Volume 4, Issue 6 **JUNE, 1982**

Price $1.50

TRS-80® Microcomputer News

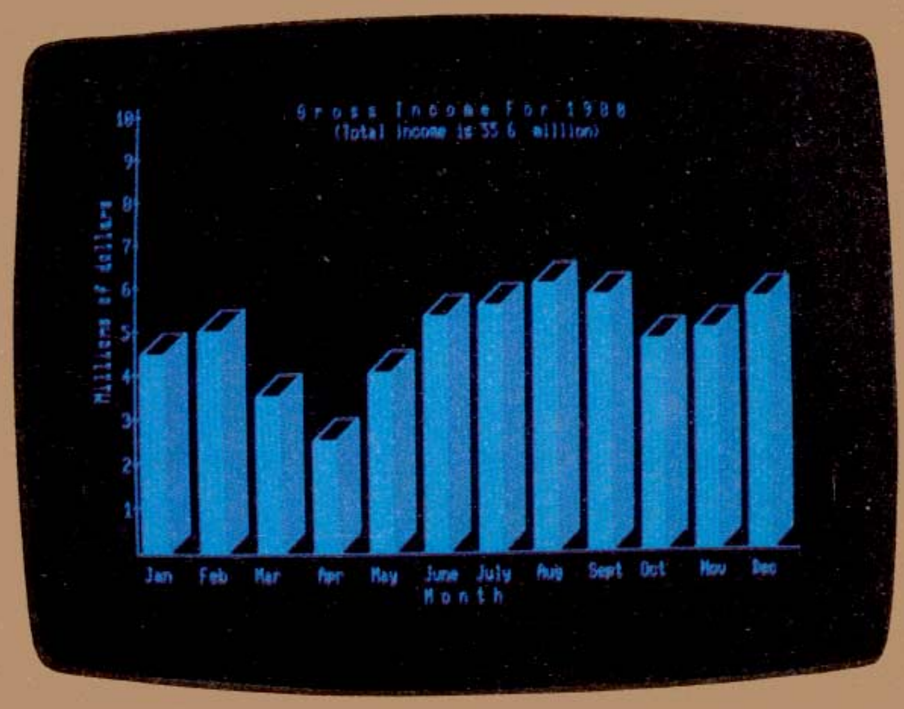
Information Published for TRS-80 users.

Graphics! Graphics! Graphics!

Model II High Resolution

PC-2 Plotter Printer

Programming with the TRS-80 Digitizer



Fort Worth Scene

GRAPHICS, GRAPHICS, GRAPHICS

Graphics is a subject that no matter how much has been written about it, there always seems to be something more to be said. Judging by the large number of graphic programs that we receive almost daily, our readers share that opinion. At one time or another nearly every Microcomputer publication has devoted at least one issue (and sometimes more) to this diverse subject. We will not be the exception.

We have tried to squeeze user programs into every available space so that you could enjoy sharing in the discoveries of other TRS-80 users. In an effort to include something for everyone, the content runs the gamut from introductory graphics to some pretty sophisticated graphics programs.

PRODUCTS WITH GRAPHICS CAPABILITIES

It has been an interesting few weeks for us as we've gotten to explore the capabilities of the Model II graphics board, the digitizer, and the four color plotter on the PC-2. We had hoped to include information on the six color plotter but time did not permit us to do that. Maybe sometime in the future.

CHOOSING USER PROGRAMS

Probably the most difficult thing we had to do in preparing this issue was to choose the material for it. There was so much material to choose from and we had several lively discussions on what would be included in the graphics issue and what would not. Right now there are enough terrific unused programs to fill several more issues. We have included a variety of programs and articles so there should be something to pique your interest in this issue.

Color computer owners will enjoy the graphic character program in the Color Computer section. It has intrigued us ever since we first loaded it in, and we hope that you have as much fun with it as we have.

AND MORE . . .

We tried to keep to the subject of graphics in all the sections, but keep in mind some columns lend themselves to graphics more than others. Of course Bill Barden adds another to his series of excellent articles on the 6809 for which we will be eternally grateful, even if we did have to fork over a free subscription to Microcomputer News to get him to do it. We feel that the return on our investment more than justifies our expenditure.

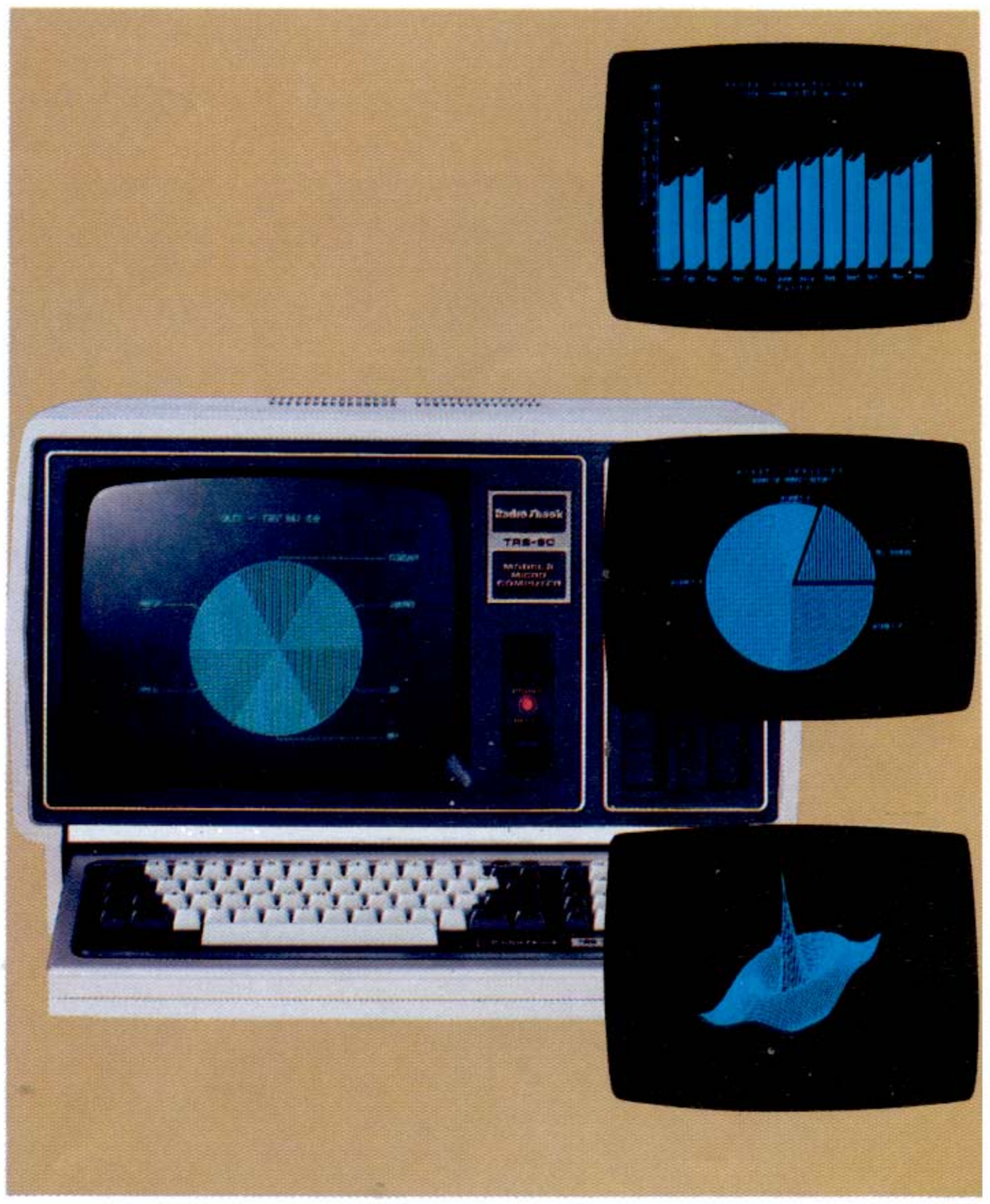
You will find the long promised Model III head cleaner program in this issue. Due to a miscommunication we published another Model I head cleaner program in May when what we really wanted to do was publish one for the Model III. Sorry about that! We have been taken to task by enough readers to know that it was imperative to get the Mod III head cleaner program into the June issue.

If you're a sports fan, you might enjoy playing the football game in the feature story. Who knows, even if you're not a sports fan you may enjoy it. Anyway, kick it around a little bit before forming any opinions.

A lot of work has gone into this issue—by all our contributors (including readers) as well as by the Microcomputer News staff. Our thanks to all who made contributions. Be sure and read View from the 7th Floor, not just because it's nice to be vindicated occasionally, but if you're like me, you'll get a chuckle from his reference to a previously seen wall plaque.

OOPS

We goofed. Quite unintentionally, we used a trademarked phrase in the March, 1982 article on Model 16. The term Maxi-Micro® is a trademark of Max Ule and Company, Inc. Our apologies for the misuse.



IMPORTANT NOTICE: The July issue of TRS-80 Microcomputer News is scheduled to be delivered in late July or early August, and will be labeled July/August. This change will in no way change the number of issues that you are supposed to receive.

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single six month subscription is available free to purchasers of new TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, Canada and APO or FPO addresses. Subscriptions to other addresses are not available.

The subscription rate for renewals and other interested persons with U.S., APO or FPO addresses is twelve dollars ($12.00) per year, check or money order. Single copies of the Microcomputer News may be purchased from Radio Shack Computer Centers or Computer Departments for $1.50 suggested retail each. The subscription rate for renewals and other interested persons with Canadian addresses is fifteen dollars ($15.00) per year, check or money order in U.S. funds. All correspondence related to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76113-2910.

Retail Prices in this newsletter may vary at individual stores and dealers. The company cannot be liable for pictorial and typographical inaccuracies.

Back issues of Microcomputer News prior to January, 1981 are available through your local Radio Shack store as stock number 26-2115 (Suggested Retail Price $4.95 for the set). Back issues of 1981 copies are not available.

The TRS-80 Newsletter welcomes the receipt of computer programs, or other material which you would like to make available to users of TRS-80 Microcomputer systems. In order for us to reprint your submission, you must specifically request that your material be considered for reprinting in the newsletter and provide a notice that you retain no copyrights or other exclusive rights in the material. This assures that our readers may be permitted to recopy and use your material without creating any legal hassles.

Material may be submitted by mail to P.O. Box 2910, Fort Worth, Texas 76113-2910, or through CompuServe. The Microcomputer News' CompuServe user ID number is 70007,535.

Notes to Program Users:

Programs published in the Microcomputer News are provided as is, for your information. While we make reasonable efforts to ensure that the programs we publish here work as specified, Radio Shack can not assume any liability for the accuracy either of the programs themselves, or of the results provided by the programs.

Further, while Microcomputer News is a product of Radio Shack, the programs and much of the information published here are not Radio Shack products, and as such cannot be supported by our Computer Customer Service group. If you have questions about a program in the Microcomputer News, your first option is to write directly to the author of the program. When possible, we are not including author's addresses to facilitate communications. If the address is not published, or if you are not happy with the response you get, please write us here at Microcomputer News. We will try (given the limited size of our staff) to find an answer to your question and, in many cases will publish the answer in an upcoming issue of Microcomputer News.

Comments on our program listing style:

In order to make the program listings we publish easier to read, we have adopted a style of inserting spaces to enhance readability, and we separate each program statement onto a separate line. While these techniques increase program readability, they also require more memory, and may execute more slowly than the original program did.

When you are entering a program for your own use, you may wish to eliminate many of the extra blanks (see your owners manual for required blanks), and you should certainly move multiple statements up to a single line where possible.

Trademark Credits

CompuServe™	CompuServe, Inc.
DIF™	Software Arts, Inc.
Dow Jones™ NEWS/RETRIEVAL Service®	Dow Jones & Co., Inc.
ReformaTTer™	Microtech Exports
Time Manager™	Image Producers
VisiCalc®	Personal Software, Inc.
Program Pak™	Tandy Corporation
SCRIPSIT™	Tandy Corporation
TRSDOS™	Tandy Corporation
TRS-80®	Tandy Corporation

Volume 4, Issue 6 **JUNE, 1982**

Price $1.50

TRS-80® Microcomputer News

Information Published for TRS-80 users

Contents:

Assembly Language Programming ... 13
by William Barden, Jr.

Color Computer
Programs
Draw for 4K by David Shapiro ... 47
Graphic Height Correction by R. A. Esquivel ... 38
High Resolution Graphics Characters by George Fraser ... 41
Rotational Grafx by John Ennis ... 45
Texture by George Fraser ... 44

Computer Customer Service ... 18
Graphic Basics

Data Bases
CompuServe ... 20
New E-Mail Service
Dow Jones ... 22
You Can Quote Us

Educational Products
Graphics Creation in Author I ... 29
Computer Literacy ... 30
The Computer Future Is Now ... 31

Feature Story ... 10
Football for Summer

Fort Worth Scene ... 2

Model I/III
Bugs, Errors, and Fixes
Accounts Payable (26-1554) ... 33
Accounts Receivable (26-1555) ... 33
Inventory Control (26-1553) ... 33
Model III TRSDOS (26-312) ... 33
Product Line Manager's Page ... 25
Three Dimensional Graphics Package by Charles Burton
Programs
Concentric Circles by Stephen Havens ... 32
Graphic Keys by Robby Bennett ... 17
Disk Drive Head Cleaner by Robert Brown ... 17
Special Character Mode by Randy Butts ... 33

Model II
Bugs, Errors, and Fixes
Accounts Payable (26-4505) ... 37
General Ledger (26-4501) ... 36
Inventory Management (26-4502) ... 36
Payroll (26-4503) ... 36
Scripsit (26-4531) ... 37
TRSDOS (26-4910) ... 37
TRSDOS-HD (26-4150) ... 38
Product Line Manager's Page ... 34
High Resolution Business Graphics for Models II and 16
Programs
Histogram by Kenneth Willoughby ... 40

Notes on Previous Newsletters
June, 1981 PCLEAR ... 38
February, 1982
Bouncing Box ... 39
No Gaps in Names ... 39
Typewriter ... 40
March, 1982
Subdestroyer ... 40

Peripherals
Product Line Manager's Page ... 23
Digitizer Graphics Subroutines

Pocket Computer
Product Line Manager's Page ... 5
Printing Features of the PC-2

View from the 7th Floor by Jon Shirley ... 4

All prices in TRS-80 MICROCOMPUTER NEWS are in U.S. Funds.

View From the Seventh Floor

Jon Shirley
Vice President, Radio Shack Computer Merchandising

I recently received two letters, neither of which was sent to me directly, but both certainly have an interesting story to tell. One was sent to a supplier of TRS-80 add on/in equipment by a TRS-80 owner, and the other was sent to an owner by a similar supplier. I will not identify the individuals or companies involved, but the companies can be found advertising in various computer magazines.

The first letter concerns a Model III owner who attended a computer show in Los Angeles and found a booth with a show special on RAM chips, 32K for only $29! "All you have to do is open up yer Model III and stick 'em in!" Well, he did not buy at the show, but he did call them up later and finally sent in his check.

Although the company with the cheap chips was about 30 minutes away, the order took two weeks and three phone calls to be filled. Well, he installed the chips, and ?MEM said 48082 BUT the memory could not be accessed. To quote from the letter "After three weeks of almost daily phone calls to ——— / ——— and leaving messages for some mysterious person named 'Bill' whom I am told is the only living person in the world with the answer to my problem, I gave it up."

And the letter ends, "I'm pleased about one thing. It only cost me $33 to find out that having a Radio Shack Computer Center a few blocks from my home is worth the extra dollars I will spend in the future for 'Pure Radio Shack.'

My thanks to the customer who sent me a copy of this letter.

Letter number two is quite a bit different. This letter was sent from one of the several companies, who make Model III add in disk systems, to one of their unfortunate customers. It came to me via many hands but seems to have been first seen at a repair department where the customer was showing the letter. To see why he was at our repair center let me quote some of the letter.

"We have exhaustively tested your disk controller board and report the following:

1. The unit works properly in our test computer.
2. It has been tested with TRSDOS 1.2, NewDos 80 v 2.0, LDOS v 5.1.0A and DosPlus 3.3, 3.4, and 4.0D. We have determined that it works flawlessly with all of these operating systems, and, therefore, assume that the computer must be at fault."

I don't know why people still use TRSDOS 1.2 so long after 1.3 has been released. Now comes the good stuff.

"From the large number of complaints we and other disk upgrade manufacturers are having reported, it appears that Radio Shack is experiencing a high failure rate from their PC board manufacturing facility. We have also heard that ROM C is in its sixth revision, thus indicating that Tandy is still trying to make the basic computer work properly. Unconfirmed reports are that Tandy is shipping units incapable of working in a disk environment as 16K Level-II cassette only machines. This is being reported by a number of Radio Shack Franchise stores, a large number of whom are using our board for disk upgrades instead of the Tandy unit. Tandy assumes that if a unit is upgraded to disk, they can replace the defective PC board with one that will work with their disk. Incidentally, Tandy is having the same problems as the aftermarket manufacturers such as ———, except they can recycle the defective PC boards, and we cannot."

Now what is interesting about all this is that it is 100% untrue. Or to put it another way, we deny every claim the letter makes. What they did not seem to tell the customer was that perhaps the disk drive(s) they sent had a problem, or the cable, or the installation by the customer or perhaps there was not enough shielding for the drives. They also failed to mention that a double density disk controller has three or four adjustments that are critical and require a high quality scope to make. In other words a lot of things could cause his problem.

To be fair they do offer a refund but then they suggest the customer remove the disk controller and power supplies and take the unit in to us and demand that we fix it so that disk drives can be added. Since they know we are not making Model III's that are not disk compatible they suggest the following.

"They will no doubt give you a hard time, but you might hint that the system was advertised to be upgradable to disk operation, and that if you cannot get satisfaction, you will be obliged to contact the Fair Trade Commission. You might desire to write an article for one of the magazines, and possibly contact Ed Juge in Fort Worth."

Neat. We can't help you, so yell, loudly. Reminds me of a wall plaque I saw once that said: when in worry, when in doubt, run in circles, scream and shout. What I do not understand about the letter is the lack of mention of the disk drives. Some outside suppliers do sell just the double density board and power supply and leave the drives up to you. If that is the case here, it is possible that the customer has drives that cannot be used double density at the step rates required by the Model III. These would include the Shugart SA-400 and some Japanese drives that use cam mechanisms rather than band mechanisms.

Now I will admit that we make the disk Model III's in one factory and the non disk units in another due to the volume of Model III's we make. BUT we test every board that goes in a cassette Model III for all possible upgrades — disk, RS-232, and more memory. The logic on the CPU board that is required for the disk board is all dual purpose, so if it did not function the computer would not function. The only possible problem would be in the card edge on the board, and our actual failure rate is almost too low to measure. The few that do fail are fixed, not shipped.

Printing Features of the PC-2

Last month the many features of the PC-2 were introduced in this column. This month the focus of this column is on the powerful printing capabilities of this great little (in size only!) computer.

The Printer

The PC-2 printer, a four color plotter, is part of the printer/cassette interface (26-3605). The four ball point plotter pens are available in black, blue, green, and red. This printer uses 2¼" roll paper and prints at a maximum speed of eleven characters per second when using the smallest character size.

The Printer Switch and Manual Calculations

On the PC-2 interface there is a switch labeled PRINT. When the switch is in the AUTO position, calculations performed in the RUN, PRO, or RESERVE mode are displayed on the computer. If SIN 30 is ENTERed then 0.5 is displayed on the computer. If SIN 30 is ENTERed with the switch in the ECHO position then SIN 30 is printed on the printer, a line feed is executed, and 0.5 is printed right justified.

Again with the PRINT switch in the ECHO position (where A$ is equal to "TESTING"), when A$ ⟨**ENTER**⟩ is keyed in, A$ is printed left justified on the paper and TESTING is printed left justified directly under A$ like this:

```
A$
TESTING
```

During the above operation, A$ is also displayed on the computer's display until ⟨**ENTER**⟩ is pressed. Then TESTING appears left justified on the display. When the PC-2 is being used primarily as a computer, the PRINT switch will usually be in the AUTO position. When the PC-2 is used as a calculator where hard copy of both calculations and results is desired, the PRINT switch will be in the ECHO position.

Printer Modes

The two printer modes are GRAPH and TEXT. The TEXT mode is used to print numbers and characters, and the GRAPH mode is used for graphics.

The distinction between GRAPH and TEXT is very critical since many of the BASIC commands only work in one mode or the other.

TEXT ONLY	GRAPH ONLY	TEXT/GRAPH
LCURSOR	ROTATE	CSIZE
LLIST	SORGN	COLOR
LPRINT TAB	GLCURSOR	LPRINT
LF	LINE	USING
	RLINE	TEST

COLOR n

Where n is a value 0-3, COLOR is used to select which pen (0-3) will be used. If more than one pen color is installed in the printer then a pen color is selected by indicating the position of that pen. COLOR causes the specified pen to be selected, and the pen is then repositioned in the spot it previously occupied before the COLOR command was issued.

CSIZE n

When n equals a value (1-9), the CSIZE (or character size) command is used to alter the size of the printed characters. There are nine different character sizes ranging from 1.2 mm high x 0.8 mm wide to 10.8 mm high x 7.2 mm wide (0.048" x 0.032" to 0.432" x 0.288"). When no CSIZE is indicated all printing will be done in the default CSIZE which is 2. A run of the following program yields a printout of CSIZE 1 (the smallest), CSIZE 2, and CSIZE 3.

```
10:FOR X=1 TO 3
20:LF 1
30:CSIZE X
40:LPRINT "CSIZE TEST"
50:NEXT X
```

```
CSIZE TEST
CSIZE TEST
CSIZE TEST
```

Depending on the CSIZE specified you can print from 4 (CSIZE 9) to 36 (CSIZE 1) characters per line in upper or lower case. All printing is done in the last specified CSIZE.

GLCURSOR (X,Y)

GLCURSOR is used to move the pen to an X,Y position (−2047 to +2047) without drawing a line.

LCURSOR n

LCURSOR (TEXT mode only) enables you to move the pen to a specified character position (n). The position it moves to depends on the value of n and the size (CSIZE) of the characters being printed. For example, LCURSOR 5 would move the pen a greater distance if the CSIZE were 9 than it would if the CSIZE were 1.

LF n (Line Feed)

Where n is a negative or positive number, the LF command moves the printer paper in reverse (negative numbers) or forward (positive numbers). The distance that the paper moves per line feed depends on the CSIZE last specified. The maximum distance for reverse line feeds is 10 cm (approximately 4 inches).

LINE (X1,Y1) – (X2,Y2),line-type,color,B

The LINE command is the most powerful command of the GRAPH mode. It is used to move the pen from one coordinate (X1,Y1) to another coordinate (X2,Y2). If the pen is down then a line is drawn.

Line-type—The line-type drawn can be any one of the following nine lines (0-8) or no line at all can be indicated by 9.

Line-type Value	Resulting Line
0	————————
1	.....
2	----------
3	----------
4	----------
5	----------
6	----------
7	----------
8	----------
9	

CHART OF LINE TYPES

Color—The pen color is indicated by a number (0-3) in the LINE command.

B—An additional instruction, B for Box, can be added to create a box around the X1,Y1 – X2,Y2 diagonal.

The following program will draw two boxes. The first box is drawn with a green, broken line and the second box which is drawn below and to the left of the green box is red with a solid line.

```
10:GRAPH
20:GLCURSOR (100,100)
30:SORGN
40:LINE (10,10)-(50,50),3,2,B
50:LINE (-10,-10)-(-50,-50),0,3,B
```

If the first LINE coordinate is not specified then the unspecified coordinate value defaults to the current pen position.

Still another form of the LINE command allows up to six points in a row to be specified. Line 40 below uses this form of the line command to draw a triangle.

```
10:GRAPH
20:GLCURSOR (0,0)
30:SORGN
40:LINE(0,0)-(30,30)-(50,0)-(-50,0)
```

LLIST

Several options are available using LLIST. You can LLIST the entire program, a specific line (identified by either a line number or label), from a specified line number or label to the end of the program, from the beginning of the program to a specified line number or label, and within a range defined by two line numbers or two labels.

LPRINT

LPRINT is the primary command for displaying text on the printer. It has several different forms and works differently in the TEXT mode than it does in the GRAPH mode. The LPRINT Chart below lists the forms of LPRINT, the MODE(s) in which each form is operable, and the line of the demonstration program using that form.

LPRINT FORMAT	PRINT MODES	DEMONSTRATION LINE
LPRINT item	TEXT & GRAPH modes	30
LPRINT item1,item2	TEXT mode only	40
LPRINT item1;item2	TEXT & GRAPH modes	50
LPRINT item;	TEXT & GRAPH modes	60

LPRINT Demonstration Program

```
 5:N$="NAME":N=1
10:FOR X=1TO 2
20:ON XGOSUB 1000,1010
30:LPRINT"TEST"
40:IF X=1THEN LPRINT N$,N
50:LPRINT N$;N
60:LPRINT N$;
70:NEXT X
80:END
1000:TEXT:GOTO 1020
1010: GRAPH
1020:RETURN

RUN

TEST
NAME

NAME 1
NAME



TESTNAME 1NAME
```

LPRINT TABn; item/list

LPRINT TAB can be used in the TEXT mode only. It allows you to tab to a position (n) on the line and print an item/list. The position on the line is determined by the value of n and the size (CSIZE) of the characters being printed.

Example program:

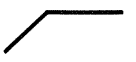
```
10:TEXT
20:CSIZE 2
30:FOR N=1 TO 5
40:LPRINT TAB N; "A";"B";"C"
50:NEXT N

RUN

ABC
 ABC
  ABC
   ABC
    ABC
```

LPRINT USING

LPRINT USING is much like a PRINT USING statement. It can be used in the TEXT and GRAPH modes.

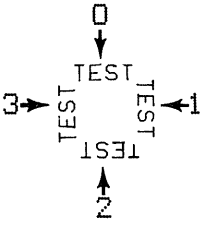
RLINE

RLINE is like the LINE command except that the origin is relative to the current pen position instead of the origin (0,0). Changing LINE to RLINE in line 40 in the LINE section results in the following figure.



ROTATE n

ROTATE n (where n is a value 0-3) indicates the four directions in which printing may occur. These directions are illustrated by the following chart.



SORGN

SORGN (Set Origin) specifies the current pen position as the origin for the graphing commands which follow.

TEST

This command is used to give the plotter a once over with most of its commands. TEST ascertains correct printer operation by checking for correct color sequence (0-1-2-3), size (1 cm) alignment, and clarity (each figure is drawn twice). TEST ⟨**ENTER**⟩ in any of the three modes (RUN, PRO, or RESERVE) causes the printer to print four boxes, each a different color. TEST can also be used in a program line.

GRAPHS FOR THE PC-2

Since there is a plotting program for the Multi-Pen Plotter (26-1191) and the Models I, II and III, it seemed reasonable that PC-2 plotter users might enjoy a similar program. (The program requires approximately 2806 steps of memory. 4K Ram Module required.)

This easy to run program plots four different graphs (or charts): Bar, Line, Band, and Circle. After entering the program, type ⟨**R**⟩⟨**U**⟩⟨**N**⟩ ⟨**ENTER**⟩ in the Run mode, and the following message appears on the display:

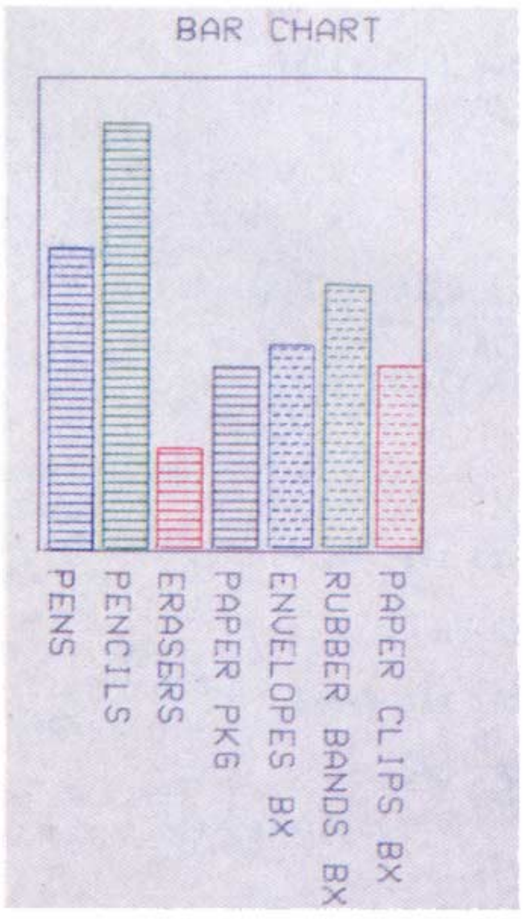
BAR/LINE = 1 BAND/CIRCLE = 2

Press ⟨**1**⟩ ⟨**ENTER**⟩ for a Bar or Line graph or press ⟨**2**⟩ ⟨**ENTER**⟩ for a Band or Circle graph.

BAR OR LINE GRAPH

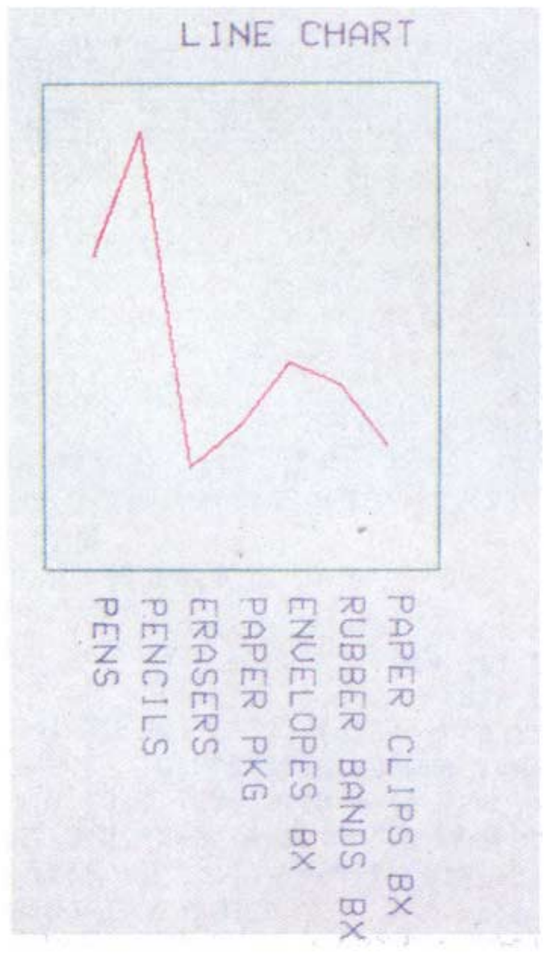
If ⟨**1**⟩ is pressed TITLE?_ appears on the display. This prompt is asking for a name for the graph. Key in a name and press ⟨**ENTER**⟩. Next BAR = 1,BROKEN LINE = 2?_ is displayed. Now press ⟨**1**⟩ ⟨**ENTER**⟩ for a Bar graph or ⟨**2**⟩ ⟨**ENTER**⟩ for a Broken Line graph.

After ⟨**1**⟩ or ⟨**2**⟩ is pressed, the prompt ITEM(1) = ? appears. Enter the name of the first item. Next, the prompt VALUE(1) = ? appears. Enter the value of item 1. This process will continue until the names and values of a maximum of eight items have been entered. To terminate entry of items before reaching the maximum (8), press the ⟨**ENTER**⟩ key in response to the ITEM prompt. The graph will now be printed. A bar graph might look something like this:



Bar Chart Example

while a line graph would look like this.

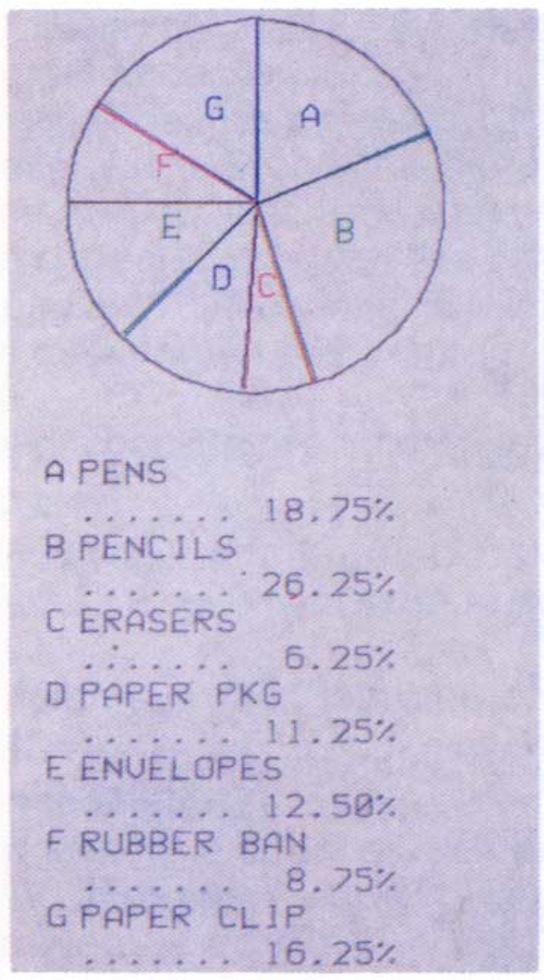


Example Line Chart

Upon completion of the chart the main menu reappears.

Band or Circle Graph

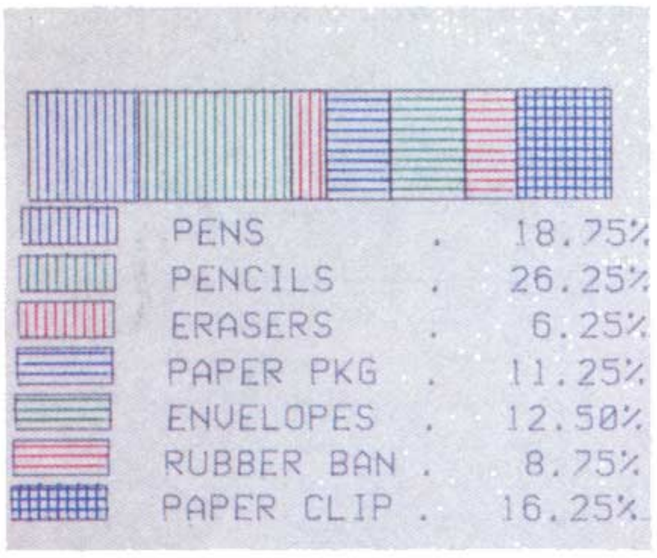
If you select ⟨**2**⟩ at the Main Menu the first prompt that appears is ITEM NAME(1)? Enter the name of the first item. Next appears the prompt VALUE(1) = ? Enter the value of the first item. The names and values of up to ten items may be entered. To terminate the entry of items press ⟨**ENTER**⟩ for the name. The prompt CIRCLE = 1 BAND = 2 is displayed. Choose 1 for a Circle or Pie chart or 2 for a Band graph. Depending on the number chosen the graph will then be reproduced.



Circle Chart Example

The Program

```
1: CLEAR
     : WAIT Ø
5: INPUT "BAR/LINE=1 BAND/CIRCLE=2";AN
7: IF AN < 1 OR AN > 2 GOTO 5
```



Band Chart Example

```
8: ON AN GOTO 1Ø, 85Ø
1Ø: DIM A$(8),A(8)
2Ø: INPUT "TITLE?"; A$(Ø)
3Ø: INPUT "BAR=1 BROKEN LINE=2?";C
4Ø: IF (C=1)+(C=2) <> 1 GOTO 3Ø
5Ø: FOR I=1 TO 8
6Ø: B$="ITEM("+ STR$ I+")="
    : PRINT B$;
65: INPUT A$(I)
    : CLS
    : GOTO 8Ø
7Ø: CLS
    : I=I-1
    : GOTO 1ØØ
8Ø: B$="VALUE("+ STR$ I+")="
    : PRINT B$;
85: INPUT A(I)
    : CLS
87: IF D < A(I) LET D=A(I)
9Ø: NEXT I
1ØØ: LPRINT "          "; A$(Ø)
1Ø5: D=45/D
11Ø: GRAPH
12Ø: GLCURSOR (Ø,-25Ø)
    : SORGN
13Ø: IF C=2 LET G=2
14Ø: LINE (Ø,Ø)-(2ØØ,25Ø),Ø,G,B
15Ø: IF C=2 GOTO 4ØØ
16Ø: G=5
17Ø: E=(4Ø-I)/I*5
18Ø: FOR J=1 TO I
19Ø: H=G+E
2ØØ: F=D*A(J)*5
22Ø: GOSUB 6ØØ
    : G=H+5
    : NEXT J
    : G=5
23Ø: FOR J=1 TO I
    : H=G+E
235: N=G+E/2-1Ø
    : GOSUB 8ØØ
24Ø: G=H+5
    : NEXT J
245: GLCURSOR (Ø,-25Ø)
25Ø: TEXT
    : LF 5
    : GOTO 1
4ØØ: E=4Ø/(I+1)*5
41Ø: FOR J=1 TO I
42Ø: H=E*J
43Ø: F=D*A(J)*5
44Ø: IF J=1 GOTO 46Ø
45Ø: LINE (G,M)-(H,F),Ø,3
46Ø: G=H
    : M=F
48Ø: NEXT J
483: FOR J=1 TO I
    : H=E*J
485: N=H
    : GOSUB 8ØØ
    : NEXT J
487: GLCURSOR (Ø,-25Ø)
49Ø: TEXT
    : LF 5
    : GOTO 1
6ØØ: M=M+1
    : L=L+1
61Ø: IF L=4 LET L=Ø
62Ø: GLCURSOR (G,Ø)
    : LINE (G,Ø)-(H,F),Ø,L,B
63Ø: P=Ø
    : IF M > 4 LET P=2
7ØØ: O=Ø
7Ø5: FOR K=1 TO 45
7Ø8: O=O+5
71Ø: IF F<= O GOTO 72Ø
713: IF K-INT (K/2) * 2=1 LINE (G,O)-(H,O),P
    : GOTO 718
715: LINE (H,O)-(G,O),P
718: NEXT K
72Ø: RETURN
8ØØ: ROTATE 1
81Ø: GLCURSOR (N,-15)
    : COLOR Ø
82Ø: LPRINT A$(J)
83Ø: ROTATE Ø
84Ø: RETURN
85Ø: Q=9
    : DIM A$(Q)*1Ø, B$(Q)*1, A(Q)
855: B$(Ø)="A"
    : B$(1)="B"
    : B$(2)="C"
    : B$(3)="D"
    : B$(4)="E"
    : B$(5)="F"
86Ø: B$(6)="G"
    : B$(7)="H"
    : B$(8)="I"
    : B$(9)="J"
87Ø: FOR I=Ø TO Q
88Ø: C$="ITEM NAME("+STR$(I+1)+")"
    : PRINT C$;
885: INPUT A$(I)
    : GOTO 892
89Ø: CLS
    : I=I-1
    : GOTO 9Ø5
892: CLS
    : C$="VALUE("+STR$ (I+1)+")="
    : PRINT C$;
895: INPUT A(I)
    : CLS
    : H=H+A(I)
9ØØ: NEXT I
9Ø5: I=I+1
91Ø: INPUT "CIRCLE=1 BAND=2 ";C
915: IF (C=1) + (C=2) <> 1 GOTO 91Ø
92Ø: IF C=2 GOTO 1Ø65
925: GRAPH
    : GLCURSOR (11Ø,-125)
    : SORGN
93Ø: D=12
    : Y=1ØØ
    : R=1ØØ
    : L=1
    : C=Ø
935: FOR J=1 TO 31
94Ø: GOSUB 117Ø
    : LINE (X,Y)-(X1,Y1)
    : X=X1
    : Y=Y1
    : C=C+D
945: NEXT J
95Ø: FOR J=Ø TO I-1
944: R=1ØØ
96Ø: F=36Ø*A(J)/H
    : F=F+G
    : IF J=I-1 LET F=36Ø
965: FOR M=1 TO 2
97Ø: IF M=1 LET C=G+.5
    : GOTO 98Ø
975: C=F-.5
98Ø: GOSUB 117Ø
    : IF L>3 LET L=1
```

```
985: LINE (Ø,Ø)-(X1,Y1),Ø,L
     : NEXT M
99Ø: R=5Ø
     : C=(F-G)/2+G
     : GOSUB 117Ø
     : X1=X1-3
995: G=F
1ØØØ: GLCURSOR (X1,Y1)
     : LPRINT B$(J)
     : L=L+1
     : NEXT J
1ØØ5: GLCURSOR (-11Ø,-15Ø)
     : SORGN
1Ø1Ø: Y=Ø
     : X=Ø
     : COLOR Ø
1Ø15: FOR J=Ø TO I-1
1Ø2Ø D=A(J)/H*1ØØ
     : D=INT((D+.ØØ5)*1ØØ)/1ØØ
     : IF J=I-1 LET D=1ØØ-N
     : GOTO 1Ø3Ø
1Ø25: N=N+D
1Ø3Ø: GLCURSOR (X,Y)
     : LPRINT B$(J)
1Ø35: GLCURSOR (18,Y)
     : LPRINT A$(J)
1Ø4Ø:Y=Y-2Ø
1Ø45:GLCURSOR (18,Y)
     : LPRINT "......."; USING
     "###.##";D;"%";USING
1Ø5Ø:Y=Y-2Ø
1Ø55:NEXT J
1Ø6Ø:TEXT
     : LF 5
     : GOTO 1
1Ø65:GRAPH
     : GLCURSOR (Ø,Ø)
     : SORGN
     : ROTATE 1
1Ø7Ø:K=1
     : L=1
     : S=16Ø
     : V=215
1Ø75:FOR J=Ø TO I-1
1Ø8Ø:D=INT(A(J)/H*1ØØ+.5)
     : E=D*3
1Ø85:W=T-E
     : IF J=I-1 LET W=-3ØØ
1Ø9Ø:IF L>3 LET L=1
     : K=K+1
1Ø95:LINE (16Ø,T)-(215,W),Ø,Ø,B
     : GOSUB 1175
11ØØ:T=W
     : L=L+1
     : NEXT J
11Ø5:K=1
     : L=1
     : W=-5Ø
     : T=Ø
111Ø:FOR J=Ø TO I-1
1115:IF L>3 LET L=I
     : K=K+1
112Ø:F=16Ø/I*(I-J-1)
     : LINE (F,Ø)-((F-5+16Ø/I),-5Ø),Ø,Ø,B
1125:S=F
     : V=F-5+16Ø/I
     : GOSUB 1175
113Ø:COLOR Ø
     : GLCURSOR (F,-8Ø)
     : LPRINT A$(J)
1135:GLCURSOR (F,-21Ø)
     : LPRINT "."
114Ø:D=A(J)/H*1ØØ
     : D=INT((D+.ØØ5)*1ØØ)/1ØØ
1145:IF J=I-1 LET D=1ØØ-G
     : GOTO 1155
115Ø:G=G+D
1155:GLCURSOR (F,-24Ø)
     : LPRINT USING "###.##";D;"%"
     : USING
116Ø:L=L+1
     : NEXT J
1165:TEXT
     : LF 5
     : GOTO 1
117Ø:X1=R*SIN C
     : Y1=R*COS C
     : RETURN
1175:IF K > 3 LET K=1
118Ø:IF K=1 GOSUB 12Ø5
1185:IF K=2 GOSUB 124Ø
119Ø:IF K=3 GOSUB 12Ø5
     : GOSUB 124Ø
12ØØ:RETURN
12Ø5:P=T
     : FOR O=1 TO 6Ø
121Ø:P=P-5
1215:IF P <= W GOTO 1235
122Ø:IF O-INT(O/2)*2=Ø LINE (S,P)-(V,P),Ø,L
     : GOTO 123Ø
1225:LINE (V,P)-(S,P),Ø,L
123Ø:NEXT O
1235:RETURN
124Ø:P=S
     : FOR O=1 TO 5Ø
1245:P=P+5
125Ø:IF P >= V GOTO 127Ø
1255:IF O-INT(O/2)*2=Ø LINE (P,T)-(P,W),Ø,L
     : GOTO 1265
126Ø:LINE (P,W)-(P,T),Ø,L
1265:NEXT O
127Ø:RETURN
```

Pocket Computer Bugs, Errors, and Fixes

Changes to BASIC programs

There are general procedures that need to be followed when any corrections are made.

1. Make a backup of the tape that contains the program to be corrected. Changes should be made on the backup copy.
2. Load the program to be changed by typing CLOAD"filename" where filename is the name of the program to be modified.
3. Make the line changes indicated in the fix. For existing line numbers, edit or retype the line to match the one in the fix. New lines should be entered.
4. Save the corrected program (the one now in memory). Type CSAVE"filename" ⟨**ENTER**⟩ where filename is the name of the program that has been modified.
5. Now make a backup of the corrected tape.

Surveying (26-3512)

In Surveying (Version 1.0) a problem exists where a PRINT format (USING) statement is left active after leaving another routine.

Make the following change in the program named CLOSE: `412 NEXT M:USING`

Personal Finance (26-3518)

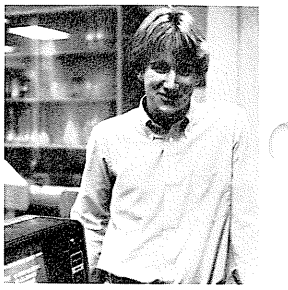
If the user gets no display of month with program "BG," option "B" after specifying "N" to prompt "DATA TAPE." The problem is documented in the instruction addendum 875-9107.

OPTIONAL: The program can be made to display current month in all cases by editing line 100.

Make the following change in the program named BG:

```
1ØØ INPUT"LOAD DATA TYPE(Y/N)?";A$:IFA$=
    "N"THEN12Ø
```

Football For Summer



The following program "FOOTBALL" was the winning entry in the recent Charleston, South Carolina JETS 1981 — Design Competition — TRS-80 Computer Game Contest.

The program is a football game between you and the computer. You have the options of naming your own team and players, or you can accept the names the computer provides. I am not a football expert, and the game promptly proceeded to clean the field with me. I hope you have better luck, someone has to champion the superiority of humans over computers.

The program was written by Ken Clark, a 17 year old junior at Walterboro High School, where he is a member of the local JETS club. Ken has had his Model I for about seven months, though his interest in computing goes back at least two years to the summer when he enrolled in Clemson's "Interaction Computing" course. Ken's hobbies include chess, golf, and band, and he plans to study Computer Engineering at Georgia Tech.

Our congratulations to Ken, his JETS Club sponsor Mrs. Georgia Spruill, as well as to the other people who participated in the competition.

Ken's program runs in a Model I with 16K and Level II BASIC (it will also run on a Model III BASIC). The program contains instructions, and seems to play a reasonable game of football.

We do have some words of caution. As is the Microcomputer News style, we have inserted many spaces in the program lines to make program entry and verification as simple as possible. The result is that some lines will now be longer than the computer will accept. As you enter the lines, you may need to remove all spaces which do not occur within quotes. Further, while we did not check the exact memory requirements of the program, it takes most of the available memory in a 16K machine, so you may need to remove all non-essential spaces. One other comment. Inside strings, you will frequently notice that Ken runs phrases or words together. If your text is positioned properly, this causes wraparound on the screen, and the text prints correctly.

```
1Ø 'KEN CLARK/ 3Ø6 COUNTRY LN/ WALTERBORO, SC
     29488/ (8Ø3)549-7164    ALL RIGHTS RESERVED
     BY THE AUTHOR - DECEMBER 21, 1981
3Ø CLEAR 1ØØØ
     : RANDOM
     : CLS
     : PRINT@537, "FOOTBALL  1.3"
     : PRINT@96Ø, "DO YOU WANT INSTRUCTIONS (Y/N)
     ?";
4Ø A$=INKEY$
     : IF A$<>"Y" AND A$<>"N" THEN 4Ø ELSE CLS
     : IF A$="N" THEN 1ØØ
45 PRINT "THE OFF. PLAYS ARE: LB-LONG BOMB
     HM-HAIL MARYOT-OVER THE TOP RUN
     ES-END SWEEPDD-DRAW
     SP-SCREEN PASSDO-DOWN AND OUT PASSFG-FIELD
     GOAL              PP-PUNT"
5Ø PRINT
     : PRINT "    THIS GAME PLAYS JUST LIKE
     REGULAR FOOTBALL.  THERE ARE 12 MINUTES IN A
     QUARTER AND FOUR QUARTERS IN A GAME.  IF THE
     SCORE IS TIED AFTER REGULATION YOU GO INTO
     OVERTIME.  IN OVERTIME THE FIRST PERSON TO
     SCORE WINS. IF IT GOES 12 MINUTE";
51 PRINT "S WITHOUT      ANYONE SCORING THEN IT
     IS RULED A TIE.    THE ABBREVIATIONS OF THE
     PLAYS WILL BE DISPLAYED WHILE THE GAME IS
     PLAYING SO YOU DON'T HAVE TO MEMORIZE THEM.
     MORE ON THE PLAYS LATER."
     : PRINT@96Ø, "HIT <ENTER> TO CONTINUE";
     : A$=INKEY$
52 IF INKEY$="" THEN 52 ELSE CLS
     : PRINT "   WHEN ENTERING PLAYS, IF YOU WANT
     TO RUN THE SAME PLAY AS THE TIME BEFORE YOU
     CAN JUST HIT <ENTER>."
53 PRINT
     : PRINT "  LONG BOMB : THIS PLAY IS JUST
     WHAT YOU THINK.  ABOUT 25 YARDS PER
     COMPLETION BUT WITH A LOW PERCENTAGE OF
     COMPLETIONS.  DOWN AND OUT : THIS PLAY HAS A
     HIGHER PERCENTAGE OF COMPLETIONS BUT ONLY
     GAINS ABOUT 7 YARDS."
54 PRINT "  PUNT : PUNTS THE BALL FOR AN AVERAGE
     OF ABOUT 4Ø YARDS.  FIELD GOAL : ATTEMPTS A
     FIELD GOAL.  OUTSIDE OF THE 35 YARD LINE
     YOUR KICKER IS VERY INACCURATE BUT IT IS
     STILL POSSIBLE TO MAKE THEM.  OUTSIDE OF THE
     48 YARD LINE IT IS ";
55 PRINT "VIRTUALLY IMPOSSIBLE."
6Ø PRINT@96Ø, "HIT <ENTER>";
62 IF INKEY$="" THEN 62 ELSE CLS
     : PRINT "THE DEF. PLAYS ARE:GL - GOAL LINE
     (IN SHORT YARDAGE SITUATIONS)RR - RUN
     DEFENSE (DRAW, SWEEP, OPTION, ETC.)PA - PASS
     DEFENSE (SHORT AND MEDIUM PASSES)PV -
     PREVENT DEFENSE (AGAINST LONG BOMBS)PP -
     PUNT (AGAINST ";
65 PRINT "PUNTS)FG - TO BLOCK FIELD GOALS"
98 PRINT@96Ø, "HIT ENTER TO CONTINUE";
99 IF INKEY$="" THEN 99
1ØØ CLS
     : PRINT "DO YOU WANT TO ENTER YOUR OWN NAMES
     FOR THE PLAYERS (Y/N) ? ";
1Ø5 A$=INKEY$
     : IF A$<>"Y" AND A$<>"N" THEN 1Ø5 ELSE PRINT
     A$
     : IF A$="N" THEN 31Ø
11Ø INPUT "YOUR TEAM'S NAME"; H$(Ø)
     : IF LEN(H$(Ø))>6 THEN PRINT "TOO LONG (6
     CHAR)"
     : GOTO 11Ø ELSE INPUT "YOUR QUARTERBACK";
     H$(1)
     : INPUT "YOUR # 1 RUNNING BACK"; H$(2)
     : INPUT "YOUR # 2 RUNNING BACK"; H$(3)
     : INPUT "YOUR # 1 WIDE RECEIVER"; H$(4)
115 INPUT "YOUR # 2 WIDE RECEIVER"; H$(5)
12Ø INPUT "YOUR TIGHT END"; H$(6)
     : INPUT "YOUR KICKER"; H$(7)
     : INPUT "YOUR PUNTER"; H$(8)
     : INPUT "YOUR KICKOFF/PUNT RETURNER"; H$(9)
     : GOTO 36Ø
31Ø H$(Ø)="TAMPA"
     : H$(1)="DOUG WILLIAMS"
     : H$(2)="JERRY ECKWOOD"
     : H$(3)="JAMES WILDER"
     : H$(4)="KEVIN HOUSE"
     : H$(5)="JIMMIE GILES"
     : H$(6)="GORDON JONES"
     : H$(7)="BILL CAPECE"
     : H$(8)="SWIDER"
     : H$(9)="TONY DAVIS"
36Ø O$(Ø)="CHICAGO"
     : O$(1)="VINCE EVANS"
     : O$(2)="WALTER PAYTON"
     : O$(3)="MATT SUHEY"
     : O$(4)="MARGERUM"
     : O$(5)="ROBIN EARL"
     : O$(6)="BRAIN BASCHNAGEL"
     : O$(7)="JOHN ROVETO"
     : O$(8)="BOB PARSONS"
     : O$(9)="D. WILLIAMS"
41Ø PRINT TAB(1Ø) "COIN TOSS.  HEADS OR TAILS ?
     ";
42Ø A$=INKEY$
     : IF A$<>"H" AND A$<>"T" THEN 42Ø ELSE IF
     A$="H" PRINT "HEADS" ELSE PRINT "TAILS"
43Ø IF RND(2)=1 PRINT "YOU WIN THE TOSS.  WILL
     YOU RECEIVE OR KICKOFF ? ";  ELSE 46Ø
44Ø A$=INKEY$
     : IF A$<>"R" AND A$<>"K" THEN 44Ø ELSE IF
     A$="K" THEN PRINT "KICKOFF"
     : PO=-1
     : KK=-1 ELSE PO=1
     : T=1
     : KK=1
     : PRINT "RECEIVE"
45Ø GOTO 47Ø
46Ø PRINT "YOU LOST THE TOSS.  "; O$(Ø); "
     CHOOSES TO RECEIVE."
     : PO=-1
     : KK=-1
47Ø GOSUB 54ØØ
     : GOSUB 515Ø
48Ø T1=12ØØ
     : TT=T1
     : GOSUB 5ØØØ
9ØØ IF PO=-1 THEN 95Ø
91Ø IF T=Ø THEN Y2=Y+1Ø
     : T=1
     : D=Ø
     : IF Y2>5Ø THEN Y2=5Ø
915 IF Y=>5Ø THEN GOSUB 55ØØ
     : GOTO 95Ø
92Ø IF Y<-49 THEN 57ØØ
93Ø D=D+1
     : IF Y>=Y2 THEN D=1
     : Y2=Y+1Ø
94Ø IF D=5 THEN PO=-1
     : GOTO 95Ø
945 PRINT@263, D;
     : PRINT@285, Y2-Y;
     : GOSUB 51ØØ
     : GOSUB 515Ø
     : GOTO 1ØØØ
95Ø IF PO=1 GOTO 9ØØ ELSE IF T=1 THEN Y2=Y-1Ø
     : T=Ø
     : D=Ø
     : IF Y2<-5Ø THEN Y2=-5Ø
955 IF Y>49 GOTO 57ØØ
965 IF Y=<-5Ø GOSUB 555Ø
     : PO=1
     : GOTO 9ØØ
97Ø D=D+1
     : IF Y<=Y2 THEN D=1
     : Y2=Y-1Ø
971 IF D=5 THEN D=1
     : PO=1
     : GOTO 9ØØ
972 PRINT@263, D;
     : PRINT@285, Y-Y2;
     : GOSUB 51ØØ
     : GOSUB 515Ø
```

```
     : GOTO 3000
1000 GOSUB 5600
     : PRINT@384, "";
     : INPUT "OFF. PLAY"; P$
     : P$=LEFT$(P$, 2)
     : PRINT@395, "";
1010 IF P$<>"LB" THEN 1100 ELSE PRINT "LONG BOMB"
     : IF RND(0)<.9 THEN 1030
1025 PRINT H$(1); " SACKED FOR A";
     : Y9=RND(15)
     : PO=-1
     : GOSUB 5200
     : PO=1
     : PRINT Y9; "YARD LOSS !!"
     : Y=Y-Y9
     : TT=TT-RND(10)-RND(10)
     : GOTO 900
1030 PRINT H$(1); " DROPS BACK."
     : FOR ZZ=0 TO 99
     : NEXT
     : IF RND(0)>.9 THEN PRINT "HE SCRAMBLES
     FOR";
     : Y9=RND(20)-5
     : GOSUB 5200
     : PRINT Y9
     : Y=Y+Y9
     : TT=TT-RND(20)-RND(20)
     : GOTO 900
1035 PRINT "THROWS TO "; H$(RND(3)+3)
     : FOR ZZ=0 TO 500
     : NEXT
     : IF RND(0)<.3 THEN PRINT "CAUGHT FOR";
     : Y9=RND(20)+15
     : GOSUB 5200
     : PRINT Y9
     : Y=Y+Y9
     : TT=TT-RND(20)-RND(20)
     : GOTO 900
1040 IF RND(0)<.09 THEN PO=-1
     : PRINT "INTERCEPTED.  RUN BACK";
     : Y9=RND(60)
     : GOSUB 5200
     : PRINT Y9; "YARDS."
     : Y=Y-Y9
     : TT=TT-RND(20)-RND(10)
     : GOTO 900
1050 PRINT "INCOMPLETE."
     : TT=TT-RND(7)-RND(7)
     : GOTO 900
1100 IF P$<>"FG" THEN 1200 ELSE PRINT "FIELD
     GOAL"; H$(7); " KICK FROM" 50-Y+18 "YARDS."
     : TT=TT-RND(7)-RND(8)
1106 IF RND(0)<.03 THEN PO=-1
     : PRINT "BLOCKED RUN BACK ";
     : Y9=RND(20)
     : IF Y9=20 THEN PRINT "FOR A TOUCHDOWN."
     : Y=-55
     : GOTO 965 ELSE GOSUB 5200
     : PRINT Y9 " YARDS."
     : PO=-1
     : Y=Y-Y9
     : GOTO 950
1110 IF RND(50)>(-Y+40) THEN TS=TS+3
     : PRINT "FIELD GOAL GOOD !!!"
     : PO=-1
     : IF QQ=4 THEN 6050 ELSE GOSUB 5400
     : GOTO 900 ELSE PRINT "FIELD GOAL NO GOOD."
     : PO=-1
     : D=0
     : GOTO 900
1200 IF P$<>"HM" THEN 1300
1202 PRINT "HAIL MARY"
     : TT=TT-RND(7)-RND(7)
1210 PRINT H$(1); " THROWS TO "; H$(4); " IN THE
     END ZONE." ABS(Y-50) "YARDS"
     : FOR ZZ=1 TO 1500
     : NEXT
1215 IF RND(0)>.93 THEN Y=55
     : PRINT " * * * C O M P L E T E * * *"
     : GOSUB 5500
     : PO=-1
     : D=0
     : GOSUB 5400
     : GOTO 950
1220 PRINT "INCOMPLETE"
     : GOTO 900
1300 IF P$<>"DO" THEN 1400
1302 PRINT "DOWN AND OUT"
     : TT=TT-RND(8)-RND(8)
1310 PRINT H$(1); " PASSES TO "; H$(RND(3)+3)
1315 IF RND(0)<.03 THEN PO=-1
     : PRINT "INTERCEPTED. RUN BACK ";
     : Y9=RND(30) ELSE 1325
1320 IF Y9=>27 PRINT "FOR A TOUCHDOWN."
     : Y=-55
     : GOTO 965 ELSE GOSUB 5200
     : PRINT Y9" YARDS."
     : D=0
     : Y=Y-Y9+RND(20)
     : GOTO 950
1325 IF RND(0)<.50 PRINT "INCOMPLETE."
     : GOTO 900
1330 PRINT "COMPLETE FOR ";
     : Y9=RND(10)+2
     : GOSUB 5200
     : PRINT Y9 " YARDS."
     : Y=Y+Y9
     : GOTO 900
1400 IF P$<>"OT" THEN 1500
1402 PRINT "OVER THE TOP RUN"
     : TT=TT-RND(20)-RND(20)-RND(20)
1405 PRINT H$(RND(2)+1); " RUNS UP THE MIDDLE."
     : FOR ZZ=0 TO 50
     : NEXT
     : PRINT "HE PICKS UP ";
     : Y9=RND(8)-3
     : GOSUB 5200
     : PRINT Y9; "YDS."
     : Y=Y+Y9
1410 IF RND(0)<.01 THEN PRINT "FUMBLE !"
     : FOR ZZ=1 TO 500
     : NEXT
     : IF RND(2)=1 THEN PRINT "RECOVERED.  NO
     GAIN."
     : GOTO 900 ELSE PO=-1
     : PRINT "PICKED UP BY "; O$(0); " !RUN BACK
     ";
     : Y9=RND(10)
     : IF Y9=10 THEN Y=-55
     : PRINT "FOR A TD."
     : GOTO 965 ELSE GOSUB 5200
     : PRINT Y9 "YARDS."
     : GOTO 950
1499 GOTO 900
1500 IF P$<>"ES" THEN 1600
1502 PRINT "END SWEEP"
     : TT=TT-RND(15)-RND(17)-RND(10)
1507 PRINT H$(1); " HANDS OFF TO "; H$(RND(2)+1);
     "."
1510 IF RND(29)=1 PRINT "HE FUMBLES ";
     : IF RND(2)=1 PRINT "RECOVERED."
     : GOTO 900 ELSE PRINT "LOST."
     : PO=-1
     : GOTO 950
1520 Y9=RND(12)-4
     : GOSUB 5200
     : Y=Y+Y9
     : PRINT "HE GETS " Y9 " YARDS ON THE PLAY."
     : GOTO 900
1600 IF P$<>"DD" THEN 1700
1602 PRINT "DRAW"
     : TT=TT-RND(20)-RND(20)-RND(10)
1605 PRINT "DELAYED RUN.  "; H$(RND(2)+1); "
     CARRIES."
1610 IF RND(0)<.02 THEN PRINT "FUMBLE.  ";
     : IF RND(0)<.5 THEN PRINT H$(0); "
     RECOVERS."
     : GOTO 900 ELSE PRINT O$(0); " RECOVERS. RUN
     BACK ";
     : Y9=RND(25)
     : PO=-1
     : GOSUB 5200
     : PRINT Y9 "YARDS."
     : Y=Y-Y9
     : GOTO 950
1620 PRINT "HE GETS ";
     : Y9=RND(15)-5
     : GOSUB 5200
     : PRINT Y9 "YARDS."
     : Y=Y+Y9
     : GOTO 900
1700 IF P$<>"SP" THEN 1800
1702 PRINT "SCREEN PASS"
     : TT=TT-RND(15)+5
1710 PRINT H$(1); " THROWS TO "; H$(RND(2)+1);
     "."
     : FOR ZZ=1 TO 300
     : NEXT
1715 IF RND(0)<.08 THEN PRINT "INTERCEPTED !"
     : PO=-1
     : FOR ZZ=1 TO 400
     : NEXT
     : Y9=RND(2)*RND(2)*RND(2)+RND(20)
     : IF Y-Y9<-49 OR Y9>24 THEN PRINT "RUN BACK
     FOR A TOUCHDOWN !"
     : Y=-55
     : GOTO 965 ELSE GOSUB 5200
     : PRINT "RUN BACK"; Y9; "YARDS."
     : Y=Y-Y9
     : GOTO 950
1720 IF RND(0)<.3 THEN PRINT "INCOMPLETE."
     : GOTO 900
1730 PRINT "COMPLETE.  HE GETS";
     : Y9=RND(30)-10
     : IF Y9=20 THEN PRINT " A TOUCH DOWN !"
     : Y=55
     : GOSUB 5500
     : GOTO 950 ELSE GOSUB 5200
     : PRINT Y9; "YARDS."
     : Y=Y9+Y
     : GOTO 900
1800 IF P$<>"OM" THEN 1900
1802 PRINT "OVER THE MIDDLE PASS"
1900 IF P$<>"PP" THEN 2000 ELSE
     TT=TT-RND(12)-RND(12)
     : PRINT "PUNT"; H$(8); " PUNTS.";
1915 IF RND(0)<.04 THEN PRINT "BLOCKED.  RUN BACK
     ";
     : Y9=RND(30)
     : IF Y9>28 PRINT "FOR A TOUCHDOWN !!!"
     : Y=-55
     : GOTO 965 ELSE PO=-1
     : GOSUB 5200
     : PRINT Y9 " YARDS."
     : Y=Y-Y9
     : GOTO 950
1930 Y9=RND(30)+20
     : GOSUB 5200
     : PRINT "PUNT COVERS " Y9 " YARDS."
     : IF Y+Y9>49 THEN Y=30 ELSE Y=Y+Y9
1932 PO=-1
     : IF Y=30 THEN PRINT "TOUCHBACK"
     : GOTO 950
1935 PO=-1
     : IF RND(0)<.3 THEN PRINT "FAIR CATCH BY ";
     O$(9)
     : GOTO 950
1940 PRINT "CAUGHT BY "; O$(9)
     : IF RND(0)<.05 THEN PRINT "HE FUMBLES IT."
     : IF RND(0)<.5 THEN PRINT "RECOVERED BY ";
     O$(0)
     : GOTO 950 ELSE PRINT "RECOVERED BY "; H$(0)
     : PO=1
     : T=1
     : GOTO 900
1950 Y9=RND(20)-5
     : IF Y9=15 THEN Y9=Y9+RND(15)
     : IF Y9>28 THEN Y9=Y9+RND(50)
1960 GOSUB 5200
     : PRINT "HE RUNS IT BACK"; Y9; "YARDS."
     : Y=Y-Y9
     : GOTO 950
2000 PRINT
     : PRINT "DELAY OF GAME PENALTY OF "; H$(0);
     ".  FIVE YARDS."
     : Y9=5
     : IF Y-Y9<-49 THEN Y=INT((Y+Y-Y9)/2)+1 ELSE
     Y=Y-Y9
     : GOSUB 5100
     : PRINT@285, Y2-Y;
     : GOTO 1000
3000 GOSUB 5600
     : PRINT@384, "";
     : INPUT "DEFENSIVE PLAY "; P$
     : P$=LEFT$(P$, 2)
3002 PRINT@401, "";
     : IF P$="PA" THEN PRINT "PASS" ELSE IF
     P$="RR" THEN PRINT "RUN" ELSE IF P$="FG"
     THEN PRINT "FIELD GOAL" ELSE IF P$="PP" THEN
     PRINT "PUNT" ELSE IF P$="PV" THEN PRINT
     "PREVENT" ELSE IF P$="GL" THEN PRINT "GOAL
     LINE" ELSE PRINT "NONE"
3010 IF (D=4 AND Y<-20) OR (CS<TS AND CS>TS-4 AND
     TT<25 AND QQ<>0 AND QQ<>2 AND Y<-10) OR
     (TT<25 AND QQ=1) OR (QQ=3 AND TT<25 AND
     CS+4>TS) AND (Y<0) THEN IF Y>0 THEN ELSE
     PRINT "TRIES FIELD GOAL."; O$(7); " KICKS
     FROM " Y+68 " YARDS IS ";
     : TT=TT-RND(10)-RND(10) ELSE GOTO 3040
3020 IF P$="FG" THEN IF RND(0)<.08 THEN
     : PO=1
     : PRINT "BLOCKED !!!  RUN BACK ";
     : Y9=RND(20)
     : IF Y9=20 PRINT "FOR A TD !!!"
     : Y=55
     : GOTO 915 ELSE GOSUB 5200
     : PRINT Y9 " YARDS."
     : Y=Y+Y9
     : GOTO 900
3025 IF -RND(50)>Y THEN PRINT "GOOD."
     : CS=CS+3
     : PO=1
     : IF QQ=4 THEN 6050 ELSE GOSUB 5150
     : GOSUB 5400
     :  GOTO 900
3030 PRINT "NO GOOD."
     : PO=1
     : D=0
     : GOTO 950
3040 IF (D=4 AND ABS(Y-Y2)>2 AND CS>TS AND TT>400
     AND QQ<>0 AND QQ<>2) OR (D=4 AND CS>TS) OR
     (D=4 AND Y- Y2>2) THEN PRINT O$(8); "
     PUNTS";
     : TT=TT-RND(15)-RND(15) ELSE 3065
3045 IF P$="PP" AND RND(0)<.13 THEN PO=1
     : D=1
     : PRINT ".IT'S BLOCKED.  RUN BACK ";
     : Y9=RND(40)
     : IF Y9>38 PRINT "FOR A TOUCHDOWN!"
     : Y=55
     : GOTO 915 ELSE GOSUB 5200
     : PRINT Y9
     : Y=Y+Y9
     : GOTO 900
3050 PRINT " FOR ";
     : Y9=RND(22)+20
```

```
     : GOSUB 52ØØ
     : PRINT Y9 " YARDS."
     : PO=1
     : D=Ø
3Ø55 IF Y-Y9<-49 THEN PRINT "TOUCHBACK."
     : Y=-3Ø
     : GOTO 3Ø6Ø
3Ø57 Y=Y-Y9
     : PRINT "CAUGHT BY "; H$(9)
     : IF RND(Ø)<.Ø7 THEN PRINT "FUMBLED!  ";
     : IF RND(Ø)<.5 THEN PRINT "RECOVERED BY ";
     O$(Ø)
     : T=1
     : PO=-1
     : GOTO 95Ø ELSE PRINT "RECOVERED BY "; H$(Ø)
     : GOTO 9ØØ ELSE Y9=RND(2Ø)-5
     : PRINT "RUN BACK "; Y9; "YARDS"
     : Y=Y+Y9
3Ø6Ø Y2=Y-1Ø
     : GOTO 9ØØ
3Ø65 Q=RND(1ØØ)
     : IF Q<35 PRINT O$(1); " PASSES TO "
     O$(RND(3)+3) "." ELSE 312Ø
3Ø7Ø Q=RND(115)
     : IF P$="PA" OR P$="PV" THEN Q=Q-15
3Ø8Ø IF Q<55 PRINT "INCOMPLETE."
     : TT=TT-RND(7)-RND(7)
     : GOTO 95Ø
3Ø9Ø IF Q<6Ø PRINT "INTERCEPTED.  RUN BACK ";
     : TT=TT-RND(1Ø)-RND(1Ø) ELSE GOTO 311Ø
3Ø95 Y9=RND(3Ø)
     : IF Y9=3Ø THEN PRINT "TOUCHDOWN !!!"
     : D=Ø
     : Y=55
     : GOTO 915
31ØØ PO=1
     : GOSUB 52ØØ
     : PRINT Y9 " YARDS."
     : Y=Y+Y9-1Ø
     : D=Ø
     : GOTO 9ØØ
311Ø PRINT "COMPLETE FOR ";
     : Y9=RND(3Ø)
     : IF Y9>28 PRINT "A TOUCHDOWN!"
     : Y=-55
     : TT=TT-RND(15)-RND(15)
     : GOTO 965 ELSE GOSUB 52ØØ
     : PRINT Y9 " YARDS."
     : Y=Y-Y9
     : TT=TT-RND(15)-RND(15)-RND(15)
     : GOTO 95Ø
312Ø IF QQ<>Ø AND QQ<>2 AND CS<TS AND TT<4ØØ THEN
     3Ø65 ELSE PRINT O$(RND(2)+1) " RUNS."
313Ø IF RND(Ø)<.Ø55 THEN PRINT "HE FUMBLES.
     RECOVERED BY ";
     : TT=TT-RND(15)-RND(15) ELSE 315Ø
3135 IF RND(Ø)<.5 THEN PRINT H$(Ø); ". RUN BACK
     ";
     : Y9=RND(2Ø)
     : IF Y9=2Ø PRINT "FOR A TOUCHDOWN."
     : Y=55
     : GOTO 915 ELSE PO=1
     : GOSUB 52ØØ
     : PRINT Y9 " YARDS."
     : D=Ø
     : GOTO 9ØØ
314Ø PRINT O$(Ø); ".  NO GAIN ON THE PLAY."
     : GOTO 95Ø
315Ø Y9=RND(16)-2
     : IF P$="RR" OR P$="GL" THEN Y9=Y9-3
3155 IF Y9>1Ø THEN Y9=Y9+RND(1Ø)
     : IF Y9>18 THEN Y9=Y9+RND(2Ø)
     : IF Y9>33 THEN Y9=Y9+RND(3Ø)
     : IF Y9> 53 THEN Y9=Y9+RND(4Ø)
316Ø GOSUB 52ØØ
     : PRINT "HE GETS " Y9 " YARDS ON THE PLAY."
     : Y=Y-Y9
     : TT=TT-RND(12)-RND(12)-RND(12)-RND(12)
     : GOTO 95Ø
5ØØØ CLS
     : PRINT@Ø, STRING$(64, 14Ø);
     : PRINT@32Ø, STRING$(64, 14Ø);
     : FOR Z=2 TO 15
     : SET(Ø, Z)
     : SET( 127, Z)
     : NEXT
5Ø2Ø PRINT@11Ø, "DEF.P / RR.PA.GL";
     : PRINT@174, "PV.PP.FG";
     : PRINT@238, "OFF.P / FG.PP.ES";
     : PRINT@3Ø2, "DD.HM.SP.OT.DO.LB";
5Ø3Ø PRINT@257, "DOWN -"; 1, "YARDS TO GO -"; 1Ø;
5Ø4Ø PRINT@211, "QRT -";
     : PRINT@22Ø, "TIME -";
51ØØ PRINT@65, "SCORE
     :  "; H$(Ø); " -" TS "* "; O$(Ø); " -"CS;
51Ø5 PRINT@129, "BALL ON THE ";
     : IF Y<Ø PRINT Y+5Ø "    IN "; H$(Ø); "
     TERRITORY. ";  ELSE IF Y>Ø PRINT 5Ø-Y "
     IN "; O$(Ø); " TERRITORY.    ";  ELSE PRINT
     Y+5Ø; CHR$(217);
511Ø PRINT@193, "";
     : IF PO=1 PRINT H$(Ø); "'S BALL.  ";  ELSE
     PRINT O$(Ø); "'S BALL.";
5115 IF TT<T1 AND TT>T1-41 THEN TT=TT-4Ø
512Ø T1=INT(TT/1ØØ)*1ØØ
     : IF TT<Ø THEN TT=Ø
     : GOTO 512Ø ELSE PRINT@226, INT(TT/1ØØ)
     CHR$(8) ":";
     : T$=STR$(TT)
     : T$=RIGHT$(T$, 2)
     : PRINT T$ "  ";
     : IF TT=Ø THEN 58ØØ
5125 PRINT@216, QQ+1;
     : IF QQ=4 THEN PRINT@296, "OVERTIME";
513Ø RETURN
515Ø A$=INKEY$
     : PRINT@98Ø, "HIT SPACE BAR TO CONTINUE";
5175 IF INKEY$="" THEN 5175 ELSE PRINT@384,
     CHR$(31);
     : RETURN
52ØØ IF PO=1 THEN IF Y+Y9>49 THEN Y9=5Ø-Y
     : RETURN ELSE RETURN ELSE IF Y-Y9<-49 THEN
     Y9=5Ø+Y
     : RETURN ELSE RETURN
53ØØ FOR Z=1 TO 1ØØØ
     : NEXT
     : RETURN
54ØØ IF PO=-1 THEN 543Ø
541Ø PRINT
     : PRINT "KICKOFF RUN BACK BY "; H$(9); "
     FROM THE " RND(1Ø)-5 " TO THE ";
     : Y7=RND(35)-4Ø
     : IF Y7=-25 Y=55
     : PRINT "TOUCH DOWN !!!"
     : GOSUB 515Ø
     : GOSUB 55ØØ
     : RETURN ELSE PRINT Y7+5Ø
     : Y=Y7
     : IF Y<-49 THEN Y=-3Ø
     : Y2=-2Ø
     : RETURN ELSE Y2=Y+1Ø
     : RETURN
543Ø IF TT>Ø THEN PRINT@65, "SCORE
     :  "; H$(Ø); " -" TS "* "; O$(Ø); " -"CS;
     : PRINT@7Ø4, "";
544Ø INPUT "DO YOU WANT TO TRY AN ONSIDE KICK";
     A$
     : IF LEFT$(A$, 1)="Y" THEN IF RND(Ø)<.2 THEN
     PRINT "YOU RECOVERED IT AT THE "; O$(Ø); "
     49 YARD LINE."
     : PO=1
     : Y=1
     : T=Ø
     : RETURN ELSE PRINT O$(Ø); " RECOVERS THE
     BALL AT THE 5Ø YARD LINE."
     : PO=-1
     : Y=Ø
     : RETURN
545Ø PRINT
     : PRINT "KICKOFF RUN BACK BY "; O$(9); "
     FROM THE " RND(1Ø)-5 " TO THE ";
     : Y7=-RND(35)+5Ø
     : IF Y7=15 Y=-55
     : PRINT "TOUCH DOWN!"
     : GOSUB 515Ø
     : GOSUB 555Ø
     : GOTO 9ØØ ELSE PRINT 5Ø-Y7
     : Y=Y7
     : IF Y>49 THEN Y=3Ø
     : Y2=2Ø
     : RETURN ELSE Y2=Y-1Ø
     : RETURN
55ØØ PRINT "TOUCHDOWN, "; H$(Ø); " !!!"
     : TS=TS+6
     : IF QQ=4 THEN 6Ø5Ø
55Ø5 PO=-1
     : PRINT "EXTRA POINT.  "; H$(7); " KICKS."
     : GOSUB 53ØØ
     : IF RND(Ø)>.13 THEN PRINT "IT'S GOOD!"
     : TS=TS+1
     : GOSUB 515Ø
     : GOSUB 54ØØ
     : RETURN ELSE PRINT "NO GOOD."
     : GOSUB 5175
     : GOSUB 54ØØ
     : RETURN
555Ø PRINT "TOUCHDOWN, " O$(Ø); " !"
     : CS=CS+6
     : IF QQ=4 THEN 6Ø5Ø
556Ø D=1
     : PO=1
     : PRINT "EXTRA POINT.  "; O$(7); " KICKS."
     : GOSUB 53ØØ
     : IF RND(Ø)>.13 THEN PRINT "IT'S GOOD."
     : CS=CS+1
     : GOSUB 515Ø
     : GOSUB 54ØØ
     : RETURN ELSE PRINT "NO GOOD."
     : GOSUB 515Ø
     : GOSUB 54ØØ
     : RETURN
56ØØ PRINT@448,;
     : IF RND(Ø)<.97 THEN RETURN ELSE PRINT
     "PENALTY FLAG DOWN. ";
     : GOSUB 53ØØ
563Ø IF RND(Ø)<.5 THEN PRINT "IT'S ON "; O$(Ø);
     "."
     : PP=1 ELSE PRINT "IT'S ON "; H$(Ø); "."
     : PP=2
564Ø P8=RND(1ØØ)
565Ø IF P8<3Ø PRINT "HOLDING - ";
     : Y9=RND(2)*5
     : PRINT Y9 " YARDS."
     : GOTO 568Ø
5655 IF P8<5Ø PRINT "ILLEGAL PROCEDURE - 5
     YARDS."
     : Y9=5
     : GOTO 568Ø
566Ø IF P8<7Ø PRINT "CLIPPING - 15 YARDS."
     : Y9=15
     : GOTO 568Ø
5665 IF P8<95 PRINT "OFFSIDE - 5 YARDS."
     : Y9=5
     : GOTO 568Ø
567Ø GOTO 564Ø
568Ø IF PP=1 THEN IF Y9+Y>49 THEN
     Y=INT((Y+INT(Y+Y9))/2)-1 ELSE Y=Y+Y9 ELSE IF
     Y-Y9<-49 THEN Y=INT((Y+INT(Y-Y9))/2)+1 ELSE
     Y=Y-Y9
569Ø IF PP=2 THEN PRINT@285, Y2-Y;  ELSE
     PRINT@285, Y-Y2;
5699 GOSUB 51ØØ
     : RETURN
57ØØ PRINT "SAFETY!"
571Ø IF PO=-1 THEN TS=TS+2
     : IF QQ=4 THEN 6Ø5Ø ELSE PRINT "KICKOFF
     RETURN TO THE ";
     : Y9=RND(2Ø)+2Ø
     : PRINT Y9 " YARD LINE."
     : PO=1
     : Y=Y9-5Ø
     : GOTO 9ØØ ELSE CS=CS+2
     : IF QQ=4 THEN 6Ø5Ø ELSE PRINT "KICKOFF
     RETURN TO THE ";
     : Y9=RND(2Ø)+2Ø
     : PRINT Y9 " YARD LINE."
     : PO=-1
     : Y=5Ø-Y9
     : GOTO 95Ø
58ØØ IF QQ=Ø THEN GOSUB 515Ø
     : PRINT@384, CHR$(31); "END OF FIRST PERIOD"
     : GOSUB 53ØØ
     : TT=12ØØ
     : T1=TT
     : QQ=1
     : GOTO 51ØØ
582Ø IF QQ=1 THEN GOSUB 515Ø
     : PRINT@384, CHR$(31); "END FIRST HALF"
     : D=Ø
     : GOSUB 53ØØ
     : TT=12ØØ
     : T1=TT
     : QQ=2
     : IF KK=1 THEN PO=-1
     : PRINT "YOU KICKOFF TO START THE SECOND
     HALF." ELSE PRINT "YOU RECEIVE THE SECOND
     HALF KICKOFF."
     : PO=1 ELSE 583Ø
5825 GOSUB 54ØØ
     : GOSUB 51ØØ
     : GOTO 9ØØ
583Ø IF QQ=2 THEN GOSUB 515Ø
     : PRINT@384, CHR$(31); "END OF THE THIRD
     QUARTER"
     : GOSUB 53ØØ
     : TT=12ØØ
     : T1=TT
     : QQ=3
     : GOTO 51ØØ
584Ø IF QQ=3 THEN PRINT@384, CHR$(31); "END
     FOURTH QUARTER"
     : GOTO 6ØØØ
585Ø IF QQ=4 THEN PRINT@384, CHR$(31); "END
     OVERTIME"
     : GOTO 6Ø5Ø
6ØØØ IF TS<>CS THEN 6Ø5Ø ELSE QQ=4
     : PRINT@296, "OVERTIME";
     : PRINT@384, CHR$(31); "FIRST TO SCORE
     WINS."
     : TT=12ØØ
     : T1=TT
     : D=Ø
     : GOSUB 53ØØ
     : GOTO 1ØØ
6Ø5Ø FOR Z=1 TO 2Ø
     : PRINT@99Ø, "FINAL";
     : FOR ZZ=1 TO 2ØØ
     : NEXT
     : PRINT@99Ø, CHR$(197);
     : FOR ZZ=1 TO 2ØØ
     : NEXT
     : NEXT
     : PRINT@96Ø, CHR$(3Ø); "DO YOU WANT TO PLAY
     AGAIN ? (Y/N)";
6Ø6Ø A$=INKEY$
     : IF A$="Y" THEN RUN ELSE IF A$="N" THEN CLS
     : END ELSE 6Ø6Ø
```

Machine Language Sort, Part II

by William Barden, Jr.

Last month we were discussing a one-dimensional string array sort using Color Computer assembly language. This month we'll continue the discussion of how a typical assembly-language program is designed and implemented, using the SRTSTR sort routine as an example.

SRTSTR INSTANT REPLAY

Here's a recap for those of you that might not have last month's issue. SECSRT uses a "bubble-sort" approach to sort a one-dimensional string array of any length and any size of strings.

We saw how string arrays were organized as array lists of a "header" and 5-byte string "descriptor blocks" that defined the elements of the string array, as shown in Figure 1. The first byte of each descriptor block is the string length, while the third and fourth bytes hold the address of the string. The string may be embedded in a BASIC statement, or may be in the "string storage" area in high RAM.

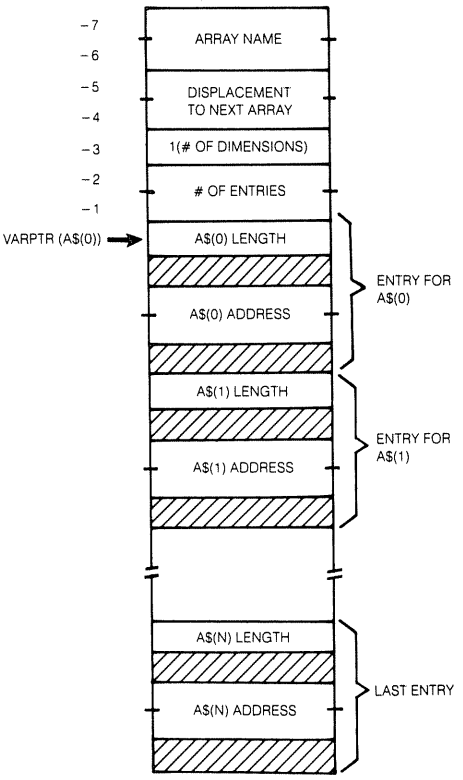


Figure 1. String Array Variable List

The header of the string array holds the number of entries in the array along with additional information about the array, including the name.

The location of the first string descriptor, and therefore, the string array list, can be found by VARPTR(A$(0)). The argument returned will be the address of the first string descriptor.

We discussed the design and flowcharting steps in SRTSTR in last month's column, and came up with a working flowchart as shown in Figure 2. The bubble-sort algorithm in the flowchart compares the entries in the string array one pair at a time, moving from top to bottom. A "lighter" bottom entry (one with lower ASCII weight) results in a swap of the two descriptor blocks. Multiple passes are made through the list until no entries have been swapped, indicating that all entries have been sorted in "ascending sequence." Only the descriptor blocks are swapped; the strings remain untouched.

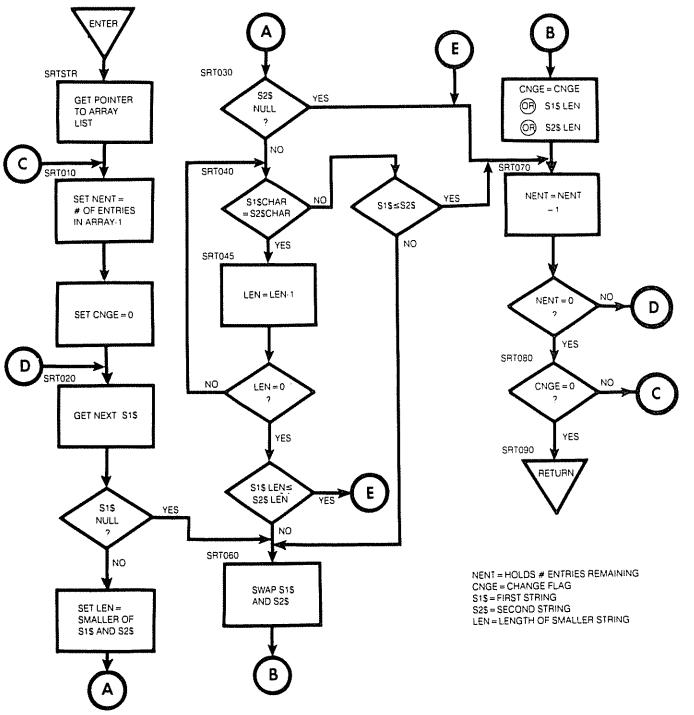


Figure 2. SRTSTR Flowchart

. . . AND NOW FOR THE CODING

If we've done our design and flowcharting properly, the assembly-language coding should be easy. The catch here is that it should be easy if you've done considerable 6809E programming. If you haven't, you'll have to struggle painfully through. In this case a good design is very important to minimize your frustration while you get experience in coding.

Let me share with you, as nearly as I can, the steps I used to translate the flowchart and design into code. The code of SRTSTR is shown in Listing 1.

```
3FØØ                    ØØ1ØØ          ORG    $3FØØ
                        ØØ11Ø  ************************************************
                        ØØ12Ø  * STRING ARRAY SORTER, SORTS 1-D STRING ARRAY     *
                        ØØ13Ø  * ENTRY:  LOC $3FFE,F=VARPTR(A$(Ø))              *
                        ØØ14Ø  * EXIT:   A$ ARRAY SORTED                        *
                        ØØ15Ø  ************************************************
3FØØ BE   3FFE          ØØ16Ø  SRTSTR  LDX    $3FFE      GET POINTER
3FØE 34   1Ø            ØØ17Ø          PSHS   X          SAVE IN STACK
                        ØØ18Ø  * START OF NEW PASS, EITHER FIRST OR ONE OF MANY
3FØ5 EE   E4            ØØ19Ø  SRTØ1Ø  LDU    ,S          GET PNTR TO ARRAY LIST
3FØ7 AE   5E            ØØ2ØØ          LDX    -2,U        GET # OF ENTRIES
3FØ9 3Ø   1F            ØØ21Ø          LEAX   -1,X        SUBTRACT ONE FOR PAIRS
3FØB 4F                 ØØ22Ø          CLRA               FOR CHANGE FLAG
3FØC 34   12            ØØ23Ø  SRTØ15  PSHS   A,X         SAVE IN STACK
                        ØØ24Ø  * THIS IS START OF COMPARE ENTRY LOOP
3FØE A6   C4            ØØ25Ø  SRTØ2Ø  LDA    ,U           GET LENGTH OF S1$
3F1Ø 27   2A            ØØ26Ø          BEQ    SRTØ6Ø       IF NULL, SWAP
3F12 A6   C4            ØØ27Ø          LDA    ,U           GET S1$ LEN
3F14 E6   45            ØØ28Ø          LDB    5,U          GET S2$ LEN
3F16 AØ   45            ØØ29Ø          SUBA   5,U          S1$LEN-S2$ LEN
3F18 24   Ø2            ØØ3ØØ          BHS    SRTØ3Ø       GO IF S1$ LEN>= S2$ LEN
3F1A E6   C4            ØØ31Ø          LDS    ,U           USE SMALLER LENGTH
                        ØØ32Ø  * NOW HAVE SENSE AND TOUCH
3F1C 34   Ø1            ØØ33Ø  SRTØ3Ø  PSHS   CC           SAVE SENSE
3F1E AE   42            ØØ34Ø          LDX    2,U          GET S1$ ADDR
3F2Ø 1ØAE 47            ØØ35Ø          LDY    7,U          GET S2$ ADDR
3F23 6D   45            ØØ36Ø          TST    5,U          TEST S2$ LEN
3F25 26   Ø4            ØØ37Ø          BNE    SRTØ4Ø       GO IF NOT NULL
3F27 32   61            ØØ38Ø          LEAS   1,S          RESET STACK
3F29 2Ø   29            ØØ39Ø          BRA    SRTØ7Ø       DO NOTHING
3F2B A6   8Ø            ØØ4ØØ  SRTØ4Ø  LDA    ,X+            GET S1$ BYTE
3F2D AØ   AØ            ØØ41Ø          SUBA   ,Y+            S1$-S2$
32F2 27   Ø4            ØØ42Ø          BEQ                   GO IF EQUAL
3F31 32   61            ØØ43Ø          LEAS   1,S            RESET STACK
3F33 2Ø   Ø5            ØØ44Ø          BRA    SRTØ5Ø         BYPASS SENSE PULL
3F35 5A                 ØØ45Ø  SRTØ45  DECB                  DECREMENT # OF BYTES
3F36 26   F3            ØØ46Ø          BNE    SRTØ4Ø         GO IF NOT DONE
                        ØØ47Ø  * DONE HERE EITHER FROM UNEQUAL OR AT END
3F38 35   Ø1            ØØ48Ø          PULS   CC           GET SENSE
3F3A 23   18            ØØ49Ø  SRTØ5Ø  BLS    SRTØ7Ø       GO IF S1$ <= S2$
                        ØØ5ØØ  * SWAP CALLED FOR EITHER FROM NULL S1$ OR NORMAL
3F3C AE   42            ØØ51Ø  SRTØ6Ø  LDX    2,U          GET S1$ ADDRESS
3F3E 1ØAE 47            ØØ52Ø          LDY    7,U          GET S2$ ADDRESS
3F41 AF   47            ØØ53Ø          STX    7,U          SWAP
3F43 1ØAF 42            ØØ54Ø          STY    2,U
3F46 A6   C4            ØØ55Ø          LDA    ,U           GET S1$ LENGTH
3F48 E6   45            ØØ56Ø          LDB    5,U          GET S2$ LENGTH
3F4A E7   C4            ØØ57Ø          STB    ,U           SWAP
3F4C A7   45            ØØ58Ø          STA    5,U          SWAP
3F4E EA   45            ØØ59Ø          ORB    5,U
3F5Ø EA   E4            ØØ6ØØ          ORB    ,S           MERGE IN PREVIOUS
3F52 E7   E4            ØØ61Ø          SIB    ,S           SET CHANGE FLAG
                        ØØ62Ø  * CONTINUE THROUGH LIST UNTIL END
3F54 33   45            ØØ63Ø  SRTØ7Ø  LEAU   5,U          POINT TO NEXT ENTRY
3F56 AE   61            ØØ64Ø          LDX    1,S          GET # ENTRIES
3F58 3Ø   1F            ØØ65Ø          LEAX   -1,X         DECREMENT BY 1
3F5A AF   61            ØØ66Ø          STX    1,S          STORE AGAIN
3F5C 26   BØ            ØØ67Ø          BNE    SRTØ2Ø       GO IF NOT AT END
                        ØØ68Ø  * AT END OF ONE PASS HERE
3F5E A6   E4            ØØ69Ø  SRTØ8Ø  LDA    ,S          GET CHANGE FLAG
3F6Ø 32   63            ØØ7ØØ          LEAS   3,S          RESET STACK
3F62 26   A1            ØØ71Ø          BNE    SRTØ1Ø       GO IF SWAP OCCURRED
3F64 32   62            ØØ72Ø  SRTØ9Ø  LEAS   2,S       RESET STACK
3F66 39                 ØØ73Ø          RTS             BACK TO BASIC
         ØØØØ           ØØ74Ø          END
```

Listing 1

START IN THE MIDDLE

First of all, I started in the middle, with the actual compare and swap. This is the heart of SRTSTR, and many people start at the main functions and add the beginning and end later.

I knew that I would have to compare a string of bytes with another string of bytes in two separate locations. The comparisons would be made of two bytes with the same relative locations in each string, and some kind of pointer would have to be incremented for both strings to point to the next byte.

The first thing that the above action makes a 6809E programmer think of is "indexing" using the 6809E index registers, X and Y. The X and Y index registers can be used in an "auto increment mode" where they are automatically incremented by one (or two) after each operation. As there are two index registers and two strings, using the index registers seems a natural for the comparison. The scheme is shown in Figure 3, where X points to the first string for the compare, and Y points to the second string for the compare. The actual compare is done by the two instructions at SRT040, and is really a subtract. The "+" sets auto-increment mode for X and Y and automatically adds one to X or Y **AFTER** the LDA and SUBA. The auto-increment mode is used in lieu of an instruction like INCA or INCB.

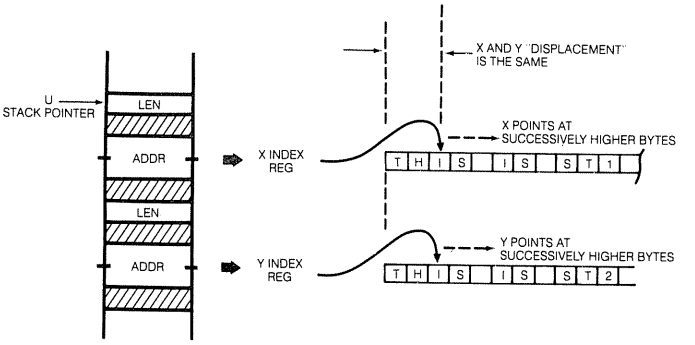


Figure 3. X and Y Index Register Use

Unfortunately, the 6809E has only two index registers to point to data. In SRTSTR we have to point to three basic things — the current pair of string descriptors in the array list (one address is sufficient, as they follow each other) and the current strings to be compared (two addresses). How can we also point to the current pair of string descriptors as well? We could save the addresses in memory variables and load X and Y as required. Another way, however, is to use the User Stack Pointer register.

IS THIS-A U STACK?

Let's digress, somewhat, and discuss 6809E stacks. The 6809E and other microprocessors have traditionally used a RAM memory stack. The stack is a 50 to 100 byte area of RAM set aside to hold temporary results, interrupt addresses, and subroutine addresses.

Whenever a BSR, JSR, or similar type of branch to subroutine instruction is executed in the 6809E, the return address after the branch to subroutine is automatically saved in the stack. A subsequent RTS (Return from Subroutine) at the end of the subroutine code "pulls" the return address and causes a branch back to the instruction following the branch to subroutine.

A PSHS instruction "pushes" one or more 6809E registers into the stack; a PULS instruction retrieves the registers.

During interrupts, the "environment" (registers and condition codes) and interrupt point address are automatically saved in the stack. An RTI (Return from Interrupt) restores the registers and returns to the point of interrupt. The interrupt occurs with no effect on any "background" program.

The stack used for all of the above operations is called the "S" stack in the 6809E and is equivalent to similar types of stacks in other microprocessors, such as the 8080, Z-80, and 6502. The "S" stack uses a stack pointer register to define the RAM memory address for the stack operation. The stack "builds down" in RAM as data is pushed onto the stack and the S register is adjusted accordingly. Data pulled from the stack adjusts the stack pointer upward.

The 6809E uses another stack, which operates similarly to the "S" stack. This "U", or User, stack can be

defined by loading a U stack pointer register to define the stack area. Unlike the S stack, there are no hardware operations (such as BSR or interrupt address pushes) that automatically use the U stack. The U stack is for the user only; the user (programmer) may use the U stack for any data he wants. The advantage of using the U stack is that many instructions allow the use of the U stack pointer in identical fashion to the X/Y index registers or S stack pointer.

To get back to the thread of our discussion . . . Because the U stack can be defined by the user to be any block of memory and because there are many instructions that operate in a U indexed addressing mode, we'll use it to advantage in SRTSTR. We'll use the U stack pointer as a pointer to the current pair of string descriptor blocks in the string array, as shown in Figures 3 and 4.

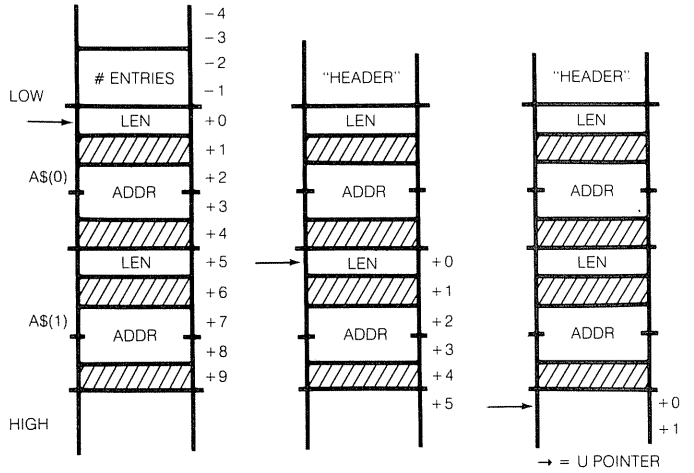


Figure 4. U Stack Use

The U stack pointer is initially set up to point to A$(0) at SRT010. As compares are made, it is incremented by 5 to point to the next set of string descriptors. The LEAU 5,U instruction at SRT070 does the increment. Swaps, when called for, are done by using the instructions at the SRT060 area. These are largely U stack indexed instructions such as "LDY 7,U" which use the U stack pointer plus a displacement to determine the "effective address" to be used in the instruction.

The S stack register is used for subroutine addresses and for temporary storage in SRTSTR, as shown in Figure 5. When SRTSTR is called from BASIC, the return point in the BASIC interpreter is saved in the S stack. Immediately upon entry to SRTSTR, the address of the string array is pushed into the S stack (PSHS X). During SRTSTR execution, the S stack is used to hold the working variable for the number of entries, the "change flag," and a "sense" variable, as shown in the figure.

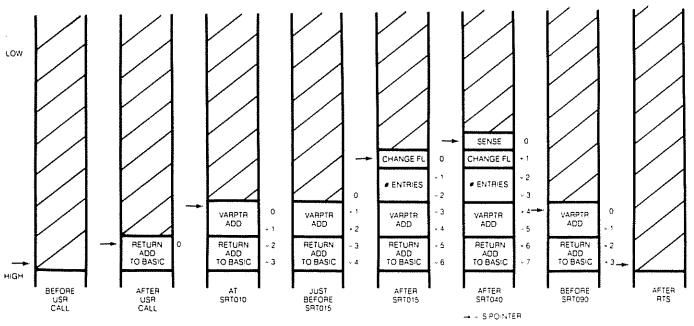


Figure 5. S Stack Use

ACCUMULATORS

The 6809E has two accumulators, A and B, which are used to perform arithmetic operations, shifts, and other operations. The A accumulator in SRTSTR is generally used as a "working" accumulator, while B holds the length of the smaller string during the compare loop (SRT040).

DETAILS, DETAILS . . .

Now that we've defined what the registers and stacks are used for, let's take a look at the actual code. Bear in mind that this code was the result of several "iterations." The original code was modified several times as the result of finding errors during debugging. It is moderately hard code, and took hours to prepare.

First of all, the pointer to the string array list is loaded into the X register from locations $3FFE and $3FFF. The pointer is then saved in the S stack by the PSHS X. This pointer was stored in the BASIC program after the address was found by a VARPTR(A$(0)). The VARPTR address points to A$(0), with the "header" information in the seven bytes before A$(0). The S stack at this point was set and maintained in BASIC.

The code from SRT010 through one instruction before SRT090 is the main loop of SRTSTR. It defines one complete pass through the array list and is executed until the change flag is zero.

At SRT010, the U stack pointer is loaded with the array address from the S stack. The S stack pointer points to the array address at this point, as shown in Figure 5.

Next, the number of entries is picked up from the header and loaded into X. Note that the "effective address" is the U stack pointer-2. One is subtracted from the number of entries in X by the LEAX −1,X, which loads the contents of X minus one into X. This is identical to a decrement X. The A accumulator is cleared for the change flag. The PSHS at SRT015 saves the number of entries and change flag in the S stack as shown in Figure 5. The number of entries will be adjusted throughout the SRT010 loop, and the change flag will be set for any significant change.

The code from SRT020 through one instruction less than SRT080 is the loop that compares each pair of descriptor blocks. During this loop, U points to the current pair and is adjusted by 5 to access all pairs.

The 7 instructions at SRT020 find the smaller of the two string lengths and puts the smaller length in B. They also test for a null entry in S1$, which results in an automatic swap (see the flowchart).

At SRT030, the A register holds the "sense" of the length of S1$ and S2$; this is a subtract of the two lengths and will be used if the compare of the two strings stops because the end of the smaller string was reached. In this case a swap will be made if the S1$ length is greater than the S2$ length. This sense is really held in the condition codes, which reflect the result of the subtract in the carry and zero flags. The condition codes are saved in the S stack at SRT030.

The X and Y index registers are now loaded with the addresses of the two strings from the current pair. The U stack pointer points to the first byte of the 10 bytes of the pair at this point. If the S2$ string is a null, no swap is made and a branch is made to SRT070. Before the branch, the S stack is "reset" by adding one to the stack pointer. This action is in lieu of a PULS CC.

The code from SRT040 through one instruction after SRT045 is the compare loop for the two strings. At this point X and Y point to the two strings and B holds the smaller length. The compare is actually a subtract of each of the two bytes with automatic indexing on X and Y. If any two bytes are not equal, a branch is made to SRT050. If the branch is made, the A register holds the "sense" of the comparison. If the length count in B is decremented beyond 0, the sense is "pulled" from the stack and used at SRT050.

The swap code at SRT060 uses the U stack pointer in indexing modes to swap the two descriptor blocks in the string array. Actually, only the length and addresses in the descriptor blocks are swapped. This is sufficient as the remaining two bytes are used only for temporary storage. If the swap is done, the change flag is set by ORing in the current value and the length of S1$ and S2$.

The location at SRT070 is entered after every compare, swap or not. An LEAU 5,U increments the U stack pointer by 5 to point to the next pair. The working count of the number of entries is decremented by one. If this is non-zero, the loop at SRT020 is reentered. If the count is zero, the code at SRT080 is executed.

The code at SRT080 is executed at the end of every pass. The change flag is tested and if non-zero, another pass through SRT010 is made. If the change flag is zero, the stack is reset and a return is made back to the BASIC interpreter.

DEBUGGING AND DOCUMENTATION

Last month we discussed the steps in developing any assembly-language program. We've covered design, flowcharting, and coding up to this point. We'll give short shift to debugging here, as it would make an interesting column all by itself. Debugging normally takes at least as long as coding and assembling, and in larger programs occupies the major part of the total job.

Documentation in many programs, large or small, is terrible. Many programmers don't want to take the time to document programs, either in comments for the code itself or in a written specification. Suffice it to say that good documentation pays for itself in "program maintenance" and operating ease. Of course, I've documented SRTSTR to death because of this column, but minimum documentation should include a comment on just about every line of the assembly language code, plus a "header block" describing the subroutine.

USING SRTSTR

At the beginning of these two columns I promised you a "neat" program. I think SRTSTR fills the bill because it does perform a useful task on a string array — a high-speed sort of any sized array with any sized entries.

SRTSTR is relocatable, and can be placed anywhere in memory you desire. The listing shown in Listing 2 is a full blown Extended Color BASIC exerciser for SRTSTR. It moves the code from DATA statements to the $3F00 area, generates a 101-element string array filled with random data, calls SRTSTR, and then displays the sorted array. Use it as a model for your own use of SRTSTR.

```
10Ø ' SRTSTR EXERCISER #2
11Ø CLEAR 5ØØØ,&H3EFF
12Ø A=Ø
     : N=Ø
     : I=Ø
     : J=Ø
13Ø DATA 19Ø,63,254,52,16,238,228,174,94,48,31,79,52,
     18,166,196,39,42,166,196,23,Ø,69,16Ø,69,36,2,23Ø,
     196,52,1,174,66,16,174,71
14Ø DATA  1Ø9,69,38,4,5Ø,97,32,41,166,128,16Ø,16Ø,
     39,4,5Ø,97,32,5,9Ø,38,243,53,1,35,24,174,66,16,
     174,71,175,71,16,175,66,166
15Ø DATA 196,23Ø,69,231,196,167,69,234,69,234,228,
     231,228,51,69,174,97,48,31,175,97,38,175,166,228,
     5Ø,99,38,161,5Ø,98,57
155 B=Ø
16Ø FOR I=&H3FØØ TO &H3F66
17Ø READ A
     : POKE I,A
175 B=B+A
18Ø NEXT I
19Ø IF B<>111Ø8 THEN PRINT "INVALID VALUES IN DATA
     STATEMENTS - CHECK!"
     : STOP
2ØØ DIM A$(1Ø)
21Ø DEFUSRØ=&H3FØØ
22Ø PRINT "***RANDOM ARRAY***"
23Ø FOR I=Ø TO 1ØØ
24Ø IF RND(1Ø)=5 THEN GOTO 28Ø
25Ø FOR J=Ø TO RND(15)
26Ø A$(I)=A$(I)+CHR$(RND(58)+32)
27Ø NEXT J
28Ø PRINT A$(I)
29Ø NEXT I
3ØØ N=VARPTR(A$(Ø))
31Ø POKE &H3FFE,INT(N/256)
32Ø POKE &H3FFF,N-INT(N/256)*256
33Ø PRINT "***SORT STARTING***"
34Ø A=USRØ(Ø)
35Ø PRINT "***SORT ENDED***"
36Ø FOR I=Ø TO 1ØØ
37Ø PRINT A$(I)
38Ø FOR J=Ø TO 5Ø
     : NEXT J
39Ø NEXT I
```

Listing 2

If you ran the BASIC version of SRTSTR is last month's column you'll find that the assembly-language version of SRTSTR runs somewhat faster — hundreds of times! So you see, all of this trouble in Color Computer assembly-language is worth while. (In spite of the disgruntled reader from Parvenu, CA who insisted that 6809E assembly language be X-rated.)

Next month we'll have more assembly-language topics. In the meantime, try your own hand at some assembly-language coding. You'll be pleasantly surprised.

Graphic Keys

Robby Bennett
608 Taft Avenue
Cheyenne, WY 82001

I am the proud owner of a TRS-80 Model III. I fully enjoy the capabilities of this computer.

This program will convert the text keys to special characters or graphics.

NOTE: I have disabled the BREAK key so that the user will not interrupt CHR$(21). You will need to press CLEAR to end the program.

```
5 GOSUB 85
7 LPRINT CHR$(21);
1Ø CLS
12 PRINT CHR$(23);
15 G$=CHR$(1Ø)+"READY"+CHR$(1Ø)+">"
2Ø PRINTCHR$(14);"READY";CHR$(1Ø);">";
3Ø X$=INKEY$
     : IF X$="" THEN 3Ø
31 POKE 164Ø9,1
33 IF X$=CHR$(31) THEN PRINT CHR$(15);CHR$(21);
     : POKE 16396,2Ø1
     : END
35 IF X$=CHR$(13) GOSUB 8Ø
4Ø IF X$=CHR$(8) AND POS(Ø)=<2 THEN 3Ø
45 IF X$=CHR$(8) PRINT X$;
     : GOTO 3Ø
5Ø IF ASC(X$)<32 OR ASC(X$)>91 THEN 3Ø
6Ø IF C%=2 LET X$=CHR$(ASC(X$)+97)
62 IF C%=1 LET X$=CHR$(ABS(X$)+161)
7Ø PRINT X$;
     : GOTO 3Ø
8Ø PRINT G$;
     : RETURN
85 PRINT "1=SPECIAL CHARACTERS";CHR$(1Ø);"2=GRAPHIC
   CHARACTERS"
87 INPUT "ENTER YOUR CHOICE ";C%
9Ø IF C%<1 OR C%>2 THEN 85
95 POKE 16396,175
     : POKE 16397,2Ø1
     : RETURN
```

Model I/III Head Cleaner

Robert E. Brown
2 Christina Drive, East
Schenectady, NY 12303

Sometime soon (I hope), TRS-80 Microcomputer News will realize the difference between the memory-mapped Model I and the port-based Model III. Enclosed is a revision to Marel Sooke's head cleaning program which WILL run on both Model I and III. In addition, it moves the head to a randomly-chosen track on the cleaning diskette so as to get maximum use from the investment. I hope that you will publish more of this kind of program listing so that we Model III users can study the differences between the two RADIO SHACK computers.

All teasing aside, TRS-80 Microcomputer News is terrific! I look forward to its arrival each month. Keep up the good work!

P.S. This program is loosely based on another head cleaning program for the Model I (only) which appeared in the March 1982 issue of 80-U.S. Journal, pages 82-83, but the listing enclosed is quite different and bears no resemblance to that program by William H. Scott, Jr.

MODEL I/III DISK HEAD CLEANER PROGRAM

```
5 ' CLEAN1/3 REVISED BOB BROWN 3/1/82
1Ø ' *** INSERT # OF LAST DRIVE (Ø-3) ON YOUR
     SYSTEM (BELOW)
15 LASTDRIVE=1
2Ø CLS
     : PRINT
     : PRINT "REMOVE ALL DISKS FROM DRIVES AND
     PRESS <ENTER>";
     : INPUTA
25 FOR N=Ø TO LASTDRIVE
     : IF N=Ø THEN DR=1 ELSE IF N=1 THEN DR=2
     ELSE IF N=2 THEN DR=4 ELSE IF N=3 THEN DR=8
3Ø ' *** DETERMINE IF MOD 1 OR MOD 3
35 IF PEEK(293)<>73 THEN M=1 ELSE M=3
4Ø ' *** SELECT A RANDOM TRACK FOR HEAD
     POSITIONING
45 IF M=1 THEN TR=RND(35) ELSE TR=RND(4Ø)
5Ø ' *** SELECT THE DRIVE
55 IF M=1 THEN POKE 143Ø4,DR ELSE OUT 244,DR
6Ø ' *** ISSUE "RESTORE" TO MOVE HEAD TO TRACK Ø
65 IF M=1 THEN POKE 14316,3 ELSE OUT 24Ø,3
7Ø ' *** LOOK FOR FDC "NOT BUSY" STATUS
75 IF M=1 THEN ST=PEEK(14316) ELSE ST=INP(24Ø)
8Ø IF (ST AND 1)=1 THEN 75
85 ' *** SELECT DRIVE AGAIN
9Ø IF M=1 THEN POKE 143Ø4,DR ELSE OUT 244, DR
95 ' *** MOVE HEAD OUT SOME PLACE
1ØØ IF M=1 THEN POKE 14319,TR ELSE OUT 241,TR
1Ø5 IF M=1 THEN POKE 14316, 19 ELSE OUT 24Ø,19
11Ø ' *** CHECK FDC STATUS AGAIN
115 IF M=1 THEN ST=PEEK(14316) ELSE ST=INP(24Ø)
12Ø IF (ST AND 1)=1 THEN 115
125 PRINT"INSERT CLEANER DISK IN DRIVE #";N;"
     THEN HIT <ENTER>";
13Ø INPUT A
135 ' *** ACTIVATE DRIVE FOR ABOUT 3Ø SECONDS
14Ø FOR I=1 TO 33ØØ
145 IF M=1 THEN POKE 143Ø4,DR ELSE OUT 244,DR
15Ø NEXT I
155 NEXT N
16Ø PRINT"*** CLEANING COMPLETED ***"
     : END
```

Graphic Basics

The theme of the NEWSLETTER this month is GRAPHICS and we decided to present some of the fundamentals of graphics. We won't make experts out of you, but we just might give your imagination something to work with. Our discussion here will be about the Models I and II.

Graphics can be broken down into three general categories:

1. Line Printer Graphics
2. Pixel Graphics
3. Character Graphics

Line printer graphics refers to pictures, drawn either on the printer or the video screen, made up of characters printed on specific positions on the screen. This type of graphic is like a still photograph. Some examples of line printer graphics include bar charts for business and wave form charts for scientific applications. Among the more frivolous uses are party banners and simple pictures.

To draw line printer graphics, you must first lay out your picture on a video worksheet. This way you can easily determine the position for each character and the character you want to use. Here is a sample program.

```
10 CLS
     : CLEAR 1000
     : PRINT
20 PRINT "    **          **        **"
30 PRINT "    **          **        **"
40 PRINT "    ** ** ** ** **        **"
50 PRINT "    **          **        **"
60 PRINT "    **          **        **"
70 PRINT
80 PRINT "**          **     **         **          **       **"
90 PRINT " **        **     **       **    **       **       **
100 PRINT "   **    **              **      **      **       **"
110 PRINT "      **                 **  **  **      **       **"
120 PRINT "      **                 **      **      **       **"
130 PRINT "      **                 **      **      ******   ******"
140 GOTO 140
```

Try it and see the message we've got for you!! To print on your line printer, change the PRINTs to LPRINTs. Line Printer graphics have their function, but let's see what we can do with the other types of graphics.

So far we've learned how to print single characters in specific positions on the screen to form pictures. Now let's examine what makes up a graphics character. The basic element of a graphics character is the pixel. This leads us to our next type of graphics — pixel graphics. Each graphics block is made of six sections, each called a pixel. The Model I and III screen will display sixty-four characters across one line. A full screen has sixteen (16), sixty-four (64) character lines. Each character, however, is two pixels wide. This means that there are 64*2 or 128 pixels across. Each character is three pixels high, giving us 16*3 or 48 pixels up and down. When we write programs using pixel positions we will refer to positions 0 to 127 across and 0 to 47 up and down. We will also refer to the screen in terms of X and Y coordinates, where X is a horizontal position on the screen and Y is a vertical position on the screen. The three basic commands used in pixel graphics are "SET", "RESET", and "POINT".

Let's start with the "SET" command. SET is used to turn on the pixel at a particular point. The correct syntax is SET(X,Y). For example, SET (35,25) lights up the block thirty-six pixels from the left of the screen and twenty-six pixels from the top of the screen. The RESET command turns off the pixel. The syntax for RESET is RESET(X,Y). Using the same example above RESET(35,25) turns off the pixel we turned on with the SET command. Try it yourself. Here is a simple program using the SET and RESET commands.

```
10 CLS
20 X=35
     : Y=25
30 SET(X, Y)
40 FOR Z=1 TO 150
     : NEXT
50 RESET(X, Y)
60 FOR Z=1 TO 150
     : NEXT
70 GOTO 30
```

The program flashes the pixel at the 35,25 position. Remember the 0 position puts this at the thirty-sixth position from the side of the screen and the twenty-sixth position from the top of the screen. Notice the timing loops in lines 40 and 60. Without these loops the flashing of the pixel would be almost imperceptible. Of course this is a very simple demonstration of the SET and RESET commands. By adding one line we can make the blinking dot move.

```
65 X=X+5: IF X>127 THEN X=X-127
```

To make the dot move diagonally we can add a second line.

```
67 Y=Y+5: IF Y>47 THEN Y=Y-47
```

Notice that when the dot reaches the bottom of the screen it reappears at the top of the screen. Can you make it bounce instead? By using various combinations of SET and RESET commands, one can design some relatively sophisticated graphics programs. Another command used in conjunction with the SET and RESET commands is the POINT command. POINT looks at a specified graphics block and can tell you if it is turned off or on. The syntax used is POINT(X,Y). If the block is "on" the value returned by POINT is a −1. If the block is "off" then POINT will return a 0. This is useful if you want to check to see if a specific block is "on" or "off" and have the BASIC program take some action when the desired conditions are met.

Now let's discuss character graphics. Character graphics take advantage of the TRS-80's ability to print a complete graphics block as a single character. These are known as character strings or as used in a BASIC program, CHR$(X). In the TRS-80 Model I character set there are 64 graphics characters, made up of different combinations of lit pixels. There are also space compression codes on both the Model I and III. Space compression

codes are a series of blank spaces from 0 to 63, which you can print as a single character string. The space compression codes are CHR$ 192 to 255 on both the Model I and III. The Model III has the same graphics characters as the Model I plus a special character set that has such characters as the English pound sign, the "happy face," a little man, a little woman, and many others. Any of these could be used in your graphics programs. These special characters are the same codes as the space compression codes, 192 to 255. However, there is a software switch to turn them on and off. To access these special characters you must turn them on with PRINTCHR$(21). If you do not PRINT CHR$(21) then the character string that you will print will be a space compression code. To turn off the special graphics, simply PRINT CHR$(21) a second time.

With various combinations of the 64 graphics codes, you can do many of the same things that you can do with pixel graphics, but because you are printing six pixels at a time, character graphics will run faster. Here is a program that demonstrates the use of both pixel and character graphics.

```
10 CLS
20 FOR Y=1 TO 47
30 FOR X=1 TO 127
40 SET(X, Y)
50 NEXT X
     : NEXT Y
60 CLS
70 FOR Y=1 TO 1024
80 PRINT CHR$(191);
90 NEXT Y
100 PRINT@540, " THE END ";
110 GOTO 110
```

The program will white out the screen using pixel graphics, clear the screen, and white it out again using character graphics. You can easily see the speed advantage of the character graphics.

Here are two short programs that demonstrate the use of character graphics, one for both the Model I and III and the last one for the Model III only.

Model I and III Graphics Program

```
10 CLS
     : CLEAR 2000
20 PRINT@384, STRING$(64, 191);
     : PRINT@896, STRING$(64, 191);
30 PRINT@540+X, CHR$(174); CHR$(140); CHR$(140);
     CHR$(153);
40 PRINT@540+(X-1), CHR$(193);
50 X=X+1
     : Z=Z+1
60 IF Z=>80 THEN GOSUB 90
70 IF Z=>180 THEN 130
80 GOTO 20
90 PRINT@540+Y, CHR$(174); CHR$(140); CHR$(140);
     CHR$(157);
100 PRINT@540+(Y-1), CHR$(193);
110 Y=Y+1
120 RETURN
130 PRINT@501, CHR$(191);
     : PRINT@885, CHR$(191);
140 PRINT@128, "WATCH OUT!!!  BARRIER AHEAD!!!";
150 IF Z=260 THEN 170
160 GOTO 20
170 PRINT@920, "THE END";
     : GOTO 170
```

Model III Graphics Program

```
10 REM REMEMBER TO PRINT CHR$(21)TO TURN ON THE
     SPECIAL GRAPHICS
20 CLS
     : CLEAR 1000
     : PRINT CHR$(23)
30 PRINT@540+X, CHR$(234);
40 PRINT@520+Y, CHR$(253);
50 PRINT@512+Z, CHR$(254);
60 PRINT@540+(X-1), CHR$(32);
70 PRINT@520+(Y-1), CHR$(32);
80 PRINT@512+(Z-1), CHR$(32);
90 X=X+1
     : Y=Y+1
     : Z=Z+1
100 IF X=100 THEN 140
110 IF X=-28 THEN 160
120 PRINT@512, "WILL THEY CATCH THEIR PET?";
130 GOTO 30
140 X=-78
     : Y=-64
     : Z=-64
150 GOTO 30
160 PRINT
     : PRINT@512,"GOTCHA!!!"
     : CHR$(254); CHR$(234); CHR$(253)
170 GOTO 170
```

We hope you have learned a little bit about the graphics capability of the Models I and III and will be able to expand on these basic ideas to write your own graphics programs.

Effective June 1, 1982 Computer Customer Services Address and Phone Number

8AM to 5PM Central Time

Computer Customer Services
400 Atrium, One Tandy Center
Fort Worth, Texas 76102

Model I/III Business Group	817-870-2041
Model II/16 Business Group	817-870-2042
Languages and Compilers	817-870-2044
Color and Pocket Computer Group	817-870-2150
Hardware and Communications Group	817-870-2571
Educational Software	817-390-3302
Games, Books, and New Products	817-870-2271

New E-Mail Service

Editor's Note: The CompuServe Information Service is one of the largest information and entertainment services available to owners of personal computers and computer terminals. With each issue of TRS-80 Microcomputer NEWS, various features of CompuServe will be discussed.

The CompuServe Information Service is sold at Radio Shack stores nationwide and in Canada.

New E-Mail Gets the Message Fast!

E. is not your everyday mail carrier. Bad weather, postal strikes and federal holidays don't slow him down.

He's not much for idle chatter, but he'll quickly tell you who your mail is from, what it's about and when it's about and when it arrived.

He's considerate. An obviously full mailbox will not be stuffed to overflowing.

And — wonder of wonders — after you've read it, he'll throw away what you don't want and file the rest.

E-MAIL, or electronic mail, is to the computer age what the Pony Express was to the frontier era: an innovative way to get there first and fastest.

CompuServe's newly-revamped electronic mail system is a logical, simplified, and user-friendly system that offers an inexpensive and efficient method for communicating. Users access the E-MAIL feature by first making a local phone call to hook into the CompuServe network. After logging on, electronic mail can be accessed directly through the Home Services, Business and Finance, and Personal Computing menus.

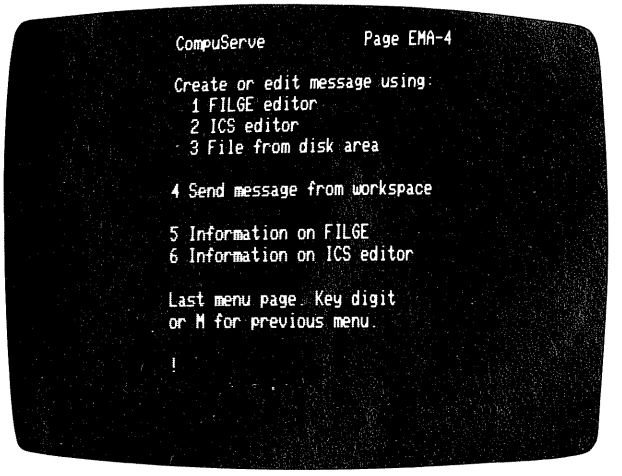
E-MAIL is stored by subject headings, which also include the sender's name and the date sent, allowing the receiver to peruse his "mailbox" at a glance and select the order in which the items will be read. Each "mailbox" will store a maximum of 20 items, and those which do not receive the command to be filed will be deleted. No need to worry if you're the absent-minded type — you must respond to the question of filing with a yes or no response to each piece of mail before dealing with the next. This method eliminates problems arising from users putting off, and then not recalling, the action taken on a particular piece of mail.

Under the revamped system, mail can be added to existing files or new files can be started, depending on the user's commands. In every instance, the user is given feedback on what action has been taken and E-MAIL commands are consistent with the commands used in other areas. The electronic mail feature uses standard editors — ICS and FILGE.

Electronic mail is easy to use, but how practical is it? The cheapest phone call from Columbus to Los Angeles (after 5 p.m.) costs 42 cents for the first minute and 30 cents for each additional minute — for a grand total of $18.12 plus tax for a one hour call. While it's unlikely that the use of electronic mail would supplant such a long phone conversation, it would solve the problem — cheaply — of callers who continue to miss one another's calls.

Unlike other systems, CompuServe's system does not require that both sender and receiver have computers. Two terminals are all that are needed with the CompuServe network. Inexpensive, portable terminals can be used by those on the go to connect with CompuServe almost anywhere and at anytime. Compared to overnight mail services — one of which charges upwards of $20 per item — the cost of using E-MAIL is very attractive indeed.

Its speed, combined with the features of privacy, easy access and storage capabilities, may well make electronic mail one of the best communications options available.



CompuServe at 29,000 Mark and Still Growing

More than 29,000 people, from all walks of life and from all areas of the country, are now customers of the CompuServe Information Service (CIS).

The 29,000 figure reflects the actual number of paying customers and does not include demonstration accounts, media accounts, or employees.

CIS started in August 1979 and reached the 10,000 customer mark in May 1981.

The geographic distribution of customers shows users in every state and throughout Canada. The largest concentration of customers are in and around New York City and the eastern seaboard, the "Silicon Valley" area of California and the areas around Los Angeles, populous areas of the South, and industrial cities of the Midwest.

CIS provides information and entertainment services to owners of personal computers and computer terminals in the main subject areas of news, finance, entertainment,

home and family information, electronic mail, and personal computing.

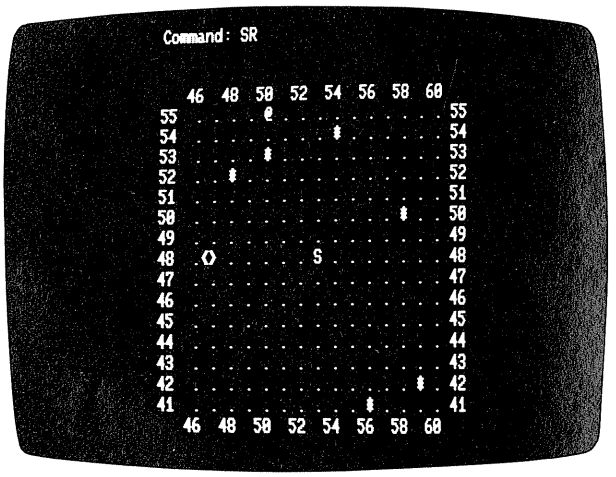
Customers access CompuServe through a local telephone call in more than 300 U.S. cities and in Canada.

CIS is sold in Radio Shack Computer Centers and electronics stores throughout the U.S. and Canada and through various computer and data equipment manufacturers.

CIS can be used with all popular makes of personal computers and many computer terminals.

Decwars: Interactive Space Game on CompuServe

The vast and inscrutable galaxy appears tranquil, but Stormcrow knows the surface calm conceals many dangers. A check with other Federation forces confirms his fears: the Klingons have already established 10 starbases while his own human forces lag five behind.



Klingons — large, forceful creatures with round faces and evil eyes — are not the only threat to the human Federation's survival. Immense caverns of nothingness dot the skies, assuring instant death to all who fall within these black holes. An evil Romulan concealed in a cloaking device may make surprise attacks on humans and Klingons alike. And even neutral planets take pot shots at careless captains who venture to within two sectors of their locations.

Undeterred by what he hopes is a temporary setback for Federation forces, Stormcrow puts out a jaunty APB over his subspace radio: "Tell all Stormcrow is here. Prepare to die, Empire monsters!"

Encouraged by the plucky captain's verbal swaggering, the Federation's Nimitz ship radios back: "Now is the time for all good men to come to the aid of their planet!"

Battered Federation forces are substantially strengthened by Stormcrow's entry. His undamaged ship brings with it 5000 units of shield energy to protect it from phaser and photon torpedo hits. His as yet unimpaired photon torpedo tubes pose a serious threat to the enemy due to their 10-sector hit range. And, perhaps, most important to the ailing Federation forces, his strength can be used to activate tractor beams to tow damaged ships away from danger.

A challenge to battle from an Empire ship named Jackal is put on hold while Stormcrow moves to capture a neutral planet. A transfer of materials and energy builds it into a starbase with the power to fire upon enemy ships within a five-sector range.

Meanwhile, his sub-space radio keeps Stormcrow informed of developments in other parts of the galaxy. An empire ship has built three more starbases and captured another planet. At 1000 points per base and 100 per planet, the enemy has added 3100 points to his individual score.

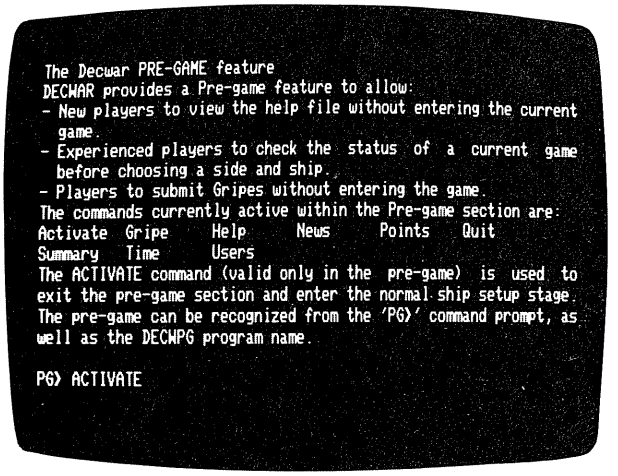
On the home front, the Nimitz rallies to make a deadly hit on an Empire ship, and the Demon is destroyed. "Let's party," he declares triumphantly to Stormcrow. But before the galactic celebration can take place, the Nimitz, too, bites the stardust — the victim of a sneak attack by the Empire ship, Cobra.

Designed for up to 10 players, Decwars is accessed under main menu item 2, Home Services. The first person to log onto the game chooses a Federation ship, while the next player gets an Empire ship. This alternation of ships continues so that sides are as evenly divided as possible.

Arranged in a grid of 75 by 75 sectors, Decwars' galaxy allows for freedom of movement. Ships can move from sector to sector to defend themselves and to attack enemy installations and ships.

Players are free to enter and leave the game as desired. And though death — as with the valiant Nimitz — may come as an instantaneous shock, it is a temporary fate. Destroyed ships may immediately log back into the game to begin racking up new scores in the perpetual battle between the forces of good and evil.

Questions and comments about the CompuServe Information Service can be sent to Richard A. Baker, editorial director, CompuServe Information Service, P.O. Box 20212, Columbus, Ohio 43220 or through Feedback on the User Information menu.



You Can Quote Us

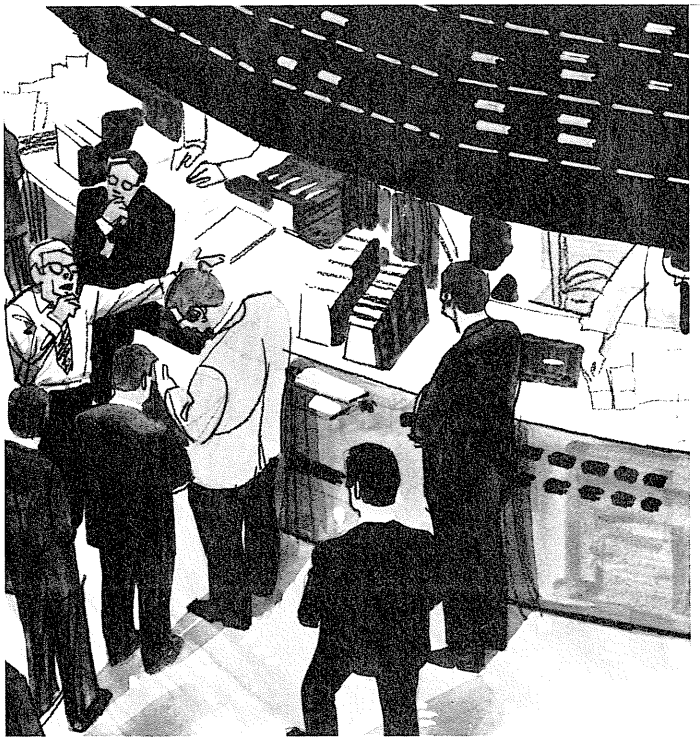
Busy business travelers can catch more than planes at New York's La Guardia airport these days . . . now they can catch the latest stock, bond, and option prices.

In the airport's supper level shopping area there is a Business Information Center that features Dow Jones News/Retrieval. Travelers stop to use this up-to-minute source of reliable business and financial information whenever they have a spare moment. Many of them keep up with their investments by retrieving Dow Jones Quotes and the news that moves the stock market.

In most cases, their interest in quotes is not new; these "stock-watchers" are used to getting their quotes from The Wall Street Journal. They know that the quality they expect from the Journal is also available electronically with News/Retrieval.

Continuously updated price information, with a minimum of a 15 minute delay from the floor of the exchange, is available for common and preferred stocks and warrants of four U.S. exchanges: New York, American, Pacific, and Midwest. Subscribers can access composite and exchange prices as well as individual prices for over 6,000 companies listed on these exchanges or traded in the over-the-counter market.

In addition, users can access corporate bond prices from three exchanges — New York, American, and Pacific. Foreign bonds are available from the New York Stock Exchange.



Options are from four exchanges — the composite of the Chicago Board of Options Exchange, and the American, Pacific, and Philadelphia Stock Exchanges.

For all the above, Dow Jones Current Quotes include previous day's close, opening price, high, low, latest price and volume.

Dow Jones Quotes are also available for the national over-the-counter market (NASDAQ), with bid and ask prices updated six times each day. Volume is updated once each business day at approximately 6 p.m. Eastern Time. Daily updates are also available for mutual funds and U.S. Treasury issues.

Many News/Retrieval customers find historical stock quotes useful for relating stock price trends to news developments on companies, and for other methods of investment analysis. Historical quotes are available for common and preferred stocks and warrants on the four major U.S. stock exchanges — New York, American, Midwest and Pacific — and for the national OTC market. High, low, close, and volume figures are provided. OTC quarterly and monthly summaries provide high bid and low bid, while daily summaries provide bid and ask. Both give volume. Users are charged *only* for the time they are signed on to the system, there are no per-quote charges. Users with screens or printers that accept at least 72 characters per line can save time and money by accessing up to *five* quotes simultaneously. This requires special passwords. Call Dow Jones Customer Service if you wish to change your password to receive multiple quotes.

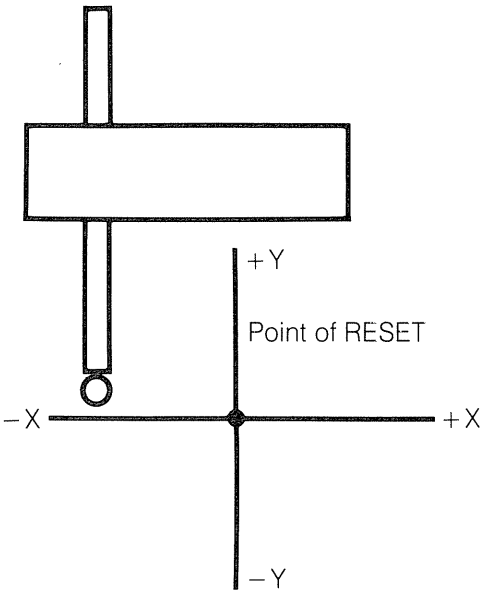
Software Adds Power

Special software packages, such as Videotex, also enable users to turn their micro-computers into "mini stock analysis laboratories." Analyzing market trends with the aid of a personal computer can help the serious investor decide when it is time to buy, sell, or sit on the sidelines.

DOW JONES NEWS/RETRIEVAL®

Graphics Subroutines

The TRS-80 Digitizer is highly mobile, capable of being placed almost anywhere, and at any angle. The digitizer "assumes" that the X axis of the world is perpendicular to the edge of the traverse arm, while the Y axis is parallel to that edge.



Drawing 1

Not only is this not a valid assumption, but it is frequently most inconvenient. You may not always be able to "exactly" position the digitizer on the X-Y axis of the material you are digitizing. What is needed is a set of routines which allow you to set up the digitizer where it is convenient, and then tell the computer where the X and Y axes are. Other things you may wish to do with the numbers the digitizer is giving the computer is to scale them, and/or offset them. The following sample routines can be used to accomplish these tasks.

GETTING DIGITIZER INFORMATION INTO THE COMPUTER

The first thing we have to be able to do is to get information into the computer. The Digitizer manual provides routines for all TRS-80s capable of RS-232 output, plus a routine for Model I TRS-80s using the cassette port. Here is the portion of the Model I/III RS-232 routine which is needed to get the information from the Digitizer and translate it to X-Y coordinates (we ran it on a 48K two drive Model III with RS-232):

```
10Ø ' RADIO SHACK DIGITIZER INPUT ROUTINE
11Ø ' MODEL I/III VERSION FOR RS-232
12Ø ' COPYRIGHT 1982 TANDY CORPORATION
13Ø '
14Ø ' INITIALIZATION PROGRAM
15Ø '
16Ø DEFINT I-N
17Ø '
18Ø ' INITIALIZE RS-232
19Ø '
2ØØ OUT 232, Ø
21Ø OUT 233, 85    '3ØØ BAUD
22Ø OUT 234, 252   '8 BIT, NO PARITY, 2 STOP BITS
23Ø '
3ØØ GOSUB 13ØØØ    'GET DIGITIZER INPUT IF ANY
31Ø GOTO 3ØØ
1ØØØØ ' READ ON CHARACTER FROM RS-232
1ØØ1Ø '
1ØØ2Ø J=INP(234)
     : IF J<127 THEN 1ØØ2Ø
1ØØ3Ø J=INP(235) AND 127
1ØØ4Ø RETURN
1ØØ5Ø '
11ØØØ ' GET AN 8-DIGIT STRING OF COORDINATES
11Ø1Ø '
11Ø2Ø GOSUB 1ØØ2Ø
     : IF J<>88 THEN 11Ø2Ø
11Ø3Ø D$=""
     : FOR I=1 TO 8
     : GOSUB 1ØØ2Ø
     : D$=D$+CHR$(J)
     : NEXT
     : RETURN
11Ø4Ø '
12ØØØ ' CONVERT 4 HEX DIGITS TO DECIMAL
12Ø1Ø '
12Ø2Ø DE=Ø
12Ø3Ø FOR I=Ø TO 3
12Ø4Ø J=ASC(MID$(HX$, I+1, 1)) -48
12Ø5Ø IF J>9 LET J=J-7
12Ø6Ø DE=DE +J * 16[ (3-I)
12Ø7Ø NEXT I
12Ø8Ø IF DE>32767 LET DE=DE-65536
12Ø9Ø DE=DE*.Ø1
121ØØ RETURN
1211Ø '
13ØØØ ' GET (X,Y) COORDINATES AND DISPLAY
131Ø1Ø '
1302Ø GOSUB 11ØØØ
1303Ø HX$=LEFT$(D$,4)
1304Ø GOSUB 12ØØØ
1305Ø X=DE
1306Ø HX$=RIGHT$(D$,4)
1307Ø GOSUB 12ØØØ
1308Ø Y=DE
1309Ø PRINT USING "LOCATION: (###.##, ###.##)";
     X, Y
131ØØ RETURN
```

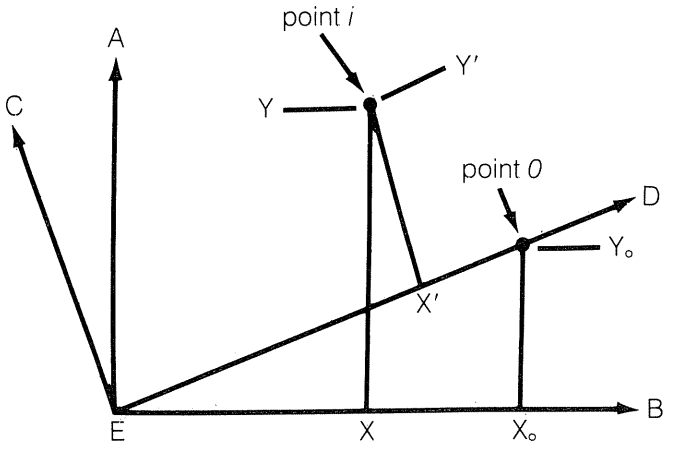
If you run this routine, with the digitizer set up according to the instructions in the manual, the computer will print X and Y values each time you press the Send switch. Notice we did not have to use SETCOM to initialize RS-232, the program did it for us, automatically.

AXIS ROTATION

The idea behind axis rotation is that we can have the computer accept the X-Y values sent from the digitizer, and then mathematically rotate those values to fit some new X-Y coordinate set.

We will assume that the digitizer is set up to input information in the X-Y axis set represented in drawing 2 by AEB,

where E is the origin (point where both X and Y are zero), the line AE represents the Y axis of the digitizer, and the line EB represents the digitizer's X axis. Further, we assume that the actual image being digitized has the same origin (E), but that the line CE represents the image's Y axis, and the line ED represents the image's X axis.



Drawing 2

What we need to do is select some point on the image's X axis (we will use the point marked 0 in the diagram) and relate that point to the digitizer's coordinate system. After all the smoke clears from doing this, we find that we have two formulae, one translates the digitizer's X values into the appropriate image X value, and the other translates the digitizer's Y values into appropriate Y image values. Here are the formulae:

```
Y'=(Y-(X*(YØ/XØ))*(XØ/SQR(XØ*XØ+YØ*YØ))

X'=(X/(XØ/SQR(XØ*XØ+YØ*YØ))+(Y'*(YØ/XØ))
```

If we examine these formulae, we see two common elements:

```
YØ/XØ and XØ/SQR(XØ*XØ+YØ*YØ)
```

Our subroutine to compute the values we need to rotate the axes is:

```
49ØØØ ' ROUTINE TO DETERMINE ROTATION VALUES
49Ø1Ø '
49Ø2Ø CLS
      : PRINT "AXIS ROTATION USING X AXIS"
49Ø3Ø PRINT "PLACE THE DIGITIZER CURSOR ON THE
      COMMON ORIGIN (Ø,Ø) AND PRESS THE RESET
      BUTTON ON THE BACK OF THE DIGITIZER"
49Ø4Ø PRINT "THIS SETS THE DIGITIZER'S ORIGIN TO
      MATCH THE ORIGIN OF YOUR MATERIAL."
49Ø5Ø PRINT "PLACE THE DIGITIZER'S CURSOR ON THE
      X - AXIS OF THE IMAGE TO BE DIGITIZED (MOVE
      THE CURSOR AT LEAST TWO INCHES TO THE RIGHT
      OF THE ORIGIN.)"
49Ø6Ø PRINT "PRESS THE SEND SWITCH."
49Ø7Ø GOSUB 13ØØØ 'GET X,Y COORDINATES
49Ø8Ø XØ=X
      : YØ=Y
49Ø9Ø T=YØ/XØ      'FIRST TRANSLATION ELEMENT
491ØØ C=XØ/SQR(XØ*XØ + YØ*YØ) 'SECOND ELEMENT
4911Ø PRINT "AXIS ROTATION COMPLETE."
4912Ø RETURN
```

To see how this works, make the following additional changes to the original program and add the subroutine:

```
3ØØ GOSUB 49ØØØ
31Ø GOSUB 13ØØØ
32Ø Y=((Y-X*T)*C)
33Ø X=(X/C+Y*T)
34Ø PRINT USING "ROTATED (###.##, ###.##)"; X, Y
35Ø GOTO 31Ø
```

Now when you run the routine, you will be asked to input a single point on the image's X axis. From that point on, any time you press the Send switch, the program will display two sets of X and Y values, first, the values in the Digitizer's X-Y system and then in the image's X-Y system.

To use this rotation feature in your programs, remove lines 13090 and 340. These are the lines which print the X-Y values. Now change line 350 to read:

```
35Ø RETURN
```

The rotated X-Y value is available for whatever purpose you wish after you do a GOSUB 310.

MOVING THE ORIGIN

What happens if the item you are digitizing is larger than the area the traverse arm can cover, and the image's origin is not inside the traverse arm area? Obviously we may have some problems making the X and Y values output by the Digitizer correspond to the X-Y values of the image you are digitizing.

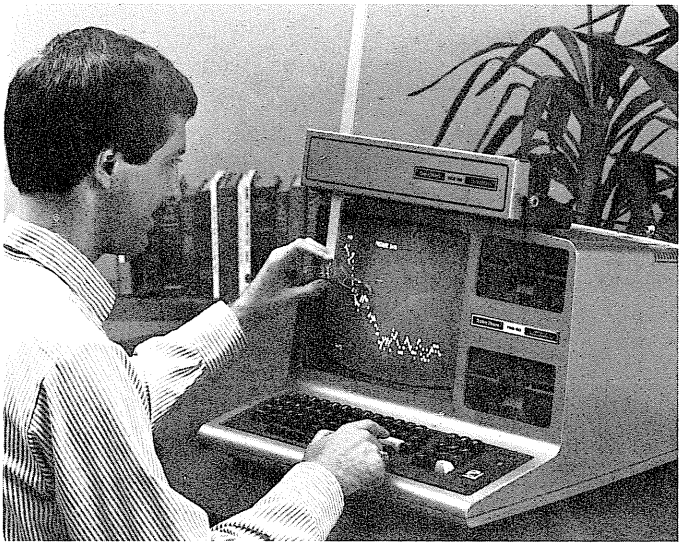
We solve this by setting the digitizer's origin (set by pushing the reset switch on the digitizer) to a point on the image to be digitized that has known X-Y values. Now we tell the program what the known image X and Y values were. The program can now adjust the points output by the digitizer to the image's coordinates. Here is the subroutine to input the image X-Y values:

```
5ØØØØ REM SUBROUTINE TO OFFSET X-Y VALUES
5ØØ1Ø CLS
5ØØ2Ø PRINT "Y-OFFSET EQUALS"; YO; ". ";
5ØØ3Ø INPUT "ENTER NEW VALUE: ";YO
5ØØ4Ø PRINT "X-OFFSET EQUALS"; XO; ". ";
5ØØ5Ø INPUT "ENTER NEW VALUE: ";XO
5ØØ6Ø RETURN
```

Now all we have to do is offset the input points. We do this by adding the offset values to the X and Y values we computed in lines 320 and 330:

```
32Ø Y=((Y-X*T)*C) + YO
33Ø X=(X/C+Y*T) + XO
```

To add the offset routine, simply use:
305 GOSUB 50,000



Three-Dimensional Graphics Package

Charles E. Burton, Ph.D.
1720 South Deframe Court
Denver, CO 80228

This material was converted from an article by F. C. Crow in BYTE Magazine, March and April, 1981. My program has most of the features outlined in that article except hidden surface/line removal. The readers can add that feature if they desire. The software is written in Model I TRSDOS 2.3 Disk BASIC. However, it should be easily converted to run on a cassette system by removing the OPEN and CLOSE statements and by replacing the PRINT #1 and INPUT #1 statements with PRINT #−1 and INPUT #−1 statements. Also, I believe that it should run directly on a Model III.

The software is fairly well structured and easy to follow. It is modularized into small subroutines, with the most used routines at or near the front of the program. That format has a tendency to speed up the execution somewhat since the interpreter starts its search for lines from the beginning of the program, but it does make the program read backwards.

This program will give you a very powerful tool for creating and exploring three dimensional objects.

After you have entered the program, type:

```
RUN <ENTER>
```

You will see the menu screen:

```
COMMANDS:
    1--START OVER
    2--DEFINE OBJECT
    3--READ OBJECT
    4--ENTER EYE POINT
    5--ENTER CENTER OF INTEREST
    6--ENTER WINDOW
    7--DRAW PICTURE
    8--QUIT
COMMAND TO EXECUTE:
```

The first thing we will want to do is to define an object. For your first object, let's draw a cube. A cube has eight corner points or vertices and six faces (each face is a polygon).

Corner Points	X	Y	Z
1	10	10	10
2	10	−10	10
3	−10	−10	10
4	−10	10	10
5	10	10	−10
6	10	−10	−10
7	−10	−10	−10
8	−10	10	−10

Face Vertexes				
Face 1	1	2	3	4
Face 2	1	5	6	2
Face 3	5	8	7	6
Face 4	8	4	3	7
Face 5	8	5	1	4
Face 6	2	6	7	3

Enter a 2 to get the Define Object option. The program will ask you for a file name, (pick one), the number of points in the object (8), and the number of polygons in the object (6). You will then be asked for the coordinates of each point, and then for the number of vertices for each polygon (four for each in our case) and the point numbers in clockwise order for each polygon.

After you have input all of the requested data, the computer will store the data on disk.

The next step is to read the data from the disk since the program does not seem to store all the values during definition. The program will ask for the object file name (the same one you used to defined the object) and a position for the object. Since we want to view the object at the origin, input the points: 0, 0, 0.

Next we will define our "eye point." The eye point is where you are in relation to the object. To start, use an eye point of 0, 0, 40.

We next define the "center of interest." That is which single point in the object or on its surface do we want to look directly at. Since our object is located with the origin at the center, we will define the origin, 0, 0,0 as our center of interest.

The next item on the menu is the window. We will use a window with four sides, and the four vertices will be: 20,20,15; 20, −20,15; −20, −20,15 and −20,20,15.

We are finally ready to have the computer draw our object as seen from our eye point with the window we just defined.

```
1 ' THREE-DIMENSIONAL GRAPHICS PACKAGE
2 '       BY C.E. BURTON
3 ' ADAPTED FROM ARTICLE: F.C. CROW,
    "THREE-DIMENSIONAL
4 ' COMPUTER GRAPHICS, PTS. 1 & 2", BYTE,
    MARCH/APRIL, 1981
5 GOTO 34ØØ            'GENERATE 3-D GRAPHICS
1Ø '*********   MOVE SCREEN CURSOR TO X,Y
2Ø '    XS=SCREEN CURSOR X COORDINATE
3Ø '    YS=SCREEN CURSOR Y COORDINATE
4Ø '    (X,Y)=DESIRED SCREEN CURSOR COORDINATE
5Ø XS=X
    : YS=Y
6Ø RETURN
1ØØ '********   DRAW LINE FROM SCREEN CURSOR
    (XS,YS) TO X,Y
1Ø1 '   XS=SCREEN CURSOR X COORDINATE
1Ø2 '   YS=SCREEN CURSOR Y COORDINATE
1Ø3 '   (X,Y)=DESIRED END OF LINE COORDINATE
11Ø '   DL=MAXIMUM(ABS(X-XS),ABS(Y-YS))
12Ø '   DX=X STEP
13Ø '   DY=Y STEP
14Ø DX=ABS(X-XS)
    : DY=ABS(Y-YS)
15Ø IF DX>DY THEN DL=DX ELSE DL=DY
16Ø IF DL<1 THEN DL=1
17Ø DX=SGN(X-XS)*DX/DL
    : DY=SGN(Y-YS)*DY/DL
18Ø FOR I=1 TO DL
185     IF (XS<Ø) OR (XS>127) OR (YS<Ø) OR
    (YS>47) THEN 21Ø
19Ø             SET(XS+.5, YS+.5)
2ØØ             XS=XS+DX
                : YS=YS+DY
21Ø NEXT I
215 IF XS<Ø OR XS>127 OR YS<Ø OR YS>47 THEN 22Ø
217     SET(XS+.5, YS+.5)
22Ø RETURN
3ØØ '********   3-D DISPLAY TO SCREEN 2-D
31Ø '   P(Ø)=POINT X COORDINATE
32Ø '   P(1)=POINT Y COORDINATE
33Ø '   P(2)=POINT Z COORDINATE
34Ø '   SS(Ø)=X SCREEN SCALE
35Ø '   SS(1)=Y SCREEN SCALE
36Ø '   SC(Ø)=X SCREEN CENTER
37Ø '   SC(1)=Y SCREEN CENTER
38Ø DEFFNP(A, B, C)=B*A/P(2)+C
385 IF P(2)=Ø THEN P(2)=1
39Ø P(Ø)=FNP(P(Ø), SS(Ø), SC(Ø))
4ØØ P(1)=FNP(P(1), SS(1), SC(1))
41Ø RETURN
5ØØ '********   VECTOR INNER PRODUCT
51Ø '   A=POINT 1 X COORDINATE
52Ø '   B=POINT 1 Y COORDINATE
53Ø '   C=POINT 1 Z COORDINATE
54Ø '   D=POINT 2 X COORDINATE
55Ø '   E=POINT 2 Y COORDINATE
56Ø '   F=POINT 2 Z COORDINATE
57Ø DEFFND(A, B, C, D, E, F)=A*D+B*E+C*F
58Ø RETURN
6ØØ '********   FORM IDENTITY MATRIX
61Ø '   MX(I, J)=4 X 4 MATRIX
62Ø FOR I=Ø TO 3
63Ø     FOR J=Ø TO 3
64Ø             IF I=J THEN MX(I, J)=1
                      ELSE MX(I, J)=Ø
65Ø     NEXT J
66Ø NEXT I
67Ø RETURN
7ØØ '********   MATRIX MULTIPLY
71Ø '   M1(I, J)=4 X 4 MATRIX 1
72Ø '   M2(I, J)=4 X 4 MATRIX 2
73Ø '   MX(I, J)=4 X 4 MATRIX MULTIPLY RESULT
74Ø FOR I=Ø TO 3
75Ø     FOR J=Ø TO 3
76Ø             MX(I, J)=Ø
77Ø             FOR K=Ø TO 3
78Ø                   MX(I, J)=MX(I, J)+
                            M2(I, K)*M1(K, J)
79Ø             NEXT K
8ØØ     NEXT J
81Ø NEXT I
82Ø RETURN
83Ø '********   MX(I, J) -> M1(I, J)
835 FOR I=Ø TO 3
84Ø     FOR J=Ø TO 3
845             M1(I, J)=MX(I, J)
85Ø     NEXT J
855 NEXT I
86Ø RETURN
865 '********   MX(I,J) -> M2(I,J)
87Ø FOR I=Ø TO 3
875     FOR J=Ø TO 3
88Ø             M2(I, J)=MX(I, J)
885     NEXT J
89Ø NEXT I
895 RETURN
9ØØ '********   DISPLAY LINE
91Ø '   P1(.)=FROM POINT
92Ø '   P2(.)=TO POINT
93Ø FOR I=Ø TO 2
94Ø     P(I)=P1(I)
95Ø NEXT I
96Ø GOSUB 3ØØ          '3-D DISPLAY TO SCREEN 2-D
97Ø X=P(Ø)
    : Y=P(1)
975 GOSUB 1Ø           'MOVE SCREEN CURSOR TO X,Y
98Ø FOR I=Ø TO 2
99Ø     P(I)=P2(I)
1ØØØ NEXT I
1Ø1Ø GOSUB 3ØØ         '3-D DISPLAY TO SCREEN 2-D
1Ø2Ø X=P(Ø)
    : Y=P(1)
1Ø3Ø GOSUB 1ØØ         'DRAW LINE FROM SCREEN
    CURSOR TO X,Y
1Ø4Ø RETURN
11ØØ '*******   TRANSFORMATION
111Ø '  MX(I,J)=4 X 4 MATRIX
112Ø '  P(Ø)=POINT X COORDINATE
113Ø '  P(1)=POINT Y COORDINATE
114Ø '  P(2)=POINT Z COORDINATE
115Ø '  P1(Ø)=POINT 1 X COORDINATE
116Ø '  P1(1)=POINT 1 Y COORDINATE
117Ø '  P1(2)=POINT 1 Z COORDINATE
118Ø P(Ø)=FND(P1(Ø), P1(1), P1(2), MX(Ø, Ø),
        MX(Ø, 1), MX(Ø, 2))+MX(Ø, 3)
119Ø P(1)=FND(P1(Ø), P1(1), P1(2), MX(1, Ø),
        MX(1, 1), MX(1, 2))+MX(1, 3)
12ØØ P(2)=FND(P1(Ø), P1(1), P1(2), MX(2, Ø),
        MX(2, 1), MX(2, 2))+MX(2, 3)
121Ø RETURN
13ØØ '*******   DRAW POLYGON
131Ø '  NP=NUMBER OF POINTS IN POLYGON
132Ø '  LJ=LAST J
133Ø '  PY(I,J)=POLYGON VERTEX I WITH COORD. J
134Ø LJ=NP-1
135Ø FOR J=Ø TO NP-1
136Ø    FOR K=Ø TO 2
137Ø            P1(K)=PY(LJ, K)
                : P2(K)=PY(J, K)
138Ø    NEXT K
139Ø    GOSUB 9ØØ
14ØØ    LJ=J
141Ø NEXT J
142Ø RETURN
15ØØ '*******   GET EYE SPACE
151Ø '  EP(.)=EYE POINT COORDINATES
152Ø '  ES(I,J)=EYE SPACE MATRIX (4 X 4)
153Ø '  CI(.)=CENTER OF INTEREST COORDINATES
1535 DEFFNH(A, B)=SQR(A*A+B*B)
154Ø GOSUB 6ØØ         'IDENTITY MATRIX
1545 MX(Ø, Ø)=-1
    : MX(1, 1)=-1
    : MX(2, 2)=-1
155Ø MX(Ø, 3)=EP(Ø)
    : MX(1, 3)=EP(1)
    : MX(2, 3)=EP(2)
1555                   'EYEPOINT TRANSLATION
1556 GOSUB 83Ø         'MX(I,J) -> M1(I,J)
156Ø FOR I=Ø TO 2
```

```
157Ø    P1(I)=CI(I)
158Ø NEXT I
159Ø GOSUB 11ØØ          'TRANSFORMATION: CTR. OF
                                INTR. TRANSL.
16ØØ HY=FNH(P(Ø), P(1))
161Ø IF HY=Ø THEN 17ØØ
162Ø    GOSUB 6ØØ        'IDENTITY MATRIX
163Ø    CA=P(1)/HY
        : SA=P(Ø)/HY
164Ø    MX(Ø, Ø)=CA
        : MX(1, Ø)=SA
165Ø    MX(Ø, 1)=-SA
        : MX(1, 1)=CA
166Ø    GOSUB 865        'MX(I,J) -> M2(I,J)
167Ø    GOSUB 7ØØ        'MATRIX MULTIPLY
168Ø    GOSUB 83Ø        'MX(I,J) -> M1(I,J)
169Ø    GOSUB 11ØØ       'TRANSFORMATION: ROTATE
                           ABOUT Z-AXIS
17ØØ HY=FNH(P(1), P(2))
171Ø IF HY=Ø THEN 18ØØ
172Ø    GOSUB 6ØØ        'IDENTITY MATRIX
173Ø    CA=P(1)/HY
        : SA=-P(2)/HY
174Ø    MX(1, 1)=CA
        : MX(2, 1)=SA
175Ø    MX(1, 2)=-SA
        : MX(2, 2)=CA
176Ø    GOSUB 865        'MX(I,J) -> M2(I,J)
177Ø    GOSUB 7ØØ        'MATRIX MULTIPLY
178Ø    GOSUB 83Ø        'MX(I,J) -> M1(I,J)
179Ø    GOSUB 11ØØ       'TRANSFORMATION: ROTATE
                           ABOUT X-AXIS
18ØØ GOSUB 6ØØ          'IDENTITY MATRIX
181Ø MX(1, 1)=Ø
     : MX(2, 2)=Ø
182Ø MX(1, 2)=1
     : MX(2, 1)=1
183Ø GOSUB 865           'MX(I,J) -> M2(I,J)
184Ø GOSUB 7ØØ           'MATRIX MULTIPLY
185Ø FOR I=Ø TO 3
186Ø    FOR J=Ø TO 3
187Ø            ES(I, J)=MX(I, J)
188Ø    NEXT J
189Ø NEXT I
19ØØ RETURN
2ØØØ '*******   GET SCREEN SCALE
2Ø1Ø '  MM(Ø)=MAX X
2Ø2Ø '  MM(1)=MAX Y
2Ø3Ø '  MN(Ø)=MIN X
2Ø4Ø '  MN(1)=MIN Y
2Ø5Ø '  WS=WINDOW SIDES
2Ø6Ø '  W(I,J)=WINDOW VERTEX I WITH COORDINATES
2Ø7Ø '  SS(.)=SCREEN SCALE COORDINATES
2Ø8Ø '  DA=DOTS ACROSS
2Ø9Ø '  DD=DOTS DOWN
21ØØ MM(Ø)=Ø
     : MM(1)=Ø
     : MN(Ø)=Ø
     : MN(1)=Ø
211Ø FOR I=Ø TO WS-1
212Ø    IF W(I, Ø)/W(I, 2)>MM(Ø)
           THEN MM(Ø)=W(I, Ø)/W(I, 2)
213Ø    IF W(I, Ø)/W(I, 2)<MN(Ø)
           THEN MN(Ø)=W(I, Ø)/W(I, 2)
214Ø    IF W(I, 1)/W(I, 2)>MM(1)
           THEN MM(1)=W(I, 1)/W(I, 2)
215Ø    IF W(I, 1)/W(I, 2)<MN(1)
           THEN MN(1)=W(I, 1)/W(I, 2)
216Ø MM(Ø)=MM(Ø)-MN(Ø)
     : MM(1)=MM(1)-MN(1)
217Ø IF MM(1)>3*MM(Ø)/4 THEN SS(2)=4*MM(1)/3 ELSE
     SS(2)=MM(Ø)
218Ø SS(Ø)=DA/SS(2)
     : SS(1)=(4*DD/3)/SS(2)
219Ø RETURN
22ØØ '*******   INITIALIZE
221Ø '  TP=TOTAL POINTS
222Ø '  TY=TOTAL POLYGONS
223Ø '  TV=TOTAL VERTICES
224Ø '  EP(.)=EYEPOINT COORDINATES
225Ø '  CI(.)=CENTER OF INTEREST COORDINATES
226Ø '  WS=WINDOW SIZE
227Ø '  W(I,J)=WINDOW VERTEX I WITH COORDINATES
2273 DA=128
     : DD=48
2275 TP=Ø
     : TY=Ø
     : TV=Ø
228Ø EP(Ø)=-5
     : EP(1)=-5
     : EP(2)=3
229Ø CI(Ø)=Ø
     : CI(1)=Ø
     : CI(2)=Ø
23ØØ WS=4
231Ø W(Ø, Ø)=-4
     : W(Ø, 1)=-3
     : W(Ø, 2)=16
232Ø W(1, Ø)=-4
     : W(1, 1)=3
     : W(1, 2)=16
233Ø W(2, Ø)=4
     : W(2, 1)=3
     : W(2, 2)=16
234Ø W(3, Ø)=4
     : W(3, 1)=-3
     : W(3, 2)=16
235Ø GOSUB 2ØØØ          'GET SCREEN SCALE
236Ø SC(Ø)=DA/2
     : SC(1)=DD/2
2365 GOSUB 5ØØ           'VECTOR DOT PRODUCT
     INITIALIZE
237Ø RETURN
24ØØ '*******   MAKE PICTURE
241Ø '  ES(.,.)=EYE SPACE TRANSFORMATION
242Ø '  MX(.,.)=4 X 4 MATRIX
243Ø '  PD(I,Ø)=STARTING VERTEX NUMBER FOR
                  POLYGON I
244Ø '  PD(I,1)=TOTAL NUMBER OF VERTICES FOR
                  POLYGON I
245Ø '  V(I)=POINTER TO COORDINATES FOR VERTEX I
246Ø '  PI(I,J)=POINT COORDINATES FOR VERTEX I
247Ø '  PY(I,J)=TEMPORARY POLYGON VERTEX
                  COORDINATES
248Ø CLS
2482 PRINT "GETTING EYE SPACE"
2485 GOSUB 15ØØ          'GET EYE SPACE
249Ø FOR I=Ø TO 3
25ØØ    FOR J=Ø TO 3
251Ø            MX(I, J)=ES(I, J)
252Ø    NEXT J
253Ø NEXT I
254Ø FOR L=Ø TO TY-1
255Ø    LØ=PD(L, Ø)
        : NP=PD(L, 1)
256Ø    FOR M=Ø TO NP-1
257Ø            MØ=V(LØ+M)-1
                :IF MØ<Ø THEN MØ=Ø
258Ø            FOR N=Ø TO 2
259Ø                    P1(N)=PT(MØ, N)
26ØØ            NEXT N
2605 PRINT "TRANSFORMING"
261Ø            GOSUB 11ØØ       'TRANSFORMATION:
                                  VERTEX TO
                                  EYESPACE
262Ø            FOR N=Ø TO 2
263Ø                    PY(M, N)=P(N)
264Ø            NEXT N
265Ø    NEXT M
2655 CLS
266Ø    GOSUB 13ØØ               'DRAW POLYGON
267Ø NEXT L
2675 PRINT@Ø, "DONE";
268Ø IF INKEY$="" THEN 268Ø
2685 CLS
269Ø RETURN
27ØØ '*******   READ OBJECT
271Ø '  PT(I,J)=POINT COORDINATES FOR VERTEX I
272Ø '  V(I)=POINTER TO COORDINATES FOR VERTEX I
273Ø '  PD(I,Ø)=STARTING VERTEX NUMBER FOR
                  POLYGON I
```

```
274Ø '  PD(I,1)=TOTAL NUMBER OF VERTICES FOR
                  POLYGON I
275Ø '  TP=TOTAL POINTS
276Ø '  TY=TOTAL POLYGONS
277Ø '  TV=TOTAL VERTICES
2771 '  PO=POINTS PER OBJECT
2772 '  YO=POLYGONS PER OBJECT
2773 '  J,X,Y,Z=POINT #, X-COORD., Y-COORD.,
            Z-COORD
2774 '  PP=POINTS PER POLYGON
2775 '  V=POINT # WHICH DEFINES POLYGON VERTEX
278Ø LINE INPUT "OBJECT FILE NAME: "; F$
279Ø PRINT "POSITION FOR OBJECT "; F$;
28ØØ INPUT " X, Y, Z: "; XP, YP, ZP
282Ø OPEN "I", 1, F$
284Ø INPUT #1, PO, YO
285Ø FOR I=Ø TO PO-1
286Ø    INPUT #1, J, X, Y, Z
287Ø    IØ=TP+I
288Ø    PT(IØ, Ø)=X+XP
289Ø    PT(IØ, 1)=Y+YP
29ØØ    PT(IØ, 2)=Z+ZP
291Ø NEXT I
292Ø FOR I=Ø TO YO-1
293Ø    INPUT #1, PP
294Ø    FOR J=Ø TO PP-1
295Ø            INPUT #1, V
296Ø            V(TV+J)=V+TP
297Ø    NEXT J
298Ø    IØ=TY+I
299Ø    PD(IØ, Ø)=TV
3ØØØ    PD(IØ, 1)=PP
3Ø1Ø    TV=TV+PP
3Ø2Ø NEXT I
3Ø3Ø TP=TP+PO
3Ø4Ø TY=TY+YO
3Ø5Ø CLOSE 1
3Ø6Ø RETURN
31ØØ '*******   DEFINE OBJECT
311Ø '  PO=POINTS PER OBJECT
312Ø '  YO=POLYGONS PER OBJECT
313Ø '  J,X,Y,Z=POINT #, X-COORD., Y-COORD.,
            Z-COORD.
314Ø '  PP=POINTS PER POLYGON
315Ø '  V=POINT # WHICH DEFINES POLYGON VERTEX
316Ø LINE INPUT "OBJECT FILE NAME: "; F$
317Ø INPUT "POINTS PER OBJECT: "; PO
318Ø INPUT "POLYGONS PER OBJECT: "; YO
3185 IF PO<3 OR YO<1 THEN 317Ø
319Ø OPEN "O", 1, F$
32ØØ PRINT #1, PO; YO
321Ø FOR I=1 TO PO
322Ø    PRINT "COORDINATES FOR POINT #"; I;
323Ø    INPUT " X, Y, Z: "; X, Y, Z
324Ø    PRINT #1, I; X; Y; Z
325Ø NEXT I
326Ø FOR I=1 TO YO
327Ø    PRINT "NUMBER OF VERTICES FOR POLYGON";I;
328Ø    INPUT ": "; PP
3285    IF PP<3 OR PP>8 THEN 327Ø
329Ø    PRINT #1, PP;
33ØØ    PRINT "ENTER POINT #'S FOR VERTICES IN
            CLOCKWISE DIRECTION"
331Ø    FOR J=1 TO PP
332Ø            PRINT "POINT FOR VERTEX #"; J;
333Ø            INPUT ": "; V
3335            IF V<1 OR V>PO THEN 332Ø
334Ø            PRINT #1, V;
335Ø    NEXT J
336Ø    PRINT #1,
337Ø NEXT I
3375 CLOSE 1
338Ø RETURN
34ØØ '*******   GENERATE 3-D GRAPHICS
341Ø '  EP(.)=EYEPOINT COORDINATES
342Ø '  CI(.)=CENTER OF INTEREST COORDINATES
343Ø '  TP=TOTAL POINTS
344Ø '  TY=TOTAL POLYGONS
345Ø '  TV=TOTAL VERTICES
346Ø '  WS=WINDOW SIZE
347Ø '  W(I,J)=WINDOW VERTEX I COORDINATES
348Ø DIM CI(2), EP(2), ES(3, 3), M1(3, 3),
        M2(3, 3), MN(1), MM(1)
349Ø DIM MX(3, 3), P1(2), P2(2), PD(199, 2),
        P(2), PT(199, 2), PY(7, 2)
35ØØ DIM SS(2), V(799), W(7, 2)
3505 GOSUB 22ØØ          'INITIALIZE
351Ø PRINT "COMMANDS:"
352Ø PRINT "    1--START OVER"
353Ø PRINT "    2--DEFINE OBJECT"
354Ø PRINT "    3--READ OBJECT"
355Ø PRINT "    4--ENTER EYE POINT"
356Ø PRINT "    5--ENTER CENTER OF INTEREST"
357Ø PRINT "    6--ENTER WINDOW"
358Ø PRINT "    7--DRAW PICTURE"
359Ø PRINT "    8--QUIT"
36ØØ INPUT "COMMAND TO EXECUTE: "; I
361Ø IF I<Ø OR I>8 THEN 351Ø
362Ø ON I GOTO 363Ø, 366Ø, 369Ø, 372Ø, 375Ø,
            378Ø, 387Ø, 39ØØ
363Ø '*******   START OVER
364Ø TP=Ø
     : TY=Ø
     : TV=Ø
365Ø GOTO 351Ø
366Ø '*******   DEFINE OBJECT
367Ø GOSUB 31ØØ
368Ø GOTO 351Ø
369Ø '*******   READ OBJECT
37ØØ GOSUB 27ØØ
371Ø GOTO 351Ø
372Ø '*******   ENTER EYE POINT
373Ø INPUT "ENTER EYE POINT X, Y, Z: "; EP(Ø),
        EP(1), EP(2)
374Ø GOTO 351Ø
375Ø '*******   ENTER CENTER OF INTEREST
376Ø INPUT "ENTER CENTER OF INTEREST X, Y, Z: ";
        CI(Ø), CI(1), CI(2)
377Ø GOTO 351Ø
378Ø '*******   ENTER WINDOW
379Ø INPUT "NUMBER OF WINDOW SIDES: "; WS
38ØØ IF WS<3 OR WS>8 THEN 379Ø
3805 PRINT "ENTER WINDOW VERTICES IN CLOCKWISE
        ORDER"
381Ø FOR I=Ø TO WS-1
382Ø    PRINT "WINDOW POINT #"; I;
383Ø    INPUT " X, Y, Z: "; W(I, Ø), W(I, 1),
            W(I, 2)
384Ø NEXT I
385Ø GOSUB 2ØØØ          'GET SCREEN SCALE
386Ø GOTO 351Ø
387Ø '*******   DRAW PICTURE
388Ø GOSUB 24ØØ
389Ø GOTO 351Ø
39ØØ '*******   QUIT
391Ø END
```

Model I/III Bugs, Errors and Fixes

Changes to BASIC programs

There are general procedures that need to be followed when any corrections are made.

1. Make a backup of the tape or disk that contains the program to be corrected. Changes should be made on the backup copy.
2. Load the program to be changed by typing CLOAD"filename" (tape) or LOAD"filename" (disk) where filename is the name of the program to be modified.

(Continued on Page 33)

Education Graphics

Graphics Creation In Radio Shack's Author I

by Dan P. Gibbs

In the AUTHOR I system by Radio Shack, a special graphics mode is available to a lesson developer while working on TEXT pages.

Both graphics and text may be displayed on both non-interactive frames (a regular TEXT page) and on interactive ones (a TEXT page containing question text).

The SHIFT key, combined with the @ key is used to enter the graphics mode in AUTHOR I. When used together in this way, these two keys actually function as a two-way "toggle" switch on the TRS-80 Model I (toggling between text and graphics), or as a three-way toggle on the TRS-80 Model III (toggling between text, regular graphics, and Model III's special character set.)

Once you enter the graphics mode, the cursor becomes one-fourth the size of the usual cursor-square. This smaller symbol is helpful in visually reminding the user that the graphics mode is in use. This is especially beneficial since, during the creation of a display, the user will probably be changing from text entry to graphics entry and back again several times.

Graphics are created by "drawing" or "erasing" with the small cursor-square. This is done by pressing appropriate keys on the TRS-80 keyboard to move the cursor-square back and forth on the video screen, drawing or erasing as needed to create the desired graphics image.

On all TRS-80 Model III's and on the TRS-80 Model I's with the numeric keypad, the graphics cursor is moved by pressing one of the numeric keys 1-4 or 6-9. The numbers on these keys have no significance during graphics mode entry: they are used only as references for moving the cursor in the direction desired. Keys 8 and 2 move the cursor up (north) and down (south), respectively. Keys 6 and 4 move the cursor right (east) and left (west), respectively. Keys 1, 3, 7, and 9 move cursor diagonally outward from the center (key 5, which is inactive — does not move the cursor at all).

On TRS-80 Model I's without the separate numeric keypad, the arrow keys on the standard keyboard are pressed to move the cursor in the directions indicated. Diagonals are drawn by holding down the SHIFT key while pressing one of the four arrow keys. (The TRS-80 actually causes a clockwise rotation of about 45 degrees from the direction of the arrow when used with the SHIFT key.)

Within AUTHOR I's graphics mode, there are three "sub"-modes: the DRAWING sub-mode, the ERASING sub-mode, and the NOT DRAWING sub-mode.

The three keys on the bottom row of the numeric keypad (0 , . , or ENTER) are pressed to change from one graphics sub-mode to another. Text messages appearing in the lower area of the screen signify which of the three sub-modes is in effect:

1. DRAWING sub-mode — Graphics are drawn in this sub-mode by leaving a "trail" of white where ever the cursor is moved.
2. ERASING sub-mode — Graphics are erased in this sub-mode. As the cursor is moved in this sub-mode, a "trail" of black will be left, erasing any characters or white graphics elements which may be present.
3. NOT DRAWING sub-mode — This sub-mode is most helpful when the user simply wants to move the cursor around on the video display without affecting or altering the current graphics creation.

Movement of the graphics cursor across text material (letters, numbers, etc.) will erase the text — regardless of which sub-mode the user is in!

A "library" of special graphics characters is also available in the TRS-80 Model III version of AUTHOR I. These special characters may be added to figures created during the graphics mode, or to textual, non-graphics displays.

Entry into the special characters mode is accomplished with the second "toggle" of the SHIFT and @ keys on the Model III. The first toggle places the user into the graphics mode; the second toggle places him in the special characters mode.

When the special characters option is accessed, the text cursor-square freezes at its current screen position, and the library of available characters appears in two rows (A: and B:) in the lower portion of the TRS-80 Model III screen. The user locates the special character desired (it will be in one of the two displayed rows of special characters) and then presses the letter of the row in which the character appears. The row of characters not being referenced disappears when A: or B: is selected.

The standard keyboard character which must be typed in order to display the special character at the current cursor position is then located in the row of characters above the remaining row A: or B:.

During the toggle routine which allows entry into the graphics mode or special characters mode, text appropriate to the current mode appears on the lower line of the screen display, e.g., an indication of the "sub"-mode during the graphics mode, or the rows of special characters in the special characters mode. If the user is changing back and forth from normal text entry to graphics mode or special characters mode, these text messages on the bottom of the display will flash on and off quickly, indicating that movement between modes is occurring.

SOME SUGGESTED USES OF GRAPHICS

Too often, in working with the graphics utility or editor, the temptation of creating "graphics for graphics' sake" exists during the courseware development process. The use of graphics should not be haphazard or random, but carefully planned. The following are three primary reasons for using graphics in the courseware being developed:

1. As an entertaining attention-getter — such as Computer Charlie and Marvin the Martian in the upcoming TRS-80 Philadelphia C.A.R.D. Program Lessons. This use of graphics is valid but should not become "the tail that wags the dog." Cute-sy graphics may very well turn off all but the youngest students.
2. As a means to separate text-heavy sequences into smaller, more accessible portions — often, a summary chart or table utilizing graphics or special characters can aid in the condensation of text material and yet still achieve the instructional intent of the sequence.
3. As instructionally-valid illustrations — Remember the old adage, "A picture is worth a thousand words." The graphics figure must be clearly discerned by the student if its addition to the lesson is to have any merit.

The use of graphics must always be carefully weighed, within the parameters of instructional intent and of memory size.

The graphics capability of AUTHOR I is just one of a myriad of features which make this new authoring system so attractive to classroom teachers.

Computer Literacy

Jerry Doyle and Ed Rolenc of Fort Madison, Iowa, have found a way to bring computer literacy to thirteen Iowa school districts. Third through eighth grade students in about fifty school buildings are learning programming on the TRS-80 Color Computer. Radio Shack's Part I: Introduction to BASIC, revised by Rolenc for use with the Color Computer, provides the major course material.

Doyle, who is a Science and Math Consultant for a tax-supported regional service center that provides educational media for the thirteen districts, says that the idea to teach programming to elementary school and junior high students grew out of requests from elementary school teachers. Three years ago computers were introduced at the high school level, and the elementary school people then wanted to introduce their students to computers.

Students in grades three through five are using 4K TRS-80 Color Computers to learn the material from Part I of the Radio Shack Computer Education Series. In most cases, elementary teachers teach programming to their own classes after some inservice training. Most of these teachers haven't had any previous computer experience. In other cases, the administration has assigned a particular teacher to teach all of the programming courses in a particular school.

Classes use the overhead transparencies and workbooks from Part I and follow the Part I lesson structure closely. Doyle says that the original Part I is 90% intact as the program is used in the districts. The major exception is Lesson Seven. Since Lesson Seven deals with graphics, it was completely rewritten for use with the Color Computer. (The Part I curriculum is designed for use with the Model III or Model I.) Doyle called Ed Rolenc, a local computer expert, "the brains behind the revision."

Although configurations vary throughout the thirteen districts, Doyle says that the typical arrangement is to have one or two Color Computers in an elementary classroom. The teacher presents the lesson on the overhead projector, then students take turns doing their assignments at the computer.

Doyle reports a favorable student reaction to learning programming. Elementary students enjoy the hands-on work with the Color Computer, which Doyle calls "a real friendly machine — very easy to work with." Most students, he says, feel comfortable with the computer, and about one third of the third through fifth-graders are becoming "very good" at elementary programming. While Doyle says that "some of the brighter students are doing lots of interesting things" with programming, he emphasizes that all groups, not just accelerated students, are involved in the classes. Programming on the Color Computer seems to offer something for everyone. While some students prefer the more mathematical programming exercises, Doyle says, others are fascinated by the color graphics they have learned to produce.

Doyle's future plans for the third through fifth grade programming curriculum include scaling down some of the Part I exercises. He mentioned that some of the programs in the original Part I (for example, the Miles Per Gallon computation), involve mathematics that most students don't learn until after the third grade. Nevertheless, Part I was an important factor in the decision to use TRS-80 computers, Doyle says, because it is a complete, ready-made program that lends itself well to general education.

Students in grades six through eight of the thirteen districts are currently using 16K TRS-80 Color Computers to learn both the Part I material that the earlier grades are covering, and some of the material from Radio Shack's Part II: BASIC Programming. The service center is currently revising Part II for larger-scale use with the upper grades. Their revised version will include music lessons for the Color Computer plus more color graphics. Generating sound with the Color Computer is currently being taught in the districts as an "Eleventh Lesson" of Part I.

Doyle speaks enthusiastically about computers in the classroom and about the TRS-80 Color Computer in particular. He describes the Color Computer as being, in his opinion, "the best product Radio Shack has." With the educational discount offered through the National Bid Department, Doyle's service center was able to put 160 computers in classrooms "for a very modest cost." Besides the user-friendliness and reasonable cost of the Color Computer, and besides the availability of the Computer Education Series, Doyle mentioned "the availability of service" as an important factor in their choice of TRS-80s.

Doyle emphasizes the importance of computer literacy as a priority of the thirteen school districts, and says that not only the students, but also the teachers, are developing this literacy. Although some teachers have been

reluctant to devote class time to computer literacy, Doyle is seeing an increasing willingness to do so. He reports that, overall, the districts have had success with the computer literacy program and are strongly supporting the idea that the computer is "not a novelty item," but an important part of education.

In This System, The Computer Future Is Now

This article is reprinted with permission from The American School Board Journal, March. Copyright 1982, the National School Boards Association. All rights reserved.

John Bristol has seen the future of schools — and it is computers. While some of you cringe at the cost of microcomputers, Superintendent Bristol has gone out and bought 220 of them. While you fret about reliable software, his teachers write their own instructional programs. And while you might have a few students queuing up for a crack at the machines, his school system has eight labs in which every student sits at his own computer.

In the two campuses of Lyons Township High School District in LaGrange and Western Springs, Illinois, virtually all 3,800 students and 275 teachers are computer literate. That is because Bristol, who became superintendent in 1980, has convinced the school board that computer literacy is a top priority and that nothing should foil the schools' efforts to ensure that all students are familiar with microcomputers by the time they are graduated.

To those skeptics who question his quarter-million-dollar expenditure — and to those who dismiss his accomplishments because his schools can afford to plunk down that kind of money — Bristol answers: "It is not a question of 'Can I afford to do this?' It is a question of 'Can I afford not to do this?'"

Bristol's goal of creating universal computer literacy among his staff and students might be the most ambitious in North America. Other school systems have accomplished parts of what Bristol has instituted, but it would be difficult to find another school system with so comprehensive a plan. So far, the program has gained momentum because Bristol has approached it in a logical, systematic manner. Before he decided what type of microcomputers the schools ought to buy, for example, he and his staff declared a few objectives: First, when a class uses microcomputers, every student should work with his own machine. Second, the machines should have "network" capability, so that a teacher using his own computer can lead a class and have evervone working on the same program; at the same time, each program should be adaptable to students who work and learn at different rates. Third, the entire staff — not only interested or expert faculty — should be able to use the machines, and every department in the school should become involved in the computer education program. If students were to be able to use microcomputers, Bristol reasoned, teachers would have to learn to do so first.

Because Bristol saw staff development as key to the success of his program, one of the first things he did upon assuming the superintendency was to speak at departmental meetings about the importance of computers. "Some teachers got excited; others were threatened by the prospect of change," explains Julie McGee, an English teacher who is one of the prime movers in the Lyons Township computer education program. "Some said, 'Why rush in just because it is new?' and some were just flat-out machine phobic."

But when Bristol asked the staff to attend a series of four, two-hour inservice courses, only two of the 275 teachers decided not to attend — a notable achievement, considering that 40 percent of the teachers have at least 20 years of teaching experience and might have had little chance to become computer literate.

Bristol hired Norman Bell, a Michigan State University professor, to conduct the computer literacy workshops last May for teachers. Eventually, eight teachers who assisted Bell volunteered to help teach sections of the literacy course to their colleagues. As McGee recalls, the most frustrating aspect of the teaching training course was that many teachers did not know how to type; they were unfamiliar with a typewriter keyboard. (With 50 teachers in a class, McGee soon learned that typists should be separated from nontypists.)

After the teachers learned how to use the computer and discovered what it could accomplish, they were asked to submit ideas about how they thought the microcomputer could be used effectively in their own classrooms. Then, during the summer, four teachers were able to combine their knowledge of the school's curriculum with their newfound interest in programming; these four worked with groups of other teachers who thought they had workable ideas for microcomputer programs. To help the four teachers with actual programming, Bristol hired a few computer whiz kids from a nearby college. Working together, the teachers and college students composed instructional software for a variety of subject areas.

According to Ed Cermak, a mathematics and data processing teacher, "In some cases, the software we produce is better than that of the publishing company, because we know the subject matter, the educational techniques, and the programming. Those three elements are the most important in producing educational software. There simply are not that many people who know all that."

Curriculum Director Estella Gahala agrees: "In most schools, experts take possession and shut out others. We wanted everyone involved so all of us had a stake in it."

Clearly, Bristol is proud of the extent to which his staff has caught the computer bug. "In the absence of good computer software on today's market, we opted for what needed to be done, not what was available. Our software fills the need of our district, whereas the software on the

market might limit the objectives of our school system. Some people build a program around available software, we built our computer literacy program around our own needs."

Last summer the staff produced 60 different programs, many of which encompass different levels of instruction. For example, in one algebra class observed last January, approximately 25 students, each seated before a microcomputer, worked on a series of a dozen or so problems. When the program was written last summer, staff members devised 24 different levels of difficulty for each series of problems. While some students in the algebra class worked on Level Five problems, for example, others in the same room worked on Level Fifteen problems. The teacher, naturally, has a record of what level each student completes as well as a record of what problem each student can or cannot answer.

Julie McGee, who also is the yearbook adviser, says the microcomputers will be used for purposes besides instruction. Instead of typing out yearbook copy on paper, for example, students will enter the copy into the microcomputer, proofread it on the video screen, and send the disk that contains all that information to a typesetting house. McGee estimates the process will save $6 per page in yearbook production costs.

Teachers in other subject areas have thought of other uses for the microcomputers: The football coach uses one to help scout future opponents; physical education teachers devise individual exercise programs for students based on the results of physical education tests; foreign language teachers drill their students on verb endings. There is a computer program on alcohol abuse, one on the U.S. Constitution, a few in genetics, and a slew in mathematics, spelling, and vocabularly drills. To take advantage of the various programs for their students, teachers simply reserve a computer lab a week in advance, tell the class to meet there, insert the program into the microcomputer, and cause the program to appear on students' screen.

Now that the foundation for computer education is in place, Bristol likes to point out, teachers will continue to expand the school's library of computer software. In fact, he says, he has been approached about marketing the software his teachers have produced.

Running students through increasing numbers of computer software programs, however, is not the extent of computer education in Lyons Township. The school also offers advanced instruction in data processing, for example. But more important, all Lyons Township students take a four-day computer literacy course devised by McGee, Cermak, and another mathematics teacher, George Robinson.

Last January 29, while students were given the day off, teachers attended a workshop that laid out exactly how to teach the four-day literacy course. Then in early February, on four consecutive days during the second period of the school day, each teacher — no matter what his or her specialty — taught computer literacy. "It is important to realize," says McGee, "that you do not have to be able to program to be able to teach with a computer. It is like a text book or any other teaching device."

On the first day of the course, students watched a film that discussed the history and impact of computers on society. On the second day, students went to a computer lab to find out how the machines work, to learn some of the terminology, and to practice on the keyboard. On the third day, they saw a film on programming and ran a program that Robinson and Cermak wrote, which simulates what happens inside the computers as the program is running. (Students were able to observe how various statements and commands affect the program.) Finally, students saw two more films that examined the sociological implications of computers and that showed computers being used by automobile mechanics, hospital personnel, and persons in other work places. The students also took a short, machine-scored quiz and were able to review most of the material in their staff-written workbooks. In future school years, only freshmen and transfer students will take the literacy course.

John Bristol can give you 101 reasons computer literacy is a major education issue in the 1980s. Because he is convinced that students will not be literate unless they are computer literate, he has no patience for school officials who want to wait until the technology is perfected or until publishing companies market better software. He is especially disturbed by school officials who plead poverty. He will scrawl numbers on a chalk board to illustrate how Lyons Township's expenditure for computers amounts to only $7.50 per student per year for the next five years; he says that is not too much to pay for computer literacy.

If you are planning to adopt a new American history textbook next year, he advises school officials, put it off a year: If you have 1,000 students, you would have to buy 1,250 books; at $20 per book, that is $25,000—enough for one computer lab. And once the lab is in place, you can begin organizing teachers to develop their own instructional software programs.

The reason for having teachers from every department write as much software as they possibly can is not to glorify technology, of course; the intent is to make the best possible instructional use of equipment that might well play an integral part in the lives of everyone who is graduated from high school from now on. "There is nothing about Lyons Township that says we can do this and others cannot," says Bristol.

In five years, he says, he hopes the vast majority of courses in his schools will use the microcomputer in one way or another. At the rate he is going, he will be years ahead of everyone else.

Concentric Circles

Stephen Havens
67 North Sable Street #15
Keeseville, NY 12944

Here is a short program I wrote on my TRS-80 Model III that will draw circles, one inside the next. This is one of the first programs I wrote on my TRS-80. By inputting the size of the first circle, the resolution of the circle, and number of spaces between each succeeding circle will produce different patterns.

CIRCLES

```
5 REM * STEPHEN HAVENS 1981*
1Ø CLS
2Ø INPUT"ENTER THE SIZE OF THE CIRCLE (1 TO 23)";Y
25 IF Y<1 OR Y>23 THEN 2Ø
3Ø INPUT "ENTER ANY NUMBER GREATER THAN 1 FOR THE
   RESOLUTION";X
35 IF X<1 THEN 3Ø
4Ø PRINT"ENTER NUMBER Ø TO 23 FOR NUMBER OF SPACES"
42 PRINT"BETWEEN EACH CIRCLE";
   : INPUT Z
45 IF Z<Ø OR Z>23 THEN4Ø
5Ø CLS
   : PI=3.14159
6Ø FOR A=Ø TO 2*PI STEP PI/X
7Ø I=63+Y*(2.6667*SINCA)
   : N=23+Y*COS(A)
8Ø SET(I,N)
9Ø NEXT A
   : Y=Y-1-Z
   : IF Y<Ø THEN 16Ø ELSE9Ø
1ØØ FOR W=1 IO 2ØØØ
   : NEXT W
   : PRINT"END"
   : END
```

Model I/III Bugs (From page 28)

3. Make the line changes indicated in the fix. For existing line numbers, edit or retype the line to match the one in the fix. Enter new lines.
4. Save the corrected program (the one now in memory). Type CSAVE"filename" ⟨**ENTER**⟩ (tape) or SAVE"filename" ⟨**ENTER**⟩ (disk) where filename is the name of the program that has been modified.
5. Now make a backup of the corrected tape or diskette.

PATCHES

PATCHes are entered from TRSDOS READY and are used to make corrections to files stored on the disk.

1. Before making a PATCH, back up the diskette that requires modification and make the PATCHes to the backup copy of the diskette.
2. Apply PATCHes according to the information given in your TRSDOS manual.

Model III TRSDOS (26-312)

When the second PATCH below was published in the February 1982 Newsletter it contained an error. The following PATCHes (which include the correction) will allow Model III DEBUG under TRSDOS 1.3 to inspect and/or change addresses in the copyrighted code areas and the ROM of the Model III.

```
PATCH *5:Ø (ADD=4EDF,FIND=38E6,CHG=ØØØØ)
PATCH *5:Ø (ADD=4FØ4,FIND=DØ,CHG=C9)
PATCH *5:Ø (ADD=5Ø6E,FIND=38E3,CHG=ØØØØ)
```

Inventory Control (26-1553)

In Version 3.1 for Model I/III, totals on the Listing by Vendor are the same as the totals in System Status.

To correct the problem make the following changes to the program "ICS."

```
179Ø GOSUB254Ø:GOSUB261Ø:RETURN

292Ø GOSUB1795:GOSUB372Ø:LPRINTIS$;TAB(12)ID$;TAB
     (33);:LPRINTUSINGLØ$;IQ%;:LPRINTTAB(39);
     :IFCC=ØTHENLPRINTUSINGL1$;IC;ELSE
     PRINT"    ****";

1795 GC#=GC#+IQ%*IC:GP#=GP#+IQ%*IP:GR#=GR#+IO%
     *IC:GI=GI+1:RETURN
```

Accounts Payable (26-1554)

The Accounts Payable program allows the user to delete any invoice at any time, even when the invoice is Selected, but does not decrement the # of invoices selected.

For version 3.0 on both the Model I and III change the following line in the program INVOICES.

```
219 K=KK:IS=IS+(I(K)<Ø):I(K)=Ø:I=I-1:IT=IT-1
    :ID=ID+1:PL=987:W1$="DELETED":GOSUB1Ø9:GOTO71
```

For versions prior to 3.0 on both the Model I and III change the following line in the program INVOICES.

```
795 K=KK:IS=IS+(I(K)<Ø):I(K)=Ø:I=I-1:IT=IT-1
    :ID=ID+1:PL=987:W1$="DELETED":GOSUB43Ø
    :GOTO13Ø
```

Accounts Receivable (26-1555)

Following are two sets of corrections for AR.

1 — While setting up Model I/III Accounts Receivable Version 3.0 under the three-drive option, you will encounter the error BAD FILE MODE in line 620.

Make the following changes to the program SETUP.

```
22Ø FL=1:GOSUB28Ø:R$=IN$:IFCF<>ØTHEN22ØELSEIFR$<>
    "Y"ANDR$<>"N"THENPRINTCHR$(8);:GOTO22Ø

88Ø PD=2:PC=5ØØ:PT=25ØØ:IFQ$="M"THENONERROR
   GOTO895:KILLPT$:PT$=LEFT$(PT$,LEN(PT$)-1)
   +"2":CLS:PRINT@458,"INSERT DATA DISK IN DRIVE
    2 AND PRESS <ENTER>":ELSEGOTO89Ø
```

Add the following line.

```
225 IFR$="N"THENGOSUB56Ø:IFQ$="I"THENZX=Ø:GOTO8Ø
```

2 — In version 3.0 and earlier of AR, when a correction entry is made to a payment, it replaces the previous payment on the statement.

Make the following changes to version 3.0 of the program ARS.

```
2Ø2Ø PR#=PR#-VE#:CB#=CB#-VE#:CV#=CV#-VE#
     :G#(VJ)=G#(VJ)-VE#:G#(2)=G#(2)-VE#
```

****NOTE: On versions prior to 3.0, this is line 1650.

Special Character Mode

Randy C. Butts
619 Barr Drive
Lancaster, OH 43130

As you may know, inputting a PRINT CHR$(21) on the Model III turns on the special character mode. However, this has limitations when using it in a BASIC program. There is no way of knowing if the special character mode is on or off. Address 16420 is the poke address of the CHR$(21) mode. Poking a 1 will turn it on, a 0 will turn it off. By peeking at this address, it is possible for a BASIC program to know if special characters are on or off and to let the program take appropriate action.

Also, address 16912 contains the CHR$(22) function. Poking a 32 will switch to alternate (kana) characters and poking a 40 will switch it back to special characters.

I hope this will help other users of the Model III to be able to use some of its new features.

High Resolution Business Graphics for Models II and 16

With the addition of Radio Shack's new high resolution graphics option, the Model II (and 16) acquired a graphics ability which may make even the Color Computer owners jealous. The graphics screen contains 640 points horizontally and 240 points vertically, for a total of 153,600 individually addressable picture elements or "pixels." (The Color Computer has a maximum of 256 horizontal and 192 vertical elements for a maximum of 49,152 pixels.)

The graphics option adds a new board to your Model II or 16 which works in conjunction with the existing video display board. This new board contains all the hardware needed to generate high resolution graphics, plus an additional 32K of memory. The additional 32K of memory on the graphics board means that you do not lose any of your current memory in order to use high resolution!

The high resolution option includes a Graphics BASIC which gives you immediate access to the full graphics capabilities of your new board. For those who want to use graphics with other languages, a set of callable machine-language subroutines is included. Let's look at the basic commands which are available in Graphic BASIC:

To start with, a few definitions:

Origin — The origin is the pixel with horizontal and vertical coordinates of 0, 0. For the Model II/16 high resolution system, the origin is in the upper left corner of the display.

Parameter — a parameter is a value used by a command to control some function. Two typical parameters are X and Y, which are used to determine where a command will begin its action.

X — a value from 0 to 639, representing the horizontal position of a pixel on the display. A value of zero places the defined point in the leftmost vertical column of the display, while a value of 639 represents the rightmost column. Two X axis (horizontal) pixels represent about the same distance as one Y (vertical) pixel.

Y — a value from 0 to 239, representing the vertical position of a pixel on the display. A value of zero places the defined point in the top horizontal row of the display, while a value of 239 places the point in the bottom row of the display.

CLS — In Graphics BASIC the familiar CLS command acquires some extensions.

CLS 0 — Clears only regular text, leaves graphics.

CLS 1 — Clears only graphics, leaves regular text.

CLS 2 — Clears both graphics and text.

PSET and PRESET — PSET and PRESET are nearly identical functions. Their purpose is to turn a single point on the video on or off. PSET defaults to white or point on, while PRESET defaults to black or point off.

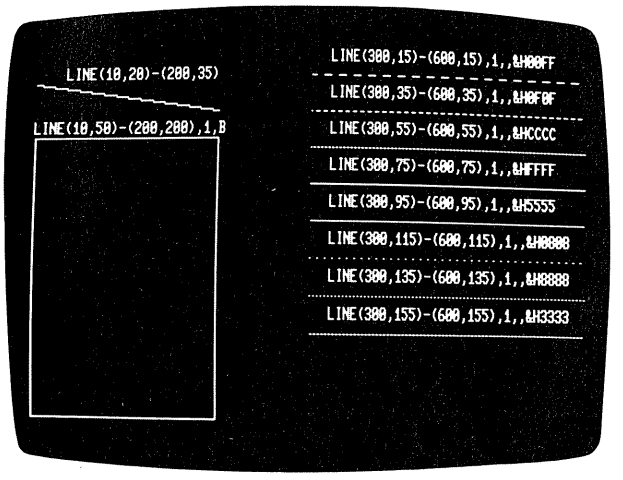
PSET and PRESET have two required parameters, X and Y; and one optional parameter C. C determines what color (zero (0) = black, 1 = white) the point specified by X and Y will be set to.

POINT — POINT is used to find out if the point specified by X and Y is on (color = 1) or off (color = 0)

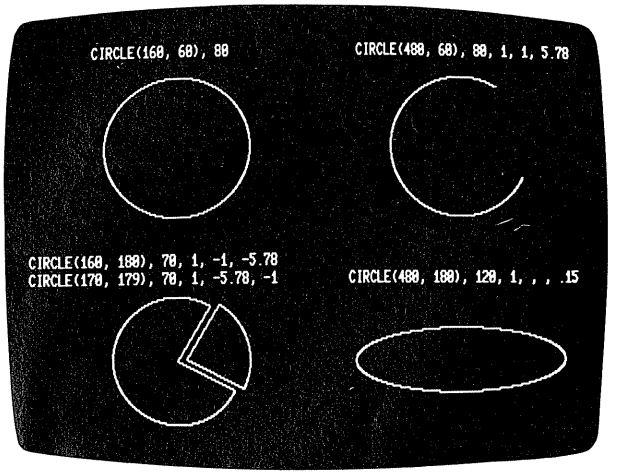
LINE — The LINE command lets you draw a line between two points. The starting point of the line can be specified, or can be left out. If the start point is left out, the most recently used endpoint is used. The ending point must be specified.

Optional parameters for LINE are C (to determine the color of the line), B (to draw a box rather than a line), BF (to draw a box and then fill or paint the box), and Style.

Style is used to set the pattern of the line to be drawn. If Style is not specified, the line will be solid white line. The permissible range of values for Style is from −32768 to 32767. This gives you 65,535 possible patterns for use in drawing lines. This wide variety of line patterns helps to overcome any graphic limitations which might exist because we are limited to black and white as our only colors.



CIRCLE — The CIRCLE command is used to draw circles, arcs, ellipses, and pie charts. The CIRCLE command has three required parameters (val-



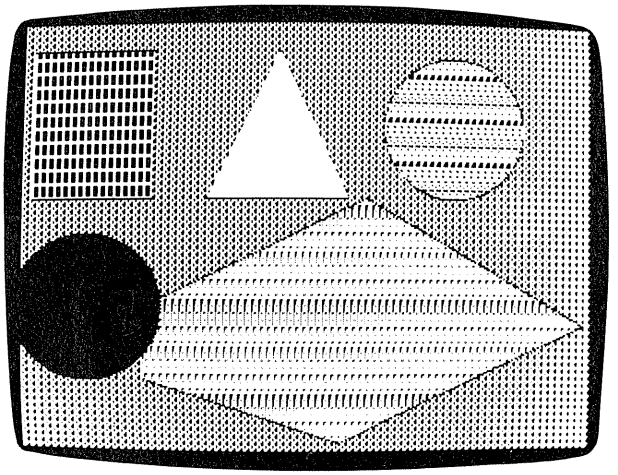
ues). The required parameters are X, Y and R. X and Y define the center of the circle, while R defines the radius of the circle in X-axis pixels. The four optional parameters are:

C (0 — draws circle in black, 1 — (default) draws circle in white.)

S (starting point where drawing is to begin. If the value for S is not zero, and is preceded by a minus sign, the computer will draw a line from the center of the circle to the starting point as well as drawing the defined portion of the circle) This feature (combined with the same feature for E) makes pie charts very easy to create.

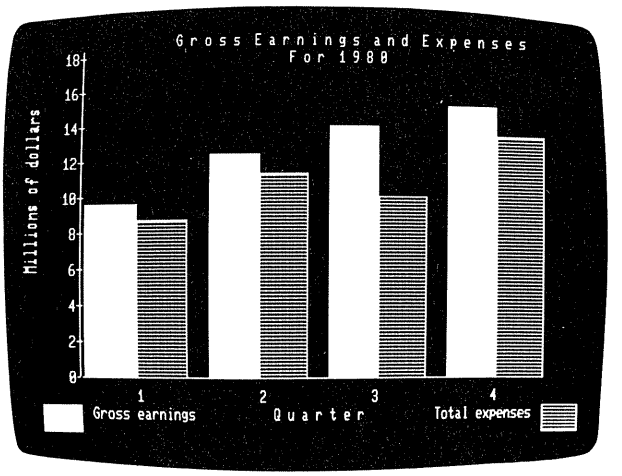
E (ending point where drawing stops)

HW (defines the height to width ratio used to draw the "circle." A value less than .5 elongates the circle along the X-axis, while a value greater than .5 elongates the circle along the Y-axis.)



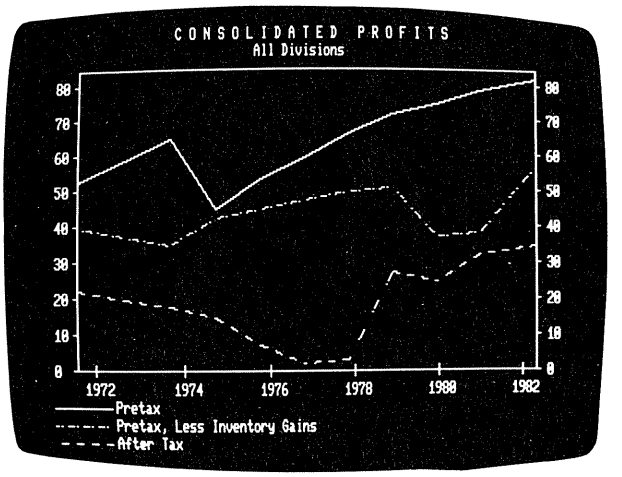
PAINT — The paint command is used to "paint" sections of the screen. The two required parameters are X and Y which are the coordinates of the point where painting begins.

The paint command will paint with an optional Style. The default style is to paint solid white. Style can be specified as a number (0 or 1) or as an alphanumeric string of 1 to 64 characters. If

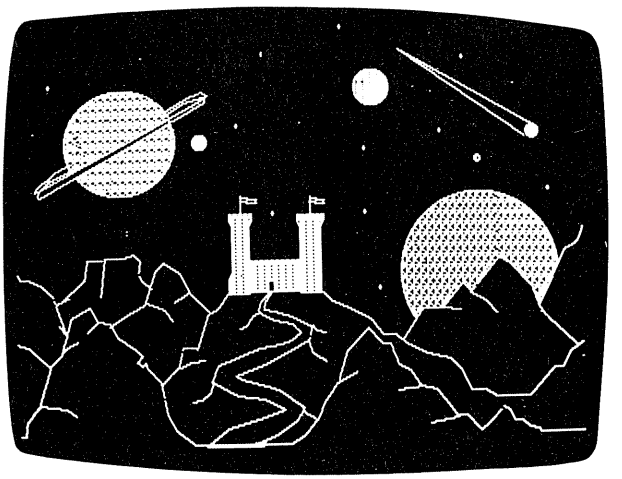


an alphanumeric string is used, the characters are used to create a pattern for painting. The characters themselves are not printed, they simply determine the Style to be used.

Painting continues until a border (default color of border is white) is reached.



In addition to the graphic functions we have covered this month, the Model II High Resolution Graphics option also supports GET, PUT, SCREEN, VIEW and VIEWPORT. We will have information on these functions in a future issue.



Model II Bugs, Errors and Fixes

Changes to BASIC programs

There are general procedures that need to be followed when any corrections are made.

1. Make a backup of the disk that contains the program to be corrected. Changes should be made on the backup copy.
2. Load the program to be changed by typing LOAD"filename" where filename is the name of the program to be modified.
3. Make the line changes indicated in the fix. For existing line numbers, edit or retype the line to match the one in the fix. New lines should be entered.
4. Save the corrected program (the one now in memory). Type SAVE"filename" ⟨**ENTER**⟩ where filename is the name of the program that has been modified.
5. Now make a backup of the corrected diskette.

PATCHES

PATCHes are entered from TRSDOS READY and are used to make corrections to files stored on the disk.

1. Before making a PATCH, back up the diskette that requires modification and make the PATCHes to the backup copy of the diskette.
2. Apply PATCHes according to the information given in your TRSDOS manual.

General Ledger (26-4501)

VERSION 1.1

In this version of General Ledger there are two potential problems that exist. First, there is no way to change the month, and secondly, unless the General Ledger's year was closed out in the 12th month the System Status will not reset to one.

In order to change the month that the system is running in, do the following.

1. Go into Accounts Maintenance
2. Press ⟨**BREAK**⟩
3. Type: FM=n (where n is desired month #)
4. Type: CONT ⟨**ENTER**⟩
5. Press the ⟨**F1**⟩ key
6. Check System Status to make sure that the month is correct.

If the General Ledger's year is closed out and it was not the 12th month, the System Status does not reset to one. This problem is corrected by making the following change to the Income program.

```
149Ø FM=Ø
```

VERSION 2.0

In this version the printer will list accounts properly until the end of the page where it should then advance paper for top of forms and continue printing accounts on the second page. The program does not do this; therefore, a new code line is required.

The problem is corrected by making the following correction to the GImaint program.

```
611Ø IFPG>ØTHENLPRINTFF$;
```

Inventory Management (26-4502)

Following are two sets of instructions to allow the user to make specific changes to Inventory Management Version 1.1.

1—In some cases, you might need to change the current period, beginning date of the period, or the beginning date of the year. The only way to do this is by starting a new inventory system, or you can run this program.

1. At TRSDOS READY type in BASIC -F:1
2. At Ready type in the following program.

```
5 CLS:PRINT"ENTER THE FOLLOWING INFORMATION"
1Ø INPUT"CURRENT PERIOD SYSTEM SHOULD BE IN";PD
2Ø INPUT"BEGINNING DATE FOR PERIOD";BP$
3Ø INPUT"BEGINNING DATE FOR YEAR"; BY$
35 OPEN "R",1,"CONTROL/DAT'",96
4Ø FIELD 1, 1Ø AS DU$,2 AS PD$, 6 AS DM$, 9
   AS PB$, 9 AS YB$
5Ø GET 1,2
6Ø LSET PD$=MKI$(PD):LSET PB$=BP$:LSET YB$=BY$
7Ø PUT 1,2
8Ø CLOSE
```

3. At Ready type RUN and answer the appropriate questions.

NOTE: All days must be entered in the following way—three characters for the month, two integers for the day, and four integers for the year with no spaces between them.

EXAMPLES: JAN011980
MAY211980
OCT031980

4. At Ready type SYSTEM"IMS"
5. When the program loads, check the period and date to make sure it is correct.

2—When IMS is run on the original master disk so that you are left without a clean master, a problem arises if you decide that you want to uninitialize the system or change the parameters (in other words you want to start over clean).

The following must be done to clear out the system information. This should only be done in cases when there is not a clean master available.

1. At "TRSDOS Ready" type BASIC -F:3
2. When BASIC has loaded, enter the following program.

```
1Ø CLEAR 1ØØØ
2Ø KILL"SALEDATA/DAT":KILL"CONTROL/DAT"
3Ø OPEN"R",1,"CONTROL/DAT",96
4Ø FIELD 1, 96 AS X$: LSET
   X$=STRING$(96,32):PUT1,1
5Ø LSET X$=STRING$(96,Ø)
6Ø FOR F=2 TO 5:PUT1,F:NEXT F:CLOSE
7Ø SYSTEM"IMS"
```

3. Type RUN and press ⟨**ENTER**⟩.
4. Type SYSTEM to get to TRSDOS READY and make a backup.

Payroll (26-4503)

The following includes information regarding three different sets of problems with Payroll.

1 — In the December 1981 issue of Microcomputer News we informed you of a number of changes that needed to be made to Payroll version 1.1 in order to accommodate the Disability Withholding requirements for California, New Jersey, and Hawaii. There are corrections

that need to be made to those changes. Contact a Radio Shack Computer Center or Computer Customer Services to obtain the changes.

2 — Standard W2 forms require that the last name be printed first. The TRS W2 prints the first name first, then the middle name, and then the last.

Add the following lines to the program named W2.

```
132Ø IFINSTR(NA$,",")<2THEN136Ø
133Ø NA$=MID$(NA$,1,INSTR(NA$+"
     "   ")-1):L=INSTR(NA$,",")
134Ø LA$=MID$(NA$,1,L-1):FR$=MID$(NA$,L+1)+" "
135Ø IFLEN(FR$)>2ANDASC(FR$)=32THENFR$=MID$
     (FR$,2):GOTO135ØELSENA$=FR$+LA$
136Ø REM
```

3 — Payroll will not print out city name in the local city name block (block 22) of the W2 form. This is needed for any city withholding taxes.

Make the following change to the program named W2.

```
1735 LPRINTTAB(36);USINGFI$;D#(3);:LPRINTTAB(48)
     ;USINGFI$;RE#(3);:LPRINTTAB(62);S$:LPRINT:
     LPRINTTAB(36);USINGFI$;D#(4);:LPRINTTAB(48)
     ;USINGFI$;RE#(4);:LPRINTTAB(62)MID$(CT$,1,5)
```

Accounts Payable (26-4505 Version 1.0/2.0)

When using the Print Checks option, the user is not able to disable the printing of check numbers on the checks.

Make the following corrections to the program APCHECKS/BAS.

To eliminate the check number from printing on the stub.

```
241Ø LPRINTL1$:LPRINT;NA$:LPRINTL1$:LPRINTL1$
     :RETURN
```

To eliminate the check number from printing on the check.

```
2515 LPRINTL1$:LPRINTL1$:LPRINTL1$:LPRINTL1$
     :LPRINTL1$
```

SCRIPSIT 2.0 (26-4531)

Following are four corrections to SCRIPSIT 2.0.

1 — Changing disk default values for line spacing in Window MODE and Format line settings is not reflected when creating a document to have the new default values in the L/S setting. Only when L/S shows correct values will document print as specified with the new default values.

The following PATCHes will correct this problem.

```
PATCH SCRIPSIT/SYS R=15 B=186 F=4947 C=63E9
PATCH SCRIPSIT/SYS R=11 B=14Ø
      F=141414141414141414 C=3A228132FØ7AC34947
PATCH STARTUP A=E4ØE F=B1B1AFB2B4AF
                     C=B1B2AFBØB4AF
```

2 — When doing a global Search and Replace under SCRIPSIT, if you specify a beginning page number greater than the highest page number in your document, SCRIPSIT will create the requested page and then proceed to fill up the document with empty pages.

Apply the following PATCHes to correct the problem.

```
PATCH SCRIPSIT A=D6A3 F=ØØØØØØØØØØØØØØØØØØØØ
                      C=CDBECC3E95CØC312C5
PATCH SCRIPSIT A=D6AC F=ØØØØØØØØØØØØØØØØØØ
                      C=CDFBCCC5E3CDFBCC
PATCH SCRIPSIT A=D6B4 F=ØØØØØØØØØØØØØØØØØØ
                      C=E3C1B7ED42DØ3A6B
PATCH SCRIPSIT A=D6BC F=ØØØØØØØØ C=84ØF3FC9
PATCH SCRIPSIT/SYS R=71 B=2Ø3 F=12C5C294
                              C=A3D6C298
PATCH SCRIPSIT/SYS R=72 B=89 F=B7ED4228
                             C=CDACD63Ø
```

3 — If the first paragraph on a page had a line spacing of 2 and was locked, doing a Define Paragraph (ESC U D P) caused the system to lock up.

Apply the following PATCHes to correct the problem.

```
PATCH SCRIPSIT A=D69A F=ØØØØØØØØØØØØØØØØØØØØ
                      C=2BCB7ECØ2BF5C3ØØE2
PATCH SCRIPSIT/SYS R=42 B=1Ø F=2B2BF5 C=C39AD6
```

After the above PATCHes for corrections 2 and 3 have been applied, apply the following PATCH to change the date in the SCRIPSIT initialization screen (The date should read 12/04/1981 before the above PATCHes are applied. If it does not, contact your Radio Shack Computer Center or Store for all the preceding PATCHes which must be applied before these PATCHes.):

```
PATCH STARTUP A=E4ØE F=B1B2AFBØB4AFB1B9B8B1
                     C=BØB1AFBØB8AFB1B9B8B2
```

When this PATCH has been made, the SCRIPSIT initialization screen will show the date 01/08/1982 at the bottom.

4 — If you attempt to create a document on an almost full SCRIPSIT diskette, no check is made to be sure that there is enough free space to create the document and its first page. This results in "garbage" being displayed as the first page. To fix this apply the following PATCHes:

```
PATCH SCRIPSIT A=D686 F=ØØØØØØØØØØØØØØØØ
                      C=2AØA811128ØØ3E96
PATCH SCRIPSIT A=D68E F=ØØØØØØØØØØØØØØØ
                      C=B7ED52D8116D4A
PATCH SCRIPSIT A=D695 F=ØØØØØØØØØØ
                      C=DDE5E1AFC9
PATCH SCRIPSIT/SYS R=15 B=87 F=116D4ADDE5E1
                             C=CD86D6C2A93D
```

These PATCHes prevent SCRIPSIT from creating a document on any diskette that has less than 40 sectors free.

After applying the above PATCHes, apply the following to change the date in the initialization screen (it should show up as 01/08/1982 before you apply these PATCHes. If it doesn't, obtain and apply all preceding PATCHes first).

```
PATCH STARTUP A=E4ØE F=BØB1AFBØB8
                     C=BØB2AFB2B3
```

TRSDOS (26-4910)

Following are five sets of PATCHes for Model II TRSDOS. The first two corrections are for TRSDOS versions 2.0 and 2.0a while the corrections in 3, 4, and 5 are for 2.0a only.

1 — In some cases, when using the PATCH utility to modify a byte, an erroneous 'String Not Found — Abort' would be returned even when a LIST of the file confirms the presence of the byte to be PATCHed at the address specified.

The following PATCHes will correct the problem described above.

```
PATCH PATCH A=3263 F=B79932A5323AA432321933CD
                   C=9132A5323AA432321933B7C4
PATCH PATCH A=3194 F=DA C=CA
PATCH PATCH A=32Ø8 F=DA C=CA
PATCH PATCH A=3221 F=DA C=CA
PATCH PATCH A=3361 F=Ø6ØØB7ED42461D15DA7233
                   C=AF47ED4246151428Ø8151D
```

2 — For some ranges of values, the "RAM Directory" Supervisor call (#53) returned an erroneous value for the number of records in the file. This was caused by an invalid offset in one of the operands in the calculation.

The following PATCHes will correct the code involved in the calculation.

```
PATCH SYSTEM/SYS R=59 B=1Ø3 F=B52811 C=CD9926
PATCH SYSTEM/SYS R=59 B=12Ø F=CF5DEB C=CD9F26
PATCH SYSTEM/SYS R=6Ø B=154
                 F=E5E5E5E5E5E5E5E5E5E5E5
                 C=B5CA7A252BC9CF5D13EBC9
```

3 — The BACKUP utility contains a bug which will cause it to report a TRSDOS ERROR 39 for any file that uses all 16 extents. The following PATCHes will circumvent the bug and allow BACKUP to operate properly.

At TRSDOS READY, apply the following PATCHes.

```
PATCH BACKUP A=28C9 F=21Ø22CCDC632 C=ØØØØØØØØØØØØ
PATCH BACKUP A=2CØ2 F=1D4F6E6C792Ø474F4F44
                    C=22B22A112ØØØ197E3263
PATCH BACKUP A=2CØC F=2Ø66696C65732Ø63616E
                    C=2B22612B36FFC32E352A
PATCH BACKUP A=2C16 F=2Ø62652Ø636F7Ø696564
                    C=612B3A632B7721FFFFC9
PATCH BACKUP A=352B F=22B22A C=C3Ø22C
PATCH BACKUP A=35F4 F=21FFFF C=C3152C
```

4 — In some unattended modes of operation, it is desirable to have HOST start up without any operator prompt for either enabling or disabling the remote ⟨**BREAK**⟩ key. Below are some optional PATCHes for TRSDOS 2.0a which, when applied, will allow the HOST function to start-up without either displaying the question "Disable remote ⟨**BREAK**⟩ key (Y/N)" or requiring the operator's reply.

When the first PATCH is applied, the remote ⟨**BREAK**⟩ key is automatically enabled at start up; if the second PATCH is applied instead, HOST starts with the remote ⟨**BREAK**⟩ key disabled.

To Enable the remote ⟨**BREAK**⟩ key:

```
PATCH SYSTEM64 R=25 B=22 F=21D9F5ØE C=Ø6ØØ1818
(for 64K HOST)

PATCH SYSTEM32 R=25 B=22 F=21D975ØE C=Ø6ØØ1818
(for 32K HOST)
```

To Disable the remote ⟨**BREAK**⟩ key:

```
PATCH SYSTEM64 R=25 B=22 F=21D9F5ØE C=Ø6Ø11818
(for 64K HOST)

PATCH SYSTEM32 R=25 B=22 F=21D975ØE C=Ø6Ø11818
(for 32K HOST)
```

5 — The communications module COMSUB uses serial channel A per its definition. Some users, however, would like to be able to use COMSUB to communicate with a device (like the multi-pen plotter, for example) attached to serial channel B.

In order to get COMSUB to use serial channel B, the following optional PATCHes should be applied to COMSUB

(Optional PATCHes for 64K version of COMSUB)

```
PATCH COMSUB64 A=EF8B F=3E6Ø C=3E62
PATCH COMSUB64 A=EF9A F=3E61 C=3E63
```

TRSDOS-HD 4.0 (26-4150)

The serial communications baud rate = 4800 is not working properly.

When 4800 BAUD operation was specified using either the SETCOM command or the RS232 Initialization Supervisor Call (55), the actual BAUD rate was being set to 9600 BAUD.

The following PATCH is for the TRSDOS-HD 4.0 only and will correct the problem described above.

```
PATCH SYSTEM/SYS R=68 B=139 F=ØD C=1A
```

Graphic Height Correction

R. A. Esquivel
2123 Fairview
Houston, TX 77019

Since March, 1981, I have been the proud owner of your excellent 16K TRS-80 Color Computer (CC), which I have used for a number of applications with great satisfaction and enthusiasm.

And now, after sampling some of the wonderful color programs in various publications, I would like to contribute my little grain of sand!

For the CIRCLE command, I have not been content with the resultant top-to-bottom elongated ellipse that I obtain, instead of a true circle. I have found this to be true not only on my computer, but also at the local TRS Computer Center as well as in published material (For example, see "Going Ahead With Extended Color Basic," page C-4 and C-7 following page 84. For another example, see the cover sheet on the December, 1981 issue of the "TRS-80 Microcomputer News").

I have been able to correct the above problem by using an "hw" value of 0.9 in the command "CIRCLE(x,y),r,c,hw". Specifically, for the beautiful program "A Christmas Wreath and Song," by Gerald Plueard, page 35 of your December 1981 issue of the TRS-80 Microcomputer News, the necessary correction is simply to change line 680 from:

```
68Ø LINE-(128-LA-SA, 96-LB-SB),PSET
```

to:

```
68Ø LINE-(128-LA-SA,(96-LB-SB)*.9),PSET
```

After various tries at graphics with my CC, I have found that a similar problem exists for true squares, 45 degree lines, and such. Alas, a glitch in CC ROM perverts the ratio of horizontal to vertical coordinates to a value of about 0.9 (See, for example, "Going Ahead With Extended Color Basic," pages C-2 and C-5 following page 84).

In conclusion, any time a true horizontal-to-vertical ratio of 1 is desired, the graphics must be corrected. Basically, this can be done by reducing (multiplying) all verticals in a program to 0.9 of the desired screen value.

Notes on Previous Newsletters

JUNE 1981

PCLEAR

Kevin Davidson
Route 10, Box 388
Easley, SC 29640

First a description of the problem: When a PCLEAR is issued, the BASIC program is relocated in memory. The bug is that BASIC continues to run the "old" copy of the

program rather than the relocated one. The old copy may be incomplete (overwritten by part of the new copy) or later damaged by graphics such as PCLS.

The solution has two essential parts: 1) placing BASIC solution statements in a location, the "old" copy of which will be intact after the PCLEAR and 2) getting BASIC back on track in executing the relocated copy of the program.

The answer to the first problem requires solution statements to be placed *both* at the beginning and at the end of the program. Depending on whether more or fewer pages are being PCLEARed, one or the other is executed. If more pages are PCLEARed the old end of the program may be overwritten and the solution must be executed at the program's start; conversely, if fewer pages are PCLEARed, the solution must be executed from the end of the program.

Which is the case? One must determine how many graphic pages are currently in use to distinguish between the two cases. The address of the beginning of the BASIC program (which immediately follows the end of graphics memory) is found in locations 25 and 26 of memory. Hence, the number of graphics pages in use is given by the formula:

```
XP=(256*PEEK(25)+PEEK(26)-1537)/1536
```

The second part of the solution is to get BASIC to begin executing the new relocated copy of the program. This is simple. After the PCLEAR, BASIC correctly stores the address of the "new" copy in locations 25 and 26. When executing *any* backwards GOTO, BASIC uses locations 25 and 26 to begin its forward scan (now in new copy!) looking for the desired statement number.

NOTE: PCLEAR cannot be used in the middle of a BASIC program because variables are not relocated. The manual clearly says that PCLEARs must be first in the program (only after CLEAR) for this reason.

What follows is a few lines of code that will allow any PCLEAR (given proper PMODE and available memory) to work. Lines 10-40 must precede any other program statements except REM and CLEAR (and maybe PMODE). Lines 60000-60010 must be the last statements in the program. The values of P and XP may not be valid on line 40 or following.

```
1Ø P=    '# PAGES TO PCLEAR
2Ø XP=(256*PEEK(25)+PEEK(26)-1537)/1536
3Ø IF P=XP THEN 4Ø ELSE IF P<XP THEN 6ØØ1Ø ELSE
     PCLEAR P:GOTO 1Ø
4Ø ' CONTINUE WITH YOUR PROGRAM
.
.
.
6ØØØØ END
6ØØ1Ø PCLEAR P:GOTO 4Ø
```

FEBRUARY 1982

BOUNCING BOX

John Lockhart
4116 Bridgehampton Drive
St. Charles, MO 63301

In the February issue there is a program called "Bouncing Box" by W. Tudor Apmadoc on page 44. In Line 190 he used POKE 65495,0. I know this is to speed up the computer, however, he did not use a POKE 65494,0 to slow it down at the end. If you do not slow it down at the end of running the program you will not be able to put the program to tape in a way that it can be read in later. Also, no other programs will be able to be read in from tape unless you type POKE 65494,0 or turn off the computer. I have been told that if the Color Computer has a Floppy Disk it will not run the fast mode at all.

I modified the program to work better in 9 steps:

STEP 1— Line 190 I delete.

STEP 2— I changed line 10 to 10 CLS: PRINT " TO SLOW THE COMPUTER DOWN TYPE ' S ' WHILE IT IS RUNNING": PRINT " TO MAKE THE PROGRAM RUN AS FAST AS THE COMPUTER WILL LET IT, TYPE ' F ' WHILE IT IS RUNNING."

STEP 3— Add line 11 PRINT " TO STOP (OR END) THE PROGRAM TYPE ' E ' : PRINT " HIT ANY KEY TO CONTINUE."

STEP 4— Add line 12 A$ = INKEY$: IF A$ = "" THEN 12

STEP 5— Add line 285 A$ = INKEY$ <> "" THEN GOSUB 890

STEP 6— Add line 890 IF A$ = " F " THEN 910 ELSE IF A$ = "S" THEN 900 ELSE IF A$ = " E " THEN 1000 ELSE RETURN.

STEP 7— Add line 900 POKE 65494,0: RETURN

STEP 8— Add line 910 POKE 65495,0: RETURN

STEP 9— Add line 1000 POKE 65494,0: CLS: END

With these changes you get to see the computer run at different speeds. Hope these changes might help someone.

NO GAPS IN NAMES

John Wagnon
Chesterfield County School District
141 Main Street
Chesterfield, SC 29709

In your February issue William Smith offered a helpful suggestion (for the program Microfiles) for storing first and last names a) in separate fields to allow for sorting by last name and b) in a single field to eliminate unsightly gaps between first and last names when printing mailing lists, etc.

A simpler solution I have found for use on the Model II requires just the two fields (first name-GN$ and last names-SN$) to accomplish both objectives.

The two fields allow for sorting by last name if an alphabetized list is required and also provide for a close abutment of the first and last name if an addressing format is needed. The latter is accomplished with the following program segment:

```
56Ø G = INSTR(GN$," "):'returns length of first
     name + 1
57Ø IF G = Ø THEN G = 11 :'if first name fills
     field Ø will be  returned so in this case
     set G =length of field + 1 (11 in this
     example)
58Ø LPRINTTAB(n)LEFT$(GN$,G);:'truncates excess
```

```
      spaces after first name and semicolon
      inhibits carriage return
590 LPRINTTAB(n+G) SN$:'prints last name
      following one space after first name
```

A similar procedure can be utilized for abutting names when printing last name first or with an additional field for middle names or initials.

I would appreciate knowing a simple means of accomplishing a multiple line feed superior to stringing together LPRINT statements or even using a loop. Also using Model II BASIC, how can you print a "©" sign?

Editor: Multiple linefeeds can be generated using the STRING$ function. To generate the © on printers which have the character, use LPRINT CHR$ (n) where n is the decimal value for © (from your line printer manual).

Typewriter

R. Donald Brough
PAR MANAGEMENT, INC.
P.O. Box 8229
Honolulu, HI 96815

With a slight variation to the Typewriter program, the left margin can be adjusted to suit different requirements, and typing errors can be corrected before printing. The original program does not provide either of these features. Here is my version:

```
10 REM ***** TYPEWRITER MODE
15 CLEAR 600
20 CLS
30 PRINT@ 112, "TYPEWRITER MODE"
40 PRINT@ 192, "TO EXIT MODE USE <<UP ARROW>>9"
41 PRINT@ 240, "PAGE LENGTH";:INPUT PG
45 PRINT@ 320, "SET LEFT MARGIN";:INPUT L
50 PRINT@ 360, "SET RIGHT MARGIN";:INPUT A
52 IF A > 79 THEN 53 ELSE 54
53 PRINT "RIGHT MARGIN MUST NOT EXCEED 79":GOTO
     50
54 SYSTEM "FORMS"
55 CLS:F=0:G=0
60 PRINT@ 400+L, "<";:PRINT@ 400+A, ">"
70 IF A > 79 THEN 165
75 Y = 0:P$ = STRING$(80,32)
80 C$ = INKEY$
90 IF C$ = CHR$(8) THEN Y = Y -1:PRINT C$;:GOTO
     80
92 Y = Y + 1
94 MID$(P$,Y,1) = C$
100 IF C$ = CHR$(92) THEN 20
110 IF POS(X) >= A THEN F = 1
120 IF F = 1 THEN IF C$=CHR$(32) THEN C$ =
     CHR$(13)
130 IF C$ = CHR$(13) THEN F=0
140 PRINT TAB(L)C$;
150 IF C$ = CHR$(13) THEN 151 ELSE 160
151 Z = INSTR(P$,CHR$(13))
155 PR$ = LEFT$(P$,Z-1)
156 LPRINT TAB(L)PR$
157 G = G+1
158 IF G = PG THEN SYSTEM "FORMST"
159 GOTO 75
160 GOTO 80
165 Y=0:P$ = STRING$(255,32)
170 C$ = INKEY$
180 IF C$ = "" THEN 170
181 IF C$ = CHR$(8) Y = Y - 1:PRINT C$;:GOTO 170
182 Y = Y + 1
185 MID$(P$,Y,1) = C$
190 IF C$ = CHR$(92) THEN 20
200 IF ROW(X)/2<>INT(ROW(X)/2) AND POS(X)>=A-79
     THEN IF C$ = CHR$(32) THEN C$=
     CHR$(13)
220 IF C$ = CHR$(13) THEN F=0
230 PRINT TAB(L)C$;
240 IF C$ = CHR$(13) THEN 241 ELSE 170
241 Z = INSTR(P$,CHR$(13))
242 PRINT Z
245 PR$ = LEFT$(P$,Z-1)
250 LPRINT TAB(L)PR$:GOTO 165
255 END
```

For an optional page length monitor make the following changes:

```
41 PRINT@240, "PAGE LENGTH";:INPUT PG
156 LPRINT TAB(L)PR$
157 G=G+1
158 IF G=PG THEN SYSTEM"FORMS T"
159 GOTO 75
```

MARCH 1982

SUB DESTROYER

The program Sub Destroyer on page 47 has errors in two lines. The corrected lines should read:

```
290 IF A$="H" THEN K=K+2
515 PRINT@6,"SCORE-"XX,"TIME-"QQ;
```

Histogram

Kenneth Willoughby
P.O. Box 317
Fairacres, NM 88033

The program was written using a RADIO SHACK TRS-80 Model II with 64K. Since it does not need much memory to run or store, it could be easily incorporated into larger programs needing this type of a presentation of the data. Program #1 is for video graphics presented on the screen. The data is stored in the DATA statements with the category first followed by the data amount. Line 50 also has a scaling factor for numbers that are too large for the graph. You can change the 100 to any number suitable to the graph.

PROGRAM #1

```
0 CLS
10 ' HISTOGRAM PROGRAM FOR SIMPLE BAR CHART
20 PRINT"SCALE->";TAB(10)"0";TAB(20)"10";TAB(30)
     "20";TAB(40)"30";TAB(50)"40";TAB(60)"50";TAB(70)"
     60"
30 FOR X=1 TO 12
40 READ A$,Y
50 Y=INT(Y/100+.5)
     : REM SCALING FACTOR IF NUMBERS TOO LARGE FOR
     CHART
60 PRINT A$;
     : PRINT@(X,10),STRING$(Y,"+")
70 NEXT X
80 RESTORE
     : END
90 DATA JAN,5000,FEB,3000,MAR,3500,APR,4500,MAY,5500,
     JUN,6500
100 DATA JUL,6100,AUG,6200,SEP,6300,OCT,6500,NOV,
     6000,DEC,4500
```

PROGRAM #2

To change the program to get hard copy of the histogram edit the following lines.

```
20 LPRINT"SCALE->";TAB(10)"0";TAB(20)"10";TAB(30)
     "20";TAB(40)"30";TAB(50)"40";TAB(60)"50";TAB(70)"
     60"
60 LPRINT A$;
     : TAB(10);STRING$(Y,"+")
```

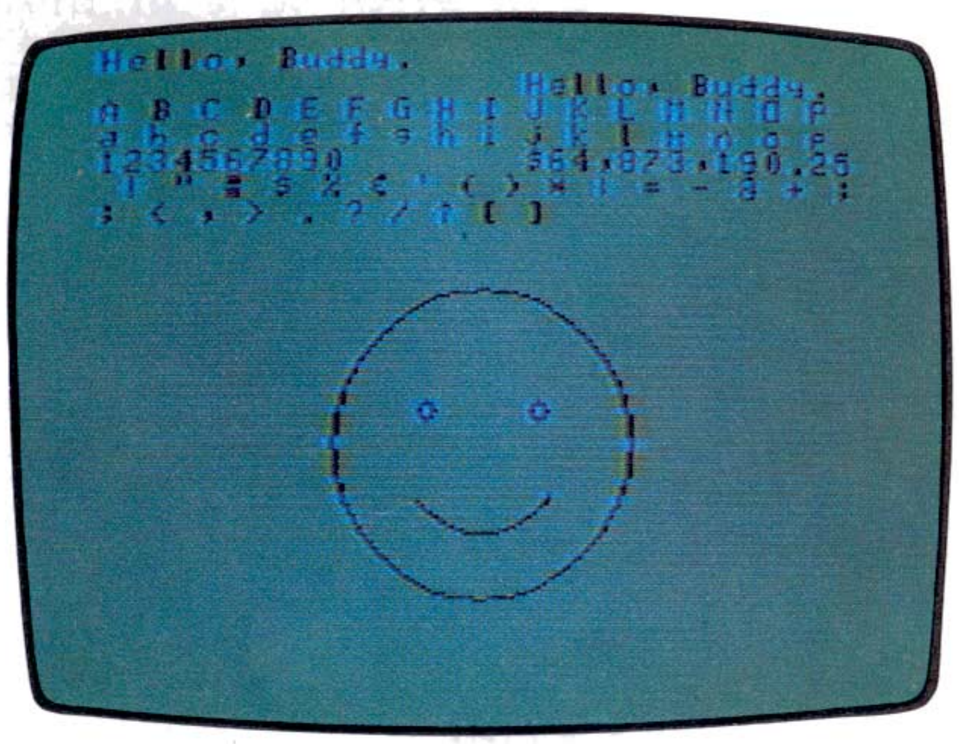
Graphics Programs

High Resolution Character Generation

George Fraser
3403 Bear Creek Drive
Newbury Park, CA 91320

I enjoy my Color Computer very much and think that Extended Color BASIC is great. A serious shortcoming for some applications, however, is the inability to print text on the high resolution screens. The program I am enclosing presents a software solution to this lack.

The program consists of two machine language subroutines and a table, all of which is prepared and saved on a "CSAVEM" tape called "CHR". Once this is prepared and loaded, your high resolution graphics program can "PRINT" text strings on the PMODE4 or PMODE3 screen almost as easily as doing a "PRINT @" instruction. You will never again have to code those long and tedious "DRAW" statements!



There are 90 printable characters accessible from the Color Computer Keyboard, including alpha lower case. This program allows (and requires) the user to pre-define as many of the 90 characters as he wants to use. This permits true lower case letters, different alphabet designs, foreign or Greek letters, math or game symbols, etc.

The Color Computer prints text characters on the screen in a 5 x 7 dot matrix, each character in an 8 x 12 dot "cell." The dot, which may be called a "pixel," is a tiny black spot on the green background when it is "set" (turned on).

My program divides the PMODE4 screen into contiguous (touching) 8 x 8 dot cells. Each cell will hold one character. The full screen equals 24 lines of 32 cells each, for a total of 768 (lower right corner). This is analogous to the "PRINT @" instruction, which uses positions 0 through 511

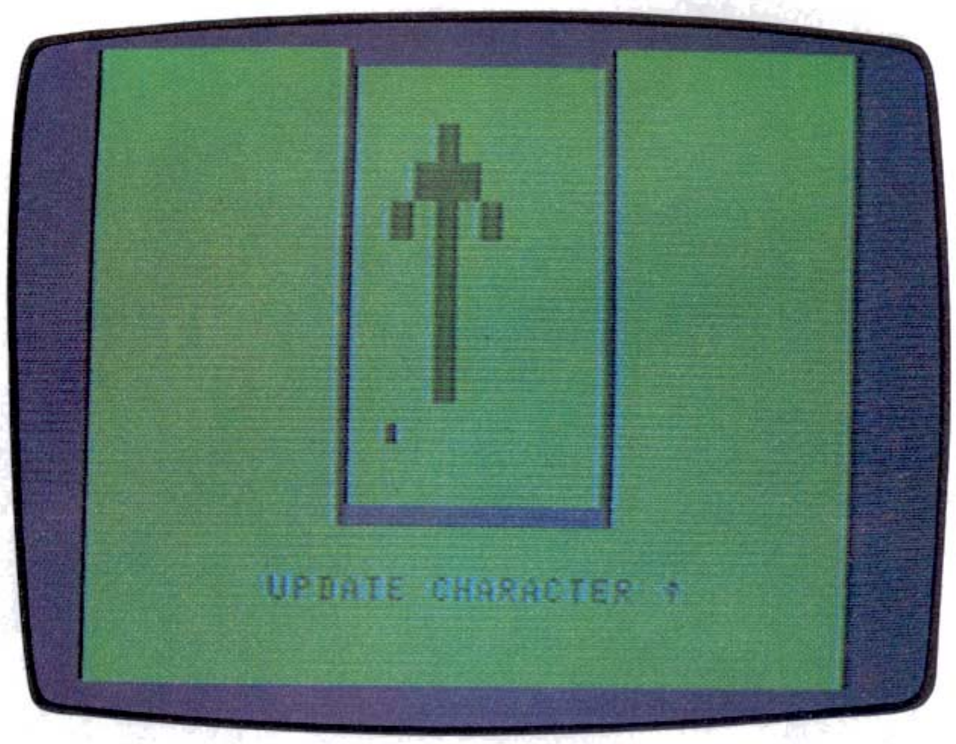
The "CSAVEM" tape you will prepare will be 820 bytes long, including 100 bytes of machine language code and 720 bytes of table. The table contains 8 bytes of bit patterns for each of the 90 characters.

The interface to a graphics program in Extended Color BASIC may be summarized as follows. Each call to USR8 or USR9 passes an argument (A or A$) to it. USR8 returns nothing (X = GARBAGE). USR9 returns X$ = A$ after putting A$ on the screen starting at cell A.

```
CLEAR 5ØØ,15563
L=15564          'LOAD ADDRESS FOR THE CSAVE TAPE
DEFUSR8 = L
DEFUSR9 = L+1Ø
X=USR8(A)        'A IS AN INTEGER FROM Ø TO 767,
                 'THE POSITION CORRESPONDING TO
                 '"PRINT @".
A$=USR9(A$)      'A$ IS THE STRING TO BE "PRINTED".
```

Here is how to prepare your "CHR" tape.

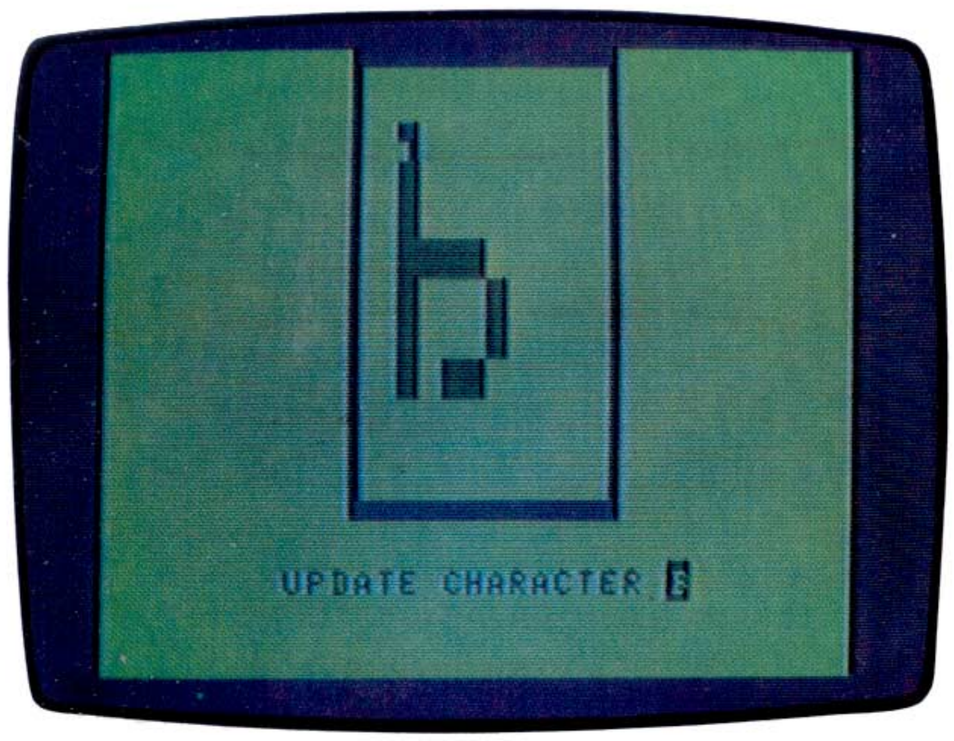
1. Key in and save "LOADTBL".
2. Key in and save "DEFINECH".
3. CLOAD "LOADTBL" and RUN.
4. CLOAD "DEFINECH" and RUN. This will take a while to define each character you need, and may seem tedious at first, but you will get the hang of it.



5. CSAVEM "CHR", 15564, 16383, 350

Here are the instructions for using the program "DEFINECH".

1. The first screen draws a large box and prompts you for a character. Type in a character and enter.
2. The program then gets the 8 x 8 matrix of pixels from your table for that character and displays them inside the box. The tiny black square is your cursor. Any pixel that is "set" shows up as a large black square.
3. Capital "X" will set the pixel and advance the cursor. Space bar will clear the pixel and advance the cursor. The down arrow key will move the cursor



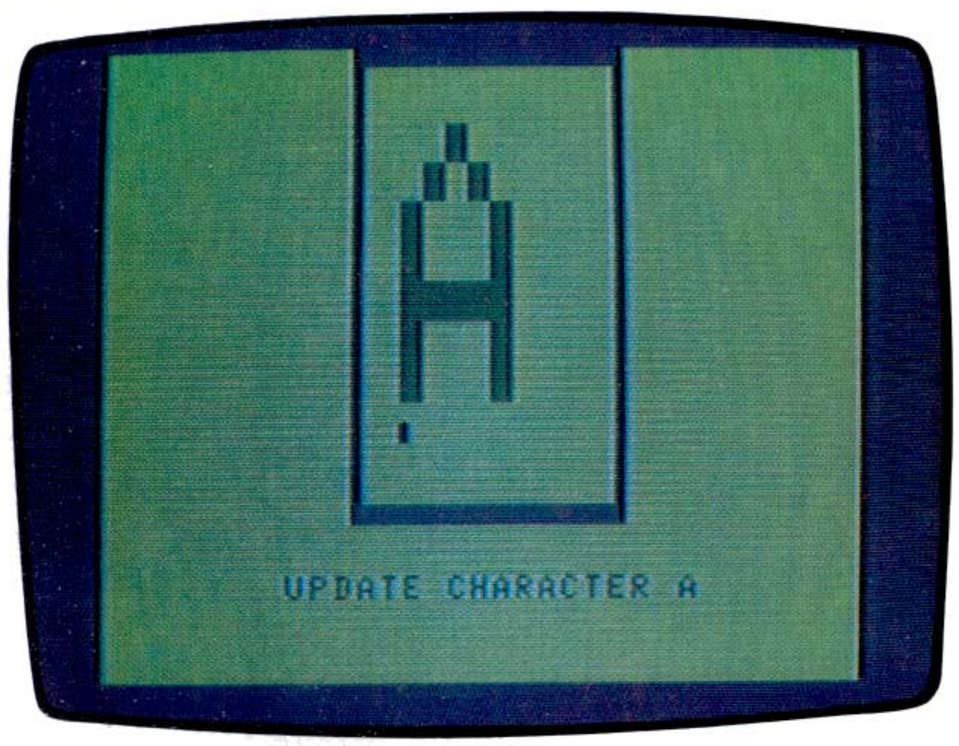
down. Any other key will advance the cursor without change. Enter key will update the table and prompt you for another character. Start off by using the upper left 5 x 7 pixels for your character.

4. If you press the ⟨**ENTER**⟩ key when it prompts you for a new character, the program will display a "PMODE4" graphics screen and will "PRINT" all 90 characters. Press any key to return to the other screen. Use this frequently to check on your progress and to make sure your letters are readable.
5. When you are finished, end the program with the break key and prepare your tape with:

```
CSAVEM "CHR",  15564,  16383,  35Ø
```

6. To update a table you have already created, do this:

```
CLEAR 5ØØ,15563
CLOADM "CHR"
CLOAD "DEFINECH" and run it as before.
CSAVEM your new table with a different
      name.
```



Here are some miscellaneous notes.

The routine will not wrap lines on the screen. When the end of a screen line is reached, any remaining characters in the string are dropped.

The "PMODE4" screen can display lower case alpha characters. Most of these will fit nicely into a 5 x 5 matrix.

For a "PMODE3" screen, a different character set will have to be defined (on a second "CSAVE" tape). Lay out the character in a 6 x 7 matrix, remembering that the final result will "PRINT" with a 3 x 7 matrix because of the lower resolution. Lower case alpha is not practical. This same character set can be used with the "PMODE1" screen, but the letters will be very tall and there are not as many print positions.

The subroutine "PRINTS" the text on graphics pages 1-4, so the calling program should reserve at least 4 pages with PCLEAR4.

It is possible to "REVERSE VIDEO" the character set at any time. The subroutine is set up to "PRINT" dark foreground on a light background in PMODE4, similar to Color BASIC text mode. The calling program can "POKE L+77,18" at any time to switch all characters to light foreground on a dark background. This converts the "COMA" instruction in the M/L routine to a "NOP". L is the USR8 entry address. To turn the reverse video off, "POKE L+77,67". This technique is probably necessary if you are trying to use the routine with PMODE3 screens. Experiment with different foreground and background colors.

```
Ø1Ø                                ;ASSEMBLY LISTING FOR "CHR"
Ø2Ø             ORG     15564
Ø3Ø  3CCC  USR8 JSR     $B3ED      ;CONVERT ARGUMENT TO INTEGER
Ø4Ø  3CCF       STD     PPOS,PCR   ;STORE PRINT @ POSITION IN PROS
Ø5Ø  3CD2       RTS                ;RETURN TO BASIC
Ø6Ø  3CD3  PPOS DEFC    'ØØ'       ;ADDRESS FOR PRINT @ POSITION
Ø7Ø  3CD5  LENG DEFC    'Ø'        ;LENGTH OF STRING TO PRINT
Ø8Ø                                ;X POINTS TO STRING DESCRIPTOR
Ø9Ø                                ;      UPON ENTRY TO USR9
1ØØ  3CD6  USR9 LDA     ,X         ;A=LENGTH OF STRING
11Ø  3CD8       STA     LENG,PCR   ;STORE IT IN LENG
12Ø  3CDB       LDY     2,X        ;Y=ADDRESS OF THE STRING
13Ø  3CDE       LDD     PPOS,PCR   ;D=PRINT @ POSITION
14Ø  3CE1       LDX     #Ø68Ø      ;X=SCREEN ADDRESS
15Ø                                ;
16Ø  3CE4  LOOP CMPD    #ØØ2Ø      ;IF D<32
17Ø  3CE8       BLO     GOT        ;   THEN GOTO GOT
18Ø  3CEA       SUBD    #ØØ2Ø      ;D=D-32
19Ø  3CED       LEAX    256,X      ;X=X+256
2ØØ  3CF1       BRA     LOOP       ;GOTO LOOP
21Ø                                ;
22Ø  3CF3  GOT  LEAX    D,X        ;X=X+D
23Ø  3CF5       CMPX    #1D9F      ;IF X IS OFF THE END OF SCREEN
24Ø  3CF8       BLO     BYTE       ;   THEN RETURN TO BASIC
25Ø  3CFA       RTS
26Ø                                ;PRINT 1 BYTE OF THE STRING
27Ø  3CFB  BYTE LDA     ,Y+        ;A= NEXT BYTE OF THE STRING
28Ø  3CFD       CMPA    #21        ;IF A<33  THEN PRINT SPACE
29Ø  3CFF       BLO     SPAC
3ØØ  3DØ1       CMPA    #7A        ;IF A>122 THEN PRINT SPACE
31Ø  3DØ3       BHI     SPAC
32Ø  3DØ5       SUBA    #21        ;A=A-33
33Ø  3DØ7       BRA     TABL       ;GOTO TABL
34Ø  3DØ9  SPAC LDA     #3F        ;A=63 '
35Ø                                ;
36Ø  3DØB  TABL LDB     #Ø8        ;B=8
37Ø  3DØD       MUL                ;D=A*B
38Ø  3DØE       TFR     PCR,U      ;U=PROGRAM COUNTER
39Ø  3D1Ø       LEAU    D,U        ;U=U+D
4ØØ  3D12       LEAU    32,U       ;U=U+32
41Ø  3D15       LDB     #8Ø        ;B=-128
42Ø                ;U NOW CONTAINS THE ABSOLUTE ADDRESS OF THE
43Ø                ;TABLE ENTRY FOR THE CHARACTER TO BE PRINTED
44Ø                                ;
45Ø  3D17  PRNT LDA     ,U+        ;A=PEEK(U):U=U+1
46Ø  3D19       COMA               ;COMPLEMENT BIT PATTERN OF A
47Ø  3D1A       STA     B,X        ;POKE (B+X),A  ON THE SCREEN
48Ø  3D1C       ADDB    #2Ø        ;B=B+32
485  3D1E       CMPB    #8Ø        ;IF B<>128
49Ø  3D2Ø       BNE     PRNT       ;   THEN GOTO PRNT
5ØØ  3D22       TFR     X,D        ;D=X  IN ORDER TO TEST LSB OF X
51Ø  3D24       CMPB    #9F        ;IF END OF A LINE ON SCREEN
52Ø  3D26       BEQ     DONE       ;   THEN GOTO DONE
53Ø  3D28       LEAX    1,X        ;X=X+1   NEXT PRINT POSITION
54Ø  3D2A       DEC     LENG,PCR   ;LENG = LENG - 1
55Ø  3D2D       BNE     BYTE       ;IF LENG<>Ø THEN GET NEXT BYTE
56Ø  3D2F  DONE RTS                ;RETURN TO BASIC
```

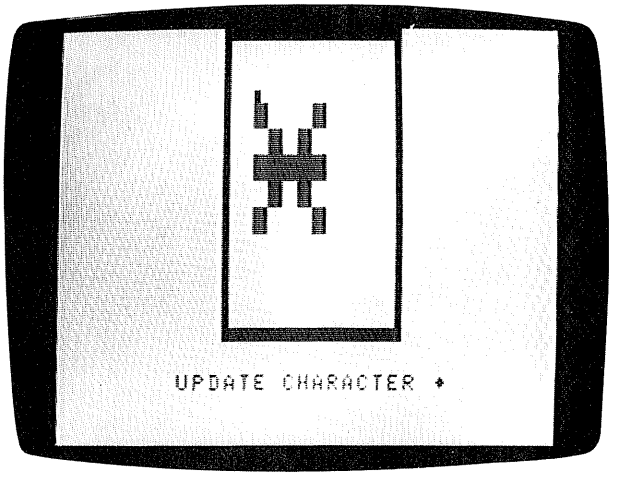
"LOADTBL"

```
1Ø CLEAR 5ØØ, 15563
2Ø FOR A=15564 TO 15663
3Ø READ B
     :  POKE A, B
     :  NEXT
4Ø FOR A=15664 TO 16383
5Ø POKE A, Ø
```

```
    :  NEXT
6Ø DATA 189, 179, 237, 237, 14Ø, 1, 57, Ø, Ø, Ø
7Ø DATA 166, 132, 167, 14Ø, 25Ø, 16, 174, 2, 236
8Ø DATA 14Ø, 242, 142, 6, 128, 16, 131, Ø, 32
9Ø DATA 37, 9, 131, Ø, 32, 48, 137, 1, Ø, 32
1ØØ DATA 241, 48, 139, 14Ø, 29, 159, 37, 1, 57
11Ø DATA 166, 16Ø, 129, 33, 37, 8, 129, 122, 34
12Ø DATA 4, 128, 33, 32, 2, 134, 63, 198, 8, 61
13Ø DATA 31, 83, 51, 2Ø3, 51, 2ØØ, 32, 198, 128
14Ø DATA 166, 192, 67, 167, 133, 2Ø3, 32, 193
15Ø DATA 128, 38, 245, 31, 16, 193, 159, 39, 7
155 DATA 48, 1, 1Ø6, 14Ø, 168, 38, 2Ø4, 57
16Ø '
17Ø '  "LOADTBL"   BY G.FRASER
18Ø DATA 32, 32, 32, 32, 32, Ø, 32, Ø          ' !
19Ø DATA 72, 72, 72, Ø, Ø, Ø, Ø, Ø             ' "
2ØØ DATA 8Ø, 8Ø, 248, Ø, 248, 8Ø, 8Ø, Ø        ' #
21Ø DATA 32, 12Ø, 128, 112, 8, 24Ø, 64, Ø      ' $
22Ø DATA 2ØØ, 2ØØ, 16, 32, 64, 152, 152, Ø     ' %
23Ø DATA 16, 12Ø, 128, 96, 128, 12Ø, 16, Ø     ' &
24Ø DATA 32, 32, 32, Ø, Ø, Ø, Ø, Ø             ' '
25Ø DATA 16, 32, 64, 64, 64, 32, 16, Ø         ' (
26Ø DATA 64, 32, 16, 16, 16, 32, 64, Ø         ' )
27Ø DATA Ø, 136, 8Ø, 248, 8Ø, 136, Ø, Ø        ' *
28Ø DATA Ø, 32, 32, 248, 32, 32, Ø, Ø          ' +
29Ø DATA Ø, Ø, Ø, 48, 48, 16, 32, Ø            ' ,
3ØØ DATA Ø, Ø, Ø, 248, Ø, Ø, Ø, Ø              ' -
31Ø DATA Ø, Ø, Ø, Ø, Ø, 48, 48, Ø              ' .
32Ø DATA 8, 8, 16, 32, 64, 128, 128, Ø         ' /
33Ø DATA 48, 72, 72, 72, 72, 72, 48, Ø         ' Ø
34Ø DATA 32, 96, 32, 32, 32, 32, 112, Ø        ' 1
35Ø DATA 112, 136, 8, 48, 64, 128, 248, Ø      ' 2
36Ø DATA 112, 136, 8, 48, 8, 136, 112, Ø       ' 3
37Ø DATA 16, 48, 8Ø, 144, 248, 16, 16, Ø       ' 4
38Ø DATA 248, 128, 24Ø, 8, 8, 136, 112, Ø      ' 5
39Ø DATA 112, 128, 128, 24Ø, 136, 136, 112, Ø  ' 6
4ØØ DATA 248, 8, 16, 32, 64, 128, 128, Ø       ' 7
41Ø DATA 112, 136, 136, 112, 136, 136, 112, Ø  ' 8
42Ø DATA 112, 136, 136, 12Ø, 8, 8, 112, Ø      ' 9
43Ø DATA Ø, 32, 32, Ø, 32, 32, Ø, Ø            ' :
44Ø DATA Ø, 48, 48, Ø, 48, 16, 32, Ø           ' ;
45Ø DATA 8, 16, 32, 64, 32, 16, 8, Ø           ' <
46Ø DATA Ø, Ø, 248, Ø, 248, Ø, Ø, Ø            ' =
47Ø DATA 128, 64, 32, 16, 32, 64, 128, Ø       ' >
5ØØ DATA 112, 136, 8, 16, 32, Ø, 32, Ø         ' ?
51Ø DATA 112, 136, 8, 1Ø4, 154, 136, 112, Ø    ' @
52Ø DATA 32, 8Ø, 136, 136, 248, 136, 136, Ø    ' A
53Ø DATA 24Ø, 72, 72, 112, 72, 72, 24Ø, Ø      ' B
54Ø DATA 112, 136, 128, 128, 128, 136, 112, Ø  ' C
55Ø DATA 24Ø, 72, 72, 72, 72, 72, 24Ø, Ø       ' D
56Ø DATA 248, 128, 128, 24Ø, 128, 128, 248, Ø  ' E
57Ø DATA 248, 128, 128, 24Ø, 128, 128, 128, Ø  ' F
58Ø DATA 12Ø, 128, 128, 152, 136, 136, 12Ø, Ø  ' G
59Ø DATA 136, 136, 136, 248, 136, 136, 136, Ø  ' H
6ØØ DATA 112, 32, 32, 32, 32, 32, 112, Ø       ' I
61Ø DATA 8, 8, 8, 8, 8, 136, 112, Ø            ' J
62Ø DATA 136, 144, 16Ø, 192, 16Ø, 144, 136, Ø  ' K
63Ø DATA 128, 128, 128, 128, 128, 128, 248, Ø  ' L
64Ø DATA 136, 216, 168, 168, 136, 136, 136, Ø  ' M
65Ø DATA 136, 2ØØ, 168, 152, 136, 136, 136, Ø  ' N
66Ø DATA 248, 136, 136, 136, 136, 136, 248, Ø  ' O
67Ø DATA 24Ø, 136, 136, 24Ø, 128, 128, 128, Ø  ' P
68Ø DATA 112, 136, 136, 136, 168, 144, 1Ø4, Ø  ' Q
69Ø DATA 24Ø, 136, 136, 24Ø, 16Ø, 144, 136, Ø  ' R
7ØØ DATA 112, 136, 64, 32, 16, 136, 112, Ø     ' S
71Ø DATA 248, 32, 32, 32, 32, 32, 32, Ø        ' T
72Ø DATA 136, 136, 136, 136, 136, 136, 112, Ø  ' U
73Ø DATA 136, 136, 136, 8Ø, 8Ø, 32, 32, Ø      ' V
74Ø DATA 136, 136, 136, 168, 168, 216, 136, Ø  ' W
75Ø DATA 136, 136, 8Ø, 32, 8Ø, 136, 136, Ø     ' X
76Ø DATA 136, 136, 8Ø, 32, 32, 32, 32, Ø       ' Y
77Ø DATA 248, 8, 16, 32, 64, 128, 248, Ø       ' Z
78Ø DATA 112, 64, 64, 64, 64, 64, 112, Ø       ' [
79Ø DATA 128, 128, 64, 32, 16, 8, 8, Ø         ' \
8ØØ DATA 112, 16, 16, 16, 16, 16, 112, Ø       ' ]
81Ø DATA 32, 112, 168, 32, 32, 32, 32, Ø       '
    (UP ARROW)
82Ø DATA Ø, 32, 64, 248, 64, 32, Ø, Ø          '
    (LEFT ARROW)
83Ø DATA Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø                '
    (SPACE)
84Ø DATA Ø, Ø, 24, 8, 12Ø, 136, 12Ø, Ø         ' a
85Ø DATA 128, 128, 128, 24Ø, 136, 136, 176, Ø  ' b
86Ø DATA Ø, Ø, 112, 136, 128, 136, 112, Ø      ' c
87Ø DATA 8, 8, 8, 12Ø, 136, 136, 12Ø, Ø        ' d
88Ø DATA Ø, Ø, 112, 136, 248, 128, 112, Ø      ' e
89Ø DATA 48, 32, 32, 248, 32, 32, 96, Ø        ' f
9ØØ DATA Ø, Ø, 12Ø, 136, 12Ø, 8, 48, Ø         ' g
91Ø DATA 128, 128, 176, 2ØØ, 136, 136, 136, Ø  ' h
92Ø DATA Ø, 32, Ø, 32, 32, 32, 112, Ø          ' i
93Ø DATA Ø, 16, Ø, 16, 16, 144, 96, Ø          ' j
94Ø DATA 128, 128, 144, 16Ø, 192, 16Ø, 144, Ø  ' k
95Ø DATA 64, 64, 64, 64, 64, 64, 96, Ø         ' l
96Ø DATA Ø, Ø, 136, 216, 168, 136, 136, Ø      ' m
97Ø DATA Ø, Ø, 176, 2Ø7, 136, 136, 136, Ø      ' n
98Ø DATA Ø, Ø, 112, 136, 136, 136, 112, Ø      ' o
99Ø DATA Ø, Ø, 24Ø, 136, 24Ø, 128, 128, Ø      ' p
1ØØØ DATA Ø, Ø, 112, 136, 136, 12Ø, 8, Ø       ' q
1Ø1Ø DATA Ø, Ø, 152, 224, 128, 128, 128, Ø     ' r
1Ø2Ø DATA Ø, Ø, 112, 128, 112, 8, 112, Ø       ' s
1Ø3Ø DATA Ø, 32, 248, 32, 32, 32, 48, Ø        ' t
1Ø4Ø DATA Ø, Ø, 136, 136, 136, 152, 1Ø4, Ø     ' u
1Ø5Ø DATA Ø, Ø, 136, 136, 8Ø, 8Ø, 32, Ø        ' v
1Ø6Ø DATA Ø, Ø, 136, 136, 168, 216, 136, Ø     ' w
1Ø7Ø DATA Ø, Ø, 136, 8Ø, 32, 8Ø, 136, Ø        ' x
1Ø8Ø DATA Ø, Ø, 136, 136, 248, 8, 56, Ø        ' y
1Ø9Ø DATA Ø, Ø, 248, 16, 32, 64, 248, Ø        ' z
```

"DEFINECH"

```
1Ø PCLEAR 4
    :  GOTO 11Ø
2Ø P=Ø
    :  C=11ØØ
3Ø IF PEEK(C)=96 THEN POKE C, 141
                 ELSE POKE C, 13Ø
4Ø I$=INKEY$
    :  IF I$="" THEN 4Ø ELSE I=ASC(I$)
    :  IF I=88 THEN V=32 ELSE IF I=32 THEN V=96
       ELSE IF PEEK(C)=13Ø THEN V=32 ELSE V=96
5Ø POKE C, V
    :  IF I=13 THEN 8Ø ELSE IF I=1Ø THEN C=C+32
    :  GOTO 6Ø ELSE IF P<7 THEN P=P+1
    :  C=C+1
    :  GOTO 3Ø ELSE P=Ø
    :  C=C+25
6Ø IF C<1332 THEN 3Ø ELSE 2Ø
7Ø '          ENTER KEY
    :  SAVE THE UPDATES TO THE TABLE
8Ø FOR Y=Ø TO 7
    :  B=Ø
    :  FOR X=Ø TO 7
    :  B=B*2
    :  V=PEEK(Y*32+11ØØ+X)
    :  IF V=32 THEN B=B+1
9Ø NEXT X
    :  POKE W+Y, B
    :  NEXT Y
1ØØ '          PROMPT SCREEN FOR CHARACTER UPDATE
```




```
11Ø CLS
    :  FOR V=21 TO 42
    :  SET(V, 1, 1)
    :  SET(V, 22, 1)
    :  NEXT V
    :  FOR V=2 TO 21
    :  SET(21, V, 1)
    :  SET(42, V, 1)
    :  NEXT V
12Ø PRINT @ 423,  "UPDATE CHARACTER ";
    :  LINEINPUT A$
    :  IF A$="" THEN 18Ø ELSE W=ASC(A$)
    :  PRINT @ 44Ø,  CHR$(W)
    :  IF W<33 OR W>122 THEN W=63 ELSE W=W-33
13Ø '   IF THE BIT IS SET,  DISPLAY BLACK ASC(32)
14Ø W=W*8+15664
    :  FOR Y=Ø TO 7
    :  B=PEEK(W+Y)
    :  IF B=Ø THEN 16Ø ELSE FOR X=7 TO Ø STEP -1
    :  V=B/2
    :  IF V-FIX(V) >Ø THEN POKE Y*32+11ØØ+X, 32
```

```
15Ø B=FIX(V)
     :  NEXT X
16Ø NEXT Y
     :  GOTO 2Ø
17Ø '     HIGH RESOLUTION GRAPHICS TEST PROGRAM
18Ø PMODE 4, 1
     :  COLOR Ø, 1
     :  L=15564
     :  DEFUSR8=L
     :  DEFUSR9=L+1Ø
     :  PCLS
     :  SCREEN 1, 1
19Ø B=USR8(Ø)
     :  B$=USR9("HELLO,  BUDDY.")
2ØØ B=5Ø
     :  A=USR8(B)
     :  A$=USR9(B$)
21Ø A=USR8(64)
     :  A$=USR9("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
22Ø A=USR8(96)
     :  A$=USR9("abcdefghijklmnopqrstuvwxyz")
23Ø A=USR8(128)
     :  A$=USR9("123456789Ø      $64,873,19Ø.25")
24Ø A=USR8(16Ø)
     :  B$=" !" + CHR$(34) + "#$%&'()*: =-@+;<,
           >.?/^ [\]"
     :  A$=USR9(B$)
25Ø CIRCLE(128, 12Ø), 52, , .9
     :  CIRCLE(128, 12Ø), 3Ø, , .9, .1, .4
     :  CIRCLE(1Ø8, 11Ø), 3
     :  CIRCLE(148, 11Ø), 3
26Ø A$=INKEY$
     :  IF A$="" THEN 26Ø ELSE 11Ø
27Ø '
28Ø '  "DEFINECH"   BY G.FRASER
29Ø '   BEFORE RUNNING,  DO THE FOLLOWING
          IMMEDIATE COMMANDS:
3ØØ '     CLEAR 5ØØ, 15563
31Ø '     CLOADM "CHR"
           (OR ELSE, RUN PROGRAM "LOADTBL")
```

TEXTURE

George Fraser
3403 Bear Creek Drive
Newbury Park, CA 91320

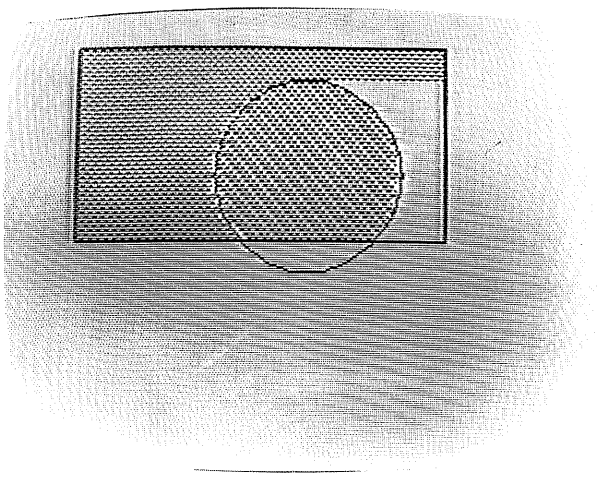
Color Computer owners, wish you had more than just black and white (or green and black) for your highest resolution PMODE 4 screen? Want to brighten things up a little? How about four colors or more in addition to your highest resolution black and white? How about three different shades of blue on the same screen? How about being able to texture or shade figures with an almost unlimited variety of patterns, colored and monochrome?

Look, Ma, no hands! Now you can have all of these features at once on the same screen by using my little program I call "Texture." It was sort of inspired by Chamberlain's "Designs" in the December 1981 Newsletter. I wanted a way to control the generation of colors and patterns on the graphics screen. My resulting machine language program is similar to Radio Shack's "PAINT" command but is a little more versatile.

Key in the BASIC program carefully and make sure you have a good, usable copy CSAVED on tape before you try to run it for the first time. A mistake in the machine language routine can easily destroy your BASIC program in memory.

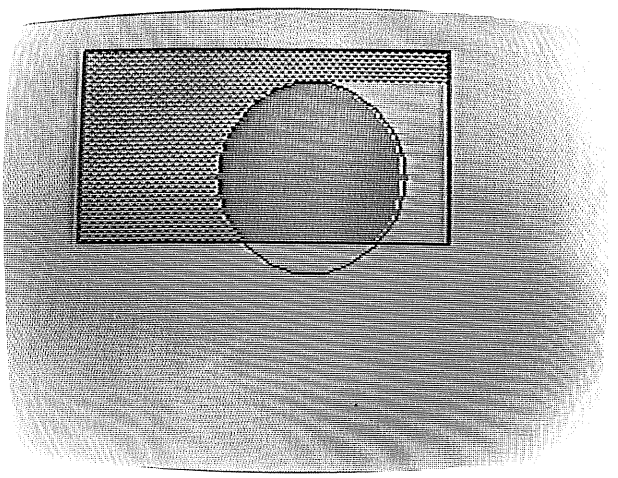
Here is how it works. You clear the graphics screen with background color (white) and draw a figure with foreground color (black). You must then supply five parameters for the M/L subroutine (USR7), which I call A, B, C, X, and Y.

X and Y have about the same meaning they do for the Radio Shack PAINT command. They are the coordinates of a point inside of a closed figure you want to paint. In addition, X must be greater than 1, and Y must be on the top line to be painted. My program is not as smart as "PAINT", because it works only from the top down, and cannot find its way to all parts of a complex figure, as line 170 of the BASIC program will show.



A, B, and C control the color or texture pattern that the figure will be "painted" with. Each is a number from 0 to 255. The first line "painted" is done using the bit pattern of A (taken to be an eight bit binary number). Similarly, the second line is done using the bit pattern of B. Then if C is not zero, the bit patterns of A and B are shifted left or right the number of bits specified by C. Then the third line is generated from A, the fourth from B, etc. If C is zero, A and B are not shifted. C = 1 means shift 1 bit right, 2 means 2 bits right, 255 means shift 1 bit left, 254 means shift 2 bits left, etc.

Here is the scoop on A and B. A value of zero causes black (foreground color). 255 causes white (background color, no change). Anything in between will generate various kinds of color and/or texture, but you will have to experiment to see what you can get. Multiples of 17 give generally good results. (In binary, they give 4 bit repeating patterns.)



Examples:

A or B	Bit Pattern (1 = white pixel)
0	00000000
255	11111111
17	00010001
34	00100010
221 (17*13)	11011101

The following table summarizes some of the better combinations of A, B, and C that I have found. The colors produced on your own TV set may be slightly different.

A,	B,	C	COLOR/TEXTURE
85,	85,	0	CLEAR BLUE
170,	170,	0	CLEAR BRIGHT RED
85,	255,	0	LIGHT BLUE STRIPES
170,	255,	0	LIGHT RED STRIPES
204,	51,	0	DARK BLUE CROSSHATCH
17,	17,	2	GRAY BLUE CROSSHATCH
34,	34,	2	BRIGHT RED CROSSHATCH
187,	187,	2	BLUE CROSSHATCH
119,	119,	2	YELLOW CROSSHATCH
153,	102,	0	GREEN TEXTURED
153,	255,	2	LIGHT GREEN TEXTURED
204,	255,	2	VIOLET TEXTURED
34,	255,	2	RED TEXTURED
17,	255,	2	GRAY TEXTURED

Edges may be a little muddy someplaces, but the overall effect is surprisingly good. Now you should be able to turn out some really professional looking colored graphs, pie charts, etc.

When you are sure the M/L routine is working OK, save it to tape with: CSAVEM "TEXML", 15882, 16167, 350. To incorporate the routine in another grahics program, you will need to include lines 30, 40, 60, 70, 90, 100, 110, 150, and 160. To protect memory for the machine language program, you will load the M/L program using the "CLOADM" command after doing a "CLEAR 200, 15871" (the extra ten bytes from 15871 to 15881 are a data area.)

The machine language routine can be relocated to any protected area of memory by changing S in line 40, provided that S is a multiple of 256. This is the reason for line 220. The first 10 bytes starting at S is a "scratch pad area" which is addressed directly (through the direct page register), so it must be properly boundary aligned to work.

To use "TEXTURE" with Radio Shack's 8 bit print driver program, you will have to change line 30 to "CLEAR 200,15359" and line 40 to "S = 15360". Then you can: CSAVE "TEXML", 15370, 15655, 350. Then, to get everything set up, do the following:

```
CLEAR 200, 15359
CLOADM "SCRPRT" (RADIO SHACK PRINT DRIVER)
EXEC
POKE 16303,0255 (REVERSE PRINT FEATURE)
CLOADM "TEXML"  (TEXTURE PROGRAM)
CLOAD           (YOUR GRAPHICS PROGRAM IN BASIC)
```

The "TEXTURE" subroutine will also work with PMODE3 graphics (try it). It also gives very good results with a black and white TV.

The Program

```
10 '  "TEXTURE"  BY G.FRASER  FEB 1982
20 '   FOR THE TRS-80 COLOR COMPUTER
30 CLEAR 200, 15871
40 S=15872     'LOAD ADDR FOR M/L
50 GOSUB 220   'LOAD M/L ROUTINE
60 PMODE 4, 1
     : COLOR 0, 1
     : CLS
70 DEFUSR7=S+10
80 PRINT "-----"
     : INPUT "A, B, C"; A, B, C
90 POKE S, A      '1ST TEXTURE
100 POKE S+1, B   '2ND TEXTURE
110 POKE S+2, C   'OFFSET
120 PCLS
     : SCREEN 1, 1
130 CIRCLE (128, 70), 50, , .9
140 LINE (10, 10)-(200, 100), PSET, B
150 X=128
     : Y=27
     : D = 128*Y + FIX(X/2)
160 Z=USR7(D)
170 POKE S, 17
     : POKE S+1, 255
     : POKE S+2, 2
     : Z=USR7(1430)
180 SOUND 100, 1
190 A$ = INKEY$
200 IF A$ = "" THEN 190 ELSE 80
210 '  LOAD M/L CODE
220 IF S/256 <> FIX(S/256) THEN STOP
230 PRINT "LOADING M/L CODE"
240 FOR A=S+10 TO S+295
250 READ B
     : POKE A, B
     : C=C+B
260 NEXT A
270 IF C=23720 THEN RETURN ELSE PRINT C
     : STOP
280 DATA 189, 179, 237, 52, 6, 31, 80, 31, 139
290 DATA 53, 6, 142, 6, 0, 16, 131, 0, 128, 37, 8
300 DATA 131, 0, 128, 48, 136, 32, 32, 242, 140
310 DATA 29, 224, 16, 34, 0, 133, 31, 18, 215, 6
320 DATA 215, 7, 84, 84, 58, 214, 6, 141, 65
330 DATA 215, 3, 15, 4, 134, 128, 151, 9, 134
340 DATA 255, 151, 8, 76, 129, 127, 34, 86, 141
350 DATA 36, 141, 44, 209, 3, 38, 12, 13, 8, 42
360 DATA 239, 145, 7, 36, 87, 151, 8, 32, 231
370 DATA 13, 8, 43, 227, 145, 6, 34, 6, 198, 255
380 DATA 215, 8, 32, 217, 151, 9, 32, 48, 31, 33
390 DATA 31, 137, 84, 84, 58, 31, 137, 57, 196
400 DATA 3, 38, 11, 198, 192, 228, 132, 84, 84
410 DATA 84, 84, 84, 84, 57, 193, 1, 38, 6, 198
420 DATA 48, 228, 132, 32, 241, 193, 2, 38, 6
425 DATA 198, 12, 228, 132, 32, 233, 228, 132
430 DATA 57, 10, 9, 214, 9, 215, 7, 214, 8, 43
440 DATA 7, 215, 6, 92, 209, 9, 37, 4, 79, 31
450 DATA 139, 57, 150, 0, 13, 4, 39, 2, 150, 1
460 DATA 151, 5, 150, 8, 32, 5, 76, 145, 9, 34
470 DATA 41, 141, 166, 196, 3, 38, 4, 198, 192
480 DATA 32, 14, 193, 1, 38, 4, 198, 48, 32, 6
490 DATA 193, 2, 38, 2, 198, 12, 52, 4, 212, 5
500 DATA 52, 4, 230, 97, 83, 228, 132, 234, 225
510 DATA 231, 132, 32, 210, 13, 4, 38, 4, 12, 4
520 DATA 32, 34, 214, 2, 215, 4, 39, 28, 43, 14
530 DATA 220, 0, 70, 6, 0, 86, 6, 1, 10, 4, 38
540 DATA 244, 32, 12, 220, 0, 73, 9, 0, 89, 9, 1
550 DATA 12, 4, 38, 244, 49, 168, 32, 16, 140, 29
560 DATA 224, 34, 141, 22, 255, 22
```

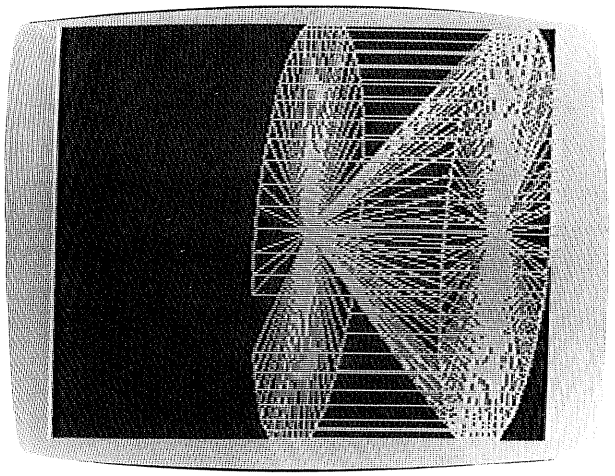
Rotational Grafx

John Ennis
436 East Redbud, #11
Knoxville, TN 37920

"Rotational Grafx" is a program that, given rational information, generates objects of rotation in a 3D-space. The conventional axis is used as a reference. An entry of co-ordinates may be a two-dimensional trace or a three-dimensional object. This object can then be rotated about any axis in the system.

The first part of the program generates the object and pokes in data at specified locations. The second part of the program then displays the object according to specified parameters.

During viewing of the object, the user can change viewing parameters by pressing the spacebar. Pressing "N" allows user to generate a new object after editing any input information he chooses.



The displacement is entered as 0,0,0 for the First generation. Scaling constant is set at 1. The rotation direction vector is a unit vector orthogonal to planes of symmetry. To rotate given points about the Z-axis, the rotation direction vector is 0,0,1. Any vector may be used with magnitude ($=SQR\ (X)^2+(Y)^2+(Z)^2$) equal to 1. The terminal rotation angle for full rotation in a clockwise direction is approx. 6.3 radians. The rotation angle increment sets radians per frame. Keep magnitude of each point <100 to be safe.

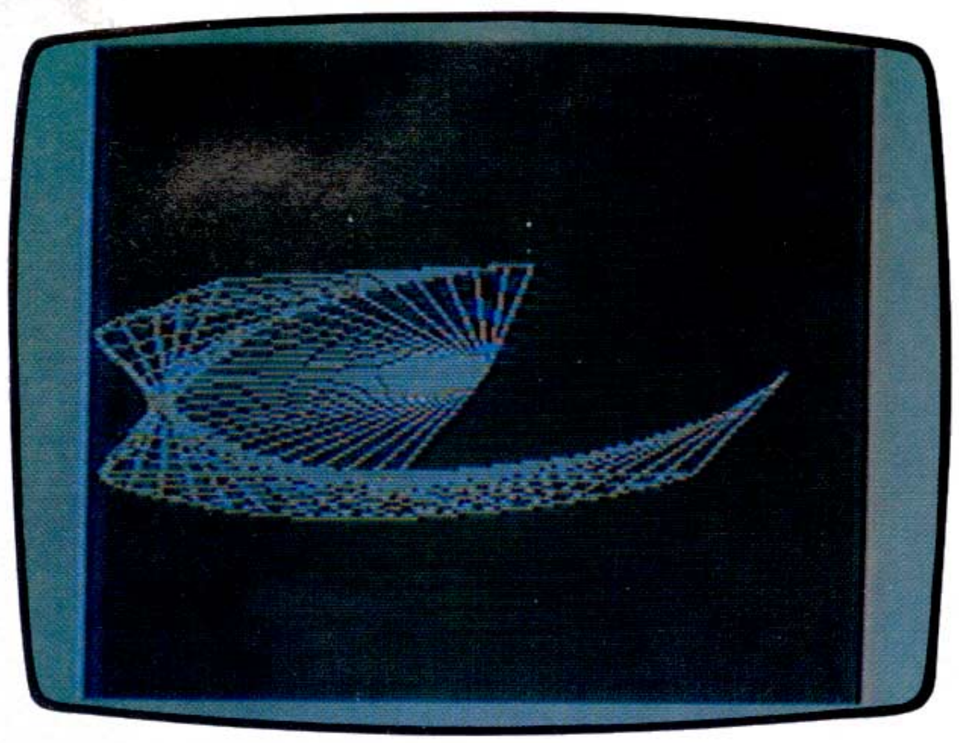
VIEWING OBJECT

A view object increment of 1 will display the object as generated. Traces may be erased to view each successive frame alone.

Other functions will become apparent with use.

```
1Ø 'REM:"ROTATIONAL GRAFX" 16K EXTENDED BASIC
     REQUIRED                JOHN ENNIS 436 E.REDBUD
     DR.#11,KNOXVILLE,TN. 37920  JAN.1,1982
2Ø 'REM:COPYRIGHT CLAIMED ONLY UPON SALE TO THIRD
     PARTY.
3Ø CLS
     : U=45ØØ
4Ø SOUND 128, 2
     : LINEINPUT "generate new object? " ; A$
5Ø IF A$="YES" THEN GOTO 7Ø
6Ø GOTO 6ØØ
7Ø SOUND 2ØØ, 1
     : CLS
     : INPUT "displacement"; XØ, YØ, ZØ
8Ø SOUND 2ØØ, 1
     : INPUT "scaling constant"; S1
9Ø SOUND 2ØØ, 1
     : INPUT "rotation dir.vector"; A, B, C
1ØØ SOUND 2ØØ, 1
     : INPUT "terminal rotation angle"; R
11Ø SOUND 2ØØ, 1
     : INPUT "rotation angle increment"; S
12Ø U2=U-5ØØ
13Ø POKE 16351, R
     : POKE 16352, S*1ØØ
14Ø IF W=Ø THEN GOTO 16Ø
15Ø SOUND 22Ø, 2
     : LINEINPUT "new points? "; D$
16Ø IF D$="NO" THEN GOTO 19Ø
17Ø IF W>Ø THEN PRINT "last n="; PEEK(1635Ø)
18Ø SOUND 2ØØ, 1
     : INPUT "number of points"; N
     : GOTO 2ØØ
19Ø N=PEEK(1635Ø)
2ØØ S2=FIX(R/S*2*N)
     : S3=S2/U2*S
21Ø S3=INT(S3*1ØØØ1)
     : S3=S3/1ØØØØ
22Ø IF S2>U2 THEN PRINT "mem.not
     protected:min.increment="; S3
23Ø PRINT "required:available"; S2; ":"; U2
     : FOR DLAY=1 TO 1ØØ
     : NEXT DLAY
24Ø IF S2>U THEN GOTO 1ØØ
25Ø POKE 1635Ø, N
26Ø IF D$="NO" THEN GOTO 33Ø
27Ø N=PEEK(1635Ø)
28Ø FOR D=12ØØØ TO 12ØØØ+(N*3)-3 STEP 3
29Ø SOUND 22Ø, 1
     : INPUT "input coordinate"; X, Y, Z
3ØØ X=1ØØ+X
     : Y=1ØØ+Y
     : Z=1ØØ+Z
31Ø POKE D, X
     : POKE D+1, Y
     : POKE D+2, Z
32Ø NEXT D
33Ø SOUND 2ØØ, 1
     : SOUND 1ØØ, 2
34Ø F=12ØØØ
     : N=PEEK(1635Ø)
35Ø W=Ø
     : O=Ø
36Ø X=PEEK(F)
     : Y=PEEK(F+1)
     : Z=PEEK(F+2)
37Ø X=X-1ØØ
     : Y=Y-1ØØ
     : Z=Z-1ØØ
38Ø T=125ØØ+W
39Ø X2=X+XØ
     : Y2=Y+YØ
     : Z2=Z+ZØ
4ØØ FOR Q=Ø TO R STEP S
41Ø V=1-COS(Q)
42Ø L1=(A*A*V)+(COS(Q))
     : L2=(A*B*V)-(C*SIN(Q))
     : L3=(A*C*V)+(B*SIN(Q))
43Ø M1=(B*A*V)+(C*SIN(Q))
     : M2=(B*B*V)+(COS(Q))
     : M3=(B*C*V)-(A*SIN(Q))
44Ø N1=(C*A*V)-(B*SIN(Q))
     : N2=(C*B*V)+(A*SIN(Q))
     : N3=(C*C*V)+(COS(Q))
45Ø X3=(L1*X2)+(M1*Y2)+(N1*Z2)
46Ø Y3=(L2*X2)+(M2*Y2)+(N2*Z2)
47Ø Z3=(L3*X2)+(M3*Y2)+(N3*Z2)
48Ø CLS
49Ø X3=X3*S1
     : Y3=Y3*S1
     : Z3=Z3*S1
5ØØ X3=FIX(X3)
     : Y3=FIX(Y3)
     : Z3=FIX(Z3)
51Ø PRINT "point#"; O+1; ":"; "X="; X3; "Y="; Y3;
     "Z="; Z3
52Ø PRINT "memory location:"; T; T+1
53Ø X3=.3*X3
     : X=128+Y3-X3
     : Y=96-Z3+X3
54Ø X7=ABS(X3)
     : Y7=ABS(Y3)
     : Z7=ABS(Z3)
55Ø X=FIX(X)
     : Y=FIX(Y)
56Ø POKE T, X
     : POKE T+1, Y
57Ø T=T+(2*N)
     : NEXT Q
58Ø W=W+2
     : O=O+1
     : IF O=N THEN GOTO 6ØØ
59Ø F=F+3
     : GOTO 36Ø
6ØØ CLS
61Ø SOUND 1ØØ, 5
     : INPUT "viewing increment"; N
62Ø SOUND 2ØØ, 1
     : INPUT "pmode"; Q
63Ø SOUND 2ØØ, 1
     : LINEINPUT "erase traces? "; B$
64Ø K=PEEK(1635Ø)
     : R=PEEK(16351)
     : S=PEEK(16352)/1ØØ
65Ø PMODE Q
     : SCREEN 1, Ø
66Ø PCLS 4
67Ø Z=125ØØ+(R/S*K*2)
     : V=2*K*N
68Ø FOR A=125ØØ TO Z STEP V
69Ø A$=INKEY$
7ØØ IF A$=" " THEN GOTO 6ØØ
```

```
71Ø IF A$="N" THEN GOTO 7Ø
72Ø IF B$="YES" THEN PCLS 4
73Ø FOR P=Ø TO (K*2)-4 STEP 2
     : LINE(PEEK(A+P), PEEK(A+P+1))-(PEEK(A+P+2),
     PEEK(A+P+3)), PSET
     : NEXT P
74Ø LINE(PEEK(A), PEEK(A+1))-(PEEK(A+2), PEEK(A+3)),
     PSET
     : NEXT A
75Ø IF B$="YES" THEN GOTO 79Ø
76Ø FOR B=125ØØ TO Z-1 STEP 2
     : LINE(PEEK(B), PEEK(B+1))-(PEEK(B+(K*2)),
     PEEK(B+(K*2)+1)), PSET
     : NEXT B
77Ø SCREEN 1, 1
78Ø FOR DLAY=1 TO 1ØØØ
     : NEXT DLAY
79Ø PCLS 4
8ØØ GOTO 67Ø
```



DRAW FOR 4K

David B. Shapiro
135 Appley Avenue Libertyville, IL 60048

This program is to show all the Extended Color BASIC owners that they are not the only ones that can do impressive graphics.

The first program loads a ML routine into the computer memory. It must be run before the draw program.

```
1Ø '  ML ROUTINE FOR DRAW PROGRAM
2Ø '
3Ø '  DAVID SHAPIRO 1/6/82
4Ø '
5Ø '  LOAD AND RUN THIS PROGRAM
6Ø '  BEFORE DRAW PROGRAM
7Ø '  ML ROUTINE WILL CLEAR
8Ø '  GRAPHICS SCREEN
9Ø CLEAR 2Ø, 2559
1ØØ FOR I=32Ø TO 332
11Ø READ K
     : POKE I, K
     : NEXT
12Ø '  321 CONTAINS CODE
13Ø DATA 134, Ø, 142
14Ø '  START ADDR.
15Ø DATA 1Ø, Ø
16Ø DATA 167, 128, 14Ø
17Ø '  END ADDR.
18Ø DATA 16, Ø
19Ø DATA 37, 249, 57
2ØØ POKE 157, 1
     : POKE 158, 64
```

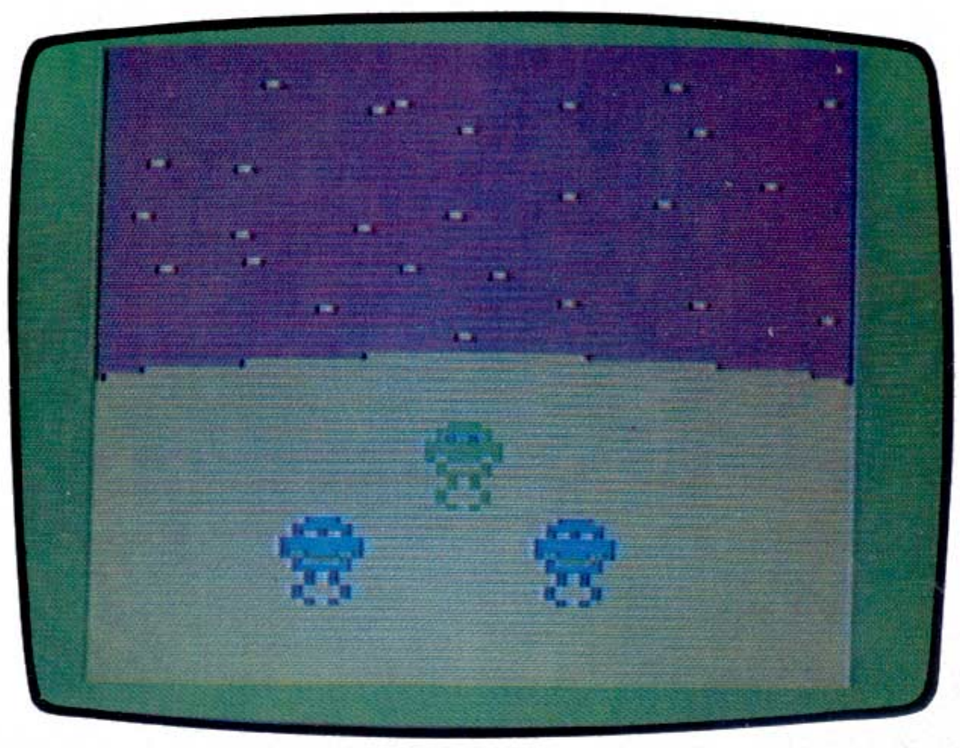
The draw program uses a 64 x 49 four color graphics mode. (I found this mode by accident. It is not mentioned in "Getting Started with Color Basic.")

I am sorry about the condition of my program, but I had to save every byte I could.

After you run the program press ⟨**C**⟩. This will clear the screen. Your right joystick controls a flashing cursor which can be moved around the screen. Press ⟨**R**⟩ to get to the bottom of the screen. To set a graphics spot press 1, 2, or 3 for color or press zero (0) to reset a spot. If you want a block all one color, place the cursor in one corner, press 'B', then move it to the opposite corner and press 'B' again. Then enter color (0, 1, 2, or 3) then ⟨**ENTER**⟩. Press ⟨**C**⟩ to clear the screen and start over.

SKRN

```
1ØØ CLEAR 5, 2559
     : DIM T(3, 3)
     : FOR J=Ø TO 3
     : FOR I=Ø TO 3
     : READ T(I, J)
     : NEXT I, J
     : FOR I=65472 TO 6549Ø STEP 2
     : READ X
     : POKE X+I, Ø
     : NEXT
     : DATA 252, 243, 2Ø7, 63, 1, 4, 16, 64, 2, 8,
     32, 128, 3, 12, 48, 192, 1, 1, Ø, 1, Ø, 1, Ø, Ø,
     Ø, Ø
     : POKE 65314, 128
3ØØ GOSUB 2ØØØ
     : IF A$="" THEN 3ØØ ELSE IF A$="C" THEN EXEC
     : GOTO 3ØØ
     : ELSE IF A$="R" THEN GOSUB 12ØØ
     : GOTO 3ØØ
     : ELSE IF A$="B" THEN C=2
     : GOSUB 11ØØ
     : X1=X
     : Y1=Y
     : GOTO 4ØØ
     : ELSE C=VAL(A$)
     : IF C>3 THEN 3ØØ ELSE GOSUB 11ØØ
     : GOTO 3ØØ
4ØØ GOSUB 2ØØØ
     : IF A$<>"B" THEN 4ØØ ELSE X2=X
     : I=Y
     : INPUT C
     : FOR X=X1 TO X2 STEP SGN(X2-X1)
     : FOR Y=Y1 TO I STEP SGN(I-Y1)
     : GOSUB 11ØØ
     : NEXT Y, X
     : GOTO 3ØØ
11ØØ T1=INT(X/4)
     : T2=256Ø+T1+Y*16
     : T3=3-X+T1*4
     : POKE T2, T(T3, Ø) AND PEEK(T2)
     : IF C=Ø THEN RETURN ELSE POKE T2, T(T3, C) OR
     PEEK(T2)
     : RETURN
12ØØ YS=YS-32*SGN(YS-1)
     : RETURN
13ØØ T1=INT(X/4)
     : T2=256Ø+T1+Y*16
     : T3=3-X+T1*4
     : T4=PEEK(T2)
     : POKE T2, T(T3, 2) OR T4
     : FOR I=Ø TO 15
     : NEXT
     : POKE T2, T(T3, Ø) AND T4
     : FOR I=Ø TO 15
     : NEXT
     : POKE T2, T4
     : RETURN
2ØØØ X=JOYSTK(Ø)
     : Y=JOYSTK(1)+YS
     : GOSUB 13ØØ
     : A$=INKEY$
     : RETURN
```



Call or Visit the Radio Shack Computer Center Near You for More Information

ALABAMA

BIRMINGHAM 2428 Green Springs Hwy., (205) 945-0792
HUNTSVILLE 1400 N. Memorial Pkwy., (205) 536-1581
MOBILE 405 Bel-Air Blvd., (205) 471-1617
MONTGOMERY #24 Union Square S/C, (205) 271-1500

ARIZONA

PHOENIX 10233 Metro Pkwy. E., (602) 861-1124; 4301 N. 7th Ave., (602) 277-3031
SCOTTSDALE 2525 N. Scottsdale Rd., (602) 990-2241
TEMPE 83 E. Broadway, (602) 894-2065
TUCSON 5622 E. Broadway, (602) 748-0101

ARKANSAS

LITTLE ROCK Town & Country S/C, University & Asher, (501) 568-5694

CALIFORNIA

ANAHEIM 509 Katella, (714) 776-9540
BERKELEY 1922 Grove St., (415) 848-9170
BEVERLY HILLS 8500 Wilshire Blvd., (213) 659-8870
CANOGA PARK 8371 Topanga Canyon, (213) 347-9800
CARMICHAEL 6305 Fairoaks Blvd., (916) 484-6815
CHULA VISTA 1201 3rd Ave., (714) 420-3810
DOWNEY 8031 Florence Ave., (213) 927-7882
ESCONDIDO 347 W. Mission Ave., (714) 741-6032
FRESNO Princeton S/C, 2721 N. Blackstone Ave., (209) 225-5551
GLENDALE 236 N. Brand Blvd., (213) 246-9310
HAYWARD 20942 Mission Blvd., (415) 278-2888
LAKEWOOD 5830 Lakewood Blvd., (213) 920-9671
LA MESA 5346 Jackson Dr., (714) 460-3610
LONG BEACH 2119 Bellflower Blvd., (213) 597-3377
LOS ANGELES 740 S. Olive St., (213) 629-2455
MONTEREY 484 Washington St., (408) 375-8430
MOUNTAIN VIEW 1933 El Camino Real W., (415) 961-0542
OAKLAND 1733 Broadway, (415) 763-3183
PASADENA 575 S. Lake Ave., (213) 449-5424
RIVERSIDE 3844 La Sierra Ave., (714) 689-0340
SACRAMENTO 4749 J. St., (916) 454-3287
SAN BERNARDINO 764 Inland Center Dr., (714) 884-6871
SAN DIEGO 3062 Clairemont Dr., (714) 276-6050; 3902 El Cajon Blvd., (714) 280-0227
SAN FRANCISCO One Market Place, (415) 777-9810
SAN JOSE 1228 S. Bascom Ave., (408) 297-2603
SAN MATEO 3180 Campus Dr., (415) 573-8607
SANTA BARBARA 4141 State St. A-1, (805) 967-4538
SANTA FE SPRINGS 14138 East Firestone Blvd., (213) 921-0702
STOCKTON College Sq. S/C, 963 West March Lane, (209) 957-3676
TARZANA 18545 Ventura Blvd., (213) 343-1696
TORRANCE 3840 Sepulveda at Hawthorne, (213) 373-0306
VENTURA 4005 E. Main St., (805) 654-0196
WEST COVINA 2516 E. Workman St., (213) 915-5791

COLORADO

BOULDER Arapahoe Plaza, 3550 Arapahoe, (303) 443-7142
COLORADO SPRINGS 4341 N. Academy, (303) 593-7500
DENVER 8000 E. Quincy, (303) 770-1362
LAKEWOOD 2099 Wadsworth Blvd., (303) 232-6277

CONNECTICUT

EAST HAVEN 51 Frontage Rd., (203) 467-8864
FAIRFIELD 1196 Kings Hwy. & Rt. 1, (203) 255-6099
MANCHESTER 228 Spencer St., (203) 649-8210
NORWALK Rt. 7-345 Main Ave., (203) 846-3418
ORANGE Woolco S/C; 538 Boston Post Rd., (203) 795-1291
WATERBURY 105 Bank St., (203) 573-8800
WATERFORD 122 Boston Port Rd., (203) 443-0716
WEST HARTFORD 39 S. Main St., (203) 523-4283

DELAWARE

WILMINGTON 3847 Kirkwood Hwy., (302) 999-0193

DISTRICT OF COLUMBIA

WASHINGTON 1800 M St. NW., (202) 822-3933

FLORIDA

ALTAMONTE SPRINGS 766 B. East Altamonte Dr., (305) 339-1313
CLEARWATER 2460-D US 19 North, (813) 797-3223
HOLLYWOOD 429 S. State Rd. #7, (305) 966-4382
JACKSONVILLE 8252 Arlington Exprwy., (904) 725-2594
LAUDERDALE LAKES 4317-25 N. State Rd. 7, (305) 486-2240
MIAMI 9459 S. Dixie Hwy., (305) 667-2316; 1601 Biscayne Blvd., (305) 374-6433; 15 SE. 2nd Ave., (305) 374-7310; 20761 S. Dixie Hwy., (305) 238-2518
N. MIAMI BEACH The Promenade, 1777 N.E. 163rd St., (305) 940-6887
ORLANDO 1238 E. Colonial Dr., (305) 894-0570
ST. PETERSBURG 3451 66th St. N., (813) 381-2366
SARASOTA 5251 S. Tamiami Tr. (Hwy.41), (813) 923-4721
TALLAHASSEE 2529 S. Adams, (904) 222-4440
TAMPA 4555 W. Kennedy, (813) 879-7470; 1825 E. Fowler Ave., (813) 971-1130
W. PALM BEACH 2271-A Palm Beach Lakes Blvd., (305) 683-3100

GEORGIA

AUGUSTA 3435 Wrightsboro Rd., (404) 738-5998
ATLANTA 2108 Henderson Mill, (404) 939-9888; 49 W. Paces Ferry, (404) 231-9604; Akers Mill S/C, 2937 Cobb Parkway NW., (404) 955-5235; 113 Peachtree St., (404) 223-5904
COLLEGE PARK 5309 Old National Hwy., (404) 761-3055
DORAVILLE 5697 Buford Hwy., (404) 458-2691
SAVANNAH Chatham Plaza, 7805 Abercorn St., (912) 355-6074

IDAHO

BOISE 691 S. Capitol Blvd., (208) 344-5450

ILLINOIS

AURORA 890 North Lake St., (312) 844-2224
CHICAGO 4355 S. Archer Ave., (312) 376-7617; CNA Plaza, 309 S. Wabash, (312) 922-0536
ELMWOOD PARK 7212 W. Grand Ave., (312) 452-7500
FAIRVIEW HEIGHTS #4 Market Place, (618) 398-6410
HOMEWOOD/GLENWOOD 329 Glenwood Lansing, (312) 758-0440
LaGRANGE One S. LaGrange Rd., (312) 482-3484
LIBERTYVILLE 1350 S. Milwaukee Ave., (312) 367-8230
LOMBARD 4 Yorktown Center, (312) 629-5350
NILES 8349 Golf Rd., (312) 470-0670
OAK LAWN 4815 W. 95th St., (312) 425-9130
PEORIA 4125 N. Sheridan Rd., (309) 685-7056
ROCKFORD North Town S/C, 3600 N. Main St., (815) 282-1001
SCHAUMBURG 651 Mall Dr., (312) 884-8600
SPRINGFIELD Sherwood Plaza, 2478 Wabash, (217) 787-3066

INDIANA

EVANSVILLE 431 Diamond Ave., (812) 426-1715
FT. WAYNE 747 Northcrest S/C, (219) 482-9547
GRIFFITH 208 W. Ridge Rd., (219) 838-3000
INDIANAPOLIS 6242 E. 82nd St., Castleton Plz., (317) 849-6896; Speedway Plaza, 6129 B Crawfordsville, (317) 244-2221; 10013 E. Washington St., (317) 898-4887
TERRE HAUTE 3460 U.S. Hwy. 41 S., (812) 234-3212

IOWA

DAVENPORT 616 E. Kimberly Rd., (319) 386-3457
DES MOINES 7660 Hickman Rd., Sherwood Forest S/M, (515) 270-0193

KANSAS

OVERLAND PARK 8619 W. 95th, (913) 642-1301
TOPEKA White Lakes Plaza, West Tower, 3715 Plaza Dr., (913) 267-6420
WICHITA 2732 Blvd. Plaza S/C, (316) 681-1212

KENTUCKY

FLORENCE 7727 Mall Rd., (606) 371-2811
LEXINGTON 2909 Richmond Rd., (606) 269-7321
LOUISVILLE 2900 Taylorsville Rd., (502) 459-9901

LOUISIANA

BATON ROUGE 7007 Florida Blvd., (504) 928-5260
METAIRIE 3750 Veterans Hwy., (504) 454-3681
NEW ORLEANS 327 St. Charles Ave., (504) 523-6408
SHREVEPORT 1545 Line Ave., (318) 221-5125

MAINE

BANGOR Maine Square, (207) 945-6491

MARYLAND

BALTIMORE 7942 Belair Rd., Putty Hill Plaza, (301) 882-9583; 115 N. Charles St. at Lexington, (301) 539-7251
CATONSVILLE One Mile West S/C, 6600 B Balt. Nat'l. Pike, (301) 788-3277
FREDERICK Shoppers World, Rt. 40W, (301) 695-8440
NEW CARROLLTON-LANHAM 7949 Annapolis Rd., (301) 459-8030
PASADENA 8120 Ritchie Hwy., (301) 544-2352
ROCKVILLE Congressional Plaza, 1673 Rockville Pike, (301) 984-0424
SALISBURY Shoppers World S/C, Rt. 50, (301) 546-9223

MASSACHUSETTS

BOSTON 730 Commonwealth Ave., (617) 739-1704; 111 Summer St., (617) 542-0361
BRAINTREE South Shore Plaza, 250 Granite St., (617) 848-9290
BROCKTON 675 Belmont, (617) 583-2270
BURLINGTON Crossroads Plaza, Rt. 3 S., (617) 229-2850
CAMBRIDGE Harvard Square, 28 Boylston St., (617) 354-7694
CHESTNUT HILL 200 Boylston St., (617) 969-2031
NATICK 1400 Worcester Rd., (617) 875-8721
SAUGUS 343 Broadway, (617) 233-4985
SPRINGFIELD 1985 Main St., Northgate Plz., (413) 732-4745
WORCESTER Lincoln Plaza, (617) 852-8844

MICHIGAN

BIRMINGHAM 3620 W. Maple Rd., (313) 647-2151
DETROIT DWNTN 1559 Woodward Ave., (313) 961-6855
FLINT G3298 Miller Rd., Yorkshire Plaza, (313) 732-2530
GRAND RAPIDS 3142 28th St. SE., (616) 957-2040
KALAMAZOO 25 Kalamazoo Center, (616) 343-0780
LANSING 2519 S. Cedar St., (517) 372-1120
LIVONIA 33470 W. 7 Mile Rd., (313) 476-6800
ROSEVILLE 31873 Gratiot Ave., (313) 296-6210
SOUTHFIELD 17651 West 12 Mile Rd., (313) 569-1027
TROY Oakland Plaza, 322 John R. Rd., (313) 585-3900

MINNESOTA

BLOOMINGTON 10566 France Ave. S., (612) 884-1641
GOLDEN VALLEY Golden Valley S/C, 8016 Olson Memorial Hwy., (612) 542-8471
ST. PAUL 6th & Wabasha, (612) 291-7230

MISSISSIPPI

JACKSON 979 Ellis Ave., (601) 352-5001

MISSOURI

FLORISSANT 47 Florissant Oaks S/C, (314) 921-7722
INDEPENDENCE 1325 S. Noland Rd., (816) 254-3701
KANSAS CITY 4025 N. Oak Trafficway, (816) 455-3381
ST. ANN 10472 St. Charles Rock Rd., (314) 428-1400
SPRINGFIELD 2684 S. Glenstone, (417) 883-4320

NEBRASKA

OMAHA 3006 Dodge St., (402) 346-4003

NEVADA

LAS VEGAS Commercial Center, 953 E. Sahara #31-B, (702) 731-3956
RENO 3328 Kietzke Lane, (702) 826-6327

NEW HAMPSHIRE

MANCHESTER Hampshire Plaza, 1000 Elm St., (603) 625-4040

NEW JERSEY

BRIDGEWATER 1472 U.S. Highway 22 East, (201) 469-3232
E. BRUNSWICK 595 A Rt. 18, (201) 238-7142
E. HANOVER Rt. 10, Hanover Plaza, (201) 884-1200
LAWRENCEVILLE Rt. 1 & Texas Ave., (609) 771-8113
NEWARK 595 Broad, (201) 622-1339
PARAMUS 175 Rt. 17 S., (201) 262-1920
SPRINGFIELD Rt. #22 Center Isle, (201) 467-9827
VOORHEES 35 Eagle Plaza, (609) 346-0600

NEW MEXICO

ALBUQUERQUE 2108 San Mateo NE., (505) 265-9587

NEW YORK

ALBANY Shoppers Pk., Wolf Rd., (518) 459-5527
BAYSHORE 1751 Sunrise Hwy., (516) 666-1800
BETHPAGE 422 N. Wantagh Ave., (516) 822-6403
BUFFALO 839 Niagara Falls Blvd., (716) 837-2590
FRESH MEADOWS 187-12 Horace Harding Exp., (212) 454-1075
JOHNSON CITY Giant Shopping Center, Harry L. Drive, (607) 729-6312
MELVILLE TSS Mall, Rt. 110, (516) 673-4646
NEW ROCHELLE 242 North Ave., (914) 636-0700
NEW YORK 385 Fifth Ave., (212) 889-1345; 139 E. 42nd St., (212) 953-6053; 19 W. 23rd St., (212) 691-1861; 347 Madison Ave., (212) 867-8650
REGO PARK 97-77 Queens Blvd., (212) 897-5200
ROCHESTER 3000 Winton Rd., (716) 244-6400
SCARSDALE 365 Central Park Ave., (914) 472-2520
SPRING VALLEY White House Center, 88 W. Rt. 59, (914) 425-2828
STATEN ISLAND 2409 Richmond Ave., (212) 698-3100
SYRACUSE 2544 Erie Blvd., (315) 446-3017
UTICA Riverside Mall, (315) 735-1933

NORTH CAROLINA

CHARLOTTE 3732 Independence Blvd., (704) 535-6320
GREENSBORO 3718 High Point Rd., (919) 294-5529
RALEIGH Townridge Sq., Hwy. 70 W., (919) 781-9380
WINSTON-SALEM 629 Peters Creek Pkwy., (919) 722-0030

OHIO

AKRON Fairlawn Plaza, 2727 W. Market St., (216) 836-9303
CANTON 5248 Dressler Rd. NW., (216) 494-7230; Mellet Plaza, 3826 W. Tuscarawas, (216) 478-1878
CENTERVILLE 2026 Miamisburg-Centerville Rd., (513) 435-5167
CINCINNATI 9725 Montgomery, (513) 793-8688
CLEVELAND 419 Euclid (Dwntwn), (216) 575-0800; 27561 Euciid Ave., (216) 289-6823
COLUMBUS 862 S. Hamilton, Great Eastern S/C, (614) 864-2806; The Patio Shop. Ctr., 4661 Karl Rd., (614) 888-8227; 400 N. High St., (614) 464-2781
DAYTON Northwest Plaza, 3279 West Siebenthaler, (513) 277-6500
NORTH OLMSTED Great Northern S/C, (216) 734-2255
TOLEDO 5844 W. Central Ave., (419) 531-5797
YOUNGSTOWN Union Square Plaza, 2543 Belmont Ave., (216) 744-4541

OKLAHOMA

OKLAHOMA CITY 4732 SE 29th St., (405) 670-4561; Springdale S/C, 4469 NW 50th, (405) 943-8712
TULSA 7218 & 7220 E. 41st St., (918) 663-2190

OREGON

EUGENE 390 Coburg Rd., (503) 687-0082
PORTLAND 7463 SW Barbur Blvd., (503) 246-1157; 9131 SE Powell, (503) 777-2223

PENNSYLVANIA

ALLENTOWN Crest Plaza S/C, Cedar Crest Blvd. US 22, (215) 395-7155
BALA CYNWYD 67 E. City Line Ave., (215) 668-9950
ERIE 5755 Peach St., (814) 868-5541
HARRISBURG Union Deposit Mall, Union Deposit Rd. #17, (717) 564-6753
LANCASTER Park City Plaza, US 30, (717) 393-5817
MONROEVILLE 3828 Wm. Penn. Hwy., (412) 823-3400
MONTGOMERYVILLE Airport Sq., Rt. 309, (215) 362-1200
PHILADELPHIA 7542 Castor Ave., (215) 342-2217 1002 Chestnut St., (215) 923-3080
PITTSBURGH 5775 Baptist Rd., Hills Plaza, (412) 831-9694; 303 Smithfield St., (412) 391-3150
SCRANTON 206 Meadow Ave., (717) 348-1801

RHODE ISLAND

E. PROVIDENCE 850 Waterman Ave., (401) 438-2860
PROVIDENCE 177 Union St., (401) 272-0201

SOUTH CAROLINA

COLUMBIA Old Sears Bldg., 1001 Harden St., (803) 799-2065
GREENVILLE N. Hills S/C, (803) 292-1835
N. CHARLESTON 5900 Rivers Ave., (803) 747-5580

TENNESSEE

CHATTANOOGA 636 Northgate Mall, (615) 870-1366
JOHNSON CITY Peerless Center, (615) 282-6829
KNOXVILLE Cedar Bluff S/C, 9123 Executive Park Dr., (615) 690-0520
MEMPHIS 4665 American Way, (901) 795-4963; 1997 Union Ave., (901) 278-7935
NASHVILLE 2115 Franklin Pike, (615) 298-5484; Rivergate Plaza, (615) 859-3414

TEXAS

ARLINGTON 2500 E. Randol Mill, Suite 113, (817) 274-3127
AUSTIN 8764 E. Research Blvd., (512) 459-4238
BEAUMONT 5330 Eastex Frwy., (713) 898-7000
CORPUS CHRISTI 1711 S. Staple St., (512) 887-8901
DALLAS 15340 Dallas Pkwy., Suite 1100, (214) 934-0275; 2930 W. Northwest Hwy., (214) 350-4144; 1517 Main St., (214) 760-8601; 2588 Royal Ln., (214) 484-9947
EL PASO 9515 Gateway West, (915) 594-8211
FT. WORTH 15 One Tandy Center, (817) 335-7198; 2801 Alta Mere, (817) 738-0251
HOUSTON 211C-FM 1960, (713) 444-7006; 10543 Gulf Fwy., (713) 943-9310; 5900 North Fwy., (713) 699-1932; 6813 SW Fwy., (713) 777-7907; 809 Dallas St., (713) 651-3002
HURST Northeast Mall, (817) 284-1518
LUBBOCK 3625 34th St., (806) 793-1467
ODESSA 1613 "A" East 8th Street, (915) 334-8355
RICHARDSON Fleetwood Sq. S/C, 202 W. Campbell Rd., (214) 669-1494
SAN ANTONIO 6018 West Ave., (512) 344-8792; 4249 Centergate, (512) 657-3958

UTAH

MURRAY 6051 S. State Ave., (801) 268-8978
SALT LAKE CITY 415 5th Ave., (801) 322-4893

VIRGINIA

ALEXANDRIA 4527 Duke St., Westend S/C, (703) 370-9000
FAIRFAX Westfair Center, 11027 Lee Hwy., (703) 273-6500
NEWPORT NEWS Newmarket South Shop. Ctr., (804) 245-8203
NORFOLK 5731 Poplar Hall Dr., (804) 461-0798
RICHMOND Willow Lawn S/C, 1617 Willow Lawn Dr., (804) 282-3453; 7728 Midlothian Turnpike, (804) 272-8803
ROANOKE Franklin Bldg., 3561 Franklin Rd. S.W., (703) 342-6335

WASHINGTON

BELLEVUE Crossroads Mall, North East 8th & 156 St., (206) 644-1804
FEDERAL WAY 33505 Pacific Hwy. South, (206) 838-6830
SEATTLE 18405 Aurora Ave. N., (206) 542-6184
SPOKANE 7702 N. Division, (509) 484-7000; E. 12412 Sprague, (509) 922-2800
TACOMA 7030 S. Sprague, (206) 473-7333
TUKWILA 15425 53rd Ave. S., (206) 248-3710
YAKIMA 1111 N. First St., (509) 248-9667

WEST VIRGINIA

HUNTINGTON 2701½ 5th Ave., (304) 523-3527

WISCONSIN

MADISON 57 West Towne Mall, (608) 833-6130
MILWAUKEE 6450 N. 76th St., (414) 353-6790
WEST ALLIS 2717 South 108th St., (414) 327-4240

Radio Shack®

TRS-80 Microcomputer News
P.O. Box 2910
Fort Worth, Texas 76113-2910

FIRST CLASS U.S. POSTAGE PAID DALLAS, TX PERMIT 2180

ADDRESS CHANGE

☐ **Remove from list** ☐ **Change as shown**

Please detach address label and mail to address shown above