The Book of Interact Programs

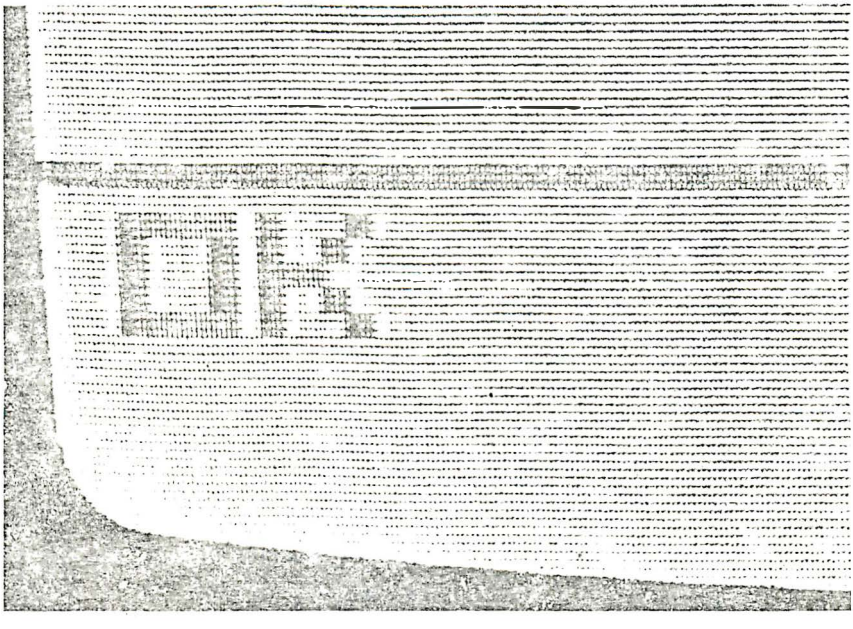
TABLE OF CONTENTS

INTRODUCTION

Your Interact is a true computer. As such, you may program it by typing in properly-stated instructions in the BASIC programming language. You can make your computer solve problems, play music, make sounds, draw pictures and patterns, and much, much more.

Interact gives you not one, but two BASIC languages. EDU-BASIC has fewer instructions, and is thus both easier to learn and a little more limited than its "big brother," Level II BASIC. We've chosen some of our favorite programs from selections written in both BASICs. Many of them were written by Interact owners like you. We've also added an appendix of slightly more advanced *tricks of the trade* for those of you who'll want to go beyond these sample programs.

If you've never programmed before, please read the section called "About Programming — The Agony and the Ecstasy." It's a light, humorous note about the joys and frustrations of working with computers. Then pick a program whose description tickles your fancy. Try it out — we're sure you'll be delighted with the results.



ABOUT PROGRAMMING — THE AGONY AND THE ECSTASY

Normally, he was a very even-tempered, pleasant man. A professional programmer, he'd been working with computers both large and small for several years. Today — as most days — he could be found sitting at his console, typing in commands and examining results intently and carefully. But today he was different, a little agitated, his fingers rapping the keys with increasing anger. Suddenly he sprang from his seat, slammed his clenched fist against the console and shrieked, "Idiot machine! Picky, Picky, Picky!" and stormed from the room.

And you know, he was right. Computers ARE picky. Although they can do many wonderful, complex things at fantastic speeds, they're in some ways very stupid. They can only do exactly as they're told, and have a much more limited vocabulary than you or I. One little typo, one comma out of place, and the "idiot machine" starts spewing error messages that sometimes make even the most experienced, cool programmer see red like an angry bull.

What we're trying to illustrate is simply this — don't be surprised or disappointed if your program doesn't run right the first time. It won't. They never do, even for the experts. That's "Murphy's Law" of programming. Another version of this famous law, popular among programmers, is "The program line you're most certain is right is the one that's causing the problem."

Yet, we certainly don't mean to imply that programming is all agony and no ecstasy. There's a deep satisfaction to be had when you type "RUN" and everything finally happens the way you intended. It's much like the satisfaction you feel when you "start from scratch" and build a model, harvest your garden or cook a fine gourmet meal. And in this modern world, there's much to be said for learning more about how computers work. Programming is a great way to do that.

What can you do to up your share of ecstasy and minimize the agony? First, read. Read the manuals and guides that come with your system. Get books from the library or bookstore. The more you read, the fewer mistakes you'll make. Second, be careful. Watch what you're doing. Try to type in instructions exactly as they should be. Remember, "Picky, picky, picky." Last but not least, be patient. Whatever the error, it can be corrected. And if you get to the end of your rope, quit for a bit. Then call us in Ann Arbor, Michigan at 313-973-0120 during normal business hours. We're here to help, and we know what it's like. Now — on to your first program! Good luck, and have fun!

ENTERING, CORRECTING AND RUNNING PROGRAMS

ENTERING PROGRAMS

Programs are composed of ordered instructions to the computer. Order is determined by the number you assign at the beginning of each program line. Lines are always arranged in ascending numerical order, regardless of the order in which you type them in.

Always type a "NEW" command before you start on a program. That will clear "garbage" or old programs which might interfere with what you're about to do. Then, type in the program lines exactly as they appear in the book. To see your program lines, use the LIST command. You might want to read about "CONTROL/S" and "CONTROL/Q" too. They help you start and stop the listing as needed. These commands and controls work the same way in both versions of the BASIC language.

CORRECTING MISTAKES

Murphy's Law of Programming says you'll make them. But how do you fix those inevitable errors? It's easy! Read on ...

Your Interact doesn't actually read what you type in until you press the "CR" key. If you hit the wrong key before you type a "CR", just backspace to erase your mistake. Then continue typing in the line from that point. What if you've already typed a "CR?" Don't panic, now. Just type the whole line in again.

What about correcting, adding and deleting numbered program statements? Use the techniques above. Don't worry if the line is a numbered program line. Type in the corrected line, number and all, and your Interact obliges by putting the line in the right place. You can also use this technique to add lines. Just pick an unused number that puts the line in the right place. Type in the number and commands for that line. Type a "LIST" command to see how your computer put things back into numerical order. To erase a line completely, type the number and a "CR." List your program, and the line has vanished.

Some letters or numbers are easily confused. For example, you might read a zero (0) when the program listing really shows the letter O ("oh"). Unfortunately, even though they may look almost the same, your Interact definitely knows the difference. To help you out we have provided the chart below which shows pairs that might be confused. If you are in doubt, or if a program line that looks right causes an error, we suggest you compare the letters and numbers in the line with those in the chart below.

0 = zero	O = the capital letter "oh"
1 = the number one	I = the capital letter "I"
2 = the number two	Z = the capital letter "zee"
5 = the number five	S = the capital letter "ess"

RUNNING YOUR PROGRAM

All done entering statements? Just type "RUN" and a "CR." The first time through you'll probably get an error message. Look up the meaning of the message in the error code guide of the manual for the version of BASIC you're using. Most error messages will give you a line number where the problem occurred. EDU-BASIC shows you the line and prints a "?" before the first character that confused it. When you find your mistake, use the techniques described above to correct it. Then type another "RUN" command. Does it work now? *Hooray for you!*

SAVING PROGRAMS ON TAPE

After you get a program working properly, you'll probably want to copy it onto a cassette tape. That way you can load it back in with a simple command next time you want to run it. Save several of these programs and you'll have a ready-made arsenal of BASIC programs to demonstrate your new-found expertise to others.

Some advice on saving programs: First, always save two copies. That way if something goes wrong with one copy, you'll have a backup. Next, ALWAYS label both the tape and the cassette insert card so you'll know for sure what programs are where. If you want to put several programs on one tape for more compact storage, first make copies of each program on separate tapes. Then load them in one at a time and save them on your master tape. Assign names to each program. Don't rewind the master tape in between saves, or new entries will write over old. If you want to guarantee that you don't accidentally write over a program, punch out the two small plastic tabs on the spine of the tape after the program is recorded. You'll be unable to record over the tape again unless you cover the holes with cellophane tape.

You may record programs on any good quality, non-metallic cassette tape of 60-minute length or less. If you use a regular audio tape, be sure not to rewind it completely before saving a program. Sometimes the leader is too long and part of your program gets lost on the leader. To avoid this problem, use Interact Data Tapes. One Data Tape can hold about 250,000 characters of information.

To save and reload EDU-BASIC programs, read about the "SAVE" and "LOAD" commands in the EDU-BASIC Program Guide. For Level II BASIC, see the descriptions of "CSAVE" and "CLOAD" in the Users Manual.

EDU-BASIC

LIVE DEMONSTRATION

A live demonstration is an effective way to introduce a friend to your programming accomplishments and to Interact's BASIC language. There's no substitute for sitting by your Interact and TV set to show how it works. In this live demo, you give commands which the computer performs immediately.

After this demo, show some of your programs from cassette tape. They'll look even more impressive and complex after you've stepped through this simple demo. We suggest that you don't try to write a program while a friend is watching. Remember "Murphy's Law of Programming." There are generally not enough "results" for the time spent typing in and correcting statements to make a good show. It's best to prepare these masterpieces in private and unveil only the finished product.

The demo should take about five minutes at most. Don't go too much longer. There's no need to "explain everything" the first time. Besides, your audience will be eager to see your other programs, and some of the Interact tapes, too. Be sure to press the "CR" key after each instruction described below. Go gettum, Programmer!

PROCEDURE FOR LIVE DEMO IN EDU-BASIC

1. Load the EDU-BASIC tape in the usual manner. While the tape is loading, explain that the information is being read into the RAM memory of your computer. This information will then permit you to instruct your computer in a special language called BASIC. (That stands for "Beginner's All-Purpose Symbolic Instruction Code," in case you're interested.) The sounds he hears are the audible equivalents of those digital instructions on the tape. Tell him that if you were to insert a regular music cassette into the machine at this point, he'd hear the music through the TV speaker.

2. When you get the "OK" from BASIC, show how your Interact can solve arithmetic problems quickly and easily. Type:

```
PRINT 3 + 4
```

and the computer responds with the answer, 7. Then type:

```
P. 4*125
```

to show how "PRINT" can be abbreviated to "P." The computer prints "500."

3. Now show how your unit can play musical notes with the TONE statement. Type:

```
TONE(168,168)
TONE(110,200)
TONE(80, 300)
```

These commands cause your Interact to play a C-G-C musical triad.

4. Now let's put some personal information on the screen. Your guest will be most amused to see his name printed on the screen. Do the following:

1. Type CLEAR to erase the screen.
2. Type P.;P."GEORGE";P. to print your friend's name surrounded by blank lines.

5. Now explain that you can put a dot on the screen at a given position in a specific color. All images are made of patterns of such dots. First type a CLEAR command to erase the screen. Then type:

```
PLOT(80, 25, 1)
```

This puts a white dot over the "1" in the "PLOT" command, up 25 positions (pixels) from the bottom of the screen and over 80 pixels from the left edge.

6. Lines, of course, are just a series of dots, and can be drawn by a "FOR...NEXT" loop. A loop is a series of instructions performed several times in a row - in this case, 112 times! Type:

```
FORX = 1TO112;PLOT(X, 30, 2);NEXTX
```

This will draw a blue line over your "FOR...NEXT" command.

Now you are ready to show your friend your favorite EDU-BASIC programs. Show him some of the programs from this book that you have typed in previously and saved on tape. Also, show him some of Interact's pre-written programs, depending on his area of interest:

INTEREST	SUGGESTED TAPES
Action Games	Trailblazers™, Showdown™, Volleyball, Breakthrough™
Strategy Games	Backgammon, Chess, Concentration
Music	Music Maestro™
Computer Art	Compute-A-Color™
Education	Touchdown, Add 'Em Up™, Hangman
Finance	Financial Library I and II, Checkbook Balancer

HELLO FROM INTERACT

This simple program prints a message on the screen repeatedly in a different color each time, while a siren wails in the background. Can you figure out why the screen goes blank every now and again? (Answer below.)

PROGRAM LISTING	
5	CLEAR
10	COLOR (7, 4, 0, 0)
20	SOUND (0, 270)
30	FOR I = 1 TO 25
40	P. '' HELLO FROM''
50	P. ''INTERACT''
60	P.
70	COLOR (7, I, 0, 0)
80	NEXT I

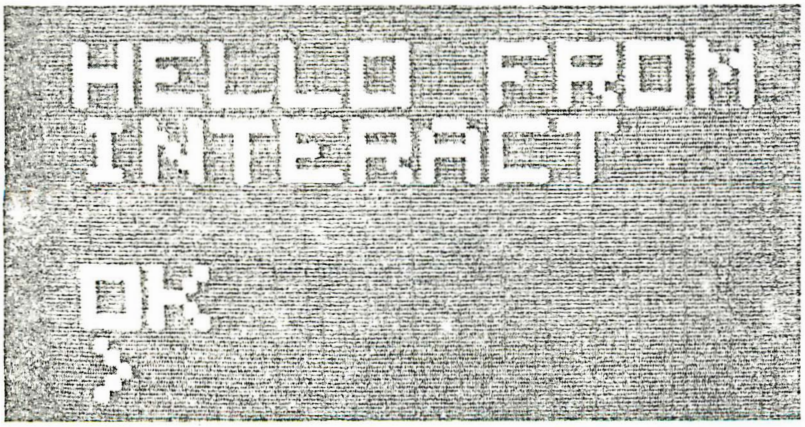
EXERCISES

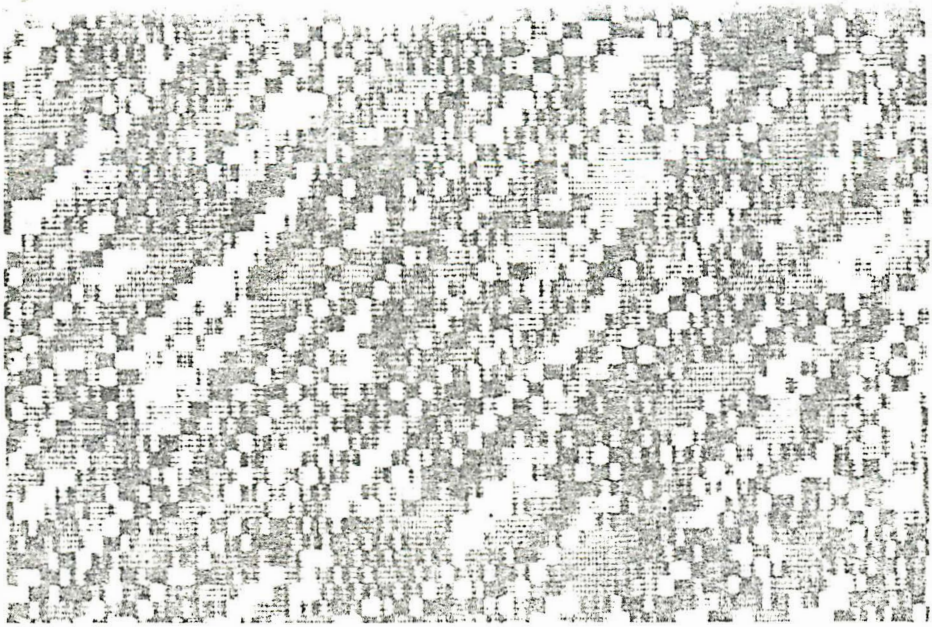
Delete line 20 by typing ''20'' and a ''CR.'' Then put in a new line 32:

32 SOUND (0, 20*I)

This change causes a different sound effect to be produced each time the message is printed on the screen.

Why does the screen go blank every so often? Every eighth time, color parameter number 2, denoted by the ''I'' in line 70, gets set to an even multiple of 7, the code for white. Since the background is also white, you don't see the messages when they are printed white-on-white. Next time through the loop, the color changes to black and the messages reappear.





WE ARE NOT ALONE

Colored communication markers dot the night sky in this program complete with alien blips in a seemingly endless pattern. Use of the random number generator to control the colors and tones is illustrated here. Stop the program by holding down the CONTROL key while you type a "C."

PROGRAM LISTING

```
  5  CLEAR
 10  X = RND (112)
 20  Y = RND (77)
 30  Z = RND (3)
 40  A = RND (7)
 50  B = RND (7)
 60  C = RND (7)
 70  COLOR (0, A, B, C)
 80  FOR I = 1 TO A
 90  PLOT (X + I, Y-I, Z)
100  TONE (X, 1000/X)
110  NEXT I
120  GOTO 10
```

ACKNOWLEDGEMENT:

Our thanks to Interact owner Patrick Wilson for this program.

ELECTRONIC SNOWFLAKES

This program fills your screen with electronic snowflakes. Add a touch of humor as did the Interact owner who gave us this program, and print out Robert Frost's "On a Snowy Evening" against the snowing background. Notice line 610 where a ";" is used to separate two commands on the same line.

PROGRAM LISTING

```
10  CLEAR
20  GOSUB 600
500 GOTO 10
600 FOR N=0 TO 50
610 X=RND(110);Y=RND(77)
620 PLOT (X, Y, 1)
630 NEXT N
640 RETURN
```

EXERCISES

1. Produce a colored snowfall by plotting the snowflakes in three colors instead of one color. First try a "COLOR" command at line 15 to set up new colors for your snowfall display. Then use the statements:

```
619 A = RND (4)
620 PLOT (X, Y, A)
```

to produce multi-colored snow.

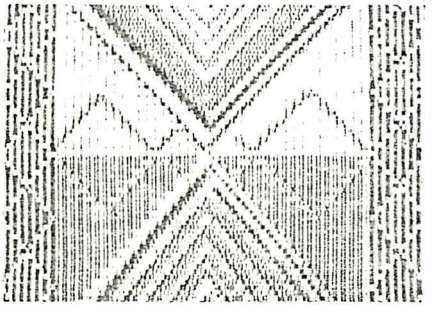
```
10 Clear
20 X = RND(110) ; Y = RND (77)
30 PLOT (X, Y, 3)
40 GOTO 20
```

Random dots of red on screen!

LEVEL II
BASIC

LIVE DEMONSTRATION

A "Live Demonstration" is an effective way to introduce a friend to your programming accomplishments and Interact's BASIC. There's no substitute for sitting by your Interact and your TV set to show how it works. The demonstration is very simply. It produces instant results using simple BASIC statements typed in Direct Mode, that is *WITHOUT* line-numbers.



After this demonstration show some of your previously programmed selections from cassette tape. They will look even more impressive and complex after you have stepped through this simple demonstration. We suggest that you don't try to write and/or debug a program while a friend is watching as there are not generally enough "results" for the time spent. It's best to prepare those masterpieces in private and unveil only the finished product.

This demonstration should take approximately five minutes. Don't go too much longer as you need not "explain everything" the first time. Besides he will want to see your programs and some of the Interact games or other application software too.

PROCEDURE

Load the Level II BASIC tape into your Interact in the standard manner. While it is loading, explain that the information on the tape is being read into the memory (RAM) of your computer. This information will then permit you to instruct the computer in a special language called BASIC. The sounds that he hears are the audible equivalents of those magnetically recorded instructions on the tape. Tell him that if you were to insert a regular music cassette into the machine at this point, he would hear the music through the TV speaker.

When you get the OK, type:

NEW

This instructs the computer that you are to type in a new program.

Explain that your Interact allows you to program "in color." The COLOR statements allow you to specify a background color and three other colors for lettering or drawing. The colors are referenced by number. Type:

COLOR 7, 2, 4, 1

The color change is dramatic and you'll have his attention right away.

RAYS

This simple, short program produces a set of multi-colored rays which slowly expand from the lower-left corner of your TV screen. Its simplicity illustrates that even short graphic programs on the Interact can produce unexpected, pleasing results. RAYS takes about two minutes to fill your screen with multi-color graphics. The program will run continuously. Stop it by pressing the RESET button, then type an "R".

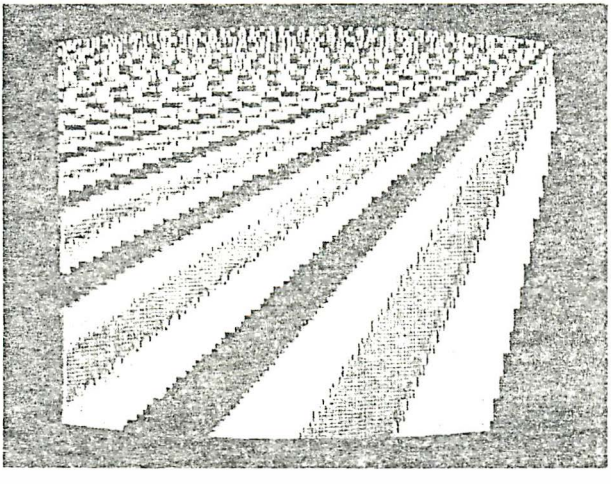
PROGRAM LISTING	
10	CLS
20	COLOR 0, 1, 2, 4
30	FORA = 5TO105
40	FORB = 2TO77
50	C = A*2/B*2
60	PLOTA, B, C
70	NEXT:NEXT
80	GOTO10

EXERCISE

Change line 50 in the program to read:

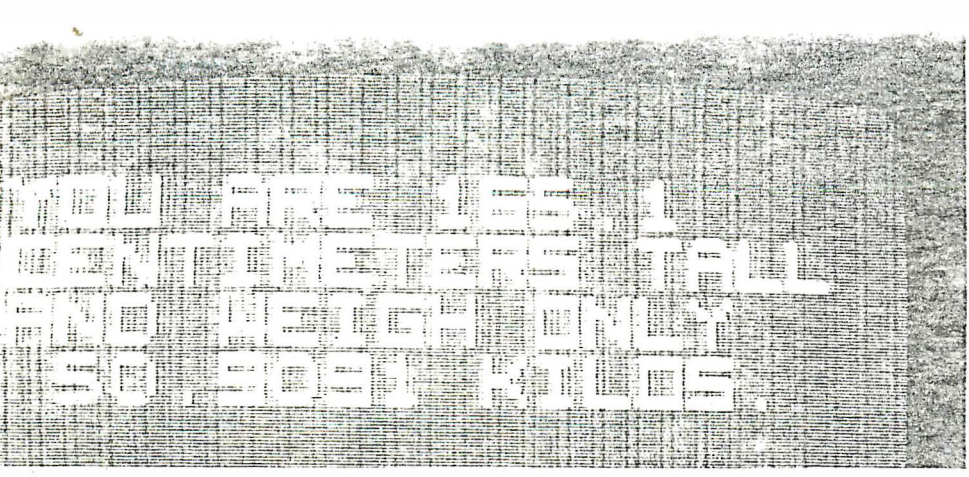
50 C = A*B/100

This change produces rays in a parabolic pattern that resembles a hand-tied rag rug.



ACKNOWLEDGEMENT

Our thanks to Interact owner Tom Matulevich for the RAYS program.



HOW BIG ARE YOU IN METRIC?

This program illustrates the ease in BASIC of asking the user for two pieces of data and then producing two output results from simple arithmetic calculations.

PROGRAM LISTING

```
 10  CLS
 20  PRINT "HOW BIG ARE"
 30  PRINT "YOU IN METRIC?"
 40  PRINT
 50  PRINT "YOUR HEIGHT"
 60  INPUT "IN INCHES";IN
 70  PRINT
 80  PRINT "YOUR WEIGHT"
 90  INPUT "IN POUNDS";P
100  CM = 2.54*IN
110  KG = P/2.2
120  CLS
130  PRINT "YOU ARE";CM
140  PRINT "CENTIMETERS TALL"
150  PRINT "AND WEIGH ONLY"
160  PRINTKG;"KILOS."
170  PRINT:PRINT:PRINT
180  GOTO 40
```

EXERCISES

1. Ask for the name of the user and store it in the string variable N$. Print out his name when you give him his metric height and weight.

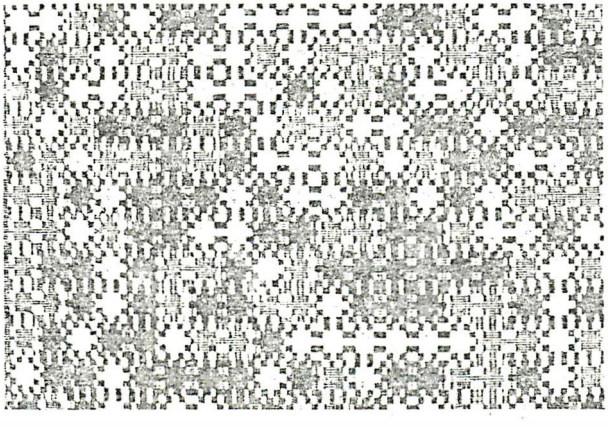
2. Accumulate the height and weight of several persons and then report the averages.

MIDDLE EASTERN TAPESTRY

This program fills the screen with an ever-changing mosaic which dramatizes the use of color on your Interact computer. The initial POKE in line 10 enables the subsequent POKES in lines 100 and 110 to set the color registers in memory locations 6144 and 4096.

PROGRAM LISTING

```
10  POKE 19215, 25
20  CLS
30  COLOR 0, 1, 2, 4
40  FOR P = 0 TO 7
50  FOR A = 0 TO 113 STEP 5
60  FOR B = 6 TO 77 STEP 5
70  OUTPUT "O", A, B, C
80  OUTPUT "*", A, B, C + 1
90  C = 3*RND(1) + 1
100 POKE6144, A
110 POKE4096, P
120 NEXT:NEXT
130 NEXT
140 GOTO 40
```



EXERCISE

Modify lines 50 and 60 so that a clear border of equal size is placed around the changing pattern.

ACKNOWLEDGEMENT

We thank Tom Matulevich for submitting "Middle Eastern Tapestry."

CANADIAN FLAG

Of primary interest in this program is the creation of the maple leaf. The leaf was first drawn by hand on narrow-ruled graph paper by coloring in appropriate squares. Then, each colored square was treated as a 1 by 1 pixel dot and put on the screen at the proper location with a PLOT command. Note that the symmetry of the leaf is exploited. The right half of the leaf is drawn first (in lines 140-240) after M is set to 1 in line 130. Then, the same PLOT commands in lines 140-240 are used to draw the left half by setting M = -1 in line 260. The flag remains undisturbed on the screen until you press any key on the keyboard.

PROGRAM LISTING

```
 10  CLS
 20  COLOR 0, 1, 7, 2
 30  REM — DRAW TRI-COLOR
 40  FORY = 72TO26STEP-5
 50  C = 1
 60  FORX = 10TO90STEP5
 70  IFX > 29THENC = 2
 80  IFX > 70THENC = 1
 90  OUTPUT "O", X, Y, C:OUTPUT "*", X, Y, C
100  NEXTX:NEXTY
110  REM — DRAW MAPLE LEAF
120  FORY = 38TO61:PLOT52, Y, 1:NEXTY
130  M = 1
140  FORY = 43TO59:PLOTM + 52, Y, 1:NEXTY
150  FORY = 43TO57:PLOT52 + M*2, Y, 1:NEXTY
160  FORY = 43TO51:PLOT52 + M*3, Y, 1:NEXTY
170  FORY = 43TO50:PLOT52 + M*4, Y, 1:NEXTY
180  FORY = 45TO51:PLOT52 + M*5, Y, 1:NEXTY
190  FORY = 46TO52:PLOT52 + M*6, Y, 1:NEXTY
200  FORY = 49TO52:PLOT52 + M*7, Y, 1:NEXTY
210  FORY = 50TO52:PLOT52 + M*8, Y, 1:NEXTY
220  FORY = 54TO55:PLOT52 + (M*3), Y, 1:NEXTY
230  FORY = 55TO56:PLOT52 + M*4, Y, 1:NEXTY
240  PLOT52 + (M*7), 47, 1
250  IFM = -1GOTO270
260  M = -1:GOTO140
270  FOR X = 46TO58:PLOTX, 43, 1:NEXT X
280  A$ = INSTR$(1)
```

EXERCISES

1. Convert this program into one that draws the French flag, the Italian flag.

2. Note that many flags resemble each other, especially the "tri-color" flag common to many nations. Build a program that randomly produces several flags, which differ only in color.

3. Write a "Name the Flag" game where the user must enter the country name and capital city represented by each of your flags. Keep score for two players.

AMERICAN FLAG

This program produces the ol' stars and stripes in blazing red, white and blue, filling your screen with patriotism. What is unique about this program is the method used to "draw" the flag. Instead of using PLOT commands — which produce dots one pixel wide and tall — an asterisk (*) and the letter "O" are superimposed using the OUTPUT command. This results in a 5 by 5 pixel "block". By using these "blocks" in OUTPUT loops rather than "dots" in PLOT loops, the flag is drawn with thick lines, so it comes out bigger, faster. The flag remains undisturbed until you press any key on the keyboard.

PROGRAM LISTING

```
 10  CLS:C = 2
 20  COLOR 0, 1, 7, 4
 30  FORY = 72TO42STEP-5
 40  IFC = 2THENC = 1:GOTO 60
 50  C = 2
 60  X = 50:GOSUB 240
 70  NEXT Y
 80  FORY = 37TO8STEP-5
 90  IFC = 2THENC = 1:GOTO 110
100  C = 2
110  X = 5:GOSUB 240
120  NEXT Y
130  C = 3
140  FORY = 72TO42STEP-5
150  FORX = 5TO45STEP 5
160  OUTPUT "O", X, Y, C
170  OUTPUT "*", X, Y, C
180  NEXT:NEXT
190  FORY = 68TO42STEP-6
200  FORX = 9TO48STEP4
210  PLOT X, Y, 2
220  NEXT:NEXT
230  GOTO 290
240  FORXX = XTO105STEP5
250  OUTPUT "O", XX, Y, C
260  OUTPUT "*", XX, Y, C
270  NEXT
280  RETURN
290  A$ = INSTR$(1)
```

EXERCISES

1. Play the first few bars of the "Star Bangled Banner" or other patriotic song after the flag is drawn by using an appropriate sequence of TONE commands at the end of the program. Be sure to delete line 290 before adding the music.

2. Recreate "Fort Sumpter" with SOUND commands used to make "bombs burst in air."

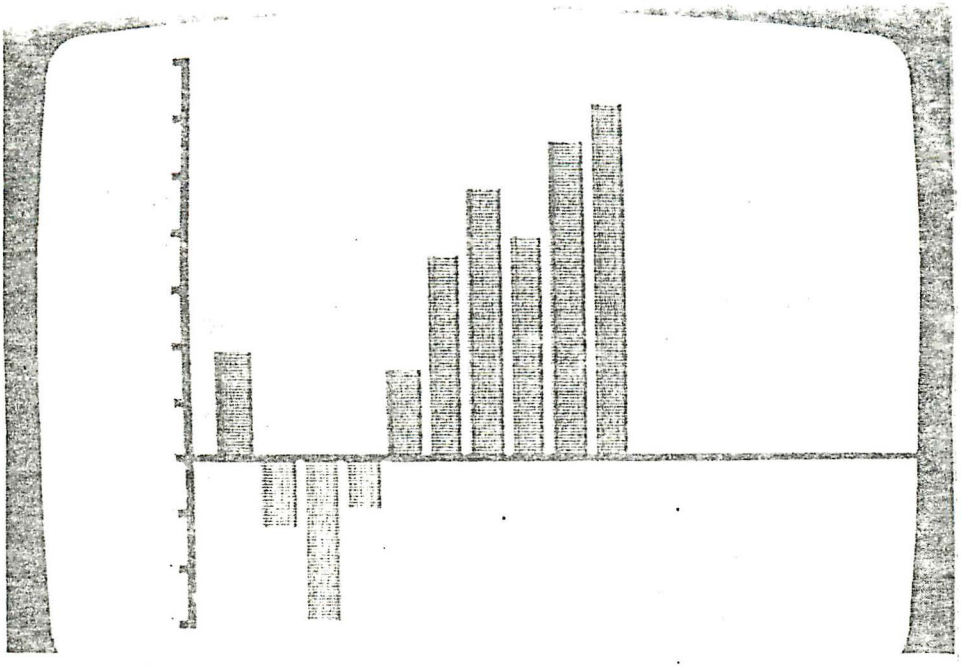
3. Change the way the stars are drawn in lines 190-220 so that only 13 stars are drawn in the original pattern of a circle.

BAR GRAPH

This program represents numerical data as a bar graph with green bars for positive values and red bars for negative values. Notice how the program uses your data values to determine the scale of the graph and the proper height of each bar. The scaling makes this same program useful for plotting any group of numbers attractively — large ones or small ones — because axis units are not preset for a certain range of numbers. Instead, units are computed to fit each set of values after you type them in.

PROGRAM LISTING

```
 10  DIM Y(15)
 20  CLS:COLOR 7, 1, 2, 0
 30  PRINT "HOW MANY DATA"
 40  INPUT "POINTS";N
 50  PRINT
 60  FOR I = 1 TO N
 70  PRINT "DATA";I
 80  INPUT Y(I)
 90  NEXT I
100  REM—SET MAX AND MIN
110  MI = 99999999
120  MA = -99999999
130  FOR I = 1 TO N
140  IF Y(I) < = MI THEN MI = Y(I)
150  IF Y(I) > = MA THEN MA = Y(I)
160  NEXT I
170  REM—PRODUCE PLOT
180  CLS
190  FOR I = 10 TO 70:PLOT 20, I, 3:NEXT I
200  FOR I = 10 TO 70 STEP 6:PLOT 19, I, 3:NEXT I
210  M1 = 1.1*MA
220  M2 = MI-.1*ABS(MI)
230  IF M2 < 0 GOTO 510
240  FOR X = 20 TO 112
250  PLOT X, 10, 3:NEXT X
260  D = M1/60
270  FOR I = 1 TO N
280  C = 2
290  IF M2 > = 0 GOTO 380
300  IF Y(I) > = 0 GOTO 350
310  C = 1
320  YH = XA
330  YL = XA + Y(I)/D
340  GOTO 400
350  YH = Y(I)/D + XA
360  YL = XA
370  GOTO 400
380  YL = 11
390  YH = Y(I)/D + 10
```



```
400  GOSUB440
410  NEXTI
420  IFM2<0THENFORI=20TO112:PLOTI,XA,3:NEXT
430  A$=INSTR$(1):GOTO 20
440  X=I*5+*19
450  IFYH<=YLTHENRETURN
460  FORA=XTOX+3
470  FORJ=YLTOYH
480  PLOTA,J,C
490  NEXT:NEXT:RETURN
510  D=(ABS(M1)+ABS(M2))/60
520  XA=10+60*ABS(M2)/(ABS(M2)+ABS(M1))
530  GOTO 270
```

EXERCISES

1. Check the value N supplied by the user to see that it doesn't exceed the dimension value, 15, of the Y array. If it is out of range, ask the user to re-input N.

2. Compute and show the vertical scaling of the plot on the top of the screen.

3. Add the capability for the user to display and make changes to individual data values before the graph is drawn, without having to enter all of the data again.

4. Plot two variables with adjacent, different-colored bars.

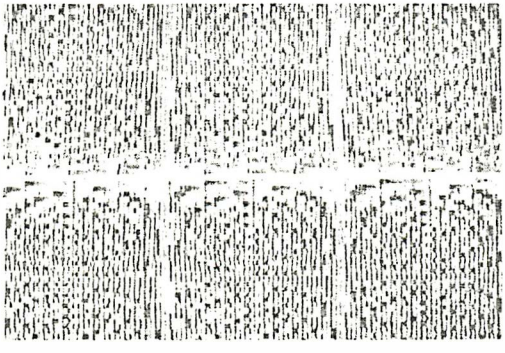
5. Allow the user to choose colors for the bars from a menu you produce for him on the screen.

GYRATING HONEYCOMBS

This program produces graphic images which show the color resolution of your Interact at its best. Six honeycomb patterns are drawn on the screen by the six PLOT statements in lines 190-240. The color roll in lines 260-290 at the end of the program has a surprising side effect — it not only changes the colors, but causes an apparent change in the pattern itself. You must see Gyrating Honeycombs to appreciate it. We liked it so well we had postcards made of the resulting picture.

PROGRAM LISTING

```
 10  POKE 19215, 25
 20  CLS
 30  GOTO 120
 40  REM—RESET COLORS
 50  A = 8*C2 + C0
 60  B = 8*C3 + C1
 70  POKE 4096, A
 80  POKE 6144, B
 90  RETURN
100  REM—PAUSE LOOP
110  FOR I1 = 1TOP:NEXT:RETURN
120  REM—SET COLORS
130  C0 = 0:C1 = 1:C2 = 2:C3 = 4
140  GOSUB 50
150  REM—DRAW 6 PATTERNS
160  FOR A = 3TO36
170  FOR B = 1TO38
180  C = A*B/9.5
190  PLOT A, B, C
200  PLOT A + 36, B, C
210  PLOT A + 72, B, C
220  PLOT A, B + 38, C
230  PLOT A + 36, B + 38, C
240  PLOT A + 72, B + 38, C
250  NEXT:NEXT
260  REM—COLOR ROLL
270  FOR I = 1 TO 384
280  T = C1:C1 = C2:C2 = C3:C3 = T
290  GOSUB 50
300  P = 500:GOSUB 110
310  NEXT
320  GOTO 20
```



EXERCISE

Speed up the color change by modifying the value of P in line 300.

ACKNOWLEDGEMENT

Boy, that Tom Matulevich is sure prolific, isn't he? He wrote this one, too!

HAVE A HAPPY DAY

This program draws a smiling cartoon face and prints a cheery message at the top of the picture. The picture remains frozen on the screen until you type any key on the keyboard.

PROGRAM LISTING

```
 10  CLS
 20  COLOR 7, 0, 2, 1
 30  H = 0
 40  REM – DRAW FACE
 50  FOR X = 25 TO 79
 60  P = (X-25)/3
 70  Y = (P*P-18*P)/4
 80  PLOT X, 35 + Y, 1
 90  PLOT X, 35-Y, 1
100  REM – SET SMILE
110  M = 47 + Y
120  IF X > 38 THEN PLOT X,M,1
130  IF X > 67 THEN PLOT X,M,0
140  NEXT X
150  REM – DRAW EYES
160  PLOT 46, 43, 1
170  PLOT 46, 42, 1
180  PLOT 58, 43, 1
190  PLOT 58, 42, 1
200  REM – GROW HAIR
210  FOR G = 1 TO 10
220  IF H = 0 GOTO 350
230  FOR X = 35 TO 75 STEP 10
240  P = (X-25)/3
250  Y = -(P*P-18*P)/4 + Z
260  IF X = 35 THEN K = -2*G
270  IF X = 55 THEN K = 0
280  IF X = 65 THEN K = G
290  IF X = 45 THEN K = -G
300  IF X = 75 THEN K = G*2
310  PLOT X + K, Y + 35, 3
320  NEXT X
330  Z = Z + 1
340  NEXT G
350  OUTPUT "HAVE A NICE DAY", 10, 63, 3
360  A$ = INSTR$(1)
```

EXERCISES

1. Change line 30 to set H = 1 and watch the face grow funny red pigtails.

2. Try converting this program to EDU-BASIC.

3. Change the face from a smiling one to a sad face by changing line 110 to read "110M = 12-Y". Change the message output in line 350 to make it appropriate to a frowning face.

4. Imbed this program in a larger program which drills a child in arithmetic. Reward them with the appropriate face at the end of a drill session.

DIGITAL CLOCK

This program turns your TV into a colorful digital clock which keeps the time in hours, minutes and seconds. The timing is done by peeking at the computer's internal, self-incrementing "clock", called CK. This internal clock is incremented each sixtieth of a second. The program begins by asking you to "set the clock" by typing in local time in hours and minutes, then specifying AM or PM. The various subroutines called in the program keep track of each digit of the time, outputting new digits as time marches on. To keep a clock handy when you're watching TV, just set the clock program running, then change the channel to your desired program and flip the TV switchbox. When you want to know the time, simply dial to channel 3 and flip the TV switchbox back to "computer". As long as the red light on the console stays on, the clock will continue to run.

NOTE: This program uses a small window setting in line 50 to keep your "clock" neat and attractive. During testing of the program you may want to omit line 50 so you can see error messages, list program lines and so on. When you know your clock works, type in line 50 to set the window to 12 when you run the clock.

PROGRAM LISTING

```
  5 CLS
 10 COLOR 7, 1, 2, 4
 20 POKE 19215, 25
 30 CK = 24559
 40 X = 57:Y = 50
 50 WINDOW 12
 60 OUTPUT "LOCAL TIME", 27, Y + 10, 3
 70 OUTPUT ":    :", X-24, Y, 1
 80 REM—SET CLOCK
 90 INPUT "TIME HOURS";TH
100 H10 = INT(TH/10):IF H10 = 0 THEN GOTO 120
110 OUTPUT H10, X-42, Y, 1
120 H = TH-(H10*10)
130 OUTPUT H, X-36, Y, 1
140 INPUT "TIME MIN.";TM
150 M10 = INT(TM/10)
160 OUTPUT M10, X-24, Y, 1
170 M = TM-(M10*10)
180 OUTPUT M, X-18, Y, 1
190 OUTPUT "00", X, Y, 1
192 B$ = "PM"
200 INPUT "AM OR PM";B$
210 AP = 1
220 IF B$ = "AM" THEN AP = 0
230 OUTPUT B$,X + 18, Y, 1
240 WINDOW 12
250 INPUT "CR TO START";I
260 REM—CLOCK LOOP
270 POKE CK,0:WINDOW 77
280 A = PEEK(CK)
290 IF A > 58 THEN POKE CK, 0:GOTO 320
300 GOTO 280
```

```
320  OUTPUT S, X, Y, 0
330  S = S + 1
340  IF S > 9 THEN S = 0:GOSUB 370
350  OUTPUT S, X, Y, 1
360  GOTO 280
370  OUTPUT S10,X-6, Y, 0
380  S10 = S10 + 1
390  IF S10 > 5 THEN S10 = 0:GOSUB 420
400  OUTPUT S10,X-6, Y, 1
410  RETURN
420  OUTPUT M,X-18, Y, 0
430  M = M + 1
432  A = PEEK(CK):A = A-5:POKE CK, A
440  IF M > 9 THEN M = 0:GOSUB 470
450  OUTPUT M, X-18, Y, 1
460  RETURN
470  OUTPUT M10, X-24, Y, 0
480  M10 = M10 + 1
490  IF M10 > 5 THEN M10 = 0:GOSUB 520
500  OUTPUT M10, X-24, Y, 1
510  RETURN
520  OUTPUT H, X-36, Y, 0
530  H = H + 1
540  IF H > 9 THEN H = 0:GOSUB 580
550  IF H = 3 AND H10 = 1 THEN H = 1:GOSUB 580
560  OUTPUT H, X-36, Y, 1
570  RETURN
580  OUTPUT H10, X-42, Y, 0
590  H10 = H10 + 1
600  IF H10 = 2 THEN H10 = 0:GOSUB 630:RETURN
610  OUTPUT H10, X-42, Y, 1
620  RETURN
630  OUTPUT B$,X + 18, Y, 0
640  AP = AP + 1
650  IF AP = 2 THEN AP = 0
660  IF AP = 0 THEN B$ = "AM"
670  IF AP = 1 THEN B$ = "PM"
680  OUTPUT B$,X + 18, Y, 1
690  RETURN
```

EXERCISES

1. Add a tick-tock or other sound for each second increment.

2. Add chimes on the hour.

3. Turn this program into an alarm clock where the program asks for the alarm time and alarm message that is to be flashed on the screen. Use a siren sound as the alarm bell.

4. Extend this program to a World Time Clock with a list of major cities and local times scrolling on the bottom half of the screen. Use color to differentiate AM and PM in these world cities.

SOUND SEARCH

Sound effects are produced in Level II BASIC by using the SOUND command. Two numbers follow the sound command. The combination of these numbers determine which of the 100's of possible sound effects is produced. Some combinations of numbers produce no sound at all. This program permits you to test various combinations of numbers by using your left controller. The first number in a sound command can vary between 1 and 7. When you push the controller lever to the left, this first number is varied. The second number can vary between 0 and 32767. Push the lever to the right, and the second number is incremented by 50. Each time a SOUND command is tested, the numbers being used are printed on the screen. By proceeding slowly, you can test virtually the entire spectrum of possible sound effects. Keep a pencil and paper handy so you can record your favorites for use in other programs.

PROGRAM LISTING

```
10  CLS
20  COLOR 7, 1, 2, 4
30  OUTPUT "SOUND SEARCH", 23, 60, 1
40  OUTPUT "USE LEFT", 35, 50, 2
50  OUTPUT "CONTROLLER", 29, 44, 2
60  WINDOW 18
70  FOR J = 0 TO 7
80  FOR K = 0 TO 32767 STEP 50
90  SOUND J, K
100 PRINT "SOUND";J;"", "";K
110 IF JOY (0) = 0 GOTO 110
120 IF JOY(0) = 2 GOTO 150
130 IF JOY(0) = 1 GOTO 160
140 GOTO 110
150 NEXT K
160 NEXT J
170 GOTO 70
```

EXERCISES

1. Modify the program to run without the controller. Use the INSTR$ function to make different keys vary the two SOUND numbers up and down.

2. Extend the program so that depressing the left FIRE button causes the current SOUND numbers to be saved on tape for later replay and review.

3. Build a program that generates series of sound effects. Make a sound, then pause, then produce the next sound.

HERE'S A LIST OF SOME OF OUR FAVORITE SOUNDS

0, 24844	Siren
7, 4096	Stops all sounds.
1, 8770	Static
0, 16416	Fog horn
5, 10	Idling jet
3, 16	Medium pitched buzz
3, 32	Lower pitched buzz
3, 48	Very low pitched buzz
3, 56	Fast motor
3, 4	High pitched beep pulsing rapidly.
3, 8	High pitched beep pulsing medium.
3, 12	High pitched beep pulsing slowly.
3, 20	Medium pitched beep pulsing rapidly.
3, 24	Medium pitched beep pulsing slowly.
3, 16436	Motor idling roughly.
3, 16440	Motor idling more smoothly.
3, 16416	Low, noisy sound.

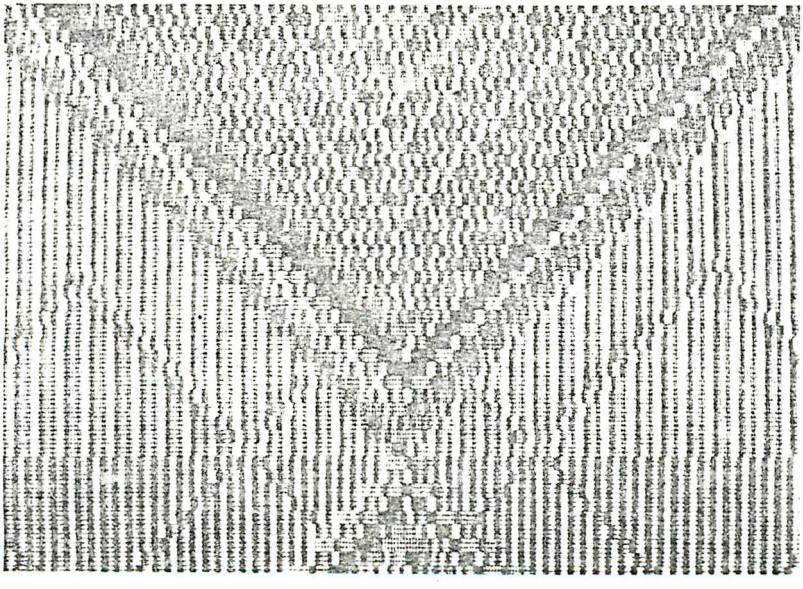
LEVEL II BASIC NOTEBOOK

This section contains some of our most-requested "tricks of the trade" for use with Level II BASIC. Rather than presenting whole programs we present concepts, subroutines, and partial examples. These are meant to be used as part of larger programs of your own design, perhaps with minor changes to customize things to fit your needs precisely.

We've given you a library of spectacular graphic effects here -- blinking objects, flashing text, partial screen wipes and color rolls. Use them to enhance your own games, write programs, or for just plain fun.

The subroutines for saving text on tape and reading it back greatly enhance the uses you'll find for Level II BASIC. Use these subroutines to create tapes of names, addresses and phone numbers. Or how about a tape of your favorite recipes? Or try this -- make a program that reads and write a list of your household valuables, including a description of the item, identifying marks and approximate value. Put the data tape in your safety deposit box. If anything happens you'll have a safe, permanent record to use for insurance purposes. These are just a few of the many uses for these subroutines. What others can you imagine?

Finally, we've shown you how to use your cassette deck to control an audio tape with your program. Now you can write programs that run with your favorite music playing the background. Or programs that use your voice to "speak" to your child, rewarding him for good performance. Just use the methodology described here to set the colors for your program. At the same time, you've turned on your tape! Insert your audio tape, press READ, and out comes the audio through the TV speaker while the program continues to run.



SAVING TEXT ON TAPE

The CLOAD* and CSAVE* commands in Level II BASIC read and write numerical information to and from cassette tape. There are no BASIC commands to read and write text (string) data. Thus statements like:

```
CSAVE* N$
CLOAD* A$
```

are not permitted, and their use results in an "?FC ERROR" message. To save strings you must first convert them to numbers, then save the numbers on tape using the CSAVE* command. After you read these numbers from tape with a CLOAD*, you'll convert them back to the original text string.

How do you convert from letters to numbers and back again? It's easy. Each letter already has a pre-assigned number that represents it. The ASC function in BASIC retrieves this number for any given letter. The CHR$ function converts the number back to the letter. Using these functions in appropriate statements, you can create short subroutines easily modified for use in any program that saves or loads textual data.

EXAMPLE

Let's assume that a program must keep track of 10 names. Each name has at most 8 characters. We'll use the following variable names in our example:

N$	String array containing the names.
WA	Numeric "working array" used to hold numeric conversions for one name.
L$	String "working variable" that holds each letter of a name temporarily.

First, we must allocate adequate string space -- 150 bytes in this example:

```
10  CLEAR(150)
```

Next, we must dimension our arrays. Since we have 10 names to store in N$, we'll set it to 10. Since each name has up to 8 characters, and since we need one extra slot in which to record the name's length, our WA array gets 9 slots:

```
20  DIM N$(10), WA(9)
```

Next, we'll need lines that will allow us to type in the 10 names:

```
25  PRINT "ENTER 10 NAMES:"
30  FOR I = 1 TO 10
40  INPUT N$(I)
50  NEXT I
```

Now, we'll add the main program lines to call the subroutine that converts a name in the N$ array to the nine numbers in the WA array, then saves the numbers on tape:

```
 60  PRINT "PRESS READ"
 65  PRINT "AND WRITE."
 70  PRINT "HIT CR WHEN"
 75  PRINT "READY."
 80  A$ = INSTR$(1)
 90  FOR I = 1 TO 10
100  GOSUB 300
110  CSAVE* WA
120  NEXT I
```

Next, we'll allow for rewinding the tape and setting the deck to read back the number arrays. Then, we'll read each number array, call the subroutine that converts numbers back to a name, and then print out each name:

```
130  PRINT "PRESS REWIND."
140  PRINT "PRESS READ."
150  PRINT "WHEN DONE"
160  FOR I = 1 TO 10
170  CLOAD* WA
180  GOSUB 400
190  PRINT N$(I)
200  NEXT I
210  STOP
```

Last — but certainly not least! — we need the subroutines that do the actual conversions. The first one goes from letters to numbers for one name:

SUBROUTINE LINE	EXPLANATION
300 WA(1) = LEN(N$(I)) + 1	Measure length of name, add 1, store result in first WA slot.
310 IF WA(1) = 1 THEN RETURN	If WA(1) = 1, the string was empty. That is, the name has no letters so there's nothing to convert. (It's good to allow for *all* possibilities in any program).
320 FOR J = 2 TO WA(1)	Start a loop — go through once for each letter in the name.
330 L$ = MID$(N$(I), J-1, J)	Pull off next letter to convert. Store it in L$.
340 WA(J) = ASC(L$)	Convert isolated letter to number, store number in WA array.
350 NEXT J	Go do the next letter.
360 RETURN	All done. Go back to main program and save converted number array WA.

The next subroutine goes from numbers back to a name:

SUBROUTINE LINE	EXPLANATION
400 N$(I) = ""	Initialize array slot for name we're about to convert.
410 IF WA(1) = 1 THEN RETURN	If WA(1) = 1 then there's nothing to convert so go back to main program. (Remember that WA(1) contains the length of the name, plus 1.)
420 FOR J = 2 TO WA(1)	Start a loop -- go through once for each letter as specified by name length in WA(1).
430 N$(I) = N$(I) + CHR$(WA(J))	Convert next number in WA array to letter, add it on to letters already converted, if any.
440 NEXT J	Go convert the next number, if any.
450 RETURN	All done. N$(I) now contains a complete name in letters, so go back to main program.

This logic is easily modified to suit any need. Got more names? Use a bigger dimension on N$ to reserve enough slots. Also make a corresponding change to the ending value for I in the loops that call the conversion subroutines. You may also need more than 150 bytes of string space (you'll know you need more if you get an "?OS ERROR" message). In that case, change the CLEAR command to use a higher number. Got more than 8 characters in a string? Just add slots to the WA array, setting its dimension to a maximum string length, plus 1. Also, change the ending values for J on the loops inside the conversion subroutines.

TAPE MOTOR CONTROL

Using the right combination of POKE commands you can activate the tape tape motor to play an audio track while your program is running. Use this capability to provide background music for "multi-media" programs. You can create exciting combinations of music and graphics, for example. Or use the capability to provide pre-recorded verbal program instructions or other messages. It's easy to do.

First, record your audio track. You may use any pre-recorded, commercially produced cassettes, too. After you load your program, insert the audio tape. Give REWIND command if you like, to make sure the tape is positioned properly. Press the READ button, then start your program. When the proper POKE command is encountered, your Interact will begin to play your audio tape through the TV speaker. While the tape motor is on, you may also use the FFWD, REWIND and WRITE buttons as described below.

If you are trying to synchronize several separate messages with a program, here are some helpful hints on how to proceed:

1. Use a stop watch to time each audio segment you'll play.

2. Experiment for the following "pause loop:"

   FOR P=1 TO N:NEXT

   This loop does nothing but slow down your program. By varying the parameter "N" you can cause pauses of various lengths anywhere in your program. Use these pauses as necessary to keep the program in sync with your audio track.

3. Run the program with the audio. If the program gets ahead of the tape, use pause loops as necessary to delay the program. Or, add statements as described below to turn off the motor for awhile, then start it up again with another POKE after the program catches up.

The tape motor is controlled by a switch stored at location 4096. This same location also contains the switches which set 2 of the 4 display colors. A COLOR command always turns off the tape motor switch. Changing just the switch can affect the colors. Therefore, we use some special POKE commands that set colors AND turn the tape on, simultaneously. Don't want a color change at the time you want the motor? That's easy — just "set" the same colors you already have, using POKEs as described below.

1. Before you do any other POKEs, put the following command in your program:

   POKE 19215, 25

   This initializes the POKE routine.

2. Decide what 4 colors you want to use. Look up the codes that identify your chosen colors using the list on the Reference Card or in your Users Manual. To keep the example general, we've called our chosen colors C0, C1, C2 and C3. C0 is the background color. C1 is used to display program line numbers. C2 is not used unless you specifically request it in an OUTPUT or PLOT command. C3 is used to display program statements output from PRINT commands and input from the keyboard.

For example, suppose you want white background, red line numbers, blue program statements, and green for the "spare" color. That's equivalent to the COLOR command:

COLOR 7, 1, 2, 4

with C0 = 7, C1 = 1, C2 = 2 and C3 = 4. But if you give that COLOR command, the tape motor won't be turned on.

3. Compute parameters to set colors and activate the motor as follows:

   A = 64 + 8*C2 + C0

   B = 8*C3 + C1

   In our example:

   A = 64 + 8*2 + 7 = 87
   B = 8*4 + 1 = 33

4. Simultaneously turn on the tape motor and set the colors with these statements:

   POKE 4096, A — Turns on the motor and sets the background and "spare" colors.

   POKE 6144, B — Sets colors for program lines and line numbers.

   In our example:

   POKE 4096, 87 — Turns on motor, sets background to white, "spare" to green.

   POKE 6144, 33 — Sets color 1 to red, color 3 to blue, causing line numbers to appear in red and program lines in blue.

The tape motor stays on until you issue a COLOR command, or until a CSAVE, CLOAD, CSAVE* or CLOAD* command you issue completes. Your program continues to run independent of the tape motor. While the motor is on, the tape control buttons determine what results:

READ button	Plays tape through TV speaker.
REWIND button	Rewinds tape.
FFWD button	Fast-forwards tape.
STOP/EJECT button	Stops and/or ejects tape. (Motor is still on, but nothing happens.)
WRITE button	Erases tape.
READ and WRITE buttons together	Erases tape.