"
110 FOR D=1 TO 500:NEXT
120 GOTO 250
130 PRINT @495,"PRESS <ENTER>";
140 A$=INKEY$:IF A$="" THEN 140
150 IF ASC(A$)<>13 THEN 140
160 RETURN
170 PLAY "L16;02;EC":RETURN
180 PLAY "L16;02;CEG;03;L8;C":RETURN
190 P$=STR$(P)
200 IF LEN(P$)=2 THEN P$=" 0"+RIGHT$(P$,1)
210 PR$=RIGHT$(P$,2)
220 PL$=LEFT$(P$,LEN(P$)-2)
230 IF LEN(PL$)<2 THEN PL$=" "+PL$
240 P$="$"+PL$+"."+PR$:RETURN
250 CLS:TP=0:A=RND(3)
260 PRINT @1,"GIVEN THIS PRICE LIST:"
270 FOR C=1 TO 5
280 D=I(A,C,2)-I(A,C,1):P=I(A,C,1)+RND(D)
290 GOSUB 190:TP=TP+P
300 PRINT TAB(4);I$(A,C);TAB(13);P$
310 NEXT C:PRINT
320 F=RND(2)
330 IF F=1 THEN PRINT "HOW MUCH WILL IT COST
  TO BUY<4 SPACES>ALL THE ITEMS ON THE LI
  ST?":GOTO 380
340 N=RND(6)
350 PRINT N$(N);" WANTS TO BUY"
360 PRINT "EVERYTHING ON THE LIST."
370 PRINT "WHAT WOULD THE TOTAL COST BE?"
380 PRINT "$";:INPUT X
390 IF ABS(X-TP/100)<.001 THEN 440
400 GOSUB 170
410 PRINT:PRINT "ADD ALL FIVE NUMBERS."
420 PRINT "THE TOTAL IS ";:P=TP:GOSUB 190:PR
  INT P$:GOSUB 130
430 GOTO 250
440 GOSUB 180
450 FOR C=1 TO 8
460 PRINT @192+32*C," "
470 NEXT C
480 IF A=1 THEN M=RND(5)+25:GOTO 510
490 IF A=2 THEN M=RND(36)+239:GOTO 510
500 M=RND(18)+100
510 P=M:GOSUB 190

```

## Programs

---

```
520 IF F=1 THEN PRINT @224,"IF YOU COULD ONL
Y SPEND ";P$:GOTO 540
530 PRINT @224,"IF ";N$(N);" COULD ONLY SPEN
D ";P$
540 PRINT @256,"WHICH OF THESE PAIRS OF ITEM
S(3 SPACES)ON THE LIST COULD ";
550 IF F=1 THEN PRINT "YOU BUY?":GOTO 580
560 IF N<4 THEN PRINT "SHE BUY?":GOTO 580
570 PRINT "HE BUY?"
580 R=RND(4)
590 FOR V=1 TO 4
600 IF V=R THEN S$(V)=H$(A):GOTO 660
610 X=RND(2)+3:S$(V)=I$(A,X)
620 X=RND(3):S$(V)=S$(V)+" AND "+I$(A,X)
630 IF V=1 THEN 660
640 FOR V1=1 TO V-1:IF S$(V1)=S$(V) THEN 610
650 NEXT V1
660 PRINT TAB(3);CHR$(64+V);" "+S$(V)
670 NEXT V
680 A$=INKEY$:IF A$="" THEN 680
690 PRINT @318,A$:IF ASC(A$)<>64+R THEN 740
700 GOSUB 180
710 PRINT @480,"TRY AGAIN? (Y/N)";
720 A$=INKEY$:IF A$="N" THEN 810
730 IF A$="Y" THEN 250 ELSE 720
740 GOSUB 170
750 PRINT @448,"THE TOTAL OF THE TWO ITEMS M
UST BE LESS THAN ";P$;"--";CHR$(64+R)
760 GOTO 710
770 DATA JENNY,CINDY,CHERY,EDDIE,BILLY,JIMMY
,PENCIL,8,15,ERASER,2,10
780 DATA NOTEBOOK,35,99,RULER,29,49,PAPER,59
,90,DOLL,.249,599
790 DATA BALL,49,89,TRUCK,100,150,GAME,270,5
00,MODEL,300,700
800 DATA CANDY,20,50,MEAT,123,425,FRUIT,24,5
0,CHIPS,100,257,BREAD,100,179
810 CLS:END
```

## Earning Money

This program illustrates how a program can be used to generate story or word problems involving earning money or wages. Twelve different names and six different jobs are read in as data. Each problem picks a name at random and chooses the appropriate pronoun in the following statements. Some of the problems list a type of job. All the numbers chosen are random numbers



within certain limits. These problems are multiplication problems — an hourly wage times the number of hours, or an amount earned per week times a number of weeks. The subroutine in lines 150-210 prints a number as money in dollars and cents. If you have Extended BASIC, you can use the PRINT USING statement.

### Program 22. Earning Money

```

10 CLS
20 PRINT @72,"MATH COMPETENCY"
30 PRINT @169,"EARNING MONEY"
40 PRINT
50 DIM N$(5),J$(5),T$(5)
60 FOR I=0 TO 5:READ N$(I),J$(I),T$(I):NEXT
70 FOR D=1 TO 500:NEXT
80 GOTO 220
90 PRINT @495,"PRESS <ENTER>";
100 A$=INKEY$:IF A$="" THEN 100
110 IF ASC(A$)<>13 THEN 100
120 RETURN
130 PLAY "L16;02;EC":RETURN
140 PLAY "L16;02;CEG;03;L8;C":RETURN
150 P=100+25*RND(10)
160 P$=STR$(P)
170 IF LEN(P$)=2 THEN P$=" 0"+RIGHT$(P$,1)
180 PR$=RIGHT$(P$,2)
190 PL$=LEFT$(P$,LEN(P$)-2)
200 IF LEN(PL$)<2 THEN PL$=" "+PL$
210 P$="$"+PL$+"."+PR$:RETURN
220 CLS
230 N=RND(6)-1:H=8+RND(10)
240 GOSUB 150
250 PRINT N$(N);" WORKS";H;" HOURS PER WEEK."
260 IF N<3 THEN PRINT "HE EARNS ";:GOTO 280
270 PRINT "SHE EARNS ";
280 PRINT P$;" PER HOUR.":PRINT
290 IF N<3 THEN PRINT "HOW MUCH DOES HE EARN
":GOTO 310
300 PRINT "HOW MUCH DOES SHE EARN"
310 PRINT "IN A WEEK? $";
320 INPUT D
330 D1=P*H/100:IF ABS(D-D1)>.001 THEN 380
340 GOSUB 140
350 PRINT @480,"TRY AGAIN? (Y/N)";
360 A$=INKEY$:IF A$="Y" THEN 220
370 IF A$="N" THEN 430 ELSE 360
380 GOSUB 130
390 PRINT:PRINT "MULTIPLY";H;" HOURS BY ";P$:
PRINT "PER HOUR."

```

## Programs

---

```
400 P=H*P:GOSUB 160
410 PRINT "THE ANSWER IS ";P$
420 GOSUB 90:GOTO 220
430 CLS
440 N=RND(6)-1:H=B+RND(10):GOSUB 150
450 PRINT N$(N);" EARNS ";P$;" PER HOUR.":PRINT
460 IF N<3 THEN PRINT "HE WORKS";:GOTO 480
470 PRINT "SHE WORKS";
480 PRINT H;"HOURS PER WEEK.":PRINT
490 IF N<3 THEN PRINT "HOW MUCH WILL HE EARN
    IN":GOTO 510
500 PRINT "HOW MUCH WILL SHE EARN IN"
510 W=RND(19)+1
520 PRINT W;"WEEKS? $";
530 INPUT D
540 D1=P*H*W/100
550 IF ABS(D-D1)>.001 THEN 600
560 GOSUB 140
570 PRINT @480,"TRY AGAIN? (Y/N)";
580 A$=INKEY$:IF A$="Y" THEN 430
590 IF A$="N" THEN 670 ELSE 580
600 GOSUB 130
610 PRINT:PRINT "MULTPLY";H;"HOURS BY "P$"
620 PRINT "PER HOUR.":PRINT
630 PRINT "THEN MULTIPLY BY";W;"WEEKS."
640 P=H*P*W:GOSUB 160
650 PRINT:PRINT "THE ANSWER IS ";P$
660 GOSUB 90:GOTO 430
670 CLS
680 J=RND(6)-1:T=RND(6)-1:GOSUB 150:W=RND(8)
    +1
690 PRINT T$(T);" EARNED ";P$;" LAST WEEK":PRINT
    J$(J);".":PRINT
700 IF T<3 THEN PRINT "IF HE EARNED THIS AMOUNT":GOTO 720
710 PRINT "IF SHE EARNED THIS AMOUNT"
720 PRINT "EVERY WEEK, WHAT WOULD THE TOTAL INCOME BE FOR";W;"WEEKS?"
730 INPUT "$";D
740 D1=P*W/100:IF ABS(D-D1)>.001 THEN 790
750 GOSUB 140
760 PRINT @480,"TRY AGAIN? (Y/N)";
770 A$=INKEY$:IF A$="Y" THEN 670
780 IF A$="N" THEN 870 ELSE 770
790 GOSUB 130
800 PRINT:PRINT "MULTIPLY ";P$;" PER WEEK"
810 PRINT "BY";W;"WEEKS."
820 P=P*W:GOSUB 160
```

```

830 PRINT:PRINT "THE ANSWER IS ";P$
840 GOSUB 90:GOTO 670
850 DATA SAM,DOING ODD JOBS,JOHN,JOE,MOWING
    LAWNS,ANDY,BOB,TENDING CHILDREN,MARK,ANN
860 DATA RUNNING ERRANDS,LENA,SUE,DOING HOUS
    EWORK,AURA,KAY,DELIVERING ADS,DAWN
870 CLS:END
    
```

## Learn the Teeth

This program is written in 16K Extended BASIC and is designed to teach the names of the teeth. The PAINT command is used to "blink" certain teeth for emphasis. The program draws the teeth on the screen in high-resolution graphics with the names next to the appropriate teeth. After you know the names of the teeth, press ENTER and the labels clear. The names will be reprinted in random order. For the quiz, certain teeth will "blink" and you must press the number of the correct answer for the name of the teeth. The teeth are chosen in a random order.

You need to be careful in planning the PAINT commands in your programs so the color will fill the area desired yet not leak out. In this program the molars consist of three teeth on each side, but there are gaps between the teeth with the border color so the paint will fill all three teeth with one command.

In order to label the teeth, the names are "drawn" on the screen. The subroutine in line 1630 reads through the DATA statements to find the appropriate DRAW instructions for each letter of a message M\$.

There are several ways to blink colors. In this program, the PAINT command is used. To determine which teeth to blink, the x and y coordinates of the teeth to blink are stored in arrays. X(I), Y(I) are the coordinates for teeth on the left half, and W(I), Y(I) are the coordinates for teeth on the right half. The procedure is in lines 1170-1200.

Line Numbers	Explanation
120-180	Clear screen; print title screen.
190-280	Initialize variables. N\$ is the name of the teeth, X and W are column coordinates, and Y the row coordinates for each type of tooth for use in PAINT commands to blink the teeth.
290	Randomize choices.
300	Delay for title screen.

## Programs

---

310-380	Print instruction screen.
390-420	Wait for user to press ENTER; clear screen.
430-440	Clear graphics screen in medium resolution.
450-490	Draw gum area around teeth.
500-870	Draw and label teeth.
880-920	Print message to press ENTER and wait.
930-980	Clear labels.
990-1010	Print "NAME THE TEETH".
1020-1040	Set W\$ array elements equal to N\$ array elements.
1050-1130	Randomly list names of teeth.
1140-1270	Perform quiz.
1150-1160	Randomly choose teeth which have not previously been chosen.
1170-1210	Blink teeth while waiting for response.
1220-1240	If answer is incorrect, play "uh-oh" and return for another response.
1250-1270	If answer is correct, play arpeggio; set A(I)=0 so teeth won't be chosen again; go to next problem.
1280-1320	Print option to try again and branch appropriately.
1330-1360	If response is Y for yes, clear printing and then branch to beginning of quiz. Teeth will be chosen in a different order.
1370-1620	DATA for drawing letters and characters on graphics screen.
1630-1700	Subroutine for drawing message on medium-resolution screen.
1710	Return to text screen.
1720	End.

### Program 23. Learn the Teeth

```
100 REM TEETN
110 REM TRS 80 CC 16K EXTENDED BASIC
120 CLS
130 PRINT @102, "*****"
140 PRINT @134, "*":PRINT @152, "*"
150 PRINT @166, "* LEARN THE TEETH *"
160 PRINT @198, "*":PRINT @216, "*"
170 PRINT @230, "*****"
180 PRINT
190 N$(1)="CENTRAL INCISORS"
200 X(1)=50:W(1)=60:Y(1)=46
210 N$(2)="LATERAL INCISORS"
220 X(2)=42:W(2)=70:Y(2)=46
```

```

230 N$(3)="CUSPIDS"
240 X(3)=36:W(3)=76:Y(3)=50
250 N$(4)="BICUSPIDS"
260 X(4)=32:W(4)=80:Y(4)=56
270 N$(5)="MOLARS"
280 X(5)=30:W(5)=82:Y(5)=70
290 R=RND(-TIMER)
300 FOR D=1 TO 500:NEXT
310 CLS:PRINT
320 PRINT " YOU WILL SEE A DIAGRAM OF"
330 PRINT " THE TEETH WITH THE NAMES"
340 PRINT " OF THE TEETH.":PRINT
350 PRINT " WHEN YOU KNOW THE NAMES,"
360 PRINT " PRESS <ENTER>.:PRINT
370 PRINT " THE LABELS WILL CLEAR AND"
380 PRINT " YOU WILL BE GIVEN A QUIZ."
390 PRINT @496,"PRESS <ENTER>";
400 A$=INKEY$:IF A$="" THEN 400
410 IF ASC(A$)<>13 THEN 400
420 CLS
430 PMODE 3,1:PCLS
440 SCREEN 1,0
450 CIRCLE(56,88),40,4,1.5,.5,1
460 CIRCLE(56,80),12,4,1.5,.5,1
470 CIRCLE(84,80),16,4,1,.1,.5
480 CIRCLE(30,80),16,4,1,0,.45
490 PAINT(56,32),4,4
500 CIRCLE(50,48),6,1,1.5,.5,1
510 CIRCLE(60,48),6,1,1.5,.5,1
520 COLOR 1,1
530 LINE(46,48)-(66,48),PSET
540 PAINT(50,46),2,1
550 PAINT(60,46),2,1
560 DRAW "C3;BM64,34;E12;R4"
570 M$="CENTRAL INCISORS"

580 GOSUB 1630
590 DRAW "C1;BM44,48;G6U8E2R2F2D2D2F2"
600 DRAW "BM68,48;F6U8H2L2G2D2G2"
610 PAINT(42,46),2,1
620 PAINT(70,46),2,1
630 DRAW "C3;BM74,44;E4R6"
640 M$="LATERAL INCISORS"
650 GOSUB 1630
660 DRAW "C1;BM40,50;G6H4U2E3R2F4"
670 DRAW "BM72,50;F6E4U2H3L2G4"
680 PAINT(36,50),2,1
690 PAINT(76,50),2,1
700 DRAW "C3;BM84,50;R10;BM+2,4"
710 M$="CUSPIDS"

```

## Programs

---

```
720 GOSUB 1630
730 DRAW "C1;BM34,54;L4G2D2F2L2G2D2F2R6E3U2H
3E2U2H2"
740 DRAW "BM78,54;R4F2D2G2NL2R2F2D2G2L6H3U2E
3H3U2E2"
750 PAINT (32,56),2,1
760 PAINT (80,56),2,1
770 DRAW "C3;BM92,60;R8;BM+4,4"
780 M$="BICUSPIDS"
790 GOSUB 1630
800 A$="L6G2D2F2NR2G2D2F2NR2G2D2F2R2F1R3E4U2
H2E2U2H2NL2E2U2H2"
810 DRAW "C1;BM32,68;XA$;"
820 DRAW "BM84,68;XA$;"
830 PAINT (30,70),2,1
840 PAINT (82,70),2,1
850 DRAW "C3BM92,76;R8;BM+4,4"
860 M$="MOLARS"
870 GOSUB 1630
880 M$="PRESS ENTER"
890 DRAW "BM128,144;"
900 GOSUB 1630
910 A$=INKEY$:IF A$="" THEN 910
920 IF ASC(A$)<>13 THEN 910
930 COLOR 1,1
940 LINE(128,144)-(220,134),PSET,BF
950 LINE(68,28)-(200,15),PSET,BF
960 LINE(82,44)-(200,34),PSET,BF
970 LINE(96,46)-(170,84),PSET,BF
980 DRAW "BM88,50;R10"
990 COLOR 3,1
1000 M$="NAME THE TEETH"
1010 DRAW "BM80,112;":GOSUB 1630
1020 FOR C=1 TO 5
1030 W$(C)=N$(C)
1040 NEXT C
1050 FOR C=1 TO 5
1060 I=RND(5)
1070 IF W$(I)="" THEN 1060
1080 M$=STR$(C)+" "+W$(I)
1090 DRAW "BM100,120"
1100 FOR II=1 TO C:DRAW "BM+0,+12;":NEXT II
1110 GOSUB 1630
1120 A(I)=C:W$(I)=""
1130 NEXT C
1140 FOR C=1 TO 5
1150 I=RND(5)
1160 IF A(I)=0 THEN 1150
1170 PAINT(X(I),Y(I)),3,1
```

```

1180 PAINT(W(I),Y(I)),3,1
1190 PAINT(X(I),Y(I)),2,1
1200 PAINT(W(I),Y(I)),2,1
1210 A$=INKEY$:IF A$="" THEN 1170
1220 IF ASC(A$)=A(I)+48 THEN 1250
1230 PLAY "O2;L32;EC"
1240 GOTO 1170
1250 PLAY "O2;L32;CEA;O3;L16;C"
1260 A(I)=0
1270 NEXT C
1280 M$="TRY AGAIN?Y/N"
1290 DRAW "BM4,188;"
1300 GOSUB 1630
1310 A$=INKEY$:IF A$="N" THEN 1710
1320 IF A$<>"Y" THEN 1310
1330 COLOR 1,1
1340 LINE(4,120)-(240,190),PSET,BF
1350 COLOR 3,1
1360 GOTO 1020
1370 DATA A,"U4E2F2D2NL4D2;BM+4,0"
1380 DATA O,"BM+2,0;H1U4E1R2F1D4G1L2;BM+7,0"
1390 DATA R,"U6R3F1D1G1L2NL1F3;BM+4,0"
1400 DATA C,"BM+1,0;H1U4E1R2F1;BM+0,4;G1L2;B
M+6,0"
1410 DATA D,"U6R3F1D4G1L3;BM+8,0"
1420 DATA E,"NR4U3NR2U3R4;BM+3,6"
1430 DATA I,"BM+2,0;NU6;;BM+4,0"
1440 DATA L,"NU6;R4;BM+3,0"
1450 DATA N,"U6F2D1F2D1NU6;BM+4,0"
1460 DATA T,"BM+2,0;U6NL2R2;BM+3,6"
1470 DATA S,"BM+0,-1;F1R2E1U1H1L2H1U1E1R2F1;
BM+4,5"
1480 DATA H,"U3NU3R4NU3D3;BM+3,0"
1490 DATA P,"U6R3F1D1G1L3;BM+7,3"
1500 DATA " ", "BM+7,0"
1510 DATA U,"BM+0,-1;NU5F1R2E1U5;BM+3,6"
1520 DATA M,"U6F2ND1E2D6;BM+4,0"
1530 DATA B,"U6R3F1D1G1NL3F1D1G1L3;BM+7,0"
1540 DATA 1,"BM+2,0;U6G1;BM+6,5"
1550 DATA 2,"NR4U1E1R1E2U1H1L2G1;BM+7,5"
1560 DATA 3,"BM+0,-1;F1R2E1H2E2H1L3;BM+7,6"
1570 DATA 4,"BM+3,0;U2NR1L3U1E3D3;BM+4,3"
1580 DATA 5,"BM+0,-1;F1R2E1U2H1L3U2R4;BM+3,6
"
1590 DATA G,"BM+1,0;H1U4E1R2F1;BM+0,2;NL1D2G
1L2;BM+6,0"
1600 DATA Y,"BM+0,-6;D2F2ND2E2U2;BM+3,6"
1610 DATA "?", "BM+0,-5;E1R2F1D1G2;BM+0,1;D1;BM
+10,0"

```

```
1620 DATA /,"U1E4U1;BM+3,6"  
1630 FOR L=1 TO LEN(M$)  
1640 RESTORE  
1650 FOR LL=1 TO 26  
1660 READ L$,A$  
1670 IF L$=MID$(M$,L,1) THEN DRAW A$:GOTO 16  
90  
1680 NEXT LL  
1690 NEXT L  
1700 RETURN  
1710 SCREEN 0,1  
1720 END
```

### New England States

Geography is another topic that can be taught with computers. "New England States" draws and labels the states. Once the student knows the names of the states, pressing ENTER will clear the labels. The states will then be numbered and listed. One at a time, in a random order, a small square will blink on the state to be identified. Press the number of the correct answer.

While the computer is waiting for the student's answer, the square is blinked by using the LINE command with the BF (Box Filled) option, first PRESET (background color) then PSET (color of the original state).

To draw the letters on the screen, an L\$ array is used. L\$(C,0) is the letter or symbol, and L\$(C,1) is the drawing instruction, where C can be 0 to 26 for the symbols. The subroutine in lines 1160-1200 checks through the symbols to draw the appropriate one.

Line Numbers	Explanation
120	Clear screen. The text screen will be Color 2.
130	Define N\$ to be a string of 23 asterisks.
140-200	Print title screen. Note: The print items end with semicolons so the printing will be green on the yellow screen. Without the semicolons, the whole line would be green.
210	DIMENSION variables. L\$ will contain the letters or symbols with drawing instructions; S is the number of each state; A(C,1) and A(C,2) are coordinates for the blinking square, and A(C,3) is the color for each of the six states.
220	Define state numbers in S array.



230-240	Read data to define A array of coordinates and colors for each state.
250-520	Read data to define L\$ array of symbol and drawing instruction.
530	Delay.
540-610	Print instructions.
620-630	Wait for user to press a key to continue.
640-820	Draw and PAINT states with labels.
830-850	Wait for user to PRESS ENTER to start quiz.
860-910	Clear names of states.
920-970	Draw names of states for answers to quiz.
980	Initialize states' numbers.
990	Randomize choices.
1000-1090	For six states, randomly choose a state which has not been chosen previously, blink a square on the state, and wait for user to press the number of the answer. If the answer is correct, play arpeggio; if the answer is incorrect, play "uh-oh" and wait for another answer. The answer must be correct to go on.
1100-1150	Print option to try again, wait for response, and branch appropriately.
1160-1200	Subroutine to draw letters on graphics screen.
1210	Clear screen and end.

### Program 24. New England States

```

100 REM NEW ENGLAND STATES
110 REM
120 CLS2:PCLS
130 N$=STRING$(23,"*")
140 PRINT @68,N$;
150 PRINT @100,"*";TAB(26);"*";
160 PRINT @132,"* IDENTIFY THE STATES (";
170 PRINT @164,"*";TAB(26);"*";
180 PRINT @196,N$;
190 PRINT @262,"NEW ENGLAND STATES";
200 PRINT
210 DIM L$(26,1),S(6),A(6,3)
220 FOR C=1 TO 6:S(C)=C:NEXT
230 FOR C=1 TO 6:READ A(C,1),A(C,2),A(C,3):N
EXT
240 DATA 128,146,4,130,120,3,142,170,2,116,1
64,3,108,86,2,184,60,4
250 FOR C=0 TO 26:READ L$(C,0),L$(C,1):NEXT C
260 DATA " ", "BM+7,0"

```

## Programs

```
270 DATA A, "U4E2F2D2NL4D2; BM+3, 0"
280 DATA C, "BM+1, 0; H1U4E1R2F1; BM+0, 4; G1L2; BM
+6, 0"
290 DATA D, "BM+1, 0U6R3F1D4G1L3; BM+7, 0"
300 DATA E, "NR4U3NR2U3R4; BM+3, 6"
310 DATA H, "U3NU3R4NU3D3; BM+3, 0"
320 DATA I, "BM+3, 0; NU6; BM+4, 0"
330 DATA L, "NU6R4; BM+3, 0"
340 DATA M, "U6F2ND1E2D6; BM+3, 0"
350 DATA N, "BM+1, 0; U6F1D1F2D1F1NU6; BM+3, 0"
360 DATA O, "BM+1, 0; H1U4E1R2F1D4G1L2; BM+6, 0"
370 DATA P, "U6R3F1D1G1L3; BM+7, 3"
380 DATA R, "U6R3F1D1G1L2NL1F3; BM+3, 0"
390 DATA S, "BM+0, -1; F1R2E1U1H1L2H1U1E1R2F1; B
M+3, 5"
400 DATA T, "BM+2, 0; U6NL2R2; BM+3, 6"
410 DATA U, "BM+0, -1; NU5F1R2E1U5; BM+3, 6"
420 DATA V, "BM+0, -6; D2F1D1F1ND1E1U1E1U2; BM+3
, 6"
430 DATA Y, "BM+0, -6; D2F2ND2E2U2; BM+3, 6"
440 DATA W, "NU6E2NU1F2U6; BM+3, 6"
450 DATA /, "U1E4U1; BM+3, 6"
460 DATA 1, "BM+1, 0; R1NR1U6G1; BM+6, 5"
470 DATA 2, "NR4U1E1R1E2U1H1L2G1; BM+7, +5"
480 DATA 3, "BM+0, -1; F1R2E1H2E1H1L3; BM+7, 6"
490 DATA 4, "BM+3, 0; U2NR1L3U1E3D3; BM+4, 3"
500 DATA 5, "BM+0, -1; F1R2E1U2H1L3U2R4; BM+3, 6"
510 DATA 6, "BM+4, -5; H1L2G1D4F1R2E1U1H1L3; BM+
7, 3"
520 DATA G, "BM+2, 0; H1U4E1R2F1; BM+0, +2; NL1D2G
1L2; BM+7, 0"
530 FOR D=1 TO 500:NEXT
540 CLS 1
550 PRINT @33, "THE NEW ENGLAND STATES WILL"
560 PRINT " BE SHOWN. WHEN YOU KNOW ALL"
570 PRINT " THE NAMES, PRESS <ENTER>."
580 PRINT:PRINT " THE LABELS WILL CLEAR AND
THE"
590 PRINT " NAMES OF THE STATES WILL BE"
600 PRINT " BE LISTED. PRESS THE CORRECT"
610 PRINT " NUMBER AS THE STATE BLINKS."
620 PRINT:PRINT " PRESS ANY KEY TO START";
630 A$=INKEY$: IF A$="" THEN 630
640 PMODE 3, 1:SCREEN 1, 0:PCLS
650 DRAW "BM146, 60; R6; E4; U6; E4; U6; E6; U6; E6; U
4; E2; U4; R4; D3; R6; E6; R4; F6; D4; F6; D25; F2; R
6; F4; D8; F4; R4; D5; G2; L6; G12; L6; G12; L10; G4
; D2; G4; D2; G4; D2; G4; D2; G4; H12; U55"
660 DRAW "L4; G2; L2; G2; D4; L2; D14; G14; D4; G4; D4
; G2; D22; F4; R30; E2; R8; E2; U4"
```

```

670 DRAW "BM 136,64;L44;D72;R25"
680 DRAW "L25;D18;R55;F4;D8;F4;D3;F4;R2;E8;R
3;D3;E10;U6;H4;D4;G4;L4;H2;U4;H2;L4;H2;U
4;E4;U2;H2;U4;L2"
690 DRAW "BM 92,154;D18;F4;D2;G6;R6;E2;R18;E
2;R18;U26"
700 DRAW "D26;R2;E2;R8;E4"
710 PAINT (176,28),4,0
720 N$="MAINE":DRAW "BM216,32;":GOSUB 1160
730 PAINT (144,62),3,0
740 N$="NEW HAMPSHIRE":DRAW "BM168,130;":GOS
UB 1160
750 PAINT (130,66),2,0
760 N$="VERMONT":DRAW "BM32,74;":GOSUB 1160
770 PAINT (116,142),4,0
780 N$="MASSACHUSETTS":DRAW "BM0,146;":GOSUB
1160
790 PAINT (94,156),3,0
800 N$="CONNECTICUT":DRAW "BM10,172;":GOSUB
1160
810 PAINT (140,156),2,0
820 N$="RHODE ISLAND":DRAW "BM136,192;":GOSU
B 1160
830 FOR C=1 TO 10:A$=INKEY$:NEXT C
840 A$=INKEY$:IF A$="" THEN 840
850 IF ASC(A$)<>13 THEN 840
860 LINE(216,32)-(250,24),PRESET,BF
870 LINE(168,130)-(255,122),PRESET,BF
880 LINE(32,74)-(80,66),PRESET,BF
890 LINE(0,146)-(88,138),PRESET,BF
900 LINE(10,172)-(88,164),PRESET,BF
910 LINE(136,192)-(250,184),PRESET,BF
920 N$="1 MASSACHUSETTS":DRAW "BM0,10;":GOSU
B 1160
930 N$="2 NEW HAMPSHIRE":DRAW"BM0,22;":GOSUB
1160
940 N$="3 RHODE ISLAND":DRAW"BM0,34;":GOSUB
1160
950 N$="4 CONNECTICUT":DRAW"BM0,46;":GOSUB 1
160
960 N$="5 VERMONT":DRAW"BM0,58;":GOSUB 1160
970 N$="6 MAINE":DRAW"BM0,70;":GOSUB 1160
980 FOR C=1 TO 6:A(C,0)=S(C):NEXT
990 R=RND(-TIMER)
1000 FOR I=1 TO 6
1010 R=RND(6):IF A(R,0)=0 THEN 1010
1020 LINE(A(R,1),A(R,2))-(A(R,1)+4,A(R,2)+4)
,PSET,BF
1030 COLOR A(R,3):LINE(A(R,1),A(R,2))-(A(R,1
)+4,A(R,2)+4),PSET,BF

```

## Programs

---

```
1040 A$=INKEY$:IF A$=""THEN 1020
1050 IF VAL(A$)=R THEN 1070
1060 PLAY"O2;L16;EC":GOTO 1020
1070 PLAY"O2;L16;CEG03L8C"
1080 A(R,0)=0
1090 NEXT I
1100 N$="TRY AGAIN":DRAW"BM170,140;":GOSUB 1
    160
1110 N$="Y/N":DRAW"BM190,150;":GOSUB 1160
1120 A$=INKEY$:IF A$="N"THEN 1210
1130 IF A$<>"Y" THEN 1120
1140 LINE(170,150)-(255,132),PRESET,BF
1150 GOTO 980
1160 FOR C=1 TO LEN(N$)
1170 FOR I=0 TO 26
1180 IF L$(I,0)=MID$(N$,C,1) THEN DRAW L$(I,
    1):GOTO 1200
1190 NEXT I
1200 NEXT C:RETURN
1210 PCLS:CLS:END
```

**Four**

---

**Appendices**








## Cassette Recorder

Small gray plug goes into REM jack.  
 Large gray plug goes into AUX jack.  
 Large black plug goes into EAR jack.

## Color Codes

- 0 Black
- 1 Green
- 2 Yellow
- 3 Blue
- 4 Red
- 5 Buff
- 6 Cyan
- 7 Magenta
- 8 Orange

## Color Set Chart

PMODE	SCREEN	COLORS AVAILABLE	RESOLUTION
4	0	Black/Green	
	1	Black/Buf	
3	0	Green/Yellow/Blue/Red	
	1	Buf/Cyan/Magenta/Orange	
2	0	Black/Green	
	1	Black/Buf	
1	0	Green/Yellow/Blue/Red	
	1	Buf/Cyan/Magenta/Orange	
0	0	Black/Green	
	1	Black/Buf	

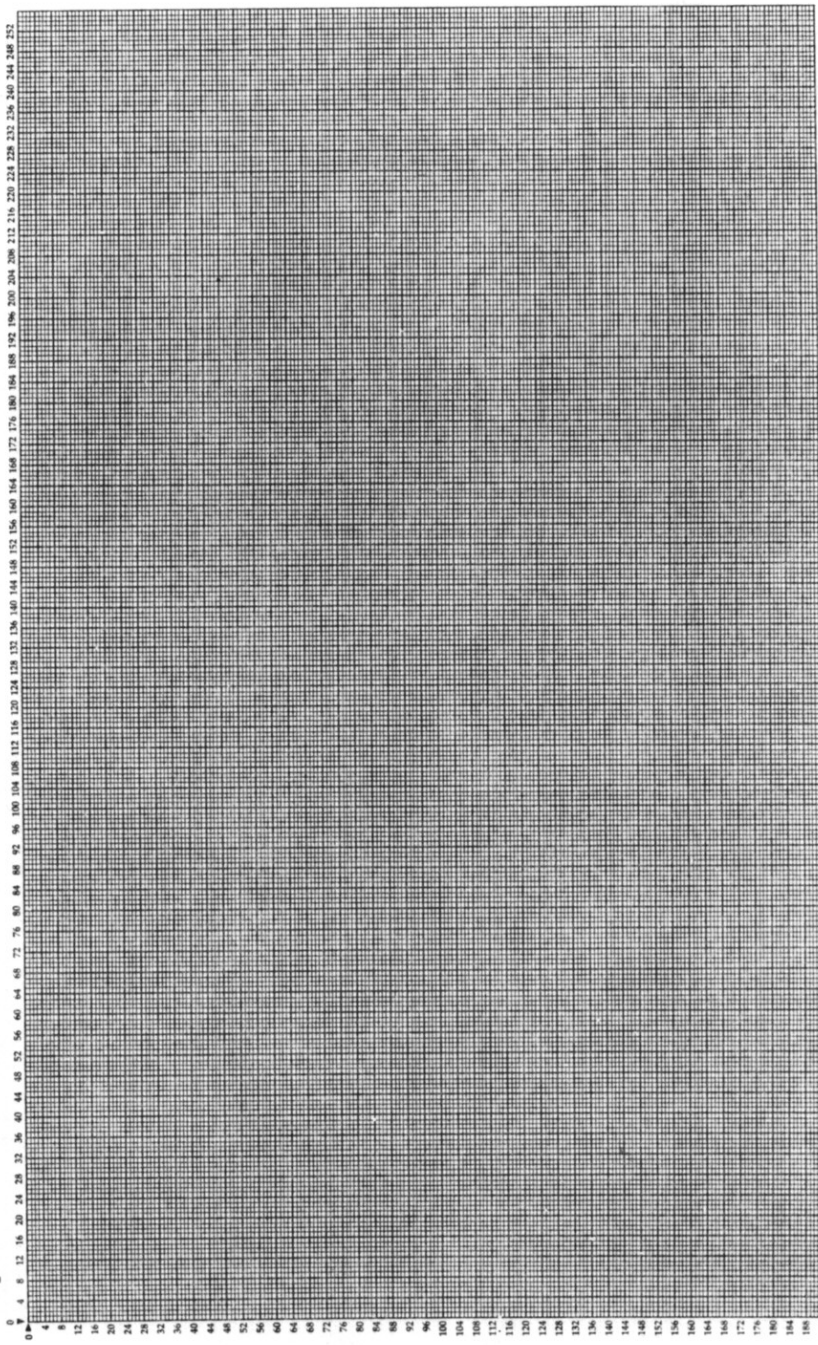
## ASCII Character Codes

Character	Decimal Code	Character	Decimal Code
BREAK	3	>	62
-	8	?	63
-	9	@	64
	10	A	65
CLEAR	12	B	66
ENTER	13	C	67
SHIFT ←	21	D	68
SPACE	32	E	69
!	33	F	70
"	34	G	71
#	35	H	72
\$	36	I	73
%	37	J	74
&	38	K	75
'	39	L	76
(	40	M	77
)	41	N	78
*	42	O	79
+	43	P	80
,	44	Q	81
-	45	R	82
.	46	S	83
/	47	T	84
0	48	U	85
1	49	V	86
2	50	W	87
3	51	X	88
4	52	Y	89
5	53	Z	90
6	54	SHIFT ↓ [	91
7	55	SHIFT CLEAR	92
8	56	SHIFT → ]	93
9	57	↑	94
:	58	SHIFT ↑	95
;	59	a	97
<	60	b	98
=	61	c	99



Character	Decimal Code	Character	Decimal Code
d	100	o	111
e	101	p	112
f	102	q	113
g	103	r	114
h	104	s	115
i	105	t	116
j	106	u	117
k	107	v	118
l	108	w	119
m	109	x	120
n	110	y	121
		z	122

## Graphics Screen Worksheet (256 × 192)



# Appendices

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

**PRINT @ Screen Locations**



# INDEX

- ABS function 3
- AND function 3-4
  - with GET 27
  - with PUT 70
- animation, using SET and RESET 73
- arrays, DIM statement and 17-18
- ASC function 4
- ASCII code 6, 39
  - chart 172-73
  - STRING\$ and 85
- ASCII text file, saving 78
- ATN function 4-5
  - with COS 13
- AUDIO command 5-6, 43
- baud rate, varying 104
- "Boxes" program 113-14
- "Buying Items" program 154-56
- "Carriage Return — Line Feed" program 103-4
- cassette recorder 10
  - OPEN and 48
  - plugs for 171
  - saving data on 13-14
- character editing 67
- CHR\$ function 6-7
- CIRCLE command 7-8, 12
  - in "William Shakespeare" program 110
- "Circles" program 112
- CLEAR command 8-9
  - and MEM 41
  - conserving memory with 97
  - NEW and 98
- CLOAD command 9-10
  - different from LOAD 41
- CLOSE command 10-11
- CLS command 11
- colons, for multiple program statements 97
- color codes 171
- COLOR command 11-12
  - and PSET 69
- color set chart 171
- communication channel 10
- computer-aided instruction 137-68
- conserving memory 96-97
  - by eliminating spaces 97
  - with CLEAR 97
  - with low line numbers 96
  - with PCLEAR 97
- CONT command 12-13
  - with STOP 85
- COS function 13
- CSAVE command 13-14
- cursor, position of 60
- DATA statement 14-15
- DEF FN statement 15-16
- DEL command 16-17
- "Dice" program 114-15
- DIM statement 17-18
  - with MEM 41
  - with PUT 70
- disk drive 10
  - OPEN and 48
- dotted note, with PLAY 55
- DRAW command 18-21
  - options explained 19-20
  - sample program 20-21
  - variables and 98
- "Earning Money" program 157-59
  - discussion 156-57
- EDIT statement 21-22
  - options and procedures 21-22
- ELSE 22-23, 31
  - with IF-THEN 23
- END statement 23
  - STOP instead of 23
- EOF statement 24
  - with OPEN 48
- EXP function 24-25
  - with LOG 24-25
- "Find All Factors" program 120
- FIX function 25
  - and INT 34-35
- "Flag program" 108-9
- FOR statement 25-26, 44
- GET statement 27-28
  - action graphics and 27
  - with DIM 17
  - with PUT 70
- GOSUB command 28
  - with RETURN 28
- GOTO command 29
- graphics programs 108-15
- graphics screen worksheet 174
- "Greatest Common Factor" program 121-22
- halting program 77
- HEX\$ function 29-30
- "Homework Helper — Division" program 117-19
  - discussion 116-17

- "Homework Helper — Fractions"
  - program 128-35
  - discussion 124-28
- IF statement 30-31
  - with AND 3
- INKEY\$ statement 31-32
  - with null strings 31
- INPUT command 32
  - different from LINE INPUT 39
- INPUT# command 32-33
  - with cassette 32-33
  - with CLOSE 33
  - with disk 33
  - with OPEN 33, 48
- input file mode 10
- INSTR function 33-34
- "Inter Sort" program 106-7
- INT function 34-35
  - with FIX 34-35
- JOYSTK function 35-36
  - with PEEK 35
- keyboard 95
- "Learn the Teeth" program 160-64
  - discussion 159-60
  - DRAW command in 159
  - PAINT command in 159
- "Least Common Multiple" program 123-24
- LEFT\$ function 36-37
- LEN function 37
- LET statement 37-38
- "Letters" program 105-6
  - discussion 104-5
- LINE command 12, 38-39
  - in "Boxes" program 113
  - with COLOR 38
  - with PRESET 38
  - with PSET 38, 69
  - with RESET 38
- LINE INPUT command 39-40
  - differences from INPUT 39
- "Lines" program 113
  - discussion 112-13
- LIST command 40
- LLIST command 40-41
- LOAD command 41
  - different from CLOAD 41
- LOG function 41-42
  - with EXP 24-25
- mathematics programs 115-37
- MEM statement 42-43
  - with CLEAR 41
  - with DIM 41
  - with PCLEAR 41
- MID\$ function 43
- "Min/Max Sort" program 106-7
- "Min Sort" program 106-7
- MOTOR command 43-44
- NEW command 9, 44, 98
- "New England States" program 165-68
  - discussion 164-65
- NEXT 44-45
  - and FOR 25-26
- note length, in PLAY 55
- NOT operator 45
  - with GET 27
  - with PUT 70
- null strings 31
- octave, in PLAY 55
- ON-GOSUB statement 45-47
- ON-GOTO statement 47-48
- OPEN command 10, 48-49
  - and PRINT# 64
- OR operator 49-50
  - with GET 27
  - with PUT 70
- output file mode 10
- PAINT command 50-51
  - with PMODE 50
- pause, in PLAY 55
- PCLEAR command 50-51
  - and MEM 41
  - and PMODE 57
  - saving memory with 97
  - with NEW 98
- PCLS command 52-53
- PCOPY command 53-54
  - with PCLS 53
  - with PMODE 53
- PEEK function 54-55
  - with joysticks 54
  - with JOYSTK 35
- PLAY command 55-57
  - options 55-56
  - variables and 98
- PMODE 27, 57-58
  - and GET 70
  - and NEW 98
  - and PAINT 50
  - and PCLEAR 57
  - and SCREEN 57
  - with RESET 38
- POINT function 58-59
- POKE command 59, 96
  - for teletype 104
- POS function 60
- PPOINT function 60-61
  - with PMODE 60
  - with PSET 60
- preset 27, 61
- PUT and 70
  - relation to RESET 61

- "Prime Factors" program 120-21
- PRINT@ statement 62-64
  - worksheet 63
- printer 10
  - other than Radio Shack 103
- printhead, position of 60
- PRINT# statement 64-65
  - with OPEN 64
  - with WRITE# 64
- PRINT MEM command 9, 42
- print separators 62
- PRINT statement 11, 62
- PRINT TAB statement 66
- PRINT USING statement 66-69, 75
  - formats 67
- PSET 2, 69
- PUT statement 70
  - with DIM 17
  - with GET 27
  - with PSET 69
- READ statement 71
  - with DATA statement 14, 71
  - with RESTORE 71
- REM statement 71-72
- RENUM command 72-73, 97
- RESET command 9, 73-74
  - animation using 73
  - relation to PRESET 61
  - with SET 73
- RESTORE statement 74-75
  - with DATA statement 14, 74
  - with READ 74
- RETURN 75-76
  - with GOSUB 28, 75
- RIGHTS function 76
- RND function 77
  - TIMER and 77, 98
- RUN command 77-78
  - with BREAK key 77
  - with CONT 77
  - with DATA statement 14
- SAVE command 78
  - options 78
  - relation to CSAVE 78
- SCREEN 10, 79
  - and PMODE 57
- screen color, with CLS 11
- SCREEN statement, PAINT and 50
- SET 79
  - animation using 73
  - with PUT 70
  - with RESET 73
- SGN function 80-81
  - branching with 80
- "Shell Sort" program 106-7
- "Simultaneous Equations" program 136-37
  - discussion 135
- SIN function 81-82
  - ATN and 81-82
- SKIPF command 82
- sorting
  - discussion 106
- sound effects 99
- SOUND statement 82-83
- SQR function 83-84
- STEP statement 13, 26, 84-85
  - in FOR-NEXT loops 84-85
- STOP command 78, 85
  - with CONT 85
- STR\$ function 86
- strings, size limitation 98
- STRING\$ function 85-86
- TAB function 86-87
- TAN function 87-88
- teletype, POKEs for 104
- tempo, in PLAY 55
- THEN 88
  - with ELSE 88
  - with IF 30-31, 88
- TIMER function 88-89
  - with RND 77
- TROFF command 89-90
- TRON command 89-90
- "Typing Trainer" program 148-54
  - discussion 146-48
- "Typing Unit 1" program 140-46
  - discussion 138-39
- USING function 90
  - see also* PRINT#; PRINT USING
- utility programs 103-8
- VAL function 91
- variables 98
- volume, in PLAY 55
- "William Shakespeare" program 110-12
- worksheet
  - graphics screen 174
  - PRINT@ 63, 175
  - SET-RESET 176
  - WRITE# statement 64