Radio Shack®

Cat. No. 26-3195

TRS-80® Color Computer & MC-10 Programs

By William Barden, Jr.

- Mathematics
- Educational
- Word Processing
- Graphics
- Practical
- Household
- Games
- Miscellaneous

BASIC Programs for the MC-10 and Color Computer

by
William Barden, Jr.

FIRST EDITION PRINTING 9 8 7 6 5 4 3 2

International Standard Book Number: Library of Congress Book Number:

Printed in USA

Preface

Want to drill yourself in multiplication, German, French, or Spanish? How about displaying the calendar month for October, 1965? Need a program that will encode messages into your own secret code? How about a way to print out an amortization schedule for your home mortgage? Want to play a musical rendition of a popular song? Interested in learning Morse code?

These are only a few of the BASIC programs that are included in this book. There are many more, ranging from tutorial programs for younger readers to financial programs for businessmen and from games to utility software. The main purpose of this book is to provide you with as many useful BASIC programs as possible. We've tried to accomplish more than just a clever screen display (although we have that too).

All of the programs in the book are designed to run on both the MC-10 and Color Computer. This is possible because the MC-10 BASIC commands are a subset of the Color Computer commands, for the most part. And the MC-10 BASIC commands are very powerful, so we're not limited by using only MC-10 commands; most Color Computer BASIC commands are found in the MC-10 as well.

All of the programs were written especially for the MC-10 and Color Computer and not adapted from programs for other computers. You won't have the situation you find with many programs in magazines and some books in which the program "blows up" after you've spent 10 minutes entering BASIC lines.

For each program, we've included a brief description of what the program does. Next is a section called "How to Use This Program" which is exactly that – it contains step-by-step, foolproof instructions on how to operate the program from the ground up. If you've never done anything with a computer before, this section will tell you exactly how to proceed. This section contains all of the screen displays that the program uses, so you'll be able to see exactly how to go about using the program.

The next section is called "Some Background on the Program." Many of the programs in the book have an interesting background. The Word Jumble Program, for example, is based on a method that English bell ringers use to ring church bells! In many cases, this section will explain the "algorithm" or approach that the program uses to solve the problem. Knowing the algorithm will help you understand the functioning of the program.

The next section is called "How the Program Works." In truth, this isn't a detailed description of each BASIC line in the program, but is a more general description of how the program goes about solving the problem. It supplements material on BASIC programming that you'll find in other Radio Shack publications.

The last section "Special Notes" lists the peculiarities of the program. In some cases, you shouldn't enter commas when entering data for the program from the keyboard – this section will remind you of that. It'll tell you about other "format" problems, notes on special system configurations that are required, and the like.

The last portion of each program description is the program itself. This is a reproduction of the actual program listing, not a typeset version. This is important, because it means that if you enter the program exactly as listed, it'll run the first time. In some cases there are minor format differences between the MC-10 and Color Computer, but these will be clearly listed (the MC-10, for example, uses "LPRINT" but the Color Computer uses "PRINT#-2;" to print on the system line printer, for example).

We had a lot of fun doing these programs and we hope that you will find many of them useful for your own MC-10 and Color Computer applications.

A great deal of credit for program development goes to my wife Janet who spent many hours developing and testing the programs in this book.

William Barden, Jr.

In memory of Joe Deutsch, a true computer buff

Table of Contents

How to Use the Programs in this Book

Follow these steps to foolproof use of the programs:

1. **Use the listing for each program** as is.
2. Enter the program exactly as it is shown, using the same number of blanks and special characters as you see on the listing. In some cases there will be minor format differences between the MC-10 and Color Computer - watch for these. In a few cases you'll require an Extended BASIC Color Computer system for the program. On the listings: A zero is "squared off" while an alphabetic "oh" is rounded.
3. After you have entered the program into your MC-10 or Color Computer, immediately save the program on cassette or diskette.
4. Get a listing of the program on your system line printer, if you have one (use the LLIST command). If you don't have a line printer, list the program on the screen by the LIST command. (Press SHIFT, followed by @ to stop the listing at any point.) Compare the listing with the listing in the book to make certain the program is correct. Remember, one incorrect character may cause the program to "blow up."
5. If you change the program from the way it is listed, we can't be responsible for whatever happens. Please, no "I just changed line 230 to make the program faster and now it won't work" letters. The programs work as they are, and if you want to modify or improve them please do, but you're on your own!
6. Have fun!

Addition and Subtraction Tutorial Program ADDSUB

Kids, how are your addition and subtraction? Moms and Dads, how're **yours?** The ADDSUB program is a tutorial program for learning or improving addition and subtraction skills. It starts off with addition and subtraction of two one-digit numbers such as 8 + 7 or 8 – 5 and progresses through addition and subtraction of two five-digit numbers such as 10500+82260.

The program keeps a count of the number of correct answers and displays the count after each answer. The program will give occasional encouraging messages.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
  ADDITION AND SUBTRACTION PROGRAM

SELECT ONE OF THE FOLLOWING:
A. ADDITION
B. SUBTRACTION
WHICH ONE?
```

You can now enter A or B followed by <ENTER>.

If you've selected addition, you'll see the first addition problem displayed on the screen, something like:

```
          ADDITION

           1
         + 9
         -----
        ?
```

You should now enter the answer, followed by <ENTER>. After you've answered, you'll see the total correct, followed by a message telling you what to do next. In this case, you'd have:

```
           1
         + 9
         -----
        ? 10

CORRECT! 1 OUT OF 1

ENTER H FOR HARDER, E FOR
EASIER, R FOR RESTART, OR
JUST <ENTER> FOR SAME?
```

If you just press <ENTER> at this point, you'll get another arithmetic problem of the same type, two one-digit numbers in this case. If you enter H, followed by <ENTER>, you'll get a problem with two two-digit numbers. You can keep entering H after each answer to get up to two five-digit numbers. You can also go back the other way, to easier problems by entering E for easier.

Entering R followed by <ENTER> brings you back to the selection between addition and subtraction.

If you select subtraction, you'll see a subtraction problem, starting with two one-digit numbers. Here again, you can get to harder problems by entering H after each answer, to easier problems by entering E, or back to the selection between addition and subtraction by entering R.

Some Background on the Program

The main problem in this program is not in doing the arithmetic – BASIC in the MC-10 and Color Computer can add and subtract numbers easily. The problem is in coming up with "random numbers." The program uses the RND function in BASIC to produce the two numbers that are used in the problems.

How This Program Works

Most of this program is involved with generating the proper sized numbers and with "formatting" the numbers used in the problems.

Array LM is a two-dimensional array that holds information for generating the numbers in the problems. There are 5 levels of difficulty, corresponding to the number of digits in the problems. The array is initialized with ten DATA values. The first set of 2 DATA values, 0 and 9, define the lowest possible number and the highest possible number that can be used for one-digit values in the problem. The next set, 10 and 99, define the lowest and highest number for two-digit values, and so forth.

Each number for the problem is generated by using RND(LM(C,2)), where C is the difficulty level from 1 through 5. The number generated is compared with LM(C,1), the lowest possible value. For subtraction, a test is also made to make certain that the number to be subtracted is less than the "minuend."

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. An encouraging message such as "GOOD WORK!" is displayed after every 10th answer if the total correct answers are over 60 per cent of the total problems.
3. Don't use commas in any of the answers. The program does not expect commas and will ignore digits after the first comma.

```
100 DIM LM(5,2), MS$(5)
110 DATA 0,9,10,99,100,999,1000,9999,10000,99999
120 DATA "GOOD WORK!","KEEP IT UP!","NICE GOING!","YOU'RE DOING GREAT"
130 DATA "FANTASTIC!"
140 FOR T=1 TO 5
150 READ LM(T,1), LM(T,2): NEXT T
160 FOR T=1 TO 5
170 READ MS$(T): NEXT T
180 CLS
190 PRINT@2,"ADDITION/SUBTRACTION PROGRAM"
200 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
210 PRINT@96,"A. ADDITION"
220 PRINT@128,"B. SUBTRACTION"
230 PRINT@160,"WHICH ONE";: INPUT A$
240 IF A$="A" THEN 290
250 IF A$="B" THEN 350
260 PRINT@224,"INVALID SELECTION--TRY AGAIN"
270 FOR T=1 TO 300: NEXT T
280 PRINT@224,"";: PRINT: PRINT@171,"": GOTO230
290 REM ADDITION
300 C=1: T1=0: T2=0
310 A1=RND(LM(C,2)): A2=RND(LM(C,2))
320 IF (A1<LM(C,1)) OR (A2<LM(C,1)) THEN 310
330 A3=A1+A2: GOSUB410
340 GOTO310
350 REM SUBTRACTION
360 C=1: T1=0: T2=0
370 A1=RND(LM(C,2)): A2=RND(LM(C,2))
380 IF (A1<LM(C,1)) OR (A2<LM(C,1)) OR (A1<A2) THEN 370
390 A3=A1-A2: GOSUB410
400 GOTO370
410 REM PRINT SUBROUTINE
420 CLS
430 IF A$="A" THEN PRINT@12,"ADDITION": PRINT@166,"+": GOTO450
440 PRINT@10,"SUBTRACTION": PRINT@166,"-"
450 PRINT@135,A1: PRINT@167,A2
460 PRINT@167,A2
470 PRINT@198,"-------"
480 B$=STR$(A2): C$=STR$(A3)
490 IF A$="A" THEN PRINT@230-(LEN(C$)-LEN(B$)),"";: INPUT T: GOTO510
500 PRINT@230+(LEN(B$)-LEN(C$)),"";: INPUT T
510 T2=T2+1
520 IF T=A3 THEN 540
530 PRINT@320,"WRONG! THE ANSWER IS";A3: GOTO570
540 T1=T1+1
550 IF ((T2/10)=INT(T2/10)) AND ((T1/T2)>.60) THEN PRINT@238,MS$(C)
560 PRINT@320,"CORRECT!";T1;"OUT OF";T2
570 PRINT@384,"ENTER H FOR HARDER, E FOR"
580 PRINT@416,"EASIER, R FOR RESTART, OR"
590 PRINT@448,"JUST <ENTER> FOR SAME";: INPUT B$
600 IF B$="R" THEN 180
610 IF B$="" THEN 650
620 IF (B$="E") AND (C>1) THEN C=C-1: GOTO650
630 IF (B$="H") AND (C<5) THEN C=C+1: GOTO650
640 GOTO570
650 RETURN
```

Amortization Schedule AMORTIZE

The AMORTIZE program calculates the monthly payments on a home or other mortgage (such as a "chattel" mortgage on an automobile). You must know the amount of money originally financed, the annual interest rate, the number of equal monthly payments, and the amount of the ending "balloon" payment, if any. The program will then print out a monthly "amortization schedule" over the entire term of the loan, showing the amount of the payment to be credited to interest, the amount credited to principal, and the balance. Although this doesn't handle every type of loan, it does provide accurate information about the most common type of home and automobile loan.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

You should see the title and first prompt message as follows:

```
     AMORTIZATION SCHEDULE

ENTER PRINCIPAL:?
```

You must supply the principal amount, the number of months of payments, the monthly payment, and the amount of any balloon payment. You can get a printout on the system line printer by answering "Y" to the last prompt.

Suppose that you have a house you purchased for $147,000 with $20,000 down at a fixed interest rate of 12.5 % per year. The term of the loan is 30 years, and the monthly payments are to be $1355.42. There is no balloon payment. You'd enter the following information, assuming you wanted a printout from your system line printer:

```
     AMORTIZATION SCHEDULE

ENTER PRINCIPAL:? 127000
 INTEREST RATE (% PER YEAR)? 12.5
 NUMBER OF MONTHS? 360
 MONTHLY PAYMENTS? 1355.42
 BALLOON PAYMENT?
 TO PRINTER (Y OR N)? Y
```

The program will now print out the complete amortization schedule for the 360 months of the loan.

If you had answered "N" to the question for the printer, the screen would look like this:

```
                  AMORTIZATION SCHEDULE
                      TO             TO
   MONTH           PRINC.         INTER.        BALANCE
      1             32.50        1322.91       126967.49
      2             32.84        1322.57       126934.65
      3             33.18        1322.23       126901.47
      4             33.52        1321.89       126867.94
      5             33.87        1321.54       126834.06
      6             34.23        1321.18       126799.83
      7             34.58        1320.83       126765.24
      8             34.94        1320.47       126730.29
      9             35.31        1320.10       126694.98
     10             35.68        1319.73       126659.29
     11             36.05        1319.36       126623.24
PRESS ANY KEY TO CONTINUE
```

At this point pressing any key will display the next screen full of information. Press any key after each screen until you get to the end of the amortization schedule. At the end of the amortization schedule on either the printout or screen, you'll see a "totals" row such as:

```
TOT 127010.3 360940.82
```

For amounts over $100,000, the last digit of the principal amount will be blank. This is normal and was done to line up the columns.

The totals row gives the accumulated total amount paid to the principal and the total amount paid to interest.

You'll notice in running this program that there will be a slight error in calculating the balance. In the example above, the balance was $10.38 to the good after 360 payments of $1355.42. This is due to approximation used in the monthly payment, and not to any errors in the MC-10 or Color Computer. In the example above, a monthly payment of $1355.41, one cent less than the previous leaves $28.73 owing. The actual monthly payment should be about $1355.4123, but its hard to pay in fractional cents!

There may be a difference of less than a dollar when you compare this amortization schedule with an amortization schedule run by your saving and loan's large computer. In this case, don't argue with them. After all, they're paying hundreds of times more than you are to get the same information and you don't want to make them feel cheated!

Balloon payments: On some loans, the loan is not "fully amortized" - there is a balloon payment due at the end of the loan. If your loan does not have a balloon payment, simply press <ENTER> when you come to that question. If you don't know what the balloon payment is, this program will calculate it for you - just press <ENTER> for the balloon payment and look at the last month's balance to see what the payment will be.

Some Background on the Program

Interest rates are usually quoted in annual rates. The simplest type of loan is for a sum of money at a specified annual interest rate with the sum of money plus interest due after a specified time. You could borrow $1000 at 12% for one year, for example, and at the end of a year you'd have to pay back the original sum of money plus interest of $1000 x .12 = $120, making the total amount due $1120.

Now suppose that you borrow $1000 at 12% per year with the money due in 12 equal installments. What should those installments be? **Not $1120/12 or $93.33!** If you paid back 1/12th of the final amount due each month, you'd be paying more interest than 12%! After all, in the first case, you had the full use of the money for a year. In the second case you had the use of $1000 for one month, the use of $906.67 for the second month, the use of $873.33 the third month, and so forth.

The proper way to figure the regular monthly payment is by the formula given in the Interest Program INTEREST. It takes into account the fact that you don't have the full use of the loan amount for the life of the loan, and computes smaller monthly payments so that the true interest rate is that given in the loan agreement.

In this program we're faced with an existing loan with payments that are already established. All we have to do here is to compute the interest due for the last month's use of the money. This can be done fairly simply by taking the annual interest rate, dividing it by 12 to get a monthly interest rate, and then multiplying the loan balance by the monthly interest rate to get the amount that should go towards interest. Any remaining amount is used to reduce the principal. If the monthly payments are correct, the principal will be reduced down to zero on the last payment.

One more thing: If you run the program and look at the accumulated interest paid over the life of the loan, it may seem **much** too large to be true. It's correct (sigh...).

How This Program Works

Most of the calculations in this program are concerned with "formatting" the screen and printer listings. The actual computation is done in line 380, where the annual interest rate is divided by 12 to find the monthly interest rate. This is then multiplied by the current balance, P, to find the interest due for the last month.

The program computes the interest due for each of the months over the life of the loan, adjusting the balance by subtracting any money that does not go to interest.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. It's possible to have the balance **increase** if the monthly payment is not adequate to meet the interest rate on the loan.
3. Commas can't be used in any of the amounts that are input. The program does not expect commas and will ignore any digits after the first comma.
4. The interest rate input must be in percentage points, and not a fraction. In other words, 18 percent should be entered as "18" and not ".18."
5. The maximum principal is $9,999,999.99. The maximum interest rate is 90%. The maximum number of months is 499. The maximum monthly payment is $9,999.99. The maximum balloon payment is $9,999,999.99.

```
100 TP=0: TI=0: T$=""
110 CLS: PRINT@5,"AMORTIZATION SCHEDULE"
120 PRINT@64,"ENTER PRINCIPAL";: INPUT P$
130 IF (VAL(P$)>0) AND (VAL(P$)<10000000) THEN P=VAL(P$): GOTO150
140 PRINT@80,"": GOTO120
150 PRINT@97,"INTEREST RATE (% /YR)";: INPUT I$
160 IF (VAL(I$)>0) AND (VAL(I$)<91) THEN I=VAL(I$)/100: GOTO180
170 PRINT@117,"": GOTO150
180 PRINT@129,"NUMBER OF MONTHS";: INPUT N$
190 IF (VAL(N$)>0) AND (VAL(N$)<500) THEN N=VAL(N$): GOTO210
200 PRINT@147,"": GOTO180
210 PRINT@161,"MONTHLY PAYMENT";: INPUT R$
220 IF (VAL(R$)>0) AND (VAL(R$)<10000) THEN R=VAL(R$): GOTO240
230 PRINT@178,"": GOTO210
240 PRINT@193,"BALLOON PAYMENT";: INPUT B$
250 IF (VAL(B$)>=0) AND (VAL(B$)<1000000) THEN B=VAL(B$): GOTO270
260 PRINT@210,"": GOTO240
270 PRINT@225,"TO PRINTER (Y OR N)";: INPUT PR$
280 IF NOT (PR$="Y" OR PR$="N") THEN PRINT@245,"": GOTO270
290 GOSUB590
300 IF PR$<>"Y" THEN 370
310 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
320 PRINT#-2, TAB(5)"AMORTIZATION SCHEDULE"
330 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
340 PRINT#-2, TAB(8)"TO        TO"
350 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
360 PRINT#-2, "MONTH PRINC.   INTER.   BALANCE"
370 FOR T=1 TO N
380 WI=P*(I/12): TI=TI+WI
390 WP=R-WI: TP=TP+WP: P=P+WI-R
400 N$=STR$(T): L=LEN(N$)
410 PRINT@P1+4-L,N$;
420 IF PR$="Y" THEN PRINT#-2, TAB(4-L)N$;
430 C=INT(WP*100): P2=10: GOSUB630
440 C=INT(WI*100): P2=19: GOSUB630
450 C=INT(P*100): P2=29: GOSUB630
460 IF P1<416 THEN P1=P1+32: GOTO510
470 IF T=N THEN 510
480 PRINT@448,"PRESS ANY KEY TO CONTINUE";: B$=INKEY$
490 IF B$="" THEN 480
500 GOSUB590
510 NEXT T
```

```
520 PRINT@P1,"TOTL";
530 IF PR$="Y" THEN PRINT#-2, "TOTL";
540 C=INT(TP*100): P2=10: GOSUB630
550 C=INT(TI*100): P2=19: T$="Y": GOSUB630
560 PRINT@480,"PRESS R TO RESTART";: B$=INKEY$
570 IF B$="R" THEN 100
580 GOTO560
590 CLS: PRINT@5,"AMORTIZATION SCHEDULE"
600 PRINT@40,"TO": PRINT@49,"TO"
610 PRINT@64,"MONTH PRINC.   INTER.   BALANCE"
620 P1=96: RETURN
630 C$=STR$(C): L=LEN(C$)
640 PRINT@P1+P2-(L-2),LEFT$(C$,L-2)+"."+RIGHT$(C$,2);
650 IF PR$<>"Y" THEN 710
660 IF (P2<>29) AND (T$<>"Y") THEN 700
670 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
680 PRINT#-2, TAB(P2-(L-2))LEFT$(C$,L-2)+"."+RIGHT$(C$,2): GOTO 710
690 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
700 PRINT#-2, TAB(P2-(L-2))LEFT$(C$,L-2)+"."+RIGHT$(C$,2);
710 RETURN
```

Bach Composition Program BACH

The BACH program turns your MC-10 or Color Computer into a harpsichord, playing a contemporary version of a BACH composition, "Jesu, Joy of Man's Desiring." Baroque computer music sounds especially good because of the timbre of the square wave generated by SOUND. Give this piece a try – you'll be pleasantly surprised at how good it sounds – probably more so if you're a casual Bach listener and don't conduct a major symphony orchestra!

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

The MC-10 produces musical tones by the SOUND command. The format of SOUND is

```
SOUND F,D
```

where F is any value from 1 through 255, and D is any value from 1 through 255. The F value establishes the "pitch" of the sound, while the D value establishes the duration of the sound.

A D value of 1 is about .075 seconds. A D value of 255 is about 19 seconds on the MC-10, and a little less on the Color Computer. D values in between 1 and 255 produce sounds of between .075 and 19 seconds. Every time D is increased by 1, another .075 seconds is added to the length of the note produced.

The F value changes the pitch, or "frequency" of the sound. A value of 1 for F is a pitch of about 151 cycles per second (151 hertz); this is about equal to D sharp below middle C on the piano keyboard. A value of 255 for F is a pitch of about 14,700 cycles per second (14,700 hertz); this is far beyond the highest-pitched key on a piano. Values of F in between 1 and 255 produce notes between 151 and 14,700 cycles per second.

The sound produced by the MC-10 is a "square wave," as shown in Figure MUSIC-1. This type of waveform is rich in "odd harmonics" and is used in electronic music synthesizers.

The Color Computer also uses the SOUND command in the same way as the MC-10. Extended Color BASIC also has the PLAY command, a powerful command that allows you to easily play sequences of notes over different octaves, durations, and loudness.

Producing a tone such as this BACH piece is largely a job of custom tailoring a piece of music. It helps to start with a simplified version of the musical piece and then to convert note values from a score into SOUND F and D values. Once this is done, you may find that you can modify the piece, speeding it up and slowing it down in places, and changing the F values slightly to get the best sound.

Start with these values for the corresponding notes:

Note	Color Computer	MC-10
E	–	25
F	5	38
F#	19	53
G	32	64
G#	45	77
A	58	87
A#	69	95
B	78	103
Middle C	89	113
C#	99	121
D	108	129
D#	117	135
E	125	142
F	133	149
F#	140	155
G	147	161
G#	153	166
A	159	171
A#	165	176
B	170	181
C	176	185
C#	180	189
D	185	193
D#	189	196
E	193	200
F	197	203
F#	200	206
G	204	209
G#	207	212
A	210	215
A#	213	217
B	216	219
C	218	221
C#	221	223
D	223	225
D#	225	227
E	227	229
F	229	231
F#	231	232

G	232	233
G#	234	235
A	236	236
A#	237	237
B	238	239
C	239	240
C#	241	241
D	242	242
D#	243	243
E	244	244

How This Program Works

The DATA statements at the beginning of the program hold the F and D values for the SOUND command. A READ statement reads one F and one D value and these are used in the following SOUND command. Two -1 values mark the end of the piece. When these are READ, the "DATA pointer" is reset to the beginning by a RESTORE and the piece is restarted.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. F values are a compromise between values suitable for the MC-10 and values suitable for the Color Computer.

```
100 DATA 89,1,125,1,147,1,176,1,147,1,125,1
110 DATA 89,1,125,1,147,1,89,1,147,1,125,1
120 DATA 89,1,125,1,147,1,176,1,147,1,125,1
130 DATA 89,1,125,1,147,1,89,1,147,1,125,1
140 DATA 89,2,108,2,125,2,147,2,133,2,133,2,159,2,147,2,147,2,176,2,170,2,176,2
150 DATA 147,2,125,2,89,2,108,2,125,2,133,2,147,2,159,2,147,2
160 DATA 133,2,125,2,108,2,125,2,133,2,32,2,78,2,108,2,133,2,125,2
170 DATA 108,2,125,2,89,2
180 DATA 108,2,125,2,147,2,133,2,133,2,159,2,147,2,147,2,176,2,170,2,176,2
190 DATA 147,2,125,2,89,2,108,2,125,2,58,2,133,2,125,2,108,2
200 DATA 89,2,58,2,32,2,89,2,78,2,89,2
210 DATA 89,1,125,1,147,1,176,1,147,1,125,1
220 DATA 89,1,125,1,147,1,89,1,147,1,125,1
230 DATA 89,1,125,1,147,1,176,1,147,1,125,1
240 DATA 89,1,125,1,147,1,89,1,147,1,125,1
250 DATA -1,-1
260 CLS
270 READ A,B
280 IF A<0 AND B<0 THEN 310
290 SOUND A,B
300 GOTO 270
310 RESTORE: GOTO 260
```

Banner Program BANNER

This is a Texas version of a display program. You jus' run this heah program, pardner, and you'll see what ah mean. BANNER is used to display a large-sized message on the screen. It displays a 1- to 36-character message on the screen in "banner" format – characters that are 7 character positions high. The message scrolls across the screen in "billboard" fashion, in color. A typical display is shown in Figure BANNER-1. BANNER requires a 16K Color Computer or MC-10 with memory expansion pack, as it is a large program that uses a lot of string space.

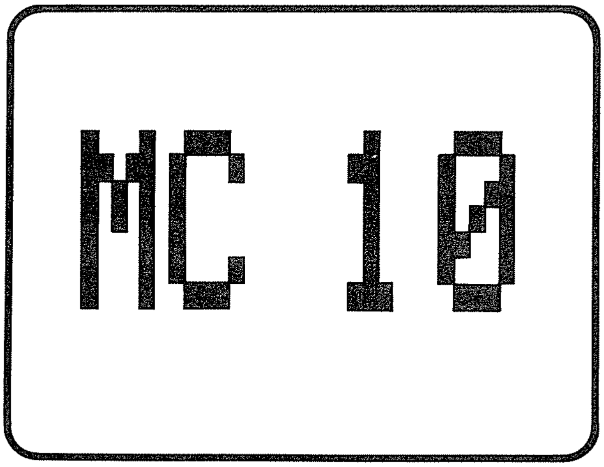


Figure BANNER-1. Typical BANNER Display

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

You should see the title and first prompt message as follows:

```
          BANNER
COLOR (1-8)?
```

You can now enter a color code for the screen characters. Color codes are

- 1 for green
- 2 for yellow
- 3 for blue
- 4 for red
- 5 for buff
- 6 for cyan
- 7 for magenta
- 8 for orange

Enter the single digit code for the color, followed by <ENTER>.

You'll now see the message:

```
ENTER 1 TO 36 CHARACTERS?
```

You can now enter any message string from 1 to 36 characters in length. Typical messages might be "LINDBERG ARRIVES IN PARIS" or "MC 10 RULES." Only the alphabetic characters A through Z and the digits 0 through 9 are allowed in the message. You can enter other characters, but they'll be replaced with a blank. Terminate the string with an <ENTER>.

Banner will now construct the strings for high-speed output. This process will take over a minute for a 36-character string, so be patient. After BANNER constructs the strings you'll see the message scroll across the screen in moving billboard fashion from right to left. The message will be continuously displayed in the color that you selected.

Some Background on the Program

If you look closely at your MC-10 or Color Computer screen, you'll see that characters are formed by a "dot-matrix" representation. The letter "A," for example, is made up of individual dots joined together to make up the character, as in

```
  OO
 O  O
O    O
OOOOOO
O    O
O    O
```

When these dots are displayed on a television, they merge together to form a series of line segments; you would be able to see the individual dots much more clearly on a high-resolution "monitor," if the MC-10 or Color Computer had this option. This scheme of "dot matrix" representation is used in many line printers as well as most television displays.

We can use the same techniques to display large characters on the screen. Instead of individual dots, we'll use "character positions" on the screen and a solid "graphics" character in each character position. The Color Computer and MC-10 have 16 lines of 32 character positions, a total of 512 character positions in all, as shown in Figure BANNER-2. Each character position can hold either a text character or a graphics character.

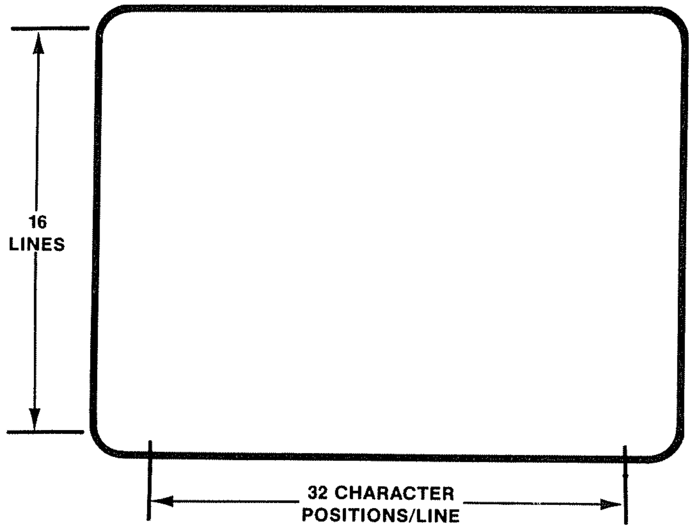


Figure BANNER-2. MC-10 and Color Computer Low-Resolution Graphics

To allow the maximum number of large characters on the screen, a character will be made up of a 5 by 7 "matrix" (a checkerboard) of graphics characters. The definitions for the characters can be held conveniently in DATA statements. Each character is represented by 5 column values and 7 row values, for a total of 35 values. Since the character element will be either "off" or "on," we can hold a complete character definition in 35 binary digits – 35 ones or zeroes.

One problem in this type of program is speed. If we were to compute each character "on the fly" and generate the proper combination of elements for each new character, the result would be much too slow for a billboard type of display. The answer is to work with the entire message all at once and compute seven "strings" representing the seven message rows, as shown in Figure BANNER-3.

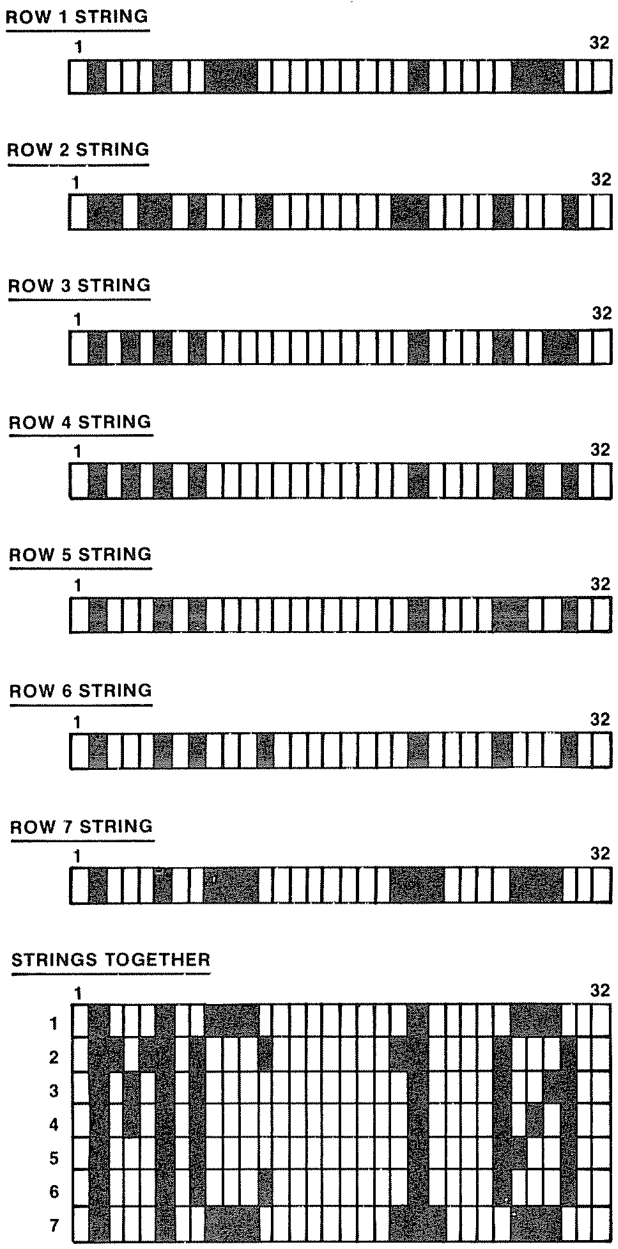


Figure BANNER-3. Seven Message Rows

How This Program Works

There are two major parts to this program, character storage and character generation.

Characters are stored in DATA statements. Each DATA statement holds the configuration for one character, in seven row values. The binary value for each row is the dot configuration for the row. Take the first DATA statement as an example. The values are 14, 17, 19, 21, 25, 17, and 14. If we convert these to binary (see the DECCONV program) we have:

```
01110
10001
10011
10101
11001
10001
01110
```

Changing each one to "O" and the zeroes to blanks, we'd get:

```
 OOO
O   O
O  OO
O O O
OO  O
O   O
 OOO
```

If you squint your eyes at this figure, you'll see a "0" digit appear. The other characters are stored similarly. All of the DATA definitions are stored in array CH for easy (and uniform speed) access.

Characters are generated by taking each separate character from the message, finding the corresponding dot matrix form from array CH, and then adding to one of seven strings S$( ) that make up each screen row. When all characters have been processed, S$(0) through S$(7) hold the seven rows of the message.

The message is moved across the screen by displaying a portion of each row, moving from the left part of the string to the right. Blanks are added initially to the message string so that the message starts from the right side of the screen.

Special Notes

1. This program will run on 16K MC-10 and Color Computer systems only.
2. You can define any characters you'd like by changing the DATA values. Make certain that you have seven values in each DATA statement, however.
3. Input strings greater than 36 characters will be ignored.

```
100 CLEAR3000
110 DIM CH(258),CL(4),R$(6),S$(6)
120 CLS: PRINT@12,"BANNER"
130 PRINT@64,"COLOR (1-8)";: INPUT CO
140 IF (CO<1 OR CO>8) THEN PRINT @ 64," ": GOTO 130
150 CO=128+(CO-1)*16+15
160 INPUT "ENTER 1 TO 36 CHARACTERS";A$
170 IF LEN(A$)>36 THEN 160
180 CLS
190 A$="      "+A$
200 DATA 14,17,19,21,25,17,14
210 DATA 4,12,4,4,4,4,14
220 DATA 14,17,1,14,16,16,31
230 DATA 14,17,1,6,1,17,14
240 DATA 2,6,10,18,31,2,2
250 DATA 31,16,30,1,1,17,14
260 DATA 6,8,16,30,17,17,14
270 DATA 31,1,2,4,8,16,16
280 DATA 14,17,17,14,17,17,14
290 DATA 14,17,17,15,1,2,12
300 DATA 0,0,0,0,0,0,0
310 DATA 4,10,17,17,31,17,17
320 DATA 30,9,9,14,9,9,30
330 DATA 14,17,16,16,16,17,14
340 DATA 30,9,9,9,9,9,30
350 DATA 31,16,16,30,16,16,31
360 DATA 31,16,16,28,16,16,16
370 DATA 15,16,16,19,17,17,15
380 DATA 17,17,17,31,17,17,17
390 DATA 14,4,4,4,4,4,14
400 DATA 1,1,1,1,1,17,14
410 DATA 17,18,20,24,20,18,17
420 DATA 16,16,16,16,16,16,31
430 DATA 17,27,21,21,17,17,17
440 DATA 17,25,21,19,17,17,17
450 DATA 14,17,17,17,17,17,14
460 DATA 30,17,17,30,16,16,16
470 DATA 14,17,17,17,21,18,13
480 DATA 30,17,17,30,20,18,17
490 DATA 14,17,16,14,1,17,14
500 DATA 31,4,4,4,4,4,4
510 DATA 17,17,17,17,17,17,14
520 DATA 17,17,17,10,10,4,4
530 DATA 17,17,17,17,21,27,17
540 DATA 17,17,10,4,10,17,17
550 DATA 17,17,10,4,4,4,4
560 DATA 31,1,2,4,8,16,31
570 FOR I=0 TO 37*7-1
580 READ CH(I)
590 NEXT I
600 FOR I=0 TO 6
610 S$(I)=""
620 NEXT I
630 FOR J=1 TO LEN(A$)
640 B$=MID$(A$,J,1): IF B$=" " THEN I=10: GOTO 690
650 I=ASC(B$)-48: IF I<0 THEN I=10: GOTO 690
660 IF I<10 THEN 690
670 IF I<17 THEN I=10: GOTO 690
680 I=I-6
690 I=I*7
700 FOR R=0 TO 6
```

```
710 A=CH(I+R)
720 FOR C=0 TO 4
730 CL(4-C)=A-INT(A/2)*2: A=INT(A/2)
740 NEXT C
750 RW$=""
760 FOR K=0 TO 4
770 IF CL(K)=0 THEN RW$=RW$+" "
780 IF CL(K)=1 THEN RW$=RW$+CHR$(CO)
790 NEXT K
800 RW$=RW$+" "
810 S$(R)=S$(R)+RW$
820 NEXT R
830 NEXT J
840 FOR K=1 TO LEN(A$)*6
850 FOR R=0 TO 6
860 R$(R)=MID$(S$(R),K,30)
870 NEXT R
880 PRINT@130,R$(0);
890 PRINT@162,R$(1);
900 PRINT@194,R$(2);
910 PRINT@226,R$(3);
920 PRINT@258,R$(4);
930 PRINT@290,R$(5);
940 PRINT@322,R$(6);
950 NEXT K
960 GOTO 840
```

"Cadnza" CADNZA

BACH, CAMP, GREEN, ODE, and SAINTS are all fun songs to play on your MC-10 or Color Computer, but here's a grandstand version of a computer song. This one was commissioned specifically for this book. We told a professional musician and computer programmer/analyst, Anthony (Craig) Verbeck to compose something "showy." Craig has experience on various electronic and computer music synthesizers and is not a typical MC-10 or Color Computer user, but his composition shows what can be done with a little bit of effort and some musical background.

CADNZA uses two voices in counterpoint and sounds terrific. It runs almost two minutes so it provides a good chance to exercise a lot of interesting techniques that you might want to copy for your own electronic music compositions.

How to Use This Program

Enter the program into the MC-10 or Color Computer. There are two versions of CADNZA, one for the Color Computer (CADNZA) and one for the MC-10 (CADNZAMC). The Color Computer version with comments fits into a 16K Color Computer. The MC-10 version has comments deleted and will fit into a 4K MC-10 without memory expansion pack. Data values are different in the two versions because of speed differences between the two systems.

Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

See the writeup for BACH. This program uses the same principals to generate tones by the SOUND command in BASIC.

Unlike BACH, though, CADNZA doesn't play from DATA values. The DATA values are used to initialize array CD. This makes it easier to access chords in a "random" fashion and cuts down the overall length of DATA that has to be maintained.

Unfortunately, it's not easy to write a generalized program to play a song. One of the reasons for this is that the "overhead" of decoding note values, tempos, and other parameters is too high and slows down the overall speed of a composition. CADNZA has a fast tempo in parts that simply could not be maintained in a generalized BASIC PLAY routine. Your best bet for a generalized music routine is the PLAY command in Color Computer Extended Color BASIC, which gives you a great deal of flexibility in playing music primarily because it's "built-in" as a BASIC command in assembly language and not a BASIC subroutine.

CADNZA "Cadnza"

How This Program Works

The DATA statements at the beginning of the program hold the F values for the SOUND command. These values are used to initialize array CD and array MD. The CD array is the chord array and the MD array holds the melody.

The Play Chord subroutine is used to play a chord at a given speed and for a given number of times and uses the CD array only. The Play a Melody and Chord is used to play both the melody and chord and uses both the MD and CD arrays.

The "main sequence" calls both the chord and melody/chord subroutines and also has SOUND commands interspersed with the calls.

Special Notes

1. The "CADNZA" version of this program requires a 16K Color Computer.
2. The "CADNZAMC" version of this program will run on any MC-10 system.

CADNZA

```
100 CLS
110 GOTO 790
120 REM ****************************************
130 REM *   READ ARRAYS SUBROUTINE             *
140 REM *   THIS SUBROUTINE READS THE CHORD    *
150 REM *   ARRAY AND THE MELODY ARRAY         *
160 REM ****************************************
170 DIM CD(8,4),MD(200)
180 FOR I=1 TO 8
190 FOR J=1 TO 4
200 READ CD(I,J)
210 NEXT:NEXT:I=0
220 I=I+1:READ MD(I)
230 IF MD(I)<0 THEN RETURN
240 GOTO 220
250 REM ****** DATA ************
260 DATA 125,147,170,147
270 DATA 140,159,176,159
280 DATA 147,170,185,170
290 DATA 159,176,193,176
300 DATA 78,117,140,117
310 DATA  89,125,147,125
320 DATA 108,140,159,140
330 DATA 193,204,216,204
340 REM ***** MELODY ***********
350 DATA 204,200,193,185
360 DATA 159,170,176,185
370 DATA 204,200,193,204
380 DATA 210,204,200,216
390 DATA 218,216,210,204
400 DATA 200,204,210,216
410 DATA 210,204,200,193
420 DATA 204,200,193,185
430 REM ***** C PART *****
440 DATA 176,147,125,108
```

```
450 DATA 89,125,140,147
460 DATA 159,176,170,193
470 DATA 185,200,210,216
480 DATA 218,227,232,231
490 DATA 227,218,223,210
500 DATA 204,193,185,176
510 DATA 170,147,125,89
520 DATA -1
530 REM ************************************
540 REM *      PLAY A CHORD                 *
550 REM *          N=NUMBER OF TIMES        *
560 REM *          S=SPEED                  *
570 REM *          C=WHICH CHORD            *
580 REM ************************************
590 FOR N1=1 TO N
600 FOR N2=1 TO 4
610 SOUND CD(C,N2),S
620 NEXT:NEXT:RETURN
630 REM ***************************************
640 REM *  PLAY A MELODY AND CHORD            *
650 REM *  PLAYS THE MELODY AS A FIRST NOTE   *
660 REM *  USES THE SAME PARAMETERS AS PLAY A *
670 REM *  CHORD WITH THE ADDITION OF NT      *
680 REM *  WHICH IS THE NOTE IN THE MELODY TO *
690 REM *  START PLAY                         *
700 REM ***************************************
710 IF MD(NT)<1 THEN RETURN
720 FOR N1=1 TO N
730 FOR N2=1 TO 4
740 SOUND MD(NT),S:NT=NT+1
750 SOUND CD(C,N2),S
760 IF MD(NT)<1 THEN RETURN
770 NEXT:NEXT:RETURN
780 NEXT:RETURN
790 REM ******** SONG *********
800 GOSUB 120 : REM READ ARRAY
810 S=4:N=4:C=1:GOSUB 530
820 C=6:GOSUB 530
830 C=5:N=2:GOSUB 530
840 SOUND 78,4:SOUND 125,4
850 SOUND 147,4:SOUND 159,4
860 C=3:N=1:GOSUB 530
870 REM **** COUNTERPOINT ****
880 N=4
890 C=1:S=2:NT=1:GOSUB 710
900 C=6:NT=1:GOSUB 710
910 C=1:NT=1:GOSUB 710
920 C=6:NT=1:GOSUB 710
930 C=5:N=2:NT=8:GOSUB 710
940 SOUND 78,2:SOUND 170,2
950 SOUND 125,2:SOUND 193,2
960 SOUND 147,2:SOUND 204,2
970 SOUND 159,2:SOUND 210,2
980 NT=8:C=3:N=1:GOSUB 710
990 S=1
1000 N=8:NT=1:C=1
1010 GOSUB 710
1020 C=6:GOSUB 710
1030 NT=1
1040 C=1:GOSUB 710
1050 C=6:GOSUB 710
```

```
1060 NT=1
1070 C=5:GOSUB 710
1080 NT=1
1090 GOSUB 710
1100 C=1:N=4:S=2:NT=1:GOSUB 710
1110 C=4:N=2:GOSUB 710
1120 C=5:GOSUB 710
1130 C=1:N=4:NT=1:GOSUB 710
1140 C=4:N=2:GOSUB 710
1150 C=5:GOSUB 710
1160 C=6:GOSUB 710
1170 C=7:GOSUB 710
1180 NT=1
1190 C=8:N=8:GOSUB 710
1200 NT=1
1210 S=8:N=1:GOSUB 710
1220 SOUND 125,32
```

CADNZAMC

```
100 CLS
110 GOTO570
120 DIM CD(8,4),MD(200)
130 FOR I=1 TO 8
140 FOR J=1 TO 4
150 READ CD(I,J)
160 NEXT:NEXT:I=0
170 I=I+1:READ MD(I)
180 IF MD(I)<0 THEN RETURN
190 GOTO170
200 DATA 135,155,176,155
210 DATA 149,166,181,155
220 DATA 155,176,189,176
230 DATA 166,181,196,181
240 DATA 95,129,149,129
250 DATA 103,135,155,135
260 DATA 121,149,166,149
270 DATA 196,206,217,206
280 DATA 206,203,196,189
290 DATA 166,176,181,189
300 DATA 206,203,196,206
310 DATA 212,206,203,217
320 DATA 219,217,212,206
330 DATA 203,206,212,217
340 DATA 212,206,203,196
350 DATA 206,203,196,189
360 DATA 181,155,135,121
370 DATA 103,135,149,155
380 DATA 166,181,176,196
390 DATA 189,203,212,217
400 DATA 219,227,232,231
410 DATA 227,219,223,212
420 DATA 206,196,189,181
430 DATA 176,155,135,103
440 DATA -1
450 FOR N1=1 TO N
460 FOR N2=1 TO 4
470 SOUND CD(C,N2),S
480 NEXT:NEXT:RETURN
490 IF MD(NT)<1 THEN RETURN
500 FOR N1=1 TO N
```

```
510 FOR N2=1 TO 4
520 SOUND MD(NT),S:NT=NT+1
530 SOUND CD(C,N2),S
540 IF MD(NT)<1 THEN RETURN
550 NEXT:NEXT:RETURN
560 NEXT:RETURN
570 GOSUB120 : REM READ ARRAY
580 S=4:N=4:C=1: GOSUB450
590 C=6: GOSUB450
600 C=5:N=2: GOSUB450
610 SOUND 95,4:SOUND 135,4
620 SOUND 155,4:SOUND 166,4
630 C=3:N=1: GOSUB450
640 N=4
650 C=1:S=2:NT=1: GOSUB490
660 C=6:NT=1: GOSUB490
670 C=1:NT=1: GOSUB490
680 C=6:NT=1: GOSUB490
690 C=5:N=2:NT=8: GOSUB490
700 SOUND 95,2:SOUND 176,2
710 SOUND 135,2:SOUND 196,2
720 SOUND 155,2:SOUND 206,2
730 SOUND 166,2:SOUND 212,2
740 NT=8:C=3:N=1: GOSUB490
750 S=1
760 N=8:NT=1:C=1
770 GOSUB490
780 C=6: GOSUB490
790 NT=1
800 C=1: GOSUB490
810 C=6: GOSUB490
820 NT=1
830 C=5: GOSUB490
840 NT=1
850 GOSUB490
860 C=1:N=4:S=2:NT=1: GOSUB490
870 C=4:N=2: GOSUB490
880 C=5: GOSUB490
890 C=1:N=4:NT=1: GOSUB490
900 C=4:N=2: GOSUB490
910 C=5: GOSUB490
920 C=6: GOSUB490
930 C=7: GOSUB490
940 NT=1
950 C=8:N=8: GOSUB490
960 NT=1
970 S=8:N=1: GOSUB490
980 SOUND 135,32
```

Perpetual Calendar Program CALPGM

Ever wonder on which day of the week your mother-in-law's birthday would fall in the year 1990? Maybe I picked a bad example. . .How about finding out whether 1900 was a leap year? Or what day of the week July 4th, 1850 was? One way to do it is by checking an Almanac that gives a "Perpetual Calendar." This is usually a list of all possible yearly calendars with a cross reference number. You would find the year in question, note the reference number, and then look up the calendar that shows the correct arrangement of starting dates, days in February, and so forth.

This program will replace the "Perpetual Calendar" in the Almanac with a BASIC program. The BASIC program will do the calculations for you and display the month you're looking for on the screen. All you need to know is the month and the year.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start by entering

```
    RUN
```

You should see the title and first prompt message as follows:

```
   PERPETUAL CALENDAR PROGRAM

ENTER CURRENT MONTH(1-12)?
```

Enter the number of the month for which you want the calendar displayed, from 1 (January) to 12 (December).

Next, you'll see the message

```
    YEAR(1601-2399)?
```

displayed. Enter a year from 1601 to 2399 for the month in question.

If you enter an incorrect month or year, you'll see a brief error message, and the program will erase what you've entered:

```
   PERPETUAL CALENDAR PROGRAM

ENTER CURRENT MONTH(1-12)? 13

      YEAR(1601-2399)? 1900

MONTH 13 INVALID--TRY AGAIN
```

Once you've entered a good month and year, the program will display a calendar as shown in Figure CALPGM-1.

```
  SUN    MON    TUE    WED    THU    FRI    SAT

          1      2      3      4      5      6

   7      8      9     10     11     12     13

  14     15     16     17     18     19     20

  21     22     23     24     25     26     27

  28     29     30     31

                AUGUST, 1983
```

Figure CALPGM-1. Calendar Display

Pressing any key after the display will bring you back to a new set of questions for the month. You can keep entering months and years forever – perpetually.

Some Background on This Program

The calendar is really based on the earth's and moon's rotation. The time it takes the earth to circle the sun is one year. The time it takes the moon to circle the earth is roughly one month. The time it takes the earth to rotate on its axis is called one day. The problem is that one year is not exactly equal to 365 days – it equals 365 days, 6 hours, 9 minutes, and 9.5 seconds (a "sidereal" year).

Julius Caesar established the Julian calendar, which was based on the best information he had (from a Greek, Sosigenes). In the Julian calendar, one year equals 365 1/4 days. The 1/4 day was accounted for by making every fourth year a leap year. This was close to the actual "sidereal time," but not close enough.

By 1582, the accumulated error of 9 minutes, 9.5 seconds per year had made the calendar 10 days too slow. Pope Gregory decreed that the day following October 4, 1582 would be called October 15, 1582 to erase this accumulated error. He also made a further correction. Every year in the Julian calendar evenly divisible by 4 was a leap year (1500, 1504, 1508, etc). Pope Gregory made "century" years (1600, 1700, 1800, 2000) "normal" years, except for century years divisible by 400, which would remain leap years. This makes the calendar very close to the "sideral" year, but again not exactly equal.

The Gregorian calendar was adopted at different times by different countries throughout history, until at this time most major countries reckon time by this method.

How This Program Works

This program starts counting from a known point, January 1st of 1600, which was a Saturday and a leap year. From this point it calculates the number of years between 1600 and the year you enter, adding years divisible by 4 (leap years), and subtracting century years except for the year 2000. It then calculates the number of days in the months prior to the month you've entered. The result is an "offset" which gives the starting day of the week for the first of the month. The program also knows the number of days in each month so that it can easily print out the month display.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 DIM M$(12)
110 DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE
120 DATA JULY,AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
130 FOR I=1 TO 12
140 READ M$(I)
150 NEXT I
160 CLS
170 PRINT@3,"PERPETUAL CALENDAR PROGRAM"
180 PRINT@64,"ENTER CURRENT MONTH(1-12)";: INPUT MM
190 PRINT@106,"YEAR(1601-2399)";: INPUT YY
200 IF (MM>0) AND (MM<13) THEN 230
210 PRINT@160,"MONTH ";MM;" INVALID--TRY AGAIN"
220 FOR T=1 TO 400: NEXT T: GOTO160
230 IF (YY>1600) AND (YY<2400) THEN 260
240 PRINT@224,"YEAR ";YY;" INVALID--TRY AGAIN"
250 FOR T=1 TO 400: NEXT T: GOTO160
260 REM COMPUTE LEAP YR, DAYS OF MO AND WEEK
270 OF=6
280 IF YY>2000 THEN OF=0
290 OF=(OF+(YY-1600))+INT((YY-1600-1)/4)+1-INT((YY-1600-1)/100)
300 OF=OF-INT(OF/7)*7
310 LY=0
320 IF YY-INT(YY/4)*4=0 THEN LY=1
330 IF YY-INT(YY/100)*100=0 THEN LY=0
340 IF YY=2000 THEN LY=1
350 IF MM=2 THEN OF=OF+3
360 IF MM=3 THEN OF=OF+3+LY
370 IF MM=4 THEN OF=OF+6+LY
380 IF MM=5 THEN OF=OF+8+LY
390 IF MM=6 THEN OF=OF+11+LY
400 IF MM=7 THEN OF=OF+13+LY
410 IF MM=8 THEN OF=OF+16+LY
420 IF MM=9 THEN OF=OF+19+LY
430 IF MM=10 THEN OF=OF+21+LY
440 IF MM=11 THEN OF=OF+24+LY
450 IF MM=12 THEN OF=OF+26+LY
460 OF=OF-INT(OF/7)*7
470 DM=31
480 IF MM=2 THEN DM=28+LY
490 IF (MM=4) OR (MM=6) OR (MM=9) OR (MM=11) THEN DM=30
500 REM DRAW LINES AND PRINT DAYS OF WEEK
510 CLS
520 PRINT@34,"SUN MON TUE WED THU FRI SAT"
530 FOR I=65 TO 93
540 PRINT@I,CHR$(131);
550 NEXT I
560 FOR I=93 TO 477 STEP 32
570 J=138
580 IF I=93 THEN J=130
590 PRINT@I,CHR$(J);
600 NEXT I
```

```
610 FOR I=477 TO 449 STEP -1
620 J=140
630 IF I=477 THEN J=136
640 PRINT@I,CHR$(J);
650 NEXT I
660 FOR I=449 TO 65 STEP -32
670 J=133
680 IF I=449 THEN J=132
690 IF I=65 THEN J=129
700 PRINT@I,CHR$(J);
710 NEXT I
720 REM PRINT CURR MONTH,YEAR
730 ML=LEN(M$(MM))
740 PRINT@496-(ML+5)/2,M$(MM);",";YY;
750 REM PRINT CALENDAR DAYS
760 PP=0
770 FOR I=1 TO DM
780 IF I=1 THEN PP=99+(OF*4): GOTO840
790 IF (PP=187) AND (I>9) THEN PP=PP+39: GOTO840
800 IF (PP=123) OR (PP=186) OR (PP=187) THEN PP=PP+40: GOTO840
810 IF (PP=250) OR (PP=314) OR (PP=378) THEN PP=PP+40: GOTO 840
820 IF I=10 THEN PP=PP+3: GOTO840
830 PP=PP+4
840 B$=STR$(I)
850 PRINT@PP,B$;
860 NEXT I
870 REM MONITOR KEYBOARD FOR RESTART
880 A$=INKEY$
890 IF A$="" THEN 880
900 SOUND 220,3
910 GOTO160
```

"Camptown Races" CAMP

This is a computer rendition of the old Stephen Foster song "Camptown Races." It provides some variation by first playing the stock version of the song and then switching to a faster version. To keep you in the spirit of things, the message "DOO-DAHH" is displayed on the screen at appropriate times.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

See the writeup for BACH. This program uses the same principals to generate tones by the SOUND command in BASIC.

How This Program Works

The DATA statements at the beginning of the program hold the F and D values for the SOUND command. A READ statement reads one F and one D value and these are used in the following SOUND command. Two -4 values mark the end of the piece. When these are READ, the "DATA pointer" is reset to the beginning by a RESTORE and the piece is restarted.

Interspersed with the note values in the DATA statements are "flag" values of less than 0. These are used to flag when "DOO-DAHH" should be displayed on the screen and when to clear the screen after displaying the "DOO-DAHH" message.

The P variable is incremented each time through the song and serves to flip the speed from slow to fast and back to slow again.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. F values are a compromise between values suitable for the MC-10 and values suitable for the Color Computer.

CAMP "Camptown Races"

```
100 DATA 176,4,176,4,159,4,176,4,185,4,176,4,159,8,-1,-1,159,4,147,8,147,4
110 DATA -2,-2,159,4,147,8,147,4,-3,-3,176,4,176,4,159,4,176,4,185,4,176,4,159,8
120 DATA 147,8,159,4,147,4,133,8,133,4
130 DATA 176,8,176,4,159,4,176,4,185,4,176,4,159,8,-1,-1,159,4,147,8,147,4
140 DATA -2,-2,159,4,147,8,147,4,-3,-3,176,4,176,4,159,4,176,4,185,4,176,4,159,8
150 DATA 147,8,159,4,147,4,133,8,133,4
160 DATA 133,6,133,2,159,4,176,4,197,12,185,8,185,2,197,4
170 DATA 185,4,176,8,176,4,176,4,159,4,176,4,185,4
180 DATA 176,4,159,8,147,8,159,4,147,4,133,8,133,4,-4,-4
190 P=0
200 CLS
210 P=P+1
220 READ A,B: IF A=<0 THEN 260
230 IF P/2=INT(P/2) THEN B=B/2
240 SOUND A,B
250 GOTO 220
260 IF A=-1 THEN PRINT@266,"DOO-DAHH":GOTO 220
270 IF A=-2 THEN PRINT@301,"DOO-DAHH":GOTO 220
280 IF A=-3 THEN CLS: GOTO 220
290 RESTORE: GOTO 200
```

Coin Flipper CFLIP

I'll bet you never thought that you'd be using a couple of hundred dollars worth of computer equipment in place of flipping a coin, eh? But that's all this program does – it flips a coin, figuratively, to give you a "heads" or "tails" indication. Of course, we put in a little punch and made the display of heads or tails colorful – a lot more than you'd get from even flipping a silver dollar. But still in all, this is an expensive way to say yes or no.

How to Use This Program

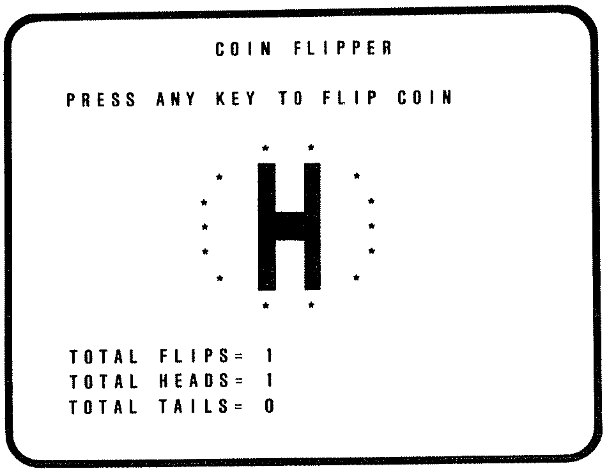
Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
         COIN FLIPPER
PRESS ANY KEY TO FLIP COIN
```

At this point you can press a key to "flip the coin." After pressing a key, you'll see the display shown in Figure CFLIP-1.



```
         COIN FLIPPER

PRESS ANY KEY TO FLIP COIN

TOTAL FLIPS= 1
TOTAL HEADS= 1
TOTAL TAILS= 0
```

Figure CFLIP-1. Coin Flipper Display

The message at the bottom of the screen indicates the total number of flips, total number of heads, and total number of tails:

```
TOTAL NUMBER OF FLIPS=  1
TOTAL NUMBER OF HEADS=  1
TOTAL NUMBER OF TAILS=  0
```

These counts will change if you keep on flipping the coin. For a large number of flips, the total number of heads and tails should be approximately the same.

Some Background on the Program

This is a simple program, but the concept of "random numbers" is not that easily explained. In fact, this program "simulates" a coin toss. The program "flips a coin" by generating a "pseudo-random" number by the RND(2) command. This command generates either a 1 or 2 value.

Pseudo-random numbers are not **random** numbers, because the sequence of numbers is **repeatable.** The same sequence of heads and tails will be generated each time the program is used, assuming that the program is loaded and executed directly after turning on the computer.

Random (unpredictable) numbers can't be easily generated in a computer. One method of generating them is to look at an external counter that is counting very rapidly. If the computer samples the count at irregular times, you'll get a random sampling of values. It's very similar to spinning a roulette wheel and coming up with a series of numbers.

A good pseudo-random number generator simulates randomness by making the predictable sequence extremely long before it repeats, but repeat it shall. In the case of the MC-10 or Color Computer the random sequence will probably repeat after hundreds of thousands of numbers. We'll let you investigate how long a typical sequence is!

A good pseudo-random number generator also will generate an even distribution of numbers over a long period of time. For example, if you were generating pseudo-random numbers from 1 through 1000, you'd expect to see about the same totals of 1s, 2s, 3s, 4s, and so forth after thousands of passes.

The coin tosser is therefore interesting to observe - you should note the sequence of numbers and look for an "even distribution" of heads and tails the more coin flips that are made.

How This Program Works

The heart of this program is in line 150, which is the pseudo-random number generator in the form of a BASIC RND statement. The remainder of the program draws a large "H" or "T" on the screen in color by using CHR$ strings of graphics blocks.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 T1=0: T2=0: T3=0
110 CLS: PRINT@10,"COIN FLIPPER"
120 PRINT@64,"PRESS ANY KEY TO FLIP COIN"
130 A$=INKEY$
140 IF A$="" THEN 130
150 PRINT@128:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
160 R=RND(2)
170 ON R GOSUB230,330
180 T1=T1+1
190 PRINT@385,"TOTAL FLIPS=";T1
200 PRINT@417,"TOTAL HEADS=";T2
210 PRINT@449,"TOTAL TAILS=";T3
220 GOTO130
230 REM - PRINT HEADS
240 PRINT@141,"*  *"
250 PRINT@170,"*  "+CHR$(191)+"  "+CHR$(191)+ "  *"
260 PRINT@201,"*   "+CHR$(191)+"  "+CHR$(191)+ "   *"
270 PRINT@233,"*   "+CHR$(191)+CHR$(191)+CHR$(191)+CHR$(191)+ "   *"
280 PRINT@265,"*   "+CHR$(191)+"  "+CHR$(191)+ "   *"
290 PRINT@298,"*  "+CHR$(191)+"  "+CHR$(191)+ "  *"
300 PRINT@333,"*  *"
310 T2=T2+1
320 RETURN
330 REM - PRINT TAILS
340 PRINT@141,"*  *"
350 PRINT@170,"* "+CHR$(175);
360 PRINT@173,CHR$(175)+CHR$(175)+CHR$(175)+CHR$(175)+CHR$(175)+" *"
370 PRINT@201,"*    "+CHR$(175)+CHR$(175)+ "    *"
380 PRINT@233,"*    "+CHR$(175)+CHR$(175)+ "    *"
390 PRINT@265,"*    "+CHR$(175)+CHR$(175)+ "    *"
400 PRINT@298,"*   "+CHR$(175)+CHR$(175)+ "   *"
410 PRINT@333,"*  *"
420 T3=T3+1
430 RETURN
```

Checkbook Program CHECKS

Every month it's the same thing – hours of agonizing cross-checking of dog-eared check stubs and bank statements, only to find a check amount that doesn't match the record in the check record! Isn't there an easier way? One way is your own CPA, but this is better, and less expensive – the CHECKS program. CHECKS will let you balance your checkbook with ease. It compiles three separate lists of outstanding checks, unrecorded deposits, and miscellaneous charges. The lists can be displayed and modified at will. When the lists are accurate, a final reconciliation does the balancing act better than the Great Tandini!

This program requires a 16K Color Computer or MC-10 with memory expansion pack.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

The screen should clear and you should see the following display:

```
         CHECKBOOK

SELECT ONE OF THE FOLLOWING:
 1. ENTER BALANCE FROM STATEMENT
 OUTSTANDING CHECKS:
 2. ENTER    3. LIST    4. DELETE
 UNRECORDED DEPOSITS:
 5. ENTER    6. LIST    7. DELETE
 MISCELLANEOUS CHARGES:
 8. ENTER    9. LIST   10. DELETE
11. COMPUTE NEW BALANCE
WHICH ONE?
```

You can now choose one of the menu options by entering a value of 1 through 11 followed by ENTER.

Entering Balance
This can be done at any time to enter the final balance from your bank statement:

```
BALANCE FROM STATEMENT?
```

Enter the amount in dollars and cents without commas as in:

```
BALANCE FROM STATEMENT? 100.98
```

The program will not accept amounts that are not in cents. After entering the amount, the "menu" of items will again be displayed. If you've made an error in this, you can reenter the amount at any time by selecting menu item 1 another time.

Entering Outstanding Checks
If you select option 2 you'll see:

```
ENTER OUTSTANDING CHECKS:
   CK #        AMOUNT
?
```

You can now enter a check number, followed by <ENTER> and then the check amount in dollars and cents, followed by <ENTER>. The program will not accept amounts unless they are in cents.

Enter as many checks as you'd like. After the last check, enter R followed by <ENTER> to return to the menu.

Listing Checks
To see the list of outstanding checks, select menu option 3. You'll see something like this:

```
LIST OUTSTANDING CHECKS:
   CK #        AMOUNT
      1234     567.89
      1236      56.00
      1237       3.00
      1239      15.89
```

Enter R followed by <ENTER> to return to the menu.

Deleting an Outstanding Check
If you've made an error in an outstanding check, you can delete the check by selecting menu item 4. You'll then see:

```
DELETE OUTSTANDING CHECKS:
   CK #        AMOUNT
?
```

You can now enter the check number and amount to be deleted. The program will then search the list of outstanding checks to find the check you specified. If it can't be found, you'll see the message "NO MATCH." If the check is found, you'll see the message "DELETED" displayed next to the check entry. To delete additional checks, enter a new number and amount. To get back to the menu, enter R followed by <ENTER>.

Entering, Listing, and Deleting Deposits
To enter, list, and delete unrecorded deposits, follow the directions above, except use menu items 5, 6, and 7. A "DATE" is substituted for a check number. The Date may be any string of characters, but would normally be something like 11-24 or 1/24. The same string must be specified for deletions. Again, dollar amounts must include cents.

Entering, Listing, and Deleting Charges
To enter, list, and delete miscellaneous charges, follow the directions above, except use menu items 8, 9, and 10. A "DATE" is substituted for a check number. The Date may be any string of characters, but would normally be something like 11-24 or 1/24. The same string must be used for deletions. Again, dollar amounts must include cents.

Computing the New Balance
When you have entered all outstanding checks, all unrecorded deposits, and all miscellaneous charges and have reviewed them and found them to be correct, enter menu item 11. You'll see a final reconciliation:

```
COMPUTE NEW BALANCE:
BALANCE FROM STMT      $  1007.98
+ UNRECORDED DEPOSITS      300.00
- OUTSTANDING CHECKS       257.99
- MISC. CHARGES              6.50

    NEW BALANCE =      $  1043.49
```

The new balance should agree with the balance in your checkbook. If it does not, go over the lists again, modify the lists as required, and run menu item 11 another time.

Background on the Program

The arithmetic in CHECKS is simple. The final balance on your check statement represents the balance at a certain point in time with the checks and deposits considered that are listed on the statement. To reconcile the statement with your own checkbook, you must take the statement balance, add in unrecorded deposits, subtract outstanding checks, and subtract miscellaneous charges such as check printing charges or credit card debits. The result should agree with your current checkbook balance.

CHECKS is primarily concerned with compiling three separate lists of data. The lists are maintained in one large string array of two dimensions. The first dimension is the check number or date, and the second is the check amount.

Maintaining the lists as strings simplifies data "formatting" when the data is displayed on the screen and amounts are adjusted to dollars and cents. If this were not done, the cents amounts might not be printed or might be printed as a decimal point and one digit, or a decimal point and more than two digits.

How This Program Works

All data is kept in a two-dimensional string array T$. The first 74 elements of the array are the outstanding checks that have been entered. The next 13 elements are the unrecorded deposits. The next 9 elements are miscellaneous charges. The first element of each entry is the check number or date, in string form. The second number is the dollars and cents amount, in string form. At this point there are always two digits for the "cents" as the format is verified upon data entry.

Entering checks, deposits, or charges involves the same operations – checking for dollars and cents format and then storing the number/date and amount in the proper section of the T$ array.

Listing checks, deposits, and charges also is done identically for the three types of lists. The proper section of array T$ is found and the items are listed out in the order that they were entered.

Deleting a check, deposit, or charge causes a search of the appropriate section of the array. Both the number/date and amount are compared to the deletion entry. If a "match" is found, the entry is deleted by "blanking out" the entry – making the entry a "null" string of zero length.

The final function, new balance, first prints the balance from the statement. It then searches the array and subtotals all unrecorded deposits. The outstanding checks and charges are subtotaled in similar fashion. Finally, the deposits are added to the balance figure and the checks and charges are subtracted. The result is the new balance.

Special Notes

1. This program will run on 16K Color Computer systems and MC-10 systems with memory expansion packs only.
2. There can a maximum of 74 checks, 13 deposits, and 9 miscellaneous charges.
3. If there is a miscellaneous **credit**, which sometimes occurs, enter a negative amount for the miscellaneous charge.

```
100 CLEAR 2000: DIM T$(100,2), LM(3,3)
110 DATA 1,74,0,76,89,75,91,100,90
120 FOR I=1 TO 100: T$(I,1)="": T$(I,2)="": NEXT I
130 FOR I=1 TO 3: READ LM(I,1), LM(I,2), LM(I,3): NEXT I
140 CLS: PRINT@11,"CHECKBOOK"
150 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
160 PRINT@97,"1. ENTER BALANCE FROM STATEMENT"
170 PRINT@129,"OUTSTANDING CHECKS:"
180 PRINT@161,"2. ENTER   3. LIST   4. DELETE"
190 PRINT@193,"UNRECORDED DEPOSITS:"
200 PRINT@225,"5. ENTER   6. LIST   7. DELETE"
210 PRINT@257,"MISCELLANEOUS CHARGES:"
220 PRINT@289,"8. ENTER   9. LIST  10. DELETE"
230 PRINT@320,"11. COMPUTE NEW BALANCE"
240 PRINT@352,"WHICH ONE";: INPUT A$
250 IF (VAL(A$)>0) AND (VAL(A$)<12) THEN A=VAL(A$): GOTO290
260 PRINT@448,"INVALID SELECTION--TRY AGAIN"
270 FOR I= 1 TO 300: NEXT I
280 PRINT@448,"": PRINT@360,"": GOTO240
290 IF A>1 AND A<5 THEN J=1: N1$="OUTSTANDING CHECKS": N2$="CK #"
300 IF A>4 AND A<8 THEN J=2: N1$="UNRECORDED DEPOSITS": N2$="DATE"
```

```
310 IF A>7 AND A<11 THEN J=3: N1$="MISC. CHARGES": N2$="DATE"
320 ON A GOSUB340,390,590,800,390,590,800,390,590,800,1070
330 GOTO140
340 REM -- ENTER BALANCE SUBR.
350 PRINT@416,"BALANCE FROM STATEMENT";: INPUT B$
360 L=LEN(B$): IF L<3 THEN PRINT@438,"": GOTO350
370 IFMID$(B$,L-2,1)<>"." THEN PRINT@438,"": GOTO350
380 B=VAL(B$): RETURN
390 REM--ENTER CKS,DEPOSITS,MISC. SUBR.
400 CLS: PRINT@32,"ENTER ";N1$;":": PRINT@67,N2$;"      AMOUNT"
410 P=64: PRINT@480,"ENTER R TO RETURN TO MENU";
420 IF LM(J,3)<LM(J,2) THEN 450
430 PRINT@160,"TOO MANY ";N1$
440 PRINT@192,"CHECKBOOK RUN ABORTED": STOP
450 IF P<416 THEN P=P+32: GOTO500
460 PRINT@480,"PRESS C TO CONTINUE      ";
470 B$=INKEY$
480 IF B$="C" THEN 400
490 GOTO470
500 PRINT@P,"";: INPUT B$
510 IF B$="R" THEN 580
520 LM(J,3)=LM(J,3)+1: T$(LM(J,3),1)= B$
530 PRINT@P+11,"";: INPUT B$
540 IF B$="R" THEN LM(J,3)=LM(J,3)-1: GOTO580
550 L=LEN(B$): IF L<3 THEN PRINT@P+11,"": GOTO530
560 IFMID$(B$,L-2,1)<>"." THEN PRINT@P+11,"": GOTO530
570 T$(LM(J,3),2)=B$: GOTO420
580 RETURN
590 REM--LIST CKS,DEPOSITS,MISC. SUBR.
600 IF LM(J,3)<LM(J,1) THEN CLS: PRINT@98,"NO ";N1$;" FOUND": GOTO720
610 I=LM(J,1)-1: GOSUB770: P=66
620 IF NOT (I<LM(J,3)) THEN 720
630 I=I+1: IF T$(I,1)="" AND T$(I,2)="" THEN 620
640 IF P<418 THEN P=P+32: GOTO700
650 PRINT@448,"PRESS C TO CONTINUE";: PRINT@480,"OR R TO RETURN TO MENU   ";
660 B$=INKEY$
670 IF B$="C" THEN GOSUB770: P=98: GOTO700
680 IF B$="R" THEN 760
690 GOTO660
700 PRINT@P+8-LEN(T$(I,1)),T$(I,1);
710 PRINT@P+18-LEN(T$(I,2)),T$(I,2): GOTO620
720 PRINT@480,"PRESS R TO RETURN TO MENU";
730 B$=INKEY$
740 IF B$="R" THEN 760
750 GOTO730
760 RETURN
770 CLS: PRINT@32,"LIST ";N1$;":"
780 PRINT@67,N2$;"      AMOUNT"
790 RETURN
800 REM--DELETE CKS,DEPOSITS,MISC. SUBR.
810 IF LM(J,3)=>LM(J,1) THEN 870
820 CLS: PRINT@98,"NO ";N1$;" FOUND": PRINT@130,"TO DELETE"
830 PRINT@480,"PRESS R TO RETURN TO MENU";
840 B$=INKEY$
850 IF B$="R" THEN 1060
860 GOTO840
870 CLS: PRINT@32,"DELETE ";N1$;":": PRINT@67,N2$;"      AMOUNT"
880 P=64: PRINT@480,"ENTER R TO RETURN TO MENU";
890 IF P<416 THEN P=P+32: GOTO940
900 PRINT@480,"PRESS C TO CONTINUE      ";
910 B$=INKEY$
```

```
920 IF B$="C" THEN 870
930 GOTO910
940 PRINT@P,"";: INPUT B$
950 IF B$="R" THEN 1060
960 X$=B$
970 PRINT@P+11,"";: INPUT B$
980 IF B$="R" THEN 1060
990 L=LEN(B$): IF L<3 THEN PRINT@P+11,"": GOTO970
1000 IFMID$(B$,L-2,1)<>"." THEN PRINT@P+11,"": GOTO970
1010 Y$=B$
1020 I=LM(J,1)-1: REM-BEGIN TBL MATCH
1030 IF NOT (I<LM(J,3)) THEN PRINT@P+22,"NO MATCH": GOTO890
1040 I=I+1: IF NOT (T$(I,1)=X$ AND T$(I,2)=Y$) THEN 1030
1050 PRINT@P+22,"DELETED": T$(I,1)="": T$(I,2)="": GOTO890
1060 RETURN
1070 REM -- NEW BALANCE SUBR.
1080 C=0: D=0: M=0
1090 CLS: PRINT@32,"COMPUTE NEW BALANCE:"
1100 PRINT@96,"BALANCE FROM STMT     $";
1110 I=INT(B*100): B$=STR$(I): L=LEN(B$)
1120 PRINT@126-L,LEFT$(B$,L-2)+"."+RIGHT$(B$,2)
1130 IF LM(2,3)<LM(2,1) THEN 1150
1140 FOR I=LM(2,1) TO LM(2,3): D=D+VAL(T$(I,2)): NEXT I
1150 PRINT@128,"+ UNRECORDED DEPOSITS";
1160 I=INT(D*100): B$=STR$(I): L=LEN(B$)
1170 PRINT@158-L,LEFT$(B$,L-2)+"."+RIGHT$(B$,2)
1180 IF LM(1,3)<LM(1,1) THEN 1200
1190 FOR I=LM(1,1) TO LM(1,3): C=C+VAL(T$(I,2)): NEXT I
1200 PRINT@160,"- OUTSTANDING CHECKS";
1210 I=INT(C*100): B$=STR$(I): L=LEN(B$)
1220 PRINT@190-L,LEFT$(B$,L-2)+"."+RIGHT$(B$,2)
1230 IF LM(3,3)<LM(3,1) THEN 1250
1240 FOR I=LM(3,1) TO LM(3,3): M=M+VAL(T$(I,2)): NEXT I
1250 PRINT@192,"- MISC. CHARGES";
1260 I=INT(M*100): B$=STR$(I): L=LEN(B$)
1270 PRINT@222-L,LEFT$(B$,L-2)+"."+RIGHT$(B$,2)
1280 N=B+D-C-M
1290 PRINT@260,"NEW BALANCE =     $";
1300 I=INT(N*100): B$=STR$(I): L=LEN(B$)
1310 PRINT@286-L,LEFT$(B$,L-2)+"."+RIGHT$(B$,2)
1320 PRINT@480,"PRESS R TO RETURN TO MENU";
1330 B$=INKEY$
1340 IF B$="R" THEN 1360
1350 GOTO1330
1360 RETURN
```

Morse Code Program CODE

Radio amateurs ("hams") use a code that is sometimes called Morse code, but is not the same as code originally used with telegraph stations. The code used by hams and by most radio services at present is called "International Morse Code." It represents letters, numbers, and special characters by short and long pulses called dots and dashes. Here are the code representations:

A	.-	N	-.	0	-----
B	-...	O	---	1	.----
C	-.-.	P	.--.	2	..---
D	-..	Q	--.-	3	...--
E	.	R	.-.	4	....-
F	..-.	S	...	5	.....
G	--.	T	-	6	-....
H	....	U	..-	7	--...
I	..	V	...-	8	---..
J	.---	W	.--	9	----.
K	-.-	X	-..-	-	-....-
L	.-..	Y	-.--	.	.-.-.-
M	--	Z	--..	/	-..-.

The Morse code program converts a text string of any of the above characters into International Morse code at speeds of 5 to 16 words per minute; you'll hear the dots and dashes as audio tones over the television speaker. The program can also generate a continuous stream of characters at random for code practice at speeds of 5 to 16 words per minute.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
        RUN
```

You should see the title and first prompt message as follows:

```
        MORSE CODE
SELECT ONE OF THE FOLLOWING:
     1. SEND AN INPUT STRING
     2. CODE PRACTICE

WHICH ONE?
```

You can now select one or the other by entering 1 or 2, followed by <ENTER>.

Sending an Input String

If you select the first function, you'll see the message:

```
SEND AN INPUT STRING ENTER SPEED?
```

You should now enter a code speed of from 5 through 16, representing rates of 5 through 16 words per minute. After the speed, press <ENTER>. The program will ignore any entries that are outside of this range.

The next prompt message is:

```
ENTER TONE-H(I),M(ED),L(OW)?
```

You can choose the most pleasant sounding tone for the code by choosing either H, M, or L, followed by <ENTER>. Next, you'll see:

```
ENTER STRING ?
```

You can now enter any of the characters in the table above, up to 249 characters. When you've entered the string you want converted to code, press <ENTER>.

The program will now transmit the characters as audio tones, at the speed you designated.

After the characters have been transmitted, you can restart the program or transmit another string by entering an R or <ENTER> as a response to:

```
ENTER R FOR RESTART, OR JUST <ENTER> FOR SAME
```

Code Practice
If you choose the second function, the program will generate a continuous stream of characters for code practice. You'll see a message for speed and tone as before, and then see a printer message:

```
CODE PRACTICE
ENTER SPEED? 16
ENTER TONE-H(I),M(ED),L(OW)? H
TO PRINTER (Y OR N)?
```

As the characters are transmitted, they'll be displayed on the screen. If you also want a printout on your system line printer, answer Y for the printer message.

Random characters from the above list will now be transmitted as audio tones and simultaneously displayed (and printed if you specified printing). A typical sequence might look like this:

```
RAT/M 7.Z03 .160L 7L1L6 VE27T RR
58W DRE93 1E7CH -548M /TMJ4 42B0
```

The characters are grouped in 5 letter groups to make it easier for you to compare the list with your own code practice copy. By the way, "Novice" class amateur radio licenses require a sending and receiving speed of 5 words per minute. The next grade, General class, requires a speed of 13 words per minute. The highest grade, Extra class, requires a speed of 20 words per minute.

To stop the code practice at any time, press any key and the program will restart.

Some Background on the Program

The characters above are a "subset" of all the characters defined for International Morse Code, but are the most commonly used.

The basic unit for generating code is a "dot" time. If a dot is considered one unit long, than a dash is three units long. There's a dot time between any dot or dash as well. Between individual characters, such as the three characters "TRS," there are three dot times, the same length as a dash. Between words, such as between "RADIO" and "SHACK" there are seven dot times.

A typical sequence is shown in Figure CODE-1, which shows the string of text:

```
THE QUICK BROWN FOX
```

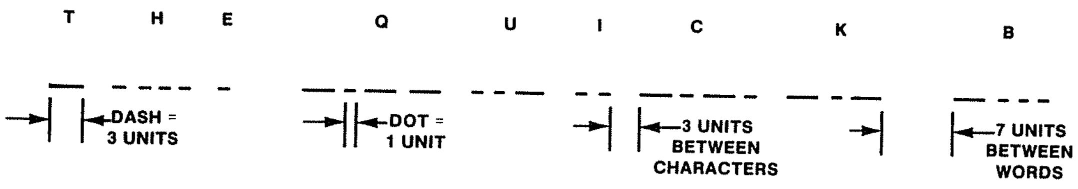


Figure CODE-1. Code Timing

The proportions just described are kept the same, regardless of the code speed. At 5 words per minute and at 25 words per minute, a dash would be three times as long as a dot.

How fast can one receive code? It's not too hard to learn how to receive code at 5 words per minute. A speed of 13 words per minute requires a little more effort, but it's also easy for most people. Radio Amateurs that have an Extra class license have to be able to receive and transmit at 20 words per minute. About 10% of radio amateurs fall in this category. People who are old time "brass pounders" can receive and transmit at speeds of 40 words per minute and above; certain Navy chiefs can be recording code on a typewriter at 35 words per minute, pause, take a good gulp of Navy coffee and continue from where they left off with no problem!

How the Program Works

Array C$ is a one-dimensional string array that holds 39 strings, representing the code characters

```
-./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The 39 strings are held in DATA statements at the beginning of the program, and stored in the array during the first part of the program.

Each string is literally a string of dots and dashes. The string for "L," for example is ".-../". The "/" character marks the end of the string.

The program finds the proper string during either of the two modes and "strips" the string of dots or dashes from left to right, one character at a time. Depending upon whether the character is a dot or dash, a tone that's one unit long or three units long is produced by the SOUND command.

Proper delays are put between dots/dashes, characters, and words. Because BASIC has some "overhead" and takes time, certain compensations are made for higher code speeds and for other timing. The speeds are approximately correct.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
90 CLEAR 300
100 DATA "-...-/",".-.-.-","-..-./","-----/"
110 DATA ".----/","..---/"
120 DATA "...--/","....-/","...../","-..../","--.../"
130 DATA "---../","----./"
140 DATA ".-/","-.../","-.-./","-../","./","..-./","--./"
150 DATA "..../","../",".---/","-.-/",".-../","--/"
160 DATA "-./","---/",".--./","--.-/",".-./",".../"
170 DATA "-/","..-","...-/",".--/","-..-/","-.--/","--../"
180 T=100
190 DIM C$(39)
200 FOR I=0 TO 38: READ A$: C$(I)=A$: NEXT I
210 CLS: PRINT @7,"MORSE CODE"
220 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
230 PRINT@100,"1. SEND AN INPUT STRING"
240 PRINT@132,"2. CODE PRACTICE"
250 PRINT@196,"WHICH ONE";:A$=INKEY$
260 IF (A$<>"1" AND A$<>"2") THEN 250
270 IF A$="2" THEN GOTO 400
280 CLS: PRINT @64,"SEND AN INPUT STRING"
290 GOSUB 570: GOSUB 620
300 PRINT @160,"ENTER STRING":INPUT A$
310 FOR I=1 TO LEN(A$)
320 V=ASC(MID$(A$,I,1))-52
330 IF V=-20 THEN GOSUB 800: GOTO 370
340 IF V>38 THEN 840
350 IF V<13 THEN V=V+7: IF V<0 THEN 840
360 GOSUB 720: GOSUB 800
370 NEXT I
380 GOSUB 670: IF R$="R" THEN GOTO 210
390 PRINT @ 192,"":PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:GOTO 300
400 CLS: PRINT @64,"CODE PRACTICE"
410 GOSUB 570: GOSUB 620
420 PRINT @160,"TO PRINTER (Y OR N)";: INPUT PR$
430 IF NOT(PR$="Y" OR PR$="N") THEN PRINT @245,"": GOTO 420
440 V=RND(90): IF V<45 THEN 440
450 IF (V>57 AND V<65) THEN 440
460 C=C+1: IF C-INT(C/6)*6=0 THEN GOSUB 800: PRINT " ";:GOTO 440
470 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
480 PRINT CHR$(V);: IF PR$="Y" THEN PRINT#-2,CHR$(V);
490 V=V-52: IF V<13 THEN V=V+7
```

```
500 GOSUB 720
510 IF INKEY$<>"" THEN 210
520 GOTO 440
530 SOUND T,S1:FOR K=0 TO S:NEXT K:RETURN
540 RETURN
550 SOUND T,S3:FOR K=0 TO S:NEXT K:RETURN
560 RETURN
570 PRINT @96,"ENTER SPEED";:INPUT S
580 IF (S<5 OR S>16) THEN PRINT @96,"": GOTO 570
590 IF S>10 THEN S=2*(S-10)+10
600 S=528/S:S1=S/24:S3=S/8
610 RETURN
620 PRINT@128,"ENTER TONE-H(I),M(ED),L(OW)";:INPUT T$
630 IF T$="L" THEN T=50
640 IF T$="M" THEN T=100
650 IF T$="H" THEN T=150
660 RETURN
670 PRINT @416,"ENTER R FOR RESTART, OR JUST"
680 PRINT @448,"<ENTER> FOR SAME";: INPUT R$
690 IF NOT (R$="R" OR R$="") THEN PRINT @465,"": GOTO 670
700 PRINT @ 416,"":PRINT
710 RETURN
720 D$=C$(V)
730 FOR J=1 TO 6
740 EL$=MID$(D$,J,1)
750 IF EL$="." THEN GOSUB 530
760 IF EL$="-" THEN GOSUB 550
770 IF EL$="/" THEN 790
780 NEXT J
790 RETURN
800 FOR K=0 TO 1*S: NEXT K
810 RETURN
820 FOR K=0 TO 9*S: NEXT K
830 RETURN
840 PRINT @416,"BAD CHARACTER--INPUT AGAIN"
850 FOR I=1 TO 600: NEXT I: PRINT @416,"": GOTO 390
```

Cryptography Program CRYPTO

Want to have your computer encode a string of characters for you? This program will act as a "secret decoder ring" and transform text such as

```
NOW IS THE TIME FOR ALL GOOD PROGRAMMERS
```

into code such as

```
DEM*2I:J36*J2C6*5EH*0BB*4EE7*FHE4H0CC6HI
```

Of course, it will also reverse the process and **decode** coded text back into plain text.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
      RUN
```

You should see the title and first prompt message as follows:

```
          CRYPTOGRAPHY

SELECT ONE OF THE FOLLOWING:
1. DECODE (CHANGE CODE TO
   ENGLISH) BY TABLE
2. ENCODE (CHANGE ENGLISH TO
   CODE) BY TABLE
WHICH ONE?
```

You can now select either the decode or encode function by entering a 1 or 2 followed by <ENTER>.

English-to-Code
Entering 2 starts the "encode" function. In this operation, plain English text is transformed into coded text.

The display is:

```
      ENGLISH-TO-CODE

ENTER TABLE:
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  0123456789
```

At this point, the "cursor" for the video display is positioned under the "A" in the text string. The program is waiting for you to construct a "code table" that represents the code character into which plain text is to be encoded.

Taking a simple case, suppose that we wanted to use a code that changed

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
```

into

```
0987654321ZYXWVUTSRQPONMLKJIHGFEDCBA
```

In this case you'd enter the second string directly below the A through 9 string like so:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0987654321ZYXWVUTSRQPONMLK
0123456789
JIHGFEDCBA
```

The program will then display:

```
ENTER TEXT TO BE TRANSLATED:
 (PRESS <ENTER> TO END)
```

You can now enter any plain text message, and the program will translate it into coded form. End the message with an <ENTER>. Type slowly, as the program takes some time to read in each character. Also, the program will not accept backspaces to delete a previously typed character. If we used the message

```
     SEND 25 CENTS FOR A SECRET DECODER RING
```

we'd end up with the following display:

```
ENTER TEXT TO BE TRANSLATED:
 (PRESS <ENTER> TO END)

 SEND 25 CENTS FOR A SECRET DEC
 R5W6*HE*75WQR*4VS*9*R57S5Q*657

 ODER RING
 V65S*SIW3

PRESS N FOR NEW TABLE,
 1 FOR DECODE, 2 FOR ENCODE
```

The encoded text for the message is below the original. Spaces are marked with an asterisk (*).

If you wanted to encode another message, you'd type 2, followed by <ENTER> at this point, and the "enter text" message would be displayed again with the program waiting for your next entry.

If you wanted to define a new table, you'd enter an N, followed by <ENTER>, and the program would restart so that you could enter a new code table.

If you wanted to decode a previously coded string, you'd enter 1, followed by <ENTER>. The decoding process is described in the next section.

Code-to-English
If you selected function 1, the program would process previously encoded text and change it into plain text in English. The display would look like this:

```
        CODE-TO-ENGLISH

ENTER TABLE:
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  0123456789
```

If you've just started the program, you'll have to enter the code characters as you did in the section above. However, if you've already entered them once, there's no need to reenter them. Just press <ENTER> in this case, and the program will use the previous code table.

After entering the code table, or pressing <ENTER> to use the existing table, you'll see:

```
ENTER TEXT TO BE TRANSLATED:
  (PRESS <ENTER> TO END)
```

At this point you can enter the text string to be decoded. Suppose we used the text that we encoded above – we'd have:

```
R5W6 HE 75WQR 4VS 9 R57S5Q 657
SEND*25*CENTS*FOR*A*SECRET*DEC

V65S SIW3
ODER*RING
```

You could use asterisks in place of spaces if you wanted.

After the text has been decoded, you can then enter a 1 followed by <ENTER> for another decoding action, 2 followed by <ENTER> for an encoding action, or N followed by <ENTER> to define a new code table.

Code Table

It's only necessary to define the code table once in the program. After you've defined the table once, the code translation will remain in force. Also, it's not necessary to define **every** character in the table. To skip a character, just type a space. The code character for the skipped character will remain at what it was previously. This is handy for defining new characters while leaving the old definitions alone as you might do in working code translations in puzzle books.

The code table is initially set up for an unencoded form – an "A" is an "A," a "B" a "B," and so forth.

Some Background on the Program

This is a very simple code translation, and it won't get you into the cryptography section of the CIA. This type of code can be broken very easily by analyzing the occurrences of each code character. In all languages certain letters occur more frequently than others. In English, E occurs most frequently with letters such as T, O, A, and I also occurring frequently and with letters such as X, Q, and Z occurring least frequently. If a coded string of characters is long enough and the code character "W" occurs frequently, there's a good chance that it's an "E" in plain text. Other letters can be found in a similar fashion.

You can modify this basic code by using nonsense characters interspersed with "good" text. For example, you might have

```
S1E7NSD 2W5 CDEGNFTGS F4OSR A SKEQCFRFE7T
```

where the message "SEND 25 CENTS FOR A SECRET" is every other letter.

You can also send text in 5-letter groups as in

```
SEND2 25CEN TSFOR ASECR ETDEC ODERR ING
```

This makes it harder for someone decoding the text as they can't recognize helpful words in encoded form, such as "A" and "THE."

Sophisticated codes use sequences of pseudo-random numbers as are generated in your MC-10 or Color Computer by the RND command. In these codes, the receiving and sending end would have to be "in sync," starting from the same pseudo-random number.

How the Program Works

The key to the program is array T$, which holds the code characters for the sequence A through Z and 0 through 9. This array is initialized to unencoded form (A-Z, 0-9). During the table entry portion of the program, your code string replaces each of the code positions in the table, unless you indicate that the code position should be left as is by entering a space for that position.

On encoding, the program translates plain text by finding the code character in the equivalent position in the table. The code character for "A" is in the first position, the code character for "B" in the second position, and so forth.

On decoding, the program searches the code table for the coded character, and then translates the position of the found code character into plain text. If the program was searching for the code character "Z" and found it in the second position of the code table, for example, it would know that the plain text version was a "B."

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. End any input of the code table or string to be translated by an <ENTER>. Don't use <ENTER> if you're not at the end of the text.

3. A warning beep is sounded at the end of each input line. You may lose characters if you type rapidly after the beep without pausing for code translation.

4. An asterisk is displayed if an input character is not found.

```
100 CLEAR 500: DIM T$(36,2)
110 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S
120 DATA T,U,V,W,X,Y,Z,0,1,2,3,4,5,6,7,8,9
130 FOR I=1 TO 36: READ T$(I,1): NEXT I
140 CLS: PRINT@10,"CRYPTOGRAPHY"
150 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
160 PRINT@96,"1. DECODE (CHANGE CODE TO"
170 PRINT@131,"ENGLISH) BY TABLE"
180 PRINT@160,"2. ENCODE (CHANGE ENGLISH TO"
190 PRINT@195,"CODE) BY TABLE"
200 PRINT@224,"WHICH ONE";: INPUT A$
210 IF A$="1" OR A$="2" THEN 250
220 PRINT@448,"INVALID SELECTION--TRY AGAIN"
230 FOR I=1 TO 300: NEXT I
240 PRINT@234,"": GOTO200
250 IF A$="1" THEN P$="CODE-TO-ENGLISH"
260 IF A$="2" THEN P$="ENGLISH-TO-CODE"
270 CLS: PRINT@8,P$
280 PRINT@64,"ENTER TABLE:"
290 FOR I=1 TO 36
300 IF I=1 THEN PP=98
310 IF I=27 THEN PP=194
320 PRINT@PP,T$(I,1);
330 PP=PP+1: NEXT I
340 FOR I=1 TO 36
350 IF I=1 THEN PP=129: B$="": C$=""
360 IF I=27 THEN PP=225: C$=""
370 IF B$=CHR$(13) THEN T$(I,2)=T$(I,1): GOTO450
380 PRINT@PP,"";: B$=INKEY$
390 IF B$="" THEN 380
400 IF B$=CHR$(13) THEN 370
410 C$=C$+B$: PRINT@PP+2-LEN(C$),C$
420 IF B$=CHR$(32) THEN T$(I,2)=T$(I,1): GOTO440
430 T$(I,2)=B$
440 PP=PP+1
450 NEXT I
460 IF A$="1" THEN A1=2: A2=1
470 IF A$="2" THEN A1=1: A2=2
480 CLS: PRINT@0,"ENTER TEXT TO BE TRANSLATED:"
490 PRINT@33,"(PRESS <ENTER> TO END)"
500 PL=32
510 IF PL<384 THEN PL=PL+64: GOTO550
520 PRINT@480,"PRESS ANY KEY TO CONTINUE";: B$=INKEY$
530 IF B$="" THEN 520
540 CLS: PL=0
550 PP=0: C$=""
560 PRINT@PL+PP,"";: B$=INKEY$
570 IF B$="" THEN 560
580 IF B$=CHR$(13) THEN GOSUB700: GOTO620
590 C$=C$+B$: PRINT@PL+PP+2-LEN(C$),C$
600 IFLEN(C$)>29 THEN GOSUB700: GOTO510
610 PP=PP+1: GOTO560
620 PRINT@448,"PRESS N FOR NEW TABLE,"
630 PRINT@480,"1 FOR DECODE, OR 2 FOR ENCODE";: A$=INKEY$
```

```
640 IF A$="" THEN 630
650 IF A$="N" THEN 140
660 IF A$="1" OR A$="2" THEN 460
670 IF A$="1" OR A$="2" THEN 460
680 GOTO630
690 REM - PRINT TRANSLATED TEXT
700 L=LEN(C$): IF L<2 THEN 810
710 SOUND 125,3
720 PL=PL+32: PP=0: D$=""
730 FOR I=1 TO L
740 J=0: B$=MID$(C$,I,1)
750 IF J<36 THEN J=J+1: GOTO770
760 B$="*": GOTO790
770 IF B$=T$(J,A1) THEN B$=T$(J,A2): GOTO790
780 GOTO750
790 D$=D$+B$: PRINT@PL+PP+2-LEN(D$),D$
800 PP=PP+1: NEXT I
810 RETURN
```

Binary, Decimal, and Hexadecimal Converter Program DECCONV

Hexed by hexadecimal? The DECCONV program is used to convert from one number "base" to another. All computers use binary numbers internally to represent data. Machine- and asssembly-language programs use binary numbers and their close relative, "hexadecimal" numbers, to represent data. Even some BASIC operations on the Color Computer use binary or hexadecimal values.

This program will enable you to rapidly convert from ordinary decimal numbers to their binary or hexadecimal equivalents or to reverse the process, in addition to converting between binary and hexadecimal.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
  DECIMAL TO BASE X CONVERTER

CONVERT FROM:
 1 . DECIMAL TO BINARY
 2 . BINARY TO DECIMAL
 3 . DECIMAL TO HEXADECIMAL
 4 . HEXADECIMAL TO DECIMAL
 5 . BINARY TO HEXADECIMAL
 6 . HEXADECIMAL TO BINARY
WHICH ONE?
```

You should then enter one of the menu items, from 1 to 6. After you select a menu item, you'll see a prompt message for the number to be converted.

Decimal to Binary

In this case you'll see:

```
NUMBER IN DECIMAL?
```

Enter the decimal number, followed by <ENTER>. The decimal number can be any number from 0 through 65,535. After you enter the number you'll see the result:

```
NUMBER IN DECIMAL? 65535
NUMBER IN BIN?     1111111111111111
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from decimal to binary, or enter R followed by <ENTER> to get to another type of conversion.

Binary to Decimal
In this case you'll see:

```
NUMBER IN BIN?
```

Enter the binary number, followed by <ENTER>. The binary number can be any number from 0 through 1111111111111111 (decimal 65,535). After you enter the number you'll see the result:

```
NUMBER IN BINARY? 11001111
NUMBER IN DEC? 207
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from binary to decimal, or enter R followed by <ENTER> to get to another type of conversion.

Decimal to Hexadecimal
In this case you'll see:

```
NUMBER IN DEC?
```

Enter the decimal number, followed by <ENTER>. The decimal number can be any number from 0 through 65,535. After you enter the number you'll see the result:

```
NUMBER IN DECIMAL? 10000
NUMBER IN HEX?     2710
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from decimal to hexadecimal, or enter R followed by <ENTER> to get to another type of conversion.

Hexadecimal to Decimal
In this case you'll see:

```
NUMBER IN HEX?
```

Enter the hexadecimal number, followed by <ENTER>. The hexadecimal number can be any number from 0 through FFFF (decimal 65,535). After you enter the number you'll see the result:

```
NUMBER IN HEX? AAAA
NUMBER IN DEC? 43690
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from hexadecimal to decimal or enter R followed by <ENTER> to get to another type of conversion.

Binary to Hexadecimal
In this case you'll see:

```
NUMBER IN BIN?
```

Enter the binary number, followed by <ENTER>. The binary number can be any number from 0 through 1111111111111111 (decimal 65,535). After you enter the number you'll see the result:

```
NUMBER IN DECIMAL? 110001100
NUMBER IN HEX?     18C
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from binary to hexadecimal, or enter R followed by <ENTER> to get to another type of conversion.

Hexadecimal to Binary
In this case you'll see:

```
NUMBER IN HEX?
```

Enter the hexadecimal number, followed by <ENTER>. The decimal number can be any number from 0 through FFFF (65,535). After you enter the number you'll see the result:

```
NUMBER IN HEX? A0AF
NUMBER IN BIN? 1010000010101111
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

You can now press <ENTER> to convert another number from hexadecimal to binary, or enter R followed by <ENTER> to get to another type of conversion.

Input Errors
If you enter an invalid number, you'll get an error message, as in:

```
INVALID HEX NUMBER--TRY AGAIN
```

Invalid numbers are decimal numbers less than 0 or greater than 65,535; binary numbers greater than 1111111111111111 (16 ones, or 65,535 decimal); or hexadecimal numbers larger than FFFF (decimal 65,535). You'll also get an error if you use anything other than 0-9 and A through F for hexadecimal digits.

Some Background on the Program

Binary numbers represent data by only two digits, a 0 or a 1. The position of the binary digits, or bits, in the binary number represent powers of two. The binary number 1010, for example, represents:

```
1 x 2↑3 + 0 x 2↑2 + 1 x 2↑1 + 0 x 2↑0 or
    8   +    0    +    2    +    0    = 10
```

In hexadecimal notation, the hexadecimal digits are 0 through 9 and A through F, representing powers of 16. The hexadecimal number A0B7, for example, represents:

```
A x 16↑3 + 0 x 16↑2 + B x 16↑1 + 7 x 16↑0 or
  40,960 +     0    +   176    +    7     = 41,136
```

Hexadecimal notation is usually used as a kind of shorthand to represent binary data. Four binary digits can be converted to one hexadecimal digit as follows:

0000=0	0100=4	1000=8	1100=C
0001=1	0101=5	1001=9	1101=D
0010=2	0110=6	1010=A	1110=E
0011=3	0111=7	1011=B	1111=F

How This Program Works

This program converts from decimal to binary or hexadecimal by a method known as "divide by the base and save the remainders." In this method the decimal number is divided by either 2 for binary or 16 for hexadecimal. The remainder of the divide is saved. The quotient result is saved and divided by 2 or 16 again. When the quotient becomes 0, all of the remainders, in reverse order, represent the binary or hexadecimal number.

This is not an easy method to understand, but look at this example:

```
What is 100 decimal in binary?:
100/2=50 reminder 0
50/2 =25 remainder 0
25/2 =12 remainder 1
12/2 =6 remainder 0
6/2  =3 remainder 0
3/2  =1 remainder 1
1/2  =0 remainder 1
```

The remainders, in reverse order, are 1100100, the binary equivalent to decimal 100.

The program converts from binary to hexadecimal by a method known as "double-dabble" or "hexa-dabble." In this method each binary or hex digit is multiplied by 2 or 16 and added to the next digit to the right. The sum is then multiplied by 2 or 16 and the result added to the next rightmost digit. When the rightmost digit has been added in, the conversion is complete. Here's an example:

```
What is 10001111 binary in decimal?:
  1 x 2 =   2 + 0 = 2
  2 x 2 =   4 + 0 = 4
  4 x 2 =   8 + 0 = 8
  8 x 2 =  16 + 1 =17
 17 x 2 =  34 + 1 =35
 35 x 2 =  70 + 1 =71
 71 x 2 =142 + 1 =143
```

To convert between binary and hexadecimal, the program first converts to a decimal number and then converts from decimal to binary or hexadecimal.

The DATA in the DATA statements represent the six sets of possible conversions. The first number in each set is the maximum number of digits allowed in the "from" number base, the next number is the "from" base, the next number is the "to" base, and the two remaining strings are the "from" and "to" bases in text form. The DATA is stored in arrays T and T$ for ease of use.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. Don't use commas in any of the answers. The program does not expect commas and will ignore digits after the first comma.

```
100 DIM R$(16), T(6,3), T$(6,2)
110 DATA 5,10,2,DECIMAL,BINARY,16,2,10,BINARY,DECIMAL
120 DATA 5,10,16,DECIMAL,HEXADECIMAL
130 DATA 4,16,10,HEXADECIMAL,DECIMAL,16,2,16,BINARY,HEXADECIMAL
140 DATA 4,16,2,HEXADECIMAL,BINARY
150 FOR I=1 TO 6
160 READ T(I,1), T(I,2), T(I,3), T$(I,1), T$(I,2): NEXT I
170 CLS: PRINT@2,"DECIMAL TO BASE X CONVERTER"
180 PRINT@64,"CONVERT FROM:"
190 FOR I=1 TO 6
200 PRINT@64+(I*32),I;". ";T$(I,1);" TO ";T$(I,2): NEXT I
210 PRINT@288,"WHICH ONE";: INPUT A
220 IF (A>0) AND (A<7) THEN 250
230 PRINT@352,"INVALID SELECTION--TRY AGAIN"
240 FOR I=1 TO 300: NEXT I: GOTO170
250 PRINT@352,"NUMBER IN ";LEFT$(T$(A,1),3);: INPUT A$
260 L=LEN(A$)
270 IF L<=T(A,1) THEN 310
280 PRINT@448,"INVALID ";LEFT$(T$(A,1),3);" NUMBER--TRY AGAIN"
290 FOR I=1 TO 300: NEXT I
300 PRINT@448,"";:PRINT:PRINT@367,"": GOTO250
310 ON A GOSUB380,720,380,720,720,720
320 IF ER$="Y" THEN ER$="": GOTO280
330 PRINT@416,"ENTER R FOR RESTART, OR JUST"
340 PRINT@448,"<ENTER> FOR SAME";: INPUT A$
350 IF A$="R" THEN 170
```

```
360 IF A$="" THEN PRINT@416,"":PRINT@448,"":PRINT@367,"":PRINT@399,"":GOTO250
370 GOTO330
380 REM DECIMAL SUBROUTINE
390 FOR I=1 TO L
400 N$=MID$(A$,I,1)
410 IF NOT (N$>="0" AND N$<="9") THEN 460
420 NEXT I
430 N1=VAL(A$)
440 IF N1>65535 THEN 460
450 GOSUB480: GOTO470
460 ER$="Y"
470 RETURN
480 REM CONVERT DEC TO BIN/HEX SUBROUTINE
490 FOR I=1 TO 16
500 R$(I)="": NEXT I
510 FOR I=16 TO 1 STEP -1
520 N2=INT(N1/T(A,3))
530 R$(I)=STR$(N1-(T(A,3)*N2))
540 IF N2=0 THEN 560
550 N1=N2: NEXT I
560 L=I
570 REM PRINT DEC TO BIN/HEX
580 PRINT@384,"NUMBER IN ";LEFT$(T$(A,2),3);"  ";
590 P=398
600 FOR I=L TO 16
610 N2=LEN(R$(I))
620 N$=MID$(R$(I),2,N2)
630 IF N$="10" THEN N$="A"
640 IF N$="11" THEN N$="B"
650 IF N$="12" THEN N$="C"
660 IF N$="13" THEN N$="D"
670 IF N$="14" THEN N$="E"
680 IF N$="15" THEN N$="F"
690 P=P+1: PRINT@P,N$;
700 NEXT I
710 RETURN
720 REM BIN/HEX SUBROUTINE
730 N1=0: N2=0
740 FOR I=1 TO L
750 N$=MID$(A$,I,1)
760 IF (N$="0") OR (N$="1") THEN 820
770 IF NOT (A=4 OR A=6) THEN 890
780 IF (N$>="2") AND (N$<="9") THEN 820
790 N3=ASC(N$)-55
800 IF (N3<=15) THEN 830
810 GOTO890
820 N3=VAL(N$)
830 N2=N1
840 N1=(N2*T(A,2))+N3
850 NEXT I
860 IF (A=5) OR (A=6) THEN GOSUB480: GOTO900
870 PRINT@384,"NUMBER IN ";LEFT$(T$(A,2),3);" ";N1
880 GOTO900
890 ER$="Y"
900 RETURN
```

Diet Calculator DIETCAL

Is that chair next to your computer desk sagging somewhat? Does a marble dropped anywhere in your house seek you out? DIETCAL calculates a weight loss schedule based on your current weight, amount of daily activity, age, and daily caloric intake. It isn't meant to replace physician's advice, but is meant as a general rule of thumb for average weight loss in cases without medical complications.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
        RUN
```

The screen should clear and you should see the following display:

```
        DIET CALCULATOR

ENTER AGE, 20 TO 65?
```

You should now enter a two-digit age, followed by <ENTER>. After you've entered the age, you should see:

```
WEIGHT NOW, 100 TO 250?
```

Enter your current weight, from 100 through 250, followed by ENTER. You should now see:

```
ENTER AMOUNT OF DAILY ACTIVITY,
I FOR INACTIVE, S FOR SOME,
M FOR MODERATE, AND G FOR GREAT?
```

Enter a single character that best descibes your physical activity, followed by <ENTER>. You should now see:

```
ENTER CALORIES PER DAY FOR
DIET, 1000 TO 4999?
```

Enter the number of calories you intend to use on your diet program, followed by <ENTER>. You should now see:

```
TO PRINTER (Y OR N)?
```

Enter either Y for printer output, or N for no printer output, followed by <ENTER>.

If you've selected the printing option, you should now "ready" the printer by turning on the power, loading paper, and selecting the "on-line" condition.

You should now see the diet schedule displayed on the screen and optionally printed. It will look something like this:

```
CHART OF CALCULATED DAILY
WEIGHTS AT 1500 CALORIES/DAY

DAY          NET LOSS       NEW WEIGHT

 1             .42            199.5
 2             .42            199.1
 3             .42            198.7
 4             .42            198.2
 5             .42            197.8
 6             .41            197.4
```

The list continues until the new weight goes below 100 or above 250. (Yes, the program also works as a "weight gain" program!)

Some Background on the Program

The American Medical Association (AMA) says that a person that has moderate activity requires about 15 calories per pound per day. As an example, suppose that you weighed 200 pounds. You'd require 200 x 15 or 3000 calories per day to maintain your 200 pounds. A person leading a sedentary (very inactive) life requires about 12 calories per pound per day. In the 200 pound example, you'd require 2400 calories per day to maintain your weight.

If you eat less than the number of calories required to maintain your weight, you'll lose weight. If you eat more than the number of calories required to maintain your weight, you'll gain weight. How fast will you gain or lose weight? A pound of body fat is roughly equivalent to 3500 calories. If you eat 3500 calories less than what is required to maintain your body weight you'll lose one pound. For example, suppose that you need 3000 calories per day to maintain your weight of 200 pounds. If you eat 2000 calories per day for one week, you've deprived yourself of 7000 calories and will have lost 2 pounds.

DIETCAL calculates your daily weight loss based on your current weight and the calories per day in your diet. It also makes a minor adjustment based on daily activity and age. The age adjustment subtracts a maximum of 20 percent of the calories per pound figure for age 65, based on less activity with age.

How This Program Works

Most of this program is related to screen formatting and data input. The computational portion first finds the number of calories required to maintain the current weight by multiplying current weight (variable CW) by variable MC. Variable MC is the number of calories required per pound for a given amount of activity (variable AC) and age (variable AG). The calories per day figure (variable CD) is then subtracted from the CW*MC figure. When the result is divided by 3500, the new figure is the weight loss (positive) or gain (negative). This figure is then subtracted from the current weight to produce the next "current weight."

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 CLS: PRINT@5,"DIET CALCULATOR"
110 PRINT@64,"ENTER AGE, 20 TO 65";:INPUT AG$
120 IF (VAL(AG$)>=20) AND (VAL(AG$)<=65) THEN AG=VAL(AG$): GOTO 140
130 PRINT@64,"": GOTO 110
140 PRINT@96,"WEIGHT NOW, 100 TO 250";: INPUT CW$
150 IF (VAL(CW$)>=100) AND (VAL(CW$)<=250) THEN CW=VAL(CW$): GOTO 170
160 PRINT@96,"": GOTO 140
170 PRINT@128,"ENTER AMOUNT OF DAILY ACTIVITY, I FOR INACTIVE, S FOR SOME,"
180 PRINT@192,"M FOR MODERATE, AND G FOR GREAT";: INPUT AC$
190 IF AC$="I" THEN AC=12: GOTO 240
200 IF AC$="S" THEN AC=13.5: GOTO 240
210 IF AC$="M" THEN AC=15: GOTO 240
220 IF AC$="G" THEN AC=16.5: GOTO 240
230 PRINT@192,"": GOTO 180
240 MC=AC*(1-(AG-20)*(.2/45))
250 PRINT @224,"ENTER CALORIES PER DAY FOR ";
260 PRINT@256,"DIET, 1000 TO 4999";: INPUT CD$
270 IF (VAL(CD$)>999) AND (VAL(CD$)<5000) THEN CD=VAL(CD$): GOTO 290
280 PRINT@256,"": GOTO 250
290 PRINT @288,"TO PRINTER (Y OR N)";: INPUT PR$
300 IF NOT (PR$="Y" OR PR$="N") THEN PRINT@288,"": GOTO 290)
310 DA=1
320 A$="": GOSUB 510
330 A$="CHART OF CALCULATED DAILY"+CHR$(13)
340 A$=A$+"WEIGHTS AT"+STR$(CD)+" CALORIES/DAY": GOSUB 510
350 A$="": GOSUB 510
360 A$="DAY    NET LOSS  NEW WEIGHT": GOSUB 510
370 A$="": GOSUB 510
380 CL=(CW*MC)-CD
390 NL=CL/3500
400 CW=CW-NL
410 A$=STR$(DA): IF DA<10 THEN A$=A$+"      ": GOTO 440
420 IF (DA>9) AND (DA<100) THEN A$=A$+"     ": GOTO 440
430 A$=A$+"    "
440 A$=A$+LEFT$(STR$(NL),4)+"      "+LEFT$(STR$(CW),6): GOSUB 510
450 DA=DA+1
460 IF (CW<100) OR (CW>250) THEN 480
470 GOTO 380
480 PRINT "PRESS R TO RESTART";
490 A$=INKEY$: IF A$<>"R" THEN 490
500 GOTO 100
510 PRINT A$
520 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
530 IF PR$="Y" THEN PRINT#-2,A$
540 RETURN
```

French Drill Program FRENCH

Parlez Vous Francais? Non? This program is a tutorial program that will drill you in translating French nouns to English and vice versa. We've included a short list of words for 4K MC-10 systems, but you can add as many as you would like by adding to the DATA statements at the program end. In 32K Color Computer systems, the maximum number of French words would be approximately 500.

How to Use This Program

See the German Drill Program GERMAN for instructions on using the program and program content. The French version works exactly the same way.

```
100 CLEAR 500: CLS: DIM T$(50,2), N(5)
110 L=0
120 IF L<50 THEN 130
125 PRINT@64,"TOO MANY TABLE ENTRIES,"
127 PRINT@96,"FRENCH DRILL ABORTED": STOP
130 L=L+1
140 READ T$(L,1), T$(L,2)
150 IF (T$(L,1)="-1") OR (T$(L,2)="-1") THEN L=L-1: GOTO170
160 GOTO120
170 R=0: T=0
180 CLS: PRINT@7,"FRENCH NOUN DRILL"
190 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
200 PRINT@96,"1. TRANSLATE FROM FRENCH-TO"
210 PRINT@131,"ENGLISH"
220 PRINT@160,"2. TRANSLATE FROM ENGLISH-TO-"
230 PRINT@195,"FRENCH"
240 PRINT@224,"WHICH ONE";: INPUT A$
250 IF A$="1" OR A$="2" THEN A=VAL(A$): GOTO290
260 PRINT@448,"INVALID SELECTION--TRY AGAIN"
270 FOR I=1 TO 300: NEXT I
280 PRINT@234,"": GOTO240
290 IF A$="1" THEN B=2: N$="FRENCH-TO-ENGLISH"
300 IF A$="2" THEN B=1: N$="ENGLISH-TO-FRENCH"
310 CLS: PRINT@7,N$
320 FOR I=0 TO 5
330 N(I)=RND(L)
340 IF N(I)=0 THEN 330
350 J=0
360 IF (N(I)=N(J)) AND (I<>J) THEN 330
370 IF J<(I-1) THEN J=J+1: GOTO360
380 NEXT I
390 I=RND(5)
400 IF I<1 THEN 390
410 PRINT@64,T$(N(0),A)
420 N(I)=N(0): N(0)=I
430 PRINT@128,"CHOOSE ONE OF THE FOLLOWING:"
440 J=128
450 FOR I=1 TO 5
460 J=J+32: B$=STR$(I)
470 PRINT@J,MID$(B$,2,1)+".  "+T$(N(I),B)
480 NEXT I
490 PRINT@320,"WHICH ONE";: INPUT B$
500 IF (B$>"0") AND (B$<"6") THEN 520
510 PRINT@329,"": GOTO490
520 T=T+1
```

```
530 IFVAL(B$) <> N(0) THEN 560
540 R=R+1
550 PRINT@384,R;"OUT OF";T;"CORRECT!": GOTO580
560 PRINT@384,"NO! THE CORRECT ANSWER IS";N(0)
580 PRINT@416,"ENTER R FOR RESTART, OR JUST"
590 PRINT@448,"<ENTER> FOR SAME";: INPUT B$
600 IF NOT (B$="R" OR B$="") THEN PRINT@465,"": GOTO590
610 IF B$="R" THEN GOTO170
620 GOTO310
1000 DATA "LES FRUITS","THE FRUIT","LE VIN","THE WINE","LE PAIN","THE BREAD"
1010 DATA "LA CIEL","THE SKY","LA LUNE","THE MOON","LE SOLEIL","THE SUN"
1020 DATA "LA TETE","THE HEAD","LES MAINS","THE HANDS","LE NEZ","THE NOSE"
1030 DATA "L'OEIL","THE EYE","L'OREILLE","THE EAR","L'AVION","THE AIRPLANE"
1040 DATA "LA SOEUR","THE SISTER","LA MERE","THE MOTHER","LA MER","THE SEA"
1050 DATA "LE PETIT DEJEUNER","THE BREAKFAST","LE DINER","THE DINNER"
1060 DATA "LE PRIX","THE PRIZE","LE CAFE","THE COFFEE","LA PLUME","THE PEN"
1070 DATA "LE SAVON","THE SOAP","LE CHAPEAU","THE HAT","L'EAU","THE WATER"
1100 DATA -1,-1
```

German Drill Program GERMAN

Sprechen Sie Deutsch? No? This program is a tutorial program that will drill you in translating German nouns to English and vice versa. We've included a short list of words for 4K MC-10 systems, but you can add as many as you would like by adding to the DATA statements at the program end. In 32K Color Computer systems, the maximum number of German words could be approximately 500.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
     GERMAN NOUN DRILL

SELECT ONE OF THE FOLLOWING:
1. TRANSLATE FROM GERMAN-TO-
   ENGLISH
2. TRANSLATE FROM ENGLISH-TO-
   GERMAN
WHICH ONE?
```

You can now select one or the other by entering 1 or 2, followed by <ENTER>.

If you select German-to-English, you'll see the first multiple choice selection, which will look something like this:

```
     GERMAN-TO-ENGLISH

DAS MADCHEN

CHOOSE ONE OF THE FOLLOWING:
1. THE MAN
2. THE EVENING
3. THE APPLE
4. THE GIRL
5. THE SHOE
WHICH ONE?
```

Entering 1 through 5, followed by <ENTER> will record your choice, and the program will display either the correct answer or the total number correct:

```
 1 OUT OF 1 CORRECT! ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

Pressing <ENTER> will display the next set of multiple choice entries. Entering R followed by <ENTER> will bring you back to the program start so that you can select the alternate drill.

Selecting English-to-German works exactly the same way, except that the multiple choice entries give German nouns, one of which has to be selected for the English equivalent at the top of the screen.

To add entries to the program, do this:

1. Add a DATA statement before the last DATA statement of DATA -1,-1. You can add as many data statements as you have room for in memory. Each data statement has this form:

   line# DATA "(German noun)", "(English equivalent)"

   You can put several nouns in one line, as long as each German noun is followed by its English equivalent in the same line.

   Be sure to put double quotation marks around each German noun and each English equivalent, and be certain to include a comma between each entry.

   You can add as many DATA statements as you'd like, as long as you have enough memory, and as long as the last DATA statement is DATA -1,-1.

2. Change the DIM command in line 100 for the number of entries that you have. If you have 100 DATA statements total and each DATA statement has two German/English entries, then you'll have 200 entries total. Change the DIM command in this case to DIM T$(100,2). The "2" in T$(100,2) doesn't change.

3. Delete line 120 by entering 120 from BASIC.

4. Run the program. If you get the title message, then you have enough memory for the entries you've added. You may get an "OM" or "Out of Memory" error. In this case you'll have to delete some of the entries you've added.

5. When the program starts successfully, save the program on cassette by the CSAVE command or on disk by the SAVE command.

Some Background on the Program

Computers make very efficient instructors for repetitive material such as this. There are two main limitations for these types of programs, memory and speed.

Each character of a text string (as in the DATA statements) takes up one byte of memory. The string "DAS MADCHEN", for example, takes up 11 bytes of memory, as it has 11 characters, counting the blanks. Assuming that the average German/English string has about 22 characters, we can get about 45 German/English equivalents in 1000 bytes of memory. However, if a string array is used, this will also take up an equivalent amount of memory, if data is transferred from DATA statements to an array.

Speed is another consideration for a program such as this. It takes a long time to "scan through" a list of data, such as a long sequence of DATA statements. Reading random entries from DATA statements would mean starting at the beginning of the list (by doing a BASIC RESTORE) for each entry. For this reason, the entries in DATA statements are put into a string array to speed up the access, even though this uses additional memory.

How the Program Works

Array T$ is a two-dimensional string array that holds the German noun in the first element and the English equivalent in the second element. The array is initialized from the entire set of DATA statements that define the German/English nouns to be used.

For the multiple choice translation, a random German or English noun is computed by using the RND command with L, the number of German/English entries. The program knows the equivalent noun, as it is the adjacent entry in the array. The program then finds 4 multiple choice entries at random by using RND(L) again and displays the 4 entries along with the proper selection.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 CLEAR 500: CLS: DIM T$(50,2), N(5)
110 L=0
120 IF L<50 THEN 130
125 PRINT@64,"TOO MANY TABLE ENTRIES,"
127 PRINT@96,"GERMAN DRILL ABORTED": STOP
130 L=L+1
140 READ T$(L,1), T$(L,2)
150 IF (T$(L,1)="-1") OR (T$(L,2)="-1") THEN L=L-1: GOTO170
160 GOTO120
170 R=0: T=0
180 CLS: PRINT@7,"GERMAN NOUN DRILL"
190 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
200 PRINT@96,"1. TRANSLATE FROM GERMAN-TO-"
210 PRINT@131,"ENGLISH"
220 PRINT@160,"2. TRANSLATE FROM ENGLISH-TO-"
230 PRINT@195,"GERMAN"
240 PRINT@224,"WHICH ONE";: INPUT A$
250 IF A$="1" OR A$="2" THEN A=VAL(A$): GOTO290
260 PRINT@448,"INVALID SELECTION--TRY AGAIN"
270 FOR I=1 TO 300: NEXT I
280 PRINT@234,"": GOTO240
290 IF A$="1" THEN B=2: N$="GERMAN-TO-ENGLISH"
300 IF A$="2" THEN B=1: N$="ENGLISH-TO-GERMAN"
310 CLS: PRINT@7,N$
320 FOR I=0 TO 5
330 N(I)=RND(L)
340 IF N(I)=0 THEN 330
350 J=0
360 IF (N(I)=N(J)) AND (I<>J) THEN 330
370 IF J<(I-1) THEN J=J+1: GOTO360
380 NEXT I
390 I=RND(5)
```

```
400 IF I<1 THEN 390
410 PRINT@64,T$(N(0),A)
420 N(I)=N(0): N(0)=I
430 PRINT@128,"CHOOSE ONE OF THE FOLLOWING:"
440 J=128
450 FOR I=1 TO 5
460 J=J+32: B$=STR$(I)
470 PRINT@J,MID$(B$,2,1)+". "+T$(N(I),B)
480 NEXT I
490 PRINT@320,"WHICH ONE";: INPUT B$
500 IF (B$>"0") AND (B$<"6") THEN 520
510 PRINT@329,"": GOTO490
520 T=T+1
530 IFVAL(B$) <> N(0) THEN 560
540 R=R+1
550 PRINT@384,R;"OUT OF";T;"CORRECT!": GOTO580
560 PRINT@384,"NO! THE CORRECT ANSWER IS";N(0)
580 PRINT@416,"ENTER R FOR RESTART, OR JUST"
590 PRINT@448,"<ENTER> FOR SAME";: INPUT B$
600 IF NOT (B$="R" OR B$="") THEN PRINT@465,"": GOTO590
610 IF B$="R" THEN GOTO170
620 GOTO310
1000 DATA "DER APFEL","THE APPLE","DIE TUR","THE DOOR"
1010 DATA "DER MENSCHEN","THE PERSON","DER KUCHEN","THE CAKE"
1020 DATA "DER SCHLUSSEL","THE KEY","DER MANN","THE MAN","DER HUT","THE HAT"
1030 DATA "DAS BILD","THE PICTURE","DAS MADCHEN","THE GIRL"
1040 DATA "DAS AUGE","THE EYE","DAS OHR","THE EAR","DIE FRUCHT","THE FRUIT"
1050 DATA "DAS LICHT","THE LIGHT","DAS WASSER","THE WATER"
1060 DATA "DIE SCHULERIN","THE STUDENT","DER MALER","THE ARTIST"
1070 DATA "DER SCHUH","THE SHOE","DER ABEND","THE EVENING"
1100 DATA -1,-1
```

"Greensleeves" GREEN

Here's an old folk song played on a modern instrument, your MC-10 or Color Computer. Like BACH and CAMP, this song can get a little repetitious, as it plays continuously. It may well be a good song to demonstrate to those visiting relatives. Crank up the volume and strap that distant aunt or cousin in front of the computer. Guaranteed results – they'll leave by the next plane, especially when they see the colors that change on the display in time with the music...

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

See the writeup for BACH. This program uses the same principals to generate tones by the SOUND command in BASIC.

How This Program Works

The DATA statements at the beginning of the program hold the F and D values for the SOUND command. A READ statement reads one F and one D value and these are used in the following SOUND command. Two -1 values mark the end of the piece. When these are READ, the "DATA pointer" is reset to the beginning by a RESTORE and the piece is restarted.

After each note is played, a CLS RND(8) statement clears the screen to a new random color.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. F values are a compromise between values suitable for the MC-10 and values suitable for the Color Computer.

GREEN "Greensleeves"

```
100 DATA 125,8,147,16,159,8,170,8,170,4,176,4,170,8,159,16,140,8
110 DATA 108,12,125,4,140,8
120 DATA 147,16,125,8,125,8,125,4,117,4,125,8,140,16,117,8,78,16,125,8
130 DATA 147,16,159,8,170,8,170,4,176,4,170,8,159,16,140,8,108,8,108,4
140 DATA 125,4,140,8
150 DATA 147,8,140,8,125,8,117,8,117,4,99,4,117,8,125,16,125,8
160 DATA 125,16,125,8
170 DATA 185,16,185,8,185,8,185,4,180,4,170,8,159,16,140,8,108,8
180 DATA 108,4,125,4,140,8
190 DATA 147,16,125,8,125,8,125,4,117,4,125,8,140,16,117,8
200 DATA 78,16,78,8
210 DATA 185,16,185,8,185,8,185,4,180,4,170,8,159,16,140,8
220 DATA 108,8,108,4,125,4,140,8
230 DATA 147,8,140,8,125,8,117,8,117,4,99,4,117,8,125,16,125,8
240 DATA 125,16,-1,-1
250 CLS
260 READ A,B: IF A=<0 THEN 300
270 SOUND A,B
280 CLS RND(8)
290 GOTO 260
300 RESTORE: GOTO 250
```

Interest Formulas INTEREST

Your MC-10 or Color Computer can easily calculate interest formulas. If you're concerned about home mortgage payments you might want to look at the MORTGAGE program first to find mortgage payment amounts, or at the AMORTIZE program to get a listing of payment schedules. If you need information about other loans, MORTGAGE and AMORTIZE will also provide it – they're good for any loans (or investments) in which the loans are paid back monthly.

The interest formulas in the Interest program are more general. They let you figure out:

- The amount of money you'll have in a savings acccount at 9.7 percent after 18 years if the interest is compounded daily
- The amount of money you'll have in the same savings account if the money is compounded monthly, or quarterly
- How much money you'll need to invest initially at 10% per annum compounded daily to have $10,000 after 10 years
- How much money you'll have to deposit every month at a given interest rate to have $10,000 after 10 years
- How much the monthly payments would be to pay off a loan of $3500 at 15 percent interest for 5 years
- How much you'll have at the end of 10 years if you deposit $100 per month every month for that time in an account that draws 11 per cent interest
- What you should pay someone for a "note" that has regular payments of $100 per month for 5 years

How to Use This Program

Enter the program into the MC-10 or Extended BASIC Color Computer. Start the program by entering

```
    RUN
```

You should see the title and first prompt message as follows:

```
        INTEREST FORMULAS

SELECT ONE OF THE FOLLOWING:
1. GIVEN AN INITIAL AMOUNT, HOW
   MUCH WILL IT BE AFTER X YEARS
2. GIVEN A FINAL AMOUNT, WHAT
   IS THE INITIAL INVESTMENT AMOUNT
3. GIVEN A FINAL AMOUNT, WHAT
   ARE THE REGULAR DEPOSITS
4. GIVEN A LOAN, WHAT
   ARE THE REGULAR PAYMENTS
```

```
5. GIVEN REGULAR PAYMENTS, WHAT
   IS THE FINAL AMOUNT OF MONEY
6. GIVEN REGULAR PAYMENTS, FIND
   PRESENT WORTH OF FINAL AMT
WHICH ONE?
```

You now select one of the six functions by pressing 1 through 6. We'll discuss each in turn:

1. Given an Initial Amount, How Much Will It Be After X Years

Here's a typical entry for this function:

```
      FIND FINAL AMOUNT
  (GIVEN INITIAL INVESTMENT)

ENTER:
 PRINCIPAL? 1000
 INTEREST (% /YEAR)? 12
 # COMPOUNDING PERIODS/YR? 12
 # OF YEARS? 10

 FINAL AMOUNT= 3300.38

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

In this example we computed how much an initial investment of $1000 would be worth after 10 years at 12% with compounding monthly.

2. Given a Final Amount, What is the Initial Investment Amount

Here's a typical entry for this function:

```
       FIND PRESENT WORTH
      (GIVEN FINAL AMOUNT)

ENTER:
 FINAL AMOUNT? 1200
 INTEREST (% /YEAR)? 10
 # COMPOUNDING PERIODS/YR? 4
 # OF YEARS? 5

 PRESENT WORTH= 732.32

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

In this example we computed how much an initial investment was required for $1200 after 5 years at 10% with compounding every quarter.

3. Given a Final Amount, What are the Regular Deposits

Here's a typical entry for this function:

```
     FIND PAYMENT OR RECEIPT
       (GIVEN FINAL AMOUNT)

ENTER:
 FINAL AMOUNT? 100000
 INTEREST (% /YEAR)? 12
 # COMPOUNDING PERIODS/YR? 365
 # OF YEARS? 25

 PAYMENT OR RECEIPT= 1.72

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

In this example we computed how much of a daily deposit would be required to accumulate $100,000 after 25 years at 12%. The answer is $1.72. (Maybe you could cut down on lunch...)

4. Given a Loan Amount, What are the Regular Payments

Here's a typical entry for this function:

```
     FIND PAYMENT OR RECEIPT
       (GIVEN PRINCIPAL)

ENTER:
 PRINCIPAL? 100000
 INTEREST (% /YEAR)? 12
 # OF EQUAL PAYMENTS/YR? 12
 # OF YEARS? 30

 PAYMENT OR RECEIPT= 1028.61

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

In this example we computed how much the monthly payments would be for 30 years to repay a $100,000 loan at 12% interest. The answer is $1028.61.

5. Given Regular Payments, What is the Final Amount of Money

Here's a typical entry for this function:

```
       FIND FINAL AMOUNT
        (GIVEN PAYMENT)
```

```
ENTER:
 PAYMENT AMOUNT? 100
 INTEREST (% /YEAR)? 12
 # OF EQUAL PAYMENTS/YR? 12
 # OF YEARS? 25

 FINAL AMOUNT= 187884.62

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

In this example we computed how much the final amount would be if we made regular monthly payments of $100.00 for 25 years at 12% per year. The answer is $187,884.62.

6. Given Regular Payments, What is the Present Worth of the Final Amount

Here's a typical entry for this function:

```
        FIND PRESENT WORTH
    (GIVEN PAYMENT OR RECEIPT)

ENTER:
 PAYMENT AMOUNT? 100
 INTEREST (% /YEAR)? 12
 # OF EQUAL PAYMENTS/YR? 12
 # OF YEARS? 25

 PRESENT WORTH=  9494.65

ENTER R FOR RESTART, OR JUST
 <ENTER> FOR SAME?
```

In this example we computed how much the "present worth" is of regular monthly payments of $100.00 for 25 years at 12% per year. The answer is $9494.65. The present worth answers the question "What would have to be my equivalent "lump sum" investment to get the same amount of money as regular payments made over the term of the loan?." In this case, investing a lump sum of $9494.65 at 12% compounded monthly is the same as $100 payments made monthly.

Some Background on the Program

All of these interest formulas are different ways of saying the same thing. They are all derived from money earning interest at a certain annual rate. If interest is "compounded," the annual rate is divided into smaller periods, such as months or days. In this case the interest rate is simply divided by the number of periods per year – if interest is compounded monthly, the interest is computed at the annual rate divided by 12, if interest is compounded daily, the interest is computed at the annual rate divided by 365.

All of the interest formulas use the same quantities. P is the principal, an initial sum of money. I is the interest rate, an annual rate. N is the number of compounding periods over the life of the loan. R is the regular payment (sometimes called the "uniform series" payment). You can see the formulas used in the program in the next section and we won't repeat them here.

How the Program Works

The computations for each of the six functions are done in program lines 320, 390, 460, 530, 600, and 670. The remainder of the program is involved with displaying "menus" on the screen and testing "limits" of the numbers that are input.

Special Notes

1. This program will run on all MC-10 and Extended BASIC Color Computer systems.
2. Commas can't be used in any of the amounts that are input.
3. The interest rate input must be in percentage points, and not a fraction. In other words, 18 percent should be entered as "18" and not ".18".
4. The program will generally calculate amounts up to $1,000,000. The rational for this is that any user requiring larger amounts will probably be using a Model 16, and not an MC-10 or Color Computer!
5. The maximum payment amount is $9,999.99. The maximum interest rate is 80%. The maximum number of compounding periods or equal payments is 369. The maximum number of years is 80.

```
100 CLEAR 500: CLS: PRINT@7,"INTEREST FORMULAS"
110 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
120 PRINT@96,"1. GIVEN AN INITIAL AMOUNT, HOW"
130 PRINT@131,"MUCH WILL IT BE AFTER X YEARS"
140 PRINT@160,"2. GIVEN A FINAL AMOUNT, WHAT"
150 PRINT@195,"IS THE INITIAL INVESTMENT AMT"
160 PRINT@224,"3. GIVEN A FINAL AMOUNT, WHAT"
170 PRINT@259,"ARE THE REGULAR DEPOSITS"
180 PRINT@288,"4. GIVEN A LOAN AMOUNT, WHAT"
190 PRINT@323,"ARE THE REGULAR PAYMENTS"
200 PRINT@352,"5. GIVEN REGULAR PAYMENTS, WHAT"
210 PRINT@387,"IS THE FINAL AMOUNT OF MONEY"
220 PRINT@416,"6. GIVEN REGULAR PAYMENTS, FIND"
230 PRINT@451,"PRESENT WORTH OF FINAL AMT"
240 PRINT@480,"WHICH ONE";: A$=INKEY$
250 IF NOT (A$>"0" AND A$<"7") THEN 240
260 A=VAL(A$)
270 ON A GOTO280,350,420,490,560,630
280 NM$="FINAL AMOUNT"
290 CLS: PRINT@7,"FIND "+NM$
300 PRINT@34,"(GIVEN INITIAL INVESTMENT)"
310 GOSUB700: GOSUB850
320 P=P*((1+(I/N))^(N*M))
330 GOSUB970: IF R$="R" THEN 100
340 GOTO290
350 NM$="PRESENT WORTH"
```

```
360 CLS: PRINT@7,"FIND "+NM$
370 PRINT@38,"(GIVEN FINAL AMOUNT)"
380 GOSUB750: GOSUB850
390 P=S*(1/((1+(I/N))^(N*M)))
400 GOSUB970: IF R$="R" THEN 100
410 GOTO360
420 NM$="PAYMENT OR RECEIPT"
430 CLS: PRINT@4,"FIND "+NM$
440 PRINT@38,"(GIVEN FINAL AMOUNT)"
450 GOSUB750: GOSUB850
460 P=S*((I/N)/(((1+(I/N))^(N*M))-1))
470 GOSUB970: IF R$="R" THEN 100
480 GOTO430
490 NM$="PAYMENT OR RECEIPT"
500 CLS: PRINT@4,"FIND "+NM$
510 PRINT@39,"(GIVEN PRINCIPAL)"
520 GOSUB700: GOSUB850
530 P=P*(((I/N)*((1+(I/N))^(N*M)))/((((1+(I/N))^(N*M))-1)))
540 GOSUB970: IF R$="R" THEN 100
550 GOTO500
560 NM$="FINAL AMOUNT"
570 CLS: PRINT@7,"FIND "+NM$
580 PRINT@40,"(GIVEN PAYMENT)"
590 GOSUB800: GOSUB850
600 P=R*((((1+(I/N))^(N*M))-1)/(I/N))
610 GOSUB970: IF R$="R" THEN 100
620 GOTO570
630 NM$="PRESENT WORTH"
640 CLS: PRINT@7,"FIND "+NM$
650 PRINT@35,"(GIVEN PAYMENT OR RECEIPT)"
660 GOSUB800: GOSUB850
670 P=R*((((1+(I/N))^(N*M))-1)/((I/N)*((1+(I/N))^(N*M))))
680 GOSUB970: IF R$="R" THEN 100
690 GOTO640
700 PRINT@96,"ENTER:"
710 PRINT@129,"PRINCIPAL";: INPUT P$
720 IF (VAL(P$)>0) AND (VAL(P$)<1000000) THEN P=VAL(P$): GOTO740
730 PRINT@140,"": GOTO710
740 RETURN
750 PRINT@96,"ENTER:"
760 PRINT@129,"FINAL AMOUNT";: INPUT S$
770 IF (VAL(S$)>0) AND (VAL(S$)<1000000) THEN S=VAL(S$): GOTO790
780 PRINT@140,"": GOTO760
790 RETURN
800 PRINT@96,"ENTER:"
810 PRINT@129,"PAYMENT AMOUNT";: INPUT R$
820 IF (VAL(R$)>0) AND (VAL(R$)<10000) THEN R=VAL(R$): GOTO840
830 PRINT@145,"": GOTO810
840 RETURN
850 PRINT@161,"INTEREST (% /YR)";: INPUT I$
860 IF (VAL(I$)>0) AND (VAL(I$)<81) THEN I=VAL(I$)/100: GOTO880
870 PRINT@179,"": GOTO850
880 IF A$<"3" THEN N2$="# COMPOUNDING PERIODS/YR": GOTO900
890 N2$="# EQUAL PAYMENTS/YR"
900 PRINT@193,N2$;: INPUT N$
910 IF (VAL(N$)>0) AND (VAL(N$)<370) THEN N=VAL(N$): GOTO930
920 PRINT@214,"": GOTO900
930 PRINT@225,"# OF YEARS";: INPUT M$
940 IF (VAL(M$)>0) AND (VAL(M$)<81) THEN M=VAL(M$): GOTO960
950 PRINT@237,"": GOTO930
960 RETURN
```

```
970 IF P>999999 THEN PRINT@289,NM$;"=";P: GOTO1000
980 P=INT(P*100): P$=STR$(P): L=LEN(P$)
990 PRINT@289,NM$+"="+LEFT$(P$,L-2)+"."+RIGHT$(P$,2)
1000 PRINT@416,"ENTER R FOR RESTART, OR JUST"
1010 PRINT@448,"<ENTER> FOR SAME";: INPUT R$
1020 IF NOT (R$="R" OR R$="") THEN PRINT@465,"": GOTO1010
1030 RETURN
```

Mailing Label Program LABELS

LABELS is a mailing label program that prints mailing labels on your system line printer from names and addresses contained within the program. The program allows you to print from 1 to 1000 labels each, vary the spacing between labels, change the left margin for the labels, make a test run before starting the printing, and lets you output a special code sequence for printers that have features such as selection of special typefaces, boldface, and so forth.

How to Use This Program

This program operates with mailing label data contained as DATA statements within the program. To see a sample printout, continue reading. To enter your own data, see **Entering Your Own Data** below. Put the mailing labels into your printer and align them with the left margin or within several inches of the left margin. Turn on the printer power and put the printer "on-line."

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
    RUN
```

You should see the title message and a "prompt" message appear as follows:

```
   MAILING LABEL PROGRAM

SPCL CODE SEQUENCE, Y OR N?
```

If you don't want to use special features on your printer such as boldface or special type fonts, just press N followed by <ENTER> to continue. If you want to initialize your printer to a special feature, however, press Y followed by enter. (In most cases, you'll probably want "normal" printing and will be entering N.)

Special Code Sequence

Suppose that you have a printer that allows you to bold face type in a special type face. According to the printer manual, boldface is selected by sending the code sequence 27, 31 to the printer and a "correspondence" type face is selected by sending the code sequence 27, 18. These two "escape sequences" are typical codes to set up the printer to perform special functions. In this case, enter Y for the SPCL CODE SEQUENCE question, and you'll see:

```
READY PRINTER
```

displayed on the bottom of the screen. This will disappear, and you'll then see:

```
ENTER CODE SEQUENCE,
END BY -1?
```

You should now "ready" the printer by turning on power and putting it "on line." You can now enter the code sequence 27, 31, 27, 18 by entering each value followed by an <ENTER>. After entering the four values, enter a -1 followed by <ENTER>.

If the printer was "ready," you'll see the message:

```
# OF LABELS EACH?
```

Enter the number of labels of each entry you want printed, from 1 through 1000, followed by <ENTER>. (Normally you'd probably want only 1 label of each entry and you'd be entering a "1".)

You'll now see:

```
# OF LINES PER LABEL?
```

Enter a value from 1 through 100 for the number of lines **between** labels. End the input by an <ENTER>. Most printers print at 6 lines per vertical inch. If all of your mailing list entries have 3 lines and you're using mailing labels that are spaced one inch apart (typical) then you'll want a value of 6 for this value.

You'll now see:

```
LEFT MARGIN, 0-60?
```

Enter a value for the left margin from 0 through 60, followed by <ENTER>. If the mailing labels are aligned with the printer's left margin, type 5, otherwise use a value corresponding to 10 counts per horizontal inch plus a few character positions for indentation from the left side of the label. You'll be able to make test runs, so don't be too concerned about this now.

You'll now see

```
TEST RUN, Y OR N?
```

If you want a test run, enter Y followed by <ENTER>, otherwise reply with N followed by <ENTER>. If you selected a test run, you'll see the message

```
READY PRINTER
```

on the bottom of the screen, and you'll then see a printout of one label. After the test run, you'll see the "Test Run" message displayed again. If the margin is off, restart the program by entering BREAK followed by RUN. If the labels need to be adjusted vertically, use the line printer paper advance to reposition the labels. If the margin and vertical spacing are correct, type N for the "Test Run" message.

If you've entered N for the test run prompt, the program will print out all the labels contained within the DATA statements. If you want to stop the printing at any time, press BREAK, followed by RUN to restart the program. The program will restart after the last label is printed.

Entering Your Own Data
The program as it stands has several sets of sample names and addresses. To enter your own, follow these instructions:

1. Delete all of the sample data statements except the one that has DATA "*****".
2. Enter your own names and addresses by using DATA statements. Surround each separate string by double quotes and put a comma between each string of characters. In the sample data "WM. BARDEN, JR." is one line and "250 N.S. MEMORY LANE" is the next line.
3. You can use as many DATA statements per entry as you want, as long as you use the double quotes and commas in between strings.
4. After each name and address, use a terminating "*" entry in the DATA statement.
5. The DATA statements should be located at the end of the program and must end with the "*" of the last entry, followed by the DATA "*****" line.

Some Background on the Program

The special code sequences described above are normally called "escape" sequences, as the first character of each is a decimal 27, or "escape" code. They are not normal ASCII characters and the printer will decode the sequence as a "special code sequence coming."

Most printers operate in standard mode at 10 characters per inch horizontally and 6 lines per inch vertically. Your printer may have the capability of elongation (increasing the horizontal spacing), changing the number of characters per inch horizontally, proportional spacing, or changing the vertical spacing by escape sequences.

There are no special techniques in printing data except that the program must be aware of the end of each entry and the start of the next. This is done in LABELS by marking the end of each entry with a "*" character.

How This Program Works

LABELS first reads in and checks all of the parameters for the print. It also reads in any special code sequence and outputs the code sequence to the line printer.

The program then READs each DATA value as a character string. If the character string is not equal to "*", it prints the line. If the character string is equal to "*", it subtracts the number of lines printed for the current entry from the number of lines per label. The result is the number of blank lines to be printed between the current label and the next.

When the program READs a "*****" string, the program restarts.

Two print subroutines are used. The first prints character string A$ but does not print a carriage return. The second prints a carriage return only.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. If your printer is not "ready," the program will appear to "hang up" either for the special code sequence or for the first label. The printer "ON LINE" indicator must be on, indicating power on, paper, etc.
3. Change the "PRINT#-2," statements to "LPRINT" statements where indicated for MC-10 systems.

```
100 DIM A$(20)
110 CLS: PRINT @5,"MAILING LABEL PROGRAM"
120 NL=1: LB=6: LM=0
130 PRINT @32,"SPCL CODE SEQUENCE, Y OR N";: INPUT S$
140 IF (S$<>"Y") AND (S$<>"N") THEN PRINT @ 54,"": GOTO 130
150 IF S$="N" THEN 250
160 GOSUB 440
170 PRINT @64,"ENTER CODE SEQUENCE,"+CHR$(13)+"END BY -1"
180 PRINT @105,"":PRINT:PRINT
190 PRINT@105,"";:INPUT S: IF S=-1 THEN 230
200 IF (S<0) OR (S>255) THEN PRINT @448,"INVALID CODE": GOSUB 450:GOTO 180
210 A$=CHR$(S):GOSUB 490
220 GOTO 180
230 PRINT @64,"":PRINT:PRINT
240 GOSUB 520
250 PRINT @64,"# OF LABELS EACH";:INPUT NL
260 IF (NL<1) OR (NL>1000) THEN PRINT @80,"": GOTO 250
270 PRINT @96,"# OF LINES PER LABEL";:INPUT LB
280 IF (LB<1) OR (LB>100) THEN PRINT @ 116,"": GOTO 270
290 PRINT @128,"LEFT MARGIN, 0-60";: INPUT LM
300 IF (LM<0) OR (LM>60) THEN PRINT @145,"": GOTO 290
310 PRINT @160,"TEST RUN, Y OR N";: INPUT T$
320 IF (T$<>"Y") AND (T$<>"N") THEN PRINT @176,"": GOTO 310
330 RESTORE: GOSUB 440
340 I=1
350 READ A$(I): IF A$(I)="******" THEN 110
360 IF A$(I)="*" THEN 380
370 I=I+1: GOTO 350
380 I=I-1: FOR J=1 TO NL
390 GOSUB 540
400 IF T$<>"Y" THEN 420
410 PRINT @176,"": GOTO 310
420 NEXT J
430 GOTO 340
440 PRINT@448,"READY PRINTER"
450 FOR I=1 TO 1000: NEXT I
460 PRINT@448,"             "
470 RETURN
480 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
490 PRINT#-2,A$;
500 RETURN
510 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
520 PRINT#-2
530 RETURN
540 FOR K=1 TO I
550 FOR L=1 TO LM: A$=" ": GOSUB 490: NEXT L
560 A$=A$(K): GOSUB 490: GOSUB 520
570 NEXT K
```

```
580 M=LB-I: IF M<0 THEN PRINT @448,"TOO MANY LINES IN LABEL":GOSUB 450: GOTO 600
590 FOR K=1 TO M: GOSUB 520: NEXT K
600 RETURN
610 DATA "WM. BARDEN, JR.", "250 N.S. MEMORY LANE"
620 DATA "MICRO CITY, CA 99999","*"
630 DATA "FORREST MIMS III","987 TEXAN FOREVER DRIVE"
640 DATA "LITTLE SPRING, TX 77777","*"
650 DATA "DENNIS KITZ","ARKHAM CORNERS","MISKATONIC CITY, MA 03333","*"
660 DATA "Wm. Barden","*"
670 DATA "******"
```

Metric Conversion One METRIC1

The metric system is used worldwide. If you're traveling in Germany, for example, you'll see road signs in kilometers rather than miles. Even European photocopier paper is geared to centimeters, rather than inches! This program will enable you to easily convert between the English system of measurement, where distances are in miles, feet, and inches and the metric system where distances are in kilometers, meters, and centimeters.

Just to give you some ground rules: A kilometer is about 0.62 miles, a meter is about 39.37 inches, and a centimeter is about 0.39 inches. METRIC1 will make it easy to manipulate these relationships.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
    RUN
```

You should see the title and first prompt message as follows:

```
     METRIC CONVERSION

SELECT ONE OF THE FOLLOWING:
 1. KILOMETERS TO MILES
 2. MILES TO KILOMETERS
 3. METERS/CM TO FT/INCHES
 4. FT/INCHES TO METERS/CM
WHICH ONE?
```

You can now select one of the four options by entering 1, 2, 3, or 4, followed by <ENTER>.

Kilometers to Miles

If you selected this option, you'll see

```
KILOMETERS?
```

displayed on the screen. Enter the distance in kilometers, followed by <ENTER>. The answer will then be displayed. Suppose you had entered 100 for "KILOMETERS." You'd see

```
KILOMETERS? 100
MILES= 62.09

PRESS R TO RESTART
```

Pressing the R key would bring you back to the main "menu."

Miles to Kilometers

Miles to kilometers works the same way as the first function. In this case, though, you'd enter the distance in miles, and the conversion would be to kilometers. Here's a sample:

```
MILES? 100
KILOMETERS= 160.89

PRESS R TO RESTART
```

Again, pressing R will bring you back to the main selection menu.

Meters/Cm to Feet/Inches

Selecting this function lets you convert from metric meters and centimeters to feet and inches. You'll be asked for both the "meters" value and the "cm" value. It's not ncessary to convert to the nearest meter – just enter any number of meters and centimeters. If you want to enter either meters or centimeters, just leave the other value blank by pressing <ENTER> without any value. End each entry with an <ENTER>. Here are some samples:

```
METERS? 100        CM? 34
FT= 329.00         INCHES= 2.38

PRESS R TO RESTART


METERS?            CM? 1000
FT= 32.00          INCHES= 9.70

PRESS R TO RESTART


METERS? 9999       CM?
FT= 32805.00       INCHES= .63

PRESS R TO RESTART
```

Feet/Inches to Meters/Cm

This selection works the same way as the previous one except in reverse. Enter any number of feet and inches, and the answer will be displayed in meters and centimeters:

```
FT? 1000           INCHES? 45
METERS= 305.00     CM= 94.36

PRESS R TO RESTART


FT? 123            INCHES?
METERS= 37.00      CM= 49.04

PRESS R TO RESTART


FT?                INCHES? 9999
METERS= 253.00     CM= 97.51

PRESS R TO RESTART
```

Some Background on the Program

There's really nothing very complicated about this program as far as computations go. Meters, centimeters, and feet are changed to an internal representation of inches. Multiplying by 39.37 changes meters to inches, multiplying by .3937 changes centimeters to inches, and multiplying by 12 changes feet to inches. Converting from inches to meters and centimeters is about as easy, except that inches are divided by 39.37 for centimeters and any remainder is divided by .3937 to get centimeters. Conversions between kilometers and miles is done by multiplying by 0.621 (kilometers to miles) or 1.609 (miles to kilometers).

How This Program Works

Most of the work in this program is in displaying the menu, inputting values, and displaying the conversion values on the screen. An "input" subroutine inputs the values and checks them against maximum limits. An "output" subroutine displays the result, to two decimal places.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. The maximum value for any input is 9999.
3. Negative values are not allowed.

```
100 CLS: PRINT@7,"METRIC CONVERSION"
110 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
120 PRINT@97,"1. KILOMETERS TO MILES"
130 PRINT@129,"2. MILES TO KILOMETERS"
140 PRINT@161,"3. METERS/CM TO FT/INCHES"
150 PRINT@193,"4. FT/INCHES TO METERS/CM"
160 PRINT@224,"WHICH ONE";: INPUT A$
170 IF (VAL(A$)>0) AND (VAL(A$)<5) THEN A=VAL(A$): GOTO210
180 PRINT@256,"INVALID SELECTION--TRY AGAIN"
190 FOR I=1 TO 300: NEXT I
200 PRINT@234,"": PRINT: GOTO160
210 ON A GOSUB250,300,350,410
220 PRINT@480,"PRESS R TO RESTART";: A$=INKEY$
230 IF A$="R" THEN 100
240 GOTO220
250 A1$="KILOMETERS": A2$="MILES"
260 GOSUB490
270 A2=A1*.621
280 GOSUB580
290 RETURN
300 A1$="MILES": A2$="KILOMETERS"
310 GOSUB490
320 A2=A1*1.609
330 GOSUB580
340 RETURN
350 A1$="METERS": B1$="CM": A2$="FT": B2$="INCHES"
360 GOSUB490: GOSUB530
370 A1=(A1*39.37)+(B1*.3937): REM-METERS/CM TO INCHES
```

```
380 A2=INT(A1/12): B2=A1-(A2*12)
390 GOSUB580: GOSUB610
400 RETURN
410 A1$="FT": B1$="INCHES": A2$="METERS": B2$="CM"
420 GOSUB490: GOSUB530
430 A1=(A1*12)+B1: REM-FT/INCHES TO INCHES
440 A2=INT(A1/39.37)
450 B2=(A1-(A2*39.37))/.3937
460 GOSUB580: GOSUB610
470 RETURN
480 REM -- INPUT SUBROUTINES
490 PRINT@384,A1$;: INPUT B$
500 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN A1=VAL(B$): GOTO520
510 PRINT@384,"": GOTO490
520 RETURN
530 PRINT@400,B1$;: INPUT B$
540 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN B1=VAL(B$): GOTO560
550 PRINT@400,"": GOTO530
560 RETURN
570 REM -- OUTPUT SUBROUTINES
580 A2=INT(A2*100): B$=STR$(A2): L=LEN(B$)
590 PRINT@416,A2$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
600 RETURN
610 B2=INT(B2*100): B$=STR$(B2): L=LEN(B$)
620 PRINT@432,B2$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
630 RETURN
```

Metric Conversion Two METRIC2

This program is a follow on to METRIC1. METRIC1 converts between the metric units of meters and centimeters and the English units of feet and inches. METRIC1 also converts between kilometers and miles. In METRIC2, liquid measure, weight, and temperature are converted. The metric unit for liquid measure is liters (a little over a quart), and these units are converted to gallons, quarts, pints, and ounces and vice versa. The metric units for weight are kilograms and grams. A kilogram is approximately equal to 2.2 pounds. METRIC2 converts between kilograms and grams and pounds and ounces. Metric temperatures are measured in degrees Celsius, and the last conversion in METRIC2 converts between degrees Celsius and degrees Farenheit.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
    RUN
```

You should see the title and the "menu" as follows:

```
    METRIC CONVERSION

SELECT ONE OF THE FOLLOWING:
 1. GALS/QTS/PTS/OZS TO LITERS
 2. LITERS TO GALS/QTS/PTS/OZS
 3. KGS/GRAMS TO LBS/OZS
 4. LBS/OZS TO KGS/GRAMS
 5. FARENHEIT TO CELSIUS
 6. CELSIUS TO FARENHEIT
WHICH ONE?
```

You can now select one of the six options by entering 1, 2, 3, or 4, 5, or 6 followed by <ENTER>.

Gals/Qts/Pts/Ozs to Liters

If you selected this option, you'll see

```
GALS?
```

displayed on the screen. Enter the amount in gallons, followed by <ENTER>. You'll then be given prompts for QTS?, PTS?, and OZS?. Enter a value for each, or simply press <ENTER>. The answer in liters will then be displayed. Suppose you had entered 2 for GALS?, 3 for QTS?, 4 for PTS?, and 10 for OZS?. You'd see

```
GALS? 2          QTS? 3
PTS? 4           OZS? 10
LITERS= 12.59
PRESS R TO RESTART
```

Pressing the R key would bring you back to the main "menu."

Liters to Gals/Qts/Pts/Ozs

If you selected this option, you'll see

```
LITERS?
```

displayed on the screen. Enter the amount in liters, followed by <ENTER>. The answer in gallons, quarts, pints, and ounces will then be displayed. Suppose you had entered 10 for LITERS?. You'd see

```
LITERS? 10
GALS= 2.00          QTS= 2.00
PTS= 1.00           OZS= 2.13
PRESS R TO RESTART
```

Pressing the R key would bring you back to the main "menu."

Kilograms/Grams to Pounds and Ounces

If you selected this option, you'll see

```
KGS?
```

displayed on the screen. Enter the amount in kilograms, followed by <ENTER>. You'll then be given a prompt for GRAMS? Enter a value for grams, or simply press <ENTER>. The answer in pounds and ounces will then be displayed. Suppose you had entered 100 for KGS? and 34 for GRAMS?. You'd see

```
KGS? 100            GRAMS? 34
LBS= 220.00         OZS= 9.18

PRESS R TO RESTART
```

Pressing the R key would bring you back to the main "menu."

Pounds and Ounces to Kilograms/Grams

If you selected this option, you'll see

```
LBS?
```

displayed on the screen. Enter the amount in pounds, followed by <ENTER>. You'll then be given a prompt for OZS? Enter a value for ounces or simply press <ENTER>. The answer in kilograms and grams will then be displayed. Suppose you had entered 100 for LBS? and 12 for OZS?. You'd see

```
LBS? 100            OZS? 34
KGS= 45.00          GRAMS= 697.14

PRESS R TO RESTART
```

Pressing the R key would bring you back to the main "menu."

Farenheit to Celsius

If you selected this option, you'd see:

```
FARENHEIT?
```

Enter the temperature in degrees Farenheit, followed by <ENTER>. Use a minus sign for readings below zero degrees Farenheit. A plus sign isn't necessary. The temperature in degrees Celsius will then be displayed:

```
FARENHEIT? 212
CELSIUS= 100.00

PRESS R TO RESTART
```

Again, pressing R will bring you back to the main selection menu.

Celsius to Farenheit
If you selected this option, you'd see:

```
CELSIUS?
```

Enter the temperature in degrees Celsius, followed by <ENTER>. Use a minus sign for readings below zero degrees Celsius. The temperature in degrees Farenheit will then be displayed:

```
CELSIUS? -40
FARENHEIT= -40.00

PRESS R TO RESTART
```

Again, pressing R will bring you back to the main selection menu.

Some Background on the Program

As in METRIC1, there's really nothing very complicated about this program as far as computations go. Gallons, quarts, pints, and liters are changed to an internal representation of ounces. Multiplying by 128 changes gallons to ounces, multiplying by 32 changes quarts to ounces, and multiplying by 16 changes pints to ounces. Dividing by 33.814 converts the ounces to liters. Converting from liters to gallons, quarts, pints, and ounces works similarly, except that liters are multiplied by 33.814 and divisors of 128, 32, and 16 are used to find the other quantities.

Converting from kilograms is about as easy – kilograms are changed to ounces by multiplying by 35.28 and grams to ounces by multiplying by 0.035. The result is divided by 16 to give pounds and ounces. Converting from pounds and ounces to kilograms and grams works similarly except that 35.28 is divided into the total number of ounces to yield kilograms and grams.

A value of 100 degrees Celsius by definition is the boiling point of water, 212 degrees Farenheit. A value of 0 degrees Celsius is the freezing point of water, 32 degrees Farenheit. The temperature difference between the boiling point of water and the freezing point of water is 180 degrees Farenheit or 100 degrees Celsius, so one degree Farenheit is 180/100 or 9/5 degree Celsius and one degree Celsius is 100/180 or 5/9 degree Farenheit.

METRIC2 Metric Conversion 2

To convert from Farenheit to Celsius, the formula

C=(5/9)*(F-32)

is used.

To convert from Celsius to Farenheit, the formula

F=(9/5)*C+32

is used.

How This Program Works

Most of the work in this program is in displaying the menu, inputting values, and displaying the conversion values on the screen. A series of "input" subroutines read in the values and check them against maximum limits. A set of "output" subroutines display the results, to two decimal places.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. The maximum value for any input is 9999. Negative values are allowed only for temperature readings.
3. It's not necessary to round off to the nearest unit. You could enter 1 pound and 75 ounces, for example, rather than 5 pounds and 11 ounces.

```
100 CLS: PRINT@7,"METRIC CONVERSION"
110 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
120 PRINT@97,"1. GALS/QTS/PTS/OZS TO LITERS"
130 PRINT@129,"2. LITERS TO GALS/QTS/PTS/OZS"
140 PRINT@161,"3. KGS/GRAMS TO LBS/OZS"
150 PRINT@193,"4. LBS/OZS TO KGS/GRAMS"
160 PRINT@225,"5. FAHRENHEIT TO CELSIUS"
170 PRINT@257,"6. CELSIUS TO FAHRENHEIT"
180 PRINT@288,"WHICH ONE";: INPUT A$
190 IF (VAL(A$)>0) AND (VAL(A$)<7) THEN A=VAL(A$): GOTO230
200 PRINT@320,"INVALID SELECTION--TRY AGAIN"
210 FOR I=1 TO 300: NEXT I
220 PRINT@298,"": PRINT: GOTO180
230 ON A GOSUB270,310,380,430,490,520
240 PRINT@480,"PRESS R TO RESTART";: A$=INKEY$
250 IF A$="R" THEN 100
260 GOTO240
270 A1$="GALS": B1$="QTS": A2$="PTS": B2$="OZS": A3$="LITERS"
280 GOSUB560: GOSUB630: GOSUB670: GOSUB710
290 A3=((A1*128)+(B1*32)+(A2*16)+B2)/33.814
300 GOSUB820: RETURN
310 A1$="LITERS": A2$="GALS": B2$="QTS": A3$="PTS": B3$="OZS"
320 GOSUB560: A1=A1*33.814
330 A2=INT(A1/128): B2=INT((A1-(128*A2))/32)
340 A3=INT((A1-(128*A2)-(32*B2))/16)
350 B3=A1-(128*A2)-(32*B2)-(16*A3)
360 GOSUB760: GOSUB790: GOSUB820: GOSUB850
370 RETURN
380 A1$="KGS": B1$="GRAMS": A2$="LBS": B2$="OZS"
```

```
390 GOSUB560: GOSUB630
400 A1=(A1*35.28)+(B1*.035): REM-KGS/GRAMS TO OZS
410 A2=INT(A1/16): B2=A1-(16*A2)
420 GOSUB760: GOSUB790: RETURN
430 A1$="LBS": B1$="OZS": A2$="KGS": B2$="GRAMS"
440 GOSUB560: GOSUB630
450 A1=(A1*16)+B1: REM-LB/OZ TO OZ
460 A2=INT(A1/35.28)
470 B2=(A1-(A2*35.28))/.035
480 GOSUB760: GOSUB790: RETURN
490 A1$="FAHRENHEIT": A2$="CELSIUS"
500 GOSUB560: A2=(5/9)*(A1-32)
510 GOSUB760: RETURN
520 A1$="CELSIUS": A2$="FAHRENHEIT"
530 GOSUB560: A2=(9/5)*A1+32
540 GOSUB760: RETURN
550 REM -- INPUT SUBROUTINES
560 PRINT@384,A1$;: INPUT B$
570 IF A<5 THEN 600
580 IF VAL(B$)<10000 THEN A1=VAL(B$): GOTO 620
590 GOTO 610
600 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN A1=VAL(B$): GOTO620
610 PRINT@384,"": GOTO560
620 RETURN
630 PRINT@400,B1$;: INPUT B$
640 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN B1=VAL(B$): GOTO660
650 PRINT@400,"": GOTO630
660 RETURN
670 PRINT@416,A2$;: INPUT B$
680 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN A2=VAL(B$): GOTO700
690 PRINT@416,"": GOTO670
700 RETURN
710 PRINT@432,B2$;: INPUT B$
720 IF (VAL(B$)=>0) AND (VAL(B$)<10000) THEN B2=VAL(B$): GOTO740
730 PRINT@432,"": GOTO710
740 RETURN
750 REM -- OUTPUT SUBROUTINES
760 A2=INT(A2*100): B$=STR$(A2): L=LEN(B$)
770 PRINT@416,A2$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
780 RETURN
790 B2=INT(B2*100): B$=STR$(B2): L=LEN(B$)
800 PRINT@432,B2$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
810 RETURN
820 A3=INT(A3*100): B$=STR$(A3): L=LEN(B$)
830 PRINT@448,A3$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
840 RETURN
850 B3=INT(B3*100): B$=STR$(B3): L=LEN(B$)
860 PRINT@464,B3$;"=";LEFT$(B$,L-2);".";RIGHT$(B$,2)
870 RETURN
```

Mortgage Calculator MORTGAGE

The Mortgage Calculator program performs two calculations relating to home mortgages or other loans, such as a "chattel" mortgage on an automobile. The first calculation the program performs is this: given the monthly payment, interest rate, and years, find the amount financed. The second calculation it performs is to find the monthly payment, given an amount to be financed, the interest, and the number of years in the loan. You might look upon the two calculations this way: The first is for people who say: "Let's see, I can afford monthly payments of $200 a month for a new car. I wonder how expensive a car I can afford if I finance it for 3 years?." The second calculation is for impulse buyers, who, having signed the paperwork say "Gee, I wonder what the monthly payments will be?"

The two calculations are based on a "fully amortized" loan that has monthly payments. In other words, the loan will be repaid monthly and after the last payment, the loan will be completely repaid.

How to Use This Program

Enter the program into the MC-10 or Extended BASIC Color Computer. Start the program by entering

```
    RUN
```

You should see the title and first prompt message as follows:

```
    MORTGAGE CALCULATION

SELECT ONE OF THE FOLLOWING:
1. GIVEN MONTHLY PAYMENT,
   INTEREST RATE, AND YEARS,
   FIND PRINCIPAL

2. GIVEN PRINCIPAL, INTEREST,
   AND YEARS, FIND MONTHLY
   PAYMENT

WHICH ONE?
```

Select either function by pressing 1 or 2 and <ENTER>.

Finding the Monthly Payment

If you selected the first function, you'll see this message and prompt question:

```
          FIND PRINCIPAL

ENTER MONTHLY PAYMENT?
```

Enter the monthly payment and answer the two following questions. The program will then print out the monthly payment. If, for example, you wanted monthly payments of $200 for a 3-year loan on an automobile at 12.5% annual rate, you'd have:

```
       FIND PRINCIPAL
ENTER MONTHLY PAYMENT? 200
 INTEREST RATE (% PER YEAR)? 12.5
 NUMBER OF YEARS? 3

 PRINCIPAL= 5978.42

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

Don't use commas in any of the inputs, and use "percentage points" rather than a number less than 1. You can now figure out another monthly payment by pressing <ENTER>, or restart the program by entering R, followed by <ENTER>.

Finding the Monthly Payment

The second function works pretty much like the first. Here's a typical set of inputs for finding the monthly payments to finance a $1000 water bed at an interest rate of 13% for 10 years:

```
      FIND MONTHLY PAYMENT

ENTER PRINCIPAL? 1000
 INTEREST RATE (% PER YEAR)? 13
 NUMBER OF YEARS? 10

 MONTHLY PAYMENT= 14.93

ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

Some Background on the Program

Interest rates are usually quoted in annual rates. The simplest type of loan is for a sum of money at a specified annual interest rate with the sum of money plus interest due after a specified time. You could borrow $1000 at 12% for one year, for example, and at the end of a year you'd have to pay back the original sum of money plus interest of $1000 x .12 = $120, making the total amount due $1120.

Now suppose that you borrow $1000 at 12% per year with the money due in 12 equal installments. What should those installments be? **Not $1120/12 or $93.33!** If you paid back 1/12th of the final amount due each month, you'd be paying more interest than 12%! After all, in the first case, you had the full use of the money for a year. In the second case you had the use of $1000 for one month, the use of $906.67 for the second month, the use of $873.33 the third month, and so forth.

The interest formula used to find a "uniform series" payment can be found in books on investment and is:

```
      R=P*((i*(1+i)↑n)/((1+i)↑n-1)))
```

where P is the principal, or amount borrowed, i is the interest rate per month, n is the number of months, and R is the monthly payment. (We've used computer nomenclature here - "*" for multiply, "/" for divide, "↑" for "exponentiation," and parentheses to group expressions.

If you use the example above - $1000 at 12 percent paid back in 12 monthly installments, you'll find that the monthly payment is $88.84, less than the $98.83 found by simply dividing the total amount financed plus interest by 12, but reflecting the true interest rate of 12 percent.

The formula used to find the original principal amount is:

```
P=R*(((1+i)↑n-1)/(i*(1+i)↑n))
```

where i, n, P, and R are the same quantities as before.

How the Program Works

The computation for the monthly payment is done in line 410. It follows the same formula as described above. The annual interest rate is changed to a monthly interest rate by dividing by 12. The computation for principal is in line 300. It also follows the corresponding formula above. The remainder of the program is devoted to displaying menus and messages.

Special Notes

1. This program will run on all MC-10 and Extended BASIC Color Computer systems.
2. Commas can't be used in any of the amounts that are input.
3. The interest rate input must be in percentage points, and not a fraction. In other words, 18 percent should be entered as "18" and not ".18."
4. The maximum principal is $9,999,999.99. The maximum monthly payment is $9,999.99. The maximum interest rate is 90%. The maximum number of years is 99.

```
100 CLS: PRINT@6,"MORTGAGE CALCULATOR"
110 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
120 PRINT@96,"1. GIVEN MONTHLY PAYMENT,"
130 PRINT@131,"INTEREST RATE, AND YEARS,"
140 PRINT@163,"FIND PRINCIPAL"
150 PRINT@224,"2. GIVEN PRINCIPAL, INTEREST,"
160 PRINT@259,"AND YEARS, FIND MONTHLY"
170 PRINT@291,"PAYMENT"
180 PRINT@352,"WHICH ONE";: INPUT A$
190 IF A$="1" THEN 230
200 IF A$="2" THEN 350
210 PRINT@448,"INVALID SELECTION--TRY AGAIN"
220 FOR T=1 TO 300: NEXT T: GOTO100
230 REM - COMPUTE PRINCIPAL
240 CLS: PRINT@9,"FIND PRINCIPAL"
250 PRINT@64,"ENTER MONTHLY PAYMENT";: INPUT R$
260 IF (VAL(R$)>0) AND (VAL(R$)<10000) THEN R=VAL(R$): GOTO280
270 PRINT@87,"": GOTO250
```

```
280 GOSUB470
290 P=R*(((1+(I/12))^(Y*12)-1)/((I/12)*((1+(I/12))^(Y*12))))
300 P=INT(P*100): P$=STR$(P): L=LEN(P$)
310 PRINT@193,"PRINCIPAL="+LEFT$(P$,L-2)+"."+RIGHT$(P$,2)
320 GOSUB550
330 IF A$="R" THEN 100
340 GOTO230
350 REM - COMPUTE MONTHLY PAYMENT
360 CLS: PRINT@6,"FIND MONTHLY PAYMENT"
370 PRINT@64,"ENTER PRINCIPAL";: INPUT P$
380 IF (VAL(P$)>0) AND (VAL(P$)<10000000) THEN P=VAL(P$): GOTO400
390 PRINT@80,"": GOTO370
400 GOSUB470
410 R=P*(((I/12)*((1+(I/12))^(Y*12)))/((1+(I/12))^(Y*12)-1))
420 R=INT(R*100): R$=STR$(R): L=LEN(R$)
430 PRINT@193,"MONTHLY PAYMENT="+LEFT$(R$,L-2)+"."+RIGHT$(R$,2)
440 GOSUB550
450 IF A$="R" THEN 100
460 GOTO350
470 PRINT@97,"INTEREST RATE (% /YR)";: INPUT I$
480 IF (VAL(I$)>0) AND (VAL(I$)<91) THEN I=VAL(I$)/100: GOTO510
490 PRINT@117,"": GOTO470
510 PRINT@129,"NUMBER OF YEARS";: INPUT Y$
520 IF (VAL(Y$)>0) AND (VAL(Y$)<100) THEN Y=VAL(Y$): GOTO540
530 PRINT@146,"": GOTO510
540 RETURN
550 PRINT@416,"ENTER R FOR RESTART, OR JUST"
560 PRINT@448,"<ENTER> FOR SAME";: INPUT A$
570 IF NOT (A$="R" OR A$="") THEN PRINT@465,"": GOTO550
580 RETURN
```

Multiplication and Division Tutorial Program MULTDIV

Multiplication and division have you baffled? The MULTDIV program is a tutorial program for learning Multiplication and Division. It starts off with multiplication and division of two one-digit numbers such as 9 x 3 and progresses through multiplication and division of a five-digit by four-digit number such as 74469 x 4996.

The program keeps a count of the number of correct answers and displays the count after each answer. The program will give occasional encouraging messages.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
      RUN
```

You should see the title and first prompt message as follows:

```
 MULTIPLICATION/DIVISION PROGRAM

SELECT ONE OF THE FOLLOWING:
A. MULTIPLICATION
B. DIVISION
WHICH ONE?
```

You can now enter A or B followed by <ENTER>.

If you've selected multiplication, you'll see the first addition problem displayed on the screen, something like:

```
          MULTIPLICATION

   9
 * 7
 ------
?
```

The "*" stands for multiplication. It's the BASIC symbol that's used in place of "x." You should now enter the answer, followed by <ENTER>. After you've answered, you'll see the total correct, followed by a message telling you what to do next. In this case, you'd have:

```
   9
 * 7
 ------
? 63

CORRECT! 1 OUT OF 1

ENTER H FOR HARDER, E FOR
EASIER, R FOR RESTART, OR
JUST <ENTER> FOR SAME?
```

If you just press <ENTER> at this point, you'll get another problem of the same type, two one-digit numbers in this case. If you enter H, followed by <ENTER>, you'll get a problem with one two-digit number and one one-digit number, such as 49 x 1. You can keep entering H after each answer to get up to one five-digit number multiplied by one four-digit number. You can also go back the other way, to easier problems by entering E for easier.

Entering R followed by <ENTER> brings you back to the selection between multiplication and division.

If you select division, you'll see a division problem, starting with two one-digit numbers. Here again, you can get to harder problems by entering H after each answer, to easier problems by entering E, or back to the selection between addition and subtraction by entering R.

The format for division is:

```
 5/2                Q=?
```

The program is asking for the Quotient of the division problem of 5 divided by 2. Enter the quotient, followed by <ENTER>. After you've entered the quotient, the program will ask for the remainder, as in:

```
 5/2                Q=2

                    R=?
```

Enter the remainder, followed by <ENTER>, and you'll see the same messages as described above, a correct answer message and the question on what you'd like to do next.

Some Background on the Program

The main problem in this program is not in doing the arithmetic – BASIC in the MC-10 and Color Computer can multiply and divide numbers easily, even mixed numbers such as 1.234 x 5.667 or large numbers such as 1,234,567 x 8,884.568. The problem is in coming up with "random numbers." The program uses the RND function in BASIC to produce the two numbers that are used in the problems.

How This Program Works

Most of this program is involved with generating the proper sized numbers and with "formatting" the numbers used in the problems.

Array LM is a two-dimensional array that holds information for generating the numbers in the problems. There are 5 levels of difficulty, corresponding to the number of digits in the problems. The array is initialized with ten DATA values. The first set of 2 DATA values, 0 and 9, define the lowest possible number and the highest possible number that can be used for one-digit values in the problem.

The next set, 10 and 99, define the lowest and highest number for two-digit values, and so forth.

The multiplicand in a multiply problem (the number on top) or the dividend in a division problem (the number to be divided) is generated by using RND(LM(C,2)), where C is the difficulty level from 1 through 5. The multiplier in a multiply problem (the number on the bottom) or the divisor in a division problem (the number that "goes into") is generated by using RND(LM(D,2)), where D is one less than the difficulty level from 1 through 4. This ensures that the multiplier and divisor are always smaller than the multiplicand and dividend.

For division, a check is made to make certain that the divisor is smaller than the dividend.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. An encouraging message such as "GOOD WORK!" is displayed after every 10th answer if the total correct answers are over 60 per cent of the total problems.
3. Don't use commas in any of the answers. The program is not expecting commas and will ignore digits after the first comma.

```
100 DIM LM(5,2), MS$(5)
110 DATA 0,9,10,99,100,999,1000,9999,10000,99999
120 DATA "GOOD WORK!","KEEP IT UP!","NICE GOING!","YOU'RE DOING GREAT"
130 DATA "FANTASTIC!"
140 FOR T=1 TO 5
150 READ LM(T,1), LM(T,2): NEXT T
160 FOR T=1 TO 5
170 READ MS$(T): NEXT T
180 CLS
190 PRINT@1,"MULTIPLICATION/DIVISION PROGRAM"
200 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
210 PRINT@96,"A. MULTIPLICATION"
220 PRINT@128,"B. DIVISION"
230 PRINT@160,"WHICH ONE";: INPUT A$
240 IF A$="A" THEN 290
250 IF A$="B" THEN 430
260 PRINT@224,"INVALID SELECTION--TRY AGAIN"
270 FOR T=1 TO 300: NEXT T
280 PRINT@224,"";: PRINT: PRINT@171,"": GOTO230
290 REM MULTIPLICATION
300 C=1: T1=0: T2=0
310 IF C<3 THEN D=1: GOTO330
320 D=C-1
330 A1=RND(LM(C,2)): A2=RND(LM(D,2))
340 IF (A1<LM(C,1)) OR (A2<LM(D,1)) THEN 330
350 A3=A1*A2
360 B$=STR$(A1): C$=STR$(A3): D$=STR$(A2)
370 CLS
380 PRINT@9,"MULTIPLICATION": PRINT@166,"*"
390 PRINT@135,A1: PRINT@167+(LEN(B$)-LEN(D$)),A2
400 PRINT@198,"-------"
```

```
410 PRINT@230-(LEN(C$)-LEN(B$)),"";: INPUT T
420 GOSUB570: GOTO310
430 REM DIVISION
440 C=1: T1=0: T2=0
450 IF C<3 THEN D=1: GOTO470
460 D=C-1
470 A1=RND(LM(C,2)): A2=RND(LM(D,2))
480 IF (A1<LM(C,1)) OR (A2<LM(D,1)) OR (A1<A2) THEN 470
490 A3=INT(A1/A2)
500 R1=A1-(A2*A3)
510 CLS
520 PRINT@12,"DIVISION"
530 PRINT@161,A1;"/";A2
540 PRINT@175,"Q=";: INPUT T
550 PRINT@207,"R=";: INPUT R
560 GOSUB570: GOTO450
570 REM PRINT SUBROUTINE
580 T2=T2+1
590 IF A$="B" THEN 620
600 IF T=A3 THEN 640
610 PRINT@320,"WRONG! THE ANSWER IS";A3: GOTO670
620 IF (T=A3) AND (R=R1) THEN 640
630 PRINT@320,"WRONG! ANSWER IS Q";A3;"R";R1: GOTO670
640 T1=T1+1
650 IF ((T2/10)=INT(T2/10)) AND ((T1/T2)>.60) THEN PRINT@238,MS$(C)
660 PRINT@320,"CORRECT!";T1;"OUT OF";T2
670 PRINT@384,"ENTER H FOR HARDER, E FOR"
680 PRINT@416,"EASIER, R FOR RESTART, OR"
690 PRINT@448,"JUST <ENTER> FOR SAME";: INPUT B$
700 IF B$="R" THEN 180
710 IF B$="" THEN 750
720 IF (B$="E") AND (C>1) THEN C=C-1: GOTO750
730 IF (B$="H") AND (C<5) THEN C=C+1: GOTO750
740 GOTO670
750 RETURN
```

Keyboard Music Program MUSIC

MUSIC makes your MC-10 or Color Computer into a Steinway! Well... maybe not a Steinway...In any event, this program turns the upper row of keys – 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, :, and – into the 12 notes of G (below middle C), A, B, C, D, E, F, G, A, B, C, and D. Sorry, no sharps or flats. We could have included them but we tried to give you the maximum number of useful notes instead, and you can still hack away on this computer Steinway in the key of C. Every time you press one of the top rows of keys, you'll get a tone corresponding to the key you've pressed. You'll also see a musical staff on the screen and the note that you've sounded will light up. The note duration can also be set to either quarter notes, half notes, or whole notes.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and the musical staff as follows:

```
      MUSIC PROGRAM

-------------------------

-----------------------0-
                     0
-------------------0-----
                 0
---------------0---------
             0
-----------0-------------
         0
      -0-
     0
  -0-
 0
```

Pressing any of the keys in the top row will sound a note in the sequence GABCDEFGABCD (keyboard keys 1 through -). Don't use the SHIFT key when you press a key or the note will not sound.

Pressing the Q key at any time will change the note duration to quarter notes. Pressing the H key will make the duration half notes. Pressing the W key will sound whole notes. When the program is first started, the "default" duration is half notes.

Some Background on the Program

The MC-10 produces musical tones by the SOUND command. The format of SOUND is

SOUND F,D

where F is any value from 1 through 255, and D is any value from 1 through 255. The F value establishes the "pitch" of the sound, while the D value establishes the duration of the sound.

A D value of 1 is about .075 seconds. A D value of 255 is about 19 seconds on the MC-10, and a little less on the Color Computer. D values in between 1 and 255 produce sounds of between .075 and 19 seconds. Every time D is increased by 1, another .075 seconds is added to the length of the note produced.

The F value changes the pitch, or "frequency" of the sound. A value of 1 for F is a pitch of about 151 cycles per second (151 hertz); this is about equal to D sharp below middle C on the piano keyboard. A value of 255 for F is a pitch of about 14,700 cycles per second (14,700 hertz); this is far beyond the highest-pitched key on a piano. Values of F in between 1 and 255 produce notes between 151 and 14,700 cycles per second.

The sound produced by the MC-10 is a "square wave," as shown in Figure MUSIC-1. This type of waveform is rich in "odd harmonics" and is used in electronic music synthesizers.

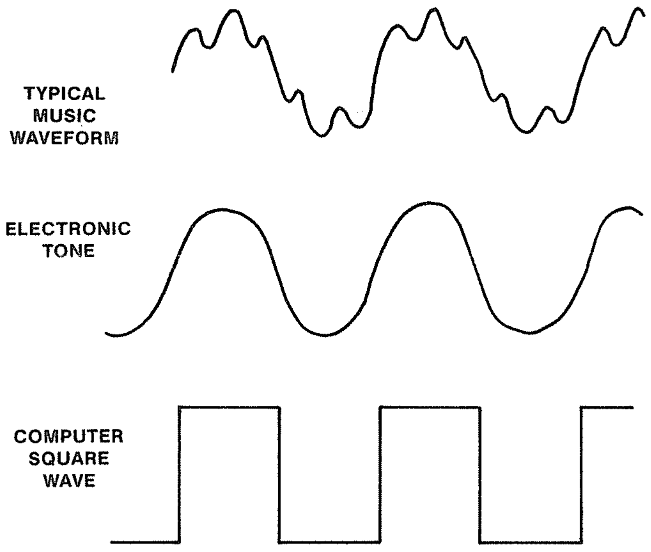


Figure MUSIC-1. MC-10 and Color Computer Waveform

The main problem in MUSIC is to find the proper values for the F value to correspond to the notes on the piano keyboard. Most of the values come quite close, and although the notes are not an exact match, they are adequate for playing simple tunes.

The Color Computer uses the SOUND command in the same way as the MC-10. Extended Color BASIC on the Color Computer also has the PLAY command, a powerful command that allows you to easily play sequences of notes over different octaves, durations, and loudness.

How This Program Works

The DATA statements at the beginning of the program hold 12 different sets of values. Each set contains an F value corresponding to the F value for the musical notes of GABCDEFGABCD and a second value. The second value is the "PRINT@" value for printing the note on the musical staff displayed on the screen.

The DATA values are stored into array T when the program starts, to make it easy to access the F and screen position values.

The program reads the top row of keys by doing an INKEY$ operation. If the key is not a "null," the ASCII value of the key is used to access the T array for the F and screen location values. The program then uses the F value in a SOUND command, along with the current value for the note duration from variable J.

At the same time that the note is played, the "O" on the screen is changed to a red graphics character (CHR$(191)) for a short time to "flash" the note. The screen position used is the one picked up from the T array.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. When playing notes from "1" to "-" on the Color Computer, don't continue and press BREAK as the next key! If you do, just enter RUN again to restart the program.

```
110 DIM T(12,2)
120 DATA 38,452,58,422,78,392,89,362,108,332,125,302,133,272,147,242
125 DATA 159,212,170,182,176,152,185,122
130 FOR I=1 TO 12
140 READ T(I,1),T(I,2): NEXT I
150 CLS: PRINT@9,"MUSIC PROGRAM"
160 FOR I=34 TO 290 STEP 64
170 FOR J=1 TO 25
180 PRINT@I+J,"-";
190 NEXT J,I
200 PRINT@361,"- -";
210 PRINT@421,"- -";
220 J=92
230 FOR I=1 TO 12
240 J=J+30
250 PRINT@J,"O";
260 NEXT I
265 J=4
300 A$=INKEY$
310 IF A$="" THEN 300
```

```
320 A=ASC(A$)
360 IF (A>48) AND (A<58) THEN I=A-48: GOTO410
370 IF A=45 THEN I=12: GOTO410
380 IF A=48 THEN I=10: GOTO410
390 IF A=58 THEN I=11: GOTO410
392 IF A=70 THEN J=8: REM-FULL NOTE
394 IF A=72 THEN J=4: REM-HALF NOTE
396 IF A=81 THEN J=2: REM-QTR NOTE
400 GOTO300
410 SOUND T(I,1),J
430 N=T(I,2): PRINT@N,CHR$(191);
432 FOR Z=1 TO 25: NEXT Z
435 PRINT@N,"O";
440 GOTO300
```

"Ode to Joy" ODE

Here's another classical piece. Roll over Beethoven! This is the famous "Ode to Joy" from Beethoven's Ninth Symphony. It sounds pretty good on the MC-10 or Color Computer, especially when you see the relaxing random colored points being produced on the screen. Use it to strengthen your alpha waves...

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

See the writeup for BACH. This program uses the same principals to generate tones by the SOUND command in BASIC.

How This Program Works

The DATA statements at the beginning of the program hold the F and D values for the SOUND command. A READ statement reads one F and one D value and these are used in the following SOUND command. Two -1 values mark the end of the piece. When these are READ, the "DATA pointer" is reset to the beginning by a RESTORE and the piece is restarted.

After each tone is played by the SOUND command, a random point with a random color is set on the screen. As the screen is initially cleared to black, the result is a pleasing display of random colors.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. F values are a compromise between values suitable for the MC-10 and values suitable for the Color Computer.

ODE "Ode to Joy"

```
100 DATA 170,8,170,8,176,8,185,8,185,8,176,8,170,8,159,8,147,8
110 DATA 147,8,159,8,170,8,170,12,159,4,159,16
120 DATA 170,8,170,8,176,8,185,8,185,8,176,8,170,8,159,8,147,8
130 DATA 147,8,159,8,170,8,159,12,147,4,147,16
140 DATA 159,8,159,8,170,8,147,8,159,8,170,4,176,4,170,8,147,8
150 DATA 159,8,170,4,176,4,170,8,159,8,147,8,159,8,108,16
160 DATA 170,8,170,8,176,8,185,8,185,8,176,8,170,8,159,8,147,8
170 DATA 147,8,159,8,170,8,159,12,147,4,147,16
180 DATA  -1,-1
190 CLS 0
200 READ A,B
210 IF A<0 AND B<0 THEN 250
220 SOUND A,B
230 SET(RND(63),RND(31),RND(8))
240 GOTO 200
250 RESTORE: GOTO 200
```

Ohm's Law Calculator OHMSLAW

The Ohms Law Calculator program is a representation of a basic electrical circuit that has a resistance, voltage source, and current. The program finds either resistance, voltage, or current, given the other two values. The basic circuit is shown in Figure OHMSLAW-1.

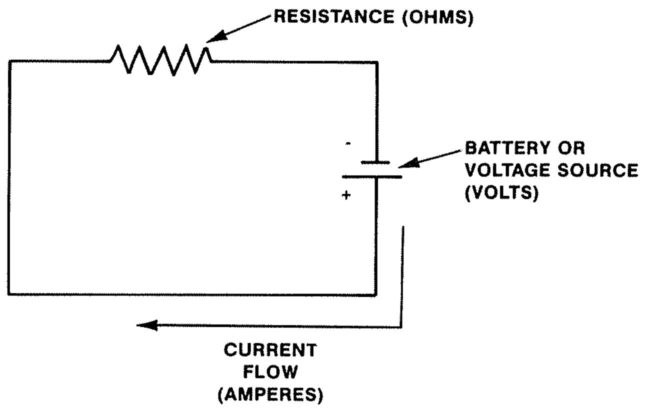


Figure OHMSLAW-1. Basic Electrical Circuit

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
        RUN
```

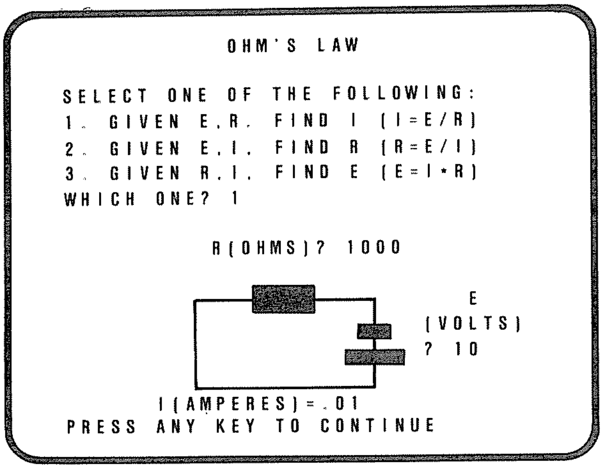
You should see the title and first prompt message as follows:

```
          OHMS LAW

SELECT ONE OF THE FOLLOWING:
1. GIVEN E,R, FIND I (I=E/R)
2. GIVEN E,I, FIND R (R=E/I)
3. GIVEN R,I, FIND E (E=I/R)
WHICH ONE?
```

You should then enter one of the menu items, from 1 to 3. After you select a menu item, you'll see a circuit diagram as shown in Figure OHMSLAW-2. The rectangle represents a resistance, the two horizontal lines are the schematic diagram for a battery, and the boundary between the colors indicates the circuit path, or wiring.

Based on your menu selection, the program asks for the two "given" values and then computes the third value, as shown in the figure.



```
                OHM'S LAW

SELECT ONE OF THE FOLLOWING:
1. GIVEN E,R. FIND I (I=E/R)
2. GIVEN E,I. FIND R (R=E/I)
3. GIVEN R,I. FIND E (E=I*R)
WHICH ONE? 1

             R(OHMS)? 1000

                                 E
                              (VOLTS)
                              ? 10

          I(AMPERES)=.01
PRESS ANY KEY TO CONTINUE
```

Figure OHMSLAW-2. OHMSLAW Display

Pressing any key will restart the program for a new calculation.

Some Background on the Program

Current flow in a simple circuit occurs by the flow of electrons, subatomic particles that make up atoms. It requires a great many electrons flowing in a circuit to make up even the smallest current.

Current flow results when there is a deficiency of electrons at one point of a circuit and a surplus of electrons at another point that is electrically connected. A battery produces such a condition by chemical action – there is a surplus of electrons on one set of plates of the battery, and a deficiency of electrons on the other set of plates. This "potential difference" can be produced by other means, such as moving a coil of wire through a magnetic field, as in a generator, or by exposing a "photovoltaic" cell to light, as in a solar cell.

We've shown all sources of electrical energy as a battery symbol, even though the energy source could be a generator, solar cell, or another type.

The potential difference between the plates of a battery or between the poles of a generator or solar cell is measured in "volts," named after Alessandro Volta, an early electrical experimenter. The larger the voltage value, the greater is the difference in the surplus and deficiency of electrons.

When a wire or "conductor" is connected between the plates of a battery or poles of a generator or solar cell, a current of electrons flows from the point of surplus electrons to the point of deficiency of electrons.

The amount of current is measured in "amperes," named for Andre Ampere, an 18th century electrical experimenter. One ampere is a rate of 16,000,000,000,000,000,000 electrons flowing per second!

The number of electrons that flow is dependent not only upon the potential difference, but the "resistance" of the wire or conductor. Some materials are excellent conductors, such as gold or copper; others are extremely poor conductors, or "insulators," such as glass or plastic. Other materials, such as germanium and silicon, are part way between conductors and insulators, so they are called "semiconductors."

A "resistor" is an electrical part that is purposely made to have a higher resistance than a good conductor. This is done by making the resistance thin, to reduce the size of the path like a small diameter garden hose, or by choosing a material for the resistance that is not a good conductor.

Resistance is measured in "ohms," named for Georg Ohm, another early electrical experimenter. A 100 foot length of #14 gauge copper wire used in home wiring has a resistance of about .25 ohms, but a typical resistor used in the MC-10 or Color Computer might have a resistance of 1000 ohms.

All of which brings us to "Ohms Law"... Georg Ohm found that a voltage source of 1 volt produced a current flow of 1 ampere when connected by a resistance of 1 ohm. Knowing any of two quantities, resistance (ohms), voltage (volts), or current (amperes), the third can easily be calculated. The formulas he discovered were:

$$I=E/R \qquad R=E/I \qquad E=I*R$$

where I is current in amperes, E is voltage (potential difference) in volts, and R is resistance in ohms.

How This Program Works

The calculations in this program are easy, as BASIC in the MC-10 or Color Computer can rapidly caluculate the simple division required to find one quantity from the other two. Variables E, I, and R represent voltage, current, and resistance in the program.

Most of the program is devoted to displaying the messages on the screen and drawing the "schematic diagram" of the electrical circuit. The diagram is drawn by doing a series of PRINTs of strings. The strings in this case are not made up of text characters, but are made up of graphics blocks to represent the resistance and battery symbols. The boundary line between the two colored screen areas represents the electrical path connecting the resistance and battery.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

2. Don't use commas in any of the inputs. The program does not expect commas and will ignore digits after the first comma.

3. The maximum value for resistance is 99,999,999 ohms and for voltage is 99,999 volts.

```
100 CLEAR 500: CLS: PRINT@11,"OHM'S LAW"
110 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
120 PRINT@96,"1. GIVEN E,R, FIND I  (I=E/R)"
130 PRINT@128,"2. GIVEN E,I, FIND R  (R=E/I)"
140 PRINT@160,"3. GIVEN R,I, FIND E  (E=I*R)"
150 PRINT@192,"WHICH ONE";: INPUT A$
160 IF (VAL(A$)>0) AND (VAL(A$)<4) THEN 190
170 PRINT@448,"INVALID SELECTION--TRY AGAIN"
180 FOR I=1 TO 300: NEXT I: GOTO100
190 REM - PRINT FIGURE
200 B$=CHR$(140): C$=CHR$(207)
210 D$=CHR$(204): E$=CHR$(195)
220 L1$=B$+B$+B$+B$
230 L2$=C$+C$+C$+C$+E$+E$+E$+E$+C$+C$+C$+C$
240 L3$=C$+C$+C$+C$+C$+C$+C$+C$+C$+C$+C$+D$+B$
250 L4$=C$+C$+C$+C$+C$+C$+C$+C$+C$+C$+D$+D$+B$+B$
260 L5$=C$+C$+C$+C$+C$+C$+C$+C$+C$+C$+C$+C$
270 PRINT@301,L1$;: PRINT@329,L2$;
280 PRINT@361,L3$;: PRINT@393,L4$;
290 PRINT@425,L5$;
300 IF A$="2" THEN 340
310 PRINT@266,"R(OHMS)";: INPUT R$
320 IF (VAL(R$)>0) AND (VAL(R$)<99999999) THEN R=VAL(R$): GOTO340
330 PRINT@266,"": GOTO310
340 IF A$="3" THEN 390
350 PRINT@347,"E";: PRINT@376,"(VOLTS)";
360 PRINT@408,"";: INPUT E$
370 IF (VAL(E$)>0) AND (VAL(E$)<99999) THEN E=VAL(E$): GOTO390
380 PRINT@409,"      ";: GOTO360
390 IF A$="1" THEN 430
400 PRINT@454,"I(AMPERES)";: INPUT I$
410 IF (VAL(I$)>0) AND (VAL(I$)<99999999) THEN I=VAL(I$): GOTO460
420 PRINT@454,"": GOTO400
430 I=E/R: I$=STR$(I)
440 PRINT@454,"I(AMPERES)=";MID$(I$,2,14)
450 GOTO530
460 IF A$="3" THEN 500
470 R=E/I: R$=STR$(R)
480 PRINT@266,"R(OHMS)=";MID$(R$,2,13)
490 GOTO530
500 E=I*R: E$=STR$(E)
510 PRINT@347,"E";: PRINT@376,"(VOLTS)"
520 PRINT@408,MID$(E$,2,6)
530 PRINT@480,"PRESS ANY KEY TO CONTINUE";: A$=INKEY$
540 IF A$="" THEN 530
550 GOTO100
```

Graph Plotter PLOTTER

The PLOTTER program lets you easily plot points on a graph. The points can be rapidly entered into the program and the program will take care of "scaling" the x and y axes of the graph for you. An example is shown in Figure PLOTTER-1, where the X axis represents values from 1 to 967, and the Y axis represents values from 1 to 433.

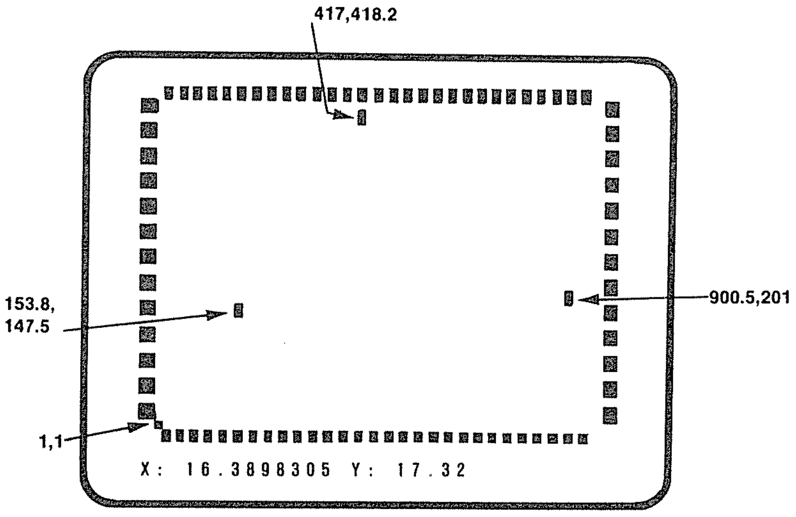


Figure PLOTTER-1. Plot Example

Using PLOTTER, you can input as many points as you wish, temporarily view the results, and then continue entering points.

PLOTTER is set up to plot "positive" X and Y values with the X axis of the graph at the bottom of the screen, the Y axis at the left side of the screen, and the "origin" at the lower left corner of the screen.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
    RUN
```

The screen will clear and you'll see the following title and prompt message:

```
    POINT GRAPH PROGRAM
INPUT MAXIMUM X VALUE?
```

You can now enter a maximum value for X from a fractional value less than 1 to 999,999. End the value by pressing the <ENTER> key.

You'll now see the message:

```
INPUT MAXIMUM Y VALUE?
```

Enter a maximum value for Y from a fractional value less than 1 to 999,999. End the value by pressing the <ENTER> key.

The next message is:

```
COLOR OF POINTS (1-8)?
```

Enter the color that you want for the points of the graph, from 1 through 8. The colors are

1=green	5=buff
2=yellow	6=cyan
3=blue	7=magenta
4=red	8=orange

(The axes of the graph will always be a green color, and the background of the graph will be black.)

You'll now see the message:

```
X,Y (END WITH -1,-1)?
```

You should now enter the point values that you want plotted on the graph, X, followed by Y. Each set of values must be less than or equal to the maximum X and Y values you specified previously. If either the X or Y values are greater than the maximum values or less than 0, you'll see the momentary message:

```
INVALID X,Y - TRY AGAIN
```

at the bottom of the screen, and the current values will not be accepted. Fractional values are allowed.

When you've entered the last value, enter -1,-1 followed by <ENTER>. The graph will now be drawn and the points plotted. As each point is plotted, you'll hear a "beep." This is useful for those cases where points overlap a previous point. The beep indicates that a point was processed. The resolution of the graph is 59 points wide by 25 points high. If you want to continue entering points from this point press "C" for continue, and you'll see the

```
X,Y (END WITH -1,-1)?
```

message again. Enter the next set of points and end with a -1,-1 as before.

If you want to restart the plotting after the graph is displayed, press any key (except C) after the graph has been displayed.

The values displayed for X and Y at the bottom of the screen represent the values for the increments along the axis. Each "tic mark" and the space in between is equal to the value of X or Y specified in the message.

Some Background on the Program

Graphing on the MC-10 and Color Computer is done by using the SET command. The SET command plots one point on the screen. In the graphics mode used on the MC-10 and in the "low resolution" mode of the Color Computer, there are 64 points horizontally by 32 points vertically. Any of the points may be set by performing a:

```
      SET X,Y,C
```

where X is 0 through 63, Y is 0 through 31, and C is a color code of 1 through 8.

Because some of the screen must be used for the axes of the graph, the number of X points in PLOTTER is 59 and the number of Y points is 25.

The main problem in PLOTTER is to convert from a coordinate system where increasing Y moves up on the graph; in the MC-10 and Color Computer, increasing Y values move **down.** Also, some adjustment must be made to X and Y for the axis. This entire process is called "coordinate translation."

Another problem is to "scale" the increments of X and Y to the given maximum values of X and Y. This is easily done in the Y case by dividing the value of Y by the maximum value of Y and multiplying by 25, and in the X case by dividing the value of X by the maximum value of X and multiplying by 59. The results are the X and Y increment values.

How This Program Works

Much of the logic in the program is involved with drawing the X and Y axes. The two horizontal axes are drawn by creating two strings, A$ and B$. Each character in A$ and B$ are made up of a single graphics character that represents one "tic mark." The tic mark is on the bottom of the graphics character for the A$ string and on the top of the graphics character for the B$ string.

The vertical axes are drawn by a short loop, which sets two out of four graphics elements per character position.

Plotting is done by converting the given X and Y values to computer coordinates and doing a SET with the color specified in the initialization sequence. Each time a point is plotted, a SOUND command sounds a short beep.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. To end input, both the X and Y values must be -1. Entering -1,20, for example, will result in an "INVALID X,Y" message and will not end the input.

PLOTTER Graph Plotter

```
100 DIM X(100),Y(100)
110 CLS: PRINT@5,"POINT GRAPH PROGRAM"
120 PRINT@32,"INPUT MAXIMUM X VALUE";:INPUT MX
130 IF (MX<0 OR MX>1000000) THEN PRINT@32,"":GOTO 120
140 PRINT@64,"INPUT MAXIMUM Y VALUE";:INPUT MY
150 IF (MY<0 OR MY>1000000) THEN PRINT@64,"":GOTO 140
160 SX=(MX/59): SY=(MY/25)
170 PRINT@96,"COLOR OF POINTS (1-8)";:INPUT C
180 IF (C<1 OR C>8) THEN PRINT@96,"":GOTO 170
190 J=0
200 PRINT@128,"X,Y (END WITH -1,-1";:INPUT X,Y
210 IF (X=-1 AND Y=-1) THEN 290
220 IF (X<0 OR X>MX) THEN 240
230 IF (Y>=0 AND Y<=MY) THEN 280
240 PRINT@480,"INVALID X,Y - TRY AGAIN";
250 FOR I=0 TO 300: NEXT I
260 PRINT@480,"                          ";
270 PRINT@128,"":PRINT:PRINT:PRINT:PRINT:GOTO 200
280 X(J)=X: Y(J)=Y: J=J+1: GOTO 270
290 CLS0: A$="":B$=""
300 FOR I=2 TO 30
310 A$=A$+CHR$(128+1)
320 B$=B$+CHR$(128+4)
330 NEXT I
340 FOR I=2 TO 26
350 IF INT(I/2)=I/2 THEN SET(0,I,0):SET(1,I,0):SET(62,I,0):SET(63,I,0)
360 NEXT I
370 PRINT@1,A$;:PRINT@449,B$;
380 PRINT@480,"X:";SX;"Y:";SY;
390 IF J=0 THEN 470
400 FOR I=0 TO J-1
410 X=X(I): Y=Y(I): XD=(59/MX)*X: YD=(25/MY)*Y
420 IF XD-INT(XD)>.5 THEN XD=XD+1
430 IF YD-INT(YD)>.5 THEN YD=YD+1
440 XD=INT(XD)+2:YD=27-INT(YD):SET(XD,YD,C)
450 SOUND 100,3
460 NEXT I
470 A$=INKEY$:IF A$="" THEN 470
480 IF A$="C" THEN CLS:GOTO 270
490 GOTO 110
```

"When the Saints Go Marching In"
SAINTS

This is a good old time jazz number from the heart of New Orleans. SAINTS not only plays the tune, but also displays the words on the screen so that you can sing along.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and the music will start. Don't forget to turn up the sound on your television receiver!

Some Background on the Program

See the writeup for BACH. This program uses the same principals to generate tones by the SOUND command in BASIC.

How This Program Works

The DATA statements at the beginning of the program hold the F and D values for the SOUND command. A READ statement reads one F and one D value and these are used in the following SOUND command. Two -6 values mark the end of the piece. When these are READ, the "DATA pointer" is reset to the beginning by a RESTORE and the piece is restarted.

Interspersed with the DATA values for the SOUND command are "flags" for the words. Whenever the program READs a value less than 0, it checks to see which lyrics should be displayed on the screen. The proper words are then displayed immediately before the notes that apply.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. F values are a compromise between values suitable for the MC-10 and values suitable for the Color Computer.

SAINTS "When the Saints Go Marching In"

```
100 DATA -1,-1,89,4,125,4,133,4,147,16,-2,-2,147,4,89,4,125,4,133,4,147,16
110 DATA -3,-3,147,4,89,4,125,4,133,4,147,8,125,8,89,8,125,8,108,16
120 DATA -4,-4,108,8,125,4,108,4,89,8,89,4,89,4,108,8,147,4,147,4,147,4
130 DATA -5,-5,133,8,133,4,133,8,125,4,133,4,147,8,125,8,89,8,108,8
140 DATA 89,16,-6,-6
150 CLS
160 READ A,B: IF A=<0 THEN 190
170 SOUND A,B
180 GOTO 160
190 IF A=-1 THEN PRINT@263,"OH WHEN THE SAINTS":GOTO 160
200 IF A=-2 THEN PRINT@263,"  GO MARCHING IN":GOTO 160
210 IF A=-3 THEN PRINT@263, "OH WHEN THE SAINTS"
220 IF A=-3 THEN PRINT@297,"GO MARCHING IN": GOTO 160
230 IF A=-4 THEN PRINT@259,"I WANT TO BE IN THAT NUMBER":PRINT:GOTO 160
240 IF A=-5 THEN PRINT@257,"WHEN THE SAINTS GO MARCHING IN":GOTO 160
250 RESTORE: GOTO 150
```

Screen Save Subroutine SAVESC

SAVESC is a short subroutine that can be incorporated into your own programs to save the contents of the video screen on cassette tape. The file created on the tape can then be reloaded at any time by using another part of SAVESC. In this manner you can save colorful displays, text data, or anything that appears on the screen during a BASIC program. Color Computer users note that this is for the "text" screen only; the program does not save graphics pages.

How to Use This Program

This program is a **subroutine** that must be merged with your own programs. To merge SAVESC, enter your own program with lines less than 10000. Now enter SAVESC for the Color Computer or SAVESCMC for the MC-10 as it appears in this section.

In addition to incorporating SAVESC within your own program, you'll need a:

```
DIM A(255)
```

statement somewhere within your own program.

To "call" SAVESC from within your own program, just do a:

```
GOSUB 10000
```

to write out the screen contents at any point. Do a:

```
GOSUB 11000
```

to read in the screen contents at any point.

For writes, the cassette tape must be rewound and positioned to the proper point for writing out a file. For reads the cassette tape must be positioned just before a previously written SAVESC file.

When the write portion of SAVESC is called, the file will be written out – no further action will be taken. Writing takes about 40 seconds. For the read portion of SAVESC, you'll see the screen fill up with the contents of the file. Reading also takes about 40 seconds.

Here's a short program that writes out graphics data and then reads it in again:

```
100 DIM A(255): CLS 0
110 FOR I=1 TO 300
120 SET (RND(63),RND(31),RND(8))
130 NEXT I
140 GOSUB 10000
150 A$=INKEY$: IF A$="" THEN 150
160 CLS 0
170 GOSUB 11000
180 GOTO 180
```

SAVESC Save Screen Subroutine

Some Background on the Program

The contents of the video display for the Color Computer and MC-10 are located in RAM memory. The "hardware" for the two systems reads the text data in memory and converts it into a television-compatible signal to update the video display.

The video display "buffer" for the Color Computer starts at location 1024 and continues through location 1535. The video display buffer for the MC-10 is located from RAM locations 16384 through 16895.

The video display buffer can be read by PEEK statements and written into by POKE statements. SAVESC uses PEEK to transfer data from the video display buffer to cassette tape and POKE to transfer data from cassette tape to the video display buffer.

How This Program Works

A "screenful" of data is made up of 512 bytes for the text screen. These 512 bytes are found by PEEKing from the video display buffer. As each byte is found, it is stored in array A. Array A is not present in SAVESC, but must be specified in a statement in the user program that uses SAVESC. To save space, two bytes from two video display buffer locations are "packed" into a single array element. Array elements A(0) through A(255) will hold all 512 bytes of a single video display.

The read function of SAVESC (11000) reverses the procedure, "unpacking" the 256 elements of array A to get 512 bytes, which are stored into the video display buffer thereby updating the screen.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. If you attempt to CLOAD this program, you will get an "OM" or "Out of Memory" error. It must be incorporated in your existing program and your program must include a DIM A(255).

SAVESC

```
10000 REM SCREEN TO CASSETTE SUBROUTINE
10010 FOR I=1024 TO 1534 STEP 2
10020 J=(I-1024)/2
10030 A(J)=PEEK(I)*256+PEEK(I+1)
10040 NEXT I
10050 OPEN"O",-1,"DISPLAY"
10060 FOR I=0 TO 255
10070 PRINT#-1,A(I)
10080 NEXT I
10090 CLOSE-1
10100 RETURN
11000 REM CASSETTE TO SCREEN SUNROUTINE
11010 OPEN"I",-1,"DISPLAY"
11020 FOR I=0 TO 255
11030 INPUT#-1,A(I)
11040 NEXT I
11050 CLOSE-1
11060 FOR I=1024 TO 1534 STEP 2
11070 J=(I-1024)/2
11120 B=INT(A(J)/(256))
11130 POKE I,B:A(J)=A(J)-B*256
11140 POKE I+1,A(J)
11150 NEXT I
11160 RETURN
```

SAVESCMC

```
10000 REM SCREEN TO CASSETTE SUBROUTINE
10010 FOR I=16384 TO 16894 STEP 2
10020 J=(I-16384)/2
10030 A(J)=PEEK(I)*256+PEEK(I+1)
10040 NEXT I
10050 CSAVE*A,"DISPLAY"
10060 RETURN
11000 REM CASSETTE TO SCREEN SUNROUTINE
11010 CLOAD*A,"DISPLAY"
11020 FOR I=16384 TO 16894 STEP 2
11030 J=(I-16384)/2
11040 B=INT(A(J)/(256))
11050 POKE I,B:A(J)=A(J)-B*256
11060 POKE I+1,A(J)
11070 NEXT I
11080 RETURN
```

Computer Sketchpad SKETCH

The SKETCH program lets you draw a figure on the screen using the keyboard. The eight keys shown in Figure SKETCH-1 let you move a "cursor" on the screen up, down, left, right, or diagonally. As the cursor moves it leaves a line. The cursor moves one character position at a time, so you can draw any type of figure you want by moving the cursor.

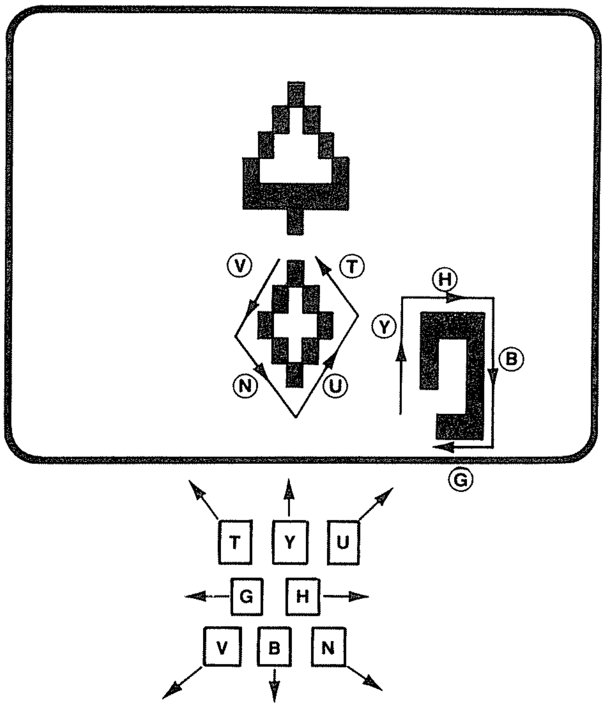


Figure SKETCH-1. Cursor Movement Keys

You can also change color on the cursor, clear the screen, or move the cursor without leaving a "trail," or colored line.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

The screen will clear and you'll see a yellow-colored cursor on a blue background in the center of the screen. If you want a different colored cursor or background, see I(nk) and P(aper) below.

To move the cursor up, press the Y key. Each time you press the Y key, the cursor will move up one screen character position. When the cursor gets to the top character position, it will appear at the bottom of the screen and move up from that point.

To move the cursor down, press the B key. The cursor will move one screen character position for every key press, appearing at the top of the screen after it gets to the bottom of the screen.

To move the cursor right or left, press the H (right) or G (left) keys. The cursor will move right or left one character position and appear on the opposite side of the screen when it goes past the side of the screen.

To move the cursor diagonally use these keys:

- The U key moves the cursor up and to the right
- The N key moves the cursor down and to the right
- The V key moves the cursor down and to the left
- The T key moves the cursor up and to the left

Again, like the other functions, the cursor will appear at the opposite side of the screen when it reaches a screen boundary.

To change the "ink" color, press the I key. Pressing the I key continuously will cycle the cursor color through green, yellow, blue, red, buff, cyan, magenta, and orange. Whatever color was last chosen will remain as the color for lines drawn by moving the cursor.

To erase a previous line, change the cursor color by the I key so that it matches the color of the line. Then move the cursor along the same route as you followed before and the line will be erased.

To turn off the cursor, press the "E" key (for "end"). The cursor will disappear, and you can move the cursor without drawing any lines. This is handy for repositioning the cursor to a new spot on the screen.

To check where you are, press the "S" key (for "start"), and then press the E key again. When you finally get to the spot on the screen in which you're interested, press "S" again to bring back the cursor.

To clear the screen press the P key. Continuously pressing the P key will clear the screen to different colors in the sequence of black, green, yellow, blue, red, buff, cyan, magenta, and orange.

Some Background on the Program

The main problem in SKETCH is in reading a single key from the keyboard. We can't use an INPUT command, as we'd like to detect a single keystroke and not a sequence of characters ended by pressing <ENTER>. The INKEY$ command allows us to read a single key and returns a single character string equal to the key pressed. Pressing the A key, for example, returns the single character string "A".

How This Program Works

SKETCH is a fairly simple program. The INKEY$ function in BASIC is used to read in the key press. If no key is pressed, INKEY$ responds with a "null" character or zero length string. Whenever a key is pressed, INKEY$ generates a single character string equal to the key.

Variables X and Y are set equal to the initial cursor position at X=32, Y=16. Every time a key is pressed, X and Y are adjusted based on the direction to travel. If the H key is pressed, for example, X is adjusted by adding 1. X and Y are checked for the screen boundaries, which occur at X=0, X=63, Y=0, and Y=31. If the boundaries are reached, X or Y or both are set equal to values corresponding to the opposite side of the screen.

A SET command sets the current position on the screen to the "ink" color, and then a loop is made back to detect the next key.

Special Notes

1. This program will work on any MC-10 or Color Computer system.
2. SKETCH operates on character positions (512 total) rather than graphics elements (2048 total).

```
100 C=1: I=2: P=3: X=32: Y=16
110 CLS(P)
120 A$=INKEY$
130 IF A$<>"" THEN 160
140 IF C=1 THEN RESET(X,Y): SET(X,Y,I)
150 GOTO120
160 IF A$<>"Y" THEN 190
170 IF Y>1 THEN Y=Y-2: GOTO560
180 Y=31: GOTO560
190 IF A$<>"B" THEN 220
200 IF Y<30 THEN Y=Y+2: GOTO560
210 Y=0: GOTO560
220 IF A$<>"H" THEN 250
230 IF X<62 THEN X=X+2: GOTO560
240 X=0: GOTO560
250 IF A$<>"G" THEN 280
260 IF X>1 THEN X=X-2: GOTO560
270 X=63: GOTO560
280 IF A$<>"U" THEN 330
290 IF X<62 THEN X=X+2: GOTO310
300 X=0
310 IF Y>1 THEN Y=Y-2: GOTO560
320 Y=31: GOTO560
330 IF A$<>"N" THEN 380
340 IF X<62 THEN X=X+2: GOTO360
350 X=0
360 IF Y<30 THEN Y=Y+2: GOTO560
370 Y=0: GOTO560
380 IF A$<>"V" THEN 430
390 IF X>1 THEN X=X-2: GOTO410
400 X=63
410 IF Y<30 THEN Y=Y+2: GOTO560
420 Y=0: GOTO560
430 IF A$<>"T" THEN 480
440 IF X>1 THEN X=X-2: GOTO460
450 X=63
460 IF Y>1 THEN Y=Y-2: GOTO560
```

```
470 Y=31: GOTO560
480 IF A$="E" THEN SET(X,Y,P): C=0: GOTO120
490 IF A$="S" THEN SET(X,Y,I): C=1: GOTO120
500 IF A$<>"I" THEN 530
510 IF I<8 THEN I=I+1: GOTO120
520 I=1: GOTO 120
530 IF A$<>"P" THEN 120
540 IF P<8 THEN P=P+1: GOTO110
550 P=0: GOTO110
560 IF C=1 THEN SET(X,Y,I)
570 GOTO120
```

Slot Machine SLOT

Here's a chance to play a slot machine without making a long trip to Las Vegas. SLOT draws a "one-armed bandit" on the screen, complete with movable arm (see Figure SLOT-1). Pressing a key cranks the arm, and you're off and running! The nicest thing about SLOT is that it doesn't require any silver dollars to play the machine; the worst thing about SLOT is that it also "pays off" in imaginary dollars. Jackpots are accompanied by a slightly off-key rendition of "We're in the Money"!

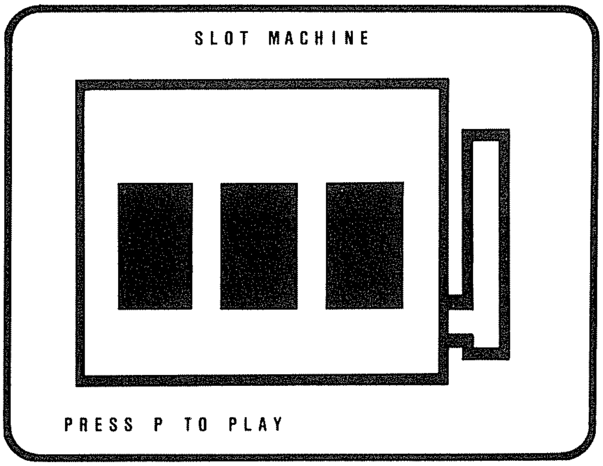


Figure SLOT-1. Slot Machine Figure

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the slot machine on the screen as shown in the figure above, with the message

```
PRESS P TO PLAY
```

Press the P key and you'll see the arm pull down and the three black windows on the face of the machine will display three combinations. Every time a new item appears in one of the windows, you'll hear a beep.

After each play, you'll see a payoff record on the bottom of the screen, as in:

```
CURR. PAYOFF=$ 6           WON=$ 213
PRESS P TO PLAY           LOST=$ 197
```

The "current payoff" indicates how much, if any, the three combinations in the window are worth. The "won" figure indicates how much you've won so far. The "lost" figure indicates how much you've lost so far. The total amount you're ahead or behind is the "won" amount minus the "lost" amount." We've done it this way so that you can see by the "lost" amount how many times you've played the machine. Each play is worth one "Las Tandy" dollar.

There are 8 separate symbols that may appear in the windows, shown in Figure SLOT-2. Three "diamonds" pay off $50 and give you four bars of "We're in the Money." Three of any kind (other than diamonds) pays off $6 and gives you two bars of "We're in the Money." Two diamonds or two "squares" in windows 1 and 2 also pay off $6, but without the song. Two of any kind (other than diamonds or squares) in windows 1 and 2 pay off $3. One square in window 1 also pays off $3.

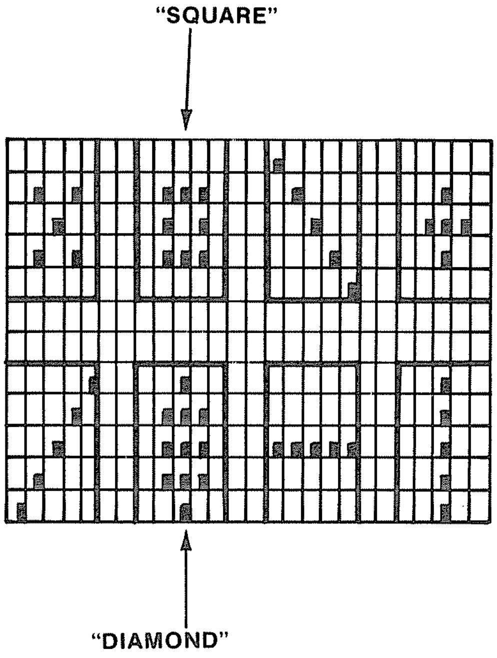


Figure SLOT-2. SLOT Symbols

SLOT will play just as fast as you can hit the P key. As in Las Vegas, the odds are with the "house," in this case an MC-10 or Color Computer, so don't expect to win heavily!

Some Background on the Program

Each window of the slot machine can display one of 8 patterns. The payoff values are related to the "order" of the patterns - a single square in window 1 pays off, but a single square in window 3 doesn't. The appearance of the pattern is more or less "random" - you can't predict when a pattern is going to show up. How can you figure out the odds in this game?

Suppose that we give each pattern a number from 0 to 7. How many different combinations of patterns can we have? We could start listing them quite easily:

```
000
001
002
003
004
005
006
007
010
011
012
```

The list goes on and on until it ends up with:

```
770
771
772
773
774
775
776
777
```

How many different combinations were represented? What we really have here is an "octal" representation of the octal numbers (base 8) from 000 through 777. In octal representation, the position of the digit represents a power of 8, instead of a power of 10, as in decimal. The maximum number represented is octal 777, or 7 times 8 cubed + 7 times 8 squared + 7 times 8 to the first power. This is:

```
7 × 64 = 448
7 ×  8 =  56
7 ×  1 =   7
         ---
         511
```

Counting 0, there are 512 combinations of patterns that can appear in the window. Actually, we're using the wrong word – since the order of the patterns is important, there are 512 **permutations,** or arrangements of patterns.

How many times do three diamonds in a row come up in the windows? Once every 512 times, just as "777" or "666" appear only once in the 512 patterns of 000 through 777. Similarly, we can find out when the other winning patterns come up and pay off accordingly – any three patterns appear 8 times out of 512 times – 000, 111, 222, 333, 444, 555, 666, and 777. A square comes up 64 times out of 512 – if a square is a "0," then we'd have 000, 001, 002, . . . 077.

Using random numbers, we can predict that over many plays, the winning patterns will appear in this known proportion and pay off accordingly, keeping the odds in the "house's favor."

How This Program Works

There are a number of parts to this program, but the heart of it is in statement 770. The RND(8) generates a number from 1 through 8 which is used to access pattern 1 through 8. The patterns are stored initially as values in DATA statements, but are moved to array T$ when the program starts. There are 5 "rows" per pattern and there are 8 patterns, so there are a total of 40 DATA values. Each DATA value defines 5 "columns" of the figure by five digits of "0" or "1" ("off" or "on"). The characters are accessed using the LEFT$, MID$, and RIGHT$ commands in the "Print Figures" portion of the program.

The last line of DATA values represents the SOUND frequency and duration values to play "We're in the Money." (There's more on SOUND generation in the BACH program.)

The main code of the program is towards the beginning. It "calls" the "Print Figure" code as a subroutine to get the three pattern values (P1, P2, and P3 – 1 through 8) and display the patterns in the windows. The "main-line" code then tests for the payoff patterns of P1, P2, and P3 and adjusts the "Won" and "Lost" totals accordingly.

A large portion of the program is devoted to displaying the outline of the slot machine, the handle, and animating the movement of the handle. The animation of the handle is done by rapidly displaying strings of CHR$ data that make up the handle outlines at the proper times and place on the screen.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. To get more "randomness," the statement at line 190 performs a series of pseudo-random operations until the P key is pressed. If this were not present, the sequence of patterns would be exactly repeatable each time the computer was turned on and the program run.
3. Change line 380 from "GOTO180" to "GOTO220" to make the program run continuously without human intervention.

```
100 TL=0: TW=0: DIM T$(8,5)
110 CLS: PRINT@9,"SLOT MACHINE"
120 FOR I=1 TO 8
130 FOR J=1 TO 5
140 READ T$(I,J)
150 NEXT J,I
160 GOSUB390
170 GOSUB530
180 PRINT@480,"PRESS P TO PLAY";
190 A$=INKEY$: ZZ=RND(1000)
200 IF A$="" THEN 190
210 IF A$<>"P" THEN 190
220 GOSUB690
230 IF NOT (P1=P2 AND P1=P3) THEN 270
240 GOSUB 980
250 IF P1=6 THEN TC=50: TW=TW+50: GOSUB 980: GOTO310
260 TC=6: TW=TW+6: GOTO310
270 IF (P1=P2) AND (P1=6 OR P1=2) THEN TC=6: TW=TW+6: GOTO310
280 IF P1=P2 THEN TC=3: TW=TW+3: GOTO310
290 IF P1=2 THEN TC=3: TW=TW+3: GOTO310
300 TC=0: TL=TL+1
310 REM -- PRINT TOTALS
320 PRINT@461,"      ";
330 PRINT@448,"CURR.PAYOFF=$";TC;
340 PRINT@473,"      ";
350 PRINT@468,"WON=$";TW;
360 PRINT@505,"      ";
370 PRINT@499,"LOST=$";TL;
380 GOTO180
390 REM -- SUBR. FOR OUTLINE
400 FOR I=66 TO 88
410 PRINT@I,CHR$(131);: NEXT I
420 PRINT@89,CHR$(130);
430 FOR I=121 TO 409 STEP 32
440 PRINT@I,CHR$(138);: NEXT I
450 PRINT@441,CHR$(136);
460 FOR I=440 TO 418 STEP -1
470 PRINT@I,CHR$(140);: NEXT I
480 PRINT@417,CHR$(132);
490 FOR I=385 TO 97 STEP -32
500 PRINT@I,CHR$(133);: NEXT I
510 PRINT@65,CHR$(129);
520 RETURN
530 REM -- SUBR. FOR HANDLE UP
540 PRINT@347,"  ";
550 PRINT@410,CHR$(131)+CHR$(132)+CHR$(140)+CHR$(136);
560 PRINT@381,CHR$(138);
570 FOR H=347 TO 187 STEP -32
580 PRINT@H,CHR$(133)+" "+CHR$(138);: NEXT H
590 PRINT@155,CHR$(129)+CHR$(131)+CHR$(130);
600 PRINT@346,CHR$(140);
610 RETURN
620 REM--SUBR. FOR HANDLE DOWN
630 FOR H=154 TO 410 STEP 32
640 PRINT@H,"      ";: NEXT H
650 PRINT@410,CHR$(131)+CHR$(132)+CHR$(140)+CHR$(136);
660 PRINT@381,CHR$(138);
670 PRINT@346,CHR$(140)+CHR$(129)+CHR$(131)+CHR$(130);
680 RETURN
690 REM -- PRINT FIGURES
700 GOSUB620
```

```
710 FOR J=0 TO 4
720 FOR P=196 TO 210 STEP 7
730 PRINT@P+(J*32),CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128);
740 NEXT P,J
750 GOSUB530
760 FOR P=196 TO 210 STEP 7
770 I=RND(8)
780 IF P=196 THEN P1=I
790 IF P=203 THEN P2=I
800 IF P=210 THEN P3=I
810 K=((I-1)*16)+1
820 FOR J=1 TO 5
830 A=VAL(LEFT$(T$(I,J),1))*K
840 B=VAL(MID$(T$(I,J),2,1))*K
850 C=VAL(MID$(T$(I,J),3,1))*K
860 D=VAL(MID$(T$(I,J),4,1))*K
870 E=VAL(RIGHT$(T$(I,J),1))*K
880 F=128
890 PRINT@P+((J-1)*32),CHR$(F+A)+CHR$(F+B)+CHR$(F+C)+CHR$(F+D)+CHR$(F+E);
900 NEXT J: SOUND 135,1
910 NEXT P
920 RETURN
930 DATA 00000,01010,00100,01010,00000,00000,01110,01010,01110,00000
940 DATA 10000,01000,00100,00010,00001,00000,00100,01110,00100,00000
950 DATA 00001,00010,00100,01000,10000,00100,01110,01110,01110,00100
960 DATA 00000,00000,11111,00000,00000,00100,00100,00100,00100,00100
970 DATA -1,185,4,197,2,1,1,185,4,189,4,197,6,0,0
980 RESTORE
990 READ ZZ: IF ZZ<>-1 THEN 990
1000 READ ZT,ZD: IF ZT=0 THEN 1050
1010 IF ZT=1 THEN 1030
1020 SOUND ZT,ZD
1030 FOR ZT=1 TO 20:NEXT ZT
1040 GOTO 1000
1050 RETURN
```

Spanish Drill Program SPANISH

¿Habla usted español? No? This program is a tutorial program that will drill you in translating Spanish nouns to English and vice versa. We've included a short list of words for 4K MC-10 systems, but you can add as many as you would like by adding to the DATA statements at the program end. In 32K Color Computer systems, the maximum number of Spanish words would be approximately 500.

How to Use This Program

See the German Drill Program GERMAN for instructions on using the program and program content. The Spanish version works exactly the same way.

```
100 CLEAR 500: CLS: DIM T$(50,2), N(5)
110 L=0
120 IF L<50 THEN 130
125 PRINT@64,"TOO MANY TABLE ENTRIES,"
127 PRINT@96,"SPANISH DRILL ABORTED": STOP
130 L=L+1
140 READ T$(L,1), T$(L,2)
150 IF (T$(L,1)="-1") OR (T$(L,2)="-1") THEN L=L-1: GOTO170
160 GOTO120
170 R=0: T=0
180 CLS: PRINT@6,"SPANISH NOUN DRILL"
190 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
200 PRINT@96,"1. TRANSLATE FROM SPANISH-TO-"
210 PRINT@131,"ENGLISH"
220 PRINT@160,"2. TRANSLATE FROM ENGLISH-TO-"
230 PRINT@195,"SPANISH"
240 PRINT@224,"WHICH ONE";: INPUT A$
250 IF A$="1" OR A$="2" THEN A=VAL(A$): GOTO290
260 PRINT@448,"INVALID SELECTION--TRY AGAIN"
270 FOR I=1 TO 300: NEXT I
280 PRINT@234,"": GOTO240
290 IF A$="1" THEN B=2: N$="SPANISH-TO-ENGLISH"
300 IF A$="2" THEN B=1: N$="ENGLISH-TO-SPANISH"
310 CLS: PRINT@7,N$
320 FOR I=0 TO 5
330 N(I)=RND(L)
340 IF N(I)=0 THEN 330
350 J=0
360 IF (N(I)=N(J)) AND (I<>J) THEN 330
370 IF J<(I-1) THEN J=J+1: GOTO360
380 NEXT I
390 I=RND(5)
400 IF I<1 THEN 390
410 PRINT@64,T$(N(0),A)
420 N(I)=N(0): N(0)=I
430 PRINT@128,"CHOOSE ONE OF THE FOLLOWING:"
440 J=128
450 FOR I=1 TO 5
460 J=J+32: B$=STR$(I)
470 PRINT@J,MID$(B$,2,1)+". "+T$(N(I),B)
480 NEXT I
490 PRINT@320,"WHICH ONE";: INPUT B$
500 IF (B$>"0") AND (B$<"6") THEN 520
510 PRINT@329,"": GOTO490
```

```
520 T=T+1
530 IFVAL(B$) <> N(0) THEN 560
540 R=R+1
550 PRINT@384,R;"OUT OF";T;"CORRECT!": GOTO580
560 PRINT@384,"NO! THE CORRECT ANSWER IS";N(0)
580 PRINT@416,"ENTER R FOR RESTART, OR JUST"
590 PRINT@448,"<ENTER> FOR SAME";: INPUT B$
600 IF NOT (B$="R" OR B$="") THEN PRINT@465,"": GOTO590
610 IF B$="R" THEN GOTO170
620 GOTO310
1000 DATA "EL PADRE","THE FATHER","EL HIJO","THE SON","LA VACA","THE COW"
1010 DATA "LA MADRE","THE MOTHER","EL CUERPO","THE BODY","EL SOL","THE SUN"
1020 DATA "EL DINERO","THE MONEY","EL POETA","THE POET","EL DIA","THE DAY"
1030 DATA "LA MANZANA","THE APPLE","LA NARANJA","THE ORANGE"
1040 DATA "EL MAPA","THE MAP","EL CALOR","THE HEAT",LA MANO","THE HAND"
1050 DATA "EL CLIMA","THE CLIMATE","EL GATO","THE TOMCAT","EL PAN","THE BREAD"
1060 DATA "LA PRIMAVERA","THE SPRING","EL OTONO","THE FALL","EL TORO","THE BULL"
1070 DATA "EL INVIERNO","THE WINTER","EL VERANO","THE SUMMER"
1100 DATA -1,-1
```

Statistical Analysis STATS

STATS performs statistical analysis functions on a list of numerical data. The data is ordered from lowest value to highest (this is called "sorting" in computer terms) and the ordered list is then displayed. The "median" value (midpoint of all values) is marked. Next, the "frequency distribution" of the list is displayed. Each unique value along with the number of occurrences of that value are displayed, from lowest value to highest. Next, the sum total of all the entries in the list, the arithmetic average (mean), the sum of the squares, the variance, and standard deviation are displayed. The program can then be restarted for an entry of a new list.

How to Use This Program

Enter the program into the MC-10 or Extended BASIC Color Computer. Start the program by entering

```
     RUN
```

You should see the title message and a "prompt" message appear as follows:

```
     STATISTICAL ANALYSIS

ENTER NUMBER,
 END BY 1E30?
```

You should now enter a list of values. If you had a list of 20 test scores, for example, you would enter 20 values, ending each by pressing the <ENTER> key. After the last value, enter

```
1E30
```

followed by <ENTER>. "1E30" is used as a unique ending value so that you can enter any values up to 1E30, or 1 times 10 to the 30th power.

After entering "1E30", you'll see the screen clear, and the message

```
SORTING..................
```

will be displayed. Periods will be printed as long as the sorting takes place. The time for the sort will be greater for a longer list of data. For 20 items, it's about 6 seconds.

At the end of the sort, you'll see a display of the list of ordered data. In the case of 20 test scores of 23, 50, 55, 75, 60, 98, 62, 20, 75, 72, 68, 66, 82, 85, 57, 96, 66, 70, 69, and 10, for example, you'd see:

```
LIST OF ORDERED DATA
 N       VALUE
 1         10
 2         20
 3         23
 4         50
 5         55
 6         57
 7         60
 8         62
 9         66
 10        66 (MEDIAN)
 11        68
 12        69
PRESS ANY KEY TO CONTINUE
```

To see the remainder of the ordered test scores, press any key. Note that the median value is denoted by a (MEDIAN) tag.

After the last set of ordered values is displayed, you'll see a

```
PRESS ANY KEY TO CONTINUE
```

message. After pressing any key, you'll see the frequency distribution list. In the case of the test scores above, you'll see:

```
FREQUENCY DISTRIBUTION
 #             # TIMES
 10               1
 20               1
 23               1
 50               1
 55               1
 57               1
 60               1
 62               1
 66               2
 68               1
 69               1
 70               1
PRESS ANY KEY TO CONTINUE
```

Pressing any key continues the list. After displaying the last portion of the frequency distribution, the program will display the remaining statistical parameters. In the case of the test scores above, you'd see:

```
SUM OF ENTRIES= 1259
MEAN= 62.95
SUM OF SQUARES= 89447
VARIANCE= 509.647503
STAN DEVIATION= 22.5753738
PRESS ANY KEY TO CONTINUE
```

Pressing any key restarts the program for entry of the next list of data.

Some Background on the Program

There are many sets of numbers whose graph is a "normal" or "bell-shaped" curve shown in Figure STATS-1. Some examples are the heights of potato plants, weights of TRS-80 programmers, and algebra test scores. The "standard deviation" of such a set of numbers is handy to know, because in any set whose graph is a normal curve, 68% of the values are within 1 standard deviation, and 96% of the values are within 2 standard deviations.

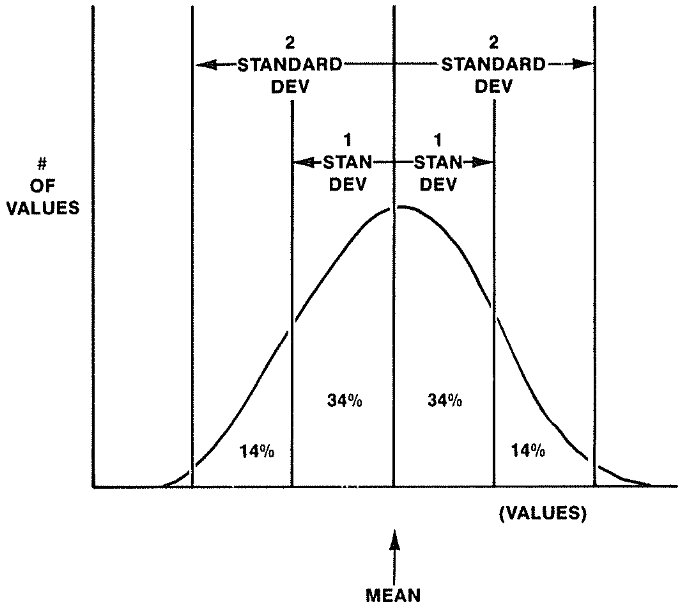


Figure STATS-1. Normal Curve

The standard deviation we've been talking about is computed by the following procedure:

1. Take all of the values and find the total sum.
2. Divide by the total number of values to find the "mean average."
3. Find the difference between each number in the set and the mean average.
4. Square the differences.
5. Sum the squares.

6. Find the mean average of the squares.
7. Take the square root of this average. The result is the "standard deviation."

In the test scores example above, we'd have

```
    Number          Number-Mean          Square of Diff
      23               -39.95              1596.0025
      50               -12.95               167.7025
      55                -7.95                63.2025
      75                12.05               145.2025
      60                -2.95                 8.7025
      98                35.05              1228.5025
      62                -0.95                 0.9025
      20               -42.95              1844.7025
      75                12.05               145.2025
      72                 9.05                81.9025
      68                 5.05                25.5025
      66                 3.05                 9.3025
      82                19.05               362.9025
      85                22.05               486.2025
      57                -5.95                35.4025
      96                33.05              1092.3025
      66                 3.05                 9.3025
      70                 7.05                49.7025
      69                 6.05                36.6025
      10                52.95              2803.7025
      ---                                  ---------
    1259/20=                             10192.9500/20=
      62.95                                509.6475

Square root of 509.6475 = 22.575 = standard deviation
```

A related formula that can be used is:

Standard deviation = Square Root of the variance

where

Variance= (Sum of all the squares of the values)/
(# of values) – mean squared

How This Program Works

The first part of this program reads in all of the user values, arranging them in array A from first to last. After the last entry, this array is sorted by a "bubble sort." The bubble sort operates by swapping two pairs of entries at a time, working from beginning to end and making as many passes as required to sort all of the entries. After the sort, array A is listed.

Next, the array is scanned for the frequency distribution of values. Each value is printed (along with the current count) only if the value following is different.

The sum of entries, mean, sum of the squares, variance, and standard deviation are then printed. The standard deviation is found by the second formula above, subtracting the square of the mean from the sum of the squares (variable TS) divided by the number of entries.

Special Notes

1. This program will run on all MC-10 and Extended BASIC Color Computer systems.
2. There are no restrictions on the size of each entry except that it cannot be equal to or exceed 1E+30 (10 to the 30th power).

```
100 DIM A(300)
110 I=1
120 CLS: PRINT @5,"STATISTICAL ANALYSIS"
130 PRINT @64,"ENTER NUMBER,";CHR$(13);" END BY 1E30";: INPUT A(I)
140 IF A(I)=1E+30 THEN 160
150 I=I+1: PRINT @111,"" : PRINT : GOTO 130
160 IF I>3  THEN 180
170 PRINT"MUST BE 3 ENTRIES OR GREATER": FOR K=1 TO 1000: NEXT K:GOTO 110
180 CLS: PRINT"SORTING";
190 C=0
200 FOR J=1 TO I-2
210 PRINT".";
220 IF A(J+1)>=A(J) THEN 240
230 B=A(J): A(J)=A(J+1): A(J+1)=B: C=1
240 NEXT J
250 IF C=1 THEN 190
260 CLS: LP=3
270 PRINT "LIST OF ORDERED DATA"
280 PRINT " N      VALUE"
290 T=0: TS=0
300 FOR J=1 TO I-1
310 T=T+A(J): TS=TS+A(J)^2
320 PRINT J;TAB(8);A(J);
330 IF J=INT(I/2) THEN PRINT "(MEDIAN)": GOTO 350
340 PRINT
350 GOSUB 580
360 NEXT J
370 GOSUB 600
380 CLS: LP=3
390 PRINT"FREQUENCY DISTRIBUTION"
400 PRINT " #             #TIMES"
410 C=1: LC=1: N=1
420 FOR J=1 TO I-1
430 IF A(J+1)=A(J) THEN C=C+1: GOTO 470
440 PRINT A(J),C: IF C>LC THEN  LC=C: N=A(J)
450 GOSUB 580
460 C=1
470 NEXT J
480 GOSUB 600
490 CLS
500 PRINT "SUM OF ENTRIES=";T
510 PRINT "MEAN=";T/(I-1)
```

```
520 PRINT"SUM OF SQUARES=";TS
530 V=(TS/(I-1))-(T/(I-1))^2
540 PRINT "VARIANCE=";V
550 PRINT"STAN DEVIATION=";SQR(V)
560 GOSUB 600
570 GOTO 110
580 LP=LP+1: IF LP<>15 THEN 630
590 LP=1
600 PRINT"PRESS ANY KEY TO CONTINUE"
610 A$=INKEY$: IF A$="" THEN 610
620 CLS
630 RETURN
```

Telephone Directory Program TELDIR

Need to look up a telephone number? The Telephone Directory program is an easy way to do it. It will also look up any string of characters that you want, so that you can use it not only for telephone numbers, but for addresses, notes, or any other text information that you wish to save on a permanent basis.

The Telephone Directory program also lists all the entries that it finds that contain the text you're searching for, so it's a convenient way to list all "714" area code telephone numbers, or all names that have "BILL" as the first name.

How to Use This Program

Enter the program into the MC-10 or Color Computer.

Establishing the DATA list
List the program by doing a

```
LIST
```

command and look at the last BASIC lines. In this version, the last lines from 480 through 560 are DATA statements containing names and telephone numbers. These are "dummy" numbers put in for examples in this section. You'll have to replace these lines with your own DATA. To do this delete lines 480 through 550 by entering

```
480
490
500
510
520
530
540
550
```

LIST the program again. You should see line 470 followed by line 560. **The last line of the program should always be "DATA -1".**

You can now enter your own telephone or other type of list. You don't have to alphabetize the list – just enter it in any order. You can start numbering the lines from 480 and increment the line numbers by 1 – 480, 481, 482, and so forth, or you can use any other line number increment you wish.

Each line must have a line number, a DATA command, a double quote, and then your text entry. Put another double quote at the end of the entry.

Make each DATA line a separate entry – in this example we've used a name and telephone number. The maximum number of characters in the entire line can be about 250, but don't forget that you'll have to display 250 characters on 8 separate lines, so it's a good idea to keep the entries short. It's also a good idea to keep short entries to save memory when you're using a 4K MC-10.

Leave the program as it is with the "dummy" entries if you want to see how it works before adding your own DATA.

Remember to save the program and your DATA by doing a

```
      CSAVE
```

to cassette or a

```
      SAVE
```

to disk. This will save the program and the entries you have made, which are part of the program.

Running the Program

To run the program enter

```
      RUN
```

You'll see the title and prompt message

```
    TELEPHONE DIRECTORY

ENTER A STRING OF DATA TO BE
USED AS A SEARCH KEY
?
```

You can now enter any number of characters from one character to up to the size of your longest entry. Suppose you want to search for all telephone numbers with a (212) prefix. You'd enter:

```
? (212)
```

followed by <ENTER>.

The program would now display the message

```
SEARCH LIST
```

and look for any entry with a "(212)" in it. In the case of the dummy DATA originally in the program, you'd see

```
SEARCH LIST
HOLLERITH, H. (212) 555-2399
```

Remember that the data must be "unique" for the search to work. If you wanted to search for all "212" occurrences, you'd get not only the (212) prefix, but the entry that contained the string "555-1212":

```
SEARCH LIST
BARDEN, BILL (714) 555-1212
HOLLERITH, H. (212) 555-2399
```

If you want to display the entire list, simply type in <ENTER> in place of a string. You'll see the entire list displayed on the screen.

If the list is too long to fit on the screen, you'll see the message

```
PRESS ANY KEY TO CONTINUE
```

after the screen has been filled. Pressing any key will display the next portion of the list.

After the entire list has been displayed, the message

```
PRESS R TO RESTART
```

is displayed. Entering R followed by <ENTER> will bring you back to the message for entering the search string.

You can add entries to the list as described above to the limits of memory. If you go over 50 entries however, do this:

1. Change line 110 from

```
    110 CLS: DIM T$(50)
```

to

```
    110 CLS: DIM T$(XXX)
```

where XXX is the total number of DATA entries you've added plus the last

```
    LLLL DATA -1
```

entry.

2. Delete line 130.
3. If you're adding entries and get an "OM" error, for "Out of Memory," you'll have to reduce the number of your entries, or abbreviate them somewhat.

Some Background on the Program

Again, as in some of the other programs in this book, there are two main limitations for these types of programs, memory and speed.

Each character of a text string (as in the DATA statements) takes up one byte of memory. The string "BARDEN, BILL (714) 555-1212," for example, takes up 27 bytes of memory, as it has 27 characters, counting the blanks. Assuming that the average telephone entry has about 30 bytes, we can get about 33 entries in 1000 bytes of memory. However, if a string array is used, this will also take up an equivalent amount of memory when DATA is transferred from DATA statements to an array.

Speed is another consideration for a program such as this. It takes a long time to "scan through" a list of data, such as a long sequence of DATA statements. Reading random entries from DATA statements would mean starting at the beginning of the list (by doing a BASIC RESTORE) for each entry. For this reason, the entries in DATA statements are put into a string array to speed up the access, even though this uses additional memory.

How the Program Works

Array T$ is a one-dimensional string array that holds all entries. The array is initialized from the entire set of DATA statements that define the telephone list entries.

A search is done by comparing the search string with each entry of the array. The BASIC MID$ command is used to do this. The MID$ command searches the middle of a text string for a search string, and that's exactly what we're after here. Of course, each entry in the array (which is really a DATA entry) is searched from beginning to the point at which the remaining characters in the string are smaller than the search string.

There is no search made between entries - you couldn't search for "1212 PENNINGTON", for example, and have the program find the string in entries 1 and 2 of the dummy data.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.
2. If there's more than one occurrence of a search string within a line, the line is only shown once.

```
100 CLEAR 200
110 CLS: DIM T$(50)
120 L1=0
130 IF L1<50 THEN 160
140 PRINT@64,"TOO MANY TABLE ENTRIES,"
150 PRINT@96,"TELEPHONE DIRECTORY RUN ABORTED": STOP
160 L1=L1+1
170 READ T$(L1)
180 IF T$(L1)="-1" THEN L1=L1-1: GOTO200
190 GOTO130
200 CLS: PRINT@6,"TELEPHONE DIRECTORY"
210 PRINT@64,"ENTER A STRING OF DATA TO BE"
220 PRINT@96,"USED AS THE SEARCH KEY"
230 PRINT@160,"";: INPUT A$
240 CLS: PRINT@0,"SEARCH LIST": PRINT
250 T=0: PP=64
260 L3=LEN(A$)
270 FOR I=1 TO L1
280 J=0
290 L2=LEN(T$(I))
300 IF (J=>L2) OR (L3>(L2-J+1)) THEN 420
310 J=J+1
320 B$=MID$(T$(I),J,L3)
330 IF A$<>B$ THEN 300
340 T=T+1
350 C=INT((L2-1)/32)+1
360 IF (PP+(C*32))<480 THEN 400
370 PRINT@480,"PRESS ANY KEY TO CONTINUE";: K$=INKEY$
```

```
380 IF K$="" THEN 370
390 CLS: PP=64
400 PRINT@PP,T$(I)
410 PP=PP+(C*32)
420 NEXT I
430 IF T<1 THEN PRINT "NO MATCH FOUND"
440 PRINT@480,"PRESS R TO RESTART";: K$=INKEY$
450 IF K$="" THEN 440
460 IF K$="R" THEN 200
470 GOTO440
480 DATA "BARDEN, BILL (714) 555-1212"
490 DATA "PENNINGTON, HARV (714) 555-1000"
500 DATA "PASCAL, BLAISE (213) 555-2342"
510 DATA "HOLLERITH, H. (212) 555-2399"
520 DATA "BLECHMAN, FRED (213) 555-8765"
530 DATA "LUTZ, LEON (617) 555-9173"
540 DATA "SCROEDER, BILL (414) 555-9999"
550 DATA "E.T. (1) (%$$#) (2(&%$) %)4-7#!&"
560 DATA -1
```

Timer/Clock Program TIMPGM

The MC-10 or Color Computer can be used to replace an inexpensive digital clock. Although a computer is a **little** more expensive than a digital clock (10 or 20 times) it makes a very impressive clock display for relatives and friends. One hard fact of computer life, though – the digital clock may be more accurate than the computer for keeping time!

The TIMPGM program has two functions – it acts as a digital clock, displaying the hours, minutes, and seconds on the video display, or it acts as a stopwatch, displaying the "elapsed time" on the video display.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
        RUN
```

You should see the title and first prompt message as follows:

```
        TIMER/CLOCK PROGRAM

ENTER C TO BEGIN THE CLOCK, OR T
  FOR THE TIMER?
```

If you press T followed by <ENTER> at this point, you'll see an elapsed time display at the bottom of the screen that looks like this:

```
        0: 0: 0
```

The display represents hours, minutes, and seconds and will update every second. After an hour and 37 minutes and 5 seconds, for example, you'll have:

```
        1:37: 5
```

You can reset the timer by pressing any key. This will bring the program back to the main prompt message so that you can restart the timer or go to the clock function.

The clock function is started by answering C to the main prompt message. After you do this, you'll see:

```
        TIMER/CLOCK PROGRAM

ENTER C TO BEGIN THE CLOCK, OR T
 FOR THE TIMER? C

ENTER HOURS, MINUTES, SECONDS,
AM OR PM
(HH,MM,SS,AM/PM)?
```

You can now enter the time in the format HH,MM,SS,AM. For example, if the time was 1:37:34 PM, you'd enter

```
(HH,MM,SS,AM/PM)? 1,37,34,PM
```

followed by <ENTER>. Enter four seconds more than the actual time – it takes the program some time to "setup" the actual counting loop. Don't forget the "AM" or "PM."

If you make a mistake, you'll get the message

```
INVALID TIME--TRY AGAIN
```

After you've entered the time, you'll see a display of hours, minutes, and seconds on the screen, along with the AM or PM indication. It will update every second:

```
     1:41:17:PM
```

In addition to the display, you'll also hear a "beep" every second from the television speaker, and a double beep on the minute.

To get back to the main prompt message, just hold down any key.

Adjustment: Line 360 has a timing loop that uses the T variable to control the count of seconds. Adjust this timing loop by using a different ending value for T. You might want to start out with 500 for the Color Computer and 450 for the MC-10. Different versions of the MC-10 and Color Computer will require different ending counts for T. Here's a change that uses an ending value of 489 for T:

```
     360 FOR T=1 TO 489: NEXT T: SOUND 220,3
```

To calibrate the clock, run it for half a day or more, and keep adjusting the count for T.

Some Background on the Program

The MC-10 and Color Computer have an internal "clock" which controls the operation of machine-language instructions. The machine-language instructions in the BASIC interpreter operate at rates of hundreds of thousands of instructions per second, and the computer clock is even faster – millions of times per second. The computer clock is a precise frequency, but not nearly as accurate as the local power line frequency of 60 cycles per second (60 hertz), which controls a common digital or sweep-hand clock. In many cases, the power company adjusts the 60 cycle per second rate so that there are exactly 60 times 60 times 60 times 24 pulses per day.

The time for executing each type of machine-language instruction is fixed. The BASIC interpreter is made up of machine-language instructions. If the same "loop" of BASIC statements is "executed" continuously, the total number of machine-language instructions will be fixed and will require the same time each time through the loop.

We can make a BASIC clock program by establishing a timing loop as simple as

```
     100 FOR I=0 TO 1000: NEXT I
```

You might try this simple loop and vary the end value to different values to see how long each value takes.

If we establish a loop that takes exactly a second and then add one to a count, we've got a simple "seconds" counter.

How This Program Works

This program uses a simple timing loop of

```
100 FOR T=1 TO XXX: NEXT T: SOUND 220,3
```

to count seconds. Each time the loop is done, variable SS is incremented by one in the "FOR SS=0 TO 59" statement.

When SS reaches 60, the "minutes" count in MM is adjusted by one and SS is reset to 0. When MM reaches 60, the "hours" count in HH is adjusted by one and MM is reset to 0. When HH reaches 13, HH is reset to 0. The entire cycle repeats continuously.

Each time through the "seconds" loop, the display is updated with the correct time by changing the HH,MM,and SS values into strings and displaying them on the screen.

Special Notes

1. This program will run on all MC-10 and Color Computers, but will require adjustment of the timing loop as described above.
2. You can leave off leading zeroes for the time. Entering 1,7,4,PM is fine in place of 01,07,04,PM.
3. The "best case" adjustment of the timing loop in line 360 may still result in several minutes lost every 8 hours. For a finer adjustment, adjust the timing loop so that the clock is slightly slow and then add "dummy lines" between lines 350 and 360 to waste time. Start with a few REMark lines. If these do not do the trick, add lines with computation, such as ZZ=5.6/1.999 to perform useless but time consuming operations.
4. Pressing any key while the clock or timer is running will restart the program.

```
100 HH=0: MM=0: SS=0: TD$=" "
110 CLS
120 PRINT@6,"TIMER/CLOCK PROGRAM"
130 PRINT@64,"ENTER C TO BEGIN THE CLOCK, OR T FOR THE TIMER";
140 INPUT CT$
150 IF CT$="T" THEN 290
160 IF CT$="C" THEN 200
170 PRINT@160,"INVALID CLOCK OR TIMER INPUT -- TRY AGAIN"
180 FOR T=1 TO 400: NEXT T
190 PRINT@160,"": PRINT: GOTO130
200 PRINT@160,"ENTER HOURS, MINUTES, SECONDS,  AM OR PM"
210 PRINT@224,"(HH,MM,SS,AM/PM)";
```

```
220 INPUT HH,MM,SS,TD$
230 IF (HH<=-1) OR (HH>12) OR (MM<=-1) OR (MM>59) OR (SS<=-1) THEN 260
240 IF (SS>59) OR ((TD$<>"AM") AND (TD$<>"PM")) THEN 260
250 GOTO 290
260 PRINT@288,"INVALID TIME--TRY AGAIN"
270 FOR T=1 TO 400: NEXT T
280 PRINT@288,"";: PRINT: PRINT@241,"";: PRINT: GOTO200
290 REM TIMING LOOP
300 FOR HH=HH TO 12: HH$=STR$(HH)
310 FOR MM=MM TO 59: MM$=STR$(MM)
320 FOR SS=SS TO 59: SS$=STR$(SS)
330 PRINT@394,RIGHT$(HH$,2);":";RIGHT$(MM$,2);":";RIGHT$(SS$,2)
340 IF CT$="C" THEN PRINT@402,":";TD$
350 REM***CHANGE "T=1 TO 500" TO "T=1 TO 450" FOR COLOR COMPUTER***
360 FOR T=1 TO 500: NEXT T: SOUND 220,3
370 REM MONITOR KEYBOARD INPUT
380 A$=INKEY$
390 IF A$<>"" THEN 100
400 IF NOT(HH=11 AND MM=59 AND SS=59) THEN 430
410 IF TD$="AM" THEN TD$="PM": GOTO 430
420 TD$="AM"
430 NEXT SS: SS=1
440 SOUND 220,3
450 NEXT MM: MM=0
460 NEXT HH
470 HH=1: MM=1: SS=1
480 GOTO 290
```

Telephone Toll/Timer Program TOLLTIM

Spending too much money on long-distance calls? The MC-10 or Color Computer can be used to time telephone toll calls and display a running total of currect charges. The TOLLTIM program will display the elapsed time from the start of a telephone call on the screen, updating it every second. It also displays the current charages, based on either a one-minute initial charge plus additional minutes or a three-minute initial charge plus additional minutes. A warning beep is sounded ten seconds before the next minute has elapsed and at the minute time. The program will handle rates up to $100 and elapsed times up to 12 hours, which should handle all telephone calls except for extra terrestials calling home.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
      TELEPHONE TOLL/TIMER

SELECT ONE OF THE FOLLOWING:
1. FIRST 3-MINUTE RATE PLUS
   ADDITIONAL MINUTE RATE
2. FIRST 1-MINUTE RATE PLUS
   ADDITIONAL MINUTE RATE
WHICH ONE?
```

If your toll call is for a fixed dollar amount for the first three minutes, enter 1, followed by <ENTER>. If your toll call is for a fixed dollar amount for the first one minute, enter 2, followed by <ENTER>.

After you enter either a 1 or 2, you'll see:

```
     TELEPHONE TOLL/TIMER

FIRST 3-MINUTE RATE=    ?
```

or

```
     TELEPHONE TOLL/TIMER

FIRST 1-MINUTE RATE=    ?
```

Enter the dollar amount for the first period; use a decimal point to show cents. You'll now see:

```
ADDITIONAL MINUTE RATE=?
```

Enter the rate for each additional minute, using a decimal point.

You'll now see the message

```
PRESS ANY KEY TO BEGIN AFTER CALLING
PARTY ANSWERS
```

Dial the phone number, and when your party answers, press any key. You'll see the time and charges displayed in the middle of the screen:

```
TIME= 0: 0:23
TOTAL CHARGES= $ 1.00
```

The time will change every second and will "beep" 10 seconds before the next minute and at the next minute. The total charges will update on the minute.

When you are done with the call, press any key to end the elapsed time and charges.

Adjustment: Line 410 has a timing loop that uses the T variable to control the count of seconds. Adjust this timing loop by using a different ending value for T. You might want to start out with 500 for the Color Computer and 450 for the MC-10. Different versions of the MC-10 and Color Computer will require different ending counts for T. Here's a change that uses an ending value of 489 for T:

```
    360 FOR T=1 TO 489: NEXT T
```

Some Background on the Program

The MC-10 and Color Computer have an internal "clock" which controls the operation of machine-language instructions. The machine-language instructions in the BASIC interpreter operate at rates of hundreds of thousands of instructions per second, and the computer clock is even faster – millions of times per second. The computer clock is a precise frequency, but not nearly as accurate as the local power line frequency of 60 cycles per second (60 hertz), which controls a common digital or sweep-hand clock. In many cases, the power company adjusts the 60 cycle per second rate so that there are exactly 60 times 60 times 60 times 24 pulses per day.

The time for executing each type of machine-language instruction is fixed. The BASIC interpreter is made up of machine-language instructions. If the same "loop" of BASIC statements is "executed" continuously, the total number of machine-language instructions will be fixed and will require the same time each time through the loop.

We can make a BASIC clock program by establishing a timing loop as simple as

```
    100 FOR T=1 TO 1000: NEXT T
```

You might try this simple loop and vary the end value to different values to see how long each value takes.

If we establish a loop that takes exactly a second and then add one to a count, we've got a simple "seconds" counter.

How This Program Works

This program uses a simple timing loop of

```
100 FOR T=1 TO XXX: NEXT T
```

to count seconds. Each time the loop is done, variable SS is incremented by one in the "FOR SS=0 TO 59" statement.

When SS reaches 60, the "minutes" count in MM is adjusted by one and SS is reset to 0. When MM reaches 60, the "hours" count in HH is adjusted by one and MM is reset to 0. When HH reaches 13, the program terminates. The entire cycle repeats continuously for 12 hours.

Each time through the "seconds" loop, the display is updated with the correct time by changing the HH,MM,and SS values into strings and displaying them on the screen.

Special Notes

1. This program will run on all MC-10 and Color Computers, but will require adjustment of the timing loop as described above.
2. The "best case" adjustment of the timing loop in line 410 may still result in several minutes lost every 8 hours. For a finer adjustment, adjust the timing loop so that the clock is slightly slow and then add "dummy lines" between lines 350 and 360 to waste time. Start with a few REMark lines. If these do not do the trick, add lines with computation, such as ZZ=5.6/1.999 to perform useless but time consuming operations.

```
100 CLS: PRINT@6,"TELEPHONE TOLL/TIMER"
110 PRINT@6,"TELEPHONE TOLL/TIMER"
120 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
130 PRINT@96,"1. FIRST 3-MINUTE RATE PLUS"
140 PRINT@131,"ADDITIONAL MINUTE RATE"
150 PRINT@160,"2. FIRST 1-MINUTE RATE PLUS"
160 PRINT@195,"ADDITIONAL MINUTE RATE"
170 PRINT@224,"WHICH ONE";: INPUT A$
180 IF A$="1" THEN M$="3": GOTO230
190 IF A$="2" THEN M$="1": GOTO230
200 PRINT@448,"INVALID SELECTION--TRY AGAIN"
210 FOR I=1 TO 300: NEXT I
220 PRINT@234,"": PRINT@448,"": GOTO170
230 CLS: PRINT@6,"TELEPHONE TOLL/TIMER"
240 PRINT@64,"FIRST "+M$+"-MINUTE RATE=   ";: INPUT B$
250 IF (VAL(B$)>0) AND (VAL(B$)<100) THEN B=VAL(B$): GOTO270
260 PRINT@85,"": GOTO240
270 PRINT@96,"ADDITIONAL MINUTE RATE=";: INPUT B$
280 IF (VAL(B$)>0) AND (VAL(B$)<100) THEN C=VAL(B$): GOTO300
290 PRINT@119,"": GOTO270
300 PRINT@416,"PRESS ANY KEY TO BEGIN AFTER"
310 PRINT@448,"CALLING PARTY ANSWERS";: B$=INKEY$
```

```
320 IF B$="" THEN 310
325 PRINT@416,"PRESS ANY KEY TO END CHARGES": PRINT@448,""
330 REM - BEGIN TOLL/TIMER LOOP
335 T1=B
340 FOR HH=0 TO 12: HH$=STR$(HH)
350 FOR MM=0 TO 59: MM$=STR$(MM)
360 FOR SS=0 TO 59: SS$=STR$(SS)
370 PRINT@201,"TIME="+RIGHT$(HH$,2)+":"+RIGHT$(MM$,2)+":"+RIGHT$(SS$,2)
380 T2=INT(T1*100): T$=STR$(T2): L=LEN(T$)
390 PRINT@256,"TOTAL CHARGES= $"+LEFT$(T$,L-2)+"."+RIGHT$(T$,2)
400 REM***CHANGE "T=1 TO 500" TO "T=1 TO 450" FOR COLOR COMPUTER***
410 FOR T=1 TO 500: NEXT T
420 IF SS=50 THEN SOUND 220,3
430 B$=INKEY$
440 IF B$<>"" THEN 520
450 NEXT SS: SS=1
460 SOUND 220,3
480 IF (MM<2) AND (A$="1") THEN 500
490 T1=T1+C
500 NEXT MM: MM=0
510 NEXT HH
520 PRINT@416,"ENTER R TO RESTART, OR JUST"
530 PRINT@448,"<ENTER> FOR SAME RATE";: INPUT B$
540 IF B$="R" THEN 100
550 IF B$="" THEN PRINT@206,"": PRINT@271,"": GOTO300
560 GOTO530
```

Trigonometry Functions TRIG1

TRIG1 performs basic trigonometry functions. A similar program, TRIG2, is a quiz on trigonometry functions. TRIG1 works with a right triangle as shown in Figure TRIG1-1. To use TRIG1, you enter the length of any one of the three sides and one of the two unknown angles. The remaining sides and angles are then found by the program and displayed.

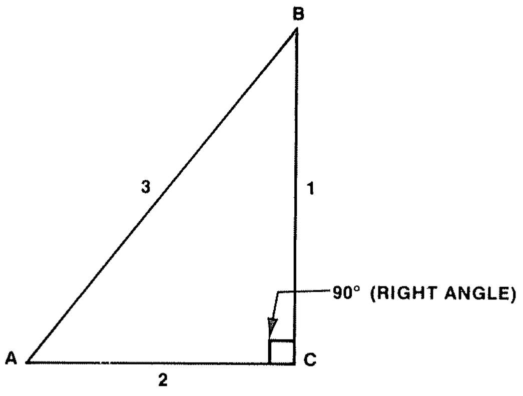


Figure TRIG1-1. Right Triangle

How to Use This Program

Enter the program into the MC-10 or Extended BASIC Color Computer. Start the program by entering

```
     RUN
```

You should see the title message and the following display:

```
   TRIGONOMETRIC FUNCTIONS

FIND VALUES (ENTER ONE SIDE
AND ONE ANGLE):

                 SIDE 1=?
                      2=
                      3=

                ANGLE A=
                      B=
                      C=90
```

A triangle is also displayed on the left side of the screen and looks like Figure TRIG1-1.

You can now enter either side 1, 2, or 3. To enter a value for side 1, enter the value and then press <ENTER>. To enter a value for side 2 or 3, press <ENTER> alone for the preceding values. <ENTER> alone is the same as telling the program that a side is "unknown."

After you've entered a side value the program will skip down to the angle entry. Again, you can enter a value followed by <ENTER> for any one of the two angles. Angles are in degrees from 1 through 89.

After you've entered the angle, TRIG1 will print out the results. Here's an example:

```
   TRIGONOMETRIC FUNCTIONS

 FIND VALUES (ENTER ONE SIDE
 AND ONE ANGLE):

                    SIDE 1=? 1
                         2=  1.732052
                         3=  2.000001

                   ANGLE A=  30
                         B=  60
                         C=  90

 PRESS R TO RESTART
```

You can now press the R key to restart the program for the next calculation.

Some Background on the Program

A "right triangle" is a triangle with a 90 degree angle as one of the angles. The total of all the angles in a triangle is 180 degrees, so the remaining two angles must total 90 degrees.

If you know one side and one angle of a right triangle, the remaining sides and angle may be easily found by using the sine, cosine, and tangent functions. These functions are called "trigonometric" functions.

The sine of angle A (see Figure TRIG1-1) is defined as the value of side 1 divided by side 3. The cosine is defined as side 2 divided by side 3. The tangent is defined as side 1 divided by side 2. No matter how large or how small the triangle, the sine, cosine, and tangent values for any given angle remain constant. If you had a table of all sine, cosine, and tangent values for a range of angles and knew one angle and the length of one side, you could use the table to find the remaining sides and angle.

Suppose, for example, that you knew angle A was 30 degrees and that side 3 was 2 feet. The sine function is side 1 divided by side 3, so we'd have:

sine 30 = side 1/side 3 = (side 1)/2

Looking up the value of sine 30 in a table from a math book, we'd find 0.5, so we'd now have:

0.5 = (side 1)/2 or (side 1)= 0.5 x 2 = 1

Cosine and tangent values can be used in the same way. Before digital computers you had to use a table or "slide rule" to find cosine, sine, and tangent values. In microcomputers, however, you can easily find any sine, cosine, or tangent values by using the SIN, COS, or TAN functions.

If any two sides of a right triangle are known, then the third can be found by using the "Pythagorean Theorem," thousands of years old. It says that side 3 equals the square root of the length of side 1 squared plus side 2 squared, or in computer terms:

Side 3 = SQR ((side 1)↑2 +(side 2)↑2)

How This Program Works

Most of this program is concerned with creating the display. A small triangle is drawn on the left side of the display by using graphics characters (CHR$(136), CHR$(137), CHR$(138)). The text is then displayed by PRINT@ statements.

After the side and angle have been entered, the remaining angle is found by subtracting the given angle from 90 degrees.

The missing sides are found by using the TAN and COS functions, along with the Pythagorean Theorem. The SIN, COS, and TAN functions all use "radians" for their arguments, so the input values in degrees must first be translated to radians. A radian is the radius of a circle laid around the circumference of a circle. About 6.28 radii can be laid around the circumference of a circle – actually, exactly 2 times pi radians, as the circumference of a circle is equal to 2*pi*r. One radian equals about 57.2958279 degrees and dividing by this value converts degrees to radians for use in SIN, COS, and TAN.

Special Notes

1. This program will run on all MC-10 and Extended BASIC Color Computer systems.
2. Angle values less than 1 degree or greater than 89 degrees are not accepted.
3. Side values greater than 0 and less than 100000 are valid.

```
100 DIM AG(2), SD(3)
110 CLS: PRINT@4,"TRIGONOMETRIC FUNCTIONS"
120 PRINT@98,"FIND VALUES (ENTER ONE SIDE"
130 PRINT@130,"AND ONE ANGLE):"
140 PRINT@262,CHR$(136);: REM--DRAW TRIANGLE
150 PRINT@386,CHR$(136)+"    "+CHR$(136);
160 FOR I=387 TO 389
170 PRINT@I,CHR$(140);: NEXT I
180 FOR I=294 TO 358 STEP 32
190 PRINT@I,CHR$(138);: NEXT I
200 FOR I=293 TO 355 STEP 31
210 PRINT@I,CHR$(137);: NEXT I
220 PRINT@385,"A";: PRINT@391,"C";
230 PRINT@231,"B";: PRINT@323,"3";
240 PRINT@327,"1";: PRINT@420,"2";
```

```
250 PRINT@237,"SIDE 1=";: PRINT@274,"2=";
260 PRINT@306,"3=";: PRINT@364,"ANGLE A=";
270 PRINT@402,"B=";: PRINT@434,"C=  90"
280 REM -- FIND VALUES
290 S=0: P=212: REM--FIND SIDE
300 IF NOT (S<3) THEN 290
310 S=S+1
320 PRINT@P+(S*32),"";: INPUT A$
330 IFVAL(A$)=0 THEN 300
340 IF (VAL(A$)>0) AND (VAL(A$)<=99999) THEN SD(S)=VAL(A$): GOTO360
350 PRINT@P+(S*32),"": GOTO320
360 A=0: P=340: REM--FIND ANGLE
370 IF NOT(A<2) THEN 360
380 A=A+1
390 PRINT@P+(A*32),"";: INPUT A$
400 IFVAL(A$)=0 THEN 370
410 IF (VAL(A$)>=1) AND (VAL(A$)<=89) THEN AG(A)=VAL(A$): GOTO430
420 PRINT@P+(A*32),"": GOTO390
430 REM -- FORMULAS
440 IF A=1 THEN AG(2)=90-AG(1)
450 IF A=2 THEN AG(1)=90-AG(2)
460 IF S<>1 THEN 480
470 SD(2)=SD(1)/TAN(AG(1)/57.2958279): GOTO500
480 IF S<>2 THEN 510
490 SD(1)=SD(2)*TAN(AG(1)/57.2958279)
500 SD(3)=SQR((SD(1)^2)+(SD(2)^2)): GOTO530
510 SD(2)=SD(3)*COS(AG(1)/57.2958279)
520 SD(1)=SQR((SD(3)^2)-(SD(2)^2))
530 P=212: REM--PRINT SIDES
540 FOR I=1 TO 3
550 IF I=S THEN 580
560 A$=STR$(SD(I))
570 PRINT@P+(I*32),"   "+MID$(A$,2,8)
580 NEXT I
590 P=340: REM--PRINT ANGLE
600 FOR I=1 TO 2
610 IF I=A THEN 640
620 A$=STR$(AG(I))
630 PRINT@P+(I*32),"   "+MID$(A$,2,8)
640 NEXT I
650 PRINT@480,"PRESS R TO RESTART";: A$=INKEY$
660 IF A$="R" THEN 110
670 GOTO650
```

Trigonometry Quiz TRIG2

TRIG2 is a quiz on basic trigonometry functions. A similar program, TRIG1, calculates trigonometry functions. TRIG2 works with a right triangle as shown in Figure TRIG1-1. TRIG2 quizzes you by displaying a right triangle with one side and one angle known. You must then enter the remaining two sides and one angle (the third angle is always 90 degrees).

How to Use This Program

Enter the program into the MC-10 or Extended BASIC Color Computer. Start the program by entering

```
    RUN
```

You should see the title message and a display similar to this one:

```
   TRIGONOMETRIC FUNCTIONS

 QUIZ (GIVEN ONE SIDE AND
 ONE ANGLE):

                SIDE 1=?
                     2=  14
                     3=

               ANGLE A=  58
                     B=
                     C=  90
```

A triangle is also displayed on the left side of the screen and looks like Figure TRIG1-1.

You must now enter the remaining two sides and one angle. Only a value within one degree or one unit for a side's length will be accepted. Values that are greater than one degree or unit length will be ignored. Enter a value by typing in the amount followed by <ENTER>. After you enter each value, the program will automatically advance to the next entry.

If you've entered values for the two unknown sides and one unknown angle correctly (within one degree or unit), you'll see the message

```
PRESS R TO RESTART
```

displayed on the bottom of the screen. Pressing the R key will display the next quiz.

Some Background on the Program

See the discussion in TRIG1.

How This Program Works

Most of this program is concerned with creating the display. A small triangle is drawn on the left side of the display by using graphics characters

(CHR$(136), CHR$(137), CHR$(138)). The text is then displayed by PRINT@ statements.

A random length side from 1 through 99 is found by RND(99). A random angle from 5 through 85 is found by RND(81)+4. These values are displayed on the screen, and the program then waits for the entry of the other values. As each value is entered it is compared with the previously computed value for the side or angle.

Special Notes

1. This program will run on all MC-10 and Extended BASIC Color Computer systems.
2. Angles values less than 5 degrees or greater than 85 degrees are not used in the quiz.
3. Side values greater than 99 units are not used in the quiz.

```
100 DIM AG(2), SD(3)
110 CLS: PRINT@4,"TRIGONOMETRIC FUNCTIONS"
120 PRINT@99,"QUIZ (GIVEN ONE SIDE AND"
130 PRINT@131,"ONE ANGLE):"
140 PRINT@262,CHR$(136);: REM--DRAW TRIANGLE
150 PRINT@386,CHR$(136)+"   "+CHR$(136);
160 FOR I=387 TO 389
170 PRINT@I,CHR$(140);: NEXT I
180 FOR I=294 TO 358 STEP 32
190 PRINT@I,CHR$(138);: NEXT I
200 FOR I=293 TO 355 STEP 31
210 PRINT@I,CHR$(137);: NEXT I
220 PRINT@385,"A";: PRINT@391,"C";
230 PRINT@231,"B";: PRINT@323,"3";
240 PRINT@327,"1";: PRINT@420,"2";
250 PRINT@237,"SIDE 1=";: PRINT@274,"2=";
260 PRINT@306,"3=";: PRINT@364,"ANGLE A=";
270 PRINT@402,"B=";: PRINT@434,"C=  90"
280 REM -- QUIZ
290 S=RND(3): A=RND(2): REM--SELECT RND SIDE/ANGLE
300 SD(S)=RND(99): AG(A)=RND(81)+4
310 A$=STR$(SD(S))
320 PRINT@214+(S*32),MID$(A$,2,6)
330 A$=STR$(AG(A))
340 PRINT@342+(A*32),MID$(A$,2,4)
350 REM -- FORMULAS
360 IF A=1 THEN AG(2)=90-AG(1)
370 IF A=2 THEN AG(1)=90-AG(2)
380 IF S<>1 THEN 400
390 SD(2)=SD(1)/TAN(AG(1)/57.2958279): GOTO420
400 IF S<>2 THEN 430
410 SD(1)=SD(2)*TAN(AG(1)/57.2958279)
420 SD(3)=SQR((SD(1)^2)+(SD(2)^2)): GOTO450
430 SD(2)=SD(3)*COS(AG(1)/57.2958279)
440 SD(1)=SQR((SD(3)^2)-(SD(2)^2))
450 I=0: P=212: REM--QUIZ SIDES
460 IF NOT(I<3) THEN 510
470 I=I+1: IF I=S THEN 460
480 PRINT@P+(I*32),"";: INPUT A$
490 IF (VAL(A$)=<(SD(I)+1)) AND (VAL(A$)=>(SD(I)-1)) THEN 460
```

```
500 PRINT@P+(I*32),"": GOTO480
510 I=0: P=340: REM--QUIZ ANGLES
520 IF NOT (I<2) THEN 570
530 I=I+1: IF I=A THEN 520
540 PRINT@P+(I*32),"";: INPUT A$
550 IF (VAL(A$)=<(AG(I)+1)) AND (VAL(A$)=>(AG(I)-1)) THEN 520
560 PRINT@P+(I*32),"": GOTO540
570 PRINT@480,"PRESS R TO RESTART";: A$=INKEY$
580 IF A$="R" THEN 110
590 GOTO570
```

Vocabulary Drill Program VOCAB

Do you consider yourself an erudite person? No? Maybe this program will help. It's a vocabulary drill that quizzes you about random words. We've included a short list of words for 4K MC-10 systems, but you can add as many as you would like by adding to the DATA statements at the program end. In 32K Color Computer systems, the maximum number of Vocabulary words would be approximately 500.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
     RUN
```

You should see the title and first prompt message as follows:

```
      VOCABULARY DRILL

SELECT ONE OF THE FOLLOWING:
1. FIND WORD FROM MEANING
2. FIND MEANING FROM WORD
WHICH ONE?
```

You can now select one or the other by entering 1 or 2, followed by <ENTER>.

If you select the "word from meaning" quiz, you'll see the first multiple choice selection, which will look something like this:

```
     WORD FROM MEANING

REMISSION

CHOOSE ONE OF THE FOLLOWING:
1. SEA-GOING VESSEL
2. ABATEMENT OR DIMINUTION
3. GROWING BY ACCUMULATION
4. COMPOSURE UNDER STRAIN
5. AWAKENING MEMORIES
WHICH ONE?
```

Entering 1 through 5, followed by <ENTER> will record your choice, and the program will display either the correct answer or the total number correct, as in:

```
 1 OUT OF 1 CORRECT!
ENTER R FOR RESTART, OR JUST
<ENTER> FOR SAME?
```

Pressing <ENTER> will display the next set of multiple choice entries. Entering R followed by <ENTER> will bring you back to the program start so that you can select the alternate drill.

Selecting the "meaning to word" quiz works exactly the same way, except that the multiple choice entries give words, one of which has to be selected for the meaning at the top of the screen.

To add entries to the program, do this:

1. Add a DATA statement before the last DATA statement of DATA -1,-1. You can add as many data statements as you have room for in memory. Each data statement has this form:

   line# DATA "(vocabulary word)","(meaning of word)"

   You can put several word/meanings in one line, as long as each Vocabulary word is followed by its meaning in the same line.

   Be sure to put double quotation marks around each Vocabulary word and each meaning and be certain to include a comma between each entry.

   You can add as many DATA statements as you'd like, as long as you have enough memory, and as long as the last DATA statement is

   ```
   DATA -1,-1
   ```

2. Change the DIM command in line 100 for the number of entries that you have. If you have 100 DATA statements total and each DATA statement has two word/meaning entries, then you'll have 200 entries total. Change the DIM command in this case to DIM T$(100,2). The "2" in T$(100,2) doesn't change.

3. Delete line 120 by entering

   ```
   120
   ```

   from BASIC.

4. Run the program. If you get the title message, then you have enough memory for the entries you've added. You may get an "OM" or "Out of Memory" error. In this case you'll have to delete some of the entries you've added.

5. When the program starts successfully, save the program on cassette by the CLOAD command or on disk by the SAVE command.

Some Background on the Program

Computers make very efficient instructors for repetitive material such as this. There are two main limitations for these types of programs, memory and speed.

Each character of a text string (as in the DATA statements) takes up one byte of memory. The string "ANTIDISESTABLISHMENTARIANISM", for example, takes up 28 bytes. Assuming that the average word and meaning string has about 30 characters, we can get about 33 word/meaning equivalents in 1000 bytes of memory. However, if a string array is used, this will also take up an equivalent amount of memory as DATA is transferred from DATA statements to an array.

Speed is another consideration for a program such as this. It takes a long time to "scan through" a list of data, such as a long sequence of DATA statements. Reading random entries from DATA statements would mean starting at the beginning of the list (by doing a RESTORE) for each entry. For this reason, the entries in DATA statements are put into a string array to speed up the access, even though this uses additional memory.

How the Program Works

Array T$ is a two-dimensional string array that holds the word in the first element and the meaning in the second element. The array is initialized from the entire set of DATA statements that define the word/meanings to be used.

For the multiple choice translation, a random word/meaning is computed by using the RND command with L, the number of word/meaning entries. The program knows the equivalent noun, as it is the adjacent entry in the array. The program then finds 4 multiple choice entries at random by using RND(L) again and displays the 4 entries along with the proper selection.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 CLEAR: CLS: DIM T$(50,2), N(5)
110 L=0
120 IF L<50 THEN 130
125 PRINT@64,"TOO MANY TABLE ENTRIES,"
127 PRINT@96,"VOCABULARY DRILL ABORTED": STOP
130 L=L+1
140 READ T$(L,1), T$(L,2)
150 IF (T$(L,1)="-1") OR (T$(L,2)="-1") THEN L=L-1: GOTO170
160 GOTO120
170 R=0: T=0
180 CLS: PRINT@7,"VOCABULARY DRILL"
190 PRINT@64,"SELECT ONE OF THE FOLLOWING:"
200 PRINT@96,"1. FIND WORD FROM MEANING"
220 PRINT@160,"2. FIND MEANING FROM WORD"
240 PRINT@224,"WHICH ONE";: INPUT A$
250 IF A$="1" OR A$="2" THEN A=VAL(A$): GOTO290
260 PRINT@448,"INVALID SELECTION--TRY AGAIN"
270 FOR I=1 TO 300: NEXT I
280 PRINT@234,"": GOTO240
290 IF A$="1" THEN B=2: N$="WORD FROM MEANING"
300 IF A$="2" THEN B=1: N$="MEANING FROM WORD"
310 CLS: PRINT@7,N$
320 FOR I=0 TO 5
330 N(I)=RND(L)
340 IF N(I)=0 THEN 330
350 J=0
360 IF (N(I)=N(J)) AND (I<>J) THEN 330
370 IF J<(I-1) THEN J=J+1: GOTO360
```

```
380 NEXT I
390 I=RND(5)
400 IF I<1 THEN 390
410 PRINT@64,T$(N(0),A)
420 N(I)=N(0): N(0)=I
430 PRINT@128,"CHOOSE ONE OF THE FOLLOWING:"
440 J=128
450 FOR I=1 TO 5
460 J=J+32: B$=STR$(I)
470 PRINT@J,MID$(B$,2,1)+".  "+T$(N(I),B)
480 NEXT I
490 PRINT@320,"WHICH ONE";: INPUT B$
500 IF (B$>"0") AND (B$<"6") THEN 520
510 PRINT@329,"": GOTO490
520 T=T+1
530 IFVAL(B$) <> N(0) THEN 560
540 R=R+1
550 PRINT@384,R;"OUT OF";T;"CORRECT!": GOTO580
560 PRINT@384,"NO! THE CORRECT ANSWER IS";N(0)
580 PRINT@416,"ENTER R FOR RESTART, OR JUST"
590 PRINT@448,"<ENTER> FOR SAME";: INPUT B$
600 IF NOT (B$="R" OR B$="") THEN PRINT@465,"": GOTO590
610 IF B$="R" THEN GOTO170
620 GOTO310
1000 DATA "ERUDITE","LEARNED","FACILE","DONE WITH EASE"
1010 DATA "PREEMPTIVE","IN PREFERENCE TO"
1020 DATA "SCURRILOUS","ABUSIVE","GALLERY","LONG COVERED AREA"
1030 DATA "GALLEY","SEA-GOING VESSEL"
1040 DATA "UNANIMOUS","IN COMPLETE ACCORD"
1050 DATA "EQUANIMITY","COMPOSURE UNDER STRAIN"
1060 DATA "REMISSION","ABATEMENT OR DIMINUTION"
1070 DATA "REMINISCENT","AWAKENING MEMORIES"
1080 DATA "AMBIENCE","MOOD OR TONE"
1090 DATA "CUMULATIVE","GROWING BY ACCUMULATION"
1100 DATA "SCUD","MOVE QUICKLY OR HURRIDLY"
1110 DATA "FONT","TYPE STYLE"
1120 "BAS-RELIEF","SCULPTURE WITH FIGURES"
1130 DATA "ALLUSION","CASUAL REFERENCE"
1140 DATA "ILLUSION","SOMETHING THAT DECEIVES"
1150 DATA "ANTIDISESTABLISHMENTARIANISM","A LONG WORD"
5000 DATA -1,-1
```

Simple Word Processor WORDS

WORDS is a simple word processor that will allow you to enter text into the system, modify the text, save the text in memory and on cassette, and print out the text on your system line printer. It isn't meant to be a full-blown "word processor" but does allow you to easily handle up to about 400 words of text on a cassette-based system. That's enough to draft an indignant "letter to the editor" or to handle most business correspondence.

The word processor works with a screenful of text at a time. A colored "cursor" can be moved around the screen so that you can add to or modify text. Commands are present to let you store and read 5 different screen pages, to print all of the pages, or to save and read the pages to cassette tape.

This program requires a 16K Color Computer or MC-10 with memory expansion pack.

How to Use This Program

There are two versions of this program. The Color Computer version is called WORDS and the MC-10 version is called WORDSMC. Enter the proper program into the MC-10 or Color Computer. Start the program by entering

```
RUN
```

You should see the screen clear. An orange-colored cursor that blinks on and off will appear in the upper left-hand corner of the screen.

Entering Text

To enter text, simply start typing. The screen will fill up with the text you're typing. When the end of the line is reached, the cursor will move to the start of the next line. When the end of the 15th line is reached, the cursor will move back to the beginning of the screen. Note that the last line of the screen is left free for messages as shown in Figure WORDS-1.

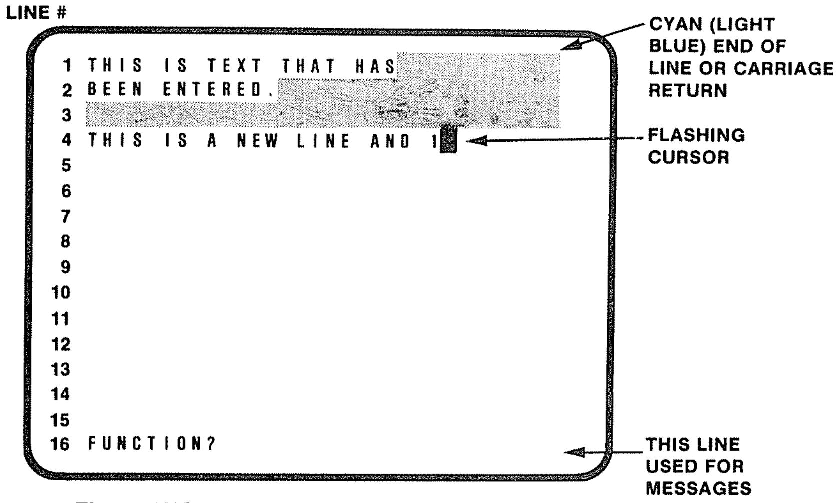

LINE #
1 THIS IS TEXT THAT HAS
2 BEEN ENTERED.
3
4 THIS IS A NEW LINE AND 1
5
6
7
8
9
10
11
12
13
14
15
16 FUNCTION?
CYAN (LIGHT BLUE) END OF LINE OR CARRIAGE RETURN
FLASHING CURSOR
THIS LINE USED FOR MESSAGES

Figure WORDS-1. WORDS Program Screen Layout

Moving the Cursor
Use the arrow keys to move the cursor to any part of the screen. Down arrow moves the cursor down until the 15th line, at which point it will reappear at the top of the screen. Up arrow moves the cursor up until it goes past the top of the screen: it will reappear on the 15th line. Right arrow moves the cursor to the right. Moving it past the right edge makes it reappear at the start of the next line. Left arrow moves the cursor to the left. Moving it past the left edge makes it reappear at the end of the previous line.

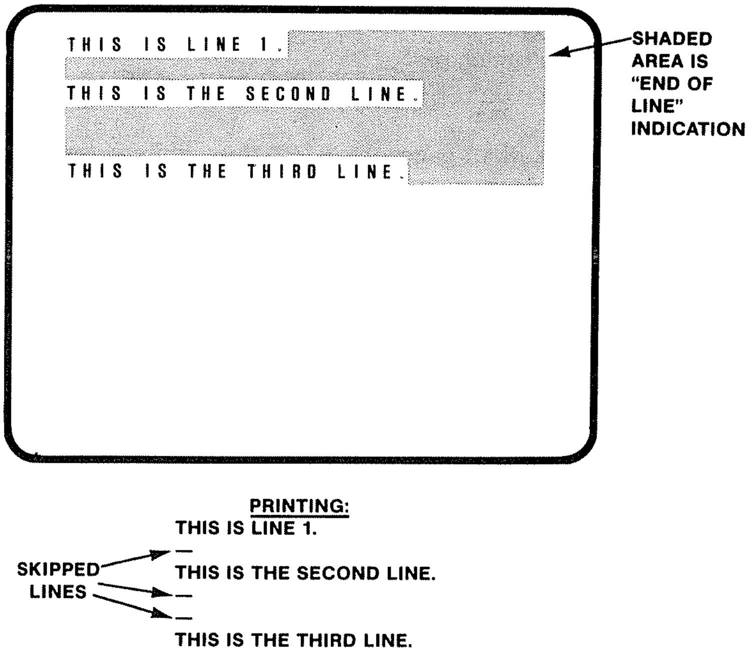
Note that on the MC-10 arrow keys are controlled by pressing the CONTROL key.

To Overwrite Text
To overwrite any text, move the cursor by the arrow keys to the point at which you want to overwrite existing text. Then type normally.

To Erase Text
To erase text, position the cursor and type blanks by pressing the SPACE bar.

To Back Up and Delete
To erase while moving to the left, position the cursor and press the @ key repeatedly. The cursor will move to the left and erase text.

To End a Line
A line is ended by pressing <ENTER>. If you don't press <ENTER> all of the text from the beginning of the line to the next <ENTER> will be printed on one line on the system line printer when you print the text. If there are more characters in the line then you have on the line printer, you may have incorrect formatting. Figure WORDS-2 shows lines of text and the result as it appears after printing.



```
THIS IS LINE 1.

THIS IS THE SECOND LINE.


THIS IS THE THIRD LINE.
```

PRINTING:

```
THIS IS LINE 1.
_
THIS IS THE SECOND LINE.
_
_
THIS IS THE THIRD LINE.
```

Figure WORDS-2. Ending a Line

Pressing ENTER results in a cyan (light blue) shaded area from the point at which you pressed <ENTER> to the end of the line. This is normal.

To skip lines press <ENTER> at the beginning of each line.

To Save a Screen

To save a screenful of information after you've edited it on the screen, press SHIFT, followed by up arrow. You'll see the message

```
FUNCTION?
```

appear on the bottom of the screen.

Now enter S1 through S5, **without pressing <ENTER>**. The screen will be saved in memory in "page" 1 through 5. This will take a few seconds. You'll know that the save is done when the cursor reappears. The contents of the screen will not be altered after a save.

To Retrieve a Page

To retrieve a page of information previously saved, press SHIFT, followed by up arrow. You'll see the message

```
FUNCTION?
```

appear on the bottom of the screen.

Now enter G1 through G5, **without pressing <ENTER>**. The page will be retrieved from memory and restored to the screen. Any previous screen contents will be lost. The "get" will take a few seconds. You'll know that the get is done when the cursor reappears.

To Write to Cassette

To write the contents of all pages to cassette tape, rewind the tape to the point at which you wish to write and setup the cassette for writing (PLAY and RECORD).

Now press SHIFT, followed by up arrow. You'll see the message

```
FUNCTION?
```

appear on the bottom of the screen.

Now enter W (for write) **without pressing <ENTER>.** The text from pages 1 through 5 will be saved on cassette as a file. The screen contents won't be altered. Note that this write copies all of the **page** contents to cassette, but does not write the **screen contents.** The write will take about two minutes. You'll know when the write is done when the cursor reappears. On the Color Computer version of this program, you'll see periods displayed on the screen as the write is taking place.

To Read from Cassette

To read the contents of a previously written file from cassette, rewind the tape to the point at which you file is located and setup the cassette for reading (PLAY).

Now press SHIFT, followed by up arrow. You'll see the message

```
FUNCTION?
```

appear on the bottom of the screen.

Now enter L (for load) **without pressing <ENTER>.** The text from pages 1 through 5 will be read from cassette as a file. After the load, the first page of the file will appear on the screen. The load will take two minutes or so. You'll know that the load is done when the cursor reappears. On the Color Computer version of this program, periods will be displayed on the screen as the load takes place.

To Print the Pages

To print the five text pages (not the screen), "ready" the printer by turning it on and putting it "on line." Press SHIFT, followed by up arrow. You'll see the message

```
FUNCTION?
```

appear on the bottom of the screen.

Enter P (print) **without an <ENTER>.** The contents of the 5 text pages will now be printed out on the system line printer.

Some Background on the Program

The contents of the video display for the Color Computer and MC-10 are located in RAM memory. The "hardware" for the two systems reads the text data in memory and converts it into a television-compatible signal to update the video display.

The video display "buffer" for the Color Computer starts at location 1024 and continues through location 1535. The video display buffer for the MC-10 is located from RAM locations 16384 through 16895.

Normally we'd write data to the video display by doing BASIC PRINTs, PRINT@s, and so forth. When we use these BASIC commands we're working with a software "display driver" in the BASIC interpreter that handles the overhead tasks of putting the character data in the proper place in the display buffer, doing "line feeds," and so forth.

To implement a word processor, however, we can't easily work through the display driver because it has too much "overhead." We can get to the video display buffers directly, though, by using PEEKs and POKEs. A BASIC PEEK allows us to read the contents of the video display buffer in RAM. A POKE allows us to write values out to the video display buffer.

When PEEKs and POKEs are used in this fashion, we simply treat the video display buffer as any other memory location, which it is. We avoid the time-consuming overhead of the software display driver in BASIC. On the other side of the coin, though, we have to perform all of the functions normally handled in the software display driver in our own program.

WORDS handles text by using the INKEY$ function to input a character at a time. That character is POKEd into the video display buffer, if it is a "normal" text character. A cursor is displayed in WORDS by POKEing a graphics character to the proper point on the screen. A "cursor location" is kept by recording the current row (0 through 14) and column (0 through 31) and updating it based on the screen actions.

Text is transferred from screen to memory by PEEKing from the video display buffer and converting the value from the PEEK into a numeric value. This numeric value is then stored in an array. As there are 5*480 characters in 5 pages, a 2400-entry array is used.

Text is transferred from memory to the screen by POKEing data from the array "page" into the video display buffer area.

How This Program Works

The cursor position is kept in two variables, R and C. Variable R is the row number from 0 through 14 (the 16th row, row 15, is used for messages). Variable C is the column number from 0 through 31. R and C are updated based on the character input from the keyboard. If the character input is a normal text character, then 1 is added to the C value until the end of the line is reached, at which point C is set to 0 and 1 is added to R. At the end of the screen both R and C are reset to 0.

If the character input from the keyboard is a cursor movement character, then R and C are adjusted based on the action taken. An up arrow, for example, will cause 1 to be subtracted from R and for R to be reset to 14 if the cursor is at the top row.

The special character created by SHIFT, up arrow initiates a special function. A following "S" causes the current video display buffer to be stored in 480 bytes of array A. There are 5 separate blocks in the array, based on which "page" is being used.

The "G" character causes a block of 480 bytes from the array to be read and stored in the video display buffer thereby updating the screen.

A "W" writes out the entire contents of array A to cassette. The commands for doing this are somewhat different for the Color Computer and MC-10. The Color Computer version of the program uses an "OPEN", "PRINT#-1", and "CLOSE" sequence to write a sequential file on cassette. The MC-10 uses the special CSAVE* command to write out array A directly to cassette.

An "L" character reads in a cassette file in similar fashion to the Write. Again, there are two versions of this, one for the Color Computer, and one for the MC-10.

A "P" character simply sends all characters from the A array to the line printer. The PRINT command does check for "carriage return," however, and ignores all characters after the first carriage return of the line.

Special Notes

1. This program will run on 16K Color Computer or MC-10 systems with expansion packs only.
2. If your printer is not "ready," the program will appear to "hang up" for the P command. The printer "ON LINE" indicator must be on, indicating power on, paper, etc.
3. You can overwrite a line in the "carriage return" area, but the text will be ignored during printing. Only the text up to the carriage return will be printed.

WORDS

```
100 DIM A(480*5): A(480*5)=0
110 CLS: R=0: C=0
120 CC=PEEK(1024+R*32+C)
130 A$=INKEY$: IF A$<>"" THEN 160
140 IF K=0 THEN POKE 1024+R*32+C,128: K=1: GOTO 130
150 POKE 1024+R*32+C,255: K=0: GOTO 130
160 POKE 1024+R*32+C,CC:A=ASC(A$):
170 IF A=94 THEN 330
```

```
180 IF A=95 THEN 400
190 IF (A<32) OR (A>127) THEN 250
200 IF A=64 THEN POKE 1024+R*32+C,96: GOTO 270
210 IF (A>31) AND (A<65) THEN A=A+64
220 POKE 1024+R*32+C,A
230 C=C+1: IF C=32 THEN C=0: R=R+1: IF R=15 THEN R=0
240 GOTO 120
250 IF (A<8) OR (A>13) THEN 120
260 ON A-7 GOTO 270,290,310,120,120,350
270 C=C-1: IF C=-1 THEN C=31: R=R-1: IF R=-1 THEN R=14
280 GOTO 120
290 C=C+1: IF C=32 THEN C=0: R=R+1: IF R=15 THEN R=0
300 GOTO 120
310 R=R+1: IF R=15 THEN R=0
320 GOTO 120
330 R=R-1: IF R=-1 THEN R=14
340 GOTO 120
350 FOR J=1 TO 32-C
360 POKE 1024+R*32+C,223
370 C=C+1: NEXT J
380 R=R+1: C=0: IF R=15 THEN R=0
390 GOTO 120
400 PRINT@480,"FUNCTION?";
410 A$=INKEY$: IF A$="" THEN 410
420 PRINT @490,A$;
430 IF A$="W" THEN 720
440 IF A$="L" THEN 810
450 IF A$="P" THEN 650
460 IF A$="S" THEN 560
470 IF A$<>"G" THEN 900
480 A$=INKEY$: IF A$="" THEN 480
490 PRINT @491,A$;:IF (A$<"1") OR (A$>"5") THEN 900
500 B=(VAL(A$)-1)*480
510 J=0
520 FOR I=B TO B+479
530 POKE 1024+J,A(I)
540 J=J+1: NEXT I
550 GOTO 900
560 A$=INKEY$: IF A$="" THEN 560
570 PRINT @491,A$;:IF (A$<"1") OR (A$>"5") THEN 900
580 B=(VAL(A$)-1)*480
590 J=0
600 FOR I=B TO B+479
610 A(I)=PEEK(1024+J)
620 J=J+1: NEXT I
630 A(480*5)=I-1
640 GOTO 900
650 FOR I=0 TO A(480*5)
660 IF A(I)=223 THEN PRINT#-2: I=INT(I/32)*32+31: GOTO 690
670 B=A(I): IF B>90 THEN B=A(I)-64
680 PRINT#-2,CHR$(B);
690 A$=INKEY$: IF A$<>"" THEN 900
700 NEXT I
710 GOTO 900
720 REM CASSETTE OUTPUT
730 CLS
740 OPEN "O",-1,"SCREEN"
750 FOR I=0 TO 480*5
760 PRINT#-1,A(I)
770 PRINT".";
780 NEXT I
```

```
790 CLOSE-1
800 A$="1": GOTO 500
810 REM CASSETTE INPUT
820 CLS
830 OPEN "I",-1,"SCREEN"
840 FOR I=0 TO 480*5
850 INPUT#-1,A(I)
860 PRINT".";
870 NEXT I
880 CLOSE-1
890 A$="1": GOTO 500
900 PRINT @480,"                ";:GOTO 120
```

WORDSMC

```
100 DIM A(480*5): A(480*5)=0
110 CLS: R=0: C=0
120 CC=PEEK(16384+R*32+C)
130 A$=INKEY$: IF A$<>"" THEN 160
140 IF K=0 THEN POKE 16384+R*32+C,128: K=1: GOTO 130
150 POKE 16384+R*32+C,255: K=0: GOTO 130
160 POKE 16384+R*32+C,CC:A=ASC(A$)
170 IF A=94 THEN 330
180 IF A=142 THEN 400
190 IF (A<32) OR (A>127) THEN 250
200 IF A=64 THEN POKE 16384+R*32+C,96: GOTO 270
210 IF (A>31) AND (A<65) THEN A=A+64
220 POKE 16384+R*32+C,A
230 C=C+1: IF C=32 THEN C=0: R=R+1: IF R=15 THEN R=0
240 GOTO 120
250 IF (A<8) OR (A>13) THEN 120
260 ON A-7 GOTO 270,290,310,120,120,350
270 C=C-1: IF C=-1 THEN C=31: R=R-1: IF R=-1 THEN R=14
280 GOTO 120
290 C=C+1: IF C=32 THEN C=0: R=R+1: IF R=15 THEN R=0
300 GOTO 120
310 R=R+1: IF R=15 THEN R=0
320 GOTO 120
330 R=R-1: IF R=-1 THEN R=14
340 GOTO 120
350 FOR J=1 TO 32-C
360 POKE 16384+R*32+C,223
370 C=C+1: NEXT J
380 R=R+1: C=0: IF R=15 THEN R=0
390 GOTO 120
400 PRINT@480,"FUNCTION?";
410 A$=INKEY$: IF A$="" THEN 410
420 PRINT @490,A$;
430 IF A$="W" THEN 720
440 IF A$="L" THEN 760
450 IF A$="P" THEN 650
460 IF A$="S" THEN 560
470 IF A$<>"G" THEN 800
480 A$=INKEY$: IF A$="" THEN 480
490 PRINT @491,A$;:IF (A$<"1") OR (A$>"5") THEN 800
500 B=(VAL(A$)-1)*480
510 J=0
520 FOR I=B TO B+479
530 POKE 16384+J,A(I)
540 J=J+1: NEXT I
```

```
550 GOTO 800
560 A$=INKEY$: IF A$="" THEN 560
570 PRINT @491,A$;:IF (A$<"1") OR (A$>"5") THEN 800
580 B=(VAL(A$)-1)*480
590 J=0
600 FOR I=B TO B+479
610 A(I)=PEEK(16384+J)
620 J=J+1: NEXT I
630 A(480*5)=I-1
640 GOTO 800
650 FOR I=0 TO A(480*5)
660 IF A(I)=223 THEN LPRINT: I=INT(I/32)*32+31: GOTO 690
670 B=A(I): IF B>90 THEN B=A(I)-64
680 LPRINT CHR$(B);
690 A$=INKEY$: IF A$<>"" THEN 800
700 NEXT I
710 GOTO 800
720 REM CASSETTE OUTPUT
730 CLS
740 CSAVE*A,"SCREEN"
750 A$="1": GOTO 500
760 REM CASSETTE INPUT
770 CLS
780 CLOAD*A,"SCREEN"
790 A$="1": GOTO 500
800 PRINT @480,"                  ";:GOTO 120
```

Jumbled Word Puzzle Program WJUMBLE

I'll have to admit that my mother-in-law is a whiz at Jumbled Word Puzzles – you know, the ones that appear in the daily paper. In these puzzles you're given four or five 5- or 6-letter words that are mixed up as far as the letters – PAPER might appear as AREPP, for example. Your task is to unscramble the letters. The way I finally got the better of my mother-in-law was to develop this program. Now I can just get the words unscrambled faster than she can.

How to Use This Program

Enter the program into the MC-10 or Color Computer. Start the program by entering

```
        RUN
```

You should see the title and first prompt message as follows:

```
          JUMBLED WORD PUZZLE

ENTER:

 JUMBLED WORD (2-6 CHR)?
```

You can now enter the jumbled word of from one to 6 characters, followed by an <ENTER>.

The program will then ask you

```
 TO PRINTER?
```

Answer Y or N, followed by <ENTER>.

If you've selected only a screen display, you'll now see all possible configurations of the words.

Suppose that you've entered the word UCDK ("DUCK" unscrambled). You'll see

```
PERMUTATION #          WORD
       1               UCDK
       2               UCKD
       3               UDKC
       4               UDCK
       5               UKCD
       6               UKDC
       7               CDKU
       8               CDUK
       9               CKUD
      10               CKDU
      11               CUDK
      12               CUKD
      13               DKUC
      14               DKCU
PRESS ANY KEY TO CONTINUE
```

To continue through all the listings, press any key. You'll see another screen full of words.

If you entered Y to the printer question, you'll also get a printout of all words along with the screen display.

Some Background on the Program

The program works the same way that English bell ringers used to (and probably still do) ring sequences of bells. If a bell ringer had 10 bells, he would go through as many "unique" sequences as he could, never repeating the entire sequence. If the bells were labeled ABCDEFGHIJ, for example, he might ring ABCDEFGHIJ, ABCDEFGHJI, ABCDEFGIHJ, and so forth, until he had no more sequences that had not been rung.

How many such sequences are there? Suppose we had 2 bells or two scrambled word letters. Call them A and B. Obviously there are only two sequences here, AB and BA.

Now consider three bells or letters - A, B, and C. Writing down all sequences we can think of, we can come up with

```
ABC, ACB, BAC, BCA, CAB, and CBA
```

For four bells or letters, we've got:

```
ABCD, ABDC, ACDB, ACBD, ADBC, ADCB, BCDA, BCAD, BDAC, BDCA,
BACD, BADC, CDAB, CDBA, CABD, CADB, CBDA, CBAD, DABC, DACB,
DBCA, DBAC, DCAB, DCBA.
```

For two things we had 2 configurations, for 3 things, 6 configuations, and for 4 things, 24 configurations. I'm calling these arrangements of letters "configurations" for a good reason. They're not combinations because they are dependent on a sequence or order. They're really **permutations** of bells or letters, in mathematical terms.

It turns out that if we have "n" things, then we have n! permutations. The term n! is called "n factorial" and it's what we get when we multiply all of the digits in n together. For example, that English bell ringer with 10 bells would have 1*2*3*4*5*6*7*8*9*10 permutations or sequences of bells to ring, or exactly 3,628,800 sequences. For smaller numbers of bells or letters we have:

```
1!=1
2!=1*2=2
3!=1*2*3=6
4!=1*2*3*4=24
5!=1*2*3*4*5=120
6!=1*2*3*4*5*6=720
```

How the Program Works

The program goes through all permutations of letters by dividing up to 6 letters into strings denoted by A$, B$, C$, D$, E$, and F$. A two-letter jumbled word would have only E$ and F$, a three-letter jumbled word would have D$, E$, and F$, on up to a six-letter word, which would have one-character strings A$, B$, C$, D$, E$, and F$. The word "USEMO" (MOUSE) would result in these string assignments:

```
A$=nothing
B$="U"
C$="S"
D$="E"
E$="M"
F$="O"
```

Based upon how many letters there are in the word, the letters are rotated into new arrangements. At the lowest level, E$ and F$ are swapped. The next level swaps D$, E$, and F$. The next, C$, D$, E$, and F$, and so on. The highest level calls the next lower level, which calls the next lower level, and so forth. The current configuration is printed out at the lowest level, along with a permutation number.

Special Notes

1. This program will run on all MC-10 and Color Computer systems.

```
100 P=0: LC=0
110 A$="": B$="": C$="": D$=""
120 CLS: PRINT@6,"JUMBLED WORD PUZZLE"
130 PRINT@64,"ENTER:"
140 PRINT@97,"JUMBLED WORD (2-6 CHR)";: INPUT N$
150 IFLEN(N$)>1 ANDLEN(N$)<7 THEN N=LEN(N$): GOTO170
160 PRINT@120,"": GOTO140
170 PRINT@129,"TO PRINTER (Y OR N)";: INPUT P$
```

```
180 IF P$="Y" OR P$="N" THEN 200
190 PRINT@149,"": GOTO170
200 F$=RIGHT$(N$,1): E$=MID$(N$,N-1,1)
210 IF N>2 THEN D$=MID$(N$,N-2,1)
220 IF N>3 THEN C$=MID$(N$,N-3,1)
230 IF N>4 THEN B$=MID$(N$,N-4,1)
240 IF N>5 THEN A$=MID$(N$,N-5,1)
250 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
260 IF P$="Y" THEN PRINT#-2, "PERMUTATION #        WORD"
270 ON N GOSUB310,360,350,340,330,320
280 PRINT@480,"PRESS R TO RESTART";: N$=INKEY$
290 IF N$="R" THEN 100
300 GOTO280
310 REM - PRINT SUBROUTINE
320 FOR I6=1 TO 6
330 FOR I5=1 TO 5
340 FOR I4=1 TO 4
350 FOR I3=1 TO 3
360 FOR I2=1 TO 2
370 IF LC<14 THEN 410
380 PRINT@480,"PRESS ANY KEY TO CONTINUE";: K$=INKEY$
390 IF K$="" THEN 380
400 LC=0
410 IF LC>0 THEN 440
420 CLS: PRINT@0,"PERMUTATION #        WORD"
430 PP=39
440 P=P+1: PRINT@PP-LEN(STR$(P)),P;
450 PRINT@PP+12,A$+B$+C$+D$+E$+F$
460 IF P$<>"Y" THEN 490
470 REM***CHANGE NEXT "PRINT#-2," TO "LPRINT" FOR MC-10***
480 PRINT#-2,TAB(7-LEN(STR$(P)))P;: PRINT#-2, TAB(20)A$+B$+C$+D$+E$+F$
490 LC=LC+1: PP=PP+32
500 T$=E$: E$=F$: F$=T$
510 NEXT I2
520 IF N=2 THEN 640
530 T$=D$: D$=E$: E$=F$: F$=T$
540 NEXT I3
550 IF N=3 THEN 640
560 T$=C$: C$=D$: D$=E$: E$=F$: F$=T$
570 NEXT I4
580 IF N=4 THEN 640
590 T$=B$: B$=C$: C$=D$: D$=E$: E$=F$: F$=T$
600 NEXT I5
610 IF N=5 THEN 640
620 T$=A$: A$=B$: B$=C$: C$=D$: D$=E$: E$=F$: F$=T$
630 NEXT I6
640 RETURN
```

Radio Shack
A Division of Tandy Corporation
Fort Worth, Texas 76102