ENCYCLOPEDIA FOR THE TRS-80*

A library of useful information
for your TRS-80

Business
Education
Games
Graphics
Hardware
Home Applications
Interface
Tutorial
Utility

VOLUME 6

ENCYCLOPEDIA for the TRS-80*

ENCYCLOPEDIA for the TRS-80*

VOLUME 6

wayne GREEN
PETERBOROUGH NH 03458

FIRST EDITION
FIRST PRINTING FEBRUARY 1982

Printed in the United States of America

Edited by Kate Comiskey and Katherine Putnam
Production: Margaret Baker, Gary Ciocci,
Linda Drew, Tom Villeneuve, Bob Villeneuve,
Sandra Dukette, Karen Stewart
Technical Assistance by Jake Commander
and Kenniston W. Lord Jr., CDP
Illustrations by Howard Happ

FOREWORD

The Biggest Difference

There are lots of arguments about which computer is the best. The answer to this question lies not in which hardware is best. That is really irrelevant, when you understand the field. The major value of any computer lies in the software and the information available for it. Hence this encyclopedia.

The TRS-80 is by no means the best computer on the market as far as its hardware is concerned, but with the support of *80 Microcomputing* magazine and this encyclopedia series, you have an almost unlimited source of information on how to use your computer—and of programs. With this information source the TRS-80 is by far the most valuable computer system ever built. No other computer, at any price, has anything approaching this amount of user information and programs available.

Most encyclopedias try to freeze everything at one time and are thus able to divide the material up alphabetically. This is a new kind of encyclopedia —a living one—with each new volume keeping you up to date on the very latest information on using your computer and the newest of programs.

Your computer can be a fantastic teaching device, a simulator, a way to play all sorts of fascinating games, a business aid, a scientific instrument, a control unit for machinery. . . . It is one of the most flexible gadgets ever invented. All of these applications are possible *if* you have the information and the programs. This encyclopedia will give you these.

To get the best use of your TRS-80, don't miss a single volume of the *Encyclopedia for the TRS-80.*

WAYNE GREEN
Publisher

CONTENTS

Please note: Before typing in any listing in this book, see Appendix A.

contents

BUSINESS

Exponential Smoothing
Voter Registration

Exponential Smoothing

by Leonard Gorney

The eventual success of a business is based, in part, on its management's ability to forecast accurately the future demand for a product. Forecasting is the science of predicting future events based in whole or in part on past performances. Many forecasting methods are available, running the gamut from educated guesses to highly mathematical techniques.

The technique known as exponential smoothing is a variation on a moving-average forecast. In particular, exponential smoothing is based on a geometric progression which results in nonuniform weights being assigned to the available historical data. Exponential smoothing and the moving-average techniques effectively reduce fluctuations in demand and, at the same time, remain sensitive to trends. In other words, the values calculated for future demands are not subject to instantaneous, random peaks or drops, nor is a resulting trend ignored before proper adjustments are made to the forecast values.

Requirements

The required variables for exponential smoothing include historical data which corresponds to previous actual demand figures, the number of time periods to project into the future, and a constant known as the smoothing constant. Historical data is usually obtained from sales figures for the product during a particularly representative time period. Monthly sales figures, for example, can be gathered and used as the basis for the historical data points.

A business which deals with a high-turnover product may want to forecast only one to three time periods into the future. On the other hand, a longer forecast period is logical for a product whose demand remains widespread and steady, such as home heating oil.

The major problem with exponential smoothing lies in choosing an appropriate smoothing value—the constant. The smoothing constant must be a positive value between 0.00 and 1.00; 0.00 results in complete smoothing, while a smoothing constant of 1.00 results in a forecast with no smoothing ability. These extreme values are rarely used.

A smoothing constant between 0.01 and 0.03 generally yields reasonably accurate results; that is, a prompt response to change without a larger response to random fluctuations. The smaller the smoothing constant value, the slower the response of the procedure to change. On the other hand, the larger the value of the smoothing constant, the quicker the response to change, as a much greater emphasis is placed on the most recent data.

A product's market history will usually give a clue as to the correct value of the smoothing constant. If the historical data is relatively stable, a small smoothing constant usually gives accurate forecasts. A seasonal pattern or other trend usually requires a large smoothing constant. Experience in your business is the prerequisite for forecasting future demand for a product.

One technique, known as retrospective simulation, is often used to determine the approximate values for the smoothing constant and the number of projected time periods. Retrospective simulation (or Monday morning quarterbacking) uses the most recent marketing figures as control data and applies them to historical market information. Various smoothing constants and projected time periods are applied to the historical data, until a reasonably good fit results. These same values are then applied to future demand predictions.

Operating Instructions

The initial response to the following program is either a Y or an N. Entering a Y will present the necessary background information and operating instructions for the program; entering N branches directly to the data input phase. The program will then ask for the number of time periods for which actual market demand information is available; that is, the number of time periods for which historical data is present. Entering a zero for this question ends the program, while entering any other positive integer value dimensions the necessary arrays.

An OM ERROR may occur if the value for this variable is too large. Using a 16K TRS-80, the maximum value for this variable may approach 250 with no adverse effects. If no error occurs, the program asks for each actual demand value. After all the actual demand values are entered, the program asks for the smoothing constant value. It expects a positive value between 0.00 and 1.00.

The number of projected time periods is the next request. It must be a positive integer, one or above. This value, as well as the value for the previously entered smoothing constant, determines the accuracy of the forecast. Finally, a Y or an N answer is required if you want a tabular output of the various forecast values during the execution of the program. These values are important if you want to see the intermediate results of the procedure.

The program then commences its calculations. If your answer to the tabular question is a Y, the program displays and freezes intermediate values during each time period for the actual demand, exponential average, current trend, smoothed trend, forecast demand, and forecast error. Eventually, the screen displays the final results of the program. These results include the standard deviation of forecast errors.

The number of times the actual demand was greater than and less than the forecast value is also listed. The cumulative forecast errors, the largest positive forecast error, the largest negative forecast error, and the average forecast error will also be listed. You can adjust the smoothing constant and/or the number of time periods projected into the future without reentering the historical data by following the instructions on the last screen display.

Program Listing. *Exponential Smoothing*

```
1000 REM    EXPONENTIAL SMOOTHING
1010 REM         BY: LEN GORNEY
1020 REM             BOX 91 R.D. 5 SALISBURY ROAD
1030 REM             CLARKS SUMMIT PA 18411
1040 REM    VARIABLE              DESCRIPTION
1050 REM     AD            ACTUAL DEMANDS (HISTORICAL DATA)
1060 REM     AF            AVERAGE FORECAST ERROR
1070 REM     BF            LARGEST FORECAST ERROR
1080 REM     CT            CURRENT TREND VALUES
1090 REM     DG            # OF TIMES DEMAND > FORECAST
1100 REM     DL            # OF TIMES DEMAND < FORECAST
1110 REM     EA            EXPONENTIAL AVERAGES
1120 REM      F            FORECAST VALUES
1130 REM     FE            FORECAST ERRORS
1140 REM     F2            FORECAST ERRORS SQUARED
1150 REM      I            GENERAL LOOP COUNTER
1160 REM     LF            SMALLEST FORECAST ERROR
1170 REM      N            # OF ACTUAL DEMAND TIME PERIODS
1180 REM     NC            # OF TIMES FORECAST >< ZERO
1190 REM     NL            # OF LINES PRINTED FOR TABULAR OUTPUT
1200 REM     NP            # OF PROJECTED TIME PERIODS
1210 REM      Q            ANSWER ( Y OR N ) TO QUESTION
1220 REM     QN            PRINT USING PARAMETER
1230 REM     Q1            PRINT USING PARAMETER
1240 REM      S            SUM OF FORECAST ERRORS
1250 REM     SC            SMOOTHING CONSTANT VALUE
1260 REM     SD            STANDARD DEVIATION OF FORECAST
1270 REM     SE            STANDARD DEVIATION OF FORECAST ERROR
1280 REM     ST            SMOOTHED TRENDS
1290 REM     S1            1.0 LESS THE SMOOTHING CONSTANT VALUE
1300 REM     S2            SUM OF THE FORECAST ERRORS SQUARED
1310 DEFINT D, I, N
1320 DEFSNG A, B, C, E, F, L, S
1330 DEFSTR Q
1340 QN = "#######.##":
     Q1 = "###"
1350 CLS
1360 INPUT "ENTER > Y < FOR INSTRUCTIONS ELSE ENTER > N <";Q
1370 IF Q = "Y"
      THEN
       2560 :
      ELSE
       IF Q = "N"
        THEN
         1380 :
        ELSE
         1350
1380 CLS
1390 INPUT "ENTER NUMBER OF TIME PERIODS OF ACTUAL DEMAND";N
1400 IF N < = 0 OR N > < INT(N)
      THEN
       STOP
1410 DIM AD(N), EA(N), CT(N), ST(N), F(N), FE(N), F2(N)
1420 CLS
1430 FOR I = 1 TO N
1440  PRINT ,"ACTUAL DEMAND #";I;
1450  INPUT AD(I)
1460  NEXT I
1470 CLS
1480 INPUT "ENTER SMOOTHING CONSTANT";SC
1490 IF SC < = 0.00 OR SC > 1.00
      THEN
       GOSUB 2430 :
       GOTO 1480
1500 CLS
1510 INPUT "ENTER NUMBER OF PROJECTED TIME PERIODS";NP
1520 IF NP < = 0 OR NP > < INT(NP)
      THEN
```

```
      GOSUB 2470:
      GOTO 1510
1530 CLS
1540 INPUT "ENTER > Y < FOR TABULAR OUTPUT ELSE ENTER > N <";Q
1550 IF Q = "Y"
      THEN
       GOSUB 2300 :
      ELSE
       IF Q = "N"
        THEN
         1570 :
        ELSE
         1530
1560 REM   INITIALIZE VARIABLES TO ZERO
1570 AF = 0.0:
     BF = 0.0:
     DG = 0.0:
     DL = 0.0:
     LF = 0.0:
     NC = 0
1580 S1 = 1.0 - SC
1590 FOR I = 1 TO N
1600  CT(I) = 0.0:
      EA(I) = 0.0:
      F(I) = 0.0
1610  FE(I) = 0.0:
      F2(I) = 0.0:
      ST(I) = 0.0
1620  NEXT I
1630 S = 0.0
1640 S2 = 0.0
1650 REM   SET FIRST EXPONENTIAL AVERAGE TO FIRST ACTUAL DEMAND
1660 NL = 1
1670 EA(1) = AD(1)
1680 I = 1
1690 IF Q = "Y"
      THEN
       GOSUB 2380
1700 REM   START OF MAIN LOOP
1710 FOR I = 2 TO N
1720  REM    CALCULATE EXPONENTIAL AVERAGE, CURRENT TREND
1730  REM    IF SMOOTHED TREND = ZERO
                 THEN SMOOTHED TREND = CURRENT TREND
                  ELSE CALCULATE SMOOTHED TREND
1740  EA(I) = (SC * AD(I)) + (S1 * EA(I - 1))
1750  CT(I) = EA(I) - EA(I - 1)
1760  IF ST(I - 1) = 0.0
       THEN
        ST(I) = CT(I) :
       ELSE
        ST(I) = (SC * CT(I)) + (S1 * ST(I - 1))
1770  IF NP + 1 > = I
       THEN
        1920
1780  REM    CALCULATE FORECAST VALUE
1790  F(I) = EA(I - NP) + ((S1 / SC + FIX(NP)) * ST(I - NP))
1800  REM    ACCUMULATE DEMAND > FORECAST OR DEMAND < FORECAST
1810  IF AD(I) > F(I)
       THEN
        DG = DG + 1 :
       ELSE
        DL = DL + 1
1820  REM    CALCULATE FORECAST ERROR VALUE
1830  FE(I) = F(I) - AD(I)
1840  REM    DETERMINE LARGEST FORECAST ERROR
                AND SMALLEST FORECAST ERROR
1850  IF FE(I) > BF
       THEN
        BF = FE(I)
1860  IF FE(I) < LF
       THEN
```

Program continued

```
      LF = FE(I)
1870  REM    CALCULATE FORECAST ERRORS SQUARED,
                        SUM OF FORECAST ERRORS,
                         SUM OF FORECAST ERRORS SQUARED
1880  F2(I) = FE(I) * FE(I)
1890  S = S + FE(I)
1900  S2 = S2 + F2(I)
1910  IF F(I) > < 0.0
       THEN
        NC = NC + 1
1920  IF Q = "Y"
       THEN
        GOSUB 2380
1930  IF NL = 13
       THEN
        GOSUB 2510:
        GOSUB 2300
1940  REM    END OF MAIN LOOP
1950  NEXT I
1960 REM    CALCULATE AVERAGE FORECAST ERROR,
      STANDARD DEVIATION OF FORECAST,
      STANDARD DEVIATION OF FORECAST ERRORS
1970 FOR I = 1 TO N
1980  AF = AF + FE(I)
1990  NEXT I
2000 AF = AF / N:
     IF NC = 0 OR NC = 1
      THEN
       SE = 0:
       SD = 0:
       GOTO 2030
2010 SE = SQR( ABS((S2 - (S * S) / (NC)) / (NC - 1)))
2020 SD = SQR( ABS(S2 / (NC - 1)))
2030 GOSUB 2510
2040 REM    OUTPUT RESULTS ROUTINE
2050 PRINT @64,"SMOOTHING CONSTANT =";SC
2060 PRINT @128,NP;"PROJECTED TIME PERIODS"
2070 PRINT @256,"STANDARD DEVIATION OF FORECAST DISCREPANCY =";
2080 PRINT ,SD
2090 PRINT @320,"STANDARD DEVIATION OF FORECAST ERROR        =";
2100 PRINT ,SE
2110 PRINT @448,"DEMAND GREATER THAN FORECAST";DG;"TIMES"
2120 PRINT @512,"DEMAND    LESS THAN FORECAST";DL;"TIMES"
2130 PRINT @640,"CUMULATIVE FORECAST ERRORS       =";S
2140 PRINT @704,"LARGEST POSITIVE FORECAST ERROR =";BF
2150 PRINT @768,"LARGEST NEGATIVE FORECAST ERROR =";LF
2160 PRINT @832,"AVERAGE FORECAST ERROR          =";AF
2170 GOSUB 2510
2180 CLS
2190 PRINT @912,"ENTER APPROPRIATE OPTION NUMBER"
2200 PRINT @320,"OPTION   DESCRIPTION"
2210 PRINT @389,"1  USE SAME ACTUAL DEMAND FIGURES BUT";
2220 PRINT " DIFFERENT"
2230 PRINT @457,"SMOOTHING CONSTANT AND/OR PROJECTED TIME";
2240 PRINT " PERIOD VALUES"
2250 PRINT @517,"2  USE DIFFERENT ACTUAL DEMAND FIGURES"
2260 PRINT @645,"3  STOP PROGRAM"
2270 PRINT @768,"ENTER OPTION NUMBER";
2280 INPUT I
2290 IF I = 1
      THEN
       1470 :
      ELSE
       IF I = 2
        THEN
         RUN 1310 :
        ELSE
         STOP
2300 REM    TABULAR OUTPUT ROUTINE
2310 CLS
2320 NL = 1
```

```
2330 PRINT @65,"TIME  ACTUAL EXPONENTIAL CURRENT  SMOOTHED";
2340 PRINT @119,"FORECAST";
2350 PRINT @128,"PERIOD DEMAND   AVERAGE    TREND    TREND";
2360 PRINT @173,"FORECAST    ERROR"
2370 RETURN
2380 NL = NL + 1
2390 PRINT USING Q1;I;
2400 PRINT USING QN;AD(I);EA(I);CT(I);ST(I);F(I);FE(I)
2410 RETURN
2420 REM    ERROR MESSAGE ROUTINES
2430 CLS
2440 PRINT "0.01 <= SMOOTTHING CONSTANT <= 1.00"
2450 GOSUB 2510
2460 RETURN
2470 CLS
2480 PRINT "NUMBER OF PROJECTED TIME PERIODS >= 1"
2490 GOSUB 2510
2500 RETURN
2510 PRINT @960,"PRESS > ENTER < TO CONTINUE";
2520 INPUT QE
2530 CLS
2540 RETURN
2550 REM    INSTRUCTION OUTPUTT ROUTINE
2560 CLS
2570 PRINT @10,"FORECASTING BY EXPONENTIAL SMOOTHING""
2580 PRINT
2590 PRINT "   EXPONENTIAL SMOOTHING IS A VARIATION ON A MOVING"
2600 PRINT "AVERAGE FORECAST BASED ON A GEOMETRIC PROGRESSION "
2610 PRINT "RESULTING IN THE ASSIGNMENT OF NONUNIFORM WEIGHTS "
2620 PRINT "TO THE HISTORICAL DATA."
2630 PRINT "   THE REQUIRED INPUT PARAMETERS INCLUDE:"
2640 PRINT "1. THE HISTORICAL DATA; I.E. PAST ACTUAL DEMAND,"
2650 PRINT "2. A SMOOTHING CONSTANT,"
2660 PRINT "3. THE NUMBER OF PROJECTED TIME PERIODS."
2670 PRINT "   THIS PROGRAM USES A TYPE OF RETROSPECTIVE "
2680 PRINT "SIMULATION TO EXHIBIT THE EXPONENTIAL SMOOTHING "
2690 PRINT "FORECAST TECHNIQUE."
2700 GOSUB 2510
2710 PRINT "IINPUT PARAMETER # 1"
2720 PRINT "   NUMBER OF ACTUAL DEMAND PERIODS; I.E. THE"
2730 PRINT "   NUMBER OF PERIODS FOR WHICH HISTORICAL DATA "
2740 PRINT "   IS AVAILABLE."
2750 PRINT "INPUT PARAMETER " 2"
2760 PRINT "   THE VALUES FOR EACH OF THE ACTUAL DEMAND "
2770 PRINT "   FIGURES."
2780 PRINT "INPUT PARAMETER # 3"
2790 PRINT "   SMOOTHING CONSTANT VALUE.  THIS VALUE MUST BE "
2800 PRINT "   BETWEEN +0.01 AND +1.00"
2810 PRINT "INPUT PARAMETER # 4"
2820 PRINT "   NUMBER OF PROJECTED TIME PERIODS IS THE "
2830 PRINT "   FORECASTED FUTURE TIME PERIODS."
2840 GOSUB 2510
2850 PRINT "INPUT PARAMETER # 5"
2860 PRINT "   INTERMEDIATE OUTPUT IS AVAILABLE BY ANSWERING "
2870 PRINT "   'Y' TO THIS QUESTION."
2880 GOSUB 2510
2890 GOTO 1380
2900 END
```

BUSINESS

Voter Registration

by Kenniston W. Lord Jr., CDP

The small municipality, like its larger counterparts, is responsible for the registration of voters. This is usually the duty of the town clerk. When a voter is registered, data is collected, such as the voter's address as of the beginning of the current year, the address as of the beginning of last year, occupation, location of employment, party of preference, citizenship information, and whether or not the individual must register through the use of an informant (generally in the case of a person who is unable to speak English). Some states even gather data on the registrant's dogs.

This program (see Program Listing) is a data collection package for the town clerk's registration of voters. It fits in a 32K system; disk is not required. The program is well commented; so if you must compress for a 16K system, you can use a compression utility to remove REMark statements. The program is set up to accept one day's business, but you can expand it by changing the size of the DIMensioned statements in line 160 and by making corresponding changes to the routines which manipulate the arrays throughout the program.

When the program begins, you must gather some general information.

- ENTER YEAR—The range test is set up to cover the period from 1970–1999. It would be advisable to narrow that period by making the appropriate changes in line 220.
- WILL REGISTRATIONS BE LISTED BY ONE PERSON (Y/N)?—The response is usually Y, after which the program asks the name of the person and inserts that person's name into each record. If several persons are doing the registrations, an N answer structures the program to request the name of the person gathering the data with each individual registration.
- ENTER TODAY'S DATE WITH NO PUNCTUATION—The limitations of punctuation require care here. It would have been possible to use LINE-INPUT, but that would have kept anyone without a disk system from using the program. This date will be used on reports; so enter it as you would like it to appear. You *can* enter the punctuation if you are careful to enclose the entire expression in quotation marks. This entry, like all entries in this program, is presented for verification and is accepted only when the user verifies the entry.

Once this information is gathered, the main menu appears.

```
PROCESSING OPTIONS:
        A. REGISTER A MALE
        B. REGISTER A FEMALE
        C. DISPLAY REGISTRATIONS
        D. PRINT REGISTRATONS
        E. END THE PROGRAM
```

While each of these appears to be straightforward, I will explain each of them separately. Concentrate first on option E. This terminates the program by offering:

DO YOU WISH TO PREPARE A TRANSACTION TAPE (Y/N)

If you answer Y, you are asked to set up the tape recorder to record. The program simply writes a data tape with all registrants for the period. To produce periodic reports, you must input this tape to a transaction-combining program. Since the records are fixed in length (20 fields), reading the tape or tapes and combining the data is a simple matter. After the tape is written, or if you specify N, the program ends. If you have used a tape, the program generates the appropriate messages.

Options A and B are similar in function. Each requests the same data, but the program keeps separate arrays for males and females to facilitate reporting. If you request a combined report, those arrays are combined and sorted on request. Each request for information must be corroborated by the user before proceeding. If there is still an error after you have made all the entries, the program provides a reentry capability. In sequence, options A and B request the following data:

NAME
AGE
PARTY AFFILIATION
(Program allows Democrat, Republican, Independent, and another category which allows specification.)
WARD OR PRECINCT
RESIDENCE THIS YEAR
RESIDENCE LAST YEAR
OCCUPATION
WHERE EMPLOYED
CITIZENSHIP
(If not United States, then origin is requested.)
INFORMANT
(If necessary)
LISTED BY
(If not entered at the beginning)
DOGS
(If the voter has dogs, distribution by sex. Females are taxed higher.)

Options C and D are similar in that one displays the registrants on the screen and the other prints it. The menu options appear as follows:

```
DISPLAY/PRINT OPTIONS:
<F>EMALE REGISTRANTS
<M>ALE REGISTRANTS
<A>LL REGISTRANTS
```

Once you select an option, the program asks

```
SORT THE DATA (Y/N)?
```

These options give you many opportunities to sort the data. Once the sort is complete, you cannot resort; so you should reserve sorting until you have run the final daily reports. For a half dozen registrants, this does not require much time, but the memory sort in BASIC is slow, and a single sort is preferable. A sample of output is shown in Figure 1.

All selected options, once completed, return you to the main menu. This program does not disable the BREAK key, but if that is anticipated to be a problem, you may find it necessary to disable that key.

NAME: JOHN DOE	AGE: 22 MALE	
VOTER NO.: 111	WARD/PCT NO.: 2	
RES.(LAST YR):	123 MAIN STREET	
RES.(THIS YR):	456 ELM STREET	
OCCUPATION:	FACTORY WORKER	
WHERE EMPLOYED:	JONES BROTHERS MILL	
CITIZEN:	Y	
NATIONALITY:	U.S.A.	
DATE OF REGISTRATION:	JANUARY 1 1982	
INFORMANT:	HIMSELF	
LISTED BY:	JANE SMITH	
PARTY AFFILIATION:	DEMOCRAT	
DOGS (NUMBER): 2	MALE: 1	FEMALE: 1

Figure 1. *Sample output*

Program Listing. *Voter registration*

Encyclopedia Loader

```
140 :
    ' *
150 CLEAR 2000
160 DIM VM$(220):
    DIM VF$(220):
    DIM VT$(20):
    DIM AV$(440)
170 M = 1:
    F = 1
180 GOSUB 5330
190 FOR N = 1 TO 4:
     PRINT :
     NEXT N
200 PRINT "  VOTER REGISTRATION PROGRAM":
    PRINT
210 PRINT "PLEASE WAIT - DOING HOUSEWORK"
220 FY = 1970:
    LY = 1999:
    LS = 0:
    M = 1:
    F = 1:
    GOSUB 5380:
    CLS
230 SM = 1:
    SF = 1
240 GOSUB 5330
250 FOR N = 1 TO 4:
     PRINT :
     NEXT N
260 PRINT TAB(5);"REGISTRAR OF VOTERS":
    PRINT
270 PRINT TAB(5);"TOWN OF XXXXXXXXXXX"
280 PRINT :
    PRINT SB$
290 Z$ = INKEY$
300 IF Z$ = " "
     THEN
      320
310 GOTO 290
320 GOSUB 5330:
    PRINT "ENTER YEAR (4-NR. FORMAT)"
330 PRINT :
    PRINT "EXAMPLE: 1982"
340 PRINT :
    INPUT YR
350 IF (YR < FY) OR (YR > LY)
     THEN
      410
360 PRINT :
    PRINT "YEAR ENTERED IS ACCEPTABLE"
370 PRINT :
    PRINT SB$
380 Z$ = INKEY$
390 IF Z$ = " "
     THEN
      460
400 GOTO 380
410 PRINT "YEAR OUTSIDE RANGE OF ";FY;"TO ";LY
420 PRINT "RE-ENTER":
    PRINT SB$
430 Z$ = INKEY$
440 IF Z$ = " "
     THEN
      320
450 GOTO 430
460 GOSUB 5330
470 PRINT "WILL REGISTRATIONS BE":
```

Program continued

```
    PRINT
480 PRINT "LISTED BY ONE PERSON (Y/N)?"
490 Z$ = INKEY$
500 IF Z$ = "Y"
     THEN
      530
510 IF Z$ = "N"
     THEN
      LS = 1:
      GOTO 600
520 GOTO 490
530 PRINT :
    PRINT "WHO IS THAT PERSON?"
540 INPUT NV$
550 PRINT "CONFIRM: ";NV$;" (Y/N)"
560 Z$ = INKEY$
570 IF Z$ = "Y"
     THEN
      600
580 IF Z$ = "N"
     THEN
      460
590 GOTO 560
600 GOSUB 5330
610 PRINT "ENTER TODAY'S DATE"
620 PRINT :
    PRINT "WITH NO PUNCTUATION"
630 PRINT :
    PRINT "EXAMPLE: JANUARY 1 1982":
    PRINT
640 INPUT DT$:
    PRINT
650 PRINT "CONFIRM ";DT$;" (Y/N)"
660 Z$ = INKEY$
670 IF Z$ = "Y"
     THEN
      710
680 IF Z$ = "N"
     THEN
      600
690 GOTO 660
700 GOSUB 5330
710 CLS :
    GOSUB 5330:
    PRINT EN$;VN$;"TO BE USED"
720 PRINT :
    INPUT VN:
    VN = VN - 1
730 PRINT :
    GOSUB 5350
740 Z$ = INKEY$
750 IF Z$ = "Y"
     THEN
      780
760 IF Z$ = "N"
     THEN
      700
770 GOTO 740
780 GOSUB 5330:
    PRINT :
    PRINT :
    PRINT "PROCESSING OPTIONS":
    PRINT
790 GOSUB 7830
800 PRINT TAB(2);"A.  REGISTER A MALE":
    PRINT
810 PRINT TAB(2);"B.  REGISTER A FEMALE":
    PRINT
820 PRINT TAB(2);"C.  DISPLAY REGISTRATIONS":
    PRINT
830 PRINT TAB(2);"D.  PRINT REGISTRATIONS":
```

```
    PRINT
840 PRINT TAB(2);"E.  END THE PROGRAM"
850 Z$ = INKEY$
860 IF Z$ < > ""
     THEN
      890
870 IF Z$ = " "
     THEN
      850
880 GOTO 850
890 A = ASC(Z$) - 64
900 IF A < 1
     THEN
      850
910 IF A > 5
     THEN
      850
920 ON A GOTO 930,990,5980,6770,7420:
    GOTO 780
930 GOSUB 5330
940 PRINT "MALE REGISTRATION (Y/N)?"
950 Z$ = INKEY$
960 IF Z$ = "N"
     THEN
      780
970 IF Z$ = "Y"
     THEN
      S$ = "M":
      GOTO 1050
980 GOTO 950
990 GOSUB 5330
1000 PRINT "FEMALE REGISTRATION (Y/N)?"
1010 Z$ = INKEY$
1020 IF Z$ = "N"
      THEN
       780
1030 IF Z$ = "Y"
      THEN
       S$ = "F":
       GOTO 1050
1040 GOTO 1010
1050 GOSUB 5330
1060 :
     '
1070 :
     ' *****************
1080 :
     ' ** VOTER NAME  **
1090 :
     ' *****************
1100 :
     '
1110 PRINT EN$;"NAME OF VOTER"
1120 INPUT VT$(1)
1130 GOSUB 5350
1140 Z$ = INKEY$
1150 IF Z$ = "N"
      THEN
       1050
1160 IF Z$ = "Y"
      THEN
       1230
1170 GOTO 1140
1180 :
     '
1190 :
     ' *****************
1200 :
     ' *  CAPTURE AGE  *
1210 :
     ' *****************
```

Program continued

```
1220 :
     '
1230 GOSUB 5330
1240 PRINT EN$;AG$
1250 INPUT VT$(2)
1260 GOSUB 5350
1270 Z$ = INKEY$
1280 IF Z$ = "N"
      THEN
       1230
1290 IF Z$ = "Y"
      THEN
       1310
1300 GOTO 1270
1310 GOSUB 5330
1320 :
     '
1330 :
     ' ************************
1340 :
     ' **  PARTY AFFILIATION  **
1350 :
     ' ************************
1360 :
     '
1370 PRINT EN$;PA$:
     PRINT
1380 PRINT TAB(5);"D.  ";PD$
1390 PRINT TAB(5);"R.  ";PR$
1400 PRINT TAB(5);"I.  ";PI$
1410 PRINT TAB(5);"O.  ";PO$
1420 PW$ = INKEY$
1430 IF PW$ = "D"
      THEN
       PW$ = PD$:
       GOTO 1480
1440 IF PW$ = "R"
      THEN
       PW$ = PR$:
       GOTO 1480
1450 IF PW$ = "I"
      THEN
       PW$ = PI$:
       GOTO 1480
1460 IF PW$ = "O"
      THEN
       PRINT EN$;PA$:
       INPUT PW$:
       GOTO 1480
1470 GOTO 1420
1480 GOSUB 5330:
     PRINT PA$;" IS ":
     PRINT PW$
1490 VT$(19) = PW$
1500 VN = VN + 1
1510 VT$(20) = STR$(VN)
1520 PRINT :
     GOSUB 5350
1530 Z$ = INKEY$
1540 IF Z$ = "Y"
      THEN
       1570
1550 IF Z$ = "N"
      THEN
       1310
1560 GOTO 1530
1570 GOSUB 5330
1580 :
     '
1590 :
     ' ***********************
```

```
1600 :
     ' **  WARD OR PRECINCT   **
1610 :
     ' ************************
1620 :
     '
1630 PRINT EN$;PC$:
     PRINT
1640 INPUT PC:
     VT$(15) = STR$(PC)
1650 GOSUB 5350
1660 Z$ = INKEY$
1670 IF Z$ = "Y"
      THEN
       1700
1680 IF Z$ = "N"
      THEN
       1570
1690 GOTO 1660
1700 GOSUB 5330
1710 :
     '
1720 :
     ' ***********************************
1730 :
     ' **  CAPTURE RESIDENCE THIS YEAR   **
1740 :
     ' ***********************************
1750 :
     '
1760 PRINT EN$;RA$
1770 PRINT J1$;TY$
1780 INPUT VT$(3)
1790 GOSUB 5350
1800 Z$ = INKEY$
1810 IF Z$ = "N"
      THEN
       1310
1820 IF Z$ = "Y"
      THEN
       1840
1830 GOTO 1800
1840 GOSUB 5330
1850 :
     '
1860 :
     ' ***********************************
1870 :
     ' **  CAPTURE RESIDENCE LAST YEAR   **
1880 :
     ' ***********************************
1890 :
     '
1900 PRINT EN$;RA$
1910 PRINT J1$;LY$
1920 INPUT VT$(4)
1930 GOSUB 5350
1940 Z$ = INKEY$
1950 IF Z$ = "N"
      THEN
       1840
1960 IF Z$ = "Y"
      THEN
       1980
1970 GOTO 1940
1980 GOSUB 5330
1990 :
     '
2000 :
     ' **************************
2010 :
```

Program continued

```
     ' **  CAPTURE OCCUPATION  **
2020 :
     ' ***************************
2030 :
     '
2040 PRINT EN$;OC$
2050 INPUT VT$(5)
2060 GOSUB 5350
2070 Z$ = INKEY$
2080 IF Z$ = "N"
      THEN
       1980
2090 IF Z$ = "Y"
      THEN
       2110
2100 GOTO 2070
2110 GOSUB 5330
2120 :
     '
2130 :
     ' *******************************
2140 :
     ' **  CAPTURE WHERE EMPLOYED  **
2150 :
     ' *******************************
2160 :
     '
2170 PRINT EN$;WE$
2180 INPUT VT$(6)
2190 GOSUB 5350
2200 Z$ = INKEY$
2210 IF Z$ = "N"
      THEN
       2110
2220 IF Z$ = "Y"
      THEN
       2240
2230 GOTO 2200
2240 GOSUB 5330
2250 :
     '
2260 :
     ' ***************************
2270 :
     ' **  CAPTURE CITIZENSHIP  **
2280 :
     ' ***************************
2290 :
     '
2300 PRINT CZ$
2310 Z$ = INKEY$
2320 IF Z$ = "N"
      THEN
       VT$(7) = "N":
       GOTO 2420
2330 IF Z$ = "Y"
      THEN
       VT$(7) = "Y":
       GOTO 2350
2340 GOTO 2310
2350 VT$(8) = "U.S.A."
2360 GOTO 2540
2370 :
     '
2380 :
     ' ***************************
2390 :
     ' **  CAPTURE NATIONALITY  **
2400 :
     ' ***************************
2410 :
```

```
2420 PRINT EN$;NA$
2430 INPUT VT$(8)
2440 GOSUB 5350
2450 Z$ = INKEY$
2460 IF Z$ = "N"
      THEN
       GOSUB 5330:
       GOTO 2420
2470 IF Z$ = "Y"
      THEN
       2540
2480 GOTO 2450
2490 :
     '
2500 :
     ' *********************
2510 :
     ' **  STORE THE DATE  **
2520 :
     ' *********************
2530 :
     '
2540 VT$(9) = DT$
2550 GOSUB 5330
2560 :
     '
2570 :
     ' ************************
2580 :
     ' **  CAPTURE INFORMANT  **
2590 :
     ' ************************
2600 :
     '
2610 PRINT EN$;IN$
2620 INPUT VT$(10)
2630 GOSUB 5350
2640 Z$ = INKEY$
2650 IF Z$ = "N"
      THEN
       2550
2660 IF Z$ = "Y"
      THEN
       2680
2670 GOTO 2640
2680 GOSUB 5330
2690 :
     '
2700 :
     ' *************************
2710 :
     ' **  LISTED BY STANDARD  **
2720 :
     ' *************************
2730 :
     '
2740 IF LS < > 1
      THEN
       VT$(11) = NV$:
       GOTO 2870
2750 :
     '
2760 :
     ' **********************
2770 :
     ' **  LISTED BY ENTRY  **
2780 :
     ' **********************
2790 :
     '
```

Program continued

```
2800 PRINT LB$
2810 INPUT VT$(11)
2820 GOSUB 5350
2830 Z$ = INKEY$
2840 IF Z$ = "N"
      THEN
       2680
2850 IF Z$ = "Y"
      THEN
       2870
2860 GOTO 2830
2870 GOSUB 5330
2880 :
     '
2890 :
     ' *******************
2900 :
     ' **  CAPTURE DOGS  **
2910 :
     ' *******************
2920 :
     '
2930 PRINT "DOGS (Y/N)?"
2940 Z$ = INKEY$
2950 IF Z$ = "N"
      THEN
       2980
2960 IF Z$ = "Y"
      THEN
       3000
2970 GOTO 2940
2980 VT$(12) = "0":
     VT$(13) = "0":
     VT$(14) = "0"
2990 GOTO 3150
3000 GOSUB 5330
3010 PRINT "HOW MANY DOGS?"
3020 INPUT VT$(12)
3030 GOSUB 5350
3040 Z$ = INKEY$
3050 IF Z$ = "Y"
      THEN
       3080
3060 IF Z$ = "N"
      THEN
       3000
3070 GOTO 3040
3080 PRINT "HOW MANY MALE DOGS?"
3090 INPUT VT$(13)
3100 PRINT "HOW MANY FEMALE DOGS?"
3110 INPUT VT$(14)
3120 IF VAL(VT$(12)) = VAL(VT$(13)) + VAL(VT$(14))
      THEN
       3150
3130 PRINT "COUNT DOES NOT BALANCE"
3140 FOR Z = 1 TO 1000:
      NEXT Z:
     GOTO 3000
3150 VT$(15) = STR$(PC)
3160 IF S$ = "M"
      THEN
       VT$(18) = "MALE":
       GOTO 3450
3170 IF S$ = "F"
      THEN
       VT$(18) = "FEMALE":
       GOTO 3770
3180 PRINT "ERROR CONDITION":
     PRINT "RE-ENTER"
3190 FOR Z = 1 TO 1000:
```

```
     NEXT Z:
    GOTO 1050
3200 :
    '
3210 :
    ' ********************************************
3220 :
    ' *   VT$(1)     =     NAME                  *
3230 :
    ' *   VT$(2)     =     AGE                   *
3240 :
    ' *   VT$(3)     =     RESIDENCE THIS YEAR   *
3250 :
    ' *   VT$(4)     =     RESIDENCE LAST YEAR   *
3260 :
    ' *   VT$(5)     =     OCCUPATION            *
3270 :
    ' *   VT$(6)     =     WHERE EMPLOYED        *
3280 :
    ' *   VT$(7)     =     CITIZENSHIP           *
3290 :
    ' *   VT$(8)     =     NATIONALITY           *
3300 :
    ' *   VT$(9)     =     DATE OF REGISTRATION  *
3310 :
    ' *   VT$(10)    =     INFORMANT             *
3320 :
    ' *   VT$(11)   =      LISTED BY             *
3330 :
    ' *   VT$(12)   =      TOTAL DOGS            *
3340 :
    ' *   VT$(13)   =      MALE DOGS             *
3350 :
    ' *   VT$(14)   =      FEMALE DOGS           *
3360 :
    ' *   VT$(15)   =      PRECINCT/WARD         *
3370 :
    ' *   VT$(16)   =      AVAILABLE             *
3380 :
    ' *   VT$(17)   =      AVAILABLE             *
3390 :
    ' *   VT$(18)   =      SEX                   *
3400 :
    ' *   VT$(19)   =      PARTY AFFILIATION     *
3410 :
    ' *   VT$(20)   =      VOTER NUMBER          *
3420 :
    ' ********************************************
3430 :
    '
3440 :
    ' **********************
3450 :
    ' **  LOAD MALE ARRAY  **
3460 :
    ' **********************
3470 NM = NM + 1
3480 VM$(M) = VT$(1):
    VM$(M + 1) = VT$(2)
3490 VM$(M + 2) = VT$(3):
    VM$(M + 3) = VT$(4)
3500 VM$(M + 4) = VT$(5):
    VM$(M + 5) = VT$(6)
3510 VM$(M + 6) = VT$(7):
    VM$(M + 7) = VT$(8)
3520 VM$(M + 8) = VT$(9):
    VM$(M + 9) = VT$(10)
3530 VM$(M + 10) = VT$(11):
    VM$(M + 11) = VT$(12)
3540 VM$(M + 12) = VT$(13):
```

Program continued

```
     VM$(M + 13) = VT$(14)
3550 VM$(M + 14) = STR$(PC)
3560 VM$(M + 15) = VT$(16)
3570 VM$(M + 16) = VT$(17)
3580 VM$(M + 17) = VT$(18)
3590 VM$(M + 18) = VT$(19)
3600 VM$(M + 19) = VT$(20)
3610 VM$ (M + 20) = "ZZZZZ"
3620 GOSUB 4420:
     PRINT @921,"CORRECT (Y/N)?";
3630 FOR X = 1 TO 127:
      SET (X,45):
      SET (X,40):
      NEXT X
3640 Z$ = INKEY$
3650 IF Z$ = "N"
      THEN
       VN = VN - 1:
       GOTO 3680
3660 IF Z$ = "Y"
      THEN
       M = M + 20:
       GOTO 780
3670 GOTO 3640
3680 FOR Z = M TO M + 19
3690  VM$(Z) = " "
3700  NEXT Z
3710 IF VN < 1
      THEN
       VN = 1
3720 GOTO 1050
3730 :
     '
3740 :
     ' *************************
3750 :
     ' **  LOAD FEMALE ARRAY   **
3760 :
     ' *************************
3770 :
     '
3780 NF = NF + 1
3790 VF$(F) = VT$(1):
     VF$(F + 1) = VT$(2)
3800 VF$(F + 2) = VT$(3):
     VF$(F + 3) = VT$(4)
3810 VF$(F + 4) = VT$(5):
     VF$(F + 5) = VT$(6)
3820 VF$(F + 6) = VT$(7):
     VF$(F + 7) = VT$(8)
3830 VF$(F + 8) = VT$(9):
     VF$(F + 9) = VT$(10)
3840 VF$(F + 10) = VT$(11):
     VF$(F + 11) = VT$(12)
3850 VF$(F + 12) = VT$(13):
     VF$(F + 13) = VT$(14)
3860 VF$(F + 14) = STR$(PC)
3870 VF$(F + 15) = VT$(16)
3880 VF$(F + 16) = VT$(17)
3890 VF$(F + 17) = VT$(18)
3900 VF$(F + 18) = VT$(19)
3910 VF$(F + 19) = VT$(20)
3920 VF$ (F + 20) = "ZZZZZ"
3930 GOSUB 4880:
     PRINT :
     PRINT @921,"CORRECT (Y/N)?";
3940 FOR X = 1 TO 127:
      SET (X,45):
      SET (X,40):
      NEXT X
```

```
3950 Z$ = INKEY$
3960 IF Z$ = "N"
      THEN
       VN = VN - 1:
       GOTO 3990
3970 IF Z$ = "Y"
      THEN
       F = F + 20:
       GOTO 780
3980 GOTO 3950
3990 FOR Z = F TO F + 19
4000  VF$(Z) = " "
4010  NEXT Z
4020 IF VN < 1
      THEN
       VN = 1
4030 GOTO 1050
4040 :
     '
4050 :
     ' *******************
4060 :
     ' *  REVIEW ROUTINE  *
4070 :
     ' *******************
4080 :
     '
4090 SM = M:
     SF = F
4100 M = 1:
     F = 1
4110 GOSUB 5330
4120 PRINT SB$:
     PRINT "TO PAGE FORWARD"
4130 Z$ = INKEY$
4140 IF Z$ = " "
      THEN
       4160
4150 GOTO 4130
4160 IF NF = 0
      THEN
       4240
4170 FOR N = 1 TO NF
4180  GOSUB 4880
4190  F = F + 19
4200  Z$ = INKEY$
4210  IF Z$ = " "
       THEN
        4230
4220  GOTO 4200
4230  NEXT N
4240 IF NM = 0
      THEN
       4320
4250 FOR N = 1 TO NM
4260  GOSUB 4420
4270  M = M + 19
4280  Z$ = INKEY$
4290  IF Z$ = " "
       THEN
        4310
4300  GOTO 4280
4310  NEXT N
4320 IF NF = 0 PRINT "NO FEMALES"
4330 IF NM = 0 PRINT "NO MALES"
4340 FOR Z = 1 TO 1000:
      NEXT Z
4350 M = SM:
     F = SF
4360 GOTO 780
```

Program continued

```
4370 :
     '
4380 :
     ' ******************
4390 :
     ' *  DISPLAY MALES  *
4400 :
     ' ******************
4410 :
     '
4420 CLS
4430 PRINT "NAME: ";VM$(M); TAB(30);"AGE: ";VM$(M + 1); TAB(40);VM$(M
      + 17)
4440 PRINT VN$;": ";VM$(M + 19);
4450 PRINT TAB(30);PC$;": ";VM$(M + 14)
4460 PRINT "RES.(LAST YR): "; TAB(30);VM$(M + 2)
4470 PRINT "RES.(THIS YR): "; TAB(30);VM$(M + 3)
4480 PRINT OC$;": "; TAB(30);VM$(M + 4)
4490 PRINT WE$;": "; TAB(30);VM$(M + 5)
4500 PRINT "CITIZEN: ";VM$(M + 6);
4510 PRINT TAB(30);NA$;": ";VM$(M + 7)
4520 PRINT DR$;": "; TAB(30);VM$(M + 8)
4530 PRINT IN$;": "; TAB(30);VM$(M + 9)
4540 PRINT LB$;": "; TAB(30);VM$(M + 10)
4550 PRINT PA$;": "; TAB(30);VM$(M + 18)
4560 PRINT DG$;": "; TAB(15);VM$(M + 11);
4570 PRINT TAB(30);DM$;": "; TAB(40);VM$(M + 12);
4580 PRINT TAB(50);DF$;": "; TAB(60);VM$(M + 13)
4590 RETURN
4600 :
     '
4610 :
     ' ****************
4620 :
     ' *  PRINT MALES  *
4630 :
     ' ****************
4640 :
     '
4650 LPRINT "NAME: ";VM$(M); TAB(30);"AGE: ";VM$(M + 1); TAB(40);VM$(
     M + 17)
4660 LPRINT VN$;": ";VM$(M + 19);
4670 LPRINT TAB(30);PC$;": ";VM$(M + 14)
4680 LPRINT "RES.(LAST YR): "; TAB(30);VM$(M + 2)
4690 LPRINT "RES.(THIS YR): "; TAB(30);VM$(M + 3)
4700 LPRINT OC$;": "; TAB(30);VM$(M + 4)
4710 LPRINT WE$;": "; TAB(30);VM$(M + 5)
4720 LPRINT "CITIZEN: "; TAB(30);VM$(M + 6)
4730 LPRINT NA$;": "; TAB(30);VM$(M + 7)
4740 LPRINT DR$;": "; TAB(30);VM$(M + 8)
4750 LPRINT IN$;": "; TAB(30);VM$(M + 9)
4760 LPRINT LB$;": "; TAB(30);VM$(M + 10)
4770 LPRINT PA$;": "; TAB(30);VM$(M + 18)
4780 LPRINT DG$;": "; TAB(10);VM$(M + 11);
4790 LPRINT TAB(30);DM$;": "; TAB(40);VM$(M + 12);
4800 LPRINT TAB(50);DF$;": "; TAB(60);VM$(M + 13)
4810 LPRINT " ":
     LPRINT " "
4820 RETURN
4830 :
     '
4840 :
     ' ********************
4850 :
     ' *  DISPLAY FEMALES  *
4860 :
     ' ********************
4870 :
     '
4880 CLS
```

```
4890 PRINT "NAME: ";VF$(F); TAB(30);;"AGE: ";VF$(F + 1); TAB(40);VF$(
     F + 17)
4900 PRINT VN$;": ";VF$(F + 19);
4910 PRINT TAB(30);PC$;": ";VF$(F + 14)
4920 PRINT "RES.(LAST YR): "; TAB(30);VF$(F + 2)
4930 PRINT "RES.(THIS YR): "; TAB(30);VF$(F + 3)
4940 PRINT OC$;": "; TAB(30);VF$(F + 4)
4950 PRINT WE$;": "; TAB(30);VF$(F + 5)
4960 PRINT "CITIZEN: ";VF$(F + 6);
4970 PRINT TAB(30);NA$;": ";VF$(F + 7)
4980 PRINT DR$;": "; TAB(30);VF$(F + 8)
4990 PRINT IN$;": "; TAB(30);VF$(F + 9)
5000 PRINT LB$;": "; TAB(30);VF$(F + 10)
5010 PRINT PA$;": "; TAB(30);VF$(F + 18)
5020 PRINT DG$;": "; TAB(15);VF$(F + 11);
5030 PRINT TAB(30);DM$;": "; TAB(40);VF$(F + 12);
5040 PRINT TAB(50);DF$;": "; TAB(60);VF$(F + 13)
5050 RETURN
5060 :
     '
5070 :
     ' ******************
5080 :
     ' *   PRINT FEMALES  *
5090 :
     ' ******************
5100 :
     '
5110 LPRINT "NAME: ";VF$(F); TAB(30);;"AGE: ";VF$(F + 1); TAB(40);VF$
     (F + 17)
5120 LPRINT VN$;": ";VF$(F + 19);
5130 LPRINT TAB(30);PC$;": ";VF$(F + 14)
5140 LPRINT "RES.(LAST YR): "; TAB(30);VF$(F + 2)
5150 LPRINT "RES.(THIS YR): "; TAB(30);VF$(F + 3)
5160 LPRINT OC$;": "; TAB(30);VF$(F + 4)
5170 LPRINT WE$;": "; TAB(30);VF$(F + 5)
5180 LPRINT "CITIZEN: "; TAB(30);VF$(F + 6)
5190 LPRINT NA$;": "; TAB(30);VF$(F + 7)
5200 LPRINT DR$;": "; TAB(30);VF$(F + 8)
5210 LPRINT IN$;": "; TAB(30);VF$(F + 9)
5220 LPRINT LB$;": "; TAB(30);VF$(F + 10)
5230 LPRINT PA$;": "; TAB(30);VF$(F + 18)
5240 LPRINT DG$;": "; TAB(10);VF$(F + 11);
5250 LPRINT TAB(30);DM$;": "; TAB(40);VF$(F + 12);
5260 LPRINT TAB(50);DF$;": "; TAB(60);VF$(F + 13)
5270 LPRINT " ":
     LPRINT " "
5280 RETURN
5290 :
     '
5300 :
     ' ************************************
5310 :
     ' ** CLEAR SCREEN AND SHIFT PRINT SIZE **
5320 :
     '
5330 CLS :
     PRINT CHR$(23):
     RETURN
5340 :
     ' ***************************************************
5350 PRINT "CORRECT (Y/N)?":
     RETURN
5360 :
     ' ***************************************************
5370 :
     '
5380 EN$ = "ENTER "
5390 AG$ = "AGE "
5400 RA$ = "RESIDENCE ADDRESS AS OF "
```

Program continued

```
5410 J1$ = "JANUARY 1, "
5420 TY$ = "THIS YEAR (NO COMMAS) "
5430 LY$ = "LAST YEAR (NO COMMAS) "
5440 OC$ = "OCCUPATION "
5450 WE$ = "WHERE EMPLOYED "
5460 CZ$ = "CITIZEN OF U.S. (Y/N)?"
5470 NA$ = "NATIONALITY "
5480 DR$ = "DATE OF REGISTRATION "
5490 IN$ = "INFORMANT "
5500 LB$ = "LISTED BY "
5510 DG$ = "DOGS (NUMBER) "
5520 DM$ = "MALE "
5530 DF$ = "FEMALE "
5540 PA$ = "PARTY AFFILIATION"
5550 PD$ = "DEMOCRAT"
5560 PR$ = "REPUBLICAN"
5570 PI$ = "INDEPENDENT"
5580 PO$ = "OTHER"
5590 PW$ = " "
5600 SB$ = "PRESS THE SPACE BAR "
5610 VN$ = "VOTER NO. "
5620 VN = 0
5630 PC$ = "WARD/PCT NO. "
5640 PC = 0
5650 Z5$ = "ZZZZZ"
5660 RETURN
5670 :
     ' ************************************************************
5680 :
     ' **  THIS IS THE MENU RETURN TEST
5690 IF MR$ = "99"
      THEN
        780
5700 RETURN
5710 :
     '
5720 :
     ' ****************************************
5730 :
     ' **  SORT COMBINED ARRAY SUBROUTINE  **
5740 :
     ' ****************************************
5750 :
     '
5760 GOSUB 5330:
     GOSUB 7830:
     SS = 0:
     W = 0:
     TL = 0
5770 PRINT "SORTING THE ARRAY"
5780 FOR N = 1 TO 440 STEP 20
5790  TL = TL + 1
5800  IF AV$(N) < = AV$(N + 20)
       THEN
         5910
5810  IF SS = 0 AND TL = NM + NF
       THEN
         5960
5820  SS = 1
5830  FOR Z = N TO N + 19
5840   VT$(W) = AV$(Z)
5850   PRINT @128,"
5860   PRINT @128,VT$(W)
5870   AV$(Z) = AV$(Z + 20)
5880   AV$(Z + 20) = VT$(W)
5890   W = W + 1
5900   NEXT Z
5910  IF TL = NM + NF
       THEN
         AV$((TL * 20) + 1) = Z5$:
```

```
         GOTO 5940
5920  W = 0
5930  NEXT N
5940 IF SS = 1
      THEN
       5760
5950 SD = 1
5960 RETURN
5970 :
     ' *******************************************************
5980 GOSUB 7880
5990 IF SD = 1
      THEN
       6060
6000 GOSUB 5330:
     PRINT "SORT THE DATA (Y/N)?"
6010 Z$ = INKEY$
6020 IF Z$ = "Y"
      THEN
       GOSUB 5760:
       GOTO 6050
6030 IF Z$ = "N"
      THEN
       6060
6040 GOTO 6010
6050 PRINT :
     PRINT "DATA SORTED"
6060 M = 1:
     F = 1:
     P = 1
6070 GOSUB 5330:
     PRINT :
     PRINT "DISPLAY OPTIONS:"
6080 PRINT
6090 PRINT "<F>EMALE REGISTRANTS"
6100 PRINT
6110 PRINT "<M>ALE REGISTRANTS"
6120 PRINT
6130 PRINT "<A>LL REGISTRANTS"
6140 PRINT
6150 PRINT "SELECT: ";
6160 Z$ = INKEY$
6170 IF Z$ = "A" PRINT Z$:
     GOSUB 6250:
     GOTO 780
6180 IF Z$ = "F" PRINT Z$:
     GOSUB 6460:
     GOTO 780
6190 IF Z$ = "M" PRINT Z$:
     GOSUB 6610:
     GOTO 780
6200 GOTO 6160
6210 :
     '
6220 :
     ' *****************
6230 :
     ' *  DISPLAY ALL   *
6240 :
     ' *****************
6250 FOR N = 1 TO 440 STEP 20
6260  F = 1:
      M = 1
6270  IF AV$(N) = Z5$
       THEN
        6400
6280  IF AV$(N + 17) = "MALE"
       THEN
        6340
6290  FOR Z = N TO N + 19
```

Program continued

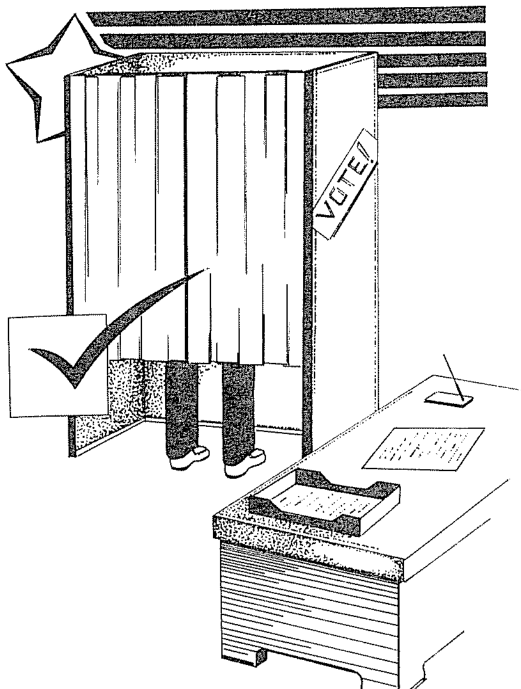
```
6300   VF$(F) = AV$(Z)
6310   F = F + 1
6320   NEXT Z
6330  GOSUB 4880:
      GOSUB 7760:
      GOTO 6390
6340  FOR Z = N TO N + 19
6350   VM$(M) = AV$(Z)
6360   M = M + 1
6370   NEXT Z
6380  GOSUB 4420:
      GOSUB 7760
6390  NEXT N
6400 RETURN
6410 :
     '
6420 :
     ' ********************
6430 :
     ' *  DISPLAY FEMALE  *
6440 :
     ' ********************
6450 :
     '
6460 FOR N = 1 TO 440 STEP 20
6470  F = 1
6480  IF AV$(N) = Z5$
       THEN
        6560
6490  IF AV$(N + 17) = "MALE"
       THEN
        6550
6500  FOR Z = N TO N + 19
6510   VF$(F) = AV$(Z)
6520   F = F + 1
6530   NEXT Z
6540  GOSUB 4880:
      GOSUB 7760
6550  NEXT N
6560 RETURN
6570 :
     '
6580 :
     ' ******************
6590 :
     ' *  DISPLAY MALE  *
6600 :
     ' ******************
6610 FOR N = 1 TO 440 STEP 20
6620  M = 1
6630  IF AV$(N) = Z5$
       THEN
        6710
6640  IF AV$(N + 17) = "FEMALE"
       THEN
        6700
6650  FOR Z = N TO N + 19
6660   VM$(M) = AV$(Z)
6670   M = M + 1
6680   NEXT Z
6690  GOSUB 4420:
      GOSUB 7760
6700  NEXT N
6710 RETURN
6720 :
     '
6730 :
     ' *******************
6740 :
     ' *  PRINT OPTIONS  *
```

```
6750 :
     ' *******************
6760 :
     '
6770 GOSUB 7880
6780 IF SD = 1
      THEN
       6840
6790 GOSUB 5330:
     PRINT "SORT THE DATA (Y/N)?"
6800 Z$ = INKEY$
6810 IF Z$ = "Y"
      THEN
       GOSUB 5760:
       GOTO 6840
6820 IF Z$ = "N"
      THEN
       6840
6830 GOTO 6800
6840 M = 1:
     F = 1:
     P = 1
6850 PRINT :
     PRINT "PRINT OPTIONS:"
6860 PRINT
6870 PRINT "<F>EMALE REGISTRANTS"
6880 PRINT
6890 PRINT "<M>ALE REGISTRANTS"
6900 PRINT
6910 PRINT "<A>LL REGISTRANTS"
6920 PRINT
6930 PRINT "SELECT: ";
6940 Z$ = INKEY$
6950 IF Z$ = "A" PRINT Z$:
     GOTO 7000
6960 IF Z$ = "F" PRINT Z$:
     GOTO 7190
6970 IF Z$ = "M" PRINT Z$:
     GOTO 7310
6980 GOTO 6940
6990 :
     ' *********************
7000 FOR N = 1 TO 440 STEP 20
7010  IF AV$(N) = Z5$
       THEN
        780
7020  IF AV$(N + 17) = "MALE"
       THEN
        7110
7030  W = 1
7040  FOR Z = N TO N + 19
7050   VF$(W) = AV$(Z)
7060   W = W + 1
7070   NEXT Z
7080  F = 1:
      GOSUB 5110
7090  NEXT N
7100 GOTO 780
7110 W = 1
7120 FOR Z = N TO N + 19
7130  VM$(W) = AV$(Z)
7140  W = W + 1
7150  NEXT Z
7160 M = 1:
     GOSUB 4650
7170 GOTO 7090
7180 :
     ' *********************
7190 FOR N = 1 TO 440 STEP 20
7200  IF AV$(N) = Z5$
```

Program continued

```
       THEN
        780
7210  IF AV$(N + 17) = "MALE"
       THEN
        7280
7220  W = 1
7230  FOR Z = N TO N + 19
7240   VF$(W) = AV$(Z)
7250   W = W + 1
7260   NEXT Z
7270  F = 1:
      GOSUB 5110
7280  NEXT N
7290 GOTO 780
7300 :
     ' *******************
7310 FOR N = 1 TO 440 STEP 20
7320  IF AV$(N) = Z5$
       THEN
        780
7330  IF AV$(N + 17) = "FEMALE"
       THEN
        7400
       THEN
        7980
7340  W = 1
7350  FOR Z = N TO N + 19
7360   VM$(W) = AV$(Z)
7370   W = W + 1
7380   NEXT Z
7390  M = 1:
      GOSUB 4650
7400  NEXT N
7410 GOTO 780
7420 :
     ' *******************
7430 GOSUB 5330
7440 PRINT "DO YOU WISH TO"
7450 PRINT "PREPARE A TRANSACTION"
7460 PRINT "TAPE (Y/N)?"
7470 Z$ = INKEY$
7480 IF Z$ = "Y" PRINT "WRITING TRANSACTION TAPE":
     GOTO 7570
7490 IF Z$ = "N"
      THEN
       7560
7500 GOTO 7470
7510 :
     '
7520 :
     ' ****************************
7530 :
     ' *  PREPARE TRANSACTION TAPE  *
7540 :
     ' ****************************
7550 :
     '
7560 END
7570 PRINT @256,"PREPARE TAPE - PRESS SPACE"
7580 Z$ = INKEY$
7590 IF Z$ = " "
      THEN
       7610
7600 GOTO 7580
7610 FOR N = 1 TO 440 STEP 20
7620  PRINT @256,"
7630  FOR Z = N TO N + 19
7640   PRINT # - 1, AV$(Z)
7650   PRINT @256, AV$(Z)
7660   NEXT Z
```

```
7670  IF AV$(Z) = Z5$
       THEN
        7690
7680  NEXT N
7690 PRINT :
     PRINT "TRANSACTION TAPE COMPLETE"
7700 END
7710 :
     '
7720 :
     '  ********************
7730 :
     '  *  STEP THE SCREEN  *
7740 :
     '  ********************
7750 :
     '
7760 PRINT @915,"SPACE BAR TO CONTINUE"
7770 Z$ = INKEY$
7780 IF Z$ = " "
      THEN
       RETURN
7790 GOTO 7770
```




```
7800 :
     '  ***************************
7810 :
     '  *  CLEAR TRANSACTION ARRAY  *
7820 :
     '  ***************************
7830 FOR N = 1 TO 20
7840  VT$(N) = " "
```

Program continued

```
7850  NEXT N
7860 RETURN
7870 :
     ' ************************************************
7880 IF CL = 1
      THEN
       8070
7890 IF SD = 1
      THEN
       8070
7900 GOSUB 5330:
     PRINT "COMBINING THE LIST":
     PRINT
7910 IF NF = 0 PRINT "NO FEMALES"
7920 IF NM = 0 PRINT "NO MALES"
7930 IF NF = 0 AND NM = 0
      THEN
       780
7940 FOR N = 1 TO 220
7950  IF VM$(N) = Z5$
7960  AV$(N) = VM$(N)
7970  NEXT N
7980 W = N
7990 FOR N = 1 TO 220
8000  AV$(W) = VF$(N)
8010  IF VF$(N) = Z5$
       THEN
        8040
8020  W = W + 1
8030  NEXT N
8040 PRINT "LIST COMBINED"
8050 FOR Z = 1 TO 500:
      NEXT Z
8060 CL = 1
8070 RETURN
```

EDUCATION

Keeping Track—
Student Scheduling and Attendance
Part I

Keeping Track—
Student Scheduling and Attendance
Part II

Keeping Track—
Student Scheduling and Attendance
Part I

by Ulderic F. Racine

Keeping Track is a series of 13 programs that perform two primary functions. First, they allow you to create student class schedules. The class schedules can be entered by student, or by teacher and period as a class roster. Second, you can keep attendance data on students. The data can be entered by student or by teacher and period. I developed the series as part of an educational management information system that includes enrollment records, class attendance, and individual education plan management and review.

The programs were designed to provide you with maximum flexibility. The method of schedule input, attendance input, the number of class periods per student, the number of teachers, and the number of class periods for a full day's attendance credit are all user-defined. The minimum system requirements for running the programs are a 48K Model I with two drives or a 48K Model III with one drive. The number of students it can handle is theoretically limited to 999. I would suggest, however, that if you plan to keep schedule and attendance data on more than 330 students with eight or more class periods per day, you should consider creating two or more separate data files by grade or other natural division of the data.

Part I of the series explains how the individual programs work. The programs presented in each part are:

Part I

Master menu (CLASMENU)
Schedule initialization (SCHEDINT)
Schedule input by teacher (TEASCHED)

Part II

Schedule input by student (STDSCHD)
Schedule change program (STDCHANG)
Printout of class roster by teacher (PNTTEACH)

Part III

Attendance initialization (ATTENDIT)
Attendance input by teacher (TEATTEND)
Printout of class schedules by student (PNTSTCHD)
Printout by class name (PNTCLASS)

Part IV

Attendance input by student (STDATEND)
Printout of year-to-date attendance (PNTATEND)
Teacher name change (TEACHANG)

The programs are menu driven. Individual programs will not fit on a regular DOS disk. If you plan to use all the programs with a Model I with TRSDOS 2.3, you should eliminate BACKUP, COPY, and FORMAT. If you are using NEWDOS 2.1 or NEWDOS 80, you must create a minimum system disk. To use a Model III, you do not need to eliminate any programs.

You may wish to configure the DOS disk to boot the master menu automatically. To do this on the Model III, enter the following code from DOS Ready: AUTO BASIC CLASMENU. This allocates three files and boots to the master menu after you enter the date and time. For NEWDOS 2.1 or NEWDOS 80, enter: AUTO BASIC RUN "CLASMENU". With TRSDOS 2.3, you must go through the initialization sequence and run CLASMENU manually unless you have a utility that will run a BASIC program on power-up.

Program Listing 1 is the master menu (CLASMENU). It allows you to load and run any of the options of the program. If you decide to change the name of any or all of the programs, be sure to change the names in this program. The POKEs in the listing are used with the student record change program (STDCHANG).

```
                 CLASSROOM MASTER MENU

OPTIONS:
0—EXIT THIS PROGRAM
1—ENTER CLASS SCHEDULE BY STUDENT
2—ENTER CLASS ROSTER BY TEACHER
3—CHANGE AN EXISTING STUDENT'S SCHEDULE
4—ADD A NEW STUDENT TO A CLASS ROSTER
5—ENTER ATTENDANCE DATA
6—PRINT SCHEDULE DATA BY STUDENT, TEACHER, OR CLASS
  NAME
7—PRINT LISTS OF STUDENTS, TEACHERS, OR CLASS NAMES
8—REPLACE, DELETE, OR CHANGE THE SPELLING OF A
  TEACHER'S NAME
9—PRINT YEAR-TO-DATE ATTENDANCE DATA

ENTER OPTION NUMBER? (0-9)
```

Select the option you wish, type the number, and press ENTER. The CLASMENU program loads the program to execute the option you select and begins execution of that program.

Each time you select the scheduling function, options 1 and 2 on the master menu, the program called Schedule Initialization (SCHEDINT) in

Program Listing 2 is run. This program first checks the drives for the schedule data file. If no file is found on any of the drives, or if you have neglected to put the schedule data disk in a drive, the program displays the message:

```
I HAVE READ THE DISKS CURRENTLY IN THE DRIVES.
THERE IS NO SCHEDULE DATA ON THESE DISKS.
DO YOU HAVE A DISK WITH SCHEDULE DATA? (Y/N)
```

If you have a disk with data already entered, place it in one of the drives, type Y and press ENTER. The program responds:

```
PLEASE PUT THE DISK IN ONE OF THE DRIVES AND PRESS
ENTER.
```

When you are ready, press the ENTER key. If you have already entered schedule data, the program reads the management record in the file and runs the proper program for schedule input.

The second function of the schedule initialization program is to allow you to configure the scheduling function based on your specific needs. If you have not previously entered schedule data, enter N when asked if you have a disk with schedule data. The program then asks you seven questions regarding scheduling. The first three concern which drive will be used to store the three data files you must create before scheduling can begin.

```
ON WHICH DRIVE SHALL I WRITE THE STUDENT DATA?  (1-2-3)

ON WHICH DRIVE SHALL I WRITE THE TEACHER DATA?  (1-2-3)

ON WHICH DRIVE SHALL I WRITE CLASS DATA ? (1-2-3)
```

Type the drive number on which you wish the computer to store the data you are entering. Generally, you will want to store the student, teacher, and class data on the same disk since these files will not fill a disk. Use a blank formatted disk for data storage. After the data files have been initialized, it is no longer necessary to place the data disk(s) in the drive specified above.

The remaining four questions concern the estimated size of the data files.

```
HOW MANY TEACHERS DO YOU WISH TO ENTER ON
THIS SCHEDULE PROGRAM? (99 MAXIMUM)
ENTER NUMBER
```

Enter the number of teachers who will be a part of the schedule program. You should be as accurate as possible, but the program does allow for several extra names automatically.

HOW MANY CLASSES PER STUDENT
DO YOU WANT TO SCHEDULE? (16 MAX)

Enter the number of class periods per day you wish to enter. For example, if you have seven class periods per day, enter 7 in response to this question.

HOW MANY STUDENTS DO YOU WISH TO
ENTER IN THIS SCHEDULE PROGRAM?

Enter the approximate number of students to be scheduled based on the time frame you wish to use. If you plan to reenter the data on a semester or quarterly basis, then the number of students should reflect the number you expect for the time period. You can base the estimate on the number of students per grade if you plan to enter them by grade.

DO YOU WANT TO ENTER THE DATA
BY TEACHER AS A CLASS ROSTER
OR
BY STUDENT AS A SCHEDULE

ENTER 'T' FOR TEACHER OR 'S' FOR STUDENT

Data can be entered by teacher or by student. The type of entry you select has no effect on subsequent programs. If you choose to enter data by teacher in a class roster format, you can enter attendance by student or by teacher. The choice of a scheduling format affects only the method of entering the schedule data. After you answer this last question, the program initializes the data files on the drives you selected and runs the appropriate scheduling program.

Program Listing 3 is the class roster input program (TEASCHED). The scheduling input is by teacher. The program displays each period, and you enter the name of the class and all students in the class for that period. If you have entered schedule data, the program first reads the data already entered. It might take several minutes to respond. If you ended a previous entry session before entering all the periods for a particular teacher, the program begins with the next period to be entered. If you have not entered any schedule data or you ended a previous session after completing all the periods for a teacher, the program asks for the next teacher's name. There is no limit on the number of students that can be entered in a class. When you have entered all the students for a class, type FULL and press ENTER when the program asks for the next student's name. Before the class roster is accepted, the program asks you if the roster is correct. At this time, you can delete a student or change the spelling of a student's name. The screen for the input appears as follows:

TEACHER: CARLSON FRED PERIOD 3
AMERICAN HISTORY

1—BELL MIKE
2—JAMES MARY
3—BARNES JON

ENTER THE FOURTH STUDENT'S NAME OR 'FULL' IF DONE
LAST NAME (SPACE) FIRST NAME (SPACE) MIDDLE INITIAL (IF ANY)
(ENTER) STUDENT'S NAME: FULL

Program Listing 1. *Master menu*

Encyclopedia Loader

```
10 :
   '  MASTER MENU FOR CLASSROOM II ( CLASMENU )
20 :
   '  COPYRIGHT OCTOBER 1, 1981
30 :
   '  ULDERIC F. RACINE
40 :
   '  2520 S.E. ALEXANDER DRIVE
50 :
   '  TOPEKA, KANSAS 66605
100 CLEAR 1000:
    ON ERROR GOTO 370:
    CLS
110 PRINT TAB(20)"CLASSROOM MASTER MENU":
    PRINT :
    PRINT "0 - EXIT THIS PROGRAM":
    PRINT "1 - ENTER CLASS SCHEDULE BY STUDENT":
    PRINT "2 - ENTER CLASS ROSTER BY TEACHER":
    PRINT "3 - CHANGE A EXISTING STUDENT'S SCHEDULE"
120 PRINT "4 - ADD A NEW STUDENT TO A CLASS ROSTER":
    PRINT "5 - ENTER ATTENDANCE DATA":
    PRINT "6 - PRINT SCHEDULE DATA BY STUDENT, TEACHER, OR CLASS NAM
    E":
    PRINT "7 - PRINT LISTS OF STUDENTS, TEACHERS, OR CLASS NAMES"
130 PRINT "8 - REPLACE, DELETE OR CHANGE THE SPELLING OF A TEACHER'S
     NAME"
140 PRINT "9 - PRINT YEAR-TO-DATE ATTENDANCE DATA":
    PRINT :
    LINE INPUT "<ENTER> OPTION # ( 0 - 9 ) ";OP$:
    OP = VAL(OP$):
    IF OP < 0 OR OP > 9
     THEN
      PRINT @832, CHR$(31);:
      GOTO 140
150 IF OP = 0
     THEN
      170
160 ON OP GOTO 190,190,200,205,220,230,290,350,360
170 CLS :
    CLOSE :
    PRINT @448,"YOU MAY REMOVE YOUR DATA DISKS NOW."
180 END
190 RUN "SCHEDINT"
200 POKE 16424,1:
    GOTO 210
205 POKE 16424,5
210 RUN "STDCHANG"
220 RUN "ATTENDIT"
230 CLS :
    PRINT @128,"PRINTOUT OPTIONS :":
    PRINT @256,"1 - PRINT STUDENTS BY SCHEDULE":
    PRINT "2 - PRINT CLASS ROSTER BY TEACHER":
    PRINT "3 - PRINT CLASSES BY STUDENT/TEACHER/PERIOD"
240 PRINT :
    LINE INPUT "<ENTER> OPTION # ( 1 - 3 ) ";OQ$:
    OQ = VAL(OQ$):
    IF OQ < 1 OR OQ > 3
     THEN
      230
250 ON OQ GOTO 260,270,280
260 RUN "PNTSTCHD"
270 RUN "PNTTEACH"
280 RUN "PNTCLASS"
290 CLS :
    PRINT @128,"LIST OPTIONS :":
    PRINT @256,"1 - LIST ALL TEACHERS CURRENTLY ON FILE":
    PRINT "2 - LIST ALL CLASSES CURRENTLY ON FILE":
```

```
    PRINT "3 - LIST ALL STUDENTS CURRENTLY ON FILE":
    PRINT
300 LINE INPUT "<ENTER> OPTION # ( 1 - 3 ) ";OS$:
    OS = VAL(OS$):
    IF OS < 1 OR OS > 3
     THEN
      290
310 ON OS GOTO 320,330,340
320 POKE 16424,2:
    GOTO 210
330 POKE 16424,3:
    GOTO 210
340 POKE 16424,4:
    GOTO 210
350 RUN "TEACHANG"
360 RUN "PNTATEND"
370 RESUME 100
```

Program Listing 2. *Schedule initialization*

```
 10 :
    '   SCHEDULE INITIALIZATION  PROGRAM ( SCHEDINT )
 20 :
    '   COPYRIGHT OCTOBER 1, 1981
 30 :
    '   ULDERIC F. RACINE
 40 :
    '   2520 S.E. ALEXANDER DRIVE
 50 :
    '   TOPEKA, KANSAS 66605
100 CLS :
    CLEAR 5000:
    ON ERROR GOTO 430:
    Z$ = CHR$(31)
110 OPEN "R",1,"STDSCHED":
    RN = LOF (1):
    IF RN = 0
     THEN
      CLOSE :
      KILL "STDSCHED":
      GOTO 130
120 UR = 1:
    GOTO 420
130 CLS :
    AN$ = "":
    PRINT @448,"I HAVE READ THE DISKS CURRENTLY IN THE DRIVES.":
    PRINT "THERE IS NO SCHEDULE DATA ON THESE DISKS.":
    LINE INPUT "DO YOU HAVE A DISK WITH SCHEDULE DATA ? ( Y/N )  ";A
    N$:
    GOSUB 480
140 IF AN$ = "Y"
     THEN
      150:
     ELSE
      IF AN$ < > "N"
       THEN
        130:
       ELSE
        160
150 CLS :
    PRINT @448,"PLEASE PUT THE DISK IN A DRIVE OTHER THAN ZERO.":
    LINE INPUT "PRESS <ENTER> ";AN$:
    GOTO 100
160 PRINT @448,Z$:
    LINE INPUT "ON WHICH DRIVE SHALL I WRITE STUDENT DATA ? ( 1 - 2
    - 3 )  ";DR$:
```

Program continued

```
    IF VAL(DR$) < 1 OR VAL(DR$) > 3
     THEN
      CLS :
      DR$ = "":
      GOTO 160
170 LINE INPUT "ON WHICH DRIVE SHALL I WRITE TEACHER DATA ? ( 1 - 2
    - 3 )  ";DS$:
    IF VAL(DS$) < 1 OR VAL(DS$) > 3
     THEN
      CLS :
      DS$ = "":
      GOTO 170
180 LINE INPUT "ON WHICH DRIVE SHALL I WRITE CLASS DATA ? ( 1 - 2 -
    3 )  ";DT$:
    IF VAL(DT$) < 1 OR VAL(DT$) > 3
     THEN
      CLS :
      DT$ = "":
      GOTO 180
240 DR$ = "STDSCHED:" + DR$:
    DS$ = "TEACHER:" + DS$:
    DT$ = "CLASSES:" + DT$
250 PRINT @448,Z$;"HOW MANY TEACHERS DO YOU WISH TO SCHEDULE ? (99 M
    AXIMUM) ":
    LINE INPUT "TYPE NUMBER. PRESS <ENTER> ";NT$:
    NT = VAL(NT$):
    IF NT < 1
     THEN
      250:
     ELSE
      IF NT > 99
       THEN
        250:
       ELSE
        IF NT + 10 < 99
         THEN
          NT = NT + 10:
         ELSE
          NT = 99
260 PRINT @448,Z$;:
    LINE INPUT "HOW MANY CLASSES PER STUDENTDO YOU WANT TO SCHEDULE
    ? (16 MAX)  ";NP$:
    NP = VAL(NP$):
    IF NP < 1 OR NP > 16
     THEN
      260
270 PRINT @448,Z$;:
    LINE INPUT "HOW MANY STUDENTS DO YOU WISH TO ENTER IN THIS SCHED
    ULE PROGRAM ? ";NS$:
    NS = VAL(NS$):
    IF NS < 1
     THEN
      270:
     ELSE
      NS = NS + 15
280 PRINT @448,Z$;:
    PRINT "DO YOU WANT TO ENTER THE DATABY TEACHER AS A CLASS ROSTER
                ORBY STUDENT AS A SCHEDULE ?"
290 PRINT :
    PRINT "TYPE 'T' FOR TEACHER OR 'S' FOR STUDENT ":
    LINE INPUT "PRESS <ENTER> ";AN$:
    AN$ = LEFT$(AN$,1):
    IF AN$ = "T" OR AN$ = "S"
     THEN
      300:
     ELSE
      280
300 IF AN$ = "T"
     THEN
      T = 1:
     ELSE
```

```
     IF AN$ = "S"
      THEN
       T = 2:
      ELSE
       280
310 RN = 2:
    Q = 0:
    PN = 0:
    FS = 24 + (NP * 5):
    NC = ( INT(NT * NP) * .6)
320 IF FS < = 64
     THEN
      UF = 4:
      GOTO 370
330 IF FS < = 85
     THEN
      UF = 3:
      GOTO 370
340 IF FS < = 128
     THEN
      UF = 2:
      GOTO 370
350 STOP
370 OPEN "R",1,DR$:
    OPEN "R",2,DS$:
    OPEN "R",3,DT$
380 FIELD 1,2ASFA$,2ASFB$,2ASFC$,2ASFD$,2ASFE$,2ASFG$,2ASFH$,2ASFI$,
    2ASFJ$,2ASFK$
390 LSET FA$ = MKI$ (T):
    LSET FB$ = MKI$ (FS):
    LSET FC$ = MKI$ (UF):
    LSET FD$ = MKI$ (NT):
    LSET FE$ = MKI$ (NC):
    LSET FG$ = MKI$ (NP):
    LSET FH$ = MKI$ (RN):
    LSET FI$ = MKI$ (Q):
    LSET FJ$ = MKI$ (NS):
    LSET FK$ = MKI$ (PN)
400 PUT 1,1
410 CLOSE :
    IF T = 1
     THEN
      RUN "TEASCHED":
     ELSE
      RUN "STDSCHD"
420 FIELD 1,2ASFA$:
    GET 1,1:
    T = CVI (FA$):
    GOTO 410
430 CLS :
    PRINT @394,"AN ERROR HAS OCCURRED IN THE EXECUTION OF THE PROGRA
    MCALLED 'SCHEDULE INITIALIZATION'."
440 PRINT TAB(5)"ERROR TYPE = "; ERR / 2 + 1
450 PRINT TAB(5)"ERROR LINE = "; ERL
460 FOR V = 1 TO 5000:
     NEXT V
470 RESUME 350
480 AN$ = LEFT$(AN$,1):
    RETURN
```

Program Listing 3. *Schedule input by teacher*

```
10 :
   '  SCHEDULE INPUT BY TEACHER ( TEASCHED )
20 :
   '  COPYRIGHT OCTOBER 1, 1981
```

Program continued

```
 30 :
    '  ULDERIC F. RACINE
 40 :
    '  2520 S.E. ALEXANDER DRIVE
 50 :
    '  TOPEKA, KANSAS 66605
100 CLS :
    PRINT CHR$(23):
    PRINT TAB(2)"SCHEDULE INPUT BY TEACHER"
110 CLEAR 2000:
    UR = 1:
    GOTO 130
120 CLEAR T
130 OPEN "R",1,"STDSCHED"
140 FIELD 1,2ASFA$,2ASFB$,2ASFC$,2ASFD$,2ASFE$,2ASFG$,2ASFH$,2ASFI$,
    2ASFJ$,2ASFK$
150 GET 1,1
160 TX = LOF (1)
170 FS = CVI (FB$):
    UF = CVI (FC$):
    NT = CVI (FD$):
    NC = CVI (FE$):
    NP = CVI (FG$):
    NS = CVI (FJ$):
    PN = CVI (FK$)
180 IF UR = 1
     THEN
      CLOSE :
      T = (NT * 15) + (NC * 15) + (NS * FS) + 1000:
      GOTO 120
190 DIM TN$(NT),CN$(NC),SN$(NS),ST$(30)
200 ON ERROR GOTO 1660
210 IF TX < 2
     THEN
      NT = 0:
      PN = 0:
      NC = 0:
      NS = 0:
      CLOSE :
      GOTO 560
220 OPEN "R",2,"TEACHER":
    RA = 1:
    Q1 = 0:
    X = 0
230 G = Q1 * 25
240 FIELD 2,(G)ASDUMMY$,25ASA1$
250 GET 2,RA
260 IF A1$ = STRING$(25,88)
     THEN
      NT = X:
      GOTO 340
270 X = X + 1
280 FOR Y = 1 TO 25
290  IF MID$(A1$,Y,2) = "  "
      THEN
       TN$(X) = LEFT$(A1$,Y - 1):
       GOTO 320
300  NEXT Y
310 TN$(X) = A1$
320 Q1 = Q1 + 1:
    IF Q1 = 10
     THEN
      Q1 = 0:
      RA = RA + 1
330 GOTO 230
340 OPEN "R",3,"CLASSES":
    Q1 = 0:
    RA = 1:
    X = 0
350 G = Q1 * 25
360 FIELD 3,(G)ASDUMMY$,25ASA2$
```

```
370 GET 3,RA
380 IF A2$ = STRING$(25,88)
     THEN
      NC = X:
      GOTO 460
390 X = X + 1
400 FOR Y = 1 TO 25
410  IF MID$(A2$,Y,2) = "  "
      THEN
       CN$(X) = LEFT$(A2$,Y - 1):
       GOTO 440
420  NEXT Y
430 CN$(X) = A2$
440 Q1 = Q1 + 1:
    IF Q1 = 10
     THEN
      Q1 = 0:
      RA = RA + 1
450 GOTO 350
460 IF TX > 1
     THEN
      470:
     ELSE
      CLOSE :
      GOTO 560
470 RA = 2:
    Q1 = 0:
    X = 0
480 G = Q1 * FS
490 FIELD 1,(G)ASDUMMY$,(FS)ASA3$
500 GET 1,RA
510 IF A3$ = STRING$(FS,88)
     THEN
      NS = X:
      CLOSE :
      GOTO 560
520 X = X + 1
530 SN$(X) = A3$
540 Q1 = Q1 + 1:
    IF Q1 = UF
     THEN
      Q1 = 0:
      RA = RA + 1
550 GOTO 480
560 CLS :
    PRINT "CLASS ROSTER INPUT PROGRAM BY TEACHER":
    PRINT @128,"OPTIONS :":
    PRINT @256,"1 - ENTER CLASS ROSTERS BY TEACHER":
    PRINT "2 - EXIT THIS PROGRAM AND RETURN TO MASTER MENU"
570 PRINT @512,"";:
    LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
    OP = VAL(OP$):
    IF OP < 1 OR OP > 2
     THEN
      PRINT @512, CHR$(31);:
      GOTO 570
580 ON OP GOTO 590,1550
590 IF PN < > 0
     THEN
      630
600 PN = 1:
    CLS :
    PRINT @448,"<ENTER> TEACHER'S NAME":
    PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INITIAL (IF A
    NY)":
    NT = NT + 1
610 LINE INPUT "<ENTER> NAME : ";TN$(NT):
    V1 = 1:
    GOSUB 1560:
    TN$(NT) = TS$:
    B1 = NT
```

Program continued

```
620 PRINT @448, CHR$(31);:
    PRINT "TEACHER'S NAME :  ";TN$(NT):
    PRINT :
    LINE INPUT "IS THIS NAME CORRECT ? ( Y/N ) ";AN$:
    GOSUB 1710:
    IF AN$ = "Y"
     THEN
      630:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        620:
       ELSE
        NT = NT - 1:
        GOTO 600
630 CLS :
    PRINT TN$(NT) TAB(35)"PERIOD";PN:
    B1 = NT
640 PRINT @448,"";:
    LINE INPUT "<ENTER> CLASS NAME : ";CS$
650 PRINT @448, CHR$(31);:
    PRINT "CLASS NAME :  ";CS$:
    PRINT :
    LINE INPUT "IS THIS NAME CORRECT ? ( Y/N ) ";AN$:
    GOSUB 1710:
    IF AN$ = "Y"
     THEN
      660:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        650:
       ELSE
        PRINT @448, CHR$(31);:
        GOTO 640
660 K = LEN(CS$)
670 IF NC = 0
     THEN
      NC = 1:
      CN$(NC) = CS$:
      B2 = NC:
      GOTO 710
680 FOR X = 1 TO NC
690  IF LEFT$(CN$(X),K) = CS$
      THEN
       B2 = X:
       GOTO 710
700  NEXT X:
    NC = NC + 1:
    CN$(NC) = CS$:
    B2 = NC:
    GOTO 710
710 IF B1 < 10
     THEN
      D$ = "0" + RIGHT$( STR$(B1),1):
     ELSE
      D$ = RIGHT$( STR$(B1),2)
720 IF B2 < 10
     THEN
      D1$ = "00" + RIGHT$( STR$(B2),1):
      GOTO 740
730 IF B2 < 100
     THEN
      D1$ = "0" + RIGHT$( STR$(B2),2):
     ELSE
      D1$ = RIGHT$( STR$(B2),3)
740 D2$ = D$ + D1$
750 CLS :
```

```
    PRINT TN$(NT) TAB(35)"PERIOD";PN:
    PRINT CN$(B2):
    PRINT STRING$(62,45):
    ST = 0:
    Z = 192
760 ST = ST + 1
770 PRINT @768,"<ENTER>";ST;"STUDENT'S NAME OR 'FULL' IF DONE"
780 PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INITIAL (IF A
    NY)"
790 LINE INPUT "<ENTER> STUDENT'S NAME : ";ST$(ST)
800 IF ST$(ST) = "FULL"
     THEN
      ST = ST - 1:
      IF ST = 0
       THEN
        1120:
       ELSE
        860
810 V1 = 2:
    GOSUB 1560:
    ST$(ST) = TS$
820 PRINT @768, CHR$(31);:
    IF UQ = 1
     THEN
      840
830 PRINT @Z,ST;" - ";ST$(ST);:
    UQ = 1:
    Z = Z + 35:
    GOTO 850
840 PRINT @Z,ST;" - ";ST$(ST):
    UQ = 0:
    Z = Z + 29
850 GOTO 760
860 CLS :
    UQ = 0:
    FOR X = 1 TO ST
870  IF UQ = 0
      THEN
       880:
      ELSE
       890
880  PRINT X;" - ";ST$(X);:
     UQ = 1:
     GOTO 900
890  PRINT TAB(35)X;" - ";ST$(X):
     UQ = 0
900  NEXT X
910 IF UQ = 0
     THEN
      PRINT :
     ELSE
      PRINT :
      PRINT
920 LINE INPUT "IS THIS CLASS ROSTER CORRECT ? ( Y/N ) ";AN$:
    GOSUB 1710:
    IF AN$ = "Y"
     THEN
      1030:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        860:
       ELSE
        930
930 GOSUB 1640
940 PRINT :
    LINE INPUT "DO YOU WISH TO DELETE ANY OF THE STUDENTS ? ( Y/N )
    ";AN$:
    GOSUB 1710:
```

Program continued

```
    IF AN$ = "Y"
     THEN
      950:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        930:
       ELSE
        980
950 GOSUB 1640
960 PRINT :
    LINE INPUT "<ENTER> THE NUMBER OF THE STUDENT YOU WISH TO DELETE
     # ";ND$:
    ND = VAL(ND$):
    IF ND < 1 OR ND > ST
     THEN
      GOSUB 1650:
      GOSUB 1640:
      GOTO 960
970 ST$(ND) = "":
    GOTO 860
980 GOSUB 1640
990 PRINT :
    LINE INPUT "DO YOU WISH TO CHANGE THE SPELLINGOF A STUDENT'S NAM
    E ? ( Y/N ) ";AN$:
    GOSUB 1710:
    IF AN$ = "Y"
     THEN
      1000:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        980:
       ELSE
        860
1000 GOSUB 1640
1010 PRINT :
     LINE INPUT "<ENTER> THE STUDENT'S NUMBER # ";ND$:
     ND = VAL(ND$):
     IF ND < 1 OR ND > ST
      THEN
       GOSUB 1650:
       GOSUB 1640:
       GOTO 1010
1020 GOSUB 1640:
     PRINT :
     LINE INPUT "<ENTER> NEW SPELLING : ";TS$:
     GOSUB 1590:
     ST$(ND) = TS$:
     GOTO 860
1030 Y = 25 + ((PN - 1) * 5):
     Z = Y + 2:
     FOR X = 1 TO ST
1040  K = LEN(ST$(X))
1050  FOR X1 = 1 TO NS
1060   IF LEFT$(SN$(X1),K) = ST$(X)
        THEN
         1070:
        ELSE
         1080
1070   SN$(X1) = LEFT$(SN$(X1),Y - 1) + D2$ + RIGHT$(SN$(X1),FS
       - (Z + 2)):
       ST$(X) = "":
       GOTO 1110
1080   NEXT X1
1090  NS = NS + 1
1100  SN$(NS) = ST$(X) + STRING$(24 - K,32) + STRING$(Y - 25,32)
      + D2$ + STRING$(FS - (Z + 2),32):
```

```
     ST$(X) = ""
1110  NEXT X
1120 PN = PN + 1:
     IF PN > NP
      THEN
       PN = 0
1130 AN$ = "":
     CLS
1140 IF PN = 0
      THEN
       PRINT @448,"";:
       LINE INPUT "ARE YOU READY TO ENTER ANOTHER TEACHER ? ( Y/N ) "
       ;AN$:
       GOSUB 1710:
       IF AN$ = "Y"
        THEN
         590:
         :
        ELSE
         IF AN$ < > "N"
          THEN
           1130:
          ELSE
           1160
1150 PRINT @448,"ARE YOU READY TO ENTER THE";PN;"PERIOD":
     PRINT "FOR ";TN$(NT);:
     LINE INPUT " ? ( Y/N ) ";AN$:
     GOSUB 1710:
     IF AN$ = "Y"
      THEN
       630:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1130:
        ELSE
         1160
1160 OPEN "R",1,"STDSCHED"
1170 FIELD 1,18ASDUMMY$,2ASB1$
1180 GET 1,1
1190 LSET B1$ = MKI$ (PN)
1200 PUT 1,1
1210 RA = 2:
     Q = 0
1220 FOR X = 1 TO NS
1230  G = Q * FS
1240  FIELD 1,(G)ASDUMMY$,(FS)ASSD$
1250  LSET SD$ = SN$(X)
1260  IF UR = 1
       THEN
        1280
1270  Q = Q + 1:
      IF Q = UF
       THEN
        1280:
       ELSE
        1300
1280  PUT 1,RA:
      Q = 0:
      RA = RA + 1
1290  IF UR = 1
       THEN
        1320
1300  NEXT X
1310 SN$(X) = STRING$(FS,88):
     UR = 1:
     GOTO 1230
1320 OPEN "R",2,"TEACHER":
     RA = 1:
```

Program continued

```
     Q = 0:
     UR = 0
1330 FOR X = 1 TO NT
1340  G = Q * 25
1350  FIELD 2,(G)ASDUMMY$,25ASSE$
1360  LSET SE$ = TN$(X)
1370  IF UR = 1
       THEN
        1390
1380  Q = Q + 1:
      IF Q = 10
       THEN
        1390:
       ELSE
        1410
1390  PUT 2,RA:
      Q = 0:
      RA = RA + 1
1400  IF UR = 1
       THEN
        1430
1410  NEXT X
1420 TN$(X) = STRING$(25,88):
     UR = 1:
     GOTO 1340
1430 OPEN "R",3,"CLASSES":
     RA = 1:
     Q = 0:
     UR = 0
1440 FOR X = 1 TO NC
1450  G = Q * 25
1460  FIELD 3,(G)ASDUMMY$,25ASSF$
1470  LSET SF$ = CN$(X)
1480  IF UR = 1
       THEN
        1500
1490  Q = Q + 1:
      IF Q = 10
       THEN
        1500:
       ELSE
        1520
1500  PUT 3,RA:
      Q = 0:
      RA = RA + 1
1510  IF UR = 1
       THEN
        1540
1520  NEXT X
1530 CN$(X) = STRING$(25,88):
     UR = 1:
     GOTO 1450
1540 CLOSE
1550 RUN "CLASMENU"
1560 ON V1 GOTO 1570,1580
1570 TS$ = TN$(NT):
     GOTO 1590
1580 TS$ = ST$(ST)
1590 K = LEN(TS$)
1600 FOR X = 1 TO K
1610  IF (( MID$(TS$,X,1) = ",") OR ( MID$(TS$,X,1) = ".")) AND
      MID$(TS$,X + 1,1) = CHR$(32)
       THEN
        TS$ = LEFT$(TS$,X - 1) + RIGHT$(TS$,K - X):
        GOTO 1630
1620  IF ( MID$(TS$,X,1) = ",") OR ( MID$(TS$,X,1) = ".")
       THEN
        TS$ = LEFT$(TS$,X - 1) + " " + RIGHT$(TS$,(K - X))
1630  NEXT X:
     RETURN
```

```
1640 PRINT @ INT(X / 2) * 64, CHR$(31);:
     RETURN
1650 PRINT @ INT(X / 2) * 64, CHR$(31);:
     PRINT "THERE IS NO STUDENT # ";ND:
     FOR Y = 1 TO 400:
      NEXT :
     RETURN
1660 CLS :
     PRINT @394,"AN ERROR HAS OCCURRED IN THE EXECUTION OF THE PROGRA
     MCALLED 'SCHEDULE INPUT BY TEACHER'."
1670 PRINT TAB(5)"ERROR TYPE  = "; ERR / 2 + 1
1680 PRINT TAB(5)"ERROR LINE  = "; ERL
1690 FOR V = 1 TO 5000:
      NEXT V
1700 STOP
1710 AN$ = LEFT$(AN$,1):
     RETURN
```

Keeping Track— Student Scheduling and Attendance Part II

by Ulderic F. Racine

Part I of Keeping Track gave the program listings for the master menu, attendance initialization, and schedule input by teacher. Part II contains the listings for schedule input by student, changing student records, and printing schedules by teacher.

Program Listing 1 is the schedule input by student program (STDSCHD). It allows you to enter the data as a schedule by student. The program asks for the name of the teacher and the class name for each class period. If the name of the teacher or the class is the same as the previous period, you do not have to reenter the name. Type S when asked for the teacher or class name and it will be duplicated on the screen. If a student has no class and no teacher for a period, enter NT for the teacher's name and NC for the class name. If you have previously entered schedule data, the program first reads the teacher and class names already entered. It may take several minutes to make a response. An example of the video display for this program appears below.

```
STUDENT: JAMES MARY
PERIOD       TEACHER          CLASS
-----------------------------------------------------
1            JONES            MATH I
2            BURTON           GEOGRAPHY
3            CARLSON          AMERICAN HISTORY

ENTER 'NT' FOR 'NO TEACHER'
ENTER 'S' IF THE TEACHER IS THE SAME AS THE THIRD PERIOD
LAST NAME (SPACE) FIRST NAME (SPACE) MIDDLE INITIAL (IF ANY)
(ENTER) FOURTH PERIOD TEACHER'S NAME:
```

Program Listing 2 is the student schedule change program (STDCHANG). In addition to performing the change function, this program also produces listings of students, teachers, and classes. The function called from the master menu is POKEd into the printer control block and read by this program. The program then jumps to the appropriate part of the program based on the PEEK value. The student schedule change program allows you to change an existing student's schedule whether the schedule was input by teacher or by student. It also allows you to add a new student to an existing class roster if the data was input by teacher. A new student is defined as one whose schedule you have not entered previously. If the schedule for the named student already exists, the program automatically switches to the change function. If you select option 3 to change a student's schedule, the screen displays the following:

```
ENTER THE NAME OF THE STUDENT WHOSE SCHEDULE
YOU WISH TO CHANGE.
LAST NAME (SPACE) FIRST NAME (SPACE) MIDDLE INITIAL (IF ANY)
NAME: JOHNSON FRED
```

The program searches the student schedule file for the name of the student. If the student whose name you entered is not on the file, or you entered the name incorrectly, the program displays the following message:

```
I CANNOT FIND A STUDENT NAMED JOHNSON FRED
IN MY STUDENT FILE
WHAT SHALL I DO NOW?
1-TRY ANOTHER STUDENT
2-LIST ALL STUDENTS IN THE FILE
3-EXIT THIS PROGRAM
ENTER OPTION NUMBER
```

If you choose option 1, the program returns you to the previous display. If you typed the name incorrectly or have another student whose schedule you wish to see, type the name and press ENTER. If you choose option 2, the program lists all the students currently on the student schedule file. You have the option of printing a hard-copy report of the students on file. When the listing is finished, the program returns to the three options shown above. If you select option 3, the program returns to the master menu.

If the program finds the name of the student on the file, it displays the schedule on the screen. If you have entered more than 10 periods, it displays the first 10, followed by the remaining periods.

```
STUDENT: JOHNSON FRED
PERIOD      TEACHER          CLASS
------------------------------------------------
1           JONES            MATH I
2           CARLSON          LITERATURE
3           NO TEACHER       NO CLASS
4           BURTON           HISTORY
5           MORTON           SOCIAL STUDIES
6           ADAMS            GYM
THIS IS THE CURRENT SCHEDULE INFORMATION I HAVE ON
JOHNSON FRED
IS THIS INFORMATION CORRECT? (Y/N)
```

If the information is incorrect, type N and press ENTER. The program asks for the number of the period you wish to change, then displays the name of the teacher currently listed for that period.

```
THE TEACHER CURRENTLY LISTED FOR PERIOD 4 IS BURTON.
DO YOU WISH TO CHANGE THE TEACHER'S NAME? (Y/N) Y
NEW TEACHER'S NAME
LAST NAME (SPACE) FIRST NAME (SPACE) MIDDLE INITIAL (IF ANY)
ENTER 'NT' IF YOU WISH TO DROP THIS STUDENT FROM THIS CLASS
ENTER NAME:
```

If you wish to drop the student from the class, type NT and press ENTER. The entry for the teacher for that period will read NO TEACHER. If the student is changing teachers, type the name of the new teacher and press ENTER. If the student currently has no class, as Fred Johnson has no class for period 3 in the above example, the name of the current teacher is displayed as NO TEACHER. You can enter the name of the teacher in whose class the student is enrolling.

The program tries to match the name of the teacher with the names of the teachers already on file. If it does not find a match, it displays all the teachers on file whose names begin with the first character of the last name of the teacher entered above. It then asks if you want to add the name of the new teacher (the name it is unable to match) to the file. If you have typed the name wrong, type N and press ENTER. The program returns you to the schedule display of the student and starts over.

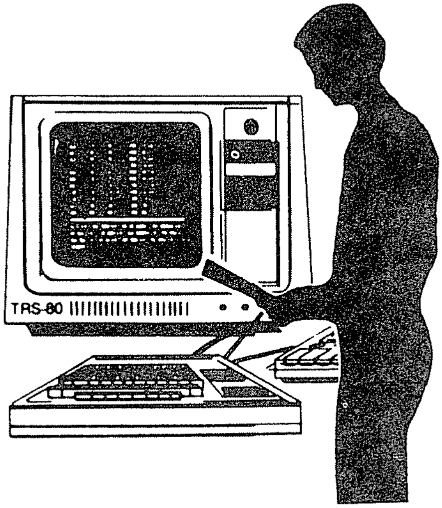
The same format is followed for the class name. The program displays the current class name for the period you select. If there is no class scheduled, the class name appears as NO CLASS. You can drop a student from a class, change classes, or add a new class. If the name you enter does not match one of the classes on file, the program displays all classes beginning with the first letter of the class you entered. It then asks if you wish to add the new class to the class file. The program displays the schedule with the changes incorporated and asks if the information is correct. If you respond with Y, the program asks for permission to file the changes made. After it has filed the changes, the program returns to the menu for the student schedule change program. If you have no more changes to make, select option 2, and the program returns you to the master menu.

Option 4 from the master menu allows you to add a student to an existing class roster. The program asks you for the name of the student to be added and then checks the student file to confirm that the student is not currently enrolled. The program displays the name of the student and the student's schedule as in option 3 above. The first time, the program shows NO TEACHER and NO CLASS for all class periods. The procedure is the same as changing an existing student schedule. You must select the period and enter the data.

Program Listing 3 is used for the teacher roster printout (PNTTEACH). You can print schedule data by teacher, class, and period for all teachers or select a specific teacher and any or all of the class periods. As a result, you can send an update to a specific teacher for a specific period or periods without printing all the teachers' rosters. With either choice, you have the option of printing hard copy. If you choose to print hard copy, the program offers to generate a test line so you can determine the correct printer setting. If you generate a test line, the computer continues to ask if you want a line generated until you type N and press ENTER. The following is an example of the printout.

TEACHER: JONES
PERIOD: 1 CLASS: MATH I

ABBOT THOMAS
DELL CHARLES
RANGLE JOSEPH



Program Listing 1. *Schedule input by student*

Encyclopedia Loader

```
10  :
    '  STUDENT SCHEDULE INPUT PROGRAM ( STDSCHD )
20  :
    '  COPYRIGHT OCTOBER 1, 1981
30  :
    '  ULDERIC F. RACINE
40  :
    '  2520 S.E. ALEXANDER DRIVE
50  :
    '  TOPEKA, KANSAS 66605
100 CLS :
    PRINT CHR$(23)
110 PRINT @448,"STUDENT SCHEDULE INPUT PROGRAM"
120 PRINT "          CLASSROOM II"
130 CLEAR 5000
140 UR = 1
150 OPEN "R",1,"STDSCHED"
160 FIELD 1,2ASFA$,2ASFB$,2ASFC$,2ASFD$,2ASFE$,2ASFG$,2ASFH$,2ASFI$,
    2ASFJ$,2ASFK$
170 GET 1,1:
    TX = LOF (1)
180 FS = CVI (FB$):
    UF = CVI (FC$):
    NX = CVI (FD$):
    NY = CVI (FE$):
    NP = CVI (FG$):
    RN = CVI (FH$):
    Q = CVI (FI$)
190 CLOSE :
    IF UR = 0
     THEN
      210
200 T = (NX * 25) + (NY * 25) + 2000:
    CLEAR T:
    GOTO 150
210 DIM TN$(NX),CN$(NY),N$(NP),NT$(NP),NC$(NP)
220 ON ERROR GOTO 1870
230 C = 0:
    FOR X = 1 TO NP:
     READ N$(X):
     NEXT X
240 IF TX > 1
     THEN
      1440
250 C = C + 1
260 CLS :
    PRINT "STUDENT NAME FORMAT"
270 PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INITIAL ( IF
    ANY )":
    PRINT
280 LINE INPUT "<ENTER> STUDENT'S NAME : ";SN$(C):
    V1 = 1:
    GOSUB 1750:
    SN$(C) = TX$
290 IF LEN(SN$(C)) > 24
     THEN
      SN$(C) = LEFT$(SN$(C),24)
300 IF LEN(SN$(C)) < 24
     THEN
      SN$(C) = SN$(C) + STRING$(24 - LEN(SN$(C)),32)
310 CLS :
    PRINT "THE STUDENT'S NAME IS : ";SN$(C):
    PRINT
320 LINE INPUT "IS THE NAME CORRECT ? ( Y/N )  ";AN$:
    GOSUB 1860:
    IF AN$ = "Y"
```

```
     THEN
      340:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        310:
       ELSE
        330
330 LINE INPUT "PLEASE <ENTER> CORRECT NAME. ";SN$(C):
    V1 = 1:
    GOSUB 1750:
    SN$(C) = TX$:
    GOTO 290
340 CLS :
    PRINT "STUDENT : ";SN$(C):
    Y1 = 128
350 PRINT :
    PRINT "PERIOD"; TAB(10)"TEACHER"; TAB(30)"CLASS":
    PRINT
360 FOR X = 1 TO NP
370  PRINT @704,"ENTER 'NT' FOR 'NO TEACHER'":
     IF X = 1
      THEN
       390:
      ELSE
       380
380  PRINT "ENTER 'S' IF THE TEACHER IS THE SAME AS THE ";N$(X
     - 1);" PERIOD."
390  PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INITIAL ( IF
      ANY )"
400  PRINT "<ENTER> ";N$(X);:
     LINE INPUT " PERIOD TEACHER'S NAME : ";NT$(X):
     V1 = 2:
     GOSUB 1750:
     NT$(X) = TX$
410  IF NT$(X) = "NT"
      THEN
       NT$(X) = "NO TEACHER":
       GOTO 450
420  IF NT$(X) = "S" AND X = 1
      THEN
       PRINT @704, CHR$(31);:
       GOTO 370
430  IF NT$(X) = "S"
      THEN
       NT$(X) = NT$(X - 1):
       GOTO 450
440  IF LEN(NT$(X)) > 25
      THEN
       NT$(X) = LEFT$(NT$(X),25)
450  PRINT @704, CHR$(31);
460  IF X = 1
      THEN
       480:
      ELSE
       470
470  PRINT "ENTER 'S' IF THE CLASS IS THE SAME AS THE ";N$(X
     - 1);" PERIOD."
480  PRINT "ENTER 'NC' FOR NO CLASS.":
     PRINT "<ENTER> ";N$(X);:
     LINE INPUT " CLASS NAME : ";NC$(X):
     V1 = 3:
     GOSUB 1750:
     NC$(X) = TX$
490  IF NC$(X) = "NC"
      THEN
       NC$(X) = "NO CLASS":
       GOTO 520
```

Program continued

```
500  IF X = 1 AND NC$(X) = "S"
      THEN
       450
510  IF NC$(X) = "S"
      THEN
       NC$(X) = NC$(X - 1)
520  IF Y1 + 64 > = 640
      THEN
       PRINT @192, CHR$(31);:
       Y1 = 128
530  Y1 = Y1 + 64:
     PRINT @Y1,X; TAB(10)NT$(X); TAB(30)NC$(X)
540  PRINT @704, CHR$(31);:
     NEXT X
550 IF NP > 8
     THEN
      Y = 1:
      PN = 8:
      :
     ELSE
      PN = NP:
      Y = 1
560 CLS :
    AN$ = "":
    IC$ = ""
570 PRINT "PERIOD"; TAB(10)"TEACHER"; TAB(30)"CLASS NAME":
    PRINT STRING$(63,45)
580 FOR X = Y TO PN
590  PRINT TAB(3);X; TAB(10)NT$(X); TAB(30)NC$(X)
600  NEXT X
610 PRINT :
    PRINT "THIS IS THE INFORMATION THAT WILL BE RECORDEDON THE RECOR
    D FOR PERIODS (";Y;" - ";PN;")."
620 LINE INPUT "IS THIS INFORMATION CORRECT ? ( Y/N )  ";AN$:
    GOSUB 1860
630 IF AN$ = "Y"
     THEN
      720:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        560:
       ELSE
        640
640 PRINT "WHICH PERIOD IS INCORRECT (";Y;" - ";PN;:
    LINE INPUT ")  ";IC$:
    NC = VAL(IC$)
650 IF NC < Y OR NC > PN
     THEN
      NC = 0:
      GOTO 560
660 CLS :
    PRINT @448,"THE ";N$(NC);" PERIOD TEACHER IS -  ";NT$(NC)
670 AN$ = "":
    LINE INPUT "IS THIS NAME CORRECT ? ( Y/N )  ";AN$:
    GOSUB 1860:
    IF AN$ = "Y"
     THEN
      690:
     ELSE
      IF AN$ < > "N"
       THEN
        660:
       ELSE
        680
680 X = NC:
    LINE INPUT "<ENTER> CORRECT NAME : ";NT$(NC):
    V1 = 2:
    GOSUB 1750:
```

```
    NT$(NC) = TX$
690 PRINT @704,"THE ";N$(NC);" CLASS IS -  ";NC$(NC)
700 X = NC:
    AN$ = "":
    LINE INPUT "IS THIS THE CORRECT CLASS NAME ? ( Y/N )  ";AN$:
    GOSUB 1860:
    IF AN$ = "Y"
     THEN
      560:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        690:
       ELSE
        710
710 LINE INPUT "<ENTER> THE CORRECT CLASS NAME : ";NC$(NC):
    V1 = 3:
    GOSUB 1750:
    NC$(NC) = TX$:
    GOTO 560
720 IF PN = NP
     THEN
      730:
     ELSE
      Y = PN + 1
725 IF PN + 8 < NP
     THEN
      PN = PN + 8:
      GOTO 560:
     ELSE
      PN = NP:
      GOTO 560
730 SC = 0
735 SC = SC + 1
740 IF C1 = 0
     THEN
      750:
     ELSE
      770
750 IF NT$(1) = "NO TEACHER"
     THEN
      SN$(C) = SN$(C) + "00":
      GOTO 850
760 SN$(C) = SN$(C) + "01":
    TN$(1) = NT$(1):
    C1 = 1:
    GOTO 850
770 IF NT$(SC) = "NO TEACHER"
     THEN
      SN$(C) = SN$(C) + "00":
      GOTO 850
780 FOR Y = 1 TO C1
790  YA = LEN(NT$(SC))
800  IF NT$(SC) = LEFT$(TN$(Y),YA)
      THEN
       C3 = Y:
       GOTO 830
810  NEXT Y
820 C1 = C1 + 1:
    TN$(C1) = NT$(SC):
    C3 = C1
830 IF C3 < 10
     THEN
      SN$(C) = SN$(C) + "0" + RIGHT$( STR$(C3),1):
      GOTO 850
840 SN$(C) = SN$(C) + RIGHT$( STR$(C3),2):
    GOTO 850
850 IF C2 = 0
     THEN
```

Program continued

```
      860:
     ELSE
      880
860 IF NC$(1) = "NO CLASS"
     THEN
      SN$(C) = SN$(C) + "000":
      GOTO 970
870 SN$(C) = SN$(C) + "001":
    CN$(1) = NC$(1):
    C2 = 1:
    GOTO 970
880 IF NC$(SC) = "NO CLASS"
     THEN
      SN$(C) = SN$(C) + "000":
      GOTO 970
890 FOR Y = 1 TO C2
900  YB = LEN(NC$(SC))
910  IF NC$(SC) = LEFT$(CN$(Y),YB)
      THEN
       C3 = Y:
       GOTO 940
920  NEXT Y
930 C2 = C2 + 1:
    CN$(C2) = NC$(SC):
    C3 = C2
940 IF C3 < 10
     THEN
      SN$(C) = SN$(C) + "00" + RIGHT$( STR$(C3),1):
      GOTO 970
950 IF C3 < 100
       EN
      SN$(C) = SN$(C) + "0" + RIGHT$( STR$(C3),2):
      GOTO 970
960 SN$(C) = SN$(C) + RIGHT$( STR$(C3),3):
    GOTO 970
970 YA = 0:
    YB = 0
975 IF SC < NP
     THEN
      735
980 CLS :
    AN$ = ""
990 FOR X = 1 TO NP:
     NT$(X) = "":
     NC$(X) = "":
     NEXT X
1000 PRINT @448,"";:
    LINE INPUT "DO YOU HAVE ANOTHER STUDENT TO ENTER ? ( Y/N )   ";A
    N$:
    GOSUB 1860
1010 IF AN$ = "Y" AND C = 10
     THEN
      CLS :
      PRINT @448,"PARDON ME FOR A MINUTE WHILE I WRITE SOME DATA. ":
      UR = 1:
      GOTO 1040
1020 IF AN$ = "Y"
     THEN
      250:
      :
     ELSE
      IF AN$ < > "N"
       THEN
        980:
       ELSE
        1030
1030 UR = 2
1040 OPEN "R",1,"STDSCHED"
1050 FOR X = 1 TO C
1060  G = Q * FS
```

```
1070  FIELD 1,(G)ASDUMMY$,(FS)ASP$
1072  IF Q = 0
       THEN
        1080
1075  GET 1,RN
1080  LSET P$ = SN$(X)
1090  SN$(X) = ""
1100  PUT 1,RN
1110  IF UR = 3
       THEN
        UR = 0:
        GOTO 1160
1120  Q = Q + 1:
      IF Q = UF
       THEN
        Q = 0:
        RN = RN + 1
1130  NEXT X
1140 C = 0:
     IF UR = 1
      THEN
       UR = 0:
       CLOSE :
       GOTO 250
1150 IF UR = 2
      THEN
       UR = 3:
       SN$(X) = STRING$(FS,88):
       GOTO 1060
1160 FIELD 1,2ASX1$,2ASX2$,2ASX3$,2ASX4$,2ASX5$,2ASX6$,2ASX7$,2ASX8$,
     2ASX9$,2ASXA$
1170 GET 1,1
1180 LSET X2$ = MKI$ (FS):
     LSET X3$ = MKI$ (UF):
     LSET X4$ = MKI$ (NX):
     LSET X5$ = MKI$ (NY):
     LSET X6$ = MKI$ (NP):
     LSET X7$ = MKI$ (RN):
     LSET X8$ = MKI$ (Q)
1190 PUT 1,1:
     CLOSE
1200 OPEN "R",2,"TEACHER"
1210 RO = 1:
     Q = 0
1220 FOR X = 1 TO C1
1230  G = Q * 25
1240  FIELD 2,(G)ASDUMMY$,25ASB$
1250  LSET B$ = TN$(X)
1260  PUT 2,RO
1270  Q = Q + 1:
      IF Q = 10
       THEN
        Q = 0:
        RO = RO + 1
1280  IF UR = 1
       THEN
        UR = 0:
        GOTO 1310
1290  NEXT X
1300 UR = 1:
     TN$(X) = STRING$(25,88):
     GOTO 1230
1310 OPEN "R",3,"CLASSES"
1320 RP = 1:
     Q = 0
1330 FOR X = 1 TO C2
1340  G = Q * 25
1350  FIELD 3,(G)ASDV$,25ASXC$
1360  LSET XC$ = CN$(X)
1370  PUT 3,RP
```

Program continued

```
1380  IF UR = 1
       THEN
        UR = 0:
        GOTO 1420
1390  Q = Q + 1:
      IF Q = 10
       THEN
        Q = 0:
        RP = RP + 1
1400  NEXT X
1410 UR = 1:
     CN$(X) = STRING$(25,88):
     GOTO 1340
1420 CLOSE
1430 RUN "CLASMENU"
1440 OPEN "R",2,"TEACHER"
1450 X = 0:
     RO = 1:
     Q1 = 0
1460 G = Q1 * 25
1470 FIELD 2,(G)ASDUMMY$,25ASA1$
1480 GET 2,RO
1490 IF A1$ = STRING$(25,88)
      THEN
       1580
1500 X = X + 1
1510 K = LEN(A1$)
1520 FOR Y = 1 TO K
1530  IF MID$(A1$,Y,2) = "  "
       THEN
        TN$(X) = LEFT$(A1$,Y - 1):
        GOTO 1560
1540  NEXT Y
1550 TN$(X) = A1$
1560 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       Q1 = 0:
       RO = RO + 1
1570 GOTO 1460
1580 C1 = X
1590 X = 0:
     Q1 = 0:
     RP = 1
1600 OPEN "R",3,"CLASSES"
1610 G = 25 * Q1
1620 FIELD 3,(G)ASDUMMY$,25ASA2$
1630 GET 3,RP
1640 IF A2$ = STRING$(25,88)
      THEN
       1730
1650 X = X + 1
1660 K = LEN(A2$)
1670 FOR Y = 1 TO K
1680  IF MID$(A2$,Y,2) = "  "
       THEN
        CN$(X) = LEFT$(A2$,Y - 1):
        GOTO 1710
1690  NEXT Y
1700 CN$(X) = A2$
1710 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       Q1 = 0:
       RP = RP + 1
1720 GOTO 1610
1730 C2 = X:
     CLOSE
1740 GOTO 250
1750 TX$ = "":
```

```
     ON V1 GOTO 1760,1770,1780
1760 TX$ = SN$(C):
     GOTO 1790
1770 TX$ = NT$(X):
     GOTO 1790
1780 TX$ = NC$(X):
     GOTO 1790
1790 K = LEN(TX$)
1800 FOR IL = 1 TO K
1810  IF (( MID$(TX$,IL,1) = ",") OR ( MID$(TX$,IL,1) = "."))
      AND MID$(TX$,IL + 1,1) = CHR$(32)
       THEN
        TX$ = LEFT$(TX$,IL - 1) + RIGHT$(TX$,K - IL):
        GOTO 1830
1820  IF ( MID$(TX$,IL,1) = ",") OR ( MID$(TX$,IL,1) = ".")
       THEN
        TX$ = LEFT$(TX$,IL - 1) + " " + RIGHT$(TX$, LEN(TX$)
        - IL)
1830  NEXT IL
1840 RETURN
1850 DATA FIRST,SECOND,THIRD,FOURTH,FIFTH,SIXTH,SEVENTH,EIGHTH,NINTH,
     TENTH,ELEVENTH,TWELFTH,THIRTEENTH,FOURTEENTH,FIFTEENTH,SIXTEENTH
1860 AN$ = LEFT$(AN$,1):
     RETURN
1870 CLS :
     PRINT @394,"AN ERROR HAS OCCURRED IN THE EXECUTION OF THE PROGRA
     MCALLED 'STUDENT SCHEDULE INPUT'."
1880 PRINT TAB(5)"ERROR TYPE = "; ERR / 2 + 1
1890 PRINT TAB(5)"ERROR LINE = "; ERL
1900 FOR V = 1 TO 5000:
      NEXT V
1910 STOP
```

Program Listing 2. *Schedule change program*

```
 10 :
    '  SCHEDULE CHANGE PROGRAM ( STDCHANG )
 20 :
    '  COPYRIGHT OCTOBER 1, 1981
 30 :
    '  ULDERIC F. RACINE
 40 :
    '  2520 S.E. ALEXANDER DRIVE
 50 :
    '  TOPEKA, KANSAS 66605
100 CLS :
    PRINT @458,"I WILL BE WITH YOU IN A MOMENT.":
    CLEAR 5000
110 OPEN "R",1,"STDSCHED":
    RN = LOF (1):
    IF RN = 0
     THEN
      CLOSE :
      KILL "STDSCHED":
      GOTO 130
120 UR = 1:
    GOTO 180
130 CLS :
    AN$ = "":
    PRINT @448,"I HAVE READ THE DISKS CURRENTLY IN THE DRIVES.":
    PRINT "THERE IS NO SCHEDULE DATA ON THESE DISKS.":
    LINE INPUT "DO YOU HAVE A DISK WITH SCHEDULE DATA ? ( Y/N ) ";AN
    $:
    GOSUB 3030:
```

Program continued

```
     IF AN$ = "Y"
      THEN
       150:
      ELSE
       IF AN$ < > "N"
        THEN
         130:
        ELSE
         140
140 RUN "CLASMENU"
150 CLS :
    PRINT @448,"";:
    LINE INPUT "PLEASE PUT THE DISK IN ONE OF THE DRIVES ( 1 - 2 - 3
     ) AND PRESS <ENTER> ";AN$:
    GOTO 110
160 CLEAR T
170 OPEN "R",1,"STDSCHED"
180 FIELD 1, 2ASFA$,2ASFB$,2ASFC$,2ASFD$,2ASFE$,2ASFG$,2ASFH$,2ASFI$
190 GET 1,1:
    FS = CVI (FB$):
    UF = CVI (FC$):
    NP = CVI (FG$):
    T = CVI (FA$)
195 IF PEEK(16424) = 5 AND T = 2
     THEN
      CLOSE :
      UR = 2:
      GOTO 1800
200 OPEN "R",2,"TEACHER":
    OPEN "R",3,"CLASSES"
210 NT = LOF (2) * 10:
    NC = LOF (3) * 10:
    RN = ( LOF (1) - 1) * UF
220 IF UR = 1
     THEN
      CLOSE :
      T = (NC * 15) + (NT * 15) + (RN * FS) + 2000:
      GOTO 160
230 ON ERROR GOTO 2980
240 DIM TN$(NT + 10),CN$(NC + 10),SN$(RN + 20)
250 Q = 0:
    RN = 2:
    X = 0
260 G = Q * FS
270 FIELD 1,(G)ASDUMMY$, (FS)ASA$
280 GET 1,RN
290 IF A$ = STRING$(FS,88)
     THEN
      SN = X:
      GOTO 330
300 X = X + 1:
    SN$(X) = A$
310 Q = Q + 1:
    IF Q = UF
     THEN
      Q = 0:
      RN = RN + 1
320 GOTO 260
330 X = 0:
    RO = 1:
    Q = 0
340 G = Q * 25
350 FIELD 2,(G)ASDUMMY$, 25ASA1$
360 GET 2,RO
370 IF A1$ = STRING$(25,88)
     THEN
      NT = X:
      GOTO 450
380 X = X + 1
390 FOR Y = 1 TO 25
```

```
400  IF MID$(A1$,Y,2) = "  "
      THEN
       TN$(X) = LEFT$(A1$,Y - 1):
       GOTO 430
410  NEXT Y
420 TN$(X) = A1$
430 Q = Q + 1:
    IF Q = 10
     THEN
      Q = 0:
      RO = RO + 1
440 GOTO 340
450 X = 0:
    RN = 1:
    Q = 0
460 G = Q * 25
470 FIELD 3,(G)ASDUMMY$,25ASA2$
480 GET 3,RN
490 IF A2$ = STRING$(25,88)
     THEN
      CN = X:
      GOTO 570
500 X = X + 1
510 FOR Y = 1 TO 25
520  IF MID$(A2$,Y,2) = "  "
      THEN
       CN$(X) = LEFT$(A2$,Y - 1):
       GOTO 550
530  NEXT Y
540 CN$(X) = A2$
550 Q = Q + 1:
    IF Q = 10
     THEN
      Q = 0:
      RN = RN + 1
560 GOTO 460
570 CLOSE :
    TN$(0) = "NO TEACHER":
    CN$(0) = "NO CLASS"
580 W = PEEK(16424):
    POKE 16424,67
590 ON W GOTO 600,1990,2250,2450,1800
600 CLS :
    PRINT "STUDENT SCHEDULE CHANGE PROGRAM":
    PRINT @128,"OPTIONS :":
    PRINT @256,"1 - CHANGE A STUDENT'S SCHEDULE":
    PRINT "2 - ADD A STUDENT TO AN EXISTING CLASS ROSTER":
    PRINT "3 - EXIT THIS PROGRAM"
610 PRINT @512,"";:
    LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
    OP = VAL(OP$):
    IF OP < 1 OR OP > 3
     THEN
      600
620 ON OP GOTO 630,1800,140
630 CLS :
    PRINT "OPTION # 1 - CHANGE A STUDENT'S SCHEDULE"
640 PRINT @448,"<ENTER> THE NAME OF THE STUDENT WHOSESCHEDULE YOU WI
    SH TO CHANGE."
650 PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INTIAL (IF AN
    Y)
660 PRINT :
    LINE INPUT "NAME :";SO$:
    K = LEN(SO$)
670 FOR X = 1 TO SN
680  IF LEFT$(SN$(X),K) = SO$
      THEN
       800
690  NEXT X:
    IF XF = 1
```

Program continued

```
     THEN
      RETURN
 700 CLS :
     PRINT @128,"I CANNOT FIND A STUDENT NAMED ";SO$
 710 PRINT "IN MY STUDENT FILE."
 720 PRINT :
     PRINT "WHAT SHALL I DO NOW ?"
 730 PRINT :
     PRINT "1 - TRY ANOTHER STUDENT"
 740 PRINT "2 - LIST ALL STUDENTS IN THE FILE"
 750 PRINT "3 - EXIT THIS PROGRAM "
 760 PRINT :
     LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
     OP = VAL(OP$)
 770 IF OP < 1 OR OP > 3
      THEN
       700
 780 IF OP = 2
      THEN
       FG = 1
 790 ON OP GOTO 630,2450,140
 800 IF XF = 1
      THEN
       XF = 9:
       RETURN
 810 Y = 25:
     Z = 27:
     IF NP > 10 AND DS = 0
      THEN
       DIM HT(NP),HN(NP):
       DS = 1
 820 CM = 0
 830 FOR X1 = 1 TO NP
 840  HT(X1) = VAL( MID$(SN$(X),Y,2))
 850  HN(X1) = VAL( MID$(SN$(X),Z,3))
 860  Y = Y + 5:
      Z = Z + 5
 870  NEXT X1
 880 CLS
 890 PRINT "STUDENT :"; LEFT$(SN$(X),24)
 900 PRINT "PERIOD " TAB(10)"TEACHER" TAB(35)"CLASSES"
 910 PRINT STRING$(62,45)
 920 TN$(0) = "NO TEACHER":
     CN$(0) = "NO CLASS"
 930 FOR X1 = 1 TO NP
 940  PRINT TAB(3)X1 TAB(10)TN$(HT(X1)) TAB(30)CN$(HN(X1))
 950  IF X1 + 1 = 10 AND NP > 9
       THEN
        PRINT :
        LINE INPUT "PRESS <ENTER> TO CONTINUE ";AN$:
        PRINT @192, CHR$(31);
 960  NEXT X1
 970 IF XF = 2
      THEN
       XF = 3
 980 PRINT @768,"THIS IS THE CURRENT SCHEDULE INFORMATION I HAVEON ";
     SO$
 990 LINE INPUT "IS THIS INFORMATION CORRECT ? ( Y/N ) ";AN$:
     AN$ = LEFT$(AN$,1):
     IF LEFT$(AN$,1) = "Y"
      THEN
       CLS :
       GOTO 2680
1000 IF AN$ < > "N"
      THEN
       880:
      ELSE
       1010
1010 PRINT @768, CHR$(31);:
     AN$ = "":
```

```
     PRINT "WHICH PERIOD DO YOU WISH TO CHANGE ? ( 1 - ";X1 - 1;:
     LINE INPUT ")  ";NC$:
     NC = VAL(NC$)
1020 IF NC < 1 OR NC > X1 - 1
      THEN
       1010
1030 CLS :
     PRINT @448,"THE TEACHER LISTED FOR PERIOD";NC;"IS ";TN$(HT(NC))
1040 AN$ = "":
     LINE INPUT "DO YOU WISH TO CHANGE THE TEACHER'S NAME ( Y/N ) ? "
     ;AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       1050:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1030:
        ELSE
         1440
1050 CLS :
     PRINT @448,"NEW TEACHER'S NAMELAST NAME <SPACE> FIRST NAME <SPAC
     E> MIDDLE INITIAL (IF ANY)"
1060 PRINT "<ENTER> 'NT' IF YOU WISH TO DROP THIS STUDENT FROM THIS C
     LASS."
1070 LINE INPUT "<ENTER> NAME : ";V$
1080 IF V$ = "NT"
      THEN
       X1 = 0:
       GOTO 1430
1090 K = LEN(V$)
1100 FOR X1 = 1 TO NT
1110  IF LEFT$(TN$(X1),K) = V$
       THEN
        1430
1120  NEXT X1
1130 CLS :
     PRINT "I DO NOT FIND A TEACHER NAMED ";V$:
     PRINT "IN MY TEACHER FILE."
1140 FOR X1 = 1 TO 500:
      NEXT X1:
     CLS :
     PRINT "THE TEACHER'S NAMES BEGINNING WITH "; LEFT$(V$,1);" ARE:"
     :
     TL$ = LEFT$(V$,1):
     PRINT
1150 FOR X1 = 1 TO NT
1160  IF LEFT$(TN$(X1),1) = TL$
       THEN
        1170:
       ELSE
        1200
1170  IF UQ = 1
       THEN
        1190
1180  PRINT TN$(X1);:
      UQ = 1:
      GOTO 1200
1190  PRINT TAB(30)TN$(X1):
      UQ = 0
1200  NEXT X1:
     PRINT
1210 AN$ = "":
     PRINT "SHALL I ADD ";V$;"'S":
     LINE INPUT "NAME TO THE FILE ( Y/N ) ? ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
```

Program continued

```
        1280:
        :
       ELSE
        IF AN$ < > "N"
         THEN
          1210:
         ELSE
          1220
1220 IF XF = 2
       THEN
        880:
       ELSE
        CLS :
        PRINT @128,"WHAT SHALL I DO NOW ? ":
        OP = 0
1230 PRINT "1 - TRY AGAIN - DISPLAY ";SO$;"'S SCHEDULE AGAIN"
1240 PRINT "2 - TRY ANOTHER STUDENT"
1250 PRINT "3 - EXIT THIS PROGRAM"
1260 PRINT :
     LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
     OP = VAL(OP$):
     IF OP < 1 OR OP > 3
      THEN
       1220
1270 ON OP GOTO 880,630,140
1280 OPEN "R",2,"TEACHER":
     RA = LOF (2):
     Q1 = 0
1290 G = Q1 * 25
1300 FIELD 2,(G)ASDUMMY$,25ASVT$
1310 GET 2,RA
1320 IF VT$ = STRING$(25,88)
      THEN
       1350
1330 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       CLOSE :
       PRINT "ERROR IN FILE MARKER.":
       STOP
1340 GOTO 1290
1350 FIELD 2,(G)ASDUMMY$,25ASVT$
1360 LSET VT$ = V$
1370 PUT 2,RA
1380 IF UR = 1
      THEN
       1420
1390 NT = NT + 1:
     TN$(NT) = V$
1400 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       Q1 = 0:
       RA = RA + 1
1410 G = Q1 * 25:
     FIELD 2,(G)ASDUMMY$,25ASVT$:
     GET 2,RA:
     V$ = STRING$(25,88):
     UR = 1:
     GOTO 1360
1420 HT(NC) = NT:
     GOTO 1440
1430 HT(NC) = X1:
     CM = 1
1440 CLS :
     PRINT @448,"THE CLASS LISTED FOR PERIOD";NC;"IS ";CN$(HN(NC))
1450 AN$ = "":
     LINE INPUT "DO YOU WISH TO CHANGE THE CLASS NAME ( Y/N ) ? ";AN$
     :
     AN$ = LEFT$(AN$,1):
```

```
     IF AN$ = "Y"
      THEN
       1460:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1440:
        ELSE
         880
1460 CLS :
     PRINT @448,"ENTER 'NC' IS YOU WISH TO DROP THE STUDENT FROM THIS
      CLASS":
     LINE INPUT "<ENTER> NEW CLASS NAME :";V$
1470 IF V$ = "NC"
      THEN
       X1 = 0:
       GOTO 1780
1480 K = LEN(V$)
1490 FOR X1 = 1 TO CN
1500  IF LEFT$(CN$(X1),K) = V$
       THEN
        1780
1510  NEXT X1
1520 CLS :
     PRINT "I CANNOT FIND A CLASS NAMED ";V$;" IN THE FILE."
1530 PRINT "THE CLASSES BEGINNING WITH "; LEFT$(V$,1);" ARE:":
     TL$ = LEFT$(V$,1):
     PRINT
1540 UQ = 0
1550 FOR X1 = 1 TO CN
1560  IF LEFT$(CN$(X1),1) = TL$
       THEN
        1570:
       ELSE
        1600
1570  IF UQ = 1
       THEN
        1590
1580  PRINT CN$(X1);:
      UQ = 1:
      GOTO 1600
1590  PRINT TAB(30)CN$(X1):
      UQ = 0
1600  NEXT X1
1610 PRINT :
     AN$ = "":
     PRINT "SHALL I ADD ";V$:
     LINE INPUT "TO THE CLASS FILE ( Y/N ) ? ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       1620:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1610:
        ELSE
         1220
1620 OPEN "R",3,"CLASSES":
     RA = LOF (3):
     Q1 = 0:
     UR = 0:
     Y1 = 0
1630 G = Q1 * 25
1640 FIELD 3,(G)ASDUMMY$,25ASVT$
1650 GET 3,RA
1660 IF VT$ = STRING$(25,88)
```

Program continued

```
      THEN
       1700
1670 Y1 = Y1 + 1
1680 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       CLOSE :
       PRINT "END OF FILE MARKER ERROR.":
       STOP
1690 GOTO 1630
1700 FIELD 3,(G)ASDUMMY$,25ASVT$
1710 LSET VT$ = V$
1720 PUT 3,RA
1730 IF UR = 1
      THEN
       1760
1740 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       Q1 = 0:
       RA = RA + 1
1750 G = Q1 * 25:
     FIELD 3,(G)ASDUMMY$,25ASVT$:
     GET 3,RA:
     CN = CN + 1:
     CN$(CN) = V$:
     HN(NC) = CN:
     V$ = STRING$(25,88):
     UR = 1:
     GOTO 1710
1760 CLOSE
1770 GOTO 880
1780 HN(NC) = X1:
     CM = 1:
     IF XF = 2
      THEN
       V = 0:
       G = G + 1:
       IF G > NP
        THEN
         XF = 3
1790 GOTO 880
1800 IF T = 1
      THEN
       1830:
      ELSE
       CLS :
       PRINT @448,"THE FILE INDICATES THAT THE SCHEDULEDATA WAS NOT E
       NTERED BY TEACHER.IF YOU WISH TO ADD A STUDENT YOU MUST USE OP
       TION # 1 ON THE MASTER MENU."
1810 PRINT :
     LINE INPUT "SHALL I RUN THAT PROGRAM FOR YOU ( Y/N ) ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       1820:
      ELSE
       IF AN$ < > "N"
        THEN
         1810
1815 IF UR = 2
      THEN
       RUN "CLASMENU":
      ELSE
       600
1820 RUN "STDSCHD"
1830 CLS :
     PRINT "OPTION # 2 - ADD A NEW STUDENT TO AN EXISTING CLASS ROSTE
     R":
     PRINT @448,"<ENTER> THE NAME OF THE STUDENT YOU WISH TO ADD"
```

```
1840 PRINT "LAST NAME <SPACE> FIRST NAME <SPACE> MIDDLE INITIAL (IF A
     NY)"
1850 LINE INPUT "<ENTER> NAME : ";SO$:
     V = 1:
     GOSUB 1890
1860 XF = 1:
     K = LEN(SO$):
     GOSUB 670:
     IF XF = 9
      THEN
       1880
1870 XF = 2:
     SN$(X) = SO$ + STRING$(24 - K,32) + STRING$(FS - 24,"0"):
     G = 1:
     GOTO 810
1880 CLS :
     PRINT @448,SO$;" IS CURRENTLY ON THE FILE.":
     FOR Z = 1 TO 300:
      NEXT Z:
     XF = 0:
     GOTO 810
1890 ON V GOTO 1900,1910
1900 TS$ = SO$:
     GOTO 1920
1910 TS$ = V$
1920 K = LEN(TS$)
1930 FOR Y = 1 TO K
1940  IF (( MID$(TS$,Y,1) = ",") OR ( MID$(TS$,Y,1) = ".")) AND
      MID$(TS$,Y + 1,1) = " "
       THEN
        TS$ = LEFT$(TS$,Y - 1) + RIGHT$(TS$,K - Y):
        GOTO 1960
1950  IF (( MID$(TS$,Y,1) = ",") OR ( MID$(TS$,Y,1) = "."))
       THEN
        TS$ = LEFT$(TS$,Y - 1) + " " + RIGHT$(TS$,K - Y)
1960  NEXT Y
1970 IF V = 1
      THEN
       SO$ = TS$:
      ELSE
       V$ = TS$
1980 RETURN
1990 CLS :
     PRINT "LISTING OF TEACHERS CURRENTLY ON FILE":
     PRINT
2000 PW$ = "     ":
     PX$ = "###% %%" + STRING$(22,32) + "%":
     PY$ = " - ":
     PZ$ = PX$ + "%     %" + PX$:
     IF UK = 1
      THEN
       RETURN
2010 UR = 0:
     GOSUB 2220:
     UQ = 0
2020 IF UR = 0
      THEN
       G = 1
2030 IF UR = 1
      THEN
       LPRINT "LISTING OF TEACHERS CURRENTLY ON FILE":
       LPRINT STRING$(66,45)
2040 FOR X1 = 1 TO NT
2050  IF UQ = 0
       THEN
        2060:
       ELSE
        2070
2060  PRINT USING PX$;X1,PY$,TN$(X1);:
      UQ = 1:
```

Program continued

```
     GOTO 2090
2070 PRINT TAB(30) USING PX$;X1,PY$,TN$(X1):
     UQ = 0
2080 IF UR = 1
      THEN
       LPRINT USING PZ$;X1 - 1,PY$,TN$(X1 - 1),PW$,X1,PY$,TN$(X1)
2090 IF UR = 0 AND X1 = G * 22
      THEN
       PRINT "";:
       LINE INPUT "PRESS <ENTER> TO CONTINUE";AN$:
       PRINT @64, CHR$(31);:
       G = G + 1
2100 NEXT X1
2110 IF UQ = 1 AND UR = 1
      THEN
       LPRINT USING PX$;X1 - 1,PY$,TN$(X1 - 1):
       UQ = 0
2120 UR = 0
2130 PRINT :
     LINE INPUT "PRESS <ENTER> TO CONTINUE ";AL$
2140 CLS
2150 PRINT @128,"WHAT SHALL I DO NOW ?"
2160 PRINT :
     PRINT "1 - LISTING OF STUDENTS"
2170 PRINT "2 - LISTING OF CLASSES"
2180 PRINT "3 - EXIT THIS PROGRAM AND RETURN TO MASTER MENU"
2190 PRINT :
     LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
     OP = VAL(OP$)
2200 IF OP < 1 OR OP > 3
      THEN
       2140
2210 ON OP GOTO 2450,2250,140
2220 AN$ = "":
     LINE INPUT "DO YOU WANT A PRINTED LISTING ( Y/N ) ? ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       UR = 1:
       GOTO 2230:
      ELSE
       IF AN$ < > "N"
        THEN
         2220:
        ELSE
         PRINT @64, CHR$(31);:
         RETURN
2230 PRINT :
     AN$ = "":
     LINE INPUT "SHALL I GENERATE A TEST LINE FOR THE PRINTER ( Y/N )
      ? ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       2240:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         2230:
        ELSE
         PRINT @64, CHR$(31);:
         RETURN
2240 LPRINT "THIS IS A TEST LINE-------------------------------------
     -----":
     GOTO 2230
2250 CLS :
     PRINT "LISTING OF CLASSES CURRENTLY ON FILE":
     PRINT
2260 UR = 0:
```

```
     GOSUB 2220:
     UQ = 0:
     IF UR = 0
      THEN
       G = 1
2270 UK = 1:
     GOSUB 2000:
     UK = 0
2280 IF UR = 1
      THEN
       LPRINT "LISTING OF CLASSES CURRENTLY ON FILE":
       LPRINT STRING$(66,45)
2290 FOR X1 = 1 TO CN
2300  IF UQ = 0
       THEN
        2310:
       ELSE
        2320
2310  PRINT USING PX$;X1,PY$,CN$(X1);:
      UQ = 1:
      GOTO 2340
2320  PRINT TAB(30) USING PX$;X1,PY$,CN$(X1):
      UQ = 0
2330  IF UR = 1
       THEN
        LPRINT USING PZ$;X1 - 1,PY$,CN$(X1 - 1),PW$,X1,PY$,CN$(X1)
2340  IF UR = 0 AND X1 = G * 22
       THEN
        PRINT "";:
        LINE INPUT "PRESS <ENTER> TO CONTINUE ";AN$:
        PRINT @64, CHR$(31);:
        G = G + 1
2350  NEXT X1
2360 IF UQ = 1 AND UR = 1
      THEN
       LPRINT USING PX$;X1 - 1,PY$,CN$(X1 - 1):
       UQ = 0:
       UR = 0
2370 LINE INPUT "PRESS <ENTER> TO CONTINUE";AN$:
     CLS
2380 PRINT @128,"WHAT SHALL I DO NOW ?"
2390 PRINT :
     PRINT "1 - LISTING OF TEACHERS"
2400 PRINT "2 - LISTING OF STUDENTS"
2410 PRINT "3 - EXIT THIS PROGRAM AND RETURN TO MASTER MENU"
2420 PRINT :
     LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
     OP = VAL(OP$)
2430 IF OP < 1 OR OP > 3
      THEN
       CLS :
       GOTO 2380
2440 ON OP GOTO 1990,2450,140
2450 CLS :
     PRINT "LISTING OF STUDENTS CURRENTLY ON FILE":
     PRINT
2460 UR = 0:
     GOSUB 2220:
     UQ = 0:
     OP = 0:
     IF UR = 0
      THEN
       G = 1
2470 UK = 1:
     GOSUB 2000:
     UK = 0
2480 IF UR = 1
      THEN
       LPRINT "LISTING OF STUDENTS CURRENTLY ON FILE":
       LPRINT STRING$(66,45)
```

Program continued

```
2490 FOR X1 = 1 TO SN
2500 IF UQ = 0
      THEN
       2510:
      ELSE
       2520
2510 PRINT USING PX$;X1,PY$, LEFT$(SN$(X1),24);:
     UQ = 1:
     GOTO 2540
2520 PRINT TAB(30) USING PX$;X1,PY$, LEFT$(SN$(X1),24):
     UQ = 0
2530 IF UR = 1
      THEN
       LPRINT USING PZ$;X1 - 1,PY$, LEFT$(SN$(X1 - 1),24),PW$,X1,PY$
       , LEFT$(SN$(X1),24)
2540 IF UR = 0 AND X1 = G * 22
      THEN
       PRINT "";:
       LINE INPUT "PRESS <ENTER> TO CONTINUE ";AN$:
       PRINT @64, CHR$(31);:
       G = G + 1
2550 NEXT X1
2560 IF UQ = 1 AND UR = 1
      THEN
       LPRINT USING PX$;X1 - 1,PY$, LEFT$(SN$(X1 - 1),24):
       UQ = 0
2570 UR = 0:
     PRINT
2580 LINE INPUT "PRESS <ENTER> TO CONTINUE ";AN$
2590 CLS
2600 IF FG = 1
      THEN
       FG = 0:
       GOTO 720
2610 PRINT @128,"WHAT SHALL I DO NOW ?"
2620 PRINT :
     PRINT "1 - LISTING OF TEACHERS"
2630 PRINT "2 - LISTING OF CLASSES"
2640 PRINT "3 - EXIT THIS PROGRAM AND RETURN TO MASTER MENU"
2650 PRINT :
     LINE INPUT "<ENTER> OPTION NUMBER ";OP$:
     OP = VAL(OP$)
2660 IF OP < 1 OR OP > 3
      THEN
       2590
2670 ON OP GOTO 1990,2250,140
2680 IF CM = 1
      THEN
       2700
2690 GOTO 600
2700 CLS :
     AN$ = ""
2710 IF XF = 3
      THEN
       PRINT @448,"SHALL I FILE ";SO$;"'S":
       GOTO 2730
2720 PRINT @448,"SHALL I FILE THE CHANGES MADE TO ";SO$;"'S"
2730 LINE INPUT "SCHEDULE ? ( Y/N )  ";AN$:
     AN$ = LEFT$(AN$,1):
     IF AN$ = "Y"
      THEN
       2740:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         2700:
        ELSE
         CM = 0:
         GOTO 2680
```

```
2740 SI$ = ""
2750 FOR Y = 1 TO NP
2760  IF HT(Y) < 10
       THEN
        SI$ = SI$ + "0" + RIGHT$( STR$(HT(Y)),1):
        GOTO 2780
2770  SI$ = SI$ + RIGHT$( STR$(HT(Y)),2)
2780  HT(Y) = 0
2790  IF HN(Y) < 10
       THEN
        SI$ = SI$ + "00" + RIGHT$( STR$(HN(Y)),1):
        GOTO 2820
2800  IF HN(Y) < 100
       THEN
        SI$ = SI$ + "0" + RIGHT$( STR$(HN(Y)),2):
        GOTO 2820
2810  SI$ = SI$ + RIGHT$( STR$(HN(Y)),3)
2820  HN(Y) = 0
2830  NEXT Y
2840 SN$(X) = LEFT$(SN$(X),24) + SI$
2850 FR = X / UF:
     IF FR - INT(FR) = 0
      THEN
       RN = FR + 1:
       Q = UF - 1:
       GOTO 2880
2860 IF FR < 1
      THEN
       RN = 2:
       Q = X - 1:
       GOTO 2880
2870 Q = X - ( INT(FR) * UF) - 1:
     RN = INT(FR) + 2
2880 OPEN "R",1,"STDSCHED"
2890 G = Q * FS
2900 FIELD 1,(G)ASDUMMY$,(FS)AS C1$
2910 GET 1,RN
2920 LSET C1$ = SN$(X)
2930 PUT 1,RN
2940 IF XF = 3
      THEN
       2950:
      ELSE
       2970
2950 Q = Q + 1:
     IF Q = UF
      THEN
       RN = RN + 1:
       Q = 0
2960 X = X + 1:
     SN$(X) = STRING$(FS,88):
     XF = 0:
     GOTO 2920
2970 CLOSE :
     XF = 0:
     CM = 0:
     SN = SN + 1:
     GOTO 600
2980 CLS :
     PRINT @394,"AN ERROR HAS OCCURRED IN THE EXECUTION OF THE PROGRA
     MCALLED 'SCHEDULE CHANGE'."
2990 PRINT TAB(5)"ERROR TYPE = "; ERR / 2 + 1
3000 PRINT TAB(5)"ERROR LINE = "; ERL
3010 FOR V = 1 TO 5000:
      NEXT V
3020 STOP
3030 AN$ = LEFT$(AN$,1):
     RETURN
```

Program Listing 3. *Printout of class roster by teacher*

```
 10 :
    '  PRINTOUT CLASS ROSTER ( PNTTEACH )
 20 :
    '  COPYRIGHT OCTOBER 1, 1981
 30 :
    '  ULDERIC F. RACINE
 40 :
    '  2520 S.E. ALEXANDER DRIVE
 50 :
    '  TOPEKA, KANSAS 66605
100 CLS
110 PRINT CHR$(23)
120 PRINT TAB(6)"  TEACHER PRINT":
    PRINT
130 PRINT TAB(4)"BY STUDENT AND CLASS"
140 OPEN "R",1,"STDSCHED"
150 FIELD 1,2ASX1$,2ASX2$,2ASX3$,2ASX4$,2ASX5$,2ASX6$,2ASX7$,2ASX8$
160 GET 1,1
170 T = CVI (X1$):
    FS = CVI (X2$):
    UF = CVI (X3$):
    NX = CVI (X4$):
    NY = CVI (X5$):
    NP = CVI (X6$):
    RN = CVI (X7$):
    Q = CVI (X8$)
180 X = LOF (1) * UF
190 OPEN "R",2,"CLASSES":
    RO = LOF (2) * 10
200 IF UR = 1
     THEN
      250
210 T = (X * FS) + (RO * 25) + 3000
220 CLOSE
230 CLEAR T
240 UR = 1:
    GOTO 140
250 UR = 0
260 ON ERROR GOTO 1300
270 DIM SN$(X),CN$(RO),CS(35),PC(NP)
280 N1 = 0:
    N2 = 0:
    N3 = 0:
    Q = 0:
    RN = 2:
    Q1 = 0:
    RP = 1:
    RO = 1:
    Q2 = 0:
    X = 0:
    SC = 0
290 G = Q * FS
300 FIELD 1,(G)ASDU$,(FS)ASDA$
310 GET 1,RN
320 IF DA$ = STRING$(FS,88)
     THEN
      CLOSE :
      GOTO 370
330 N2 = N2 + 1
340 SN$(N2) = DA$
350 Q = Q + 1:
    IF Q = UF
     THEN
      Q = 0:
      RN = RN + 1
360 GOTO 290
370 OPEN "R",2,"CLASSES"
380 G = Q1 * 25
```

```
390 FIELD 2,(G)ASDV$,25ASDB$
400 GET 2,RO
410 IF DB$ = STRING$(25,88)
     THEN
      CLOSE :
      GOTO 880
420 N1 = N1 + 1
430 CN$(N1) = DB$
440 Q1 = Q1 + 1:
    IF Q1 = 10
     THEN
      Q1 = 0:
      RO = RO + 1
450 GOTO 380
460 OPEN "R",3,"TEACHER"
470 G = Q2 * 25
480 FIELD 3,(G)ASDV$,25ASDC$
490 GET 3,RP
500 IF DC$ = STRING$(25,88)
     THEN
      CLOSE :
      GOTO 880
510 TN$ = DC$
520 Q2 = Q2 + 1:
    IF Q2 = 10
     THEN
      Q2 = 0:
      RP = RP + 1
530 CLOSE
540 N3 = N3 + 1
550 CP = 0:
    SC = 0:
    CN = 0
560 IF UR = 0
     THEN
      590
570 IF UR = 1 AND PC(CP + 1) = 1
     THEN
      600
580 CP = CP + 1:
    Y = Y + 5:
    Z = Z + 5:
    IF CP + 1 < = NP
     THEN
      570:
     ELSE
      850
590 Y = 25:
    Z = 27
600 FOR X = 1 TO N2
610  IF VAL( MID$(SN$(X),Y,2)) = N3
      THEN
       620:
      ELSE
       640
620  SC = SC + 1:
     CS(SC) = X
630  IF SC = 1
      THEN
       CN = VAL( MID$(SN$(X),Z,3))
640  NEXT X
650 CLS
660 P$ = STRING$(60,45)
670 PRINT P$:
    PRINT "TEACHER : ";TN$:
    IF SC = 0
     THEN
      PRINT "PERIOD : ";CP + 1; TAB(20)"CLASS : NO CLASS":
      PRINT :
      PRINT
```

Program continued

```
680 IF HC = 1
     THEN
      690:
      :
     ELSE
      IF SC = 0
       THEN
        810:
       ELSE
        720
690 LPRINT P$
700 LPRINT "TEACHER : ";TN$
710 IF SC = 0
     THEN
      LPRINT "PERIOD : ";CP + 1; TAB(20)"CLASS : NO CLASS":
      PRINT P$:
      GOTO 810
720 PRINT "PERIOD : ";CP + 1; TAB( 20)"CLASS : ";CN$(CN):
    PRINT P$:
    IF HC = 0
     THEN
      760
730 LPRINT "PERIOD : ";CP + 1; TAB(20)"CLASS : ";CN$(CN)
740 LPRINT P$
750 CX = 0
760 FOR X = 1 TO SC
770  PRINT LEFT$(SN$(CS(X)),24):
     IF HC = 1
      THEN
       780:
      ELSE
       790
780  LPRINT LEFT$(SN$(CS(X)),24)
790  CS(X) = 0:
     CX = CX + 1:
     IF PX = 0 AND CX = 10
      THEN
       LINE INPUT "PRESS <ENTER> TO CONTINUE ";AN$:
       PRINT @256, CHR$(31);:
       CX = 0
800  NEXT X:
    CX = 0
810 PRINT P$:
    IF HC = 1
     THEN
      820:
     ELSE
      830
820 LPRINT P$:
    LPRINT " ":
    LPRINT " "
830 PRINT :
    IF PX = 1
     THEN
      840:
     ELSE
      LINE INPUT "PRESS <ENTER> TO CONTINUE ";A$
840 IF UR = 1 AND CP + 1 < NP
     THEN
      CP = CP + 1:
      Y = Y + 5:
      Z = Z + 5:
      SC = 0:
      GOTO 570
850 IF UR = 1
     THEN
      UR = 0:
      GOTO 1170
860 IF CP + 1 < NP
     THEN
```

```
      CP = CP + 1:
      Y = Y + 5:
      Z = Z + 5:
      CN = 0:
      SC = 0:
      GOTO 600
 870 GOTO 460
 880 CLS
 890 RO = 1:
     RP = 1:
     Q2 = 0:
     Q1 = 0:
     RN = 1:
     Q = 0:
     N3 = 0:
     SC = 0
 900 PRINT "TEACHER PRINT"
 910 PRINT @128,"OPTIONS : "
 920 PRINT @256,"1 - PRINT CLASS ROSTERS FOR ALL TEACHERS"
 930 PRINT "2 - PRINT CLASS ROSTER FOR A SPECIFIC TEACHER"
 940 PRINT "3 - EXIT THIS PROGRAM"
 950 PRINT :
     PRINT
 960 LINE INPUT "<ENTER> OPTION SELECTED : ";OP$:
     OP = VAL(OP$):
     IF OP < 1 OR OP > 3
      THEN
       880
 970 IF OP = 3
      THEN
       990
 980 PX = 0:
     HC = 0:
     GOSUB 1210
 990 ON OP GOTO 460,1000,1200
1000 CLS :
     AN$ = ""
1010 PRINT @448,"";:
     LINE INPUT "<ENTER> TEACHER'S NAME : ";TN$:
     K = LEN(TN$)
1020 INPUT "DO YOU WANT A PRINT FOR ALL PERIODS ";AN$:
     GOSUB 1270:
     IF AN$ = "Y"
      THEN
       FOR X = 1 TO NP:
        PC(X) = 1:
        NEXT X:
       GOTO 1080
1030 IF AN$ < > "N"
      THEN
       CLS :
       AN$ = "":
       PRINT @448,"";:
       GOTO 1020
1040 FOR X = 1 TO NP
1050  PRINT "DO YOU WANT A PRINTOUT OF PERIOD";X;:
      LINE INPUT "? ( Y/N ) ";AN$:
      GOSUB 1270:
      IF AN$ = "Y"
       THEN
        PC(X) = 1:
        GOTO 1070
1060  IF AN$ < > "N"
       THEN
        1050:
       ELSE
        PC(X) = 0
1070  AN$ = "":
      NEXT X
1080 OPEN "R",3,"TEACHER":
```

Program continued

```
     Q1 = 0:
     RO = 1:
     N3 = 0:
     SC = 0:
     CN = 0
1090 G = Q1 * 25
1100 FIELD 3,(G)ASDY$,25ASDA$
1110 GET 3,RO
1120 IF DA$ = STRING$(25,88)
      THEN
       CLOSE :
       GOTO 1280
1130 N3 = N3 + 1
1140 IF LEFT$(DA$,K) = TN$
      THEN
       CLOSE :
       UR = 1:
       CP = 0:
       Y = 25:
       Z = 27:
       GOTO 570
1150 Q1 = Q1 + 1:
     IF Q1 = 10
      THEN
       Q1 = 0:
       RO = RO + 1
1160 GOTO 1090
1170 AN$ = "":
     CLS
1180 PRINT @448,"";:
     LINE INPUT "DO YOU WANT TO PRINT ANOTHER TEACHER'S ROSTER ? ( Y/
     N ) ";AN$
1190 GOSUB 1270:
     IF AN$ = "Y"
      THEN
       1000:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1170:
        ELSE
         880
1200 RUN "CLASMENU"
1210 CLS :
     PRINT @448,"";:
     LINE INPUT "DO YOU WANT A HARDCOPY ? ( Y/N ) ";AN$:
     GOSUB 1270:
     IF AN$ = "Y"
      THEN
       1220:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1210:
        ELSE
         RETURN
1220 HC = 1
1230 PRINT @448, CHR$(31);:
     LINE INPUT "SHALL I GENERATE A TEST LINE FOR THE PRINTER ? ( Y/N
      ) ";AN$:
     GOSUB 1270:
     IF AN$ = "Y"
      THEN
       1240:
       :
      ELSE
       IF AN$ < > "N"
        THEN
         1230:
```

```
        ELSE
         1250
1240 LPRINT STRING$(60,88):
     GOTO 1230
1250 IF OP = 1
      THEN
       PRINT @448, CHR$(31);:
       LINE INPUT "SHALL I STOP BETWEEN PRINTING CLASS PERIODS ? ( Y/
       N ) ";AN$:
       GOSUB 1270:
       IF AN$ = "N"
        THEN
         1260:
         :
        ELSE
         IF AN$ < > "Y"
          THEN
           1250:
          ELSE
           RETURN
1260 PX = 1:
     RETURN
1270 AN$ = LEFT$(AN$,1):
     RETURN
1280 CLS :
     PRINT @448,"I CANNOT FIND A TEACHER NAMED ";TN$
1290 FOR Q = 1 TO 1500:
      NEXT Q:
     GOTO 880
1300 CLS :
     PRINT @394,"AN ERROR HAS OCCURRED IN THE EXECUTION OF THE 'CLASS
      ROSTER PRINTOUT'."
1310 PRINT TAB(5)"ERROR TYPE = "; ERR / 2 + 1
1320 PRINT TAB(5)"ERROR LINE = "; ERL
1330 FOR V = 1 TO 5000:
      NEXT V
1340 STOP
```

GAMES

Space Mission
Slot Machine

GAMES

Space Mission

by Ron Goodman

A fleet of alien ships appears behind you. It seems they can't fire at you, but they can go much faster than you can. If you let them pass you, they will fly to your home planet to steal and kill and then run off into endless space. Suddenly one of the alien ships appears on your spaceship's viewing screen. As it moves away it gets smaller. You must move your laser sight quickly to point at the ship and shoot. You got it this time, but there's another and another. Can you protect the people of your planet? Their lives depend on you.

Playing the Game

In Space Mission, you control a laser sight which you must aim at the alien ships that are quickly flying away from you. The faster you hit an alien, the more points you get. The level of difficulty you choose before the game begins determines the speed of the alien ships. Level 1 is the slowest speed, and level 4 is the fastest. You begin the game with 200 time units. Each shot you take at the aliens uses up five time units. Time units are constantly being used up, and when you run out of time units, the game is over. The four arrow keys control the laser sight. If you press the left arrow, the alien moves right. Since the laser sight never leaves the center of the screen, and the arrows are used to move the sight, right is the direction the alien should go. The alien will occasionally move up, down, to the left, or to the right as well.

How the Program Works

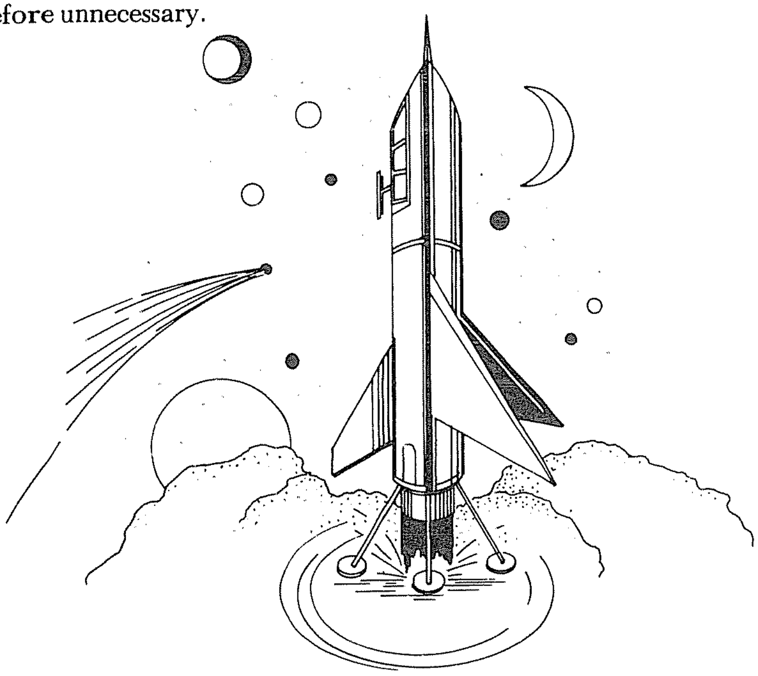
Line 40 of the Program Listing sets up nine variables. These are commonly used variables, and by placing them in the beginning of the simple variable list in program memory, the computer doesn't have to look very far to find them, thus speeding up execution.

The subroutine in lines 550 through 560 loads up the A$ array with text strings (also called packed strings or compressed graphics). Text strings are generally used for only one line of graphics, but you can put more than one line in a single string. In this program, an entire spaceship is displayed with a single PRINT statement; whereas the big spaceship normally would require three PRINT statements. A CHR$(26) shifts down to the next line, and an appropriate number of CHR$(24)s back space to the start of the next line. This may seem like a waste of memory, but it eliminates a lot of PRINT statements, as well as the confusion that goes along with printing a text string that is separated into individual lines.

Lines 290 through 330 sense if any arrows are being pressed. You can't use the INPUT statement because it would require the computer to stop the game to see which direction you want to go. Using INKEY$ would necessitate pressing the arrow key over and over. The answer seemed to be in the PEEK command. The only problem with using PEEK is that when more than one key at a time is pressed, it can get hairy. You can use the AND statement to check individual bits of a byte. This lets you press more than one key at a time, adding the ability to move the sight diagonally. The AND statement is the easiest way to simulate the assembly-language command BIT.

The routine from line 360 to line 530 causes a laser to come from the four corners of the screen when you press the space bar. The laser warps to the center of the screen and then checks to see if it hit an alien. I used a checksum method to add together the contents of the memory locations on the screen that store the picture of the laser sight. Their total is normally 2018. If any part of the alien ship is in your sight, the checksum will equal something other than 2018, and the computer will assume that you hit the alien.

Lines 620 through 810 provide an alternative to the common, boring end of game statement, DO YOU WANT TO PLAY AGAIN (Y/N). Though the program is already a short 5K, you can leave out the REM statements to save space and typing time. They are never called or referred to, and are therefore unnecessary.



Program Listing. *Space Mission*

Encyclopedia Loader

```
 10 REM ***  WRITTEN BY RON GOODMAN
 20 CLS :
    PRINT @469,"<< INITIALIZING >>":
    CLEAR 300
 30 FOR X = 0 TO 8:
     READ SC(X):
     NEXT :
    E = 0:
    GOSUB 550:
    DEFINT A,X,T
 40 DIM A,TZ,ST,X,Y,SC,SI,SA,SP
 50 R1$ = CHR$(191) + CHR$(128) + CHR$(188) + STRING$(4,191)
    + CHR$(188) + CHR$(128) + CHR$(191)
 60 R2$ = CHR$(191) + CHR$(128) + CHR$(143) + STRING$(4,191)
    + CHR$(143) + CHR$(128) + CHR$(191)
 70 CLS :
    PRINT CHR$(23):
    FOR X = 2 TO 476 STEP 66:
     PRINT @X,"SPACE";
 80  PRINT @1008 - X,"MISSION";:
     FOR T = 1 TO 150:
      NEXT T:
     PRINT @X,"     ";
 90  PRINT @1008 - X,"       ";:
     NEXT X:
    PRINT @X,"SPACE MISSION";
100 FOR T = 1 TO 150:
     X = RND(1024) + 15359
110  IF PEEK(X) = 32
      THEN
       POKE X,129:
       NEXT :
     ELSE
      NEXT
120 PRINT @832," DO YOU WANT DIRECTIONS (Y/N)";:
    D = 20:
    C = 2:
    B = 1000
130 A = 980:
    VV = 964:
    A$ = INKEY$
140 FOR X = A TO B STEP C:
     PRINT @X,"*";:
     A$ = INKEY$:
     IF A$ = "Y"
      THEN
       830
150  IF A$ < > "N"
      THEN
       PRINT @X," ";:
       NEXT :
      C = - C:
      A = A + D:
      B = B - D:
      D = - D:
      GOTO 140:
     ELSE
      160
160 PRINT @VV,"ENTER DESIRED LEVEL (1-4)";:
    A$ = INKEY$
170 A$ = INKEY$:
    A = VAL(A$):
    IF A < 1 OR A > 4
     THEN
      170:
     ELSE
      SI = A / 5:
      ST = 0:
```

Program continued

```
      SC = 0
180 PRINT @VV,"PRESS ENTER TO START GAME";
190 IF INKEY$ < > CHR$(13)
     THEN
      190:
     ELSE
      CLS :
      SP = ( RND(12) + 1) * 64 + RND(43) + 8
200 FOR TZ = 200 TO 0 STEP - 1:
     ST = ST + SI
210  REM *** PRINT LATEST SCORE AND TIME LEFT AND CLEAR LOWER
               PORTION OF SCREEN
220  PRINT @0,"SCORE";SC; STRING$(26,32);"TIME";TZ; CHR$(31);
230  REM *** REDRAW SHIP AT NEW POSITION, AND REDRAW SIGHT
240  GOSUB 500:
     IF F = 0 AND FF = 0
      THEN
       PRINT @SP,A$(9 - ST);
250  A$ = INKEY$:
     IF A$ = " "
      THEN
       360
260  IF ST - INT(ST) < .1
      THEN
       SA = (( RND(3) - 2) * 64) + RND(7) - 4:
       IF SA + SP > 0 AND SA + SP < 895
        THEN
         SP = SP + SA
270  IF ST > 8.8
      THEN
       ST = 0:
       SP = ( RND(12) + 1) * 64 + RND(43) + 8:
       F = 0:
       FF = 0:
       CLS
280  REM *** DOES PLAYER WANT TO MOVE SIGHT ???
290  A = PEEK(14400)
300  IF (A AND 8) = 8 AND F = 0
      THEN
       SP = SP + 64:
       F = 0:
       FF = 0:
       IF SP > 895
        THEN
         F = 1
310  IF (A AND 16) = 16 AND FF = 0
      THEN
       SP = SP - 64:
       F = 0:
       FF = 0:
       IF SP < 0
        THEN
         FF = 1
320  IF (A AND 64) = 64 AND SP - INT(SP / 64) * 64 > 2
      THEN
       SP = SP - 2
330  IF (A AND 32) = 32 AND SP - INT(SP / 64) * 64 < 53
      THEN
       SP = SP + 2
340  NEXT :
    GOTO 620
350 REM *** ROUTINE TO SHOOT LASER WHERE SIGHT IS POINTING
360 Y = 0:
    X = 0:
    I = 0:
    TZ = TZ - 5
370 SET(X,Y):
    SET(X,47 - Y):
    SET(127 - X,Y):
    SET(127 - X,47 - Y):
    RESET(X,Y)
```

```
380 RESET(X,47 - Y):
    RESET(127 - X,Y):
    RESET(127 - X,47 - Y):
    X = X + 6
390 IF X < 64
     THEN
      Y = Y + 2.14:
      GOTO 370:
     ELSE
      X = 15836
400 I = I + PEEK(X) + PEEK(X + 64):
    X = X + 1:
    IF X < 15844
     THEN
      400
410 IF I < > 2018
     THEN
      440:
     ELSE
      SP = ( RND(12) + 1) * 64 + RND(43) + 8:
      ST = 0:
      GOTO 270
420 REM *** DATA FOR SCORES FOR DIFFERENT SIZE SHIPS
430 DATA 200,100,80,70,60,50,40,30,10
440 SC = SC + SC(ST):
    ST = 0:
    FOR D = 1 TO 3:
     PRINT @411, STRING$(10,191);
450  PRINT @475,R1$;:
     PRINT @539,R2$;:
     PRINT @603, STRING$(10,191);
460  PRINT @411, STRING$(10,128);:
     PRINT @475, STRING$(10,128);
470  PRINT @539, STRING$(10,128);:
     PRINT @603, STRING$(10,128);
480  GOSUB 500:
     NEXT D:
    ST = 9:
    GOTO 270
490 REM *** DRAW SIGHT IN CENTER OF SCREEN
500 PRINT @476, CHR$(191); CHR$(131);
510 PRINT @482, CHR$(131); CHR$(191);
520 PRINT @540, CHR$(191); CHR$(176);
530 PRINT @546, CHR$(176); CHR$(191);:
    RETURN
540 REM *** LINES 480 & 490 LOAD GRAPHIC STRINGS IN THE A$ ARRAY
550 READ A:
    IF A > 5
     THEN
      A$(E) = A$(E) + CHR$(A):
      GOTO 550
560 IF A = 0
     THEN
      E = E + 1:
      GOTO 550:
     ELSE
      RETURN
570 REM *** DATA FOR ALIEN SPACESHIP
580 DATA 131,0,143,0,170,174,0,189,169,149,26,24,24,24,129,129,129,0
    ,181,166,164,149,26,24,24,24,24,133,137,129,133,0,181,166,153,18
    6,26,24,24,24,24,151,166,153,171,0,151,152,179,164,171,26,24,24,
    24,24,24,151,164,143,152,171,26,24,24,24,24,24,131,128,131
590 DATA 128,131,0,149,176,140,179,140,176,170,26,24,24,24,24,24,24,
    24,159,176,138,191,133,176,175,26,24,24,24,24,24,24,24,133,128,1
    31,140,131,128,138,0,149,176,140,179,179,140,176,170,26,24,24,24
    ,24,24,24,24,24,191,128,191,191,191,191,128,191,26
600 DATA 24,24,24,24,24,24,24,24,149,131,140,179,179,140,131,170,1
610 REM *** AT END OF GAME. SHOW HIGH SCORE, SCORE AND ASK IF YOU
            WANT TO PLAY AGAIN
620 CLS :
    FOR T = 1 TO 200:
```

Program continued

```
     NEXT T:
    A$ = INKEY$:
    IF SC > HS
     THEN
      HS = SC
630 CLS :
    PRINT @477, STRING$(3,191);:
    GOSUB 810
640 PRINT @538, STRING$(10,166);:
    PRINT @474, STRING$(10,166);
650 PRINT @410, STRING$(10,166);:
    GOSUB 810:
    FOR X = 272 TO 656 STEP 64
660  PRINT @X, STRING$(30,162);:
     NEXT X:
    GOSUB 810
670 FOR X = 836 TO 132 STEP - 64:
     PRINT @X, STRING$(56,140);:
     NEXT X
680 A$ = INKEY$:
    PRINT @473,"HIGH SCORE";HS;:
    VV = 724
690 PRINT @220,"SCORE";SC;
700 PRINT @723,"PRESS ANY KEY TO PLAY AGAIN";
710 FOR X = 0 TO 60:
     PRINT @X,A$(5);:
     PRINT @X," ";:
     PRINT @X + 64," ";
720  A$ = INKEY$:
     IF A$ = ""
      THEN
       NEXT X:
     ELSE
      800
730 FOR X = 60 TO 891 STEP 64:
     PRINT @X,A$(5);:
     PRINT @X,"     ";:
     A$ = INKEY$
740  IF A$ = ""
      THEN
       NEXT :
     ELSE
      800
750 PRINT @892,"     ";:
    PRINT @959," ";:
    FOR X = 954 TO 896 STEP - 1
760  PRINT @X,A$(5);:
     PRINT @X + 4," ";:
     PRINT @X + 68," ";:
     A$ = INKEY$
770  IF A$ = ""
      THEN
       NEXT :
     ELSE
      800
780 FOR X = 896 TO 64 STEP - 64:
     PRINT @X,A$(5);:
     PRINT @X + 64,"     ";
790  A$ = INKEY$:
     IF A$ = ""
      THEN
       NEXT X:
      GOTO 710:
     ELSE
      800
800 PRINT @723, CHR$(219);:
    GOTO 160
810 FOR T = 1 TO 500:
     NEXT T:
    RETURN
820 REM *** INSTRUCTIONS
```

```
830 CLS
840 PRINT STRING$(18,42)" S P A C E    M I S S I O N " STRING$(18,42
    )
850 PRINT "  YOU CONTROL A LASER'S SIGHT WITH THE FOUR ARROW KEYS.
    YOU CAN";
860 PRINT "HOLD DOWN 1 OR 2 ARROWS AT A TIME.  WHEN THE SHIP YOU ARE
    "
870 PRINT "FOCUSING ON APPEARS TO BE IN THE SIGHT, PRESS THE SPACE B
    AR TO"
880 PRINT "FIRE.  THE ALIEN WILL GET SMALLER AND SMALLER.  THE SOONE
    R YOU"
890 PRINT "HIT HIM THE MORE POINTS YOU WILL GET."
900 PRINT "  IT IS HARD TO USE THE LASER SIGHT CONTROLS, AS THEY MAY
     SEEM"
910 PRINT "BACKWARD, BUT IN TIME YOU WILL MASTER THEM AND DESTROY MA
    NY OF"
920 PRINT "THE TERRIBLE ENEMY SHIPS."
930 PRINT "  IF YOU MOVE YOUR SIGHT TOO FAR FROM THE ALIEN SHIP, IT
    WILL"
940 PRINT "DISAPPEAR FROM YOUR VIEWING SCREEN.  BE CAREFUL, WHEN YOU
    "
950 PRINT "THINK YOUR SIGHT IS MOVING LEFT, IT MAY BE MOVING RIGHT."
960 PRINT "  YOU HAVE 200 TIME UNITS BEFORE YOU MUST REFUEL FOR A NE
    W GAME."
970 VV = 979:
    GOTO 160
```

GAMES

Slot Machine

by Kerry Rasmussen

In the following program, I have utilized the special graphics capabilities on the Model III to simulate a slot machine. I wrote the Slot Machine program for a Model III Level II TRS-80 with 16K. Answer the memory size prompt with 32000.

When you turn on the Model III, it is set in the space compression mode. To change this to the special graphics mode, you would normally use the command:

```
PRINT CHR$(21)
```

If this statement is located within the program, however, every time the computer reads the statement, it switches itself from the special graphics mode to the space compression mode, and vice versa. In order to prevent this from happening, line 4 (see Program Listing) stores a non-zero in memory location 16420 so the computer will stay in the special graphics mode.

The program keeps track of the number of times you have played, and uses this to determine the amount of the jackpot. The longer you play, the more the jackpot increases.

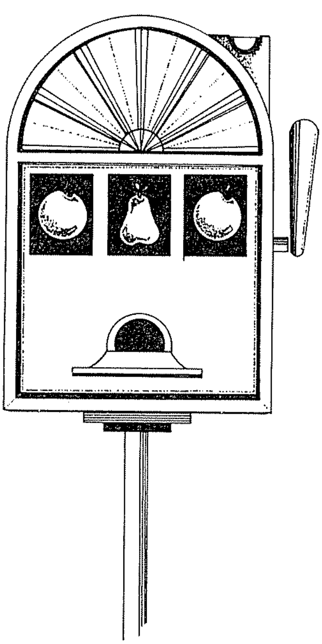
To play Slot Machine, you are given $10.00. This amount is displayed in the upper right portion of the video screen. When you push the spacebar, $1.00 is subtracted from the pot, and the handle of the slot machine goes down. It's just like the ones in Las Vegas! If you lose, it waits for you to put another dollar in. If you win, depending on the sequence of the reels (as per the win chart displayed to the left side of the screen), it begins to pay off. Coins fall out of the machine, complete with a clinking sound (if you are so equipped)! The FOR-NEXT loop in lines 10210–11065 keeps track of this. When the coins stop falling, the total amount you won flashes at the bottom of the screen and rolls to the top right to be added to the total. You can stop playing at any time, and the total amount won or lost will be displayed. The $10.00 you started with is subtracted.

Because the computer is in the special graphics mode for this program it does not use space compression characters. The program contains PRINT@ statements such as: PRINT@ 1000," "; which erases the video starting at PRINT position 1000. The number of spaces between the quotation marks is critical, since the spaces erase what is printed there.

I have added sound to this game, to add a little pizazz. Lines 20000 through 60440 contain the data for the sound. To utilize the sound effects, plug the large gray plug that normally goes to the auxiliary port of your tape player into an amplifier. The Telephone Listener sold by Radio Shack is excellent for this purpose and costs about $10.00.

A	Random number (1 to 5) determines value of the first reel
B	Random number (1 to 5) determines value of the second reel
B1	FOR-NEXT loop for sound subroutine
C	Random number (1 to 5) determines value of the third reel
H	Amount of win or loss
I	Used for a time delay
J	Amount to add to jackpot (is equal to random 0)
O	Play counter
S	The amount of money you start play with
T	Your present money total
V	Current amount won
Z	Counter for FOR-NEXT loop for payoff
N$	String value field for print using amount won
C$	INKEY$
M$	INKEY$

Table 1. *Program variables*



Program Listing. *Slot Machine*

```
  1 REM * SLOTS *
  2 REM * Copyright 1981 by KERRY RASMUSSEN, all rights reserved *
  4 POKE 16420,1
  6 RANDOM
  8 POKE 16527,125:
    POKE 16526,1:
    GOSUB 30000
  9 S = 0:
    H = 0:
    T = 0:
    P = 0
 10 CLS
 15 PRINT @29,"SLOTS"
 16 S = 10
 17 T = S + H
 18 T = T + H
 19 N$ = "$$###.00"
 20 PRINT @120, USING N$;S
 50 GOSUB 4000
100 REM * DRAWS SLOT MACHINE *
111 FOR X = 79 TO 82:
     Y = 9:
     SET(X,Y):
     NEXT X
112 X = 79:
    Y = 10:
    SET(X,Y)
113 X = 82:
    Y = 10:
    SET(X,Y)
115 FOR X = 35 TO 85:
     Y = 11:
     SET(X,Y):
     NEXT X
135 FOR X = 35 TO 85:
     Y = 36:
     SET(X,Y):
     NEXT X
160 FOR Y = 11 TO 36:
     X = 35:
     SET(X,Y):
     NEXT Y
180 FOR Y = 11 TO 36:
     X = 85:
     SET(X,Y):
     NEXT Y
200 FOR X = 38 TO 48:
     Y = 13:
     SET(X,Y):
     NEXT X
220 FOR X = 55 TO 65:
     Y = 13:
     SET(X,Y):
     NEXT X
240 FOR X = 72 TO 82:
     Y = 13:
     SET(X,Y):
     NEXT X
260 FOR Y = 13 TO 18:
     X = 38:
     SET(X,Y):
     NEXT Y
280 FOR Y = 13 TO 18:
     X = 48:
     SET(X,Y):
     NEXT Y
300 FOR Y = 13 TO 18:
```

```
     X = 55:
     SET(X,Y):
     NEXT Y
320 FOR Y = 13 TO 18:
     X = 65:
     SET(X,Y):
     NEXT Y
340 FOR Y = 13 TO 18:
     X = 72:
     SET(X,Y):
     NEXT Y
380 FOR X = 39 TO 48:
     Y = 18:
     SET(X,Y):
     NEXT X
420 FOR X = 55 TO 65:
     Y = 18:
     SET(X,Y):
     NEXT X
440 FOR X = 72 TO 82:
     Y = 18:
     SET(X,Y):
     NEXT X
460 FOR Y = 13 TO 18:
     X = 82:
     SET(X,Y):
     NEXT Y
485 FOR X = 45 TO 74:
     Y = 30:
     SET(X,Y):
     NEXT X
505 FOR X = 44 TO 44:
     Y = 31:
     SET(X,Y):
     NEXT X
510 FOR X = 75 TO 75
515  Y = 31:
     SET(X,Y):
     NEXT X
520 FOR X = 45 TO 74:
     Y = 32:
     SET(X,Y):
     NEXT X
525 FOR X = 88 TO 90:
     Y = 17:
     SET(X,Y):
     NEXT X:
    FOR X = 88 TO 90:
     Y = 18:
     SET(X,Y):
     NEXT X
530 FOR Y = 19 TO 28:
     X = 89:
     SET(X,Y):
     NEXT Y
535 FOR X = 86 TO 88:
     Y = 26:
     SET(X,Y):
     NEXT X:
    FOR X = 86 TO 88:
     Y = 27:
     SET(X,Y):
     NEXT X
550 PRINT @476,"SLOT";
551 PRINT @539,"MACHINE";
560 GOTO 979
570 GOTO 7500
799 REM * PICKS RANDOM NUMBER FOR EACH WINDOW *
800 RANDOM
802 A = RND(5)
804 B = RND(5)
```

Program continued

```
806 C =  RND(5)
810 REM  * CLEARS EACH WINDOW *
811 PRINT @340,"   ";
812 PRINT @349,"   ";
813 PRINT @357,"   ";
815 REM  * GIVES WINDOWS AN APPEARANCE OF SPINNING REELS *
816 FOR  I = 192 TO 255:
     PRINT @341, CHR$(I);@350, CHR$(I);@358, CHR$(I);:
     NEXT I
818 GOSUB 60200
819 REM  * GIVES RANDOM NUMBERS AN EQUIVALENT CHARACTER *
820 IF A = 1 PRINT @340,"BAR";
825 IF A = 2 PRINT @341, CHR$(193);
830 IF A = 3 PRINT @341, CHR$(214);
835 IF A = 4 PRINT @341, CHR$(234);
840 IF A = 5 PRINT @341, CHR$(192);
842 FOR  I = 192 TO 255:
     PRINT @350, CHR$(I);@358, CHR$(I);:
     NEXT I:
    GOSUB 60200
845 IF B = 1 PRINT @349,"BAR";
850 IF B = 2 PRINT @350, CHR$(193);
855 IF B = 3 PRINT @350, CHR$(214);
860 IF B = 4 PRINT @350, CHR$(234);
865 IF B = 5 PRINT @350, CHR$(192);
866 FOR  I = 192 TO 255:
     PRINT @358, CHR$(I);:
     NEXT I:
    GOSUB 60200
870 IF C = 1 PRINT @357,"BAR";
875 IF C = 2 PRINT @358, CHR$(193);
880 IF C = 3 PRINT @358, CHR$(214);
885 IF C = 4 PRINT @358, CHR$(234);
890 IF C = 5 PRINT @358, CHR$(192);
900 FOR  I = 1 TO 25
910  NEXT I
920 GOSUB 9000
922 IF H < 2
     THEN
      H  = 0
926 T =  T + H
927 FOR  B = 1 TO 3:
     IF  H = 0 PRINT @29," LOSE ";:
     GOSUB 60100:
      ELSE
       PRINT @29," WIN ";:
       FOR I = 1 TO 25:
        NEXT I:
       PRINT @29,"      ";:
       GOSUB 60500:
       NEXT B:
      PRINT @29," WIN ";:
      GOSUB 10210
935 PRINT @120, USING N$;T;
950 IF  T = 0 GOTO 10000
979 C$  =  INKEY$:
    IF  C$ = "Q"
     THEN
      12000
980 PRINT @930,"<SPACEBAR TO PLAY / Q TO QUIT>";:
    IF  C$ < > " "
     THEN
      979
981 PRINT @29,"       ";:
    PRINT @930,"                              ";
985 H =  0
986 P =  P + 1
987 PRINT @960,P;
988 T =  T - 1
990 PRINT @120, USING N$;T;
991 RESTORE
```

```
1000 GOTO 8000
3900 REM * DRAWS WINNING COMBINATIONS AT LEFT SIDE OF SCREEN *
4000 PRINT @128,"BAR/BAR/BAR=$20+   ";
4005 PRINT @192, CHR$(234) CHR$(234) CHR$(234)"=$18";
4010 PRINT @256, CHR$(234) CHR$(234)"BAR=$18";
4015 PRINT @320, CHR$(192) CHR$(192) CHR$(192)"=$14";
4020 PRINT @384, CHR$(192) CHR$(192)"BAR=$14";
4025 PRINT @448, CHR$(214) CHR$(214) CHR$(214)"=$10";
4030 PRINT @512, CHR$(214) CHR$(214)"BAR=$10";
4035 PRINT @576, CHR$(193) CHR$(193)"-=$5";
4040 PRINT @640, CHR$(193)"--=$2";
4060 RETURN
7500 FOR I = 1 TO 500:
      NEXT I
7900 REM * GRAPHICS FOR HANDLE MOVEMENT *
8000 GOSUB 60200:
     FOR X = 88 TO 90:
      Y = 17:
      RESET(X,Y):
      NEXT X:
     GOSUB 60200:
     FOR X = 88 TO 90:
      Y = 18:
      RESET(X,Y):
      NEXT X
8005 GOSUB 60200:
     FOR Y = 19 TO 24:
      X = 89:
      RESET(X,Y):
      NEXT Y
8006 GOSUB 60200
8020 GOSUB 60200:
     FOR Y = 28 TO 37:
      X = 89:
      SET(X,Y):
      NEXT Y:
     GOSUB 60200:
     FOR X = 88 TO 90:
      Y = 36:
      SET(X,Y):
      NEXT X
8025 GOSUB 60200:
     FOR X = 88 TO 90:
      Y = 37:
      SET(X,Y):
      NEXT X
8030 GOSUB 60200:
     FOR I = 1 TO 100:
      NEXT I
8035 FOR X = 88 TO 90:
      Y = 37:
      RESET(X,Y):
      NEXT X:
     GOSUB 60200:
     FOR X = 88 TO 90:
      Y = 36:
      RESET(X,Y):
      NEXT X
8040 GOSUB 60200:
     FOR Y = 35 TO 28 STEP - 1:
      X = 89:
      RESET(X,Y):
      NEXT Y
8045 GOSUB 60200:
     FOR Y = 28 TO 17 STEP - 1:
      X = 89:
      SET(X,Y):
      NEXT Y
8046 GOSUB 60200:
     FOR Y = 18 TO 17 STEP - 1:
      X = 88:
```

Program continued

```
      SET(X,Y):
      NEXT Y
8047 GOSUB 60200:
     FOR Y = 18 TO 17 STEP - 1:
      X = 90:
      SET(X,Y):
      NEXT Y
8050 GOTO 800
8900 REM * DETERMINES IF CHOSEN NUMBERS WIN OR LOSE *
9000 RANDOM :
     J = RND(P):
     IF A = 1 AND B = 1 AND C = 1
      THEN
       H = 20 + J
9001 IF H > = 20 PRINT @27 ,"JACKPOT ";:
     FOR I = 1 TO 150:
      NEXT I:
     GOSUB 20000:
     PRINT @27,"         ";:
     FOR I = 1 TO 100:
      NEXT I
9002 IF H > = 20 GOSUB 50000:
     PRINT @27,"JACKPOT ";:
     FOR I = 1 TO 150:
      NEXT I:
     PRINT @27,"         ";:
     FOR I = 1 TO 100:
      NEXT I
9003 IF H > = 20 GOSUB 50000:
     PRINT @27,"JACKPOT ";:
     FOR I = 1 TO 150:
      NEXT I:
     PRINT @27,"         ";:
     FOR I = 1 TO 100:
      NEXT I
9005 IF A = 1 AND B = 1 AND C < > 1
      THEN
       H = - 1
9010 IF A = 1 AND B < > 1 AND C < > 1
      THEN
       H = - 1
9015 IF A = 1 AND B < > 1 AND C = 1
      THEN
       H = - 1
9025 IF A = 2 AND B = 2 AND C > 0
      THEN
       H = 5
9030 IF A = 2 AND B < > 2 AND C > 0
      THEN
       H = 2
9040 IF A = 3 AND B = 3 AND C = 3
      THEN
       H = 10
9045 IF A = 3 AND B = 3 AND C < > 3
      THEN
       GOSUB 9500
9050 IF A = 3 AND B < > 3 AND C < > 3
      THEN
       H = - 1
9055 IF A = 3 AND B < > 3 AND C = 3
      THEN
       H = - 1
9060 IF A = 4 AND B = 4 AND C = 4
      THEN
       H = 18
9065 IF A = 4 AND B = 4 AND C < > 4
      THEN
       GOSUB 9600
9070 IF A = 4 AND B < > 4 AND C < > 4
      THEN
       H = - 1
```

```
9075 IF A = 4 AND B < > 4 AND C = 4
      THEN
       H = - 1
9085 IF A = 5 AND B = 5 AND C = 5
      THEN
       H = 14
9090 IF A = 5 AND B = 5 AND C < > 5
      THEN
       GOSUB 9700
9095 IF A = 5 AND B < > 5 AND C < > 5
      THEN
       H = - 1
9100 IF A = 5 AND B < > 5 AND C = 5
      THEN
       H = - 1
9200 RETURN
9500 IF C = 1
      THEN
       H = 10:
       RETURN :
       :
      ELSE
       H = - 1:
       RETURN
9600 IF C = 1
      THEN
       H = 18:
       RETURN :
       :
      ELSE
       H = - 1:
       RETURN
9700 IF C = 1
      THEN
       H = 14:
       RETURN :
       :
      ELSE
       H = - 1:
       RETURN
10000 M$ = INKEY$
10001 PRINT @ 151,"WANT TO TRY AGAIN? (Y OR N)";:
      IF M$ = ""
       THEN
        10000
10002 IF M$ = "N"
       THEN
        CLS :
        END
10003 IF M$ = "Y"
       THEN
        1
10005 IF M$ < > "Y" OR M$ < > "N"
       THEN
        10000
10200 REM * GRAPHICS FOR COINS FALLING..ALSO FOR NEXT LOOP TO DETERMIN
      E HOW MANY TIMES TO REPEAT *
10210 FOR Z = 1 TO H
11000  PRINT @670,"o";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @670," ";
11005  FOR X = 60 TO 61:
        Y = 30:
        SET(X,Y):
        NEXT X
11010  PRINT @734,"-";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @734," ";
11015  FOR X = 60 TO 61:
```

Program continued

```
        Y = 32:
        SET(X,Y):
        NEXT X
11020  PRINT @798,"o";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @798," ";
11025  FOR X = 60 TO 61:
        Y = 36:
        SET(X,Y):
        NEXT X
11030  PRINT @862,"-";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @862," ";
11040  PRINT @926,"o";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @926," ";
11050  PRINT @990,"-";:
       FOR I = 1 TO 2:
        NEXT I:
       PRINT @990," ";:
       FOR I = 1 TO 10:
        NEXT I
11051  GOSUB 60200
11060  PRINT @990,"x";:
       FOR I = 1 TO 10:
        NEXT I:
       PRINT @990," ";:
       FOR I = 1 TO 10:
        NEXT I
11061  GOSUB 60200 PRINT @990,"-";:
       FOR I = 1 TO 10:
        NEXT I:
       PRINT @990," ";:
       FOR I = 1 TO 10:
        NEXT I
11062  PRINT @990,"x";:
       FOR I = 1 TO 10:
        NEXT I:
       PRINT @990," ";:
       FOR I = 1 TO 10:
        NEXT I
11063  PRINT @990,"-";:
       FOR I = 1 TO 10:
        NEXT I:
       PRINT @990," ";
11065  NEXT Z
11069 REM * ROUTINE FOR LISTING AMOUNT WON AND MOVING IT TO THE BALANC
      E AT TOP RIGHT OF SCREEN *
11070 V = H:
      PRINT @993,"="; USING N$;V;
11080 FOR I = 1 TO 100:
       NEXT I:
      PRINT @993,"         ";
11085 PRINT @1000, USING N$;V;:
      FOR I = 1 TO 60:
       NEXT I:
      PRINT @1000,"         ";
11086 GOSUB 60000
11090 PRINT @1010, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @1010,"         ";I
11100 PRINT @888, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @888,"         ";
11101 GOSUB 60000
11105 PRINT @760, USING N$;V;:
```

```
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @760,"        ";
11106 GOSUB 60000
11110 PRINT @632, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @632,"        ";
11111 GOSUB 60000
11115 PRINT @504, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @504,"        ";
11116 GOSUB 60000
11120 PRINT @376, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @376,"        ";
11121 GOSUB 60000
11125 PRINT @248, USING N$;V;:
      FOR I = 1 TO 2:
       NEXT I:
      PRINT @248,"        ";
11126 GOSUB 60000
11200 RETURN
11300 END
12000 CLS :
      PRINT @29, USING N$;T - S;:
      IF T < 10 PRINT @128,"YOU CAME OUT A LOSER, ";:
      GOSUB 60400:
      GOTO 10000
12010 IF T > 10 PRINT @128,"YOU CAME OUT A WINNER, ";:
      GOSUB 20000:
      GOTO 10000
12020 IF T = 10 PRINT @128,"YOU BROKE EVEN, ";:
      GOSUB 20000:
      GOSUB 60400:
      GOTO 10000
20000 POKE 32004,255:
      POKE 32020,150
20010 FOR B1 = 1 TO 25:
       X = B1
20020  X = USR(0):
       FOR W = 1 TO 10:
        NEXT W
20030  NEXT B1
20040 RETURN
30000 FOR Y = 32001 TO 32026
30010  READ D:
       POKE Y,D
30020  NEXT Y
30030 RETURN
30040 DATA 14,255,33,0,20,58,61,64,230,253,198,2,211
30050 DATA 255,214,2,211,255,6,150,16,254,37,32,241,201
30060 DATA 85,80,75,70,65,60,55,50
50000 FOR Y = 100 TO 50 STEP - 10
50010  POKE 32020,Y
50020  X = USR(0)
50030  FOR W = 1 TO 1:
        NEXT W
50040  NEXT Y
50050 RETURN
60000 POKE 32020,255
60010 X = USR(0)
60020 RETURN
60100 FOR Y = 1 TO 30
60110  POKE 32020,15
60115  X = USR(0)
60120  NEXT Y
60130 RETURN
60200 FOR Y = 1 TO 1
```

Program continued

```
60210  POKE 32020,30
60220  X = USR(0)
60230  NEXT Y
60240 RETURN
60400 FOR Y = 10 TO 100
60410  POKE 32020,Y
60420  X = USR(0)
60430  NEXT Y
60440 RETURN
60500 FOR Y = 200 TO 150 STEP - 4
60505  POKE 32020,Y
60510  X = USR(0)
60515  NEXT Y
60520 RETURN
```

GRAPHICS

Level II Graphics Code
New Compusketch

Level II Graphics Code

by Fred Blechman

Some TRS-80 Level II programs run with unusual graphics figures, such as racing horses. This can be done in Level I to some degree, with SET commands, but Level II has a graphics code built into the ROM BASIC that is much faster in both operation and programming. This article describes a simple way to break the code and use it, and three short programs that illustrate the technique.

The TRS-80 display screen is divided into 1024 printing locations, 0 to 1023 (64 across by 16 down). Normally, each of these locations is occupied by a character, whether it be a letter, number, symbol, or blank. Each location is a rectangular area divided into six segments (two columns of three rows each), as shown in Figure 1. By proper use of the CHR$ command, you can light any single segment or combination of segments on your display. By putting these combinations together, you can form symbols, shapes, large letters, simulated playing fields, or other displays.

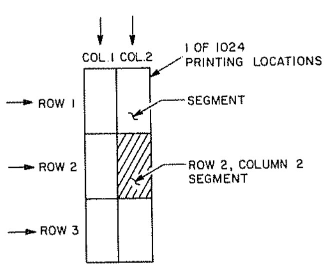


Figure 1. *Printing location division*

Figure 2 shows the graphics code for each of the possible 64 segment "on" combinations, from "all segments off" (128) to "all segments on" (191). These are used in a program as CHR$(number). For example, if you used CHR$(157) in a program after a print instruction, you'd light all column 1 segments as well as the column 2 segment of row 2 at the current printing location. As another example, CHR$(140) lights both columns of row 2 at the current printing location.

This graphics code is based on a binary code and is easy to remember if you crack the code. In Figure 3, notice that each of the six segments in a printing location is assigned a decimal number representing the powers of 2. Going from left to right, and from top to bottom, starting with 1, each number is exactly twice the value of the previous number. This is the basis of binary counting.

To determine the TRS-80 graphics code number, add the numbers of each lighted segment and then add 128. Figure 3 shows some examples. Now you

Figure 2. *TRS-80 graphics code (▩ = lighted segment)*

won't need to have Figure 2 handy all the time, since you'll be able to determine quickly the number you want for every one of the 64 possible combinations. Remember, 128 is a totally blank space, and 63 (total of all segments) plus 128 is equal to 191, a fully lighted space.

To design your own symbol or large letters, use the TRS-80 Video Display Worksheet in your manual (page E/1). Simply draw lightly in pencil whatever shape you want, noting that the heavier lines on the worksheet form the 1024 printing location rectangles, and the lighter lines subdivide these rectangles into six segments each. Now, convert your design to the proper combination of CHR$ numbers to "draw" this shape on your display screen.

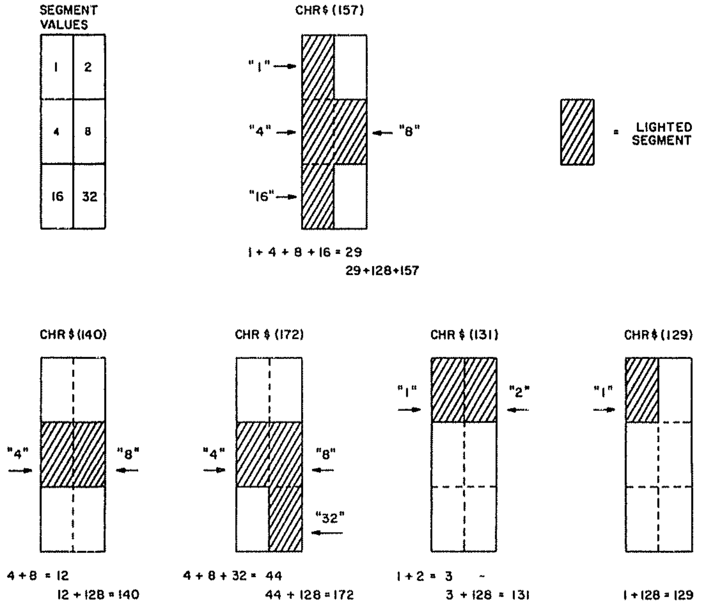


Figure 3. *Cracking the code*

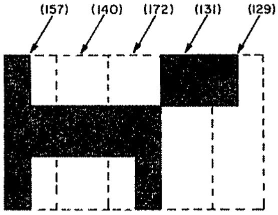


Figure 4. *Graphic horse or dog using five printing locations*

Figure 4 is a symbolic horse or dog composed of five CHR$ numbers on a single display line. You can place this animal anywhere on your screen with a PRINT @ instruction. How about a racing dog? Try the short program which I call Run, Spot, Run in Program Listing 1.

Line 40 stops the program to allow Spot to remain at one location long enough to be visually stable. Line 50 blanks out the space for the next image. (Try running without line 50 and watch what happens!) Line 70 and the semicolons at the end of lines 30 and 50 keep Spot from being chopped into small pieces that scroll up the screen.

Perhaps you'd like to see your name in huge letters on the screen. Just follow the same procedure as above, but remember that you'll need several screen lines. Since you'll be commanding a relatively large number of locations on the screen, use READ/DATA statements. If you'd like to see *my* name in big letters (over 2 inches high), try the program in Program Listing 2.

This by no means exhausts the possibilities of using the graphics code. You are limited only by your imagination, patience, and the size of the computer memory. Program Listing 3 shows a listing of a five-dog race, with graphic dogs, a finish line, and winner announcement.

Program Listing 1. *Run, Spot, Run*

```
10 CLS
20 X = 0
30 PRINT @X, CHR$(157); CHR$(140); CHR$(172); CHR$(131); CHR$(129);
40 FOR Y = 1 TO 40:
   NEXT Y
50 PRINT @X, "      ";
60 X = X + 1
70 IF X = 1018 CLS :
   X = 0
80 GOTO 30
```

Program Listing 2. *Printing of author's name*

```
 10 CLS
 20 PRINT @ 266, CHR$(191);:
    GOSUB 500
 30 PRINT @ 330, CHR$(191);:
    GOSUB 500
 40 PRINT @ 394, CHR$(191);:
    GOSUB 500
 50 PRINT @ 458, CHR$(191);:
    GOSUB 500
 60 PRINT @ 522, CHR$(191);:
    GOSUB 500
 65 GOTO 65
 70 DATA 191,191,191,191,191,191,191,191,128,128
 75 DATA 191,191,191,191,191,191,191,191,180
 80 DATA 128,128,128,191,191,191,191,191,191,191
 85 DATA 191,191,128,128,191,191,191,191,191,191,191,189,144
 90 DATA 191,191,128,128,128,128,128,128,128,128
 95 DATA 191,191,191,128,128,128,179,191,191,157,128,128
100 DATA 191,191,191,128,128,128,128,128,128
105 DATA 128,128,191,191,191,128,128,139,191,191,191
110 DATA 191,191,191,191,191,128,128,128,128
115 DATA 128,191,191,191,191,191,191,191,191,135,128,128,128
120 DATA 191,191,191,191,191,191,128,128,128
125 DATA 128,128,191,191,191,128,128,128,191,191,191
130 DATA 191,191,128,128,128,128,128,128,128
135 DATA 128,191,191,191,128,139,191,191,180,128,128,128,128
140 DATA 191,191,191,128,128,128,128,128,128
145 DATA 128,128,191,191,191,128,128,184,191,191,191
150 DATA 191,191,128,128,128,128,128,128,128
155 DATA 128,191,191,191,128,128,130,191,191,189,176,128,128
160 DATA 191,191,191,191,191,191,191,191,191
165 DATA 128,128,191,191,191,191,191,191,191,159,129
500 FOR R = 1 TO 42
510  READ X
520  PRINT CHR$(X);
530  NEXT R
540 RETURN
```

Program Listing 3. *Five-dog race*

```
 5 REM  * DOG RACE WITH GRAPHIC DOGS - LEVEL II
 7 REM  * SET SPEED AT LINES 60 - 100 (A = A + 2, ETC.)
10 CLS
15 FOR Y = 3 TO 29:
    SET (123,Y):
    NEXT
20 A = 64:
   B = 192:
```

Program continued

```
    C = 320:
    D = 448:
    E = 576
 21 PRINT @A, CHR$(157);"1"; CHR$(172); CHR$(131); CHR$(129);
 22 PRINT @B, CHR$(157);"2"; CHR$(172); CHR$(131); CHR$(129);
 23 PRINT @C, CHR$(157);"3"; CHR$(172); CHR$(131); CHR$(129);
 24 PRINT @D, CHR$(157);"4"; CHR$(172); CHR$(131); CHR$(129);
 25 PRINT @E, CHR$(157);"5"; CHR$(172); CHR$(131); CHR$(129);
 26 GOTO 55
 30 PRINT @A, CHR$(157);"1"; CHR$(172); CHR$(131); CHR$(129);:
    GOTO 50
 31 PRINT @B, CHR$(157);"2"; CHR$(172); CHR$(131); CHR$(129);:
    GOTO 50
 32 PRINT @C, CHR$(157);"3"; CHR$(172); CHR$(131); CHR$(129);:
    GOTO 50
 33 PRINT @D, CHR$(157);"4"; CHR$(172); CHR$(131); CHR$(129);:
    GOTO 50
 34 PRINT @E, CHR$(157);"5"; CHR$(172); CHR$(131); CHR$(129);
 50 IF A > 120 PRINT @730,"#1 WINS!!!":
    END
 51 IF B > 248 PRINT @730,"#2 WINS!!!":
    END
 52 IF C > 376 PRINT @730,"#3 WINS!!!":
    END
 53 IF D > 504 PRINT @730,"#4 WINS!!!":
    END
 54 IF E > 632 PRINT @730,"#5 WINS!!!":
    END
 55 X = RND(5)
 56 ON X GOTO 60, 70, 80, 90, 100
 60 PRINT @A,"     ";:
    A = A+ 1:
    GOTO 30
 70 PRINT @B,"     ";:
    B = B+ 1:
    GOTO 31
 80 PRINT @C,"     ";:
    C = C+ 1:
    GOTO 32
 90 PRINT @D,"     ";:
    D = D+ 1:
    GOTO 33
100 PRINT @E,"     ";:
    E = E+ 1:
    GOTO 34
```

New Compu-Sketch

by Phil Burton

In the December 1980 issue of *80 Microcomputing* was a fascinating article by Merl J. Hendricks called "Compu-Sketch." The 14-line program, written in BASIC for the Model I TRS-80, made it possible for a TRS-80 computer operator to draw pictures on the screen using only the left, right, up, and down arrows, and the space bar.

You can create animation by taking a picture and moving one part of it in a sequence of pictures. As an example, picture an arrow going around the outside of the screen. You can make the arrow spin around the outside of the picture by first drawing an arrow at the bottom middle of the screen. Next, draw a blank where the arrow was. Follow this with the picture with the arrow moved to the right. Then draw a blank where the arrow was. Again draw the picture with the arrow moved up one side, and so on. The number of pictures you can use depends on how much internal memory your computer has. Each picture is stored and drawn on the screen, alternating between the picture with the arrow blanked and one with it moved left, right, up, or down. If you do not want animation but do want graphics in your program, you can use this program to draw the graphics you want, then call up your picture for use in a particular program.

A Straight Line Between Two Points

The low resolution graphics on the TRS-80, make it difficult to draw a straight line between two points on the screen if they are anything but horizontal or vertical points, or 45 degrees away from each other. William Barden Jr., in his book, *Programming Techniques for Level II BASIC*, explains how to draw a straight line between any two points. By storing any x-and y-coordinates at position 1 (or point 1) and moving to a second position on the screen and storing those coordinates, the computer can draw a straight line between the two points (at least as straight as the TRS-80 will allow). Armed with this knowledge, all I had to do was combine that idea with the Compu-Sketch program, and I was in business.

Saving Screen Graphics

It is time-consuming to look in the reference manual at the graphics codes and try to come up with the right code and put it in the right sequence. Doing it this way requires a series of DATA-READ statements for each graphics character in each position of the screen display.

One place you can store information in the computer is on the screen display. By setting up dummy strings, you can let the computer pack codes into them that come from the screen itself. When you print the string, the screen displays graphics instead of alphanumeric characters.

Dress It up with a Blinking Cursor

To keep from destroying the picture when I name a program or give load and save instruction for cassette, I created a blinking cursor subroutine which I use in nearly all my programs requiring keyboard input. The new version of Compu-Sketch prints messages on the bottom line of the screen.

In the subroutine I call Blinking Cursor, the cursor is first turned on and the INKEY$ function is activated. Next the cursor is turned off, and the INKEY$ function is activated again. If you enter any value greater than a null, the subroutine branches to the section that looks for special control characters (such as back space, enter, return to head of line, etc.). If a back space is involved, one character is removed from the right end of the string, and a graphics character (131 decimal) is POKEd into that position on the screen display.

To get to the subroutine, define the number of characters you want in your field using the variable FL. If you press ENTER, or if the value of FL is 1, you return to the main program. Upon return, you must transfer the value stored in IN$ to a permanent variable. If it is a numeric variable, use VAL(IN$).

If you want to use this subroutine in any of your programs or if you decide to add more arguments to it, be sure to turn the cursor off (CHR$(15)) before returning to the main program. If you don't, strange cursor marks will appear on the screen.

I had a reason for not using the bottom line (starting at print position 960) for graphics: When the bottom line (all 64 positions) is printed, there is an automatic carriage return/line feed that causes the picture to scroll up one line, and you lose the top line of your picture. There is a way around it, but that involves drawing in the lower right corner of the picture every time you load it.

How the Program Operates

Program Listing 1 is the disk version of Compu-Sketch; Program Listing 2 is the tape version. You can draw pictures by using only the direction arrows. Line 70 loads a value into variable C from the keyboard position that controls the arrows, PEEK(14400). By looking at these codes, the computer can tell which key you have pressed and returns a value of 2, 4, 8, 16, 32, 64, or 128. All you have to do is tell the computer to do something for you based on which key you pressed. For example, lines 80–110 test for one of the four direction arrows (up, down, left, or right). Line 160 looks to see if you pressed the CLEAR key. Line 170 checks for the combination of codes that indicates the space bar is being held down.

Lines 180 and 190 are the special function keys. In addition to being able to draw a picture, you can start over by pressing the letter C to clear the screen. When you are ready to save your picture, press the letter S. Press the letter L to load a previously saved file, and press the letter E to end the program. Table 1 lists the keys you need to become a great cartoonist.

Up arrow	Moves cursor up
Down arrow	Moves cursor down
Left arrow	Moves cursor left
Right arrow	Moves cursor right
Space bar	Hold it down while pressing one of the arrows to draw a solid line.
Combination	Hold down two arrows and the space bar to draw a horizontal line.
CLEAR key	Press once for the x-coordinate. Move the cursor to another position and press it again for the y-coordinate. A straight line will be drawn between the two coordinates.
C key	Clears the screen for a new picture
E key	Clears the screen and ends the program
L key	Load. Asks for the file name of a previously saved picture and loads it from disk. On the tape version, it asks you to prepare the tape recorder.
S key	Save. Asks for a file name to save the current picture to disk. On the tape version, it asks you to prepare the tape recorder.

Table 1. *Keys used in New Compu-Sketch*

Applications

The program I wrote can be used to create simple animation as well as more complex drawings in a shorter span of time. This concept works well in other areas too if you want to put alphanumeric characters in the middle of the screen and point out special items with graphics.

The next time you use graphics in a program, try animation. It is very easy with this program. With some experimentation, you will find that you need to change only small portions of your original picture to achieve motion. By printing only the lines with changes in them, you can have a fast moving motion picture.

Program Listing 1. *New Compu-Sketch, disk version*

```
  1 :
    ' ***********************************************************
  2 :
    ' *                                                         *
  3 :
    ' *                  NEW COMPU-SKETCH                       -*
  4 :
    ' *                    PHIL BURTON ,                        *
  5 :
    ' *                 1251 WAVERLY DRIVE                      *
  6 :
    ' *          DAYTONA BEACH, FLORIDA 32018                   *
  7 :
    ' *                   (904) 252-6911                        *
  8 :
    ' *                                                         *
  9 :
    ' ***********************************************************
 10 CLEAR 100:
    ON ERROR GOTO 370
 20 CLS :
    X = 0:
    Y = 0:
    GOSUB 380:
    PRINT @960,"DO YOU WANT TO LOAD AN EXISTING FILE? ";:
    FL = 1:
    GOSUB 690:
    IF IN$ = "Y"
     THEN
      GOSUB 380:
      GOTO 290
 30 CLS :
    GOSUB 380
 40 GOSUB 70:
    X1 = X:
    Y1 = Y :
    ' SAVE FIRST X,Y COORDINATES
 50 GOSUB 70:
    X2 = X:
    Y2 = Y :
    ' SAVE SECOND X,Y COORDINATES
 60 GOSUB 400:
    GOTO 40
 70 C = PEEK(14400)
 80 IF C AND 8
     THEN
      Y = Y - 1 :
      ' UP ARROW
 90 IF C AND 16
     THEN
      Y = Y + 1 :
      ' DOWN ARROW
100 IF C AND 32
     THEN
      X = X - 1 :
      ' LEFT ARROW
110 IF C AND 64
     THEN
      X = X + 1 :
      ' RIGHT ARROW
120 IF X > 127
     THEN
      X = X - 1 :
      ' OFF SCREEN TO RIGHT
130 IF X < 0
     THEN
      X = X + 1 :
```

```
     ' OFF SCREEN TO LEFT
140 IF Y > 44
     THEN
      Y = Y - 1 :
      ' OFF SCREEN AT BOTTOM
150 IF Y < 0
     THEN
      Y = Y + 1 :
      ' OFF SCREEN AT TOP
155 :
    ' ** IF CLEAR KEY PRESSED, SAVE POSITION **
160 SET(X,Y):
    IF C = 2
     THEN
      C = 0:
      FOR T = 1 TO 100:
       NEXT :
      RETURN
170 IF C < 120 RESET(X,Y) :
    ' **IF SPACE BAR NOT DOWN, RESET X,Y**
175 :
    ' ** LETTER "S" OR LETTER "L" PRESSED **
180 IF PEEK(14340) = 8
     THEN
      220:
     ELSE
      IF PEEK(14338) = 16 GOTO 290
185 :
    ' ** LETTER "C" OR LETTER "E" PRESSED **
190 IF PEEK(14337) = 8 CLS :
    GOSUB 380:
    X = 0:
    Y = 0:
    GOTO 40 :
     ELSE
      IF PEEK(14337) = 32 GOTO 210
200 GOTO 70
210 CLEAR 50:
    CLS :
    END
215 :
    '  ** STORE SCREEN DISPLAY IN A$ ARRAY **
220 CLEAR 5000:
    ON ERROR GOTO 340:
    DIM A$(15):
    L = 15360
230 FOR R = 0 TO 14:
     A$(R) = STRING$(64,32):
     NEXT :
    ' SET UP DUMMY STRING
235 :
    '  ** GET LSB AND MSB OF A$ ARRAY **
240 FOR R = 0 TO 14:
     B = 0:
     D = 0:
     B = VARPTR(A$(R)):
     D = PEEK(B + 2) * 256 + PEEK(B + 1)
245  :
     '  ** CONVERT NUMBER TO NEGATIVE FOR MEMORY GREATER THAN 16K
250  IF D > 32767
      THEN
       D = D - 65536
255  :
     '  ** PACK SCREEN VALUES INTO DUMMY STRING **
260  FOR I = D TO D + 63:
      POKE I, PEEK(L):
      L = L + 1:
      NEXT I:
     NEXT R
265 :
    '
```

Program continued

```
    ** NAME FILE FOR SAVE **
270 PRINT @960,"ENTER FILESPEC: ";:
    FL = 22:
    GOSUB 690:
    F$ = "":
    F$ = IN$
280 PRINT @896,A$(14);:
    OPEN "O",1,F$:
    FOR R = 0 TO 14:
     PRINT #1,A$(R):
     NEXT :
    CLOSE :
    X = 0:
    Y = 0:
    ON ERROR GOTO 370:
    GOSUB 380:
    GOTO 40
290 CLEAR 5000:
    DIM A$(15):
    ON ERROR GOTO 340 :
    ' LOAD
295 :
    '  ** NAME FILE FOR LOAD **
300 PRINT @960,"ENTER FILESPEC: ";:
    FL = 22:
    GOSUB 690:
    F$ = "":
    F$ = IN$
310 OPEN "I",1,F$:
    ON ERROR GOTO 340:
    R = 0:
    CLS :
    GOSUB 380:
    PRINT @0,"";
320 IF EOF (1) CLOSE :
    X = 0:
    Y = 0:
    ON ERROR GOTO 370:
    GOSUB 380:
    GOTO 40
330 LINE INPUT #1,A$(R):
    PRINT A$(R);:
    R = R + 1:
    GOTO 320
340 IF ERR / 2 = 53 OR ERR / 2 + 1 = 53 PRINT @976,"FILE NOT FOUND";
    :
    GOTO 360
350 PRINT @976,"DISK I/O ERROR" ERR / 2"IN LINE" ERL ;
360 CLOSE :
    FOR T = 1 TO 1000:
     NEXT
370 GOSUB 380:
    X = 0:
    Y = 0:
    RESUME 40
380 PRINT @960, STRING$(63,131);:
    FOR Z = 125 TO 127:
     SET(Z,45):
     NEXT :
    RETURN
390 :
    '  ** DRAW A STRAIGHT LINE BETWEEN TWO COORDINATES **
400 IF ABS(X2 - X1) < ABS(Y2 - Y1) GOTO 550
410 DY = (Y2 - Y1) / ABS(X2 - X1)
420 IF X2 > X1 GOTO 490
430 FOR I = X1 TO X2 STEP - 1
440  SET(I,Y1)
450  Y1 = Y1 + DY
460  IF Y1 < 0
      THEN
```

```
      Y1 = 0
470  NEXT I
480 RETURN
490 FOR I = X1 TO X2
500  SET(I,Y1)
510  Y1 = Y1 + DY
520  IF Y1 < 0
      THEN
       Y1 = 0
530  NEXT I
540 RETURN
550 DX = (X2 - X1) / ABS(Y2 - Y1)
560 IF Y2 > Y1 GOTO 630
570 FOR I = Y1 TO Y2 STEP - 1
580  SET(X1,I)
590  X1 = X1 + DX
600  IF X1 < 0
      THEN
       X1 = 0
610  NEXT I
620 RETURN
630 FOR I = Y1 TO Y2
640  SET(X1,I)
650  X1 = X1 + DX
660  IF X1 < 0
      THEN
       X1 = 0
670  NEXT I
680 RETURN
690 :
    ' **BLINKING CURSOR**
700 IN$ = "":
    FL$ = INKEY$:
    W = 0:
    IF FL = W
     THEN
      FL = 1
710 PRINT CHR$(14);:
    FOR T = 1 TO 25:
     FL$ = INKEY$:
     IF FL$ < > ""
      THEN
       730:
      ELSE
       NEXT
720 PRINT CHR$(15);:
    FOR T = 1 TO 25:
     FL$ = INKEY$:
     IF FL$ < > ""
      THEN
       730:
      ELSE
       NEXT :
      GOTO 710
730 IF W = FL AND FL$ < > CHR$(8) AND FL$ < > CHR$(13) AND FL$
    < > CHR$(24) PRINT CHR$(15);:
    GOTO 710
740 :
    '   ** "ENTER" PRESSED BEFORE END OF FIELD **
750 IF FL$ = CHR$(13) AND W < FL
     THEN
      PRINT STRING$(FL - W,131); CHR$(15);:
      RETURN
760 :
    '   ** TRIED TO BACKSPACE BEYOND START OF FIELD POSITION **
770 IF FL$ = CHR$(8) AND W < = 0 PRINT CHR$(15):
    GOTO 710
780 :
    '   ** BACKSPACE **
790 IF FL$ = CHR$(8) AND W > OIN$ = LEFT$(IN$, LEN(IN$) - 1):
```

Program continued

```
    W = W - 1:
    PRINT FL$;:
    POKE 16418,131:
    GOTO 710
800 :
    ' ** SHIFT-LEFT ARROW **
810 IF FL$ = CHR$(24) PRINT STRING$(W,8); STRING$(W,131); STRING$(W,
    24);:
    GOTO 700
820 IN$ = IN$ + FL$:
    W = W + 1:
    PRINT FL$;
830 IF FL = 1 OR FL$ = CHR$(13) PRINT CHR$(15);:
    RETURN :
     ELSE
      710
```

Program Listing 2. *New Compu-Sketch, tape version*

```
 1 :
   ' ***********************************************************
 2 :
   ' *                                                         *
 3 :
   ' *                   NEW COMPU-SKETCH                      *
 4 :
   ' *                     PHIL BURTON                         *
 5 :
   ' *                 1251 WAVERLY DRIVE                      *
 6 :
   ' *           DAYTONA BEACH, FLORIDA 32018                  *
 7 :
   ' *                   (904) 252-6911                        *
 8 :
   ' *                                                         *
 9 :
   ' ***********************************************************
10 CLEAR 100:
   ON ERROR GOTO 370
20 CLS :
   X = 0:
   Y = 0:
   GOSUB 380:
   PRINT @960,"DO YOU WANT TO LOAD AN EXISTING FILE? ";:
   FL = 1:
   GOSUB 690:
   IF IN$ = "Y"
    THEN
     GOSUB 380:
     GOTO 290
30 CLS :
   GOSUB 380
40 GOSUB 70:
   X1 = X:
   Y1 = Y :
   ' SAVE FIRST X,Y COORDINATES
50 GOSUB 70:
   X2 = X:
   Y2 = Y :
   ' SAVE SECOND X,Y COORDINATES
60 GOSUB 400:
   GOTO 40
70 C = PEEK(14400)
80 IF C AND 8
    THEN
```

```
      Y = Y - 1 :
      ' UP ARROW
 90 IF C AND 16
     THEN
      Y = Y + 1 :
      ' DOWN ARROW
100 IF C AND 32
     THEN
      X = X - 1 :
      ' LEFT ARROW
110 IF C AND 64
     THEN
      X = X + 1 :
      ' RIGHT ARROW
120 IF X > 127
     THEN
      X = X - 1 :
      ' OFF SCREEN TO RIGHT
130 IF X < 0
     THEN
      X = X + 1 :
      ' OFF SCREEN TO LEFT
140 IF Y > 44
     THEN
      Y = Y - 1 :
      ' OFF SCREEN AT BOTTOM
150 IF Y < 0
     THEN
      Y = Y + 1 :
      ' OFF SCREEN AT TOP
155 :
    ' ** IF CLEAR KEY PRESSED, SAVE POSITION **
160 SET(X,Y):
    IF C = 2
     THEN
      C = 0:
      FOR T = 1 TO 100:
       NEXT :
      RETURN
170 IF C < 120 RESET(X,Y) :
    ' **IF SPACE BAR NOT DOWN, RESET X,Y**
175 :
    ' ** LETTER "S" OR LETTER "L" PRESSED **
180 IF PEEK(14340) = 8
     THEN
      220:
     ELSE
      IF PEEK(14338) = 16 GOTO 290
185 :
    ' ** LETTER "C" OR LETTER "E" PRESSED **
190 IF PEEK(14337) = 8 CLS :
    GOSUB 380:
    X = 0:
    Y = 0:
    :
    GOTO 40 :
     ELSE
      IF PEEK(14337) = 32 GOTO 210
200 GOTO 70
210 CLEAR 50:
    CLS :
    END
215 :
    '  ** STORE SCREEN DISPLAY IN A$ ARRAY **
220 CLEAR 5000:
    ON ERROR GOTO 370:
    DIM A$(15):
    L = 15360
230 FOR R = 0 TO 14:
     A$(R) = STRING$(64,32):
```

Program continued

```
     NEXT :
    ' SET UP DUMMY STRING
235 :
    '  ** GET LSB AND MSB OF A$ ARRAY **
240 FOR R = 0 TO 14:
     B = 0:
     D = 0:
     B = VARPTR(A$(R)):
     D = PEEK(B + 2) * 256 + PEEK(B + 1)
245  :
     '  ** CONVERT NUMBER TO NEGATIVE FOR MEMORY GREATER THAN 16K
250  IF D > 32767
      THEN
       D = D - 65536
255  :
     '  ** PACK SCREEN VALUES INTO DUMMY STRING **
260  FOR I = D TO D + 63:
      POKE I, PEEK(L):
      L = L + 1:
      NEXT I:
     NEXT R
265 :
    '  ** NAME FILE FOR SAVE **
270 PRINT @960,"PREPARE TAPE RECORDER, THEN PRESS ENTER ";:
    FL = 1:
    GOSUB 690 :
    '  SAVE
280 GOSUB 380:
    FOR R = 0 TO 14:
     PRINT # - 1,A$(R):
     NEXT :
    GOTO 40
290 CLEAR 1000:
    DIM A$(15):
    ON ERROR GOTO 370 :
    ' LOAD
300 PRINT @960,"PREPARE TAPE FOR PLAYBACK, THEN PRESS ENTER  ";:
    FL = 1:
    GOSUB 690:
    GOSUB 380
310 FOR R = 0 TO 14:
     INPUT # - 1,A$(R):
     NEXT R
320 CLS :
    GOSUB 380:
    PRINT @0,;:
    FOR R = 0 TO 14:
     IF LEN(A$(R) ) < 64
      THEN
       T = 64 - LEN(A$(R)):
       PRINT TAB(T)A$(R);:
       NEXT :
     ELSE
      PRINT A$(R);:
      NEXT :
      GOTO 40
370 GOSUB 380:
    X = 0:
    Y = 0:
    RESUME 40
380 PRINT @960, STRING$(63,131);:
    FOR Z = 125 TO 127:
     SET(Z,45):
     NEXT :
    RETURN
390 :
    '  ** DRAW A STRAIGHT LINE BETWEEN TWO COORDINATES **
400 IF ABS(X2 - X1) < ABS(Y2 - Y1) GOTO 550
410 DY = (Y2 - Y1) / ABS(X2 - X1)
420 IF X2 > X1 GOTO 490
```

```
430 FOR I = X1 TO X2 STEP - 1
440  SET(I,Y1)
450  Y1 = Y1 + DY
460  IF Y1 < 0
      THEN
       Y1 = 0
470  NEXT I
480 RETURN
490 FOR I = X1 TO X2
500  SET(I,Y1)
510  Y1 = Y1 + DY
520  IF Y1 < 0
      THEN
       Y1 = 0
530  NEXT I
540 RETURN
550 DX = (X2 - X1) / ABS(Y2 - Y1)
560 IF Y2 > Y1 GOTO 630
570 FOR I = Y1 TO Y2 STEP - 1
580  SET(X1,I)
590  X1 = X1 + DX
600  IF X1 < 0
      THEN
       X1 = 0
610  NEXT I
620 RETURN
630 FOR I = Y1 TO Y2
640  SET(X1,I)
650  X1 = X1 + DX
660  IF X1 < 0
      THEN
       X1 = 0
670  NEXT I
680 RETURN
690 :
    ' **BLINKING CURSOR**
700 IN$ = "":
    FL$ = INKEY$:
    W = 0:
    IF FL = W
     THEN
      FL = 1
710 PRINT CHR$(14);:
    FOR T = 1 TO 25:
     FL$ = INKEY$:
     IF FL$ < > ""
      THEN
       730:
      ELSE
       NEXT
720 PRINT CHR$(15);:
    FOR T = 1 TO 25:
     FL$ = INKEY$:
     IF FL$ < > ""
      THEN
       730:
      ELSE
       NEXT :
      GOTO 710
730 IF W = FL AND FL$ < > CHR$(8) AND FL$ < > CHR$(13) AND FL$
    < > CHR$(24) PRINT CHR$(15);:
    GOTO 710
740 :
    '   ** "ENTER" PRESSED BEFORE END OF FIELD **
750 IF FL$ = CHR$(13) AND W < FL
     THEN
      PRINT STRING$(FL - W,131); CHR$(15);:
      RETURN
760 :
    '   ** TRIED TO BACKSPACE BEYOND START OF FIELD POSITION **
```

Program continued

```
770 IF FL$ = CHR$(8) AND W < = 0 PRINT CHR$(15):
    GOTO 710
780 :
    '  ** BACKSPACE **
790 IF FL$ = CHR$(8) AND W > OIN$ = LEFT$(IN$, LEN(IN$) - 1):
    W = W - 1:
    PRINT FL$;:
    POKE 16418,131:
    GOTO 710
800 :
    ' ** SHIFT-LEFT ARROW **
810 IF FL$ = CHR$(24) PRINT STRING$(W,8); STRING$(W,131); STRING$(W,
    24);:
    GOTO 700
820 IN$ = IN$ + FL$:
    W = W + 1:
    PRINT FL$;
830 IF FL = 1 OR FL$ = CHR$(13) PRINT CHR$(15);:
    RETURN :
     ELSE
      710
```

HARDWARE

As You Like It

Add PROM Capability to Your TRS-80 with the PR-80

HARDWARE

As You Like It

by Nick Doble

The DATA statement allows you to store almost any type of information, just by entering it. If you enter a lot of data for your own programs or from published programs, however, you are well aware of the trials and tribulations of data entry.

The numeric keypad on the TRS-80 makes numerical data entry much easier, but unfortunately does not provide the comma needed for DATA statements. You must use the comma on the regular keyboard; any aspirations of touch typing do not survive the extended journey from the numeric keypad to the comma on the letter keyboard.

It is a simple process to cut the traces to the period key on the numeric keypad and attach a pair of wires to each side of the severed traces, attach a third pair of wires to the comma key on the regular keypad, and finally, attach the ends of these pairs of wires to a DPDT (double pole/double throw) switch. By throwing the switch, you can have the period key on the numeric keypad represent either a comma or a period, as you like it.

The following instructions for this modification are more complete than seasoned hardware enthusiasts will need, but the novice will find them useful. While the instructions are for the Model I TRS-80, they should work for the Model III as well, although the physical layout will be somewhat different. You will need the items listed in Table 1. Using separate wire pairs makes installation difficult. I recommend that your six-wire cable use solid wire to simplify installation. Strip six wires from the cable you order if it has more than six wires, then separate and strip the cable as shown in Figure 1.

12 inches of six-wire flat cable (RS part # 278-771)
One DPDT (double pole/double throw) switch (RS part # 275-1546)
Matte (utility) knife
25–40 watt soldering iron (preferably battery powered), and solder
Drill and 1/4 inch bit
Towel or soft cloth

Table 1. *Material needed for comma/period modification*

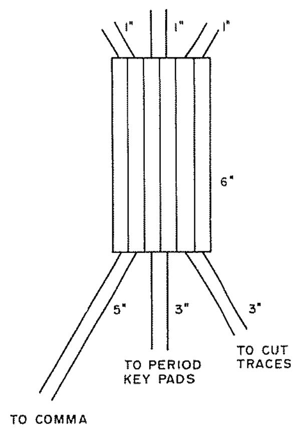


Figure 1. *Six-wire flat cable construction*

To begin, turn off all the components of your computer and unplug the ac lines to the CPU (keyboard) and the expansion interface if you have one. Disconnect all plugs from the CPU. Place a towel or soft cloth on the table in front of the CPU and turn the CPU over on top of it. Remove the six screws on the bottom of the CPU case. Each pair of screws is a different length since the CPU case is sloped. The longer screws will later go back in the longer section of the case, and so on. You will find a warning label over one of the screws. If your TRS-80 is still within its limited 90-day warranty, you should heed the warning and wait to make this modification.

Turn the CPU over again and lift off the top of the case. Holding the board at the edges, slowly pull up on the circuit board holding the two keypads. Carefully fold it over toward you and lay it face down on the towel, being very careful not to pull on the multi-wire connector on the left of the board. Under the keypad board you have just removed you will find another PCB (printed circuit board). We will not make any connections to it, and you should be very careful not to damage it or splash solder on it. Cover it with a piece of paper or another towel.

You will notice a period symbol on the right back of the key PCB above two printed circuit pads. ENTER and 0 will be printed to either side of the period symbol, each, again, near two PC pads (see Photo 1). You are looking at the traces to the bottom keys of the numeric keypad. The traces to the period key are shown in Photo 1 and Figure 2. Cut the traces where the Xs are indicated in Figure 2; do not cut the middle trace passing between the two pads. Cut the traces straight across using a matte (utility) knife and do not try to make the cut on the first try—instead, make several lighter cuts. Check that the cut has gone through the trace and that no pieces of the

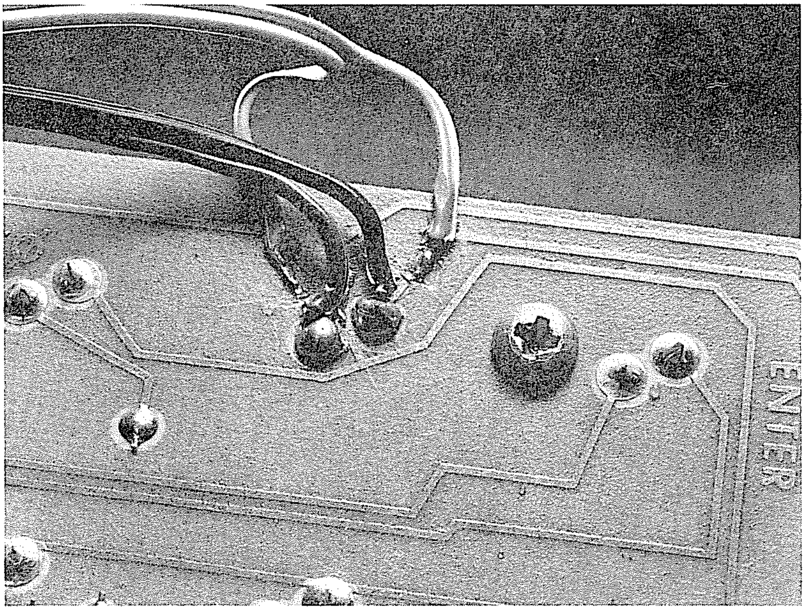


Photo 1. *The traces to the period key on the numeric keypad*

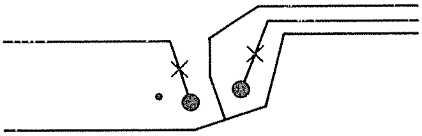


Figure 2. *The period key area. X marks the spot to cut.*

severed traces are touching each other or anything else. Now, with the tip of your matte knife, lightly scrape about one eighth of an inch of the edges of the two severed traces until they are shiny. Wet these areas with a little solder and then solder the two short wires from the outside edge of your six-wire cable to the two traces cut from the pads. Solder the two inner wires of this cable to the two pads of the period key. Use as little heat as possible for a good connection. (A good connection is a shiny connection.)

If you look to the left, you will notice at the top of the PCB as you are looking at it (actually the bottom of the board when it is in the case) the various symbols for the keys attached to the other side of the board. Locate the < and , symbols (which will be upside down) and attach the remaining pair of wires to the two pads below these symbols. Photo 2 and Figure 3 show how this area will look. Carefully check all the connections you have made to be sure that the wires and their connections touch only what they are supposed to touch.

Now you must connect the other end of the six-wire cable to the DPDT

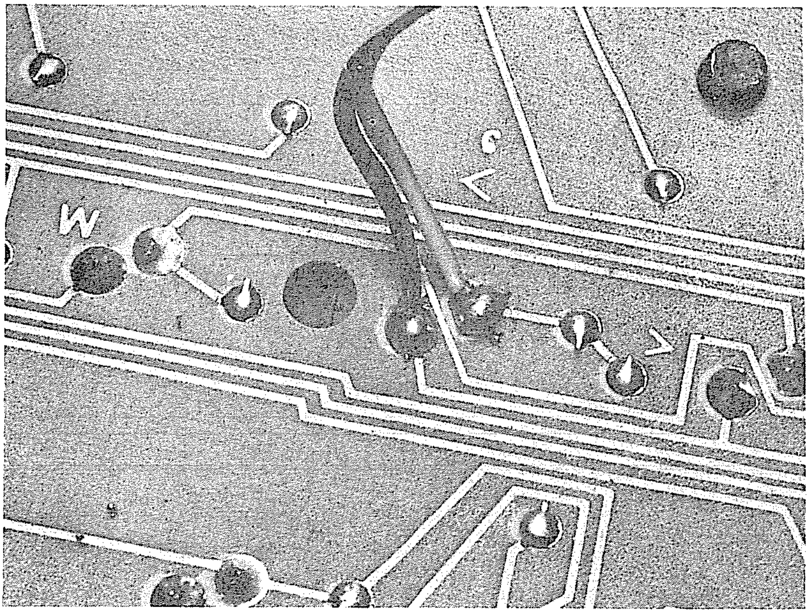


Photo 2. *The traces to the comma key on the main keyboard*

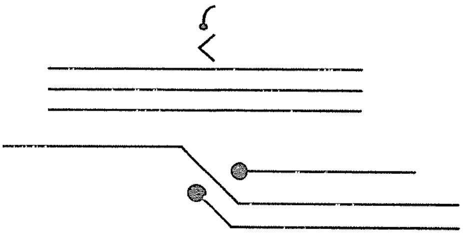


Figure 3. *The comma key area*

switch. As the period key only serves to short two traces (or wires) together, there is no need to observe any polarities. Wire the three wire pairs to the switch as indicated in Figure 4. It is a good idea to keep the period and comma wires as far from each other as possible, as I have done by putting the key wire pair between the other two pairs. Figure 5 is a schematic of this process.

Looking at Photo 3 you will see that I have mounted the switch on the case keyboard top in the space between the letter and numeric keypads. (The other switch in the photo is my upper/lowercase switch.) Avoid locating the switch at the bottom of the keyboard as it will get in the way during typing. Drill a 1/4 inch hole for the switch in the case top at the location you choose and install the switch.

Carefully fold the keyboard PCB back into the case, removing the paper or towel you placed over the main PCB. The six-wire cable should come out from under the bottom of the board as shown in Photo 4. Put the PCB on top

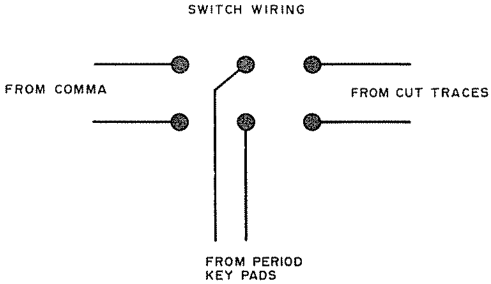


Figure 4. *DPDT (double pole/double throw) switch wiring*

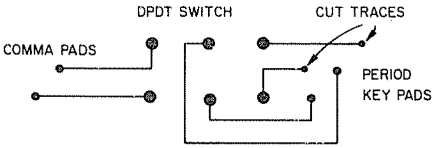


Figure 5. *Schematic*

Photo 3. *The period/comma switch mounted on the CPU. (The upper switch is an upper/lower-case switch.)*

of the supports and spacers. Finally, fold the case top up and over this whole assembly and fit it into the bottom of the case, making sure that the cable to the switch does not interfere with the keys on either side of it or get caught between the case top and bottom in front. Turn the case over on the towel and install the three pairs of screws.

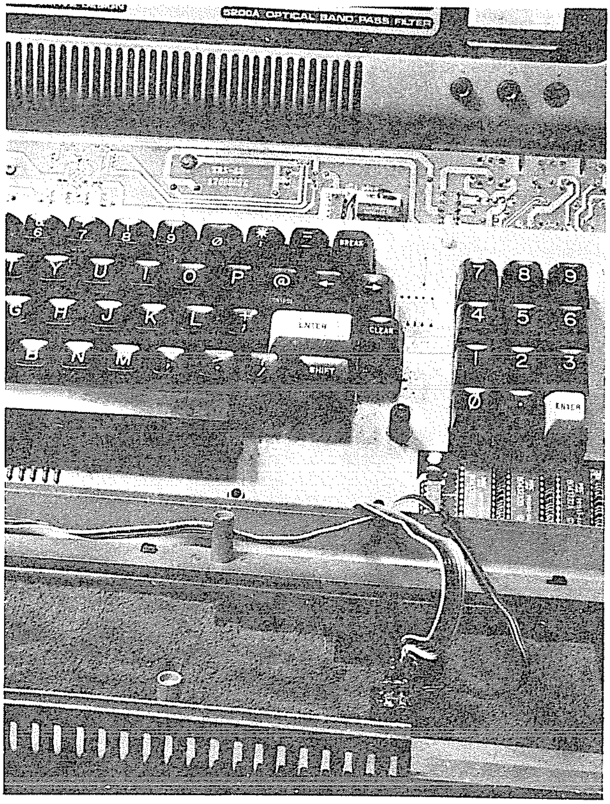


Photo 4. *Wiring from the keyboard to the period/comma switch*

When you have done this, reconnect the CPU and plug in any ac lines you disconnected. Turn on your computer and make sure it is operating normally. If it is, switch the period/comma switch back and forth while pressing the period key on the numeric keypad. This key should print either a comma or a period depending on the position of the switch. There should be no other effects from this wiring change, such as signs of instability or extraneous characters. If there are, you should check your connections and the routing of your six-wire cable to keep it away from other components. I have oriented my period/comma switch so that it is OFF, or in the period mode, when it is down.

Add PROM Capability to Your TRS-80 with the PR-80

by Frank Delfine

The project presented here describes the hardware and software required to place any block of memory (up to 8K) into 2708 EPROMs to be called via the BASIC SYSTEM utility. This PROM card, dubbed the PR-80, is designed to be constructed on an S-100 type plugboard so that it is compatible with the Deluxe Expansion Interface described in Volume 3 of the *Encyclopedia for the TRS-80*. To maintain compatibility with Radio Shack EIs, as well as the standard keyboard connections, I have included the data needed to interface to these as well. In addition to programming PROMs for storing TRS-80 utilities, you will now have the capability of burning programs for microprocessor-based devices.

Board Architecture

The PR-80 is divided into two sections: the 2708 programmer, and the 8K of PROM sockets along with their associated address decoding circuitry. (See Figures 1 and 2.) The PROMs reside in 8K of high memory at addresses E000H to FFFFH. A PROM programmed with the PR-80 may be placed in any of these sockets and accessed by entering the SYSTEM mode in BASIC and typing:

/ DECIMAL START ADDRESS

Typing / 57344, for example, will transfer program control to E000H where a user program can reside and execute. Use Table 1 to locate a starting address for a particular socket. The programmer consists of the following sections:

1) + 26-volt power supply
2) Program pulse switching circuitry
3) 24 line I/O port
4) I/O port decoding circuitry

EPROM Operation

The 2708 (see Figure 3) is organized as a block of 1024 × 8 bit words of memory. Accessing this data requires 10 address bits (A0–A9). To allow the chip to sit directly on a system data bus, place the data lines into a high impedance state via the chip select pin (pin 20). During a read operation, the device functions much like a read from a static RAM chip. An address is put out to pins A0–A9; the chip select is brought low (TTL 0); and, after the specified access time (450 ns maximum from valid address for the standard part), the data appears at the data pins D0–D7. (See the timing diagram in Figure 4.)

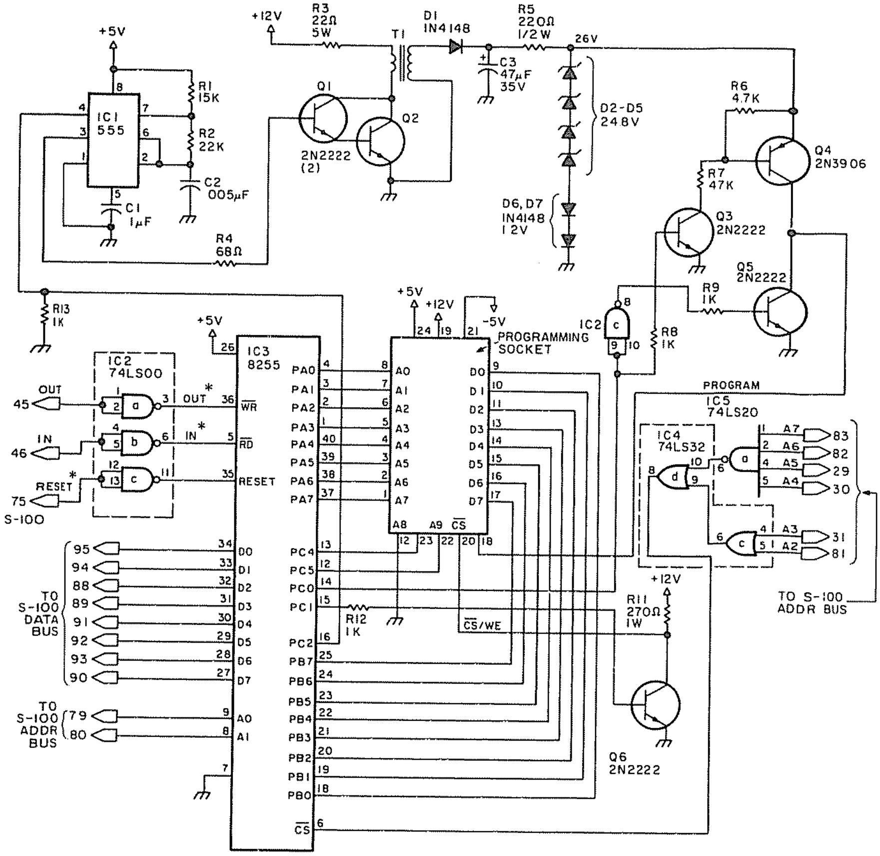


Figure 1. *2708 programmer*

A blank, unprogrammed 2708 has all its data bits set to a logical 1. (All bytes = FFH.) Logical 0 must be programmed into the desired bit locations in order to make the PROM useful to us. The only way to return the bit pattern to all 1s (erased condition) is to expose the chip's memory array to a strong, ultraviolet light source with a wavelength in the 2537 angstrom region. This is done through a small quartz window on the surface of the chip package. You can obtain the UV PROM eraser from the distributors listed in Table 3.

Prepare the chip for the programming operation by bringing the chip select pin to + 12 volts. (This now serves as the write enable.) An address is presented to the address pins A0–A9 just as it was for a read operation. A parallel eight-bit data word to be programmed is applied to data lines D0–D7 at TTL levels. A program pulse of about + 26 volts is then applied to

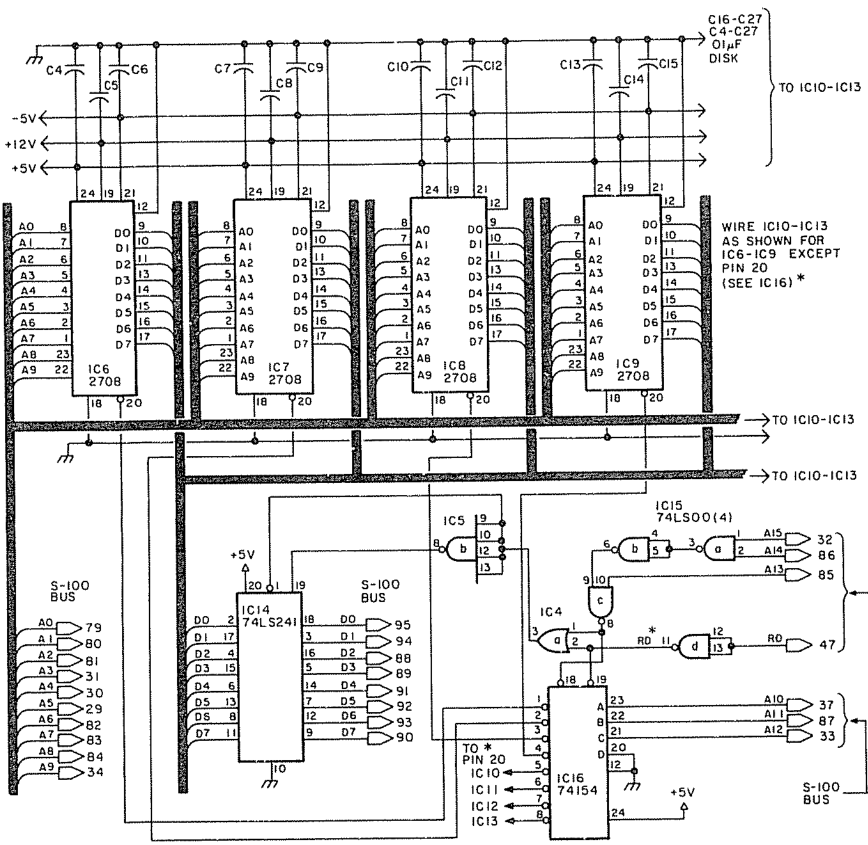


Figure 2. *8K PROM sockets*

the PROGRAM pin (pin 18). You must repeat this operation many times, sequencing through all 1024 addresses each time. The exact number of loops through the 1K bytes depends on the program pulse width used. The number may be calculated from the following relationship:

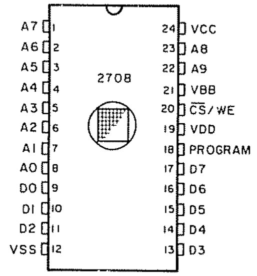
N(# of passes) X Tpw >= 100 ms

The PR-80 uses a pulse width (Tpw) of approximately .5 ms; therefore, N >= 100/.5, or at least 200 passes.

To program the PROM successfully, we must somehow sequence through the 1024 addresses, presenting the data that we wish to program at each address at least 200 times. The machine-code program PROM/CMD that controls the PR-80 takes approximately four minutes to burn all 1K bytes of a 2708.

PROM Socket	Hex Address		Decimal Address	
	Start	End	Start	End
IC6	E000	E3FF	57344	58367
IC7	E400	E7FF	58368	59391
IC8	E800	EBFF	59392	60415
IC9	EC00	EFFF	60416	61439
IC10	F000	F3FF	61440	62463
IC11	F400	F7FF	62464	63487
IC12	F800	FBFF	63488	64511
IC13	FC00	FFFF	64512	65535

Table 1. *Socket starting address*



A0-A9	ADDRESS INPUTS
D0-D7	DATA OUTPUTS/INPUTS
$\overline{CS}$/WE	CHIP SELECT/WRITE ENABLE INPUT

	VSS	PROGRAM	$\overline{CS}$/WE	VDD	VBB	VCC
READ	GND	GND	GND	+12V	-5V	+5V
PROGRAM	GND	PULSED + 26V	+12V	+12V	-5V	+5V

Figure 3. *2708 pinout definitions*

Circuit Description

The +26-volt supply generates the voltage for the programming pulse to pin 18 of the 2708. It is a dc/dc converter which uses a +12-volt supply as its input. The +12 volts are taken from the on-board regulator in the S-100 version and from the 12-volt supply shown in Figure 5 in the stand-alone version. IC1 in Figure 1, a 555, is connected as a gated oscillator. It generates a 5.7 kHz square wave that is used to switch Q1 and Q2. This oscillator is gated on or off by controlling pin 4. When this signal is low, the oscillator is inhibited. A high on this pin lets the oscillator run. The switching of Darlington pair Q1 and Q2 causes a series of current pulses approximately 500 mA in amplitude to flow through T1's primary. This induces a

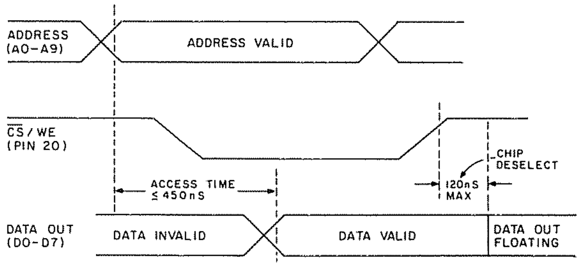


Figure 4. *Timing diagram*

voltage into the secondary circuit which is approximately 60 volts under no-load conditions. This occurs due to the turns ratio of the transformer which is approximately 100 to 1. Since the exact voltage you will get out of the circuit is dependent on the transformer you use, I have chosen a component that is available at your local Radio Shack store.

Diode D1 rectifies the pulses from T1's secondary and charges the filter capacitor C3 to the peak voltage. Since C3 charges to well over 26 volts and will vary with the load that it must supply (the 2708), we must provide a means for regulating the voltage to 26 volts under varying load conditions. That is the function of D2–D7 and R5. D2–D5 are 6.2-volt zener diodes whose drops are added together to give us 24.8 volts. Since this is below the

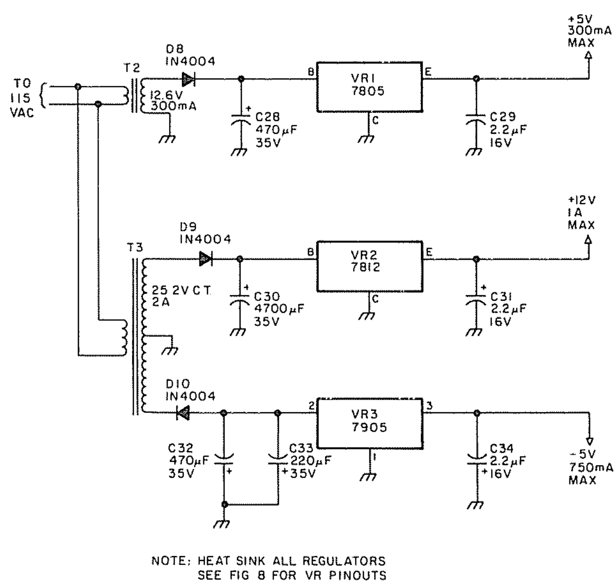


Figure 5. *Stand-alone power supply*

lower end of the allowed programming voltage tolerance band, two standard general purpose diodes, D6 and D7, are added in series with the Zeners to give an extra 1.2 volts for a total of 26.0 volts. Other combinations of Zener diodes may be used if they are on hand. Be sure to watch the power dissipation and keep the output voltage between 25 and 27 volts.

There are four critical components/parameters which should not be modified. First, you must use the specified transformer for T1. The oscillator frequency is also somewhat critical and should be kept close to 5.7 kHz. R3 is critical in both value and power rating. This resistor gets quite hot when the oscillator is running. It should be mounted so that it is supported off of the circuit board. The last component that should not be changed is R5. This value has been selected to provide the proper bias to the Zener string as well as allowing the proper current to flow to the 2708 during programming.

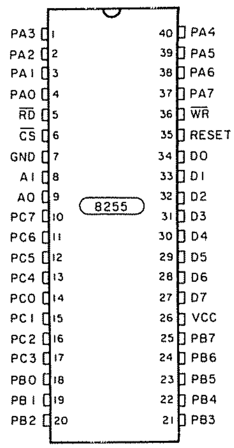
Since the circuit does have a few components which dissipate quite a bit of power when running, the oscillator shuts down when there are no programming operations going on. This is done via pin 4 on IC1. The software regulates this pin to turn the supply on when it wants to program and off when it has finished. The 1k resistor from pin 4 to ground ensures that the oscillator will power up in an OFF state when the computer is first turned on.

Program Pulse Switching Circuitry

You must control the 26-volt power source so that you can apply a pulse of 26 volts to the program pin on the 2708. The actual pulse width that is used is .5 ms, which is a value we derived earlier. The pulse is generated by a software loop which sets and resets pin 14 on the 8255. IC2-c inverts this signal so that there is a complementary set of TTL level program pulses available. The positive-going pulse turns on Q3 which causes Q4 to turn on. The 26 volts are now applied to the 2708 by Q4. Since the output of IC2-c is now low, Q5 is off and has no effect on the 2708. When the input to IC2-c goes low, Q3 and Q4 turn off (R6 helps Q4 to turn off quickly), removing the programming voltage from the 2708. Q5 now turns on and sinks any current provided by the 2708. Since there is some current being sourced by the 2708, this sort of active pull-down is necessary and should not be replaced by a passive pull-down scheme.

24-Line I/O Port

The actual interface between the programming hardware and the TRS-80 involves a single chip. The 8255 (see Figure 6) is a 40-pin device which contains three-eight bit ports which may be configured as input or output in any combination. The ports are called out as PORT A, PORT B, and PORT C. A and B are identical ports and may be configured as an eight-bit input port or an eight-bit output port. PORT C is split into two four-bit ports. The four high-order bits may be set as input bits or output bits, while the four low-order bits have the same option independent of the high-order bits. In addition, if PORT C serves as an output port, you have a bit set/reset capability available which allows you to set or reset one bit of the port without affecting any of the other bits of that port. This feature saves some software when you write control routines. There are several other modes and options available in this chip that I will not go into here. If you would like to know more about the device, see the Intel component data catalog for spec sheets and some applications information.



D0-D7	BI-DIRECTIONAL DATA BUS
RESET	RESET INPUT
$\overline{\text{CS}}$	CHIP SELECT INPUT
$\overline{\text{RD}}$	READ STROBE INPUT
$\overline{\text{WR}}$	WRITE STROBE INPUT
A0, A1	PORT SELECT ADDRESS (INPUT)
PA0-PA7	PORT A (I/O)
PB0-PB7	PORT B (I/O)
PC0-PC7	PORT C (I/O)
VCC	+5VOLTS
GND	0 VOLTS

Figure 6. *8255 pinout definitions*

A review of the 2708 pinout and the programming hardware described to this point leaves us with the following I/O requirements for programming and reading (we will want to read back the programmed PROM to verify it as a proper burn) the 2708:

1) 10 bits of address information
2) eight bits of data to be written or read
3) one bit to turn the programming voltage on and off
4) one bit to turn chip select on or off (choose read or write mode)
5) one bit to turn the 26-volt supply on and off

The following I/O assignments are then made to the 8255 ports:

Port A:	A0 through A7	/ Output Port
Port B:	D0 through D7	/ Output for program Input for a read
Upper Port C:	PC4 = A8 PC5 = A9 PC6 = Not used PC7 = Not used	/ Output Port
Lower Port C:	PC0 = Programming pulse PC1 = Chip select/write PC2 = PRGM volt on/off PC3 = Not used	/ Output Port

Note that all the ports but Port B are configured as output ports.

Since port configuration is software programmable in the 8255, this port can be switched from output to input simply by writing a word to the control register of the device. This feature saves buffers and associated control chips that would be necessary if you used discrete logic.

PC1 is the signal that controls the chip select pin (pin 20) of the 2708. I mentioned earlier that the chip select line must be brought to + 12 volts for a programming operation. Since the PC1 signal is a TTL level, Q6 and R11 serve to accomplish the level shifting.

I/O Port Decoding Circuitry

Most of the I/O functions in the TRS-80, such as the disk controller and the printer, are memory mapped. There are one or two devices which are arranged as I/O devices, and we must be careful not to conflict with them. The serial RS-232 port utilizes port addresses from E8 to EB. The cassette is also an I/O device at FF. I chose the ports from F0 to F3 as the slot for the PR-80. Table 4 lists the function of each port.

You must decode only the lower eight bits of the address bus for an I/O port. The upper eight address bits contain the data in the A register during an I/O transfer; so you can ignore that information. The A0 and A1 bits go directly into the 8255 and are decoded inside the chip to select one of four registers. The first three registers are ports A, B, and C, and the fourth is the control register. You must, therefore, decode only one word;

1111 00XX

for the four ports from F0 to F3. This is done by IC4-c, IC4-d, and IC5-a which generate a chip select signal any time a word between F0 and F3 appears on the lower half of the addrress bus. Since reading or writing to the PR-80 are not the only situations that will trigger this, an additional piece of data is supplied to the 8255 in the form of the IN* and OUT* control signals from the TRS-80 which make the device selection unique.

Designation	Description	RS P/N When Applies
IC1	555 Timer	276-1723
IC2,15	74LS00	276-1900
IC3	8255	
IC4	74LS32	276-1915
IC5	74LS20	
IC6-13	2708	
IC14	74LS241	
IC16	74154	276-1834
VR1	7805 +5V Reg	276-1770
VR2	7812 +12V REG	276-1771
VR3	7905 −5V REG	276-1773
R1	6.8K 1/4 W	271-1333
R2	22K 1/4 W	271-1339
R3	22-Ohm 5W (or two 10-Ohm 2W in series—271-080)	
R4	68-Ohm 1/4 W	271-010
R5	220-Ohm 1/2 W	271-015
R6	4.7K 1/4 W	271-1330
R7	47K 1/4 W	271-1342
R8,9,12	1K 1/4 W	271-1321
R11	270-Ohm 1W (or two 470-Ohm 1/2 W in parallel 271-091)	
C1	.1 uF/50V	272-1069
C2	.005uF/50V	272-126
C3	47 uF/35V	272-1015
C4–C27	.01uF/50V	272-131
C28,32	470uF/35V	272-1018
C29,31,34	2.2uF/16V	272-1420
C30	4700 uF/35V	272-1022
C33	220uF/35V	272-1017
Q1,2,3,5,6	2N2222	276-1617
Q4	2N3906	276-1604
D1,6,7	IN4148	276-1103
D2,3,4,5	IN4735 (6.2V Zener)	276-561
D8,9,10	IN4004	276-1103
T1	Mini Audio Transformer (1K to 8-Ohm)	273-1380
T2	12V 300 mA	273-1385
T3	25.2V 2A	273-1512

Table 2. *Parts list for PR-80*

the code using a modified version of EDTASM. (See "Assemble It Yourself" by Richard Koch in *80 Microcomputing*, December 1980, p. 212.) This version of EDTASM is a perfect adjunct for the PR-80 since it allows you to assemble your program directly into R/W memory, run and debug it, and leave a working version in RAM. You can then call PROM/CMD and burn your code directly into a PROM. This program can be run from the disk or be saved in PROM. I did this with PROM/CMD. Anytime I want to run the programmer all I have to do is enter either Disk BASIC or Level II BASIC, type SYSTEM, and answer the prompt with / 57344, and the PR-80 program prompt appears on the screen.

I made extensive use of the Level II screen and keyboard I/O routines to keep the code down to a minimum but still provide the same type of user prompting and input flexibility that is common with BASIC programming. I have tried to provide some information as to what parameters must be

Programming Socket 40-pin Zero Insertion Force
Available from:

1) Priority One Electronics
   9161 Deering Ave.
   Chatsworth, CA. 91311 — 1-800-423-5922

2) Jameco Electronics
   1355 Shoreway Rd.
   Belmont, CA. 94002 — 415-592-8097

PROM Eraser
Available from:

1) Advanced Computer Products
   P.O. Box 17329
   Irvine, CA. 92713 — 800-854-8230

2) Jameco Electronics
   1355 Shoreway Rd.
   Belmont, CA. 94002 — 415-592-8097

3) Logical Devices, Inc.
   781 W. Oakland Park Blvd
   Ft. Lauderdale, FL 33311 — 305-565-8103

4) Quest Electronics
   P.O. Box 4430X
   Santa Clara, CA. 95054 — 408-988-1640

Table 3. *Distributors*

passed to the various routines in the comments. For a more detailed explanation of how these routines work and what registers are effected, see "Inside the ROMs" by Bruce E. Stock in *80 Microcomputing*, March 1980, p. 94.

Port Address	Function
F0	PORT A - A0–A7
F1	PORT B - D0 = D7
F2	PORT C - PC4 = A8
	PC5 = A9
	PC0 = PROGRAM PULSE
	PC1 = CS/WE
	PC2 = 26 V SUPPLY ON/OFF
F3	CONTROL PORT - SETS I/O MODE & USED FOR BIT SET/RESET

Table 4. *Port functions*

The program starts by clearing the screen, placing the program I.D. header up on the screen, and prompting the user to plug in a blank PROM. It then asks for the start address of the segment of code to be copied. The address should be entered as a four-character hex number. After the program has checked the entered address to make sure that the entries fall between the 0 and F characters for hex code format, the PROM is read to verify that it is a blank PROM before attempting a program operation. If it detects an invalid entry, the program jumps back to START to allow you to try again. It gets back to START via ERR, which pops two addresses off the stack before jumping to START. This is necessary since the program is down two levels in subroutines at that point. If this was not done, and a few entry errors were made, the stack would keep growing, and some of the return addresses from the subroutine calls would be incorrect.

If data other than FFH is read from the PROM, the programming operation is aborted, and a FAIL message appears. The computer then prompts you to make a decision to run the program again or to return to TRSDOS. You can return to a program other than DOS if you change the jump address in line 1540 of Program Listing 1.

If all the tests to this point are successful, the 8255 is ready for programming. By writing an 80H to the 8255 control register, the A, B, and C ports become output ports. Notice that the chip select, program pulse, and the 26-volt supply are also controlled by writing to the control register. This is because they are on PORT C and can be controlled by the bit set/reset feature of the chip. A list of the commands for the control register is given in Table 5. Since the 26-volt power supply filter capacitor C3 is rather large, and the supply does not provide a high charge current, C3 requires several hundred milliseconds to reach a full charge. The delay loops labeled DLY1 and DLY2 form the delay for this purpose.

Once the supply is running, the PROGRAMMING message appears, and the actual programming gets underway. The code start address is loaded into HL from the stack. This address was pushed onto the stack when it was read in from the keyboard previously. It is also saved in a temporary register called TEMP so that it will be available for future passes. (Remember that you must make 200 passes of the 1024 addresses to complete the burn.) DE is the address pointer to the PROM; so you must initialize it to zero. BC is the byte counter which is set to 1024. A value stored in a memory register called DELAY determines the program pulse width. The IX register (X Index Register) is set to point to this location. From this point on, the program loops through the 1024 addresses, outputting the data from memory to the PROM, then generating a program pulse. The subroutine PROMPT handles the conversion of the PROM address in the DE register to the B and C ports in the 8255. The routine PULSE generates the program pulse of .5 ms on and .5 ms off. After 1024 addresses have been sequenced, the pass counter is decremented and tested. If the sequence has been repeated 200 times, the programming operation is complete.

At this point, the 26-volt supply switches off and the 8255 is placed in a read mode. This time you are going to compare the contents of the PROM to the data in memory. PORTIN is the routine which handles the 8255 setup for an input operation. A VERIFYING message is displayed while the compare operation is going on. If a match is achieved, a PASS message is shown; if not, FAIL appears. At this point, the operation is complete, and you can remove the PROM and place it in one of the eight PROM sockets for use by the TRS-80.

8K PROM Section

The PROM area of the board consists of the eight PROMS (IC6–IC13), IC15, IC16, and IC4-a which make up the address decoder, and IC14 which is the bus buffer. The output of IC4-a goes low whenever a memory read is requested in the address range of E000H to FFFFH. This is used to control the buffer IC14. IC16 is arranged so that it splits the 8K block into eight 1K segments where each output is used as a chip select to one of the PROMs. (See Table 1.)

The bus buffer IC14 is especially important if you use the S-100 version of the board, since the output drivers in the PROM are not sufficient to pull the bus terminators on the motherboard down to ground. It also helps in the stand-alone version by providing increased drive capability for the connecting cable to the keyboard or EI. (See Figure 7.)

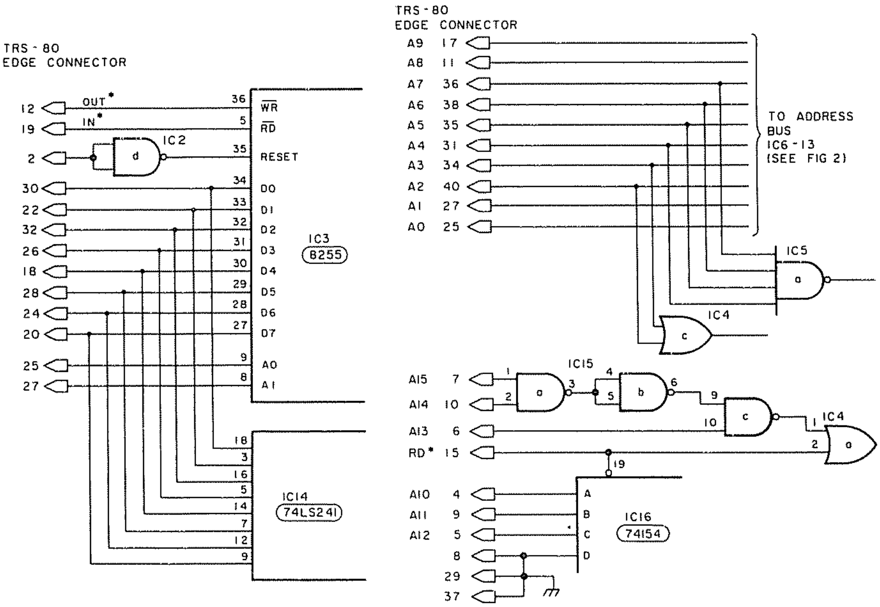


Figure 7. *Modifications for stand-alone version*

General Comments and Hints

The most accurate and most expedient way to enter the PROM/CMD program is to use an assembler such as EDTASM. You can also enter the program using the BASIC program shown in Program Listing 3. The code from 8FF1H to 8FFFH in Program Listings 1 and 3 can be eliminated if the program is only to be run from disk or tape. This code is provided to relocate the program from PROM down to R/W memory where it can be run. This scheme can be used for any program that was written for an absolute address in R/W memory, but that you would like to store in PROM. You can do this with programs like EDTASM and T-BUG. HL should be loaded with the start of the PROM code while DE should get the destination address in R/W memory. Place the number of bytes to be transferred into the BC register pair then execute an LDIR. This will cause a block transfer of code from PROM to R/W memory for the length specified in BC. A jump to the start address in R/W memory will now run the program as though it were loaded from a tape or disk. While this method provides a quick, easy means for transferring existing routines from tape or disk directly to PROM, you should avoid using it on new routines that you write, since it occupies double memory space. It is much more efficient to have the code execute directly in the PROM rather than make a second copy in R/W.

You can use wire-wrap techniques to construct the board. Locate the bypass capacitors specified in the schematics as close to the chips as you can. I solder them directly to the wire-wrap pins of the chip. Keep the cable between the keyboard and the PR-80 as short as possible in the stand-alone version to minimize inductive/capacitive effects that can cause strange program crashes for no apparent reason. I recommend that you use a ZIF (zero insertion force) socket for the programming socket. This type of socket allows you to insert the PROM without any force on the pins. A small handle on the side of the socket locks the chip into place. You will never bend a chip lead this way. If you build the S-100 version, you will probably want to mount the ZIF socket in a separate enclosure outside of the mainframe and connect it to the board with a length of 40-conductor ribbon cable (or a DIP jumper). The enclosure can be a small, plastic minibox.

The S-100 boards are VECTOR type 8801 plugboards. They are supplied with a heat sink for the 5-volt regulator and a paper layout guide which helps you to locate the pin numbers. (See Figure 8.) More information on interfacing the TRS-80 to the S-100 bus may be found in Volume 3 of the *Encyclopedia for the TRS-80*. I have split PROM/CMD into two parts to accommodate the buffer size of the version of EDTASM I was using at the time. If you have more buffer space, you can run both listings together and remove some of the redundant EQU statements.

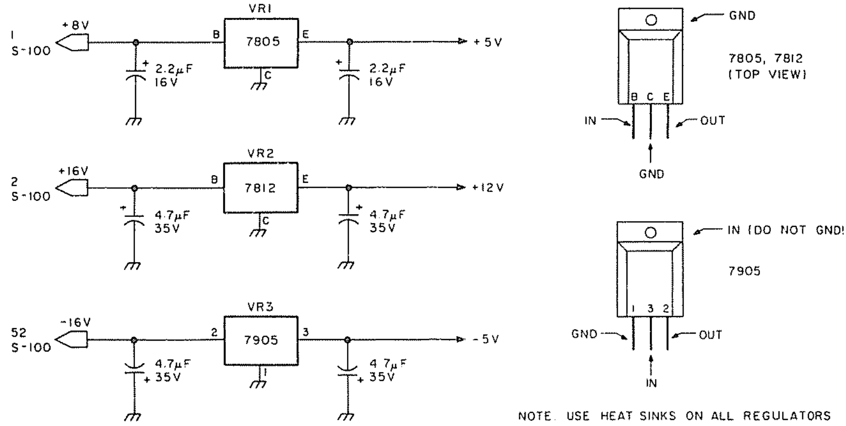


Figure 8. *S-100 power supply*

This table describes the basic commands that may be written to the control register (PORT F3H) of the 8255. The MODE 0,1, and 2 nomenclature refers to the following definitions:

MODE 0 : BASIC INPUT / OUTPUT
MODE 1 : STROBED INPUT / OUTPUT
MODE 2 : BI-DIRECTIONAL BUS

Note that D7 indicates whether the word is to be interpreted as a mode definition or as a bit SET/RESET command.
A read of the control port is not permitted in this device.

Mode Definition Format

D7 MODE SET FLAG 1 = ACTIVE
D6,5 MODE SELECTION FOR GROUP A (GROUP A = UPPER C AND PORT A)

D6	D5	
0	0	MODE 0
0	1	MODE 1
1	X	MODE 2

D4 PORT A 1 = INPUT / 0 = OUTPUT
D3 UPPER C 1 = INPUT / 0 = OUTPUT
D2 MODE SELECTION FOR GROUP B (GROUP B = LOWER C AND PORT B) 0 = MODE 0 / 1 = MODE 1
D1 PORT B 1 = INPUT / 0 = OUTPUT
D0 LOWER C 1 = INPUT / 0 = OUTPUT

Bit SET/RESET Format

D7 BIT SET / RESET FLAG 0 = ACTIVE
D6,5,4 DON'T CARE
D3,2,1 BIT SELECT

D3	D2	D1	BIT selected
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

D0 BIT SET/RESET SET = 1 / RESET = 0

Table 5. *Control register commands*

Program Listing 1. *PROM/CMD*

Encyclopedia Loader

```
                  00100 ;TITLE          PROM/CMD        8/17/81    VER 1.0
                  00110 ;
                  00120 ;THIS UTILITY, WHEN USED IN CONJUNCTION WITH THE PR-80
                  00130 ;PROM/ROM CARD IN A TRS-80 SYSTEM, WILL ALLOW THE
                  00140 ;PROGRAMMING OF 2708 TYPE EPROMS FROM R/W MEMORY OR
                  00150 ;FROM A MASTER PROM.
                  00160 ;
0080              00170 OUT82    EQU     80H            ;INIT 8255 FOR WRITE
0082              00180 IN82     EQU     82H            ;INIT 8255 FOR READ
00F0              00190 A8255    EQU     0F0H           ;PORT A - LOW ADDRESS
00F1              00200 B8255    EQU     0F1H           ;PORT B - DATA LINES
00F3              00210 CNTRL    EQU     0F3H           ;8255 CONTROL PORT
0001              00220 PLSON    EQU     1              ;PROG PULSE ON
0000              00230 PLSOFF   EQU     0              ;PROG PULSE OFF
0002              00240 WRON     EQU     2              ;WRITE SELECTED (CS=+12)
0003              00250 WROFF    EQU     3              ;READ SELECTED (CS=0)
0005              00260 PGMON    EQU     5              ;PRGM VOLTAGE ON (25 V)
0004              00270 PGMOFF   EQU     4              ;PRGM VOLTAGE OFF
91E4              00280 MSG1     EQU     91E4H          ;MESSAGE START ADDRESSES
9229              00290 MSG2     EQU     9229H
923D              00300 MSG3     EQU     923DH
9243              00310 MSG4     EQU     9243H
9249              00320 MSG5     EQU     9249H
9264              00330 MSG6     EQU     9264H
9276              00340 MSG7     EQU     9276H
928B              00350 MSG8     EQU     928BH
9150              00360 SHFT1    EQU     9150H          ;SUBROUTINE ADRS
9171              00370 LOWNIB   EQU     9171H
91B2              00380 PORTIN   EQU     91B2H
9178              00390 PROMPT   EQU     9178H
91C3              00400 PULSE    EQU     91C3H
8FF1              00410          ORG     8FF1H
8FF1 F3           00412          DI                     ;RELOCATE PROM/CMD & RUN
8FF2 210FE0       00414          LD      HL,0E00FH      ;SET START ADR IN PROM
8FF5 110090       00416          LD      DE,9000H       ;SET RELOCATE ADR IN RAM
8FF8 01AB02       00418          LD      BC,02ABH       ;SET TRANSFER LENGTH
8FFB EDB0         00420          LDIR                   ;TRANSFER TO RAM
8FFD C30090       00422          JP      9000H          ;EXECUTE PRGM IN RAM
9000 CDC901       00430 START    CALL    1C9H           ;LVL II SCREEN CLR ROUT
9003 21E491       00440          LD      HL,MSG1        ;SET POINTER TO MESSAGE
9006 CDA728       00450          CALL    28A7H          ;LVL II ROUT TO DISPLAY
9009 CDB31B       00460          CALL    1BB3H          ;GET KEY DATA (LVL II)
900C CD5091       00470          CALL    SHFT1          ;PUT START ADDRESS
900F 57           00480          LD      D,A            ;TOGETHER FROM ASCII
9010 CD7191       00490          CALL    LOWNIB         ;KEYBOARD DATA
9013 82           00500          ADD     A,D            ;PUT NIBBLES TOGETHER
9014 57           00510          LD      D,A            ;SAVE HIGH ORDER BYTE
9015 CD5091       00520          CALL    SHFT1          ;DO SAME FOR LOW ORDER
9018 5F           00530          LD      E,A
9019 CD7191       00540          CALL    LOWNIB
901C 83           00550          ADD     A,E
901D 5F           00560          LD      E,A            ;DE NOW CONTAIN START ADR
901E D5           00570          PUSH    DE             ;SAVE START ADDRESS
901F 212992       00580          LD      HL,MSG2        ;DISPLY "VERIFYING BLANK"
9022 CDA728       00590          CALL    28A7H
9025 CDB291       00600          CALL    PORTIN         ;SETUP 8255 FOR READ
9028 110000       00610          LD      DE,0           ;SET PROM START ADR
902B 010004       00620          LD      BC,1024        ;SET BYTE COUNTER
902E CD7891       00630 VERIF    CALL    PROMPT         ;OUTPUT DE ADR TO PROM
9031 00           00635          NOP                    ;DELAY FOR 8255
9032 DBF1         00640          IN      A,(B8255)      ;READ BYTE
9034 FEFF         00650          CP      0FFH           ;TEST IF = FFH
9036 C2E190       00660          JP      NZ,BAD         ;N.G. - ABORT
9039 13           00670          INC     DE             ;BUMP POINTER
903A 0B           00680          DEC     BC             ;DEC AND TEST BYTE CNTR
903B 79           00690          LD      A,C
```

```
903C FE00             CP    0
903E C22E90           JP    NZ,VERIF     ;NOT DONE YET.....
9041 78               LD    A,B
9042 FE00             CP    0
9044 C22E90           JP    NZ,VERIF     ;NOT YET !!!
9047 213D92           LD    HL,MSG3      ;DISPLAY "PASS" MESSAGE
904A CDA728           CALL  28A7H
904D 3E80             LD    A,OUT82      ;BLANK OK - NOW PROGRAM!
904F D3F3             OUT   (CNTRL),A    ;SET 8255 TO PROGRAM MODE
9051 3E02             LD    A,WRON       ;TURN ON WR LINE (+12 V)
9053 D3F3             OUT   (CNTRL),A
9055 3E00             LD    A,PLSOFF     ;TURN PRGM PULSE OFF
9057 D3F3             OUT   (CNTRL),A
9059 3E05             LD    A,PGMON      ;TURN +25 V SUPPLY ON
905B D3F3             OUT   (CNTRL),A
905D 3EFF             LD    A,OFFH       ;DELAY FOR +25 V SUPPLY
905F 06FF       DLY2  LD    B,OFFH       ;CAP TO REACH FULL CHARGE
9061 00         DLY1  NOP
9062 05               DEC   B
9063 C26190           JP    NZ,DLY1
9066 3D               DEC   A
9067 C25F90           JP    NZ,DLY2
906A 214992           LD    HL,MSG5      ;DISPLAY "PROGRAMMING"
906D CDA728           CALL  28A7H
9070 E1               POP   HL           ;LOAD START ADR INTO HL
9071 22FD90           LD    (TEMP),HL    ;SAVE FOR NEXT PASS
9074 0EC8             LD    C,200        ;SET PASS COUNTER
9076 C5               PUSH  BC           ;SAVE IT
9077 010004           LD    BC,1024      ;SET BYTE COUNT
907A DD21FF90         LD    IX,DELAY     ;SET POINTER TO PULSE
                                         ;DELAY WORD
907E 110000     AGN   LD    DE,0         ;INIT ADDRESS TO PROM
9081 CD7891     PRGM  CALL  PROMPT       ;OUT ADR TO PROM
9084 7E               LD    A,(HL)       ;GET DATA BYTE
9085 D3F1             OUT   (B8255),A    ;PUT IT OUT TO PROM
9087 CDC391           CALL  PULSE        ;GENERATE PRGM PULSE
908A 23               INC   HL           ;BUMP POINTERS
908B 13               INC   DE
908C 0B               DEC   BC           ;DEC AND TEST BYTE CNTR
908D 3E00             LD    A,0
908F B9               CP    C
9090 C28190           JP    NZ,PRGM      ;STILL MORE TO GO.....
9093 B8               CP    B
9094 C28190           JP    NZ,PRGM      ;NOT DONE YET....
9097 C1         PASS  POP   BC           ;CHECK FOR PASS  #
9098 0D               DEC   C
9099 CAA690           JP    Z,CHECK      ;PROGRAM COMPLETE- VERIFY
909C C5               PUSH  BC           ;MORE PASSES LEFT..SAVE
909D 010004           LD    BC,1024      ;RESET BYTE COUNTER
90A0 2AFD90           LD    HL,(TEMP)    ;MVE ADR PTR TO MEM START
90A3 C37E90           JP    AGN          ;DO IT AGAIN !!!!!
90A6 3E04       CHECK LD    A,PGMOFF     ;TURN 25V SUPPLY OFF
90A8 D3F3             OUT   (CNTRL),A
90AA CDB291           CALL  PORTIN       ;SET 8255 TO READ
90AD 216492           LD    HL,MSG6      ;DISPLAY "VERIFYING" MSG
90B0 CDA728           CALL  28A7H
90B3 2AFD90           LD    HL,(TEMP)    ;RESET MEMORY POINTER
90B6 110000           LD    DE,0         ;RESET PROM ADR POINTER
90B9 010004           LD    BC,1024      ;SET  BYTE COUNTER
90BC CD7891     AGAIN CALL  PROMPT       ;OUT ADR TO PROM
90BF DBF1             IN    A,(B8255)    ;READ BYTE FROM PROM
90C1 BE               CP    (HL)         ;EQUAL MEMORY ?
90C2 C2E190           JP    NZ,BAD       ;NO...ABORT
90C5 23               INC   HL
90C6 13               INC   DE           ;BUMP POINTERS
90C7 0B               DEC   BC           ;DEC AND TEST BYTE CNTR
90C8 3E00             LD    A,0
90CA B9               CP    C
90CB C2BC90           JP    NZ,AGAIN     ;KEEP GOING.....
90CE B8               CP    B
```

Program continued

```
91FA 2E         01220          DEFB     '.'
91FB 30         01230          DEFB     '0'
91FC 0D         01240          DEFB     13
91FD 2A         01250          DEFB     '*'
91FE 54         01260          DEFB     'T'
91FF 59         01270          DEFB     'Y'
9200 50         01280          DEFB     'P'
9201 45         01290          DEFB     'E'
9202 3A         01300          DEFB     ':'
9203 20         01310          DEFB     ' '
9204 32         01320          DEFB     '2'
9205 37         01330          DEFB     '7'
9206 30         01340          DEFB     '0'
9207 38         01350          DEFB     '8'
9208 0D         01360          DEFB     13
9209 0D         01370          DEFB     13
920A 2A         01380          DEFB     '*'
920B 49         01390          DEFB     'I'
920C 4E         01400          DEFB     'N'
920D 53         01410          DEFB     'S'
920E 45         01420          DEFB     'E'
920F 52         01430          DEFB     'R'
9210 54         01440          DEFB     'T'
9211 20         01450          DEFB     ' '
9212 50         01460          DEFB     'P'
9213 52         01470          DEFB     'R'
9214 4F         01480          DEFB     'O'
9215 4D         01490          DEFB     'M'
9216 0D         01500          DEFB     13
9217 2A         01510          DEFB     '*'
9218 45         01520          DEFB     'E'
9219 4E         01530          DEFB     'N'
921A 54         01540          DEFB     'T'
921B 45         01550          DEFB     'E'
921C 52         01560          DEFB     'R'
921D 20         01570          DEFB     ' '
921E 53         01580          DEFB     'S'
921F 54         01590          DEFB     'T'
9220 41         01600          DEFB     'A'
9221 52         01610          DEFB     'R'
9222 54         01620          DEFB     'T'
9223 20         01630          DEFB     ' '
9224 41         01640          DEFB     'A'
9225 44         01650          DEFB     'D'
9226 52         01660          DEFB     'R'
9227 3A         01670          DEFB     ':'
9228 00         01680          DEFB     0
9229 0D         01690 MSG2     DEFB     13
922A 2A         01700          DEFB     '*'
922B 56         01710          DEFB     'V'
922C 45         01720          DEFB     'E'
922D 52         01730          DEFB     'R'
922E 49         01740          DEFB     'I'
922F 46         01750          DEFB     'F'
9230 59         01760          DEFB     'Y'
9231 49         01770          DEFB     'I'
9232 4E         01780          DEFB     'N'
9233 47         01790          DEFB     'G'
9234 20         01800          DEFB     ' '
9235 42         01810          DEFB     'B'
9236 4C         01820          DEFB     'L'
9237 41         01830          DEFB     'A'
9238 4E         01840          DEFB     'N'
9239 4B         01850          DEFB     'K'
923A 3A         01860          DEFB     ':'
923B 20         01870          DEFB     ' '
923C 00         01880          DEFB     0
923D 50         01890 MSG3     DEFB     'P'
923E 41         01900          DEFB     'A'
923F 53         01910          DEFB     'S'
```

```
9240 53         DEFB    'S'
9241 0D         DEFB    13
9242 00         DEFB    0
9243 46   MSG4  DEFB    'F'
9244 41         DEFB    'A'
9245 49         DEFB    'I'
9246 4C         DEFB    'L'
9247 0D         DEFB    13
9248 00         DEFB    0
9249 0D   MSG5  DEFB    13
924A 2A         DEFB    '*'
924B 2D         DEFB    '-'
924C 20         DEFB    ' '
924D 2D         DEFB    '-'
924E 20         DEFB    ' '
924F 2D         DEFB    '-'
9250 20         DEFB    ' '
9251 50         DEFB    'P'
9252 52         DEFB    'R'
9253 4F         DEFB    'O'
9254 47         DEFB    'G'
9255 52         DEFB    'R'
9256 41         DEFB    'A'
9257 4D         DEFB    'M'
9258 4D         DEFB    'M'
9259 49         DEFB    'I'
925A 4E         DEFB    'N'
925B 47         DEFB    'G'
925C 20         DEFB    ' '
925D 2D         DEFB    '-'
925E 20         DEFB    ' '
925F 2D         DEFB    '-'
9260 20         DEFB    ' '
9261 2D         DEFB    '-'
9262 0D         DEFB    13
9263 00         DEFB    0
9264 2A   MSG6  DEFB    '*'
9265 56         DEFB    'V'
9266 45         DEFB    'E'
9267 52         DEFB    'R'
9268 49         DEFB    'I'
9269 46         DEFB    'F'
926A 59         DEFB    'Y'
926B 49         DEFB    'I'
926C 4E         DEFB    'N'
926D 47         DEFB    'G'
926E 20         DEFB    ' '
926F 42         DEFB    'B'
9270 55         DEFB    'U'
9271 52         DEFB    'R'
9272 4E         DEFB    'N'
9273 3A         DEFB    ':'
9274 20         DEFB    ' '
9275 00         DEFB    0
9276 2A   MSG7  DEFB    '*'
9277 4F         DEFB    'O'
9278 50         DEFB    'P'
9279 45         DEFB    'E'
927A 52         DEFB    'R'
927B 41         DEFB    'A'
927C 54         DEFB    'T'
927D 49         DEFB    'I'
927E 4F         DEFB    'O'
927F 4E         DEFB    'N'
9280 20         DEFB    ' '
9281 43         DEFB    'C'
9282 4F         DEFB    'O'
9283 4D         DEFB    'M'
9284 50         DEFB    'P'
9285 4C         DEFB    'L'
```

Program continued

```
90CF C2BC90     01400          JP      NZ,AGAIN        ;NOT YET....
90D2 213D92     01410          LD      HL,MSG3         ;PASSED..PROM PROGRAMMED
90D5 CDA728     01420          CALL    28A7H           ;DISPLAY "PASSED"
90D8 217692     01430          LD      HL,MSG7         ;DISPLAY "COMPLETE"
90DB CDA728     01440          CALL    28A7H
90DE C3E790     01450          JP      EXIT
90E1 214392     01460 BAD      LD      HL,MSG4         ;DISPLAY "FAIL" MSG
90E4 CDA728     01470          CALL    28A7H
90E7 218B92     01480 EXIT     LD      HL,MSG8         ;DISPLAY RESTART PROMPT
90EA CDA728     01490          CALL    28A7H
90ED CD4900     01500 KEYLK    CALL    049H            ;GET CHAR FROM KEYBD
90F0 FE41       01510          CP      41H             ;= "A" ?
90F2 CA0090     01520          JP      Z,START         ;YES..RUN AGAIN
90F5 FE53       01530          CP      53H             ;= "S" ?
90F7 CA2D40     01540          JP      Z,402DH         ;REBOOT  TRSDOS
90FA C3ED90     01550          JP      KEYLK
90FD 00         01560 TEMP     DEFB    0               ;TEMP  HL ADR STORAGE
90FE 00         01570          DEFB    0
90FF 00         01580 DELAY    DEFB    0
0000            01590          END
00000 TOTAL ERRORS
```

Program Listing 2. *PROM/CMD (subroutines/messages)*

```
                00100 ;TITLE           PROM/CMD (SUBROUTINES/MESSAGES)
                00110 ;
00F3            00120 CNTRL    EQU     0F3H            ;8255 CONTROL PORT
00F0            00130 A8255    EQU     0F0H            ;PORT A - LOW ADDRESS
0082            00140 IN82     EQU     82H             ;INIT 8255 FOR READ
0003            00150 WROFF    EQU     3               ;READ SELECTED (CS=0)
0000            00160 PLSOFF   EQU     0               ;PRGM PULSE OFF
0001            00170 PLSON    EQU     1               ;PRGM PULSE ON
90FF            00180 DELAY    EQU     90FFH           ;DELAY COUNTER
9001            00190 START    EQU     9001H
0004            00200 PGMOFF   EQU     4               ;PRGM VOLTAGE OFF
9150            00210          ORG     9150H
9150 D7         00220 SHFT1    RST     10H             ;GET KEYBOARD DATA
9151 D45F91     00230          CALL    NC,ALPHA        ;IF CY=0 CHAR = ALPHA
9154 D630       00240          SUB     30H             ;NUMERIC, CNVRT FRM ASCII
9156 CB27       00250          SLA     A               ;SHIFT INTO HIGH NIBBLE
9158 CB27       00260          SLA     A
915A CB27       00270          SLA     A
915C CB27       00280          SLA     A
915E C9         00290          RET
915F FE41       00300 ALPHA    CP      41H             ;CHECK A-F LIMITS
9161 DA6C91     00310          JP      C,ERR
9164 FE47       00320          CP      47H
9166 D26C91     00330          JP      NC,ERR          ;NG... ABORT
9169 D607       00340          SUB     7               ;CONVERT TO NUMERIC FORM
916B C9         00350          RET                     ;(SUB 41H & ADD 3AH=SUB 7)
916C E1         00360 ERR      POP     HL              ;DISCARD BAD RETURN ADRS
916D E1         00370          POP     HL
916E C30190     00380          JP      START           ;START OVER
9171 D7         00390 LOWNIB   RST     10H
9172 D45F91     00400          CALL    NC,ALPHA
9175 D630       00410          SUB     30H
9177 C9         00420          RET
9178 7A         00430 PROMPT   LD      A,D             ;CONVERT ADR FOR 8255
9179 FE00       00440          CP      0               ;CHK IF UPPER TWO BITS =0
917B 2828       00450          JR      Z,SHRTAD        ;YES..RESET A8 & A9
917D FE03       00460          CP      3               ;EVALUATE UPPER TWO BITS
917F 2810       00470          JR      Z,BOTH          ;BOTH SET ?
9181 FE01       00480          CP      1               ;NO..JUST A8 ?
9183 2816       00490          JR      Z,AD8           ;YES, TAKE CARE OF IT...
9185 3E0B       00500          LD      A,0BH           ;JUST LEAVES A9!!
9187 D3F3       00510          OUT     (CNTRL),A
9189 3E08       00520          LD      A,8
```

```
918B D3F3     00530        OUT    (CNTRL),A
918D 7B       00540 EXI    LD     A,E          ;OUTPUT LOWER 8 BITS
918E D3F0     00550        OUT    (A8255),A
9190 C9       00560        RET
9191 3E09     00570 BOTH   LD     A,9          ;SET PC4 & PC5
9193 D3F3     00580        OUT    (CNTRL),A
9195 3E0B     00590        LD     A,0BH
9197 D3F3     00600        OUT    (CNTRL),A
9199 18F2     00610        JR     EXI
919B 3E09     00620 AD8    LD     A,9          ;SET PC4
919D D3F3     00630        OUT    (CNTRL),A
919F 3E0A     00640        LD     A,0AH        ;ZERO PC5
91A1 D3F3     00650        OUT    (CNTRL),A
91A3 18E8     00660        JR     EXI
91A5 3E08     00670 SHRTAD LD     A,8          ;ZERO PC4 & PC5
91A7 D3F3     00680        OUT    (CNTRL),A
91A9 3E0A     00690        LD     A,0AH
91AB D3F3     00700        OUT    (CNTRL),A
91AD 18DE     00710        JR     EXI
91AF 00       00720        NOP
91B0 00       00725        NOP
91B1 00       00730        NOP
91B2 3E82     00740 PORTIN LD     A,IN82       ;SETUP 8255 FOR READ
91B4 D3F3     00750        OUT    (CNTRL),A
91B6 3E03     00760        LD     A,WROFF      ;ENABLE CS
91B8 D3F3     00770        OUT    (CNTRL),A
91BA 3E00     00780        LD     A,PLSOFF     ;TURN OFF PRGM PULSE
91BC D3F3     00790        OUT    (CNTRL),A
91BE 3E04     00800        LD     A,PGMOFF     ;TURN OFF +25 V SUPPLY
91C0 D3F3     00810        OUT    (CNTRL),A
91C2 C9       00820        RET
91C3 3E01     00830 PULSE  LD     A,PLSON      ;TURN ON PRGM PULSE
91C5 D3F3     00840        OUT    (CNTRL),A
91C7 3E18     00850        LD     A,24         ;DELAY ABOUT .5 MS
91C9 32FF90   00860        LD     (DELAY),A
91CC DD3500   00870 LP1    DEC    (IX+0)
91CF 00       00880        NOP
91D0 00       00890        NOP
91D1 20F9     00900        JR     NZ,LP1
91D3 3E00     00910        LD     A,PLSOFF     ;TURN OFF PRGM PULSE
91D5 D3F3     00920        OUT    (CNTRL),A
91D7 3E18     00930        LD     A,24
91D9 32FF90   00940        LD     (DELAY),A
91DC DD3500   00950 LP2    DEC    (IX+0)       ;DELAY AGAIN
91DF 00       00960        NOP
91E0 00       00970        NOP
91E1 20F9     00980        JR     NZ,LP2
91E3 C9       00990        RET
91E4 50       01000 MSG1   DEFB   'P'
91E5 52       01010        DEFB   'R'
91E6 2D       01020        DEFB   '-'
91E7 38       01030        DEFB   '8'
91E8 30       01040        DEFB   '0'
91E9 20       01050        DEFB   ' '
91EA 50       01060        DEFB   'P'
91EB 52       01070        DEFB   'R'
91EC 4F       01080        DEFB   'O'
91ED 47       01090        DEFB   'G'
91EE 52       01100        DEFB   'R'
91EF 41       01110        DEFB   'A'
91F0 4D       01120        DEFB   'M'
91F1 4D       01130        DEFB   'M'
91F2 45       01140        DEFB   'E'
91F3 52       01150        DEFB   'R'
91F4 20       01160        DEFB   ' '
91F5 56       01170        DEFB   'V'
91F6 45       01180        DEFB   'E'
91F7 52       01190        DEFB   'R'
91F8 20       01200        DEFB   ' '
91F9 31       01210        DEFB   '1'
```

Program continued

```
9286 45         02620           DEFB    'E'
9287 54         02630           DEFB    'T'
9288 45         02640           DEFB    'E'
9289 0D         02650           DEFB    13
928A 00         02660           DEFB    0
928B 2A         02670 MSG8      DEFB    '*'
928C 20         02680           DEFB    ' '
928D 41         02690           DEFB    'A'
928E 20         02700           DEFB    ' '
928F 20         02710           DEFB    ' '
9290 54         02720           DEFB    'T'
9291 4F         02730           DEFB    'O'
9292 20         02740           DEFB    ' '
9293 52         02750           DEFB    'R'
9294 55         02760           DEFB    'U'
9295 4E         02770           DEFB    'N'
9296 20         02780           DEFB    ' '
9297 41         02790           DEFB    'A'
9298 47         02800           DEFB    'G'
9299 41         02810           DEFB    'A'
929A 49         02820           DEFB    'I'
929B 4E         02830           DEFB    'N'
929C 20         02840           DEFB    ' '
929D 2F         02850           DEFB    '/'
929E 20         02860           DEFB    ' '
929F 20         02870           DEFB    ' '
92A0 53         02880           DEFB    'S'
92A1 20         02890           DEFB    ' '
92A2 20         02900           DEFB    ' '
92A3 46         02910           DEFB    'F'
92A4 4F         02920           DEFB    'O'
92A5 52         02930           DEFB    'R'
92A6 20         02940           DEFB    ' '
92A7 44         02950           DEFB    'D'
92A8 4F         02960           DEFB    'O'
92A9 53         02970           DEFB    'S'
92AA 00         02980           DEFB    0
0000            02990           END
00000 TOTAL ERRORS
```

Program Listing 3. *PROM/CMD loader*

```
 10 CLS :
    PRINT "* * * * *  PROM/CMD  LOADER  * * * * *"
 20 DEFINT A,B:
    C = 0
 30 FOR A = - 28687 TO - 27990
 40  READ B:
     POKE A,B:
     C = C + B:
     NEXT
 45 IF C < > 74568
     THEN
      PRINT "CHECKSUM ERROR IN DATA STATEMENT(S)":
      END
 50 PRINT "OPERATION COMPLETE"
 90 DATA 243, 33, 15,224, 17, 0,144, 1,171, 2
100 DATA 237,176,195, 0,144,205,201, 1, 33,228
110 DATA 145,205,167, 40,205,179, 27,205, 80,145
120 DATA 87,205,113,145,130, 87,205, 80,145, 95
130 DATA 205,113,145,131, 95,213, 33, 41,146,205
140 DATA 167, 40,205,178,145, 17, 0, 0, 1, 0
150 DATA 4,205,120,145, 0,219,241,254,255,194
160 DATA 225,144, 19, 11,121,254, 0,194, 46,144
170 DATA 120,254, 0,194, 46,144, 33, 61,146,205
180 DATA 167, 40, 62,128,211,243, 62, 2,211,243
190 DATA 62, 0,211,243, 62, 5,211,243, 62,255
```

```
200 DATA 6,255, 0, 5,194, 97,144, 61,194, 95
210 DATA 144, 33, 73,146,205,167, 40,225, 34,253
220 DATA 144, 14,200,197, 1, 0, 4,221, 33,255
230 DATA 144, 17, 0, 0,205,120,145,126,211,241
240 DATA 205,195,145, 35, 19, 11, 62, 0,185,194
250 DATA 129,144,184,194,129,144,193, 13,202,166
260 DATA 144,197, 1, 0, 4, 42,253,144,195,126
270 DATA 144, 62, 4,211,243,205,178,145, 33,100
280 DATA 146,205,167, 40, 42,253,144, 17, 0, 0
290 DATA 1, 0, 4,205,120,145,219,241,190,194
300 DATA 225,144, 35, 19, 11, 62, 0,185,194,188
310 DATA 144,184,194,188,144, 33, 61,146,205,167
320 DATA 40, 33,118,146,205,167, 40,195,231,144
330 DATA 33, 67,146,205,167, 40, 33,139,146,205
340 DATA 167, 40,205, 73, 0,254, 65,202, 0,144
350 DATA 254, 83,202, 45, 64,195,237,144,241,143
360 DATA 0,128,255,255,255,255, 0, 0, 64, 0
370 DATA 255,127,255,255,128, 0, 0, 64,255,255
380 DATA 255,239, 0, 64, 0, 0,255,255,111,255
390 DATA 0, 64, 48,128,255,255,255,255, 0, 0
400 DATA 0, 0,255,255,255,255, 0, 0, 0, 0
410 DATA 127,255,255,127, 32, 0, 0, 32,255,255
420 DATA 255,127, 0,160, 16,128,215,212, 95,145
430 DATA 214, 48,203, 39,203, 39,203, 39,203, 39
440 DATA 201,215,212, 95,145,214, 48,203, 39,203
450 DATA 39,203, 39,203, 39,201,254, 65,218,108
460 DATA 145,254, 71,210,108,145,214, 7,201,225
470 DATA 225,195, 1,144,215,212, 95,145,214, 48
480 DATA 201,122,254, 0, 40, 40,254, 3, 40, 16
490 DATA 150, 1, 40, 22, 62, 11,211,243, 62, 8
500 DATA 211,243,123,211,240,201, 62, 9,211,243
510 DATA 62, 11,211,243, 24,242, 62, 9,211,243
520 DATA 62, 10,211,243, 24,232, 62, 8,211,243
530 DATA 62, 10,211,243, 24,222, 0, 0, 0, 62
540 DATA 130,211,243, 62, 3,211,243, 62, 0,211
550 DATA 243, 62, 4,211,243,201, 62, 1,211,243
560 DATA 62, 24, 50,255,144,221, 53, 0, 0, 0
570 DATA 32,249, 62, 0,211,243, 62, 24, 50,255
580 DATA 144,221, 53, 0, 0, 0, 32,249,201, 80
590 DATA 82, 45, 56, 48, 32, 80, 82, 79, 71, 82
600 DATA 65, 77, 77, 69, 82, 32, 86, 69, 82, 32
610 DATA 49, 46, 48, 13, 42, 84, 89, 80, 69, 58
620 DATA 32, 50, 55, 48, 56, 13, 13, 42, 73, 78
630 DATA 83, 69, 82, 84, 32, 80, 82, 79, 77, 13
640 DATA 42, 69, 78, 84, 69, 82, 32, 83, 84, 65
650 DATA 82, 84, 32, 65, 68, 82, 58, 0, 13, 42
660 DATA 86, 69, 82, 73, 70, 89, 73, 78, 71, 32
670 DATA 66, 76, 65, 78, 75, 58, 32, 0, 80, 65
680 DATA 83, 83, 13, 0, 70, 65, 73, 76, 13, 0
690 DATA 13, 42, 45, 32, 45, 32, 45, 32, 80, 82
700 DATA 79, 71, 82, 65, 77, 77, 73, 78, 71, 32
710 DATA 45, 32, 45, 32, 45, 13, 0, 42, 86, 69
720 DATA 82, 73, 70, 89, 73, 78, 71, 32, 66, 85
730 DATA 82, 78, 58, 32, 0, 42, 79, 80, 69, 82
740 DATA 65, 84, 73, 79, 78, 32, 67, 79, 77, 80
750 DATA 76, 69, 84, 69, 13, 0, 42, 32, 65, 32
760 DATA 32, 84, 79, 32, 82, 85, 78, 32, 65, 71
770 DATA 65, 73, 78, 32, 47, 32, 32, 83, 32, 32
780 DATA 70, 79, 82, 32, 68, 79, 83, 0
```

HOME APPLICATIONS

Magazine Index

Money Minder

Groupies:
A Strategy to Group Like Objects

Magazine Index

by John Cominio

I used to read a magazine article and then forget where I had read it. When I noticed that my magazines were becoming a mess (due to my thumbing through their indexes), I wrote a program for my TRS-80 which would allow me to find different articles quickly. The program (see Program Listing) is set to run on a 16K to 48K machine with disk. If you are running without disks, you can easily modify this program to support cassette files. See lines 2510 through 2630.

When you run the program, a menu appears showing all the options open to the user. These options are listed in Table 1. ENTER MAGAZINE DATA allows you to store data pertaining to your magazines. Enter the magazine's name, title of the article, month and year of publication (year is optional), page on which the article begins, and keyword(s) applying to that article. The keywords index the article by subject. If an article is entitled "RAM" you might enter randomaccessmemory as a keyword. Notice that spaces are not needed between the words. When you search by subject, the words memory, random, and access will all produce a match. All entries are limited to 20 characters, except the keyword, which is 30. The variables the program uses are shown in Table 2.

```
(1) → ENTER MAGAZINE DATA
(2) → REVIEW STORED DATA
(3) → SEARCH THROUGH DATA
(4) → SAVE DATA ONTO DISK
(5) → LOAD DATA FROM DISK
(6) → KILL FILE ON DISK
(7) → EDIT STORED DATA
(8) → CLEAR STORED DATA

COMMAND →? _
```

Table 1. *Menu of options*

REVIEW STORED DATA displays all entries that are currently in memory. Before saving entries, you can double-check to see if you accidentally entered any wrong data and then edit that file if necessary.

A$(x)	=	Magazine's name
B$(x)	=	Article title
C$(x)	=	Month (year)
D$(x)	=	Page number
E$(x)	=	Keyword
X	=	Current file number

Table 2. *List of variables*

SEARCH THROUGH DATA has four sub-options—search by magazine's name, search by article title, search by month (year), and search by subject. When you perform a search, you can abbreviate your entry. For example, suppose one of the magazines was *80 Microcomputing*. For a search by title, you could enter 80, 80 Micro, Microcomputing, or 80 Microcomputing, and all would match. This also applies to the other types of searches. If a match is found, it is displayed, and the search continues until all entries have been checked. If you wish to return to the menu before the search is completed, enter a #.

SAVE DATA ONTO DISK dumps all entries in memory to disk under the filespec you enter.

KILL FILE ON DISK erases a file which you have saved. If you use cassette, you have to delete this option (lines 1720 through 1780).

EDIT STORED DATA corrects any error made while you entered the data. When you enter the number of the file to be corrected, the screen displays the file's information and prompts you to enter the correct data.

CLEAR STORED DATA erases all files in memory. Be sure to save your data before you type this command. When you enter this command, you will be asked if you are sure you want this option. If you enter N, you return to the menu.

Lines 50 through 70 automatically determine memory size and dimension and clear enough space to hold the maximum number of files allowed in your system.

Program Listing. *Magazine Index*

```
 10 :
    ' ****** MAGAZINE INDEX   VERSION 1.2
 20 :
    ' ****** BY JOHN COMINIO
 30 :
    ' ****** MAY 29, 1981    MODEL I OR III
 40 DEFINT A - Z
 50 CLEAR INT( MEM * .75)
 60 X = INT( MEM / 17.6)
 70 DIM A$(X), B$(X), C$(X), D(X), E$(X)
 80 X = 0
 90 ON ERROR GOTO 2440
100 CLS
110 PRINT TAB(20)"---- MAGAZINE INDEX ----"
120 PRINT
130 PRINT TAB(5)"(1) --> ENTER MAGAZINE DATA"
140 PRINT TAB(5)"(2) --> REVIEW STORED DATA"
150 PRINT TAB(5)"(3) --> SEARCH THROUGH DATA"
160 PRINT TAB(5)"(4) --> SAVE DATA ONTO DISK"
170 PRINT TAB(5)"(5) --> LOAD DATA FROM DISK"
180 PRINT TAB(5)"(6) --> KILL FILE ON DISK"
190 PRINT TAB(5)"(7) --> EDIT STORED DATA"
200 PRINT TAB(5)"(8) --> CLEAR STORED DATA"
210 PRINT
220 A$ = " "
230 PRINT @ 704,"";
240 PRINT CHR$(30);
250 INPUT "COMMAND -->";A$
260 IF VAL(A$) < 1 OR VAL(A$) > 8
     THEN
      220
270 ON VAL(A$) GOTO 290, 620, 850, 1470, 1600, 1720, 1790, 2370
280 END
290 CLS
300 X = X + 1
310 PRINT TAB(15)"====== ENTER MAGAZINE DATA  FILE #";X;"======"
320 PRINT TAB(19)"-- 'END' TO RETURN TO MENU --"
330 PRINT
340 A$(X) = ""
350 PRINT @ 192,"";
360 PRINT CHR$(30);
370 INPUT "ENTER MAGAZINE'S NAME -->";A$(X)
380 IF LEN(A$(X)) > 20 OR A$(X) = ""
     THEN
      340
390 IF A$(X) = "END"
     THEN
      A$(X) = "":
      X = X - 1:
      GOTO 100
400 B$(X) = ""
410 PRINT @ 256,"";
420 PRINT CHR$(30);
430 INPUT "ENTER ARTICLE TITLE   -->";B$(X)
440 IF LEN(B$(X)) > 20 OR B$(X) = ""
     THEN
      400
450 C$(X) = ""
460 PRINT @ 320,"";
470 PRINT CHR$(30);
480 INPUT "ENTER MONTH (YEAR)    -->";C$(X)
490 IF LEN(C$(X)) > 20 OR C$(X) = ""
     THEN
      450
500 D(X) = 0
510 PRINT @ 384,"";
520 PRINT CHR$(30);
530 INPUT "ENTER PAGE NUMBER     -->";R$
```

Program continued

```
540 D(X) = VAL(R$)
550 IF D(X) > 32767 OR D(X) = 0
     THEN
      500
560 E$(X) = ""
570 PRINT @ 448,"";
580 PRINT CHR$(30);
590 INPUT "ENTER KEYWORD(S)       -->";E$(X)
600 IF LEN(E$(X)) > 30 OR E$(X) = ""
     THEN
      560
610 GOTO 290
620 IF X = 0
     THEN
      100
630 A$ = ""
640 FOR A = 1 TO X
650  R = A
660  IF R > 9
      THEN
       A$ = " "
670  IF R > 99
      THEN
       A$ = "  "
680  CLS
690  PRINT TAB(15)"====== REVIEW STORED FILES  FILE #";R;"======"
700  PRINT TAB(15)"------ '#' TO EXIT  'ENTER' TO CONT.";A$;" ------
     "
710  PRINT
720  PRINT "MAGAZINE'S NAME --> ";A$(R)
730  PRINT "ARTICLE TITLE   --> ";B$(R)
740  PRINT "MONTH (YEAR)    --> ";C$(R)
750  PRINT "PAGE NUMBER     -->";D(R)
760  PRINT "KEYWORD(S)      --> ";E$(R)
770  I$ = INKEY$
780  IF I$ = ""
      THEN
       770
790  IF I$ = "#"
      THEN
       100
800  IF I$ = CHR$(13) AND (X < > 1)
      THEN
       810 :
      ELSE
       770
810  NEXT A
820 A = 0
830 R = 0
840 GOTO 100
850 IF X = 0
     THEN
      100
860 CLS
870 PRINT TAB(15)"====== SEARCH THROUGH FILES ======"
880 PRINT
890 PRINT "(1) --> SEARCH BY MAG. NAME"
900 PRINT "(2) --> SEARCH BY ARTICLE TITLE"
910 PRINT "(3) --> SEARCH BY MONTH (YEAR)"
920 PRINT "(4) --> SEARCH BY SUBJECT"
930 PRINT @ 448,"";
940 PRINT CHR$(30);
950 INPUT "WHICH NUMBER";Z$
960 IF VAL(Z$) < 1 OR VAL(Z$) > 4
     THEN
      930
970 ON VAL(Z$) GOTO 980, 1070, 1160, 1250
980 PRINT @ 512,"";
990 PRINT CHR$(30);
1000 INPUT "ENTER MAGAZINE'S NAME -->";A$
```

```
1010 IF LEN(A$) > 20
      THEN
       980
1020 FOR A = 1 TO X
1030  FOR R = 1 TO 20
1040   IF A$ = MID$(A$(A), R, LEN(A$))
        THEN
         GOSUB 1340 :
        ELSE
         NEXT R
1050  NEXT A
1060 GOTO 100
1070 PRINT @ 512,"";
1080 PRINT CHR$(30);
1090 INPUT "ENTER ARTICLE TITLE -->";A$
1100 IF LEN(A$) > 20
      THEN
       1070
1110 FOR A = 1 TO X
1120  FOR R = 1 TO 20
1130   IF A$ = MID$(B$(A), R, LEN(A$))
        THEN
         GOSUB 1340 :
        ELSE
         NEXT R
1140  NEXT A
1150 GOTO 100
1160 PRINT @ 512,"";
1170 PRINT CHR$(30);
1180 INPUT "ENTER MONTH (YEAR) -->";A$
1190 IF LEN(A$) > 20
      THEN
       1160
1200 FOR A = 1 TO X
1210  FOR R = 1 TO 20
1220   IF A$ = MID$(C$(A), R, LEN(A$))
        THEN
         GOSUB 1340 :
        ELSE
         NEXT R
1230  NEXT A
1240 GOTO 100
1250 PRINT @ 512,"";
1260 PRINT CHR$(30);
1270 INPUT "ENTER SUBJECT -->";A$
1280 IF LEN(A$) > 30
      THEN
       1250
1290 FOR A = 1 TO X
1300  FOR R = 1 TO 30
1310   IF A$ = MID$(E$(A), R, LEN(A$))
        THEN
         GOSUB 1340 :
        ELSE
         NEXT R
1320  NEXT A
1330 GOTO 100
1340 CLS
1350 PRINT TAB(15)"====== FILE NUMBER";A;"======"
1360 PRINT
1370 PRINT "MAGAZINE'S NAME --> ";A$(A)
1380 PRINT "ARTICLE TITLE   --> ";B$(A)
1390 PRINT "MONTH (YEAR)    --> ";C$(A)
1400 PRINT "PAGE NUMBER     -->";D(A)
1410 PRINT
1420 PRINT "HIT ENTER TO CONTINUE SEARCH   '#' TO EXIT"
1430 I$ = INKEY$
1440 IF I$ = ""
      THEN
       1430
```

Program continued

```
1450 IF I$ = CHR$(13)
      THEN
       RETURN
1460 IF I$ = "#"
      THEN
       100 :
      ELSE
       1430
1470 IF X = 0
      THEN
       100
1480 A$ = ""
1490 PRINT @ 704,"";
1500 PRINT CHR$(30);
1510 INPUT "ENTER SAVE FILESPEC --> ";A$
1520 IF LEN(A$) > 21 OR A$ = ""
      THEN
       1470
1530 OPEN "O",1, A$
1540 PRINT #1, X
1550 FOR R = 1 TO X + 3 STEP 4
1560  PRINT #1, A$(R);",";A$(R + 1);",";A$(R + 2);",";A$(R + 3);",";B
      $(R);",";B$(R + 1);",";B$(R + 2);",";B$(R + 3);",";C$(R);",";C$
      (R + 1);",";C$(R + 2);",";C$(R + 3);",";D(R);",";D(R + 1);",";D
      (R + 2);",";D(R + 3);",";E$(R);",";E$(R + 1);",";E$(R + 2);",";
      E$(R + 3)
1570  NEXT R
1580 CLOSE
1590 GOTO 100
1600 A$ = ""
1610 PRINT @ 704,"";
1620 PRINT CHR$(30);
1630 INPUT "ENTER LOAD FILESPEC -->";A$
1640 IF LEN(A$) > 21 OR A$ = ""
      THEN
       1600
1650 OPEN "I",1, A$
1660 INPUT #1, X
1670 FOR R = 1 TO X + 3 STEP 4
1680  INPUT #1, A$(R), A$(R + 1), A$(R + 2), A$(R + 3), B$(R), B$(R
      + 1), B$(R + 2), B$(R + 3), C$(R), C$(R + 1), C$(R + 2), C$(R
      + 3), D(R), D(R + 1), D(R + 2), D(R + 3), E$(R), E$(R + 1), E$(
      R + 2), E$(R + 3)
1690  NEXT R
1700 CLOSE
1710 GOTO 100
1720 A$ = ""
1730 PRINT @ 704,"";
1740 PRINT CHR$(30);
1750 INPUT "ENTER KILL FILESPEC -->";A$
1760 IF LEN(A$) > 21 OR A$ = ""
      THEN
       1720
1770 KILL A$
1780 GOTO 100
1790 IF X = 0
      THEN
       100
1800 CLS
1810 PRINT TAB(15)"====== EDIT A FILE ======"
1820 PRINT
1830 A$ = ""
1840 PRINT @ 128,"";
1850 PRINT CHR$(30);
1860 INPUT "ENTER FILE NUMBER (-1 TO EXIT) -->";A$
1870 R = VAL(A$)
1880 IF R = - 1
      THEN
       100
1890 IF R = 0 OR R > X
```

```
      THEN
       1830
1900 GOSUB 1920
1910 GOTO 1980
1920 PRINT @ 128, CHR$(31);"(1) --> ";A$(R)
1930 PRINT @ 192,"(2) --> ";B$(R)
1940 PRINT @ 256,"(3) --> ";C$(R)
1950 PRINT @ 320,"(4) -->";D(R)
1960 PRINT @ 384,"(5) --> ";E$(R)
1970 RETURN
1980 A$ = ""
1990 PRINT @ 832, CHR$(30);"WHICH NUMBER -->";
2000 INPUT A$
2010 R1 = VAL(A$)
2020 IF R1 < 1 OR R1 > 5
      THEN
       1980
2030 ON R1 GOSUB 2060, 2110, 2160, 2210, 2260
2040 GOSUB 1920
2050 GOTO 2310
2060 A$(R) = ""
2070 PRINT @ 832, CHR$(30);
2080 INPUT "NEW MAGAZINE NAME -->";A$(R)
2090 IF LEN(A$(R)) > 20 OR A$(R) = ""
      THEN
       2060
2100 RETURN
2110 B$(R) = ""
2120 PRINT @ 832, CHR$(30);
2130 INPUT "NEW ARTICLE TITLE -->";B$(R)
2140 IF LEN(B$(R)) > 20 OR B$(R) = ""
      THEN
       2110
2150 RETURN
2160 C$(R) = ""
2170 PRINT @ 832, CHR$(30);
2180 INPUT "NEW MONTH (YEAR) -->";C$(R)
2190 IF LEN(C$(R)) > 20 OR C$(R) = ""
      THEN
       2160
2200 RETURN
2210 D(R) = 0
2220 PRINT @ 832, CHR$(30);
2230 INPUT "NEW PAGE NUMBER -->";D(R)
2240 IF D(R) > 32767 OR D(R) = 0
      THEN
       2210
2250 RETURN
2260 E$(R) = ""
2270 PRINT @ 832, CHR$(30);
2280 INPUT "NEW KEYWORD(S) -->";E$(R)
2290 IF LEN(E$(R)) > 30 OR E$(R) = ""
      THEN
       2260
2300 RETURN
2310 PRINT @ 832, CHR$(30);
2320 INPUT "MORE CORRECTION (Y/N) -->";A$
2330 A$ = LEFT$(A$, 1)
2340 IF A$ = "Y"
      THEN
       1980
2350 IF A$ = "N"
      THEN
       100 :
      ELSE
       2310
2360 GOTO 100
2370 PRINT @ 768,"";
2380 PRINT CHR$(30);
2390 INPUT "ARE YOU SURE (Y/N)";A$
```

Program continued

```
2400 A$ = LEFT$(A$, 1)
2410 IF A$ = "Y"
      THEN
       2430
2420 IF A$ = "N"
      THEN
       100 :
      ELSE
       2370
2430 RUN
2440 IF ERR / 2 + 1 = 54
      THEN
       PRINT "FILE NOT FOUND"
2450 IF ERR / 2 + 1 = 65
      THEN
       PRINT "BAD FILE NAME"
2460 IF ERR / 2 + 1 = 62
      THEN
       PRINT "DISK FULL"
2470 PRINT "*** ERROR ***"
2480 FOR R = 1 TO 1000
2490  NEXT R
2500 RESUME 100
2510 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
     * * * * * * * * * * * CHANGES FOR CAS SETTE * * * * * *
      * * * * * * * * * * * * * * * * * * * * * * * * * * *
      * * * * *
2520  LINE 1510:
      INPUT "CASSETTE READY (Y/N) -->";A$:
      A$ = LEFT$(A$,1):
      IF A$ = "N"
       THEN
        100:
       ELSE
        IF A$ = "Y"
         THEN
          1520:
         ELSE
          1510
2540  LINE 1560 CHANGE PRINT #1,. . . TO PRINT # - 1,. . . LINE 1540
      CHANGE PRINT #1, X TO PRINT # - 1, X
2550  DELETE LINE 1580
2560  LINE 1630:
      INPUT "CASSETTE READY (Y/N) -->;A$:A$=LEFT$(A$,1):IFA$="Y"THEN1
      640ELSEIFA$="N"THEN100ELSE1630
2580  LINE 1680 CHANGE INPUT #1,. . . TO INPUT # - 1,. . . LINE 1660
      CHANGE INPUT #1, X TO INPUT # - 1, X
2610  DELETE LINE S 2440 - 2460
2620  DELETE LINE S 1530 AND 1650 AND 1700 AND 1580
2630  LINE 90:
      ON ERROR GOTO 2470
2640  DELETE LINE 1720 - 1780
2650  LINE 1720:
      GOTO 100
```

Money Minder

by Bill Loveys

My wife and I decided to keep better track of where our money was going. We agreed to track only our take-home pay. Money that went directly into our savings account would not enter into the picture except when withdrawn. We wanted to account only for the cash in our pockets and checking accounts.

The Money Minder Program

With only cassette capability, I first thought that any program I developed with data storage would be too slow to be of any practical use. As it turned out, I can run the complete money-tracking process in as little as two hours per month, including printing hard copy reports.

Money Minder is shown in Program Listing 1. The program can track 20 categories of your choice. Every day, you must jot down the money you spend, dividing it among the selected categories. Once every week or two, and at the end of the month, I run the program and enter our income (take-home) and expenses from the notebook. The reports generated from the program give the balance (left-over money) at that point. A check of your wallet and checking account will confirm how honest you have been in listing your expenses. You also see an income listing, a category list showing the day of the month, check number, description, and amount spent in each of the 20 categories. A percentage chart lists by category the amounts you spent that month and the percentage of income spent on that category. At month's end, a short Year-To-Date (YTD) program (Program Listing 2) lists the categories giving YTD amount, YTD average, and YTD percentage that you can compare to the monthly percentage chart to determine where you are overspending.

The program requires a 16K Level II and gives you the option of hard copy. You can modify the program format for use with different printers. Money Minder accepts up to 20 income entries and 200 expense entries per month. It is menu driven and very easy to use. The only change you need to make is to select your own categories by changing the data statements in lines 2030 and 2040 of Program Listing 1 and lines 780–820 of Program Listing 2.

The Menu and Its Features

Explanation of the menu functions and the special features of each are shown below. The program opens with the following video display:

```
                MONEY MINDER
1 ENTER EXPENSES      7 MEM & STRING STORAGE
2 LIST EXPENSES       8 BALANCE CHART
3 ENTER INCOME        9 LIST CATEGORY
4 LIST INCOME         10 PERCENTAGE CHART
5 RECORD DATA         11 CHANGE RECORD
6 VERIFY DATA TAPE    12 PRINT REPORT
              13 READ DATA TAPE
ENTER DESIRED FUNCTION?_
```

Function 1—ENTER EXPENSES

This function displays the 20 categories you select in the data statements in lines 2030 and 2040. If you choose category 1, the video screen displays:

EXPENSES

FOOD

ENTER CHECK NUMBER?_	(Press ENTER for a cash transaction, 7777 if charge card is used, or enter check number if applicable.)
ENTER DESCRIPTION?_	(Short description, under 13 characters, to conserve string storage space and for video and printer format.)
ENTER AMOUNT?_	(Enter dollar amount including decimal point. If there are no significant digits after the decimal point, an integer may be entered. For example, enter $12.00 as 12, $12.10 as 12.1.)
ENTER DAY OF MONTH?_	(Self explanatory. If the same day is used for other entries, only the ENTER key need be pressed.)

RECORD #	CHECK #	DESCRIPTION	AMOUNT	DAY
1	9999	WEGMANS	$ 34.95	1

PRESS ENTER TO CONTINUE?_

Press ENTER for a check number to indicate a cash transaction. Enter 7777 to indicate a charge purchase. The amount you enter is not added during the CATEGORY LISTING but is reflected as CHARGES on the report. In category 1, food, and only in this category, an asterisk entered as the last character in the description indicates EAT OUT on category list. EAT OUT amounts are totaled with other food expenses.

Function 2—LIST EXPENSES

This is a listing of expenses as you enter them from the notebook. The listing displays record number, category, check number, description, amount, and day of the month. The purpose of this listing is to check for mistakes in data entry. If you number your daily notebook entries to correspond to record numbers, a missed entry is easier to find after an entry session. You will also need record numbers when you use Function 11, CHANGE RECORD.

Function 3—ENTER INCOME

The video display for this function is as follows:

* * INCOME ENTRIES * *

ENTER 99 FOR DATE TO END SESSION

ENTER DATE OF INCOME?__ (Format allows nine characters such as 01 SEP 81)
ENTER INCOME SOURCE?__ (Format allows up to nine characters)
ENTER INCOME AMOUNT?__ (Same as expense input)

The session continues until you enter 99 in place of the date, then the menu is displayed. When you first use the program, enter the amount of money you have on hand as the balance forward. When you go from one month to the next, your first entry should be BAL FWD (balance) from the previous month.

Function 4—LIST INCOME

Function 4 has the same format as Function 2 (LIST EXPENSES), showing record numbers and other data you enter. See Figure 1.

DATE	SOURCE	AMOUNT
01 JUN 81	BAL FWD	$ 370.84
01 JUN 81	ARMY	$ 600.00
01 JUN 81	CARRIER	$ 435.78
01 JUN 81	ARMY	$ 154.32
01 JUN 81	CREDIT UN	$ 90.00
15 JUN 81	CARRIER	$ 450.50
16 JUN 81	MORSE	$ 193.12
21 JUN 81	ARMY	$ 82.40
21 JUN 81	CREDIT UN	$ 60.00
23 JUN 81	MORSE	$ 186.74
29 JUN 81	CARRIER	$ 450.44
30 JUN 81	MORSE	$ 192.13

```
BALANCE CHART
------------------------/ TOTAL INCOME /-----
12      ENTRIES$ 3,266.27
------------------------/ TOTAL EXPENSES /--
20      ENTRIES$   232.32
------------------------/ BALANCE /-----------
                 $ 3,033.95
------------------------------------------------
```

Figure 1. *Income list*

Function 5—RECORD DATA

It is important that you use C-30 tapes, because maximum entries will not fit on anything smaller. This is indicative of the maximum time of cassette I/O of 15 minutes for reading or writing data tapes.

FOOD				
2	111	BREAKFAST	$	5.00
5	112	WEGS W/E	$	23.44
5	113	WEGS	$	16.99
8	114	PIZZA	$	4.65
10	115	BURG KING	$	2.61
11	116	W/E	$	11.85
15	117	PIZZA	$	4.40
16	118	BURG KING *	$	4.49
17	119	W/E	$	15.79
17	120	WEGS	$	20.54
21	9999	PIZZA	$	4.65
21	121	WENDY'S	$	4.98
23	122	PIZZA	$	4.40
24	123	LOU & BONS	$	6.14
24	124	BURG KING	$	3.77
25	125	WEGS	$	57.14
29	126	PIZZA	$	4.40
30	127	ERICS *	$	8.12
30	9999	MISC W/E	$	26.57
31	9999	CANDY	$	2.39

EAT OUT $ 12.61
TOTAL $232.32

Figure 2. *Category list*

Function 6—VERIFY DATA TAPES

After you write data to tape, rewind and perform Function 6. This time-consuming procedure is useful if you do not keep backup tapes of previous sessions.

Function 7—MEM & STRING STORAGE

This function allows you to change the DIM statements for income and expense entries. As written, with 2000 string characters cleared for string storage and 20 income and 200 expense entries, the program will work well with 16K. Line 120 checks memory each time the menu is displayed and prints a warning if memory drops under 200. You can limit expense entries to the existing parameters of the program if you combine expenses as a week ending (W/E) entry (for example, combining coffee and lunch expenses, and daily paper purchases).

Function 8—BALANCE CHART

This function displays the number of income and expense entries and amounts with a balance that should agree with the money you have on hand.

Function 9—LIST CATEGORY

Like Function 1, this function displays the 20 categories. You must then list each category to tabulate the totals for the month. You must do this before you select any other report function. See Figure 2.

Function 10—PERCENTAGE CHART

This function lists the categories and the percentage of you income you have spent on each category. You must perform options 8 and 9 before option 10 or it will not work. See Figure 3.

Function 11—CHANGE RECORD

This is used to change an expense or income record. You need to know the record number. The display informs you to perform a list of expenses or income to note the desired record to be changed. The display is as follows:

```
1- CHANGE INCOME RECORD
2- CHANGE EXPENSE RECORD
99- RETURN TO MENU
SELECT FUNCTION?_
```

If you select mode 1 or mode 2, the program asks which record to change. After your input, the program displays the current record to be changed. As each question is displayed, you may reenter changes. If no change is to be made, press the ENTER key. After all changes are made, the complete changed record is displayed. Pressing ENTER returns you to the CHANGE RECORD function. Enter 99 to return to the main menu.

Function 12—PRINT REPORT

Use this function to generate hard copy output using the RS Quick Printer II. The program warns you to perform the category listing and percentage chart in the video mode before you print the report. It then displays the following menu:

```
1- INCOME LISTING
2- BALANCE CHART
3- CATEGORY LISTING
4- PERCENTAGE CHART
5- RETURN TO MAIN MENU
```

You can print these reports in any order.

Function 13—READ DATA TAPE

This reads in the previous data tape from which you can add more entries. This function also maintains continuity of income and expense record numbers.

CATEGORY	AMOUNT		PERCENTAGE
FOOD	$	232.32	7.11%
AUTO EXP	$	0.00	0.00%
RECREATION	$	0.00	0.00%
HOME EXP	$	0.00	0.00%
DOG EXP	$	0.00	0.00%
CHARGE ACCTS	$	0.00	0.00%
CHILD SPT	$	0.00	0.00%
CLOTHES	$	0.00	0.00%
UTILITIES	$	0.00	0.00%
MISC	$	0.00	0.00%
DOG SHOWS	$	0.00	0.00%
SALES TAX	$	0.00	0.00%
LUNCHES	$	0.00	0.00%
CIG	$	0.00	0.00%
MEDICAL	$	0.00	0.00%
GASOLINE	$	0.00	0.00%
COFFEE	$	0.00	0.00%
HOBBIES	$	0.00	0.00%
CLEANING MATS	$	0.00	0.00%
DRUG ITEMS	$	0.00	0.00%
TOTALS		7.11%	

Figure 3. *Percentage chart*

The Year-To-Date Report

The most beneficial report in my estimation is the YTD report. See Figure 4. The monthly percentage chart is very informative, but you may see some categories fluctuate by as much as 10 percent from one month to another. After three or four months, the average shown by the YTD report is a better guide as to the amounts you should watch in any particular category. The menu for this program is much the same as the menu for Money Minder:

1- INPUT CURRENT MONTH'S DATA
2- READ YTD DATA TAPE
3- LIST YTD REPORT
4- PRINT YTD REPORT
5- WRITE YTD DATA TO TAPE
6- VERIFY DATA TAPE

Function 1—INPUT CURRENT MONTH'S DATA

Input for this function is taken from the month end percentage chart. The program prompts you to enter category amounts one at a time. These amounts are added later to amounts from the previous YTD data tape to produce the final report. When you initiate this program for the first month, Function 2 is not performed. Upon initialization, do not enter BAL FWD (balance forward) for the monthly report as prompted. Enter zero for this amount on the first YTD report. Subsequent monthly entries should include BAL FWD from the month end report as the program asks. If you make a mistake in Function 1, it is best to reload the program and begin again; otherwise, the internally stored totals will never balance.

CAT	YTD TOT $		YTD AVG		YTD%
FOOD	$	232.32	$	38.72	8.02%
AUTO	$	0.00	$	0.00	0.00%
REC	$	0.00	$	0.00	0.00%
HOME X	$	0.00	$	0.00	0.00%
DOG EX	$	0.00	$	0.00	0.00%
CHG AC	$	0.00	$	0.00	0.00%
CHD SP	$	0.00	$	0.00	0.00%
CLOTHS	$	0.00	$	0.00	0.00%
UTILIT	$	0.00	$	0.00	0.00%
MISC	$	0.00	$	0.00	0.00%
DOG SH	$	0.00	$	0.00	0.00%
S TAX	$	0.00	$	0.00	0.00%
LUNCH	$	0.00	$	0.00	0.00%
CIG	$	0.00	$	0.00	0.00%
MEDCAL	$	0.00	$	0.00	0.00%
GAS	$	0.00	$	0.00	0.00%
COFF	$	0.00	$	0.00	0.00%
HOBBS	$	0.00	$	0.00	0.00%
CLN	$	0.00	$	0.00	0.00%
DRUGS	$	0.00	$	0.00	0.00%

TOTAL INCOME		$ 2,895.43
TOTAL EXPENSES		$ 232.32
TOTAL BALANCE		$ 2,663.11
PERCENT FOR EXP	8.02%	

Figure 4. *Year-to-date report*

Function 2—READ YTD DATA TAPE

The previous month end YTD data tape is read in at this time.

Function 3—LIST YTD REPORT

A warning cautions you that you must perform Functions 1 and 2 before listing the report. The next prompt is to enter the month of report in the form JUN 81. The program then asks for the month number, such as 6 for June or 3 for March. This input is the divisor in calculating the YTD report averages. If you initiate the program in January the number will correspond to the month. If started in some other month, be careful.

Function 4—PRINT YTD REPORT

This operates like Function 3. It asks for the month of report and month number as a double input entry. A comma *must* separate the two, such as JUN 81, 6. If you have performed Function 3, you need only press ENTER in answer to the input prompt.

Function 5—WRITE YTD DATA TO TAPE

You should use a different tape for this function so you have the previous month's data tape as a backup.

Function 6—VERIFY DATA TAPE

Rewind the tape written with function 5 and perform this step. Any difference between tape and memory will be noted and displayed.

General Summary and Hints

1) Record your income and expenses daily in a notebook or ledger. The first entry for income of the month should be your cash on hand noted as BAL FWD. Consolidate frequently noted expenses such as sales tax and daily paper purchases so you don't exceed 200 entries per month.
2) Determine the frequency of your entry sessions. The more sessions you have the more time it takes to make data tapes. I find that twice per month is adequate.
3) At the end of each entry session, write data to tape and verify it. It is helpful to write on the tape and in the notebook the number of income and expense entries as noted on the video display.
5) At month's end, perform menu functions 8, 9, and 10. Print a report if you can. Check the monthly balance with the money you have on hand. If my money on hand is less than the balance for the month, I note the difference under category MISC as $$$ LOST.
6) As a final check for accuracy, the balance amount from the monthly balance charge should equal the total balance shown on the YTD report.

Program Listing 1. *Money Minder*

Encyclopedia Loader

```
  1 REM   * PROGRAM: M O N E Y  M I N D E R  BY BILL LOVEYS
  2 REM   * 4812 JAMES ST., E. SYRACUSE, NY  13057
  3 REM   *  REQUIREMENTS: 16K, LEVEL II, OPTION: QUICK PRINTER II
 10 CLEAR 2000:
    D$ = "$##,###.##":
    M$ = "$#,###.##":
    PC$ = "##.##%"
 20 DEFINT C,I,N,T,Z
 30 DIM XD$(200),XA(200),CA(200),CK(200),XD(200),DI$(20),SI$(20),AI(
    20),XC(20),A$(20),XB(20)
 40 CLS :
    PRINT TAB(16)"M O N E Y    M I N D E R":
    PRINT
 50 PRINT "1  ENTER EXPENSES"; TAB(32)"7  MEM & STRING STORAGE"
 60 PRINT "2  LIST EXPENSES"; TAB(32)"8  BALANCE CHART"
 70 PRINT "3  ENTER INCOME"; TAB(32)"9  LIST CATEGORY"
 80 PRINT "4  LIST INCOME"; TAB(32)"10 PERCENTAGE CHART"
 90 PRINT "5  RECORD  DATA"; TAB(32)"11 CHANGE RECORD"
100 PRINT "6  VERIFY DATA TAPE"; TAB(32)"12 PRINT REPORT"
110 PRINT TAB(16)"13 READ DATA TAPE"
120 IF MEM < 200 PRINT "C A U T I O N ----MEMORY UNDER 200":
    PRINT "PERFORM FUNCTION #7 THEN RECORD EXSISTING DATA":
    PRINT "ADJUST STRING STORAGE SPACE AS NEEDED":
    PRINT
130 PRINT :
    INPUT "ENTER DESIRED FUNCTION";Z:
    IF Z > 13
     THEN
      40
140 ON Z GOTO 150,310,400,500,570,730,910,960,1060,1220,1290,1540,19
    40
150 CLS :
    PRINT TAB(16)"* * EXPENSE CATAGORIES * *":
    PRINT
160 RESTORE :
    FOR I = 1 TO 10:
     READ A$:
     PRINT I;" ";A$:
     NEXT
170 P = 160:
    FOR I = 11 TO 20:
     READ A$:
     PRINT @P,I;" ";A$;:
     P = P + 64:
     NEXT :
    IF Z = 9 GOTO 1070
180 PRINT :
    PRINT :
    PRINT "TO RETURN TO MENU ENTER 99":
    PRINT :
    INPUT "ENTER CATAGORY NUMBER";Z:
    RESTORE :
    CLS
190 IF Z = 99 GOTO 40
200 PRINT TAB(20)"* * EXPENSES * *":
    PRINT :
    PRINT
210 FOR I = 1 TO Z:
     READ A$:
     NEXT :
    TX = TX + 1:
    PRINT TAB(20)A$:
    PRINT :
    CA(TX) = Z
220 CK(TX) = 9999:
    INPUT "ENTER CHECK NUMBER";CK(TX)
230 INPUT "ENTER  DESCRIPTION";XD$(TX)
```

Program continued

```
240 IF LEN(XD$(TX)) > 12 PRINT "KEEP DESC UNDER 13 CHARACTERS":
    GOTO 230
250 INPUT "ENTER EXPENSE AMOUNT";XA(TX)
260 IF XA(TX) < .001 PRINT "REDO":
    GOTO 250
270 XD(TX) = XD(TX - 1):
    INPUT "ENTER DAY OF MONTH";XD(TX)
280 PRINT :
    PRINT "RECORD #"; TAB(10)"CHECK #"; TAB(20)"DESCRIPTION";
    TAB(40)"AMOUNT"; TAB(50)"DAY"
290 PRINT TAB(1)TX; TAB(10)CK(TX); TAB(20)XD$(TX); TAB(40);:
    PRINT USING M$;XA(TX);:
    PRINT TAB(50)XD(TX)
300 PRINT :
    INPUT "PRESS <ENTER> TO CONTINUE";Z:
    GOTO 150
310 CLS :
    PRINT TAB(20)"EXPENSE LISTING":
    PRINT
320 PRINT "REC #"; TAB(6)"CATEGORY"; TAB(20)"CHECK #"; TAB(30)"DESCR
    IPTION"; TAB(50)"AMOUNT"; TAB(60)"DAY":
    PRINT STRING$(63,61)
330 CN = 0:
    FOR I = 1 TO TX:
     RESTORE :
     CN = CN + 1
340  FOR N = 1 TO CA(I):
      READ A$:
      NEXT
350  PRINT I; TAB(6)A$; TAB(20)CK(I); TAB(30)XD$(I); TAB(50);
360  PRINT USING M$;XA(I);:
     PRINT TAB(59)XD(I)
370  IF CN = 10 PRINT @896,;:
     INPUT "PRESS <ENTER> TO CONTINUE";Z
380  IF CN = 10 PA = 192:
     FOR X = 1 TO 11:
      PRINT @PA,"":
      PA = PA + 64:
      NEXT X:
     PRINT @256,;:
     CN = 0
390  NEXT I:
    PRINT @896,;:
    INPUT "PRESS <ENTER> FOR MENU";Z:
    GOTO 40
400 CLS :
    PRINT TAB(16)"* * INCOME ENTRIES * *":
    PRINT STRING$(63,61):
    PRINT
410 PRINT "ENTER <99> FOR DATE TO END SESSION":
    PRINT STRING$(63,45):
    PRINT
420 TI = TI + 1
430 INPUT "ENTER DATE OF INCOME";DI$(TI)
440 IF DI$(TI) = "99" TI = TI - 1:
    GOTO 40
450 INPUT "ENTER INCOME SOURCE";SI$(TI)
460 IF LEN(SI$(TI)) > 9 PRINT "KEEP SOURCE UNDER 10 CHARACTORS":
    GOTO 450
470 INPUT "ENTER INCOME AMOUNT";AI(TI)
480 N = 320
490 FOR I = 1 TO 10:
     PRINT @N,"":
     N = N + 64:
     NEXT :
    PRINT @384,;:
    GOTO 420
500 CLS :
    PRINT TAB(21)"* * INCOME LISTING * *":
    PRINT STRING$(63,61):
    PRINT
```

```
510 PRINT "REC #"; TAB(10)"DATE",,"SOURCE","AMOUNT":
    PRINT STRING$(63,45)
520 FOR I = 1 TO TI:
     PRINT I; TAB(10)DI$(I),SI$(I),;:
     PRINT USING D$;AI(I)
530  IF I = 9 GOSUB 550
540  NEXT I:
    INPUT "PRESS <ENTER> TO RETURN TO MENU";Z:
    GOTO 40
550 INPUT "PRESS <ENTER> TO CONTINUE";Z
560 N = 320:
    FOR Z = 1 TO 10:
     PRINT @N,"":
     N = N + 64:
     NEXT :
    PRINT @320,;:
    RETURN
570 CLS :
    PRINT "READY TAPE TO RECORD (WRITE)"
580 INPUT "PRESS <ENTER> WHEN READY";Z
590 INPUT "ENTER MONTH OF REPORT (I.E. MAR 80)";MO$
600 CLS :
    PRINT TAB(20)"WRITTING TO TAPE":
    PRINT STRING$(63,61)
610 PRINT "DATE OF RECORD"; TAB(40)"ENTRIES"
620 PRINT MO$; TAB(32)"INCOME"; TAB(43)TI:
    PRINT TAB(32)"EXPENSES"; TAB(43)TX
630 PRINT # - 1,MO$,TI,TX:
    PRINT STRING$(63,45)
640 PRINT "REF #-DATE-CHECK #-DESCRIPTION"; TAB(45)"AMOUNT   REC #"
650 FOR N = 1 TO TX
660  PRINT CA(N); TAB(5)XD(N); TAB(10)CK(N); TAB(20)XD$(N);
     TAB(45);
670  PRINT USING M$;XA(N);:
     PRINT TAB(56)N:
     PRINT # - 1,CA(N),XD(N),CK(N),XD$(N),XA(N):
     PRINT @448,;
680  NEXT N:
    PRINT @384, STRING$(63," ")
690 PRINT @384,"DATE OF INCOME","SOURCE","AMOUNT OF INCOME"
700 FOR N = 1 TO TI:
     PRINT DI$(N),SI$(N),;:
     PRINT USING D$;AI(N)
710  PRINT # - 1,DI$(N),SI$(N),AI(N)
720  NEXT N:
    PRINT @960,;:
    INPUT "PRESS <ENTER> FOR MENU";Z:
    GOTO 40
730 CLS :
    PRINT "REWIND DATA TAPE - READY CASSETTE TO READ":
    PRINT
740 INPUT "PRESS <ENTER> WHEN READY";Z:
    CLS
750 PRINT TAB(20)"DATA TAPE VERIFICATION":
    PRINT STRING$(63,61)
760 INPUT # - 1,A$,A,B:
    PRINT "DATE OF RECORD:"; TAB(20)A$
770 PRINT "INCOME ENTRIES:"; TAB(20)A:
    PRINT "EXPENSE ENTRIES:"; TAB(20)B
780 PRINT STRING$(63,45):
    PRINT TAB(20)" IN MEMORY"
790 PRINT "REF#-DATE-CHECK#-DESCRIPTION"; TAB(40)"AMOUNT"
800 FOR I = 1 TO TX:
     PRINT @512, CA(I); TAB(5)XD(I); TAB(10)CK(I); TAB(20)XD$(I);
     TAB(40)XA(I)
810  PRINT TAB(20)"TAPE INPUT"
820  INPUT # - 1,A,B,C,E$,F
830  PRINT A; TAB(5)B; TAB(10)C; TAB(20)E$; TAB(40)F
840  IF A < > CA(I) OR B < > XD(I) OR C < > CK(I) OR E$ < > XD$(I)
     OR F < > XA(I) PRINT @896,"BAD DATA INPUT - TRY AGAIN":
     GOTO 900
```

Program continued

```
850  NEXT I:
    PA = 384:
    FOR N = 1 TO 6:
     PRINT @PA,:
     PA = PA + 64:
     NEXT N
860 FOR I = 1 TO TI :
     PRINT @448,"DATE","SOURCE","AMOUNT"
870  INPUT # - 1,A$,E$,E:
     PRINT DI$(I),SI$(I),AI(I):
     PRINT TAB(20)"TAPE INPUT"
880  PRINT A$,E$,E:
     IF DI$(I) < > A$ OR SI$(I) < > E$ OR AI(I) < > E PRINT "BAD DAT
     A INPUT":
     GOTO 900
890  NEXT I
900 INPUT "PRESS <ENTER> FOR MENU";Z:
    GOTO 40
910 CLS :
    PRINT "MONTH OF RECORD  ";MO$:
    PRINT
920 PRINT "WITH";TI;" INCOME ENTRIES AND";TX;"EXPENSE ENTRIES"
930 PRINT "REMAINING MEMORY IS"; MEM
940 PRINT "REMAINING STRING STORAGE SPACE IS "; FRE(S$):
    PRINT
950 INPUT "PRESS <ENTER> FOR MENU";Z:
    GOTO 40
960 CLS :
    PRINT TAB(18)"B A L A N C E   C H A R T":
    PRINT TAB(29)MO$:
    PRINT STRING$(63,61):
    MI = 0:
    MX = 0:
    MB = 0
970 FOR I = 1 TO TI:
     MI = MI + AI(I):
     NEXT
980 PRINT "I N C O M E"; TAB(18)TI;"ENTRIES"; TAB(42);:
    PRINT USING D$;MI:
    PRINT STRING$(63,45)
990 FOR I = 1 TO TX:
     IF CK(I) = 7777 GOTO 1010
1000  MX = MX + XA(I)
1010  NEXT I
1020 PRINT "E X P E N S E S"; TAB(18)TX;"ENTRIES"; TAB(42);:
    PRINT USING D$;MX:
    PRINT STRING$(63,45)
1030 MB = MI - MX
1040 PRINT "B A L A N C E"; TAB(42);:
    PRINT USING D$;MB:
    PRINT STRING$(63,61):
    PRINT @896,;
1050 INPUT "PRESS <ENTER> FOR MENU";Z:
    GOTO 40
1060 CN = 0:
    XS = 0:
    CLS :
    PRINT TAB(15)"* * LISTING BY CATEGORY * *":
    PRINT :
    GOTO 160
1070 PRINT :
    PRINT :
    PRINT "ENTER 99 FOR MENU":
    PRINT
1080 INPUT "ENTER CATEGORY TO BE LISTED";ZC:
    IF ZC = 99 GOTO 40
1090 XB(ZC) = 0:
    XC(ZC) = 0:
    CLS :
    RESTORE :
    FOR I = 1 TO ZC:
```

```
      READ A$:
      NEXT :
     RESTORE
1100 PRINT TAB((64 - LEN(A$)) / 2)A$:
     PRINT STRING$(63,61)
1110 PRINT "DAY  CHECK # DESCRIPTION     AMOUNT":
     PRINT STRING$(63,45)
1120 FOR I = 1 TO TX:
      IF ZC = 1 AND CA(I) = 1 AND RIGHT$(XD$(I),1) = "*" XS = XS
      + XA(I):
      PS = XS
1130  IF CA(I) = ZC PRINT XD(I); TAB(6)CK(I); TAB(13)XD$(I);
      TAB(29);:
      PRINT USING D$;XA(I):
      CN = CN + 1
1140  IF CK(I) = 7777 AND CA(I) = ZC
       THEN
        XB(ZC) = XB(ZC) + XA(I):
        GOTO 1160
1150  IF CA(I) = ZC
       THEN
        XC(ZC) = XC(ZC) + XA(I)
1160  IF CN > 9 INPUT "PRESS <ENTER> TO CONTINUE";Z:
      PA = 192:
      FOR N = 1 TO 11:
       PRINT @PA,:
       PA = PA + 64:
       NEXT N:
      CN = 0:
      PRINT @256,;
1170  NEXT I
1175 PRINT :
     IF ZC > 1 AND XR(ZC) > 0 PRINT "IRS"; TAB(10);:
     PRINT USING M$;XR(ZC)
1180 IF ZC = 1 PRINT "EAT OUT"; TAB(10);:
     PRINT USING M$;XS
1190 PRINT "TOTAL"; TAB(10);:
     PRINT USING D$;XC(ZC)
1200 PRINT "CHARGES"; TAB(10);:
     PRINT USING D$;XB(ZC)
1210 PRINT :
     INPUT "PRESS <ENTER> TO CONTINUE";Z:
     GOTO 1060
1220 CLS :
     PRINT "PERFORM FUNCTIONS 8 AND 9 IN VIDEO MODE BEFORE LISTING
            PERCENTAGE CHART":
     PRINT :
     PRINT :
     INPUT "PRESS <ENTER> TO CONTINUE";Z
1225 CLS :
     PRINT CHR$(23):
     PRINT " CATEGORY PERCENTAGE CHART":
     CN = 0
1230 PRINT STRING$(32,35):
     PRINT "CATEGORY"; TAB(22)"%%%":
     PRINT STRING$(32,45)
1240 RESTORE :
     FOR I = 1 TO 20:
      CN = CN + 1:
      IF CN = 10 INPUT "PRESS <ENTER>";Z:
      CLS :
      PRINT CHR$(23)
1250  READ A$(I)
1260  PRINT A$(I); TAB(20);:
      PRINT USING PC$;(XC(I) / MI) * 100
1270  NEXT I:
     PRINT "TOTALS"; TAB(20);:
     PRINT USING PC$;(MX / MI) * 100
1280 INPUT "PRESS <ENTER>";Z:
     GOTO 40
1290 CLS :
```

Program continued

```
     PRINT "PERFORM LIST OF INCOME OR EXPENSES TO DETERMINE RECORD NU
     MBER.":
     PRINT :
     PRINT
1300 PRINT "1 - CHANGE INCOME RECORD":
     PRINT "2 - CHANGE EXPENSE RECORD"
1310 PRINT "99- RETURN TO MENU":
     PRINT :
     PRINT :
     INPUT "SELECT FUNCTION";Z
1320 ON Z GOTO 1330,1420:
     IF Z < > 1 OR Z < > 2
      THEN
       40
1330 INPUT "ENTER INCOME RECORD # TO BE CHANGED";IC:
     CLS :
     IF IC > TI
      THEN
       40
1340 PRINT :
     PRINT "REC #","DATE","SOURCE","AMOUNT":
     PRINT STRING$(64,45)
1350 PRINT IC,DI$(IC),SI$(IC),;:
     PRINT USING M$;AI(IC):
     PRINT :
     PRINT
1360 INPUT "ENTER DATE CHANGE";DI$(IC)
1370 INPUT "ENTER SOURCE CHANGE";SI$(IC):
     IF LEN(SI$(IC)) > 9 PRINT "KEEP SOURCE UNDER 10 CHARS.":
     GOTO 1370
1380 INPUT "ENTER AMOUNT CHANGE";AI(IC)
1390 PRINT "RECORD #";IC;"NOW READS:":
     PRINT
1400 PRINT IC,DI$(IC),SI$(IC),;:
     PRINT USING M$;AI(IC)
1410 PRINT :
     INPUT "PRESS <ENTER> TO RETURN";Z:
     GOTO 1290
1420 INPUT "ENTER EXPENSE RECORD TO BE CHANGED";IC:
     CLS :
     IF IC > TX
      THEN
       40
1430 PRINT "REC #--CAT#--CK #---DESCRIPTION----AMOUNT------DAY--":
     PRINT STRING$(52,45)
1440 PRINT IC; TAB(8)CA(IC); TAB(13)CK(IC); TAB(21)XD$(IC); TAB(35);
1450 PRINT USING M$;XA(IC);:
     PRINT TAB(47)XD(IC)
1460 PRINT :
     INPUT "CATEGORY REFERENCE # CHANGE";CA(IC)
1470 INPUT "CHECK # CHANGE";CK(IC)
1480 INPUT "DESCRIPTION CHANGE";XD$(IC)
1490 INPUT "AMOUNT CHANGE";XA(IC)
1500 INPUT "DAY OF MONTH CHANGE";XD(IC):
     PRINT
1510 PRINT "RECORD NOW READS:":
     PRINT
1520 PRINT IC; TAB(8)CA(IC); TAB(13)CK(IC); TAB(21)XD$(IC); TAB(35);
1530 PRINT USING M$;XA(IC);:
     PRINT TAB(47)XD(IC):
     INPUT "PRESS <ENTER> TO RETURN";Z:
     GOTO 1290
1540 CLS :
     PRINT :
     PRINT "BEFORE LISTING PRINTER OUTPUT FUNCTION 8 (BALANCE CHART)
     AND    FUNCTION 9 (LIST CATEGORY) MUST HAVE BEEN PERFORMED USING
      THE   VIDEO MODE."
1550 PRINT :
     PRINT :
     INPUT "PRESS <ENTER> TO CONTINUE";Z
1560 CLS :
```

```
     PRINT TAB(21)"LINE PRINTER OPTIONS":
     PRINT :
     PRINT
1570 PRINT "1 - INCOME LISTING":
     PRINT "2 - BALANCE CHART"
1580 PRINT "3 - CATEGORY LISTING":
     PRINT "4 - PERCENTAGE CHART"
1590 PRINT "99- RETURN TO MAIN MENU":
     PRINT :
     PRINT
1600 INPUT "ENTER DESIRED FUNCTION";Z:
     CLS
1610 ON Z GOTO 1620,1670,1750,1860:
     IF Z > 4
      THEN
       40
1620 PRINT @468,"INCOME LISTING"
1630 LPRINT STRING$(31,127):
     LPRINT CHR$(15)"  INCOME LIST":
     LPRINT STRING$(31,127)
1640 LPRINT CHR$(13)"-- DATE -- SOURCE ---- AMOUNT -" CHR$(13)
1650 FOR I = 1 TO TI:
      LPRINT DI$(I); TAB(11)SI$(I); TAB(22);:
1660  LPRINT USING M$;AI(I):
      NEXT :
     LPRINT CHR$(13):
     GOTO 1560
1670 PRINT @468,"BALANCE CHART":
     LPRINT STRING$(31,127):
     LPRINT CHR$(15)"  BALANCE CHART"
1680 LPRINT TAB(13)MO$:
     LPRINT STRING$(31,127) CHR$(13)
1690 LPRINT "-----------/ TOTAL INCOME /----"
1700 LPRINT TI;"ENTRIES"; TAB(15);:
     LPRINT USING D$;MI
1710 LPRINT "-----------/ TOTAL EXPENSES /--"
1720 LPRINT TX;"ENTRIES"; TAB(15);:
     LPRINT USING D$;MX
1730 LPRINT "--------------/ BALANCE /-----"
1740 LPRINT TAB(15);:
     LPRINT USING D$;MI - MX:
     LPRINT STRING$(31,45):
     GOTO 1560
1750 PRINT @468,"CATEGORY LISTING"
1760 LPRINT STRING$(31,127):
     LPRINT CHR$(15)" CATEGORY LIST"
1770 LPRINT STRING$(31,127) CHR$(13)
1780 RESTORE :
     FOR N = 1 TO 20:
      READ A$
1785  IF XC(N) = 0
       THEN
        NEXT N
1790 IF N > 20 GOTO 1560 :
      ELSE
       LPRINT STRING$(31,61):
       LPRINT TAB((32 - LEN(A$)) / 2)A$:
       LPRINT STRING$(31,61)
1800 FOR I = 1 TO TX
1810  IF CA(I) = N LPRINT XD(I); TAB(3)CK(I);XD$(I); TAB(22);:
      LPRINT USING M$;XA(I)
1820  NEXT I:
     LPRINT
1830 IF N = 1 LPRINT "EAT OUT"; TAB(10);:
     LPRINT USING M$;PS
1840 IF XB(N) > 0 LPRINT "CHARGES"; TAB(10);:
     LPRINT USING M$;XB(N)
1850 LPRINT "TOTAL"; TAB(10);:
     LPRINT USING M$;XC(N):
     LPRINT :
     NEXT N:
```

Program continued

```
     GOTO 1560
1860 PRINT @468,"PERCENTAGE CHART"
1870 LPRINT STRING$(32,127) CHR$(15)"   PERCENTAGE":
     LPRINT CHR$(15) TAB(6)"CHART"
1880 LPRINT STRING$(32,127):
     LPRINT TAB(12)MO$; CHR$(13) STRING$(31,45)
1890 LPRINT "CATEGORY------AMOUNT--PERCENTAGE"
1900 RESTORE :
     FOR I = 1 TO 20:
      READ A$
1910  LPRINT A$; TAB(14);:
      LPRINT USING M$;XC(I);:
      LPRINT TAB(24);:
      LPRINT USING PC$;XC(I) / MI * 100
1920  NEXT :
     LPRINT STRING$(31,127):
     LPRINT "TOTALS"; TAB(20);
1930 LPRINT USING PC$;MX / MI * 100:
     GOTO 1560
1940 CLS :
     PRINT "READY TAPE TO READ":
     PRINT :
     INPUT "PRESS <ENTER> WHEN READY";Z
1950 CLS :
     PRINT TAB(20)"READING DATA TAPE":
     PRINT STRING$(63,61)
1960 INPUT # - 1,MO$,TI,TX
1970 PRINT TAB(30)MO$:
     PRINT TX;"EXPENSE ENTRIES":
     PRINT TI;"INCOME ENTRIES"
1980 PRINT :
     PRINT "EXPENSE RECORD #";:
     FOR I = 1 TO TX:
      INPUT # - 1,CA(I),XD(I),CK(I),XD$(I),XA(I)
1990  PRINT @401,I;:
      NEXT I:
     PRINT
2000 PRINT "INCOME RECORD #";:
     FOR I = 1 TO TI:
      INPUT # - 1,DI$(I),SI$(I),AI(I)
2010  PRINT @465,I;:
      NEXT I:
     PRINT :
     PRINT
2020 INPUT "PRESS <ENTER> FOR MENU";Z:
     GOTO 40
2030 DATA FOOD,AUTO EXP,RECREATION,HOME EXP,DOG EXP,CHARGE ACCTS,CHIL
     D SPT,CLOTHES,UTILITIES,MISC
2040 DATA DOG SHOWS,SALES TAX,LUNCHES,CIG,MEDICAL,GASOLINE,COFFEE,HOB
     BIES,CLEANING MATS,DRUG ITEMS
```

Program Listing 2. *Year-to-date report*

```
 1 REM  ** MONEY MINDER/YEAR-TO-DATE REPORT **
 2 REM  ** BILL LOVEYS
 3 REM  ** 4812 JAMES STREET
 4 REM  ** EAST SYRACUSE, NY 13057
10 CLEAR 200:
   DIM ME(19),YT(19),YD(19)
20 Y$ = "$##,###.##":
   M$ = "$###.##":
   P$ = "##.##%":
   R$ = "$#,###.##
30 CLS :
   PRINT TAB(14)"MONEY MINDER YEAR-TO-DATE REPORT
40 PRINT :
   PRINT TAB(26)"M E N U":
```

```
    PRINT
 50 PRINT "1-INPUT CURRENT MONTHS DATA"
 60 PRINT "2-READ YTD DATA TAPE"
 70 PRINT "3-LIST YTD REPORT"
 80 PRINT "4-PRINT YTD REPORT"
 90 PRINT "5-WRITE YTD DATA TO TAPE"
100 PRINT "6-VERIFY DATA TAPE":
    PRINT @960,;
110 INPUT "ENTER DESIRED FUNCTION";Z:
    IF Z > 6
     THEN
      30
120 ON Z GOTO 130,220,280,520,450,700
130 CLS :
    RESTORE :
    MI = 0:
    MX = 0:
    BF = 0
140 FOR X = 0 TO 19:
     READ A$:
     PRINT A$,:
     INPUT "ENTER MONTH END AMOUNT";ME(X)
150  IF X = 11 CLS
160  CK = CK + ME(X):
     NEXT :
    PRINT
170 INPUT "ENTER MONTH END INCOME";MI:
    INPUT "ENTER MONTH END EXPENSES";MX:
180 CK = CK * 100:
    MX = MX * 100:
    IF INT(CK) < > INT(MX) PRINT "DATA ERROR--REDO FUNCTION #1"
190 CK = CK / 100:
    MX = MX / 100
200 INPUT "ENTER THE BALANCE FORWARD AMOUNT LISTED ON PRECEDING MONT
    HS    INCOME LISTING. (EXAMPLE: IF THIS IS APR Y-T-D REPORT THEN
     ENTER BAL FWD FROM MARCH)";BF:
    MI = MI - BF
210 PRINT @960,;:
    INPUT "PRESS <ENTER> TO RETURN TO MENU";Z$:
    GOTO 30
220 CLS :
    INPUT "READY TAPE TO READ - PRESS <ENTER> WHEN READY";Z$
230 CLS :
    PRINT "R E A D I N G":
    PRINT :
    INPUT # - 1,MY$,TI,TX
240 PRINT "CATEGORY","YTD TOTALS":
    PRINT STRING$(32,61)
250 FOR X = 0 TO 19:
     INPUT # - 1,YT(X):
     POKE 16553,255:
     READ A$:
     PRINT A$,:
     PRINT USING Y$;YT(X)
260  IF X = 10 PRINT @960,;:
     INPUT "PRESS <ENTER> TO CONTINUE";Z$:
     CLS
270  NEXT :
    GOTO 210
280 CLS :
    RESTORE :
    PRINT TAB(24)"C A U T I O N !":
    PRINT
290 PRINT "FUNCTIONS 1 AND 2 MUST HAVE BEEN PERFORMED BEFORE LISTING
     REPORT":
    PRINT :
    PRINT
300 YI = TI + MI:
    YX = TX + MX
310 INPUT "ENTER MONTH OF REPORT (IE FEB 80)";MY$
320 INPUT "ENTER THE MONTH NUMBER (IE 2 FOR FEB)";MN
```

Program continued

```
330 PRINT :
    INPUT "PRESS <ENTER> TO CONTINUE";Z$:
    CLS
340 PRINT TAB(21)"M O N E Y  M I N D E R"
350 PRINT TAB(26)"YEAR-TO-DATE":
    PRINT TAB(29)MY$:
    PRINT STRING$(63,61)
360 PRINT "CATEGORY",; TAB(17)"YTD TOTAL--CURRENT---YTD AVG----YTD P
    ERCENT":
    PRINT STRING$(63,45):
    RESTORE
370 FOR X = 0 TO 19:
     YD(X) = YT(X) + ME(X):
     PC = YD(X) / YI:
     YA = YD(X) / MN
380  READ A$:
     PRINT A$,; TAB(15)"--";:
     PRINT USING Y$;YD(X);:
     PRINT "--";:
     PRINT USING M$;ME(X);:
     PRINT "--"; TAB(38);:
     PRINT USING Y$;YA;:
     PRINT "---- ";:
     PRINT USING P$;PC * 100
390  IF X = 4 OR X = 9 OR X = 14 PRINT @896,;:
     INPUT "PRESS <ENTER> TO CONTINUE";Z$:
     PRINT @384,;
400  NEXT :
    PRINT @896,;:
    INPUT "PRESS <ENTER> TO CONTINUE";Z$:
    CLS
410 PRINT CHR$(23):
    PRINT "   YTD INCOME":
    PRINT STRING$(31,61):
    PRINT USING Y$;YI:
    PRINT STRING$(31,61)
420 YX = TX + MX:
    PRINT "   YTD EXPENSES":
    PRINT STRING$(31,61):
    PRINT USING Y$;YX:
    PRINT STRING$(31,61)
430 BA = YI - YX:
    PRINT "    BALANCE":
    PRINT STRING$(31,61):
    PRINT USING Y$;BA:
    PRINT STRING$(31,61)
440 PRINT @960,;:
    INPUT "PRESS FOR MENU";Z$:
    GOTO 30
450 CLS :
    PRINT "READY TAPE TO WRITE CURRENT DATA"
460 PRINT :
    INPUT "PRESS <ENTER> WHEN READY";Z$
470 CLS :
    PRINT "W R I T I N G   T A P E"
480 PRINT # - 1,MY$,YI,YX:
    PRINT "YTD INCOME= ";:
    PRINT USING Y$;YI:
    PRINT "YTD EXPENSES= ";:
    PRINT USING Y$;YX:
    PRINT
490 FOR X = 0 TO 19:
     PRINT # - 1,YD(X):
     PRINT X;:
     NEXT
500 PRINT :
    PRINT "PERFORM FUNCTION # 6 TO VERIFY THIS TAPE"
510 GOTO 210
520 CLS :
    PRINT "READY PRINTER FOR REPORT"
530 PRINT :
```

```
    INPUT "PRESS <ENTER> WHEN READY";Z$:
    CLS :
    RESTORE
540 PRINT :
    INPUT "ENTER MONTH OF REPORT, MONTH NUMBER";MY$,MN:
    CLS
550 PRINT TAB(21)"M O N E Y  M I N D E R":
    PRINT TAB(26)"YEAR-TO-DATE":
    PRINT TAB(29)MY$
560 LPRINT CHR$(15) TAB(5)"MONEY" CHR$(13) CHR$(15) TAB(5)"MINDER"
    CHR$(13) STRING$(31,127)
570 LPRINT CHR$(15)"  YEAR-TO-DATE":
    LPRINT CHR$(15) TAB(6)MY$:
    LPRINT STRING$(31,127)
580 LPRINT " CAT  -YTD TOT $--YTD AVG--YTD%"
590 LPRINT STRING$(31,45):
    YI = TI + MI:
    YX = TX + MX
600 FOR X = 0 TO 19:
     READ A$:
     NEXT
610 FOR X = 0 TO 19:
     YD(X) = YT(X) + ME(X):
     PC = YD(X) / YI:
     YA = YD(X) / MN
620  READ A$:
     LPRINT A$; TAB(6)"-";:
     LPRINT USING R$;YD(X);:
     LPRINT "-";:
     LPRINT USING M$;YA;:
     LPRINT "-";:
     LPRINT USING P$;PC * 100
630  NEXT
640 LPRINT STRING$(31,127)
650 LPRINT "TOTAL INCOME"; TAB(18);:
    LPRINT USING Y$;YI
660 LPRINT "TOTAL EXPENSES"; TAB(18);:
    LPRINT USING Y$;YX
670 LPRINT "TOTAL BALANCE"; TAB(18);:
    BA = YI - YX:
    LPRINT USING Y$;BA
680 PC = YX / YI * 100:
    LPRINT "PERCENT FOR EXP"; TAB(18);:
    LPRINT USING P$;PC
690 LPRINT STRING$(31,127):
    GOTO 210
700 CLS :
    PRINT "REWIND DATA TAPE - READY TAPE TO READ":
    PRINT
710 INPUT "PRESS <ENTER> TO CONTINUE";Z$:
    CLS
720 PRINT "TAPE VERIFY","IN MEMORY":
    PRINT STRING$(32,45)
730 INPUT # - 1,X$,X,Y:
    PRINT X$;" ";X;Y,MY$;" ";YI;YX
740 FOR X = 0 TO 19:
     INPUT # - 1,A:
     PRINT A,YT(X):
     IF A < > YT(X) PRINT "BAD DATA TAPE":
     GOTO 770
750  NEXT
760 PRINT :
    PRINT "TAPE VERIFICATION COMPLETE":
    GOTO 210
770 PRINT "TRY AGAIN":
    GOTO 210
780 DATA FOOD,AUTO EXPENSES,RECREATION,HOME EXPENSES,DOG EXPENSES
790 DATA CHARGE ACCOUNTS,CHILD SUPPORT,CLOTHES,UTILITIES,MISC
800 DATA DOG SHOWS,SALES TAX,LUNCHES,CIGARETTES,MEDICAL
810 DATA GASOLINE,COFFEE,HOBBIES,CLEANING MATS,DRUG ITEMS
820 DATA FOOD,AUTO,REC,HOME X,DOG EX,CHG AC,CHD SP,CLOTHS,UTILIT,MIS
    C,DOG SH,S TAX,LUNCH,CIG,MEDCAL,GAS,COFF,HOBBS,CLN,DRUGS
```

Groupies: A Strategy to Group Like Objects

by Richard Ramella

A toddler's toy I saw in the waiting room of my physician provided the inspiration for this program. I have used this Level II program to index a book's subject matter, to match people of like interests for social purposes, to cross-reference lecture notes in search of linked ideas, and as a file that identifies the numbers of all photographic proof sheets on which a specified person, place, or event appears.

The toy that led to this was a plastic cube filled with beads of various sizes. There were several full shelves with holes in them. The holes in each shelf were smaller than those on the shelf above. When the toy was tipped, the beads fell through the levels until they came to a hole too small for them to fit through. It was a bright variation of an industrial grader in which a product of many pieces—whether coal or walnuts—is shaken down through a series of screens. Pieces of similar size are shunted laterally to a common bin.

My first association of the toy with a multi-dimensional array was a false lead but did set me thinking about creating a grouping program. The illustration of this article demonstrates both the mental model I envisioned and the symbolic workings of the program. I realize the program is a string comparison, but for conceptual purposes, I will describe it in terms that make its workings more understandable. I decided to create a program which simulates a cube with two shelves. The top shelf has a series of holes with objects poised above them. On a string comparison command, all objects sharing a stated quality fall through and reassemble as a subgroup on the shelf below. The version of the program given here (see Program Listing) groups and lists names of people who share interests. It can produce several lists, each identifying a specific interest, in one run.

Line 120 clears 255 bytes for string space. That's all you need, because the single-dimension status of the program deals with one data line at a time. Line 130 dimensions B$ for a depth of five elements. You must dimension B$ to contain as many elements as there are data lines. Lines 140 through 180 are the data lines. Spacing is crucial if the program is to run correctly. Include one space after DATA. The names in this program must be contained in spaces 2 through 19 after the word DATA, with blank spaces filling any leftover area. Space 20 must be blank. The three-letter codes must begin at space 21. Each code but the last one in the data line has one space after it. If you run out of space for data entry, make a new line starting with the same name as the previous line and enter the rest of the codes.

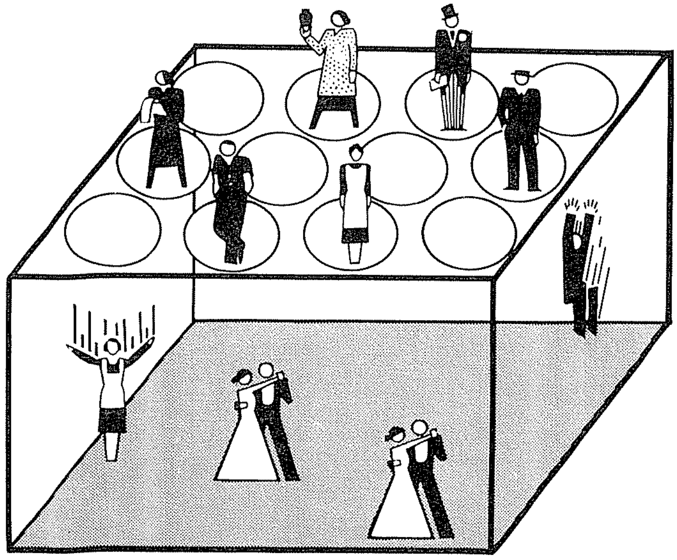
It is important to create a code menu sheet and keep it consistent. The program won't group two people interested in computers if you give one person the code COM and the other the code CPT. The code menu for this program is as follows:

REA	Reading
SEW	Sewing
BEE	Beekeeping
COM	Computers
COO	Cooking
SWI	Swimming
HIK	Hiking
DAN	Dancing

There is a gap in the Program Listing between lines 180 and 2000. This allows you to enter as many data lines as you need or as many as a system's memory allows. Line 2000 asks for entry of both the code and the full word describing the quality sought, separated by a comma.

Lines 2040 through 2120 are the workhorses. Refer to the "Strings" chapter in the Level II manual for more information. The program searches the rightward portion of the string for a code. If the computer finds the code, it prints the leftward portion of the same string. You can use the leftward area of the string to hold a page number when you use the program as an indexer. In complex indexing runs, I change all PRINT statements to LPRINT to obtain hard copy.

Meanwhile, I have purchased a toy like the one I saw in my physician's office. I keep it by my TRS-80 as a reminder that complex utilitarian ideas often flow from a simple sense of playfulness.



Program Listing. *Groupies*

```
100 REM  * GROUPIES  BY RICHARD RAMELLA *
110 CLS
120 CLEAR (255)
130 DIM B$(5)
140 DATA ELISA SUNFLOWER REA SEW BEE COM COO
150 DATA HANORA RAMELLA REA SWI COM HIK
160 DATA NATHANAEL RAMELLA DAN BEE REA SWI
170 DATA ROGER AYLWORTH REA HIK COM
180 DATA BRUCE AIKIN REA SWI COO
2000 INPUT "WHAT IS TO BE CHECKED (REA,READING)";A$,E$
2010 CLS
2020 PRINT "PEOPLE INTERESTED IN ";E$
2030 PRINT
2040 FOR A = 1 TO 5
2050  READ B$(A)
2060  C$ = LEFT$(B$(A),19)
2070  FOR I = 1 TO LEN(B$(A)) - LEN(A$) + 1
2080   B$ = B$(A)
2090   B$ = MID$(B$,15)
2100   IF A$ = MID$(B$,I, LEN(A$)) PRINT C$
2110   NEXT I
2120  NEXT A
2130 PRINT
2140 PRINT
2150 PRINT "END OF LIST"
2160 INPUT "WANT ANOTHER LIST (YES/NO)";X$
2170 CLS
2180 IF X$ = "YES"
      THEN
       110 :
      ELSE
       PRINT "BYE"
2190 END
```

INTERFACE

Stick With It

Easy Selectric™ Output for the TRS-80:

Take Me to Your Solenoids

Stick With It

by John Warren

The Klingon made a wide, sweeping turn to port, leaving himself wide open for a spread of photon torpedoes. My finger mashed down the A key and. . . nothing happened.

In my excitement, I had hit the down-arrow key by accident. For the 10E6th time, I cursed the absence of a joystick on the Model I. Joysticks don't help much with a general ledger and are useless when I'm keeping track of student grades. But when all that is past, and Luke Skywalker needs my help, a keyboard really doesn't cut it.

Alpha Products of Woodhaven, NY has solved this problem with their Stick-80, a modified Atari joystick. My starship, Enterprise, was still glowing wreckage on the CRT when I called and placed my order for one. The stick itself is relatively simple, a seemingly unmodified Atari joystick with a 40-inch cord. The stick's base is held together by four substantial screws which provide easy access to the contents for cleaning and tinkering.

Alpha Products of Woodhaven, NY has solved this problem with its Stick-80, a modified Atari joystick. My starship, Enterprise, was still glowtor. The arrangement is sturdy enough for normal wear, but I recommend careful handling. The edge connector is intended to mate with the expansion port on the Model I keyboard or on the screen printer port of the interface.

Here I encountered a problem. The plug-in modification didn't plug in. A careful examination revealed that the male edge connector on my expansion interface was about a tenth of an inch wider than the opening on the stick. Not wanting to return the stick and wait for a replacement, I opened the interface and carefully filed down the necessary edges. The connector slid on without further protest.

Stick-80 is a ported device like the cassette recorder. The TRS-80 has 256 ports numbered 0 to 255. The stick is located at port 0. Its presence there has no inherent effect on the computer's running. When the command INP(0) is encountered within a program, however, the computer returns a decimal value between 0 and 255. The stick sends only 10 values which correspond to eight directions of travel, a fire command, and a null for no movement.

Evidently, Alpha Products is planning to introduce a set of paired sticks since the direction sheet contains instructions for the use of single and paired Stick-80s. A simple form of bit masking permits the positions of two sticks to be read with a single INP statement.

Under normal conditions (no stick attached or no movement indicated), port 0 returns a decimal 255 (a binary 11111111) since all bits are normally on. The designers of the Stick-80 have chosen to have the primary stick control the right four bits while the secondary stick changes the left four. For example, if the primary stick was indicating an up, and the secondary stick was indicating a down, the bit pattern would be 11011110, where 1101 is the down signal, and 1110 is the up signal.

You can use the AND function to separate the two values. For example:

```
IN = 255 – INP(0)
S1 = 240 S1 AND IN
S2 = 15 S1 AND IN
```

would give two values (S1 for the secondary stick, and S2 for the primary stick) from the single input. An additional line:

```
F = IN AND 3
```

allows the fire command for the primary stick (value of 3 or 11 binary) to be filtered out from other commands. In this manner, a player can send a move command and a fire command simultaneously.

The four-page manual that comes with the Stick-80 contains complete information on the conversion of existing BASIC programs along with the listing for Magic Artist, a demonstration program which illustrates how to use input from the stick.

Easy Selectric™ Output for the TRS-80: Take Me to Your Solenoids

by Morton Leifer

Computer enthusiasts interested in hard-copy printout capability have a boon in the large number of input/output (I/O) Selectrics™ available on the surplus market. There are several excellent articles on the subject of interfacing specific Selectric models to the TRS-80. Most, however, require considerable hardware, including complex circuitry for handshaking and a PROM for converting ASCII to Selectric correspondence code. In many cases, the I/O Selectric may have incomplete or nonexistent documentation and malfunctioning internal logic circuits, making it difficult to implement a complicated interfacing scheme.

The information presented here can be easily implemented by the average computerist, providing smooth and error free operation of most Selectrics containing solenoids for printing and control. The Selectric will operate close to its maximum speed of 14.9 characters per second. Because its internal logic is bypassed, none of the handshaking reed switches in the Selectric are required for proper operation.

Flexibility is gained by using a software driver not found in an all hardware interface. The software driver accommodates a wide range of standard and nonstandard typing spheres, including one available from IBM called Data 1. It has a full ASCII character set and is perfect for BASIC LLISTs and LPRINTs.

The software is compatible with, and improves upon, the RS232 Electric Pencil patch described in the August 1980 issue of *80 Microcomputing.* A major problem with the Pencil is that it does not stop typing at the end of each page so that a new sheet of typing paper can be inserted. This software waits for the six or more consecutive carriage returns that the Pencil produces at the end of each page, then goes into a wait loop within the driver program. When the paper is changed a C (for continue) is entered on the TRS-80 keyboard. Control is returned to the Pencil which prints the next page.

The hardware, though standard and simple, has functioned reliably for hundreds of hours without errors or failures. Most of the parts used in the circuit are available at Radio Shack for less than $50 dollars. A complete description of the software driver and hardware interface is given so that those who wish to can improve, modify, or adapt them to meet other interfacing needs.

Software

The software consists of a machine-language program in three separate parts. The first part of the program is located between lines 210 and 320 and is the initialization portion of the program. It loads the memory location of the new print driver program (LPRT) into the printer device control block (DCB). The printer DCB is located in reserved RAM from 4025H to 402CH (16421 to 16428 decimal) and initially contains the address of the printer driver program located in the TRS-80 ROM used by Level II BASIC for LPRINTs and LLISTs. After this part of the program is executed, the DCB will contain the address of our new print driver, which will operate the Selectric and respond to all LPRINTs and LLISTs in place of the original ROM print driver.

The decimal value loaded into register A (line 240) determines the number of characters printed on each line before an automatic carriage return occurs. A value of 72 prevents typing beyond the width of standard typing paper. If your machine has an extra-wide carriage of fifteen inches, the decimal value can be increased to 135. As your needs change, you can alter the decimal value directly from BASIC by POKEing the desired decimal value into memory location 16426. Lines 260 and 270 load the value of six into DCB location 4028H. This value is the number of consecutive spaces printed by the Electric Pencil before the drive program enters a wait loop. POKEing any desired new value into decimal 16424 changes this value.

The computer controls any of a number of possible programs in lines 280-310. Implement only one of these four lines by removing the semicolon preceding the code which contains the desired control address. In most Level II machines, line 280 will take you to a READY. Line 290 will also jump control to a warm start of Level II BASIC. Your disk operating system is in line 300. If you want to concatenate this program to the Electric Pencil, and go directly to the Pencil after initializing the printer DCB, use line 310.

The second part of the program converts the ASCII character you wish to print to the proper IBM code (see Table 1). This code is sent to the parallel port which connects the printing and control solenoids of the Selectric to the data lines of the TRS-80. Correct timing intervals for printing and control characters, allowing the Selectric to be operated near its maximum speed, are inserted at this point. The program also tests the character to be printed to determine if upper- or lowercase shifting is necessary and outputs the appropriate shifting codes. This program is the equivalent of the CAPLOCK operation, which is most desirable for BASIC LPRINTs and LLISTs.

Regardless of the shift position, the output will be in uppercase letters. Numbers or punctuation will be output according to the shift position. When the simple Electric Pencil lowercase modification is added to the TRS-80, it will produce upper- and lowercase output on both the video and the Selectric. If you do not wish to modify your keyboard for lowercase

CHAR-ACTER	ASCII VALUE	IBM VALUE	CONTROL	SHIFT	T2	T1	R5	R2A	R2	R1
!	33	127	0	1	1	1	1	1	1	1
“	34	85	0	1	0	1	0	1	0	1
#	35	126	0	1	1	1	1	1	1	0
$	36	121	0	1	1	1	1	0	0	1
%	37	117	0	1	1	1	0	1	0	1
&	38	125	0	1	1	1	1	1	0	1
'	39	21	0	0	0	1	0	1	0	1
(	40	112	0	1	1	1	0	0	0	0
)	41	113	0	1	1	1	0	0	0	1
*	42	124	0	1	1	1	1	1	0	0
+	43	70	0	1	0	0	0	1	1	0
‘	44	12	0	0	0	0	1	1	0	0
–	45	0	0	0	0	0	0	0	0	0
.	46	22	0	0	0	1	0	1	1	0
/	47	9	0	0	0	0	1	0	0	1
0	48	49	0	0	1	1	0	0	0	1
1	49	63	0	0	1	1	1	1	1	1
2	50	54	0	0	1	1	0	1	1	0
3	51	62	0	0	1	1	1	1	1	0
4	52	57	0	0	1	1	1	0	0	1
5	53	53	0	0	1	1	0	1	0	1
6	54	52	0	0	1	1	0	1	0	0
7	55	61	0	0	1	1	1	1	0	1
8	56	60	0	0	1	1	1	1	0	0
9	57	48	0	0	1	1	0	0	0	0
:	58	77	0	1	0	0	1	1	0	1
;	59	13	0	0	0	0	1	1	0	1
l	60	41	0	0	1	0	1	0	0	1
=	61	6	0	0	0	0	0	1	1	0
g	62	15	0	0	0	0	1	1	1	1
?	63	73	0	1	0	0	1	0	0	1
@	64	118	0	1	1	1	0	1	1	0
A	65	92	0	1	0	1	1	1	0	0
B	66	96	0	1	1	0	0	0	0	0
C	67	108	0	1	1	0	1	1	0	0
D	68	109	0	1	1	0	1	1	0	1
E	69	101	0	1	1	0	0	1	0	1
F	70	78	0	1	0	0	1	1	1	0
G	71	79	0	1	0	0	1	1	1	1
H	72	97	0	1	1	0	0	0	0	1
I	73	84	0	1	0	1	0	1	0	0
J	74	71	0	1	0	0	0	1	1	1
K	75	100	0	1	1	0	0	1	0	0
L	76	105	0	1	1	0	1	0	0	1
M	77	95	0	1	0	1	1	1	1	1
N	78	102	0	1	1	0	0	1	1	0
0	79	89	0	1	0	1	1	0	0	1
P	80	69	0	1	0	0	0	1	0	1
Q	81	68	0	1	0	0	0	1	0	1

Table continued

R	82	93	0	1	0	1	1	1	0	1
S	83	81	0	1	0	1	1	1	0	1
T	84	103	0	1	1	0	0	1	1	1
U	85	110	0	1	1	0	1	1	1	0
V	86	94	0	1	0	1	1	1	1	0
W	87	80	0	1	0	1	0	0	0	0
X	88	111	0	1	1	0	1	1	1	1
Y	89	65	0	1	0	0	0	0	0	1
Z	90	119	0	1	1	1	0	1	1	1
	91	101	0	1	1	0	0	1	0	1
	92	128	0	1	0	0	0	0	0	0
	93	64	0	1	0	0	0	0	0	0
a	97	28	0	0	0	1	1	1	0	0
b	98	32	0	0	1	0	0	0	0	0
c	99	44	0	0	1	0	1	1	0	0
d	100	45	0	0	1	0	1	1	0	1
e	101	37	0	0	1	0	0	1	0	1
f	102	14	0	0	0	0	1	1	1	0
g	103	15	0	0	0	0	1	1	1	1
h	104	33	0	0	1	0	0	0	0	1
i	105	20	0	0	0	1	0	1	0	0
j	106	7	0	0	0	0	0	1	1	1
k	107	36	0	0	1	0	0	1	0	0
l	108	41	0	0	1	0	1	0	0	1
m	109	31	0	0	0	1	1	1	1	1
n	110	38	0	0	1	0	0	1	1	0
o	111	25	0	0	0	1	1	0	0	1
p	112	5	0	0	0	0	0	1	0	1
q	113	4	0	0	0	0	0	1	0	0
r	114	29	0	0	0	1	1	1	0	1
s	115	17	0	0	0	1	0	0	0	1
t	116	39	0	0	1	0	0	1	1	1
u	117	46	0	0	1	0	1	1	1	0
v	118	30	0	0	0	1	1	1	1	0
w	119	16	0	0	0	1	0	0	0	0
x	120	47	0	0	1	0	1	1	1	1
y	121	1	0	0	0	0	0	0	0	1
z	122	55	0	0	1	1	0	1	1	1
SPACE	32	129	1	0	0	0	0	0	0	1
C.R.	13	130	1	0	0	0	0	0	1	0
INDEX	10	132	1	0	0	0	0	1	0	0

Table 1

video, it is still possible to obtain standard upper- and lowercase output by removing the semicolons in lines 900 to 970. Direct typewriter operation of the TRS-80 will occur. Lowercase characters will be output to the Selectric when the TRS-80 is unshifted, and uppercase characters will be output when the keyboard is shifted.

The second part of the program starts at line 330. The calling routine's stack pointer and all CPU registers are saved in lines 330 to 370, so that after the desired character is output they can be restored, allowing the calling program to continue its operation. Lines 390 to 410 check to see if there is a character in register C to be printed. If so, the character is compared to the value 33 decimal, the smallest ASCII code that produces a printable character. An ASCII value of 33 decimal or greater is sent to ALPHA1 for further processing before it is finally printed. ASCII values smaller than 33 decimal are non-printing control characters and are tested in line 450 for a line feed, in line 470 for a space character, and in line 480 for a carriage return. The character is sent to a routine that outputs an IBM code that corresponds to the ASCII value control function.

The carriage return routine includes a timing loop called DELAY2. This permits the printing sphere to return and begin the next line. Printable ASCII characters are in continuous numeric order starting from 33 decimal up to 176 decimal. They can be converted into the Selectric code by using a simple lookup table. The lookup table is the third part of the program and starts at line 1480.

Once a printable ASCII character has been sent to ALPHA1 (line 990), decimal 33 is subtracted from it and the result is left in register A. The first printable character having the decimal value 33 and an exclamation point produces a zero value in the A register when subtraction is accomplished. In line 1010 the memory pointer (HL register) is adjusted so that it points to the beginning of the lookup table. The contents of register A are then transferred to register L in lines 1020 and 1030, making the memory pointer (HL) point to the beginning of the lookup table, including the A register. The memory pointed to consists of decimal 127, the Selectric code for an exclamation point. Conversion from ASCII to Selectric code has been made for the exclamation point.

If the ASCII character sent to ALPHA1 is decimal 65 (uppercase A), the result left in the A register after subtraction is 32. When this is added to register L it causes the HL register to point to the start of the lookup table plus 32. The start of the lookup table is located at memory 7F90 hex. Decimal 32 is equivalent to 20 hex. Adding 20 hex to 7F90 hex is 7FB0 hex. The contents of memory location 7FB0 hex is decimal 92, which is the Selectric code for uppercase A. The conversion cycle is complete.

Placing the Selectric code for a character into a memory location located above the start of the lookup table by an amount equal to 32 less than the numerical value of the ASCII of the character ensures a true conversion of that character. Other characters similarly placed will also be converted.

The benefits of using a lookup table for code conversion are many. The same simple routine, ALPHA1, is used to convert the entire printing character set which conserves memory space. If each character had been individually processed with its own routine, much more memory would have been needed. Nonstandard spheres can be accommodated by changing only the IBM Selectric numeric values in the lookup table to those appropriate for the sphere being used. Typing spheres such as the IBM Data 1 having the complete ASCII character set are very desirable for computer work. There is a host of special-purpose typing spheres available that can be perfectly tailored to your computer keyboard by altering the lookup table.

After the ASCII character has been converted into Selectric code, it is tested to see if it is an uppercase character. Any converted character with a value greater than decimal 64 requires a shift to uppercase. This test is accomplished in line 1050, and if a shift to uppercase is needed, the program jumps to the TEST1 routine at line 1090. This determines whether the Selectric is already in the shifted position. Shift status is sent from the interface on bit 6 of the data bus. Bit 6 is the seventh bit in an eight-bit word (D0-D7). If the shift solenoid is being energized, which causes the typing sphere to rotate 180 degrees into its uppercase position, bit 6 will be low. All of the other bits will be high, and the value input from port 13 will be 63 (X0111111). On a typing sphere that isn't rotated into its uppercase position, bit 6 will be high and the value input from port 13 will be 127 (X1111111). An uppercase character to be typed without a shift is sent to the CHOUT routine for further processing. If an uppercase character is printed, and a shift is needed, line 1140 loads decimal 64 (01000000) into the A register and outputs that value to port 13. The high sixth bit's (remember we start counting from zero) output to port 13 causes the shift solenoid to energize and rotate the typing sphere 180 degrees.

The character to be printed, which has been saved on the stack (line 1090 or 1170), is now sent to the CHOUT routine which starts at line 1230. CHOUT takes the character off the stack and masks the two most significant bits of the eight-bit number. They are not used in the Selectric code and don't need to be output to the machine. The character is again stored in the stack (line 1250) while the number of characters typed on the line up to that point is calculated and a decision made on whether or not a carriage return is necessary.

Carriage return arbitration is accomplished as follows. Back in the intialization section of the program (lines 240 and 250), you set the character count to the number of characters, including spaces, you wanted printed on each line. This decimal value was stored in the printer DCB at location 402A hex. In line 380 the IX register was initialized at 4025 hex, which is the beginning of the printer DCB. In line 670, IX + 5, five memory locations higher than where the DCB began, points to a memory location which is five more than 4025. IX + 5 is pointing to 402A where the number of characters per line is stored. That value is put into the A register, and in line 680 it is transferred to a memory location called character count (CHCNT).

This is all happening in a CRLF1 routine, which immediately prints a carriage return in line 690. This resets the memory location (CHCNT) with the number of characters you want printed on a line prior to sending a carriage return. During CHOUT, just prior to outputting the character to be printed, save the character in line 1250, then point the HL register to the memory location (CHCNT) where the number of characters per line value is stored. In line 1270 decrement that value by one, because you're going to print a character. In line 1280 test to see if the value in CHCNT has reached zero. After 72 characters are printed, the value in CHCNT will be decremented by one 72 times. The value of CHCNT will be zero and line 1290 will call for a carriage return.

The CRLF routine resets the value in CHCNT to the value specified in the DCB, as indicated above, and sends out a carriage return. In line 590, IX + 4 is a carriage return counter and is incremented each time a carriage return is sent. If six consecutive carriage returns are sent without an intervening printed character, the value of IX + 4 will equal the value put into IX + 3 during the initialization of the program (line 260). If there is a true compare, that is if IX + 4 equals six, then the program will go to the BREAK routine. This is a wait loop that will continue until the letter C is depressed. This corrects the problem that occurs when using the Electric Pencil. There is a pause in printing while typing paper is changed between pages.

In line 1300 control is sent to line 760 where the necessary extra delay (DELAY2) is inserted to permit the type sphere to return to the beginning of the next line. The character is popped from the stack and printed. The print routine starts at line 1340 by outputting the character to be printed to parallel port 13. DELAY inserts the correct time delay to provide enough time for the mechanics of the Selectric to settle down before the next character is printed. The timing byte in line 1360 sets the exact delay between characters which determines the characters per second. A value of 09FF hex produces close to 15 characters per second. An 08FF produces a speed which may slightly exceed the 14.9 characters per second the machine is designed for, and an 0AFF hex provides a very respectable speed, but slower than the maximum permissible.

The desired value can be POKEd into the correct memory location directly from BASIC using decimal values. A check to see if the BREAK key is pressed is carried out in lines 1390 to 1410. Line 1390 checks the contents of memory location 3840 hex whose eight bits represent eight individual keys on the TRS-80 keyboard. If the value of that memory location is four, then the BREAK key has been pressed. Control is immediately sent back to the calling program, and no additional characters will be printed until another LLIST or LPRINT is initiated. This feature allows for instant stopping of the printing process in the middle of an LLIST or LPRINT. After a character is printed, the program jumps to a GOBACK routine, which restores the calling routine's stack pointer and pops all registers, permitting the calling program to continue.

Hardware

The hardware interface is a standard eight-bit parallel port configuration with a small amount of additional circuitry. It uses a pair of 74LS85 magnitude comparators, and an eight-bit DIP switch to accomplish port addressing. A pair of 74LS175 four-bit latches holds the required eight bits of data for further processing. There are a minimum of 11 solenoids in the Selectric that must be independently actuated. Most of the additional interface circuitry provides the necessary timing and multiplexing for their proper operation. A 74LS367 provides a capability of up to six status signals from the interface back into the computer, but only shift status is required for proper operation. The circuit operates as follows.

The 74LS85 four-bit magnitude comparators (IC2 and IC3) compare the voltage present on B0-B7 with that present on the address bus A0-A7. By adjusting the DIP switches, as shown on the circuit diagram (see Figure 1), a binary bit pattern of 00001101 is created from B7-B0 corresponding to a value of 13 decimal. A voltage of +5 volts is represented by a one (switch open) and a voltage of zero is represented by a zero (switch closed). When data is output to port 13, the bit pattern on the address bus from A7-A0 will be 13 decimal (00001101 binary). This causes a one-to-one correspondence of voltage between A7-A0 and B7-B0; this is a true compare, and IC3 will respond by outputting a high pulse on pin 6 as long as the true compare remains.

The port address can be changed to any value from 0 to 255 by adjusting the DIP switches to the desired bit pattern. The software output must contain the same port number as that represented by the position of the DIP switches in order for a true compare to occur. The high pulse output from pin 6 of IC3 is inverted by IC1A, and becomes a low on one input of IC1B and IC1C. At the same time as the port number appears on the address bus, a NOT OUT pulse appears on the second input of IC1C causing a high pulse on the output of IC1C. This results in a high pulse on pin 9 of IC4 and IC5, causing the data appearing on the inputs of IC4 and IC5 to be latched on to their outputs.

Only six bits of latched data (D0-D5) are necessary to produce alphanumeric characters or control functions on the Selectric. They are applied to one input of their respective AND gates IC10, IC11, and IC12. Gates IC10, IC11A, and IC11B carry data to the transistors that drive the print solenoids R1, R2, R2a, R5, T1, and T2. Different combinations of data-bit patterns applied to the print solenoids through the drive transistors connected to these gates cause closure of various combinations of print solenoids. The energized solenoids become mechanically latched, determining the particular alphanumeric character to be printed (Program Listing 1). These

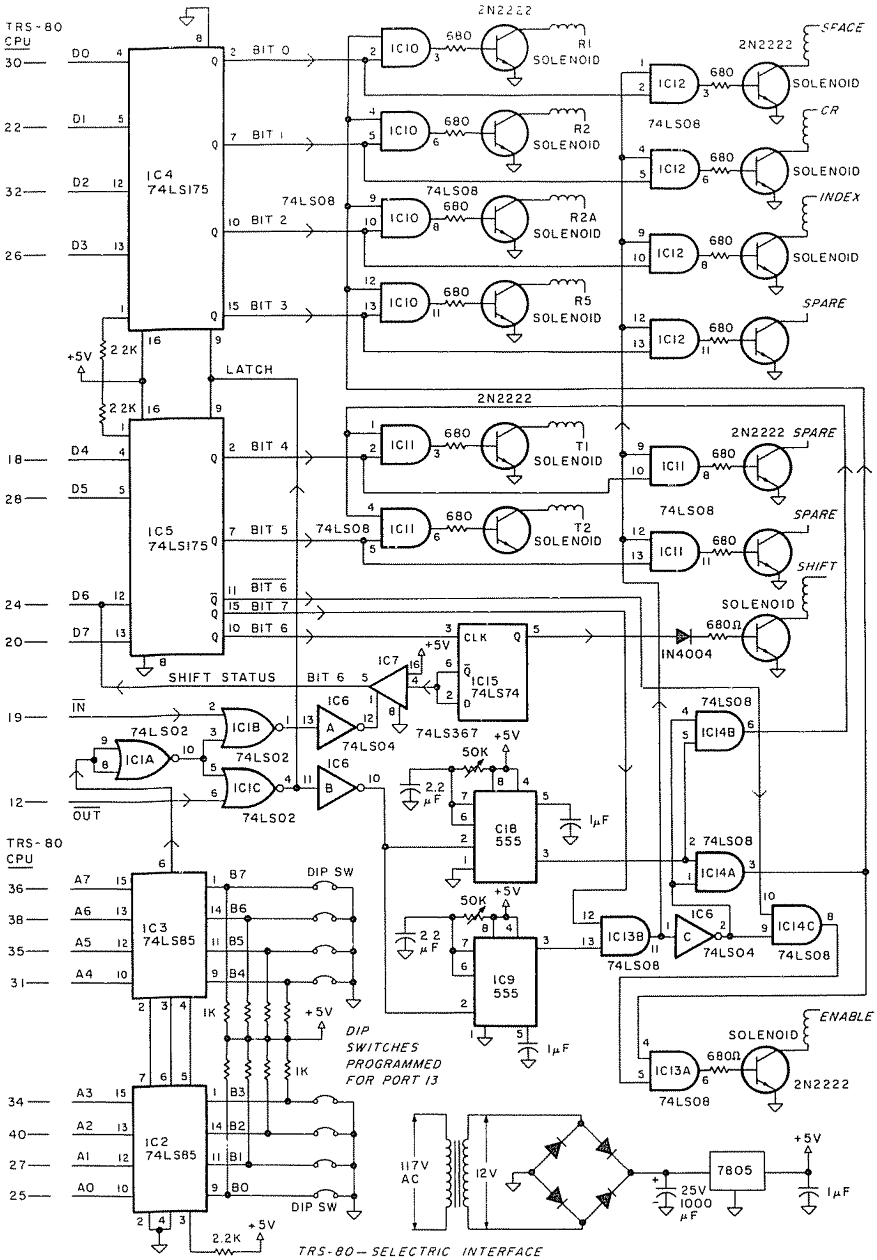


Figure 1. *TRS-80—Selectric interface. Owners of the original TRS-80 (Model I) might want to install a current-limiting device.*

mechanically latched solenoids are released only after a separate enable solenoid is energized, which prints the character and resets the solenoids for receipt of the next bit pattern. The enable solenoid must be turned on for a definite length of time once the print solenoids have been latched. If the enable solenoid is energized for too short a time, the character will not print; too long a time will cause the character to repeat. Two 55 timers, IC8 and IC9, provide a suitably stable and adjustable time interval to accommodate the needs of any Selectric.

Gates IC12, IC11B, and IC11D carry data to transistors driving the control solenoids. These solenoids produce space, carriage return, and indexing as well as spare functions for implementing additional features. None of the above gates will pass data until their second input goes high. This depends upon whether a character is to be printed or whether a control function is required. Only one set of gates (print or control) will be turned on at any given time. There are only a very few gates involved in this timing and arbitration process. The operation of this circuit can easily be understood by following through with the printing of a lowercase letter, an uppercase letter, and a control function.

Suppose a lowercase *a* is to be printed. Program Listing 1 shows that the Selectric code for *a* is decimal 28. The bit pattern on the data bus will therefore be 00011100, the binary equivalent of 28. As the data is output on the data bus, the address bus is given a value of 13 corresponding to the bit pattern set on B0-B7 by the DIP switches. This true compare causes a high output on pin 6 which is inverted by IC1A and applied to one input of IC1B and IC1C. A NOT OUT which also appears during the outputting of data causes the output of IC1C to go high, since there are now two low pulses on the two inputs of IC1C. This high pulse is applied to pin 9 of the latches IC4 and IC5 causing the data on the data bus to be latched onto the Q outputs. The bit pattern corresponding to *a* is applied to the print gate IC10 and the control gate IC12.

The data does not pass through the gates until the second input of the gates is made high. In this case, only the print gate, IC10, will be made to pass data. This is accomplished as follows. The high output on IC1C is inverted by IC6B and sent to both 555 timers, IC8 and IC9. The low on pin 2 of the timers triggers them, causing a high pulse of adjustable length to appear on their outputs (pin 3). The high output of timer IC8 is applied to one input of IC14A and IC14B, and the high output of timer IC9 is applied to one input of IC13B. The second input of IC13B is connected to bit 7 of the latched data. Referring back to the bit pattern for the letter *a*, we see that bit 7 is a low. IC13B will therefore have a low on its output and will inhibit all the control and extra function gates from passing data to the control solenoids. IC6C inverts that low and applies a high to the second input of IC14A and IC14B. The outputs of these gates are now high, allowing IC10, IC11A, and IC11B to pass data to the transistor switches energizing the appropriate print solenoids.

In the case of printing a small *a*, the R2A, R5, and T1 (see Figure 1) solenoids are energized and mechanically latch. Note that the inputs of IC14A and IC14B are wired in parallel so that the outputs of both gates always vary in the same way. This was done to reduce the fan outload on these gates. The character has not been printed yet, because we haven't energized the enable solenoid. The high output on IC6C is also applied to one input of IC14C. Because it is connected to the negative of bit six of the data bus, the second input of IC14C is also high. Pin 11 of IC5 is the NOT Q of bit six, and bit six of our data is low (look at Program Listing 1 for the bit pattern for an *a*). The output of IC14C and of IC14A are both high and are applied to the inputs of IC13A. The high on the output of IC13A turns the 2N222 switching transistor on, which energizes the Selectric enable solenoid and prints the character *a*.

The print solenoids are mechanically unlatched during this process and are ready to be energized into a new bit pattern corresponding to the next letter to be printed. Note that just prior to outputting the character, the shift status on bit six of the data bus is examined to determine whether the print ball is in the upper- or lowercase position. Port 13 is addressed again, and a high pulse is output from pin 6 of IC3, which is inverted to a low and input to IC1B. A NOT IN signal is applied to the second input of IC1B causing a high at its output. This pulse is inverted in IC6A to a low which is applied to the tri-state buffer and causes the signal condition on its input to be read into bit six of the data bus and interpreted by software into shift status.

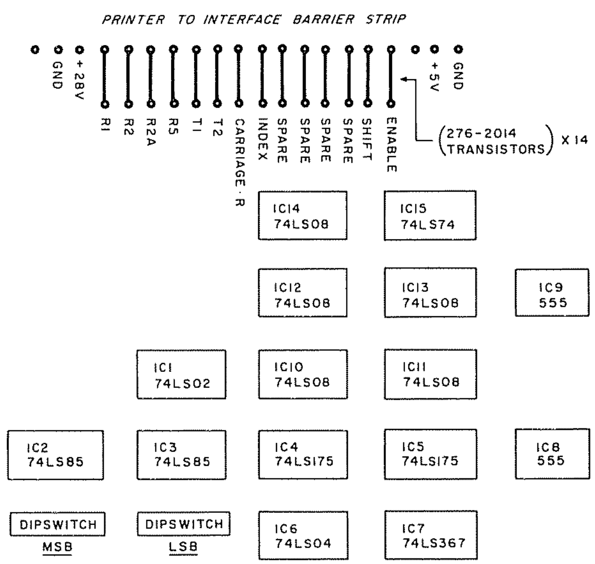


Figure 2. *Parts layout*

IC15, a 74LS74 flip-flop, alternately energizes and de-energizes the shift solenoid as each positive leading edge is clocked in at pin 3. If the Q output of IC15 is high, the shift solenoid is energized, and the NOT Q output sends a low back to the computer which is interpreted by software as an uppercase, shifted condition. If shift status indicating an uppercase condition exists when a lowercase *a* is to be sent, the software will first output a high on bit 6 of the data bus (decimal 64 or 01000000 binary). This causes the flip-flop to flip and output a low to the shift solenoid creating a lowercase condition appropriate for the letter to be sent.

A high on bit 6 will produce a low on NOT 6 which keeps the enable solenoid from being energized when it is applied to one input of IC14C. This is necessary because outputting a shift command requires sending a decimal 64 (01000000 in binary). The five lowest significant bits represent the character "-" (see Program Listing 1) which would be printed if the enable solenoid was allowed to be energized during the shifting process. To send an uppercase B, for instance, the software would look at shift status on data bit six and output a decimal 64 if shift status had to be changed. Only the shift solenoid would be energized because the low on NOT BIT 6, which is always present while shifting, would keep the enable solenoid from being energized. After a short delay, the bit pattern (00100000) corresponding to B would be output and passed through the print gates and to energize the appropriate print solenoids. The enable solenoid would, of course, be energized just as it was while printing the letter *a*.

Notice that the sixth bit of every character (see Program Listing 1) is used only to tell the software whether an upper- or lowercase character is being sent. The software checks the condition of the shift solenoid and adjusts it, if necessary, to conform to what is required by the character being sent. The interface, however, only responds to the lowest five significant bits of any character. This means that an *a* and an *A* produce the same print solenoid pattern, even though the print ball is shifted to produce the correct case.

To send a control character, such as a space or carriage return, only the control gate IC12 should be turned on. When any control function is sent, bit seven (the most significant bit) is always high. This makes the second input of IC13B high, causing the output of IC13B to be high. Under these conditions, the control gates IC12 and IC11C and IC11D pass data. The high on the output of IC13B is inverted by IC6C and applies a low on the second inputs of IC14A and IC14B. The resulting low on their outputs inhibits data from passing through IC10, IC11A, and IC11B, and no printing solenoids are energized. The print enable solenoid isn't energized either.

Conclusion

The parts layout suggested provides relatively short and direct signal paths. However, construction of the interface can be accomplished in any of a number of ways. The parts layout or lead length is not critical. Several interfaces were constructed by different people, and all worked well the very first time.

The parts are all readily available at Radio Shack with the exception of the 74LS85 which is a standard chip available at most electronics and computer stores. I also found a very nice 40-pin header assembly (#1634-NI) that fits well on a perforated circuit board and a matching female cable receptacle (#1654-NI) from the catalog of INMAC Corporation of Norwood, NJ.

74LS02	IC1	276-1902
74LS85	IC2, IC3, not available at Radio Shack	
74LS175	IC4, IC5	276-1934
74LS04	IC6	276-1904
74LS367	IC7	276-1835
555	IC8, IC9	276-1723
74LS08	IC10, IC11, IC12, IC13, IC14	276-1908
74LS74	IC15	276-1919
2N2222	Switch transistors	276-2014
Transformer	Power supply	273-1385
Rectifier	Bridge type	276-1146
Regulator	5-volt regulator	276-1770
DIP switch	To program port address	274-1301
50K potentiometers	Adjust timer pulse width	271-219
1N4004 diode	Needed only in shift circuit	276-1103
Barrier strips	Connect Selectric to interface	274-678

The IBM Selectric drive program is also presented in BASIC for those who prefer to work with BASIC rather than with machine language (see Program Listing 2). The program works just as well as the machine-language version, and can be removed by typing NEW after it is run. All of the driver code is safely stored above 32408 in protected memory.

Bibliography

Bickerton, Michael, M.D. "Selectric Hard Copy." *80 Microcomputing*, September, 1980.

Morr, David. "Teleprinter Output for TRS-80." *Kilobaud Microcomputing*, August, 1979.

Program Listing 1. *Machine-language version*

Encyclopedia Loader

```
               00070
               00080 ;IBM PRINT DRIVER PROGRAM                    8/6/81
               00090
               00100 ;           M LEIFER
               00102 ;           DEPARTMENT OF ELECTRICAL TECHNOLOGY
               00104 ;           ROCKLAND COMMUNITY COLLEGE
               00106 ;           145 COLLEGE RD SUFFERN N.Y.10901
               00108 ;
               00110
               00120 ;           REQUIREMENTS            LEVEL 2 BASIC
               00130 ;                                   16 K RAM
               00140 ;                                   PARALLEL INTERFACE
               00150 ;                                   TO SELECTRIC SOLENOIDS
               00160 ;
               00170 ;           TO IMPLEMENT  RESERVE MEM 32408
               00180 ;           LOAD PROGRAM...../32410
               00190
               00200
7E9A           00210         ORG     7E9AH           ;ORIGIN OF INIT PROGRAM
7E9A 21AB7E    00220 LOAD    LD      HL,LPRT         ;INITIALIZE DCB
7E9D 222640    00230         LD      (4026H),HL
7EA0 3E48      00240         LD      A,72D           ;SET CHAR COUNT TO 72
7EA2 322A40    00250         LD      (402AH),A
7EA5 3E06      00260         LD      A,6D            ;6 C.RETURNS IN A ROW
7EA7 322840    00270         LD      (4028H),A       ;STOPS PRINTING (IX+3)
7EAA 76        00280         HALT                    ;GO TO BASIC
               00290 ;       JP      1A19H           ;GO TO BASIC
               00300 ;       JP      402DH           ;GO TO DOS
               00310 ;       JP      17232           ;GO TO ELECTRIC PENCIL
               00320
7EAB ED73EA7F  00330 LPRT    LD      (SAVSTK),SP     ;SAVE CALLING ROUTINES S.P.
7EAF F5        00340         PUSH    AF              ;SAVE ALL
7EB0 C5        00350         PUSH    BC              ;REGISTERS
7EB1 D5        00360         PUSH    DE              ;
7EB2 E5        00370         PUSH    HL              ;
7EB3 DD212540  00380         LD      IX,4025H        ;POINT IX TO START OF DCB
7EB7 79        00390         LD      A,C             ;IS THERE A
7EB8 B7        00400         OR      A               ;CHARACTER TO BE PRINTED
7EB9 285A      00410         JR      Z,GOBACK        ;IF NOT BACK TO CALLING PROG
7EBB FE21      00420         CP      33D             ;IS CHARACTER PRINT OR CONTROL
7EBD 305F      00430         JR      NC,ALPHA1       ;GO TO CAPLOC PRINT ROUTINE
               00440 ;       JR      NC,ALPHA0       ;GO TO TYPEWRITER PRINT ROUTINE
7EBF FE0A      00450         CP      10D             ;CHECK FOR LINE FEED
7EC1 2810      00460         JR      Z,LNFD          ;IF YES OUTPUT A LINE FEED
7EC3 FE20      00470         CP      32D             ;CHECK FOR SPACE CHARACTER
7EC5 2002      00480         JR      NZ,CHEKAR       ;CHECK FOR CARRIAGE RETURN
7EC7 180F      00490         JR      SPCHR           ;OUTPUT A SPACE
7EC9 FE0D      00500 CHEKAR  CP      13D             ;CHECK AGN FOR CARRIAGE RETURN
7ECB CCDD7E    00510         CALL    Z,CRLF          ;OUTPUT A CARRIAGE RETURN
7ECE CD0A7F    00520         CALL    DELAY2          ;SPECIAL DELAY FOR CARRIAGE RETURN
7ED1 1842      00530         JR      GOBACK          ;BACK TO CALLING PROGRAM
7ED3 CD027F    00540 LNFD    CALL    LF              ;OUTPUT A LINE FEED
7ED6 183D      00550         JR      GOBACK          ;BACK TO CALLING PROGRAM
7ED8 3E81      00560 SPCHR   LD      A,129           ;OUTPUT A SPACE CODE
7EDA F5        00570         PUSH    AF              ;SAVE CHARACTER TO BE PRINTED
7EDB 186E      00580         JR      CONT            ;ADJUST CHAR PER LINE COUNTER
7EDD DD3404    00590 CRLF    INC     (IX+4)          ;INCREMENT C.R. COUNTER
7EE0 DD7E04    00600         LD      A,(IX+4)        ;PUT VALUE IN A
7EE3 DDBE03    00610         CP      (IX+3)          ;COMPARE WITH VALUE IN DCB
7EE6 200B      00620         JR      NZ,CRLF1        ;IF NOT = DO A CARRIAGE RETURN
7EE8 3A0138    00630 BREAK   LD      A,(3801H)       ;LOOK AT KEYBOARD
7EEB FE08      00640         CP      8D              ;HAS C BEEN PRESSED
7EED 20F9      00650         JR      NZ,BREAK        ;WAIT UNTIL IT HAS
7EEF DD360400  00660         LD      (IX+4),0        ;OTHERWISE RESET LINE COUNTER
7EF3 DD7E05    00670 CRLF1   LD      A,(IX+5)        ;RESET CHAR PER LINE COUNTFR
7EF6 32EC7F    00680         LD      (CHCNT),A       ;TO INITIAL DCB VALUE
7EF9 3E82      00690         LD      A,130           ;LOAD A CARRIAGE RETURN INTO A
7EFB CD5D7F    00700         CALL    PRINT           ;PRINT IT
7EFE CD5F7F    00710         CALL    DELAY           ;
7F01 C9        00720         RET                     ;
7F02 3E84      00730 LF      LD      A,132           ;OUTPUT A LINE FEED
7F04 CD5D7F    00740         CALL    PRINT           ;PRINT IT
7F07 CD5F7F    00750         CALL    DELAY           ;ONE CHARACTER DELAY NEEDED
7F0A F5        00760 DELAY2  PUSH    AF              ;NOW LONG DELAY FOR CARRIAGE RETURN
```

```
7F0B 01FF8F   00770          LD     BC,8FFFH       ;TIMING BYTE FOR LONG CR DELAY
7F0E 0B       00780 WAIT1    DEC    BC             ;STILL WAITING
7F0F 78       00790          LD     A,B            ;FOR
7F10 B1       00800          OR     C              ;CARRIAGE
7F11 20FB     00810          JR     NZ,WAIT1       ;TO RETURN
7F13 F1       00820          POP    AF             ;DONE WAITING  RESTORE A REGISTER
7F14 C9       00830                 RET            ;BACK TO CALLING ROUTINE
7F15 E1       00840 GOBACK   POP    HL             ;RESTORE ALL
7F16 D1       00850          POP    DE             ;REGISTERS
7F17 C1       00860          POP    BC             ;
7F18 F1       00870          POP    AF             ;
7F19 ED7BEA7F 00880          LD     SP,(SAVSTK)    ;RESTORE CALLING PROGRAMS STACK
7F1D C9       00890          RET                   ;OUT OF DRIVER  TO CALLING PROGRAM
              00900 ;ALPHAO  CP     97D            ;IS CHARACTER UNSHIFTED
              00910 ;        JR     NC,MINUS       ;MAKE LOWER CASE OUT OF UPPER CASE
              00920 ;        CP     65D            ;IS CHARACTER SHIFTED
              00930 ;        JR     NC,PLUS        ;MAKE UPPER CASE OUT OF LOWER CASE
              00940 ;        JR     ALPHA1         ;PRINT CHARACTER AS IS
              00950 ;MINUS   SUB    20H            ;
              00960 ;        JR     ALPHA1         ;
              00970 ;PLUS    ADD    A,20           ;
              00980
7F1E D621     00990 ALPHA1   SUB    33D            ;ADJUST ASCII CHAR. TO START OF TABLE
7F20 DD360400 01000          LD     (IX+4),0       ;RESET CR COUNTER
7F24 21907F   01010          LD     HL,TABLE       ;GET HL TO POINT TO
7F27 85       01020          ADD    A,L            ;START OF TABLE PLUS
7F28 6F       01030          LD     L,A            ;33 MINUS ASCII VALUE
7F29 7E       01040          LD     A,(HL)         ;GET SELECTRIC CHARACTER
7F2A FE40     01050          CP     64D            ;TEST IF SHIFT IS NECESSARY
7F2C 3002     01060          JR     NC,TEST1       ;SEE IF SELECTRIC IS ALREADY SHIFTED
7F2E 180E     01070          JR     TEST2          ;NO SHIFT NECESSARY SEE IF
              01080                                ;SELECTRIC IS SHIFTED
7F30 F5       01090 TEST1    PUSH   AF             ;SAVE CHARACTER IN STACK
7F31 DB0D     01100          IN     A,(13)         ;INPUT SHIFT STATUS FROM PORT 13
7F33 FE3F     01110          CP     63D            ;(X0111111) IF SEVENTH BITLOW
              01120                                ;SELECTRIC ALREADY SHIFTED
7F35 2810     01130          JR     Z,CHOUT        ;SHIFT NOT NEEDED  OUTPUT CHAR.
7F37 3E40     01140 SHIFT    LD     A,64D          ;SEND HIGH SEVENTH BIT TO
7F39 CD5D7F   01150          CALL   PRINT          ;SHIFT SELECTRIC
7F3C 1809     01160          JR     CHOUT          ;NOW OUTPUT CHARACTER
7F3E F5       01170 TEST2    PUSH   AF             ;TEST TO SEE IF SHIFTED
7F3F DB0D     01180          IN     A,(13)         ;ALREADY
7F41 FE7F     01190          CP     127D           ;(X1111111) SEVENTH BIT HIGH
              01200                                ;MEANS SELECTRIC NOT SHIFTED
7F43 2802     01210          JR     Z,CHOUT        ;OUTPUT CHAR AS IS
7F45 18F0     01220          JR     SHIFT          ;TRIGGER FLIP FLOP AND OUTPUT CHAR.
7F47 F1       01230 CHOUT    POP    AF             ;GET CHARACTER TO BE PRINTED
7F48 E63F     01240          AND    3FH            ;MASK OUT 7,8 BIT  NOT NEEDED
7F4A F5       01250          PUSH   AF             ;SAVE CHAR BACK ON STACK
7F4B 21EC7F   01260 CONT     LD     HL,CHCNT       ;POINT AT CHAR COUNTER
7F4E 35       01270          DEC    (HL)           ;DEC SAME
7F4F 2006     01280          JR     NZ,CONT1       ;IF ZERO DO A CARRIAGE RETURN
7F51 CDDD7E   01290          CALL   CRLF           ;AND LINE FEED
7F54 CD0A7F   01300          CALL   DELAY2         ;LONG DELAY AFTER CR
7F57 F1       01310 CONT1    POP    AF             ;PRINT CHARACTER
7F5B 18B8     01330          JR     GOBACK         ;BACK TO CALLING PROGRAM
7F5D D30D     01340 PRINT    OUT    (13),A         ;OUTPUT CHAR TO PORT 13
7F5F F5       01350 DELAY    PUSH   AF             ;SAVE A REGISTER ON STACK
7F60 01FF09   01360          LD     BC,09FFH       ;TIMING BYTE FOR 14 CHAR. PER SECOND
7F63 0B       01370 DELAY1   DEC    BC             ;DELAY FOR MECHANICS
              01380                                ;OF SELECTRIC TO SETTLE
7F64 3A4038   01390          LD     A,(3840H)      ;LOOK AT KEYBOARD FOR
7F67 FE04     01400          CP     4              ;BREAK KEY DEPRESS
7F69 28AA     01410          JR     Z,GOBACK       ;STOP PRINTING IF BREAK KEY PRESSED
7F6B 78       01420          LD     A,B            ;
7F6C B1       01430          OR     C              ;STILL
7F6D 20F4     01440          JR     NZ,DELAY1      ;WAITING
7F6F F1       01450          POP    AF             ;RESTORE A REGISTER FROM STACK
7F70 C9       01460          RET                   ;BACK TO CALLING PROGRAM
              01470
7F90          01480          ORG    7F90H          ;START OF TABLE
              01490 ;TABLE   DEFB   23D            ; FOR ASCII BALL
7F90 7F       01500 TABLE    DEFB   127D           ;! FOR STANDARD BALL
7F91 55       01510          DEFB   85D            ;"
7F92 7E       01520          DEFB   126D           ;#
7F93 79       01530          DEFB   121D           ;$
7F94 75       01540          DEFB   117D           ;%
7F95 7D       01550          DEFB   125D           ;&
7F96 15       01560          DEFB   21D            ;'
7F97 70       01570          DEFB   112D           ;(
```

Program continued

```
7F98 71        01580        DEFB    113D        ;)
7F99 7C        01590        DEFB    124D        ;*
7F9A 46        01600        DEFB    70D         ;+
7F9B 0C        01610        DEFB    12D         ;,
7F9C 00        01620        DEFB    0D          ;-
7F9D 16        01630        DEFB    22D         ;.
7F9E 09        01640        DEFB    9D          ;/
7F9F 31        01650        DEFB    49D         ;0
7FA0 3F        01660        DEFB    63D         ;1
7FA1 36        01670        DEFB    54D         ;2
7FA2 3E        01680        DEFB    62D         ;3
7FA3 39        01690        DEFB    57D         ;4
7FA4 35        01700        DEFB    53D         ;5
7FA5 34        01710        DEFB    52D         ;6
7FA6 3D        01720        DEFB    61D         ;7
7FA7 3C        01730        DEFB    60D         ;8
7FA8 30        01740        DEFB    48D         ;9
7FA9 4D        01750        DEFB    77D         ;:
7FAA 0D        01760        DEFB    13D         ;;
               01770 ;      DEFB    127D        ; FOR ASCII BALL
7FAB 29        01780        DEFB    41D         ;L FOR STANDARD BALL
7FAC 06        01790        DEFB    6D          ;=
               01800 ;      DEFB    87D         ; FOR ASCII BALL
7FAD 0F        01810        DEFB    15D         ;G FOR STANDARD BALL
7FAE 49        01820        DEFB    73D         ;?
7FAF 76        01830        DEFB    118D        ;@
7FB0 5C        01840        DEFB    92D         ;A
7FB1 60        01850        DEFB    96D         ;B
7FB2 6C        01860        DEFB    108D        ;C
7FB3 6D        01870        DEFB    109D        ;D
7FB4 65        01880        DEFB    101D        ;E
7FB5 4E        01890        DEFB    78D         ;F
7FB6 4F        01900        DEFB    79D         ;G
7FB7 61        01910        DEFB    97D         ;H
7FB8 54        01920        DEFB    84D         ;I
7FB9 47        01930        DEFB    71D         ;J
7FBA 64        01940        DEFB    100D        ;K
7FBB 69        01950        DEFB    105D        ;L
7FBC 5F        01960        DEFB    95D         ;M
7FBD 66        01970        DEFB    102D        ;N
7FBE 59        01980        DEFB    89D         ;O
7FBF 45        01990        DEFB    69D         ;P
7FC0 44        02000        DEFB    68D         ;Q
7FC1 5D        02010        DEFB    93D         ;R
7FC2 51        02020        DEFB    81D         ;S
7FC3 67        02030        DEFB    103D        ;T
7FC4 6E        02040        DEFB    110D        ;U
7FC5 5E        02050        DEFB    94D         ;V
7FC6 50        02060        DEFB    80D         ;W
7FC7 6F        02070        DEFB    111D        ;X
7FC8 41        02080        DEFB    65D         ;Y
7FC9 77        02090        DEFB    119D        ;Z
               02100 ;      DEFB    86D         ; FOR ASCII BALL
7FCA 65        02110        DEFB    101D        ;E FOR STANDARD BALL
7FCB 80        02120        DEFB    128D        ;
7FCC 41        02130        DEFB    65D         ;
7FCD 42        02140        DEFB    66D         ;
7FCE 43        02150        DEFB    67D         ;
               02160 ;      DEFB    76D         ;VERTICAL LINE IN ASCII
7FCF 76        02170        DEFB    118D        ;
7FD0 1C        02180        DEFB    28D         ;A
7FD1 20        02190        DEFB    32D         ;B
7FD2 2C        02200        DEFB    44D         ;C
7FD3 2D        02210        DEFB    45D         ;D
7FD4 25        02220        DEFB    37D         ;E
7FD5 0E        02230        DEFB    14D         ;F
7FD6 0F        02240        DEFB    15D         ;G
7FD7 21        02250        DEFB    33D         ;H
7FD8 14        02260        DEFB    20D         ;I
7FD9 07        02270        DEFB    7D          ;J
7FDA 24        02280        DEFB    36D         ;K
7FDB 29        02290        DEFB    41D         ;L
7FDC 1F        02300        DEFB    31D         ;M
7FDD 26        02310        DEFB    38D         ;N
7FDE 19        02320        DEFB    25D         ;O
7FDF 05        02330        DEFB    5D          ;P
7FE0 04        02340        DEFB    4D          ;Q
7FE1 1D        02350        DEFB    29D         ;R
7FE2 11        02360        DEFB    17D         ;S
7FE3 27        02370        DEFB    39D         ;T
```

```
7FE4 2E        02380          DEFB    46D          ;U
7FE5 1E        02390          DEFB    30D          ;V
7FE6 10        02400          DEFB    16D          ;W
7FE7 2F        02410          DEFB    47D          ;X
7FE8 01        02420          DEFB    1D           ;Y
7FE9 37        02430          DEFB    55D          ;Z
0002           02440 SAVSTK   DEFS    2            ;STORAGE FOR CALLING ROUTINE
0002           02450 CHCNT    DEFS    2            ;CURRENT CHARACTER ON LINE
               02460                               ;BEING PRINTED
7E9A           02470          END     7E9AH        ;
00000 TOTAL ERRORS
```

Program Listing 2. *BASIC version*

```
 10 REM     IBM PRINT DRIVER PROGRAM
 20 REM     M LEIFER        8/6/81
 21 REM     DEPARTMENT OF ELECTRICAL TECHNOLOGY
 22 REM     ROCKLAND COMMUNITY COLLEGE
 23 REM     145 COLLEGE RD SUFFERN N.Y. 10901
 25 REM
 30 REM     TO IMPLEMENT THIS PROGRAM
 40 REM     RESERVE MEM 32408 THEN LOAD AND RUN THIS PROGRAM
 50 REM     COMPUTER WILL RETURN WITH READY
 60 REM     AND YOU ARE READY TO LLIST AND LPRINT
 70 DIM A(500),B(100)
 80 POKE 16422,171:
    POKE 16423,126
 90 POKE 16424,6:
    POKE 16426,72
100 FOR I = 1 TO 198
110  READ A(I)
120  NEXT I
130 FOR J = 1 TO 89
140  READ B(J)
150  NEXT J
160 FOR I = 32427 TO 32624
170  POKE I,A(I - 32426)
180  NEXT I
190 FOR J = 32656 TO 32745
200  POKE J,B(J - 32655)
210  NEXT J
220 DATA 237,115,234,127,245,197,213,229,221,33,37,64,121
230 DATA 183,40,90,254,33,48,95,254,10,40,16,254,32,32,2
240 DATA 24,15,254,13,204,221,126,205,10,127,24,66,205,2,127
250 DATA 24,61,62,129,245,24,110,221,52,4,221,126,4,221,190
260 DATA 3,32,11,58,1,56,254,8,32,249,221,54,4,0,221
270 DATA 126,5,50,236,127,62,130,205,93,127,205,95,127,201,62
280 DATA 132,205,93,127,205,95,127,245,1,255,143,11,120,177,32
290 DATA 251,241,201,225,209,193,241,237,123,234,127,201,214,33,221
300 DATA 54,4,0,33,144,127,133,111,126,254,64,48,2,24,14
310 DATA 245,219,13,254,63,40,16,62,64,205,93,127,24,9,245
320 DATA 219,13,254,127,40,2,24,240,241,230,63,245,33,236,127
330 DATA 53,32,6,205,221,126,205,10,127,241,205,93,127,24,184
340 DATA 211,13,245,1,255,9,11,58,64,56,254,4,40,170,120
350 DATA 177,32,244,241,201
360 DATA 127,85,126,121,117,125,21,112,113,124,70,12,0,22,9,49
370 DATA 63,54,62,57,53,52,61,60,48,77,13,41,6,15,73
380 DATA 118,92,96,108,109,101,78,79,97,84,71,100,105,95,102,89
390 DATA 69,68,93,81,103,110,94,80,111,65,119,101,128
400 DATA 65,66,67,118,28,32,44,45,37,14,15,33,20,7,36,41,31,38
410 DATA 25,5,4,29,17,39,46,30,16,47,1,55
420 END
```

TUTORIAL

On Towards Better Sorts of Things
Random Distribution Graphics
Using LMOFFSET

TUTORIAL

On Towards Better Sorts of Things

by William R. Patterson

The last time you sorted your mailing list, you may have sat as the computer did its silent thing and wondered how so many nanoseconds could be used for so simple a task. If your mailing list program's sort routine uses an exchange or bubble sort, that might be the reason it takes as long as it does. This is not to disparage the programmer, as there is little in microcomputer literature to tell him or her that there is a better way. (For a discussion in a slightly different direction, see *How to Profit from Your Personal Computer*, by T. G. Lewis, published by Hayden Book Company, Inc., in 1978.)

For those of you who wonder what an exchange sort is, or what a bubble sort is, a short digression is in order. The exchange sort requires the computer to start at the beginning of the list of items, and, for each item, examine all of the items ahead of it and exchange items when they are out of order. See lines 40000 to 40070 of the Program Listing for an example of an exchange sort program. The bubble sort is slightly different; it reviews the list of items as many times as there are items and swaps pairs of contiguous items that are out of order. See lines 41000 to 41070 of the Program Listing for an example of a bubble sort program. Note that N represents the number of items, Z represents the items, and IX is a set of subscripts to Z which the sort routines rearrange.

You may have noticed that the computer must make N times N divided by 2 comparisons to sort things using either of the above methods, which can take a long time for a large number of items. The tree sort I am about to describe will generally take much less time (as long as there are more than ten items and they are very much out of order). In mathematical terms, the number of comparisons made by the tree sort will approximate N times the logarithm of N base 2 ($N(\log_2 N)$).

The reason for the reduced number of comparisons (and hence, reduced time requirements) can be seen if you observe what happens when the sort uses a logical tree of numbers (which will be connected logically inside the computer by means of subscripts, arrays RL, LL, and UL). It sorts the numbers 4, 2, 5, 3, 1, and 6. Take the first number, 4, and place it at the bottom (or "root node" to tree lovers) of the tree:

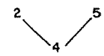
4

For the next number, 2, as it is less than 4, draw a branch up and to the left, and place the 2 there as though the tree had grown a piece of fruit:

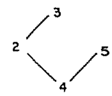
2

4

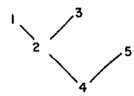
Since 5 is greater than 4, draw a branch up and to the right this time and put the 5 at the end of that branch:



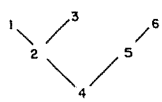
Three is less than 4 and greater than 2, so place it above and to the right of the 2:



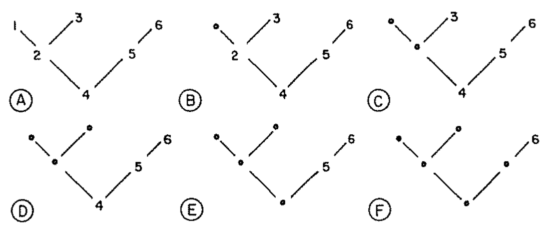
One is less than 4 and 2, so it is placed all the way to the left:



The last number is 6. It is greater than 4 and 5, so place it all the way to the right:



The tree is grown. What you must do now is go over the tree, find the least values, and take them off one by one:



You have now picked the tree clean of sorted items, which came off in order: 1, 2, 3, 4, 5, and 6.

The accompanying BASIC program sorts 100 numbers in less than a minute. The exchange and bubble sorts take about two-and-a-half minutes. This program gives you a choice of techniques. The tree sort itself is kept above line 50000 so you will not have to retype it to use it. Just delete the calling program and enter your own application program to call it.

Program Listing. *On towards better sorts of things*

```
  1 GOSUB 1000
  2 GOSUB 5000
  5 N = 0
 10 B$ = "":
    INPUT B$
 15 RANDOM
 20 IF B$ = "END" OR B$ = "/*"
     THEN
      100
 25 IF LEFT$(B$,3) = "RAN"
     THEN
      200
 26 IF NOT F2
     THEN
      F2 = - 1:
      DIM Z(500),IX(500)
 27 IF LEFT$(B$,3) = "ALP"
     THEN
      AL = - 1:
      GOSUB 300:
      GOTO 100 :
     ELSE
      AL = 0
 30 AA = ASC( LEFT$(B$,1)):
    IF AA < ASC("0") OR AA > ASC("9")
     THEN
      GOSUB 500:
      GOTO 10
 39 N = N + 1
 40 Z(N) = VAL(B$)
 50 GOTO 10
100 FOR I = 1 TO N:
     IX(I) = I:
     NEXT I
105 PRINT "BEGINNING SORT PROCESSING USING ";T$(Q)
110 ON Q GOSUB 40000,41000,50000
112 PRINT "SORT FINISHED"
115 GOSUB 120
117 END
120 FOR I = 1 TO N
140  IF AL
      THEN
       PRINT Z(IX(I)) :
      ELSE
       PRINT Z(IX(I));
150  NEXT I
155 RETURN
160 END
200 INPUT "HOW MANY RANDOM NUMBERS";N
205 IF NOT F2
     THEN
      F2 = - 1 :
      DIM Z(N),IX(N)
210 INPUT "BETWEEN 1 AND WHAT";N1
220 FOR I = 1 TO N
230  Z(I) = RND(N1)
240  NEXT I
250 GOTO 100
300 PRINT "ENTER ALPHABETIC DATA"
320 DEFSTR Z
330 DIM Z(500)
335 I = 0
340 I = I + 1
350 INPUT Z(I)
360 IF LEFT$(Z(I),2) < > "/*" GOTO 340
365 N = I - 1
370 RETURN
```

Program continued

```
 500 PRINT "ONLY ALPHABETICS ALLOWED ARE 'END', '/*', 'RAN', 'ALP' IN
     THE FIRST ENTRY, TRY AGAIN.
 510 RETURN
1000 CLS
1010 PRINT "THIS PROGRAM DEMONSTRATES AN 'IN MEMORY' SORT TECHNIQUE
1020 PRINT "KNOWN AS A 'BINARY TREE SORT' (BECAUSE OF AN INTERNAL
1030 PRINT "DATA STRUCTURE DESCRIBED IN THE ARTICLE) WHICH HAS"
1040 PRINT "CONSIDERABLE SPEED EFFICIENCY OVER THE CONVENTIONAL"
1050 PRINT "EXCHANGE OR BUBBLE SORT, IRONICALLY, AS LONG AS THE ELEME
     NTS "
1060 PRINT "BEING SORTED ARE NOT ALREADY IN ORDER, AND AS LONG AS THE
     RE"
1070 PRINT "ARE MORE THAT ABOUT 10 ELEMENTS IN ALL."
1080 PRINT ""
1090 INPUT "WOULD YOU LIKE TO KNOW MORE('Y','N', OR 'GO' TO GO PAST P
     ROMPTS)";B$
1095 IF B$ = "GO"
      THEN
       Q = 3:
       RETURN
1100 IF ( LEFT$(B$,1)) = "Y"
      THEN
       GOSUB 2000
1105 PRINT ""
1106 GOSUB 1150
1110 PRINT "WOULD YOU LIKE TO HAVE DIRECTIONS FOR USING THE"
1120 INPUT "EXAMPLE CALLING PROGRAM";B$
1130 IF ( LEFT$(B$,1)) = "Y"
      THEN
       GOSUB 3000
1132 GOTO 1170
1150 INPUT "WOULD YOU PREFER A CHOICE OF OTHER SORT TECHNIQUES";B$
1160 IF ( LEFT$(B$,1)) = "Y"
      THEN
       GOSUB 4000 :
      ELSE
       Q = 3
1165 RETURN
1170 INPUT "     ARE YOU READY TO USE THE SORT";B$
1180 IF ( LEFT$(B$,1)) = "Y"
      THEN
       RETURN
1190 INPUT "DO YOU NEED TO LOOK AT THE DIRECTIONS AGAIN";B$
1200 IF ( LEFT$(B$,1)) = "Y"
      THEN
       GOSUB 3000:
       RETURN
1210 PRINT "THEN I WILL TURN CONTROL BACK TO YOU."
1220 END
2000 CLS :
     PRINT "THE TREE SORT ROUTINE IS LOCATED ABOVE LOCATION 50000"
2010 PRINT "SO THAT YOU CAN CODE YOUR OWN ROUTINES TO USE IT AFTER"
2020 PRINT "DELETING THE EXAMPLE CALLING PROGRAM.  SPECIAL REMARKS"
2030 PRINT "ARE INCLUDED TO (1) LET YOU KNOW HOW TO USE IT FROM YOUR"
2040 PRINT "OWN PROGRAMS AND TO (2) LET YOU KNOW WHAT IS GOING ON AS
2050 PRINT "IT WORKS."
2060 PRINT ""
2070 PRINT "THIS IS AN ORIGINAL ALGORITHM BY WILLIAM R. PATTERSON
2080 PRINT "                                  8 POPLAR TERRACE
2090 PRINT "                                  SOMERDALE, N.J.  08083
2100 PRINT "ORIGINALLY CREATED IN PARTIAL FULFILLMENT OF THE
2110 PRINT "REQUIREMENTS OF A MASTER OF BUSINESS ADMINISTRATION
2120 PRINT "DEGREE FROM THE JAMES J. NANCE COLLEGE OF BUSINESS
2130 PRINT "ADMINISTRATION OF THE CLEVELAND STATE UNIVERSITY.
2140 RETURN
3000 CLS
3010 PRINT "     THE EXAMPLE CALLING PROGRAM WILL LET YOU USE THE SOR
     T"
3020 PRINT "IN ANY OF THE FOLLOWING THREE WAYS:"
3030 PRINT "         1.  TO SORT KEYED-IN NUMERIC DATA."
```

```
3040 PRINT "          2.  TO SORT KEYED-IN ALPHANUMERIC DATA."
3050 PRINT "          3.  TO GENERATE AS MUCH RANDOM NUMERIC DATA
3060 PRINT "              AS YOU WANT AND THEN TO SORT IT."
3080 PRINT "      IF YOU WANT TO ENTER NUMERIC DATA, SIMPLY ENTER IT W
     HEN"
3090 PRINT "PROMPTED AND ENTER A '/*' OR AN 'END' WHEN DONE TO TELL T
     HE
3100 PRINT "COMPUTER TO START SORTING.  IF YOU WANT TO ENTER ALPHANUM
     ERIC
3110 PRINT "DATA, ENTER 'ALP' WHEN PROMPTED, THEN ENTER THE DATA IN T
     HE"
3120 PRINT "SAME MANNER AS DESCRIBED FOR NUMERIC DATA (BUT USE '/*' T
     O STARTTHE SORT)."
3140 PRINT "      IF YOU WANT THE NUMERIC DATA GENERATED RANDOMLY, ENT
     ER"
3150 PRINT "'RAN', THEN FOLLOW THE PROMPTS FOR HOW MUCH AND HOW BIG."
3170 RETURN
4000 CLS
4010 PRINT "YOU WILL BE ALLOWED TO CHOOSE THE SORT TECHNIQUE"
4020 PRINT "FROM AMONG THE FOLLOWING:"
4030 PRINT ""
4040 PRINT "          1.  EXCHANGE SORT"
4050 PRINT "          2.  BUBBLE SORT"
4060 PRINT "          3.  TREE SORT"
4070 PRINT ""
4080 PRINT ""
4090 INPUT "PLEASE ENTER THE NUMBER OF THE TECHNIQUE YOU WANT TO USE"
     ;Q
4100 IF Q < 1 OR Q > 3
      THEN
       PRINT "IT MUST BE BETWEEN 1 AND 3.":
       GOTO 4090
4110 RETURN
5000 T$(1) = "EXCHANGE SORT"
5010 T$(2) = "BUBBLE SORT"
5020 T$(3) = "TREE SORT"
5030 RETURN
40000 REM  THIS IS THE EXCHANGE SORT
40010 REM  IT WILL HAVE THE SAME TIMING AS THE BUBBLE SORT
40020 REM  AND WILL USE NO MORE VARIABLES THAT THE TREE SORT
40030 IF N < = 1
       THEN
        RETURN
40040 FOR I = 2 TO N
40050  FOR J = 1 TO I
40060   IF Z(IX(I)) < Z(IX(J))
         THEN
          K = IX(I):
          IX(I) = IX(J):
          IX(J) = K
40070   NEXT :
       NEXT
40080 RETURN
41000 REM  THIS IS THE BUBBLE SORT
41010 REM  IT WILL HAVE THE SAME TIMING AS THE EXCHANGE SORT
41020 REM  AND WILL USE NO MORE VARIABLES THAN THE TREE SORT
41030 IF N < = 1
       THEN
        RETURN
41040 FOR I = 1 TO N
41050  FOR J = 1 TO N - I
41060   IF Z(IX(J)) > Z(IX(J + 1))
         THEN
          K = IX(J):
          IX(J) = IX(J + 1):
          IX(J + 1) = K
41070   NEXT :
       NEXT
41080 RETURN
50000 REM
```

Program continued

```
50010 REM        Z IS THE VECTOR TO BE SORTED
50020 REM        IX IS THE VECTOR OF SUBSCRIPTS TO Z
50030 REM        N IS THE NUMBER OF ELEMENTS
50040 REM       I,KK,UL,RL,LL,J,K,LK,EL,ER,KU  ARE USED
50041 REM       Z AND IX MUST BE DIMENSIONED THE SAME
50042 REM
50043 REM        THIS SUBROUTINE SORTS AN INTEGER VECTOR FROM
50044 REM        THE LOWEST TO THE HIGHEST IN ASCENDING ORDER
50045 REM        BY MANIPULATING A VECTOR OF SUBSCRIPTS TO THE
50046 REM        VECTOR: THIS MANIPULATION IS ACCOMPLISHED BY
50047 REM        MEANS OF A HIGHLY EFFICIENT MECHANISM KNOWN AS
50048 REM        A TREE-STRUCTURE.  THREE BUFFER VECTORS ARE
50049 REM        REQUIRED TO HOLD RIGHT AND LEFT BRANCH LINKS
50050 REM        AS WELL AS BACK LINKS (RL, LL, UL).  THE "TREE"
50051 REM        IS "GROWN" BY AN ARBITRARY ASSIGNMENT OF THE
50052 REM        FIRST INTEGER INDICATED BY THE VECTOR OF SUB-
50053 REM        SCRIPTS TO THE "TRUNK" AND THE SUBSEQUENT
50054 REM        ASSIGNMENT OF SUBSEQUENT NUMBERS TO RIGHT OR
50055 REM        LEFT BRANCHES DEPENDING ON WHETHER THE RESPEC-
50056 REM        TIVE SUBSEQUENT NUMBER IS EITHER LESS THAN (OR
50057 REM        EQUAL TO) OR GREATER THAN THE PRECEEDING
50058 REM        NUMBER.  AFTER THE GROWING PROCESS IS COMPLETED
50059 REM        THE "FRUIT" MAY BE PICKED FROM THE TREE BY
50060 REM        SIMPLY RESPECTING THE RULES UNDER WHICH THE
50061 REM        TREE WAS GROWN.
50062 REM
50064 IF N < = 1
       THEN
        RETURN
50065 IF NOT F1
       THEN
        F1 = - 1 :
        DIM RL(N),LL(N),UL(N)
50070 REM      INITIALIZE COUNTER AND TREE TRUNK
50080 I = 1:
      KK = IX(I):
      UL(KK) = 0:
      RL(KK) = 0:
      LL(KK) = 0
50090 REM      INCREMENT COUNTER
50100 I = I + 1
50110 REM      END OF VECTOR?  TREE WILL BE FULLY GROWN
50120 IF I > N
       THEN
        50390
50130 REM      ESTABLISH SUBSCRIPT OF NEXT NUMBER
50140 J = IX(I)
50150 REM      OBTAIN TREE TRUNK
50160 K = KK
50170 REM      THIS IS THE SORT MECHANISM
50180 IF (Z(K) < = Z(J))
       THEN
        50310
50190 REM      WE HAVE A LOWER NUMBER
50200 LK = LL(K)
50210 REM      IS THERE A LEFT LINK?
50220 IF (LK = 0)
       THEN
        50260
50230 REM      *YES*  MOVE TO THE LEFT
50240 K = LK:
      GOTO 50180
50250 REM      *NO* LINK THIS NUMBER TO THE LEFT
50260 LL(K) = J
50270 UL(J) = K:
      LL(J) = 0:
      RL(J) = 0
50280 REM      GO GET NEXT NUMBER
50290 GOTO 50100
50300 REM      WE HAVE A HIGHER NUMBER
```

```
50310 LK = RL(K)
50320 REM       IS THERE A RIGHT LINK
50330 IF LK = 0
       THEN
        50370
50340 REM       *YES* MOVE TO THE RIGHT
50350 K = LK:
      GOTO 50180
50360 REM       *NO* LINK THIS NUMBER TO THE RIGHT
50370 RL(K) = J:
      GOTO 50270
50380 REM       TREE IS FULLY GROWN AND THE FRUIT IS READY TO BEPICKED
      -- START CLIMBING AT THE TRUNK
50390 K = KK
50400 REM       RESET VECTOR POSITION COUNTER
50410 I = 0
50420 REM       LOOK TO THE LEFT
50430 EL = LL(K):
      IF EL < > 0
       THEN
        50570
50440 REM       *NO BRANCH* LOOK TO THE RIGHT
50450 ER = RL(K):
      IF ER < > 0
       THEN
        50590
50460 REM       *NO BRANCH* PUT FRUIT IN HOPPER
50470 I = I + 1:
      IX(I) = K
50480 REM       CLIMB DOWN ONE BRANCH
50490 KU = UL(K)
50500 REM       REACH BOTTOM?  IF SO:  QUITTING TIME
50510 IF KU = 0 RETURN
50520 REM       DID WE COME DOWN FROM THE RIGHT OR THE LEFT?
50530 IF LL(KU) = K
       THEN
        50610
50540 REM       *RIGHT*  CONTINUE CLIMBING DOWN
50550 K = KU:
      GOTO 50490
50560 REM       CLIMB LEFT
50570 K = EL:
      GOTO 50420
50580 REM       PUT FRUIT IN HOPPER AND CLIMB RIGHT.
50590 I = I + 1:
      IX(I) = K:
      K = ER:
      GOTO 50420
50600 REM       SAW OFF LEFT BRANCH AND CLIMB DOWN
50610 LL(KU) = 0:
      K = KU:
      GOTO 50420
```

Random Distribution Graphics

by Todd L. Carpenter

At the heart of most game programs is a statement of chance, the RANDOM statement. Having the ability to look at the shape of the RANDOM distribution can give you the power of shaping the distribution to suit your purposes.

Graphics displays on the TRS-80 certainly have their limitations, but there is one type of display the TRS-80 handles rather nicely—the bar graph. If you are interested in the statement Y = RND(X), it is important for you to understand the distribution characteristics of Y over its range (1 to X). A bar graph can display this with a touch of elegance.

Is RANDOM Really Random?

After several weeks of playing with the custom Star Trek program I wrote, I noticed that the majority of the Klingons were always located near the center of the galaxy. Rarely did I ever find a Klingon in any of the perimeter quadrants. I thought I had used a simple Y = RND(X) statement in distributing the Klingons, but it seemed that either my RANDOM statement was not truly random or the Klingons had succeeded in outsmarting Captain Carpenter. I chose to pursue the former suspicion because, after all, the Klingons are the bad guys, and they could not outsmart me—could they?

I set out to write a simple program that would show me once and for all whether or not the RANDOM statement really gave me a uniform random distribution. The purpose of the program was to display in a single picture the distribution of the RND(X) statement. The ability to see the RANDOM distribution would enable me to determine immediately the actual randomness of the statement.

I was prepared to make a shattering discovery that Radio Shack had goofed in their design of the RND(X) statement. But why had no one else discovered this biased RANDOM statement? Perhaps, I thought, the bias was slight, and I had discovered it only because my program used the RANDOM statement over 4,000 times in distributing the elements of the galaxy. As you will see, it was Captain Carpenter who had goofed, not Radio Shack.

Random Shaping

After a closer examination of my Star Trek Program, I discovered that I had inadvertently used a combination of RANDOM statements. How could

I test the distribution of this combination? After a few generalizations in my program, I was ready to run an analysis on any combination of RANDOM statements that could start with Y = . I proceeded to test my Klingon distribution. Sure enough, they were doing just what I had been telling them to do, concentrating in the middle. In separate parts of the program, I had mistakenly used what amounted to the sum of two RANDOM statements and gotten a dice-like distribution. (See Figure 1.) As all craps players should know, when rolling two dice, more 7s turn up than 2s or 12s. In fact, six times as many 7s turn up.

The advantage of seeing any RANDOM distribution before entering it is that the shape of a distribution can be selected to fit an application. Once you know how to generate some simple shapes, the next steps seem easier.

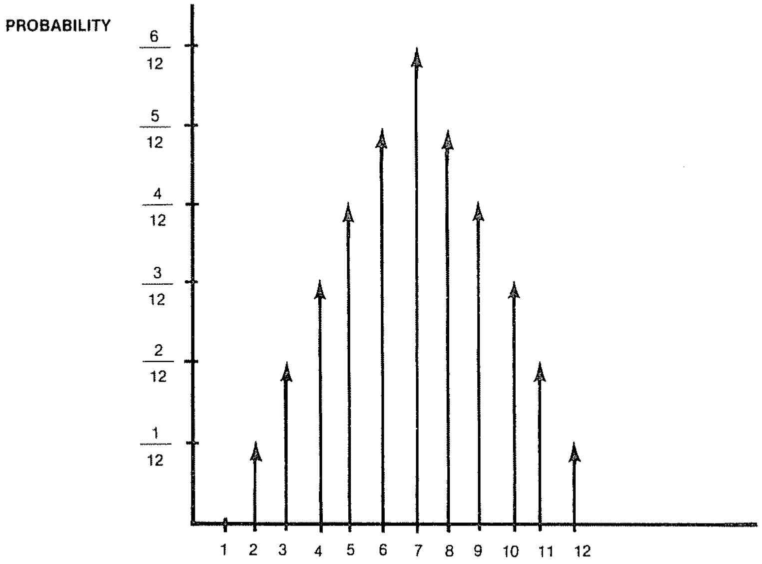


Figure 1. *Probability curve when rolling two dice*

Program Inputs

The program (see Program Listing) starts by asking for the value of X in the RND(X) statement. It can be any number greater than zero and preferably an integer (although the machine will accept a decimal value and find the integer value itself). For the case of the simplest RANDOM statement, Y = RND(X), the function Y is uniformly distributed from 1 to X. This means that for a single trial, the probability is the same for getting any integer value from 1 to X. For example, X = 6 is analogous to the case of rolling one die. With six faces, the probability that any particular face comes up is 1/6. See Figure 2.

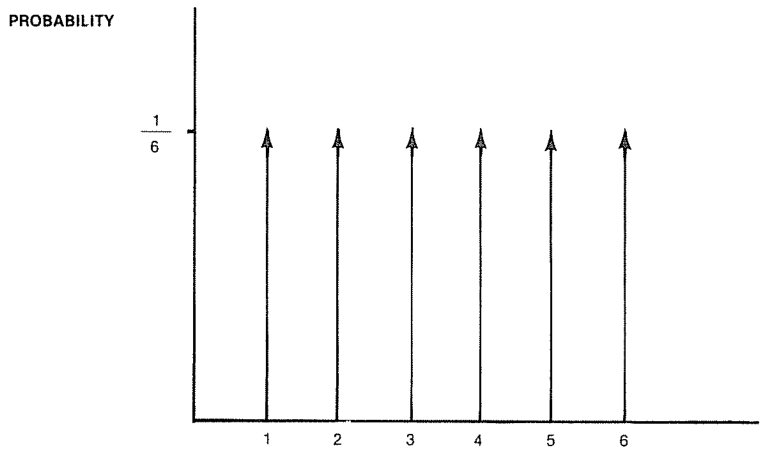


Figure 2. *Probability curve when rolling one die*

Next, input the number of trials to be made. For our example, this would be the number of rolls of the single die. The greater the number of trials performed the more the graph will be delineated. The number of trials made must be large compared to the entered value of X. As a rule of thumb, I make the number of trials at least 20 times the maximum value that Y can be. In this case, make Y equal to X or 6.

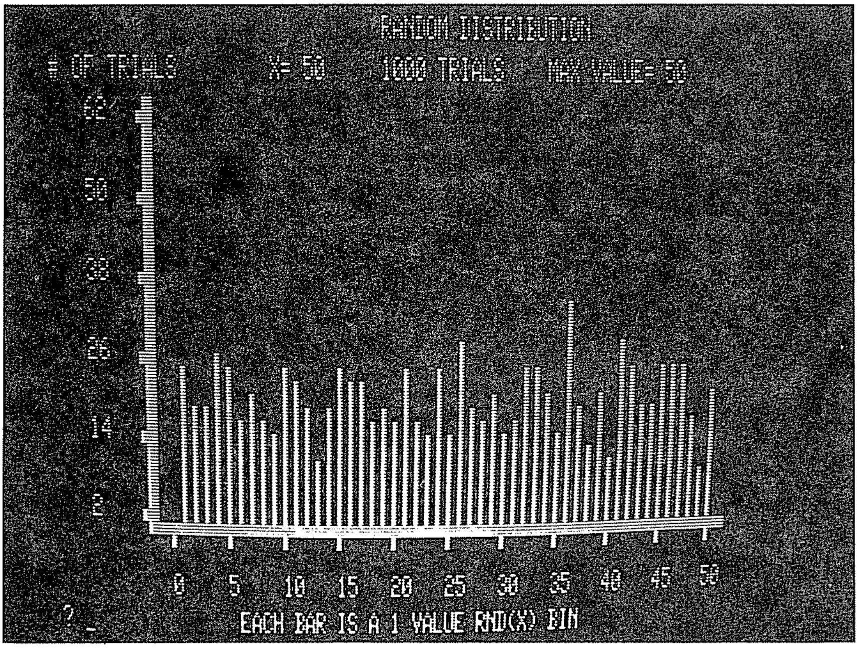


Photo 1. *Graph of Y = RND(50), 1000 trials (Photo by Yuan Chang Lo)*

You are now ready to take a peek at Photo 1 which shows a graph of the function, Y = RND(50). There were 1,000 trials, the minimum rule of thumb value, used to determine this graph. (50 values times 20 trial outcomes per value, equals 1,000 total trials.) As you can see, it yields quite an uneven distribution.

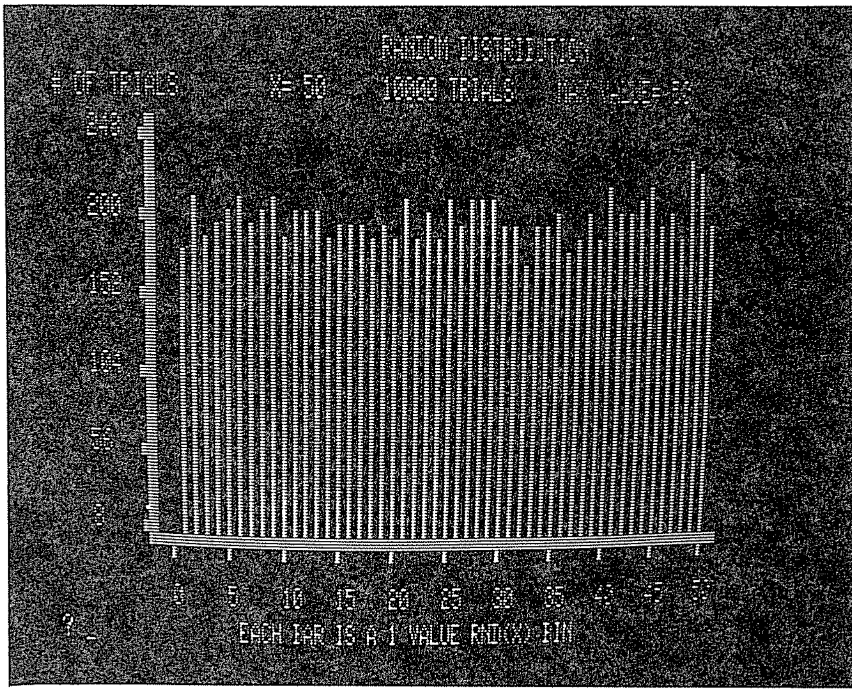


Photo 2. *Graph of Y = RND(50), 10000 trials (Photo by Yuan Chang Lo)*

I chose to use the number of trial outcomes for the vertical axis rather than probability in this case. But, either way, the shape of the graph is the same.

Now consider Photo 2. I ran the same distribution, but this time with 10,000 trials. As you would expect, the average number of values per "bin" is now 10 times what it was in the previous example, or 200. I have coined the word bin to refer to each bar of the graph. A bar getting larger can be thought of as a bin being filled.

Changing the Distribution

There are two important statements in the program. They are the RANDOM statement and the MAX VALUE statement. The RANDOM statement is at line 810 and contains the expression which determines the shape of the distribution. This statement must be edited manually whenever a new expression is desired. The MAX VALUE statement is at line 730 and defines the variable M which must be set equal to the largest possible value Y can be in the RANDOM statement. In this listing shown, Y = RND(X) + RND(X), so M = X + X. For instance, if line 810 read Y = X − RND(X), line 730 would read M = X − 1. (When a term is subtracted, use its minimum value.)

Photo 3 shows the distribution of the equation in the Program Listing. I chose to enter X = 6 so I would be able to extend the dice rolling analogy. This time I rolled two dice and got a distribution such that the most likely number to come up, 7, was in the center. This is essentially how my Klingons were distributing themselves.

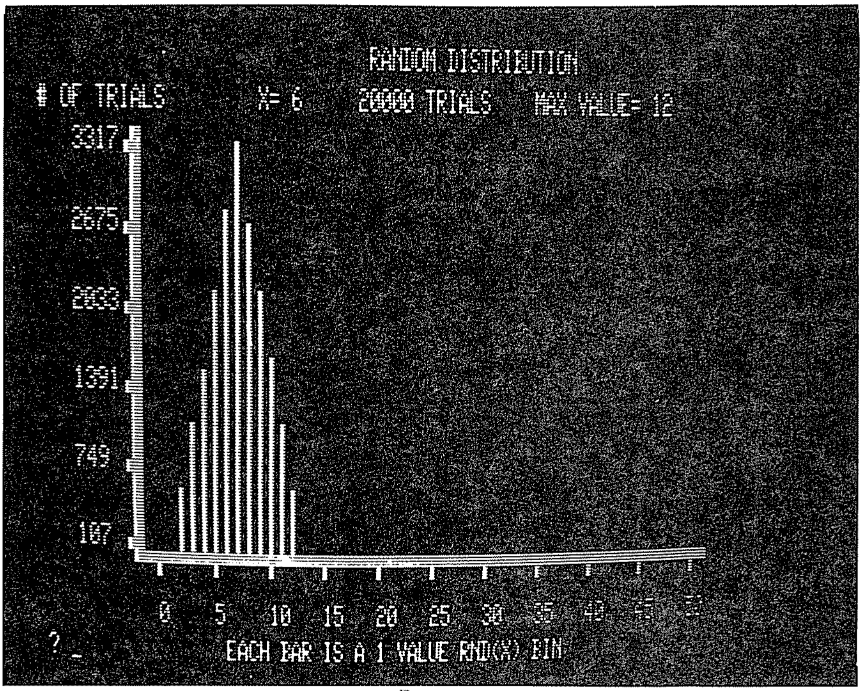


Photo 3. *Graph of Y = RND(X) + RND(X), 20000 trials (Photo by Yuan Chang Lo)*

Now, we move on to some more complicated distributions. Photo 4 shows a graph of the distribution, Y = X + RND(RND(X)) − RND(RND(X)), where M = X + X − 1. This was run with 30,000 trials, and quite a smooth graph was obtained.

Photo 5 shows a normal distribution for those of you interested in statistics. It can be approximated by using a large sum of simple RND(X) statements. I used six terms here.

As more and more sophisticated functions are used, a definite limitation crops up. A simple statement like Y = RND(X) takes about six times as long to execute as a FOR-NEXT loop pair, and the statement Y = RND(RND(X))

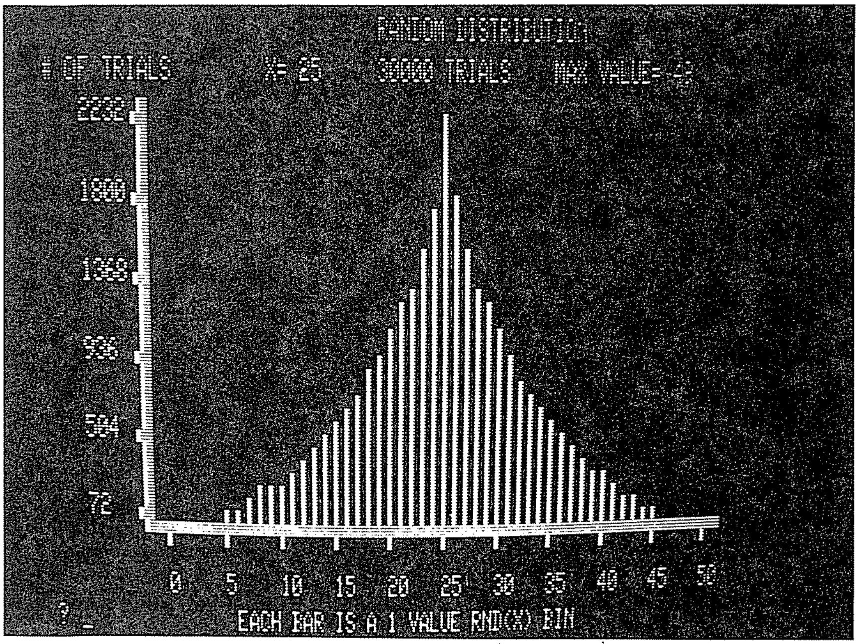


Photo 4. *Graph of Y = X + RND(RND(X)) – RND(RND(X)), 30000 trials (Photo by Yuan Chang Lo)*

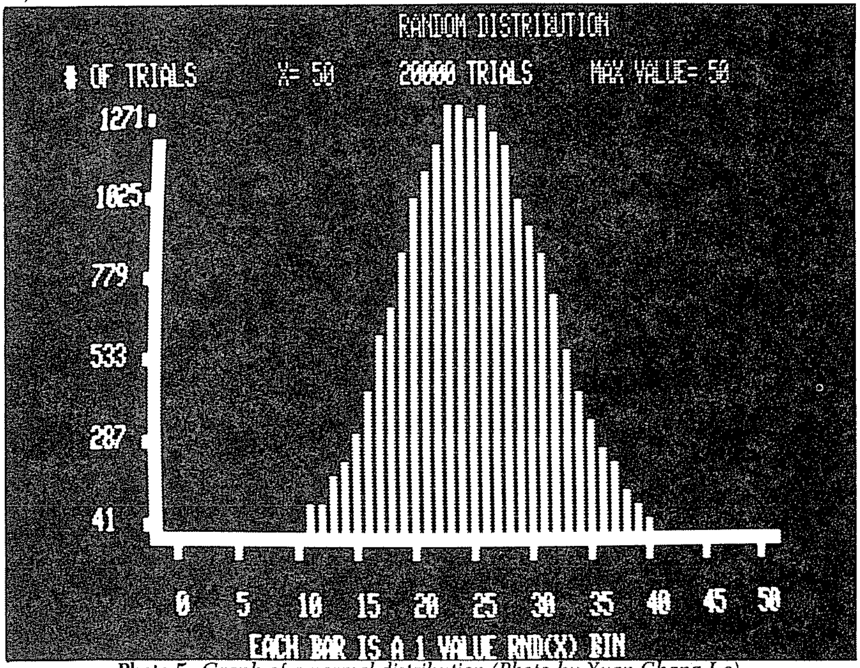


Photo 5. *Graph of a normal distribution (Photo by Yuan Chang Lo)*

takes about 10 times as long. In other words, this program can take quite a long time to run through 30,000 trials. With that in mind, it's wise to start testing a new function with the minimum rule of thumb number of trials. If Y = RND(RND(X)*2), M = X*2, and you enter X = 50, then you should enter the number of trials as 2,000 (20 times M). This will not produce a very smooth graph, but will take only about 1/15th as much time to run. Usually this is about two to three minutes.

Auto Scaling

This brings up one last significant feature of the program. You have seen how the vertical axis scales itself depending on the maximum number of trial outcomes per bin. The same thing applies to the horizontal axis. You are not limited to a maximum value of 50. It can be 51 or 135 or 1,000 or whatever you like. If the graph shape is all that is desired, this can generally be accomplished with 50 as a maximum value.

If you decide that some larger number is more convenient, then the axis will be automatically scaled. There will never be more than 50 bins in which to accumulate trial points, but if the maximum value is 64, for example, the axis will be scaled down by a factor of two. This makes each bin a two-value, rather than single-value bin.

Now you have an elegantly simple program that lets you see what the RANDOM statement can do. Thanks to this program, my Klingons have been controlled, the galaxy has been saved, and Starfleet Command will not have to give me a desk job.

Program Listing

```
100 :
    '   ** RANDOM DISTRIBUTION GRAPHICS PROGRAM **
110 :
    '   **                                      **
120 :
    '   ** TODD L. CARPENTER                    **
130 :
    '   ** 10/21/81                             **
140 :
    '   **                                      **
180 DEFINT A - Z
190 DIM A(50)
300 CLS
310 PRINT :
    PRINT "INPUT X FOR THE RND(X) STATEMENT."
320 PRINT "IT MUST BE A POSITIVE NUMBER:";
330 PRINT "      0 <= RND(X) <= X"
340 PRINT @ 330," ";:
    INPUT X
400 CLS
410 PRINT "INPUT THE DESIRED NUMBER OF TRIALS."
420 PRINT "THE GREATER THE NUMBER OF TRIALS,
430 PRINT "THE SMOOTHER THE GRAPH."
440 PRINT @ 330," ";:
    INPUT C
450 CLS
500 PRINT @ 348,"THINKING";
510 K = 0:
    F = 50
520 FOR I = 0 TO 50
530  A(I) = 0
540  NEXT I
560 IF X < = 0
     THEN
      300
700 :
    '   **  CALCULATE RND(X) VALUES   **
710 :
    '   **  MAX VALUE STATEMENT       **
720 :
    '   ******************************
730 M = X + X
740 :
    '   ******************************
760 L = M
770 IF M < 50
     THEN
      M = 50
775 IF M > 50
     THEN
      M = ( INT((M - 1) / 50) + 1) * 50
780 N = M / 50
790 FOR I = 1 TO C
800  :
     '   **  RND(X) STATEMENT  **
805  :
     '   ***********************
810  Y = RND(X) + RND(X)
820  :
     '   ***********************
830  B = Y * 50 / M
840  IF B < 0
      THEN
       870
850  A(B) = A(B) + 1
860  IF K < A(B)
      THEN
```

Program continued

```
      K = A(B)
 870  NEXT I
1000 :
     '  **  PLOT X-AXIS  **
1010 CLS
1020 J = K / 32 + 1
1030 FOR I = 16 TO 123
1040  SET (I,38)
1050  IF INT(I / 10) = I / 10
       THEN
        SET (I,39)
1060  NEXT I
1100 :
     '  **  LABEL X-AXIS  **
1110 FOR I = 0 TO 10
1120  PRINT @ 905 + 5 * I,5 * I * N;
1130  NEXT I
1200 :
     '  **  LABEL Y-AXIS  **
1210 FOR I = 0 TO 5
1220  PRINT @ 770 - I * 128,J + J * 6 * I;
1230  NEXT I
1300 :
     '   **  PLOT Y-AXIS  **
1310 FOR I = 6 TO 38
1320  IF INT((I - 2) / 6) = (I - 2) / 6
       THEN
        SET (15,I - 1)
1330  SET (16,I):
      SET (17,I)
1340  NEXT I
1400 :
     '  **  HEADING AND LABELS  **
1410 PRINT @ 29,"RANDOM DISTRIBUTION"
1420 PRINT @ 83,"X=";X;"   ";C;"TRIALS    MAX VALUE=";L;
1430 PRINT @ 64,"# OF TRIALS";
1440 PRINT @ 976,"EACH BAR IS A";N;"VALUE RND(X) BIN";
2000 :
     '  **  PLOT GRAPH  **
2010 FOR I = 0 TO 50
2020  IF A(I) < J
       THEN
        2060
2030  FOR H = 1 TO A(I) / J
2040   SET (20 + 2 * I,38 - H)
2050   NEXT H
2060  NEXT I
5000 PRINT @ 960,"";
6000 INPUT I
7000 GOTO 300
9999 :
     '  **  END  **
```

Using LMOFFSET

by John T. Blair and Peter B. Hall

Those readers who are getting started with their TRS-80s may not realize that machine language is the only language the computer understands. BASIC programs are high-level (English-like) language programs that need to be interpreted into machine language for the computer. BASIC programs are easy to understand and easy to write or modify. More efficient programs are written in machine language so an interpreter is not needed. They require less memory and execute faster.

Although a BASIC program is easy to transfer from tape to disk using conventional BASIC commands (CLOAD from tape and SAVE to disk), a utility program must be specially written to perform that function for machine-language programs. LMOFFSET, one of the utilities supplied by Apparat with their NEWDOS + disk operating system, stands for Load Module OFFSET. Its simplest usage is to move a machine-language program from tape to disk. It can also be used to copy a program from disk to disk. It cannot be used for BASIC programs.

Figure 1 shows a screen display after a typical use of LMOFFSET to load a program from tape, reassign the area of memory for it to occupy, and then save it to disk.

When LMOFFSET is first executed, it displays the sign-on message:

```
APPARAT LOAD MODULE OFFSET PROGRAM, VERSION 1.1
SOURCE FROM DISK OR TAPE? REPLY "D" or "T".
```

Assume you have a machine-language program on tape which you want to move over to your newly acquired disk. Two examples of this are SYSTEM games tapes and the tape version of Electric Pencil. (Keep in mind that just because the program has been moved to disk, it does *not* allow any disk LOADs or SAVEs unless the program originally had that option. For the Electric Pencil, you still have to save files to tape. The only thing you gain is the decrease in loading time of the program itself.)

In Figure 1, a T (for tape) is entered. LMOFFSET then loads the program from tape into a buffer area set aside in memory above LMOFFSET. Immediately after the T is entered, an asterisk appears in the upper right corner of the CRT. After the A5 sync byte (the code that the computer uses to determine the beginning of the program on tape) is found, two more asterisks appear. These blink unless the program contains an error. The error codes are as follows:

- C = Bad checksum
- P = Leading extraneous bytes
- I = Imbedded extraneous bytes

When the tape finishes loading, the designated memory area that it occupies during execution is displayed on the third line. This memory area may or may not be the same as that reserved for parts of the DOS. Next, LMOFFSET displays one of three possibilities.

1) If the program loads from 7000H (the H designates that the address is in hexadecimal) or higher, the screen displays:

```
MODULE LOADS TO XXXX-XXXX
ENTRY POINT = YYYY
NEW LOAD BASE ADDRESS (HEX)?
```

Since this program does not conflict with DOS, all you have to do is press ENTER and continue by specifying the destination filespec as described later.

2) If the program loads from 4000–51FFH, the screen displays:

```
MODULE LOADS TO XXXX-XXXX
MODULE LOAD OVERLAPS DOS RAM 4000-51FF
MODULE LOAD WILL OVERLAP "CMD" PROGRAM AREA (5200-6FFF)
ENTRY POINT = YYYY
NEW LOAD BASE ADDRESS (HEX)?
```

In this case, if any or all of the new program loads into the DOS overlay area, the new program and DOS collide and cause a reboot. You have to specify a new load base address.

3) If the program loads above 51FFH, but below 7000H, the screen displays:

```
MODULE LOADS TO XXXX-XXXX
MODULE LOAD WILL OVERLAP "CMD" PROGRAM AREA (5200-6FFF)
ENTRY POINT = YYYY
NEW LOAD BASE ADDRESS (HEX)?
```

If a program were to reside here, you would not be able to use some DOS commands without destroying the program. You may want to specify a new load base address. The new load base address is the area in memory where the user wants the program to be stored. If the program read from tape loads above 7000H, press ENTER. If it does not, enter 7000.

The program now asks SHALL APPENDAGE BE SUPPRESSED (Y OR N)? If you entered any number in the previous step, you should answer N. This appendage is a new loader which moves the program into the correct area for execution. The program calculates the new storage area and entry point and displays this in lines 9 and 10 of the printout. Again, a new load base address is requested. If you made a mistake in entering it before, this is your chance to correct it. If not, press the ENTER key. Now the interrupts can be disabled. The answer to this question should be Y.

Finally, the file specification is requested. If you plan to use this program and are moving the program from tape to disk, you must conform with the

DOS requirements. The file name should have an extension /CMD. (If you have more than one dirve, be sure to assign the drive number.) If you plan to work on this copy, however, we suggest the extension /MOV to keep from confusing it with the original. If the extension /CMD is not used, you have to type RUN, plus the complete file name in order to load and execute the program. Type LOAD and the file name to load the program and return to DOS.

Looking Deeper

What does LMOFFSET really do? To answer that let us just touch on machine language. Most machine-language programs are not relocatable. This means that when they were written, several instructions made reference to a specific memory location. Using Electric Pencil as our example, from the printout in Figure 1, the program loads from 4000H to 51FFH. Many memory cells in this program are used for data storage, such as the number of lines to be printed, the end-of-text area, et cetera. To get the data from these areas, Electric Pencil loads the contents of the desired memory cells. Suppose that 4400H contains the end-of-text pointer. If this program were moved to 7000H, the data stored in 4400H might now be at 7400H. When the program wants this information it addresses 4400H, not 7400H. The data, therefore, will be wrong. This means the program is not relocatable. Other instructions such as LDs, CALLs, and JPs, make a program nonrelocatable.

```
APPARAT LOAD MODULE OFFSET PROGRAM, VERSION 1.1
SOURCE FROM DISK OR TAPE? REPLY "D" OR "T"? T
MODULE LOADS TO 4200-5500
MODULE LOAD OVERLAPS DOS RAM (4000-51FF)
MODULE LOAD WILL OVERLAP "CMD" PROGRAM AREA (5200-6FFF)
ENTRY POINT = 4350
NEW LOAD BASE ADDRESS (HEX)?7000
SHALL APPENDAGE BE SUPPRESSED (Y OR N)? N
MODULE LOADS TO 7000-830F
ENTRY POINT = 8301
NEW LOAD BASE ADDRESS (HEX)?
INTERRUPTS TO BE DISABLED (Y OR N)? Y
DESTINATION FILESPEC? PENCILT/CMD
```

Figure 1. *LMOFFSET output for Electric Pencil*

Why do we have to move a program anyway? The reason is simple. The memory map in the back of the Level II manual shows that the memory from 43E8H and up is used for program storage. When the disk is added, memory from 43E8H to 5200H stores the Disk Operating System (DOS). If Electric Pencil were not moved up in memory as it loaded from disk, it would overwrite the DOS. The DOS would eventually crash, and you could not load Pencil. This is why we moved it up to 7000H. But didn't we just say that the program would not run in that memory area? That is the reason for the Loader.

The Loader is nothing more than a block-move program. This appendage consists of 15 bytes of machine code which are tail-ended to the original program. The first byte is an F3 (to disable interrupts) or a 00 (NOP), depending on the answer you give to the DISABLE INTERRUPTS? question. The next three bytes are 21 XX YY (LD HL,YYXX), which set where the program is moved from. The next three bytes are 11 SS TT (LD DE,TTSS) which set where the program is moved to. The next three bytes are 01 DD EE (LD BC,EEDD) which set the number of bytes of program to move (the difference between the original starting address and the original ending address without the appendage). The next two bytes are ED B0 (LDIR). This instruction first takes the byte pointed to by the HL register pair and moves it to the address pointed to by the DE register pair, subtracts one from the BC register pair, and checks to see if the number in the BC equals 00. If it does not, DE and HL are incremented, and the instruction is repeated until BC does equal 00. Then the machine goes on to the next instruction. The final three bytes are C3 FF GG (JP GGFF) which tell the computer to execute the original program. That's it! All 15 bytes, and the new program is where it wanted to be.

New Load Base Address

The new base address is entered above 6FFFH to keep it out of the DOS program area. This allows the system to load the program from disk to memory. The entry point is changed to the Loader. After the program has been loaded, execution is transferred to the Loader. Since the DOS is no longer required, the Loader moves the program down to the area where it must run. The Loader then jumps to the original entry point, and you are off and running.

What you choose for a new load base address depends on how much memory you have and what you want to do with the new module. Assuming you want to disassemble a program to modify it, you have to decide what tool (utility) you will use to do the work. Once you have decided that, you must know where this utility loads. Now, based on where the utility resides in memory, you can determine if the new module will load above 7000H and below your utility or if you must load the new module above the utility. The start of memory for the new module should be entered for the new load base address.

Working on a Program

Figure 1 is a printout of the new block-move routine. After the question SHOULD THE APPENDAGE BE SUPPRESSED?, LMOFFSET displays the new storage area and the new entry point.

To work on this program, you must load it into memory using the LOAD command from the DOS (be sure to include the extension with the file name), and execute the monitor or utility you wish to use.

There are two methods for working on a program once you have it in memory. The first is to work on it in the new area. We do not recommend this method since many of the machine-language instructions directly address a memory location and all refer to the cells where the program was designed to run. This makes deciphering the program difficult.

The second method is to load the moved version into memory and execute the monitor. Then block move the program down to where it should run or modify the Loader. To modify the Loader, disassemble it and change the address of the JP instruction to that of your monitor. Now execute the program with the new entry point. After the Loader moves the program to where it will run, it returns control to your monitor. After you modify the program, you must block move it back to where it was. Remember, you do not have any disk functions if the program is to load below 5200H, and if it loads below 6FFFH, many of the DOS utilities will write over it. To save your modified program to disk, exit the utility and use the DOS command DUMP or load your favorite disk I/O utility.

Exceptions

Scott Adams' Adventure programs may not load and run. It will be evident when the screen goes bonkers. To correct this, reboot while pressing the shift and up arrow. When the DOS READY prompt comes up, let up on the keys. You have now disabled the key debounce routine. This routine conflicts with other programs as well. You can now run and enjoy the Adventure program. If the program has a loader that loads the program, you will have to disassemble the loader and try to figure out what is going on, or find a friend who has already broken the loader and has the program on disk.

UTILITY

Extractor: An Ace in the Hole!
Page Print Your Listings
Let Your TRS-80 Do the Typing

Extractor: An Ace in the Hole!

by J. Crutcher

About a year ago, a notepad fell victim to a spring cleaning assault at my house. Unfortunately, that pad contained the index to my library of cassette tapes for my TRS-80 Level II computer. My first reaction was panic, after which reason prevailed, and I started to reconstruct the list. Several days later, I gave up, and from that experience, Extractor was born. I needed a way to extract the necessary information from Radio Shack's SYSTEM formatted tapes. The Program Listing does that. Extractor reads the name, load address, and automatic start address of SYSTEM tapes and displays them on the screen without loading the program into memory. In addition, it computes and displays the end address. I have divided the format into four sections for purposes of explanation: (1) leader and sync, (2) name header and name, (3) data header, block length, start address, data block, and checksum, (4) auto-start header and auto-start address.

The leader consists of 255 bytes of zeros, each of which consists of eight bits. These bits are separated by clock pulses which slice the leader into time slots. This same division of time is used for the entire SYSTEM formatted tape with data contained within these time slots. The sync byte (A5H) is the 256th byte on the tape, and in conjunction with the preceding 255 bytes of 00Hs, it will synchronize the computer with the incoming data. Refer to Figure 1. The name header (55H) points to the six-byte alphanumeric name which is used as a search key under the SYSTEM command to select a particular file from the tape. These sections are graphically depicted in Figure 2.

After the name comes a data header (3CH) which precedes three bytes of very important information. The first byte following the data header is the block length byte. This two-digit hex byte tells the computer how many bytes of data are in the following data block (00H represents 256 and is the maximum allowed per data block). This information tells the computer when to expect the next data header (3CH) or the auto-start header (78H). Without this, the computer would consider a 3CH code as 60 decimal or INC A. Immediately following the block length is the low byte portion of a two-byte address which is the load/start address. During a normal load, the first byte of data loads into memory at this address. Next, the data block itself, which contains the program data, is loaded. It is one to 256 bytes long, with an additional byte at the end called the checksum. This is a method Radio Shack uses to establish a measure of data integrity or validity by a type of parity derived by adding the low and high bytes of the load/start address to the absolute value of each byte of data in that data block and discarding any carry that results. This sum is then recorded on the tape for use during a cassette load to determine if the data is good or bad. Cassette load problems (high or low amplitude) can trigger a checksum error. Figure 3 details these divisions of the format. If another data block is present on the tape, another data header (3CH) would be next, and another block length, load/start address, data block, and checksum cycle would follow.

If the preceding data block was the last, the next byte would contain an auto-start header (78H) to indicate to the computer that the next two bytes contain the auto-start address or entry point for that program. When you respond to the SYSTEM command prompt (*?) with a / ENTER, the computer begins execution at the auto-start address. The code (78H) also signals the computer firmware to terminate cassette operation. Figure 4 shows this last section.

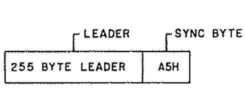


Figure 1. *Leader and sync byte*

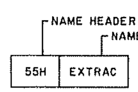


Figure 2. *Name header and name*

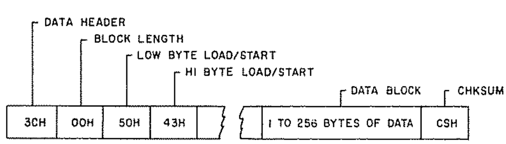


Figure 3. *Data header, block length, load/start address, data block, checksum*

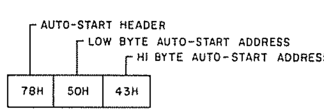


Figure 4. *Auto-start header, auto-start address*

With the preceding information, you can now develop a method to extract the name, load/start address, end address, and auto-start address from any SYSTEM format tape recorded from the TRS-80 Level II. (Warning: This program will not decode commercial tapes such as Microchess, Defender, and others which contain their own loaders.) I used Radio Shack's Editor Assembler to assemble the Program Listing at address 4350H. I have used labels throughout so that you can relocate the code wherever it is convenient, simply by changing the ORG statement in line 2530 and reassembling. You can change the message displayed by using DEFM pseudo-ops and retyping your own messages. Since DEFMs are limited to 63 bytes or less, you must use multiple DEFMs for messages which exceed this limitation. Since the message print routine uses zeros as end indicators, you must remove all such delimiters between multiple DEFMs and insert at least one NOP at the end of the message. I made no attempt to condense the program or to make it super efficient since it uses less than 600 bytes of memory. Speed is not important, as the program deals with a very slow medium—the cassette tape. Remember that during its operation, the program being indexed does not load into memory, so even a 4K system has adequate memory regardless of the length of the program on the tape. I have included some suggestions on how to proceed with a checksum check procedure and a single byte addition which will load into memory the programs being indexed. Remember that a limit does exist as to how much processing you can do between calls for data from the tape. You must consider the time of execution for program steps when you add significant amounts of code to the program in those areas where the tape is read.

The first part of Extractor deals with the display of title, authorship, and the entry prompt. It begins by clearing the screen and displaying messages (lines 2810 through 2860). The entry prompt advises you to hit ENTER to continue, and you may follow these instructions if you wish, but any key will do the job. I got into a rut a few years ago and wore out two ENTER switches before I finally changed the way I wrote program entry prompts. If your ENTER key is overworked, use the CLEAR or any other key for a while. Line 2870 establishes a buffer which is used later to store the load/start address and the end address. Lines 2940 through 3000 are the keyboard loop which puts the computer into a posture of waiting until the requested entry is made. Lines 3040 through 3060 then clear the screen again and display the column headings for the data to be extracted.

The next part of the program does three things. First, it zeroes the buffers established in line 2870 (lines 3212 through 3218). Then it turns on the cassette recorder (line 3220) and searches for and recognizes the leader and sync byte (line 3230). When it finds the name header (line 3250 and 3260), it displays the name on the screen under the column titled NAME (lines 3270 through 3300). If not found, an error message appears, (line 3260) and the cassette recorder stops. Before leaving this section, I would like to bring to your attention the NEXT label in line 3212. This is the point of reentry for multiple program indexing from the same tape.

The next part of the program concerns the data and auto-start headers and the information which follows them. Lines 3410 through 3430 test the header to determine if it is an auto-start header (78H) signifying that all the data blocks have been processed. If it is, the program jumps over the data block processing section and terminates the program (line 3430). If it is not an auto-start header, the program tests to see if it is a data header (line 3440). If it is not a data header, an error message appears (line 3450). If the header is a data header (3CH), several things happen. Line 3460 displays two asterisks in the upper right corner of the screen and makes the rightmost one blink alternately as the program finds additional data headers (3CH). Line 3470 reads the block length from the tape, and line 3480 stores it in the B register to be used later to count the bytes of data. The next byte on the tape is the low byte part of the two-byte load/start address which lines 3500 and 3510 read and load into the L register. This is followed by the high byte of the load/start address (lines 3540 and 3550) which processes into the H register. At this point the HL register pair contains the load/start address. Lines 3572 and 3574 check the contents of buffers 1 and 2 to make sure that this is the first data block and, if so, stores this address in the buffers (lines 3580 and 3590). If buffers 1 and 2 already hold an address, the computer assumes that the data block was not the first and does not disturb the buffers (line 3576). The HL register pair retains the value as the byte count to that point. (It is also the load/start address for the next data block.) You are now ready to start counting actual bytes of data from the data block (lines 3600 through 3650). This is a basic DJNZ loop which uses the value previously stored in the B register to determine the number of bytes that will be counted. During the counting of these data bytes, the HL register pair is incremented (line 3620) so that it contains a total of all the data bytes read from the tape, plus the first load/start address. (This information will be used to determine the end address of the program.) When this is finished, the program reads and discards the checksum (line 3651) and goes back to process the next data block or to recognize the auto-start header (line 3660).

If you now find the auto-start header (78H), the instruction in line 3430 will jump to line 3670 where buffers 3 and 4 save the total in the HL register pair as the end address. The buffers now contain the load/start address and the end address of the program. The program has already extracted the name and displayed it on the screen and must now read the auto-start address from the tape and store it for display (lines 3670 through lines 3800). Lines 3760 through 3790 load the low byte of the auto-start address into the L register and the high byte into the H register. Line 3800 stores this information on top of the stack until it is time to display it. Since you are now finished with the cassette recorder, line 3801 turns it off. Lines 3802 and 3804 get a C4H tab and print it immediately following the name of the program (actually at the cursor location) so that the data will be displayed directly under the column headings that are on the screen. Extractor now loads the contents of buffer 2 and buffer 1 (note the order—high first) into the A register (lines 3802 through 3840), changes the decimal data to hex, and displays it on the screen under the 1ST BYTE column. Lines 3850 and 3860 load and print a DOH tab. The process repeats for the end address (lines 3870 through 3900) in buffers 4 and 3. Line 3910 gets another DOH tab, and line 3920 prints it. The end address now appears on the screen under the LAST BYTE column, and you are ready to retrieve the auto-start address from the stack (line 3930) and display it in the same manner as the load/start and end addresses (lines 3940 through 3970). Lines 3972 and 3974 load and print a carriage return to return the cursor to the start of the next line.

I brought your attention to the label NEXT which appears in line 3212 and told you we would use it to process multiple files from the same cassette tape. The time has arrived when Extractor will respond to a decision that you must make as to how many programs on the same tape you wish to index. If you want to know about all the programs on a single tape, do nothing, and Extractor will continue processing until the end of the tape. If you are only interested in one or two programs on a tape, simply hold the SHIFT key down during the processing of the program, and the computer will return you to the control of the TRS-80 at the end of the current program. Lines 4070 through 4110 contain the code for these functions. Line 4070 checks the SHIFT key, and if it detects a true, jumps to line 4100. If it does not detect a true, line 4090 will go to NEXT at the end of each program until the end of the tape or until it senses a SHIFT. Line 4100 controls the return to BASIC. If, after exiting to a BASIC READY prompt, you decide to index additional programs, simply type SYSTEM, hit ENTER, respond to the SYSTEM prompt (*?) with /, the address in decimal of your assembly ORG statement, and press ENTER. Extractor will reinitialize and will continue from the point on the tape where it left off. (Note: previously displayed information is lost when Extractor is reinitialized.)

The remainder of the program code is used for the messages and subroutines for error printing, displaying to the screen, changing the decimal data to hex, and displaying the messages and heading.

As indicated earlier, with a few changes to the program, a checksum checking routine can be implemented which will allow the user to check the validity of the data on the tape without reading the program into memory. The formula that Radio Shack uses to compute the checksum is: low byte plus high byte of load/start address (each data block computed separately) plus the absolute hex value of each data byte contained in the data block with any resulting carry discarded. To implement this in assembly-language code, you must first zero the C register:

```
3490    LD    C,OH
```

To add the low and high bytes of the load/start address, ADD the contents of the A register at the time that it contains the low byte to the C register and load the resultant sum into the C register:

```
3520    ADD    A,C
3530    LD     C,A
```

When the high byte is in the A register, the sequence repeats, leaving the sum of the low and high bytes of the load/start address in the C register:

```
3560    ADD    A,C
3570    LD     C,A
```

To complete the formula, the program adds each data byte contained in the data block to the sum in the C register:

```
3630    ADD    A,C
3640    LD     C,A
```

The checksum is now in the C register and must be compared to the checksum recorded at the end of each data block. Some type of indicator is needed to give the results of the comparison:

```
3652         CP      C
3654         JP      NZ,ERR3
.
.
4552 ERR3    LD      HL,MSG7
4554         CALL    ERROR
.
.
5510 MSG7    DEFM    'CHECKSUM ERROR'
5520         NOP
5530         END
```

You can make the above additions without disturbing the timing of the program. They will process checksum errors without loading anything into memory, and you can extract all the pertinent data from a SYSTEM tape and check to see if the data is valid at the same time. There is a single byte addition that you can make to the program that will allow it to load data into memory as the program is being indexed.

```
3610    LD    (HL),A
```

You must enter the above changes at the locations indicated by the program step numbering sequence. If you use different line numbers when you type the program, you must integrate the numbers of the changes into their proper places. Some other modifications that would enhance the program are the addition of a print routine (both parallel and serial) to make the permanent record of the information and an automatic conversion from hex to decimal so that both would be printed.

Program Listing. *Extractor*

```
00100 ;THIS IS THE EXTRACTOR --IT'S PURPOSE IS TO READ AND
00200 ;DISPLAY THE START ADDRESS, END ADDRESS AND AUTO-START
00300 ;ADDRESS FROM CASSETTE TAPES RECORDED WITH RADIO SHACK'S
00400 ;TRS-80 LEVEL II SYSTEMS FORMAT.  IT WILL NOT FUNCTION
00500 ;FOR THOSE CASSETTE TAPES HAVING THEIR OWN LOADERS.  AN
00600 ;EXAMPLE OF THIS TYPE OF CASSETTE IS MICRO-CHESS.
00700 ;
00800 ;
02300 ;
02400 ;
02500 ;NAME AND INSTRUCTIONS ARE LOADED AND DISPLAYED
02510 ;
02520 ;
02530         ORG     4350H
02600 ;
02700 ;
02800         CALL    01C9H           ;CLEAR SCREEN
02810         LD      HL,MSG1
02820         CALL    PRTSTR
02830         LD      HL,MSG2
02840         CALL    PRTSTR
02850         LD      HL,MSG3
02860         CALL    PRTSTR
02870         LD      IY,BUF1
02940         PUSH    AF
02950         PUSH    DE
02960 LOOP1   CALL    2BH             ;KEYBOARD SCAN
02970         OR      A
02980         JR      Z,LOOP1
02990         POP     DE
03000         POP     AF
03040         CALL    01C9H           ;CLEAR SCREEN
03050         LD      HL,MSG4         ;COLUMN HEADING
03060         CALL    PRTSTR          ;PRINT IT
03160 ;
03170 ;
03180 ;TURN ON CASSETTE, READ LEADER AND FIND SYNC (A5).
03190 ;ALSO FIND NAME HEADER (55), DISPLAY NAME.
03200 ;
03210 ;
03212 NEXT    LD      (IY+00H),00H    ;ZERO FOUR BUFFERS
03214         LD      (IY+01H),00H
03216         LD      (IY+02H),00H
03218         LD      (IY+03H),00H
03220         CALL    0212H           ;SELECT DRIVE
03230         CALL    0296H           ;FIND LDR AND SYNC
03240         CALL    0235H           ;READ ONE BYTE
03250         CP      55H
03260         JP      NZ,ERR1         ;NO--PRINT ERROR MESSAGE
03270         LD      B,06H           ;SIX BYTES IN NAME
03280 LOOP2   CALL    0235H           ;READ ONE
03290         CALL    PRINT           ;PRINT IT
03300         DJNZ    LOOP2           ;GET ALL SIX CHARACTERS
03330 ;
03340 ;
03350 ;CHECK TO SEE IF NEXT HEADER IS AUTO-START HEADER (78)
03360 ;OR A DATA HEADER (3C).  IF AUTO-START, JUMP OUT OTHER-
03370 ;WISE READ AND STORE ADDRESS OF 1ST BYTE AND START
03380 ;COUNTING BYTES UNTIL NEXT DATA HEADER.
03390 ;
03400 ;
03410 LOOP4   CALL    0235H           ;READ ONE BYTE
03420         CP      78H
03430         JR      Z,ESCAPE        ;IF AUTO-START GET OUT
03440         CP      3CH
03450         JP      NZ,ERR2         ;NO--PRINT ERROR MESSAGE
03460         CALL    022CH           ;BLINK ASTERICK
03470         CALL    0235H           ;GET BLOCK LENGTH
03480         LD      B,A             ;STORE IT IN B
03500         CALL    0235H           ;GET LO-BYTE OF START ADD
03510         LD      L,A             ;STORE IT IN L
03540         CALL    0235H           ;GET HI-BYTE OF START ADD
03550         LD      H,A             ;STORE IT IN H
03572         LD      A,(IY+00H)      ;GET BUF1
03574         OR      (IY+01H)        ;SEE IF ITS EMPTY
03576         JR      NZ,LOOP3        ;NO--LEAVE IT
03580         LD      (IY+00H),L      ;STORE IT IN BUF1
03590         LD      (IY+01H),H      ;STORE IT IN BUF2
```

```
03600 LOOP3   CALL    0235H           ;READ ONE BYTE
03620         INC     HL              ;BUMP BYTE COUNT
03650         DJNZ    LOOP3           ;UNTIL BLOCK LEN IS ZERO
03651         CALL    0235H           ;READ CHECKSUM
03660         JP      LOOP4           ;GET NEXT DATA BLOCK
03670 ESCAPE  LD      (IY+02H),L      ;SAVE LO-BYTE END ADDRESS
03680         LD      (IY+03H),H      ;SAVE HI-BYTE END ADDRESS
03690 ;
03700 ;
03710 ;GET START AND END ADDRESSES FROM BUFFERS AND AUTO-START
03720 ;FROM HL REGISTER, CHANGE TO HEX AND DISPLAY ON SCREEN.
03740 ;
03750 ;
03760         CALL    0235H           ;READ ONE BYTE
03770         LD      L,A             ;LO-BYTE OF AUTO-START
03780         CALL    0235H           ;READ ONE BYTE
03790         LD      H,A             ;HI-BYTE OF AUTO-START
03800         PUSH    HL              ;SAVE IT
03801         CALL    01F8H           ;TURN OFF CASSETTE
03802         LD      A,00C4H         ;GET A FIVE SPACE TAB
03804         CALL    PRINT           ;PRINT IT
03810         LD      A,(IY+01H)      ;GET HI-BYTE OF START ADD
03820         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03830         LD      A,(IY+00H)      ;GET LO-BYTE OF START ADD
03840         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03850         LD      A,0D0H          ;GET 16 SPACE TAB
03860         CALL    PRINT           ;PRINT IT
03870         LD      A,(IY+03H)      ;GET HI-BYTE OF END ADD
03880         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03890         LD      A,(IY+02H)      ;GET LO-BYTE OF END ADD
03900         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03910         LD      A,0D0H          ;GET 16 SPACE TAB
03920         CALL    PRINT           ;PRINT IT
03930         POP     HL              ;RETRIEVE AUTO-START
03940         LD      A,H             ;GET HI-BYTE AUTO-ST
03950         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03960         LD      A,L             ;GET LO-BYTE AUTO-ST
03970         CALL    HEXER           ;CHANGE FORM AND PRINT IT
03972         LD      A,0DH           ;GET A CARRIAGE RETURN
03974         CALL    PRINT           ;PRINT IT
03980 ;
03990 ;
04000 ;IF SHIFT KEY IS BEING HELD DOWN, THEN GO BACK TO
04010 ;BASIC.  IF NOT THEN GO GET ANOTHER PROGRAM FROM THE
04020 ;SAME TAPE AND CONTINUE TO REPEAT INDEXING FUNCTION
04030 ;UNTIL END OF TAPE OR UNTIL SHIFT IS HELD DOWN.
04040 ;
04050 ;
04070 RESET   LD      A,(3880H)       ;CHECK IF SHIFT IS DOWN
04080         OR      A               ;SET FLAGS
04090         JP      Z,NEXT          ;NO--GET NEXT PROGRAM
04100 BASIC   LD      HL,(40E6H)      ;RESTORE I/O PTR.
04110         JP      0072H           ;JUMP TO BASIC
04160 ;
04170 ;
04180 ;THESE MESSAGES ARE ADDRESSED BY THE HL REGISTER PAIR
04190 ;AND WILL BE PRINTED BY A CALL TO 28A7H.
04200 ;
04210 ;
04220 MSG1    DEFM    '* * * * * * * * * * * * * *  EXTRACTOR  * * * * * * * * * * * * * *'
04230         DEFB    0DH
04240         DEFB    0DH
04250         NOP
04280 MSG2    DEFM    'WRITTEN BY : J. CRUTCHER  SCOTTSDALE, ARIZONA'
04290         DEFB    0DH
04300         DEFB    0DH
04310         NOP
04340 MSG3    DEFM    'TO READ A SYSTEMS FORMAT TAPE, LOAD TAPE AND HIT ENTER:'
04350         DEFB    0DH
04360         NOP
04380 MSG4    DEFM    'NAME     1ST-BYTE          LAST-BYTE         AUTO-START'
04390         DEFB    0DH
04400         DEFB    0DH
04410         NOP
04440 MSG5    DEFM    'NAME HEADER ERROR'
04450         NOP
04480 MSG6    DEFM    'DATA HEADER ERROR'
04490         NOP
04520 ERR1    LD      HL,MSG5
04530         CALL    ERROR
04540 ERR2    LD      HL,MSG6
04550         CALL    ERROR
04560 ERROR   PUSH    AF
```

Program continued

```
04570           PUSH    BC
04580           PUSH    DE
04590           PUSH    HL
04600           CALL    28A7H           ;PRINT ERROR MESSAGE
04610           POP     HL
04620           POP     DE
04630           POP     BC
04640           POP     AF
04650           CALL    01F8H           ;STOP CASSETTE
04660           JP      BASIC
04670           NOP
04700 BUF1      NOP
04710 BUF2      NOP
04720 BUF3      NOP
04730 BUF4      NOP
04770 ;
04780 ;
04790 ;THE FOLLOWING SUBROUTINES WILL DISPLAY THE A REGISTER
04800 ;AT THE CURSOR LOCATION AND CHANGE THE DECIMAL DATA
04810 ;TO HEX FOR DISPLAY.
04820 ;
04830 ;
04840 PRINT     PUSH    AF
04850           PUSH    DE
04860           PUSH    IY
04870           CALL    032AH           ;DISPLAY A REGISTER
04880           POP     IY
04890           POP     DE
04900           POP     AF
04910           RET
04950 HEXER     PUSH    AF
04960           LD      C,A
04970           SRL     A
04980           SRL     A
04990           SRL     A
05000           SRL     A
05010           CALL    COMP
05020           LD      A,C
05030           AND     0FH
05040           CALL    COMP
05050           POP     AF
05060           RET
05070 COMP      CP      0AH
05080           JR      C,LOOP
05090           ADD     A,07H
05100 LOOP      ADD     A,30H
05110           CALL    PRINT
05120           RET
05280 ;
05290 ;
05300 ;THIS SUBROUTINE WILL DISPLAY THE MESSAGE POINTED TO BY
05310 ;THE HL REGISTER.  MESSAGES MUST BE TERMINATED WITH ZERO.
05320 ;
05330 ;
05340 PRTSTR    PUSH    AF
05350           PUSH    BC
05360           PUSH    DE
05370           PUSH    HL
05380           PUSH    IX
05390           PUSH    IY
05400           CALL    28A7H           ;PRINT MESSAGE
05410           POP     IY
05420           POP     IX
05430           POP     HL
05440           POP     DE
05450           POP     BC
05460           POP     AF
05470           RET
05510           END
```

Page Print Your Listings

by A. P. Gitt

If you have a TRS-80 Model I with at least one disk drive, printing out information formatted by page is relatively simple thanks to the TRSDOS LINE INPUT # command. The Printer Paging program (see Program Listing) does the following:

1) Prints the title of your choice at the top of each page
2) Provides a left-hand margin of any width while adding the linefeed/carriage returns (LF/CR) which maintain the margin if a line has more characters than a normal printer line
3) Automatically numbers the pages, starting with any page number you choose
4) Counts and displays the number of program lines, as well as the actual number of lines printed.

ASCII Saving

To use this program for page formatting of BASIC program listings, before you run it, you must store the information to be printed on your disk in ASCII format. This is done by appending ,A to your file name when you save your BASIC program to disk. Two examples of how to do this are shown below:

```
Example 1)  SAVE"PAGEPRNT/TXT:1",A
Example 2)  SAVE"PROGRAM/TXT",A
```

Example 1 saves the program Pageprnt in ASCII format (,A) with the extension TXT on disk drive number 1 (:1). The TRSDOS manual recommends using the TXT extension when you save text or programs in ASCII. Example 2 saves Program with the extension TXT on disk drive zero in ASCII format.

Keep in mind that saving files in ASCII format uses more disk space on a disk than saving the file in compressed format. Remember this if you are working with a nearly full disk, in order to avoid the DISK FULL message and the resulting time it takes to KILL space or slide in a new disk. Plan ahead! You can check the number of granules available by using the FREE command before you embark on an ASCII save effort.

Using the Program

To use the Printer Paging program, simply go into Disk BASIC and load the program. The program will prompt you to enter the following information:

1) Name of program (as stored on disk) to be printed
2) Title to be printed at the top of each page
3) Number of the first page to be printed
4) Number of spaces to tab for the left-hand margin.

If you enter the above information correctly, your printer will take off, followed shortly by the whirl of your disk file. The resulting pages will pour from your printer, neatly formatted and numbered. The TRS-80 and disk drive are fast enough that there is no noticeable slowdown in printer throughput, which is 80 characters per second for my Epson MX-80 printer.

How it Works

There have been several programs and articles written on page formatting of BASIC listings. One of the most recent appeared in the December 1980 issue of *The TRS-80 Microcomputer News*, which is published monthly by Radio Shack. The concept of my Printer Paging program is based on that program by Richard Halloran.

The basic function of my Printer Paging program is to read ASCII recorded program lines from the disk and LPRINT them. The process gets more complicated when you require the program to print page titles and page numbers on each page, but doing this is still pretty straightforward. What makes execution difficult is the need for the program to maintain the left-hand margin, while printing lines which contain more characters than the total number the printer can print on any line—in my case, 80 characters per line. This means that if an ASCII line read from the disk has more than 80 characters, most line printers will add a line feed/carriage return (LF/CR) after printing 80 characters on a line. But, the printer does not know that you have tabbed the line to print the additional line flush with the left-hand margin. When the printer adds the LF/CR, the tab or left-hand margin is lost until the next program line is read from disk.

You can avoid this problem by making the length of all your program lines less than 80 characters, then subtracting the number of spaces you want as a left-hand margin, or by setting the left-hand margin to zero. Neither of these solutions has the flexibility needed for a complete page printer program. By determining the length of each program line and testing to see if its length is equal to or greater than 80 minus the number of spaces you want as the left-hand margin, you can divide the original long program line into a series of strings which equal 80 characters minus the number of spaces desired for the left-hand margin (80 − TB).

The subject of long lines brings up one of the oddities of the TRS-80. The Model I Level II manual tells you that a program line may have up to 255 characters. That is correct, but if you SAVE a program to disk or CSAVE it to cassette, and then LOAD or CLOAD it back into the computer, you get back only 248 characters. If you look on the disk by LISTing the program in

TRSDOS, you can see all 255 characters but you can't load them back into BASIC. If you like long program lines, you can save time debugging programs by remembering that the useful limit is 248 characters.

The Printer Paging program consists of 60 lines of code. Table 1 lists the program variables. Table 2 lists the various program routines by line number. Line 50 of the program clears 1000 bytes of string storage space. Lines 60 and 70 contain an opening title which will appear on the screen for several seconds after you type RUN. I usually include this opening title with all my BASIC programs. You can, of course, delete lines 60 and 70.

Variable	Description
A	Flag for long lines
N	Number of lines printed
N$	Name of program read from disk
P	Initial page number counter value
Q	Starting page number
R	Number of program lines printed
R$	Program line read from disk in ASCII
S$	Temporary long line string
S1$	Temporary long line string
S2$	Temporary long line string
U$	Temporary long line string
U1$	Temporary long line string
U2$	Temporary long line string
X	Number of program lines printed on last page
Z	Number of line feeds before printing last page number
Z$	Temporary working string variable

Table 1. *Program variables*

Line Number	Description
10–90	Program initialization
100–130	Input parameters
140	Initializes printer line counter and lines per page
150–160	Print first page title
170	Opens Disk file
180	Tests flags for long lines
190	Checks number of lines printed on current page
200	Prints page number and form-feeds to end of page
210–220	Print title at top of next page
230–260	Test for return to long lines routines
270	Checks for end of file
280	Reads line from disk and increments program line counter
290–480	Test for long lines and divide them into 80-character strings

Table continued

490	Increments lines printed counter and displays number of lines printed
500	Displays number of program lines printed
510	Returns to start next line
520	Closes disk file
530–550	Line-feed to near bottom of page and print page number and form-feed
560–600	Return to print another copy, start a new file listing, or end program

Table 2. *Line number description*

Lines 80 and 90 remind you to append files SAVEd to disk with ,A if you intend to use them with this program. Lines 100 through 130 input the parameters required to recall the program to be printed from disk and set up your page format. Line 140 sets the number of lines per page in the TRS-80 to 67 lines (POKE 16424,67), sets the printer line counter to zero (POKE 16425,0), and initializes the counter values.

Lines 150 and 160 determine the tab setting to center the title and print the title on the first page. Line 170 opens the disk file that contains the program to be printed. Line 180 and lines 230 through 260 test the status of the A flag for long lines. Line 270 checks to see if the last line of data has been read from the disk file. If it has, then the program jumps to line 520 and closes the file. Line 190 counts the number of lines on the current page. If fewer than 62 have been printed, it tells the program to continue and prints the next line. If the printer has printed 62 lines, line 200 increments the page counter, prints the page number at the bottom of the page, advances the form to the beginning of the next page, and resets the line counter to zero. Lines 210 and 220 print the title at the top of the next page. Line 280 uses the LINE INPUT # command to read each line of the program from the disk. Lines 290 through 470 test for lines longer than (80 – TB) characters. Line 510 returns to the start of the disk read/print loop to get another line.

Line 520 closes the disk file. Lines 490 and 500 display the number of lines printed and the total program lines. Lines 530 through 560 determine the amount of blank lines left on the last page, line-feed to near the end of the last page, print the last page number, and form-feed to the end of the last page. Lines 320 and 330 ask if you want to print another copy, print a new file, or end the program.

If your printer has a line length other than 80 characters, change the 80 in lines 290, 300, 330, 350, 360, 390, 420, and 430 to the number of characters in a standard line for your printer. You must also change the 81 in lines 310, 380, and 450 to the length of your line plus one.

This program was written for a TRS-80 Model I. If you wish to run it on a

Model III, change the POKE 14312,12 statements in lines 200 and 550 to OUT 251,12. These changes are needed since the printer is a memory-mapped I/O device in a Model I, and a PORT I/O device in a Model III.

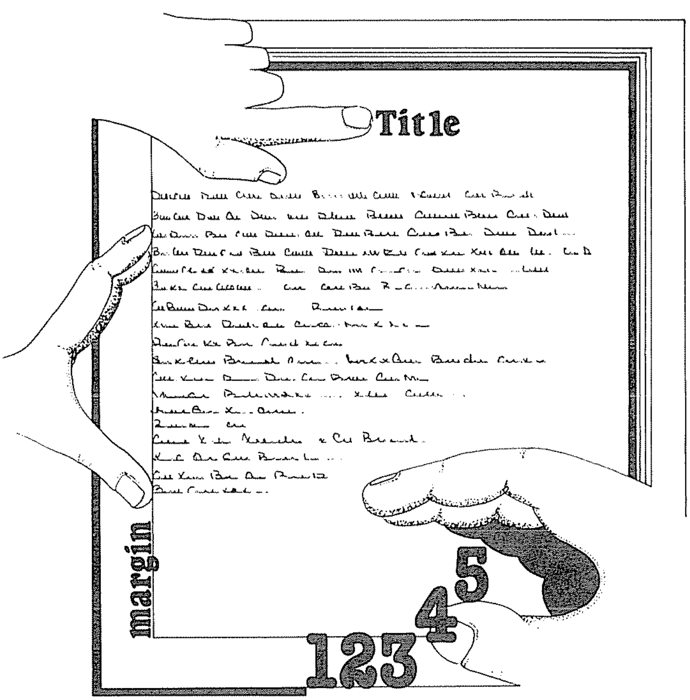
Program Listing. *Printer Paging program*

```
 10 :
    ' ******** PRINTER PAGING PROGRAM ********
 20 :
    ' WRITTEN BY A.P. GITT
 30 :
    ' DISC FILE NAME: PAGEPRNT/BAS
 40 :
    ' LAST REVISION: 04/15/81
 50 CLEAR 1000
 60 CLS :
    PRINT @454, CHR$(23);"PRINTER PAGING ROUTINE"
 70 FOR X = 1 TO 400:
     NEXT :
    CLS
 80 PRINT :
    PRINT "DISK FILES PRINTED WITH THIS PROGRAM MUST BE SAVED TO"
 90 PRINT "DISK IN ASCII (APPEND '     ,A     ' TO FILE SPEC NAME)"
100 PRINT :
    PRINT "ENTER DISK FILE SPEC NAME OF PROGRAM TO BE PRINTED":
    INPUT N$
110 PRINT "ENTER PAGE TITLE TO BE PRINTED AT TOP OF EACH PAGE":
    INPUT T$
120 INPUT "ENTER STARTING PAGE NUMBER";Q:
    P = Q - 1
130 INPUT "ENTER LEFT HAND MARGIN TAB (NO. OF SPACES)";TB
140 POKE 16424,67:
    POKE 16425,0:
    P = 0:
    R = 0:
    N = 0
150 LPRINT :
    LPRINT :
    LPRINT :
    LPRINT :
    LPRINT :
    LPRINT TAB(40 - ( LEN(T$) / 2));T$
160 LPRINT :
    LPRINT
170 OPEN "I",1,N$
180 IF A = 1 OR A = 2 OR A = 3
     THEN
      GOTO 190
190 IF PEEK(16425) = 62
     THEN
      GOTO 200 :
     ELSE
      GOTO 230
200 P = P + 1:
    LPRINT :
    LPRINT TAB(39);P:
    POKE 14312,12:
    POKE 16425,0
210 LPRINT :
    LPRINT :
    LPRINT :
    LPRINT :
    LPRINT :
    LPRINT TAB(40 - ( LEN(T$) / 2));T$
220 LPRINT :
    LPRINT
230 IF A = 1
     THEN
      GOTO 330
240 IF A = 2
     THEN
      GOTO 420
250 IF A = 3
```

```
     THEN
      GOTO 470
260 IF A = 4
     THEN
      GOTO 420
270 IF EOF (1)
     THEN
      520
280 LINE INPUT #1,R$:
    R = R + 1
290 IF LEN(R$) > (80 - TB)
     THEN
      GOTO 300 :
     ELSE
      480
300 A = 1:
    S$ = LEFT$(R$,(80 - TB))
310 U$ = MID$(R$,(81 - TB))
320 LPRINT TAB(TB)S$:
    GOTO 490
330 IF LEN(U$) > (80 - TB)
     THEN
      GOTO 350 :
     ELSE
      340
340 LPRINT TAB(TB)U$:
    A = 0:
    GOTO 490
350 S1$ = LEFT$(U$,(80 - TB))
360 IF LEN(S1$) > = (80 - TB)
     THEN
      GOTO 380
370 LPRINT TAB(TB)S1$:
    A = 0:
    GOTO 490
380 U1$ = MID$(U$,(81 - TB))
390 IF LEN(U1$) > = (80 - TB)
     THEN
      GOTO 410 :
     ELSE
      400
400 LPRINT TAB(TB)S1$:
    A = 4:
    GOTO 490
410 A = 2:
    LPRINT TAB(TB)S1$:
    GOTO 490
420 S2$ = LEFT$(U1$,(80 - TB))
430 IF LEN(S2$) > = (80 - TB)
     THEN
      GOTO 450
440 LPRINT TAB(TB)S2$:
    A = 0:
    GOTO 490
450 U2$ = MID$(U1$,(81 - TB))
460 A = 3:
    LPRINT TAB(TB)S2$:
    GOTO 490
470 LPRINT TAB(TB)U2$:
    A = 0:
    GOTO 490
480 LPRINT TAB(TB)R$
490 N = N + 1:
    PRINT @768,"TOTAL NUMBER OF LINES PRINTED: ";N;
500 PRINT @832,"NUMBER OF PROGRAM LINES: ";R;
510 GOTO 180
520 CLOSE
530 X = PEEK(16425):
    Z = 62 - X:
    FOR Y = 1 TO Z:
     LPRINT CHR$(32):
```

Program continued

```
     NEXT
540 LPRINT :
    LPRINT TAB(39);P + 1
550 POKE 14312,12
560 PRINT :
    INPUT "DO YOU WANT TO PRINT ANOTHER COPY?  (Y)ES OR (N)O";Z$
570 IF Z$ = "Y"
     THEN
      GOTO 140
580 CLS :
    PRINT :
    INPUT "DO YOU WANT TO PRINT A NEW FILE? (Y)ES OR (N)O";Z$
590 IF Z$ = "Y"
     THEN
      GOTO 60 :
     ELSE
      CLS
600 END
```



Let Your TRS-80 Do the Typing

by Susan R. Nelson

Here is a utility to save cassette-based TRS-80 owners time, tape, and typing when handling input/output (I/O) data files such as accounting files, inventories, mailing lists, or scientific computational files resulting from a lengthy Fourier or matrix calculation. This utility program generates and types those data files for you via the DATA statement.

By using the DATA statement in your BASIC program to store data files, you can save cassette I/O time as well as saving tape. Since DATA statements are a part of the BASIC program, the data file is read from cassette when the program is CLOADed. The DATA statements, now resident in memory, are almost instantly available to the program via the READ command. Instead of using the slow INPUT# to read the file from tape or rewinding the tape for another lengthy read, the RESTORE command backs up the DATA pointer. You also save cassette tape when you use DATA statements. To save newly created DATA statements, just CSAVE at the end of the computer session.

This utility was written to generate and type DATA statements, and insert them into your BASIC program. Let the TRS-80 figure out the line number, type DATA, and type the data in the proper fields in the correct format.

I became interested in this problem after reading and doing some experimenting with the Household Accountant program written by David Andersen (see the February 1980 issue of *80 Microcomputing*). I compared the use of DATA statements versus using a cassette file for the check file. Using the INPUT#/PRINT# statement, the cassette file not only took longer to do the I/O but also used a lot of tape; it took 10 minutes to INPUT#/PRINT# the check file and used almost an entire side of a 20-minute cassette tape. In addition, it took 3 1/2 minutes to CLOAD the INPUT#/OUTPUT# program. Using the DATA statement file, a READ took 15 seconds while the CLOAD/CSAVE took four minutes. The DATA statement program used only one-third of a 20-minute cassette tape. This experiment convinced me that when dealing with data on tape, DATA statements were the way to go, especially for cassette users. There was only one problem. I didn't particularly like typing in all those DATA statements, with the line numbers, the word DATA, and then all the numbers. The numbers were bad enough. Occasionally, I would type an incorrect line number, or forget which number went in what field, or I would leave DATA out. I decided to let the computer do the typing. As a result, I wrote this utility to make it easier and faster for cassette users who handle I/O files. Just answer an input prompt as to what numbers are to be input, and then let the computer build the DATA statements and type them in the program.

Building the DATA Statement

Building the DATA statement was the easiest part of the program (see the Program Listing). Lines 3110–3194 and lines 3990–3994 in the Program Listing generate the DATA statement into array Z( ). The DATA statement is Q bytes long. Figures 1, 2, and 3 show some sample DATA statements and how they should look to a BASIC program. Figure 1 shows the memory layout before the DATA statements are inserted. In DATA statement 10002 (see Figure 2), the next address comes first, followed by the least significant

Address	BASIC Line	
20496	10000 REM START OF DATA SECTION	DO NOT REMOVE THIS REMARK
20552	19999 DATA -1	
20561	0 0 (END-OF-BASIC PROGRAM)	
20563	VARIABLE STORAGE AREA	
20563	ARRAY STORAGE AREA	
20563	FREE MEMORY	

Address	Decoded BASIC	BASIC
20496	72 80 16 39 147 32 83 84 65 82 84 32 79 70 32 68	10000 REM START OF
20512	65 84 65 32 83 69 67 84 73 79 78 32 32 32 68 79	DATA SECTION
20528	32 78 79 84 32 82 69 77 79 86 69 32 84 72 73 83	DO NOT RE-
20544	32 82 69 77 65 82 75 0	MOVE THIS REMARK
20552	81 80 31 78 136 32 45 49 0	19999 DATA -1
20561	0 0	END-OF-BASIC
20563	VARIABLE STORAGE, ARRAY, FREE MEMORY AREAS	

Figure 1. *Initial memory layout before DATA statements are inserted*

byte (90) and the most significant byte (80). The line number is next, also followed by the least significant byte (18) and the most significant byte (39). Now the data token (136) is listed, a blank (32) and then the numbers (in ASCII) follow, each separated by a comma (44), and finally a 0 to end the BASIC statement. In building the DATA statement I had to determine how long the DATA statement was before figuring out the new next line (NX = NP + Q), line 3192. The line number used (LN) is initialized in lines 3019–3050 and is put in the DATA statement in line 3130. Figure 3 shows the memory layout after line 10004 is inserted.

Inserting the DATA Statement in the BASIC Program

Inserting the newly generated DATA statement into the BASIC program was the next problem to be solved. Figure 4 summarizes how the utility does this. Figures 1, 2, and 3 describe where new DATA statements are added to the BASIC program. Originally the check file data was inserted based on check number (line number = check number + 10000). This method did not work out. As I added more checks, the utility became slower and slower. To speed up the process, I made the line number arbitrary and always inserted the new DATA statement just before the last data line (19999 DATA −1). Doing the insertion this way reduced the number of bytes that had to be pushed down in memory, and required only one next address to be redefined, in lines 3200–3230 of the Program Listing. The first time through subroutine 3000 is the slowest because the program is finding the last data line (19999 −NP), and the end of BASIC (NE). It is also initializing the data line count (LN) to the first number greater than 10000 but less than 19999, in lines 3019–3050 of the Program Listing.

Problems and Solutions

When it came time to debug the program. I typed the code mentioned above for building and inserting the DATA statement into the BASIC program, and immediately tried it. It didn't work. The program seemed to die. What I finally figured out was that I had written over the variable storage area with the new DATA statements. Figure 1 shows where the variable storage, array storage, and free memory areas are usually located, following the end of the BASIC program. The solution to this problem was the subroutine at line 3700. This subroutine redefined where in memory the variable storage, arrays, and free memory areas are. For this demonstration, the pointers defining these areas, 16633 and 16634 for variable storage, 16635 and 16636 for arrays, and 16637 and 16638 for free memory, are bumped by 30*256 = 7680 memory locations. How much these areas have to be moved down in memory depends on the size of your TRS-80 and the size of your data file.

Address	BASIC Line	
20496	10000 REM START OF DATA SECTION	DO NOT REMOVE THIS REMARK
20552	10002 DATA 4004,1,67,7	
20570	19999 DATA −1	
20579	0 0 (END-OF-BASIC PROGRAM)	
28243	VARIABLE STORAGE AREA	
28346	ARRAY STORAGE AREA	
28678	FREE MEMORY	

Address	Decoded BASIC		BASIC
20496	72 80 16 39 147 32 83 84 65 82 84 32 79 70 32 68	10000	REM START OF
20512	65 84 65 32 83 69 67 84 73 79 78 32 32 32 68 79		DATA SECTION
20528	32 78 79 84 32 82 69 77 79 86 69 32 84 72 73 83		DO NOT REMOVE
20544	32 82 69 77 65 82 75 0		THIS REMARK
20552	90 80 18 39 136 32 52 48 48 52 44 49 44 54 55 44 55 0	10002	DATA 4004,1,67,7
20570	99 80 31 78 136 32 45 49 0	19999	DATA −1
20579	0 0		END-OF-BASIC
28243	VARIABLE STORAGE		
28346	ARRAY STORAGE AREA		
28678	FREE MEMORY		

Figure 2. *Memory layout after line 10002 is inserted*

Address	BASIC Line	
20496	10000 REM START OF DATA SECTION	DO NOT REMOVE THIS REMARK
20552	10002 DATA 4004,1,67,7	
20570	10004 DATA 4015,1,56.78,13	
20592	19999 DATA −1	
20601	0 0 (END-OF-BASIC PROGRAM)	
28243	VARIABLE STORAGE AREA	
28346	ARRAY STORAGE AREA	
28678	FREE MEMORY	

Address	Decoded BASIC	BASIC
20496	72 80 16 39 147 32 83 84 65 82 84 32 79 70 32 68	10000 REM START OF
20512	65 84 65 32 83 69 67 84 73 79 78 32 32 32 68 79	DATA SECTION
20528	32 78 79 84 32 82 69 77 79 86 69 32 84 72 73 83	DO NOT REMOVE
20544	32 82 69 77 65 82 75 0	THIS REMARK
20552	90 80 18 39 136 32 52 48 48 52 44 49 44 54 55 44 55 0	10002 DATA 4004,1,67,7
20570	112 80 20 39 136 32 52 48 49 53 44 49 44 53 54 46 55 56 44 49 51 0	10004 DATA 4015,1,56.78,13
20592	121 80 31 78 136 32 45 49 0	19999 DATA −1
20601	0 0	END-OF-BASIC
28243	VARIABLE STORAGE	
28346	ARRAY STORAGE AREA	
28678	FREE MEMORY	

Figure 3. *Memory layout after line 10004 is inserted*

1. Push variable storage, array, and free memory areas down -- RUN 3700 and press ENTER.
2. Remind user to run 3700 first after CLOADing.
3. Remind user to run 3800 before CSAVEing.
4. First time initialize line count (LN), find last line 19999 (NP), and end-of-BASIC (NE).
5. Generate new DATA statement Q bytes long into array Z.
6. Push down last line to end of BASIC (NP-NE) plus Q bytes.
7. Redefine NEXT address in last line and end to NEXT + Q.
8. POKE new line,Z(1 – Q), in at NP to NP + Q – Q.
9. Bump NP by Q, NE by Q, and return for next input.

Figure 4. *Program outline*

```
2910 INPUT"CHECK NO.,MONTH, AMOUNT, CATEGORY";C,M$,A,C1
2919 IFC<0THEN2950
2930 PRINT"CHECK NO.:";C;" DATE:";M$;" AMOUNT:";A;" CATEGORY:";C1
2940 GOSUB3000:REM BUILD DATA STATEMENT AND INSERT

3170 C$ = M$:GOSUB3990:GOSUB3992
```

Figure 5. *String input sample*

```
0 CLS:PRINT"TYPE IN AFTER MERGING-POKE16549,"PEEK(16549)":POKE16548"
PEEK(16548):E = 17129
1 S = E:E = PEEK(S + 1)*256 + PEEK(S):IFE>0GOTO1
2 POKE 16549,INT(S/256):POKE16548,S – INT(S/256)*256:END
```

Figure 6. *Append subroutine*

```
Before RUN
10000 REM START OF DATA SECTION DO NOT REMOVE THIS REMARK
19999 DATA -1

>RUN (enter)

********************************************************************************
RUN 3700 FIRST!!!RUN 3700 FIRST!!!
! IF INSERTING NEW DATA STATEMENTS FOR FIRST TIME SINCE CLOAD
HIT BREAK, RUN 3700, THEN RUN IF HAVEN'T ALREADY
```

```
BEFORE CSAVE!! RUN 3800 AND POKE AS INSTRUCTED

**************************************************************************

ENTER DATA OR -1,-1,-1,-1 TO STOP INPUT
CHECK NO.,DATE (1-12),AMOUNT,CATEGORY? (BREAK)

>RUN 3700 (enter)
>RUN (enter)

**************************************************************************

RUN 3700 FIRST!!! RUN 3700 FIRST!!!
! IF INSERTING NEW DATA STATEMENTS FOR FIRST TIME SINCE CLOAD
HIT BREAK, RUN 3700, THEN RUN IF HAVEN'T ALREADY

BEFORE CSAVE!! RUN 3800 AND POKE AS INSTRUCTED

**************************************************************************

ENTER DATA OR -1,-1,-1,-1 TO STOP INPUT
CHECK NO.,DATE (1-12),AMOUNT,CATEGORY? 4004,1,67,7 (enter)
ENTER DATA OR -1,-1,-1,-1 TO STOP INPUT
CHECK NO.,DATE (1-12),AMOUNT,CATEGORY? 4015,1,56.78,13 (enter)
ENTER DATA OR -1,-1,-1,-1 TO STOP INPUT
CHECK NO.,DATE (1-12),AMOUNT,CATEGORY? -1,-1,-1,-1 (enter)
IF YOU ARE READY TO CSAVE...FIRST RUN 3800 !!!!
ELSE EDIT OR CONTINUE RUNNING PROGRAM

>RUN 3800 (enter)
BEFORE CSAVE POKE 16633,123:POKE 16634,80

NOW CSAVE
AFTER CSAVE POKE 16633,83:POKE 16634,110

READY
>POKE 16633,123:POKE16634,80 (enter)
>CSAVE"1" (enter)
>READY
>POKE 16633,83:POKE16634,110 (enter)
>LIST 10000-19999

10000 REM START OF DATA SECTION   DO NOT REMOVE THIS REMARK
10002 DATA 4004,1,67,7
10004 DATA 4015,1,56.78,13
19999 DATA -1
>RUN (enter)
```

Figure 7. *Sample run*

The only other problem occurred when I did a CSAVE of the file. The program saved past the number of tape revolutions I had CLOADed it before. I finally turned the TRS-80 off and on, and reloaded the program back in almost the same number of revolutions as the original CLOAD. The subroutine at line 3800 solved this problem. The CSAVE procedure saves from the beginning of BASIC (pointer 16648, 16649) to the variable storage area (pointer 16633, 16634). Subroutine 3800 tells the user what to POKE into the variable storage pointer before and after the CSAVE. Once the CSAVE is done, if you are adding more data, POKE as directed by subroutine 3800. This moves the variable storage back down in memory, keeping it safe from the new DATA statements.

Other Programming Considerations

The data in the DATA statements could have been edited for errors before being put into DATA statements and inserted into the BASIC program. For instance, the month could be tested to see that 0 < month < 12. If not, the program could return to the input prompt. Figure 5 shows an example of using a string variable as the input for the month versus an integer. To do this line 3170 has to be modified. Data going into the file does not have to be input; instead the computer can generate it. If you want the program to do some intermediate mathematical calculation that might be used some time later, just send these numbers through the utility and CSAVE the file at the end of the job. Repetitive data could be entered once, and the TRS-80 would type those statements over and over.

This utility was written so it could be appended to any program or could be used as a stand-alone program. If used as a stand-alone program, once the data is generated and typed, it can be appended to any program needing that particular data file. I have used the append procedure, Figure 6, presented by Alan R. Moyer in "Super Graphics," *80 Microcomputing*, October 1980. Just type the BASIC lines in Figure 6 into the program to which you are appending the data or utility. Type RUN and press ENTER. CLOAD the appending file. Now POKE the beginning-of-BASIC pointer (16648, 16649) with the original pointer as directed by the subroutine in Figure 6.

Program Use

Initialization: If you are going to do DATA statement generation, type RUN 3700 and press ENTER.

Data entry: Now type RUN and press ENTER. A question asking for input, the input prompt, now comes up. Enter your data. The first time through the program takes a little more time as the utility is looking for the last line, line 19999 (NP), and the end (NE). The utility is also initializing the starting line number. REMark 10000 and line 19999 must not be taken out of the program as they tell the utility where the data begins and where it ends. The program will not work without REMark 10000 and line 19999.

CSAVE: Typing in – 1, – 1, – 1, – 1 and pressing ENTER tells the program you are finished with data generation. A reminder to RUN 3800 before the CSAVE will now come on the screen. Make a note of what to POKE before and after CSAVE. RUN 3800 and press ENTER anytime before CSAVE, and after running or editing. Figure 7 shows a sample run.

Program Listing. *The DATA statement*

```
   0 REM ************ DATA INSERT UTILITY ******************
   1 REM               SUSAN R. NELSON
   2 REM               3114 KINGS DR.
   3 REM               PANAMA CITY,FLA. 32405
  10 DIM Z(80)
  19 REM  GO GET INPUT AND UNPACK INTO DATA STATEMENTS
  20 GOSUB 2900
  30 END
2899 REM ************DATA INSERT SECTION*******************
2900 REM  DATA INSERT SECTION SUSAN R. NELSON 6/17/81
2902 PRINT "*****************************************************
     *"
2903 PRINT "RUN 3700 FIRST!!! RUN 3700 FIRST!!!"
2904 PRINT "! IF INSERTING NEW DATA STATEMENTS FOR FIRST TIME SINCE C
     LOAD"
2905 PRINT "HIT BREAK, RUN 3700, THEN RUN IF HAVEN'T ALREADY"
2906 PRINT :
     PRINT "BEFORE CSAVE!! RUN 3800 AND POKE AS INSTRUCTED"
2907 PRINT "*****************************************************
     *"
2909 PRINT "ENTER DATA OR -1,-1,-1,-1 TO STOP INPUT"
2910 INPUT "CHECK NO.,DATE(1-12),AMOUNT,CATEGORY";C,M,A,C1
2920 IF C < 0
      THEN
       2950
2930 PRINT "CHECK NO.:";C;" DATE:";M;" AMOUNT:";A;" CATEGORY:";C1
2940 GOSUB 3000:
     REM  BUILD DATA STATEMENT AND INSERT
2945 GOTO 2909
2950 PRINT "IF YOU ARE READY TO CSAVE...FIRST RUN 3800 !!!!"
2960 PRINT "ELSE EDIT OR CONTINUE RUNNING PROGRAM"
2970 RETURN
3000 REM   **********  DATA STATEMENT INSERT ROUTINE  **********
3001 REM               SUSAN R. NELSON  6/17/81
3010 IF NP < > 0 GOTO 3100
3019 REM  INITIALIZE LINE COUNT,FIND LAST, AND END   LN,NP,NE
3020 NX = 17129:
     LL = 0
3030 NP = NX:
     LN = LL:
     NX = PEEK(NP) + PEEK(NP + 1) * 256
3040 LL = PEEK(NP + 2) + PEEK(NP + 3) * 256:
     IF LL < > 19999 GOTO 3030
3045 REM  NP-LAST LINE ADDRESS (19999 DATA -1)
3046 REM  LN-LAST LINE NUMBER BEFORE 19999 >=10000  <19999
3047 REM  NE-END OF BASIC PROGRAM
3050 NE = NX + 1
3100 REM  START DATA INSERT HERE
3110 REM  *****  GENERATE NEW DATA LINE Q BYTES LONG IN Z() *****
3120 LN = LN + 2
3129 REM         LINE NUMBER-LN,  LSB,MSB INTO Z(2),Z(4)
3130 N = INT(LN / 256):
     L = LN - INT(N * 256):
     Z(3) = L:
     Z(4) = N
3140 Z(5) = 136:
     REM     DATA TOKEN
3150 Z(6) = 32:
     REM      BLANK
3159 REM        BUILD DATA FIELDS
3160 Q = 6:
     C$ = STR$(C):
     GOSUB 3990:
     GOSUB 3992
3170 C$ = STR$(M):
     GOSUB 3990:
     GOSUB 3992
3180 C$ = STR$(A):
```

```
     GOSUB 3990:
     GOSUB 3992
3190 C$ = STR$(C1):
     GOSUB 3990:
     GOSUB 3994
3191 REM        PUT NEXT ADDRESS INTO Z(1),Z(2) =NP+Q
3192 NX = NP + Q:
     N = INT(NX / 256):
     L = NX - INT(N * 256):
     Z(1) = L:
     Z(2) = N
3194 REM  *****   Z(1)-Z(Q) CONTAINS NEW DATA LINE    *****
3200 REM  *****   PUSH DOWN LAST LINE-END Q BYTES     *****
3210 FOR L = NE TO NP STEP - 1:
      POKE (L + Q), PEEK(L):
      NEXT L
3220 REM  *****   REDEFINE NP,NE TO  NP+Q,   NE+Q     *****
3222 NX = NP + Q
3224 NN = NX:
     NX = PEEK(NN + 1) * 256 + PEEK(NN):
3230 IF NX > 0
      THEN
       NX = NX + Q:
       N = INT(NX / 256):
       L = NX - INT(N * 256):
       POKE NN + 1,N:
       POKE NN,L:
       GOTO 3224
3250 REM  POKE NEW DATA LINE Z() INTO NP TO NP+Q-1
3260 FOR L = 1 TO Q:
      POKE NP + L - 1,Z(L):
      NEXT L
3270 NP = NP + Q
3280 NE = NE + Q
3290 RETURN
3700 REM  *****  POINTER PUSH DOWN  *****
3710 REM          RUN 3700 FIRST!!
3720 REM  POKE VARIABLE PTR.,ARRAY PTR.,FREE MEM. PTR. DOWN
3730 POKE 16638, PEEK(16638) + 30
3740 POKE 16636, PEEK(16636) + 30
3750 POKE 16634, PEEK(16634) + 30
3760 END
3800 REM  *****  VAR.PTR PULL BACK,CSAVE!!,&PUSH BACK DOWN  *****
3802 REM   SAVE VAR.PTR., FIND END OF BASIC PROGRAM
3804 REM   TELL USER TO POKE 16633,16634 WITH END+2 BEFORE CSAVE!
3806 REM   TELL USER TO POKE 16633,16634 WITH SAVED VAR.PTR.
3810 LS = PEEK(16633):
     MS = PEEK(16634)
3820 NX = 17129
3830 NP = NX:
     NX = PEEK(NP) + PEEK(NP + 1) * 256:
     IF NX > 0
      THEN
       3830
3840 NP = NP + 2:
     N = INT(NP / 256):
     L = NP - INT(N * 256)
3850 PRINT "BEFORE CSAVE POKE 16633,";L;":POKE 16634,";N
3851 PRINT
3860 PRINT "NOW CSAVE"
3870 PRINT "AFTER CSAVE POKE 16633,";LS;":POKE 16634,";MS
3875 NP = 0:
     REM  ZERO OUT FOR INITIALIZATION IN SUB3000
3880 END
3989 ! DATA CONVERSION ROUTINE TO ASCII CHARACTERS
3990 FOR L = 2 TO LEN(C$):
      Q = Q + 1:
      T$ = MID$(C$,L,1):
      Z(Q) = ASC(T$):
      NEXT L:
```

Program continued

```
     RETURN
3992 Q = Q + 1:
     Z(Q) = 44:
     RETURN :
     !COMMA
3994 Q = Q + 1:
     Z(Q) = Ø:
     RETURN :
      !Ø
 9998 REM  DO NOT REMOVE LINE 1ØØØØ AND LINE 19999 USED BY SUB3ØØØ
 9999 REM ****************THANK YOU******************************
1ØØØØ REM  START OF DATA SECTION   DO NOT REMOVE THIS REMARK
19999 DATA -1
```

APPENDIX

Appendix A

Appendix B

APPENDIX A

BASIC Program Listings

Debugging someone else's mistakes is no fun. In a business environment, where programs are continuously updated and programmers come and go, well-commented and structured programs are a must. Indeed, it behooves any serious programmer to learn structured technique.

The BASIC language has no inherent structure. Most interpreters allow remark lines and some are capable of ignoring unnecessary spacing, but BASIC is still more "Beginner's Instruction Code" than "All-purpose."

The listings in this encyclopedia are an attempt at formatting the TRS-80 BASICs. We think it makes them easier to read, easier to trace, and less imposing when it comes time to type them into the computer. You should *not*, however, type them in exactly as they appear. Follow normal syntax and entry procedures as described in your user's manual.

Level I Programs

Programs originally in Level I have been converted to allow running in Level II. To run in Level I, follow this procedure:

- Delete any dimension statements. Example: DIM A (25).
- Change PRINT@ to PRINTAT.
- Make sure that no INPUT variable is a STRING variable.
  Example: INPUT A$ would be changed to INPUT A and subsequent code made to agree.
- Abbreviate all BASIC statements as allowed by Level I.
  Example: *PRINT* is abbreviated *P*.

Model III Users

For the Model I, OUT255,0 and OUT255,4 turn the cassette motor off and on, respectively. For the Model III, change these statements to OUT236,0 and OUT236,2.

APPENDIX B

Glossary

A

ac input module—I/O rack module which converts various ac signals originating in user switches to the appropriate logic level for use within the processor.

ac output module—I/O rack module which converts the logic levels of the processor to a usable output signal to control a user's ac load.

access time—the elapsed time between a request for data and the appearance of valid data on the output pins of a memory chip. Usually 200–450 nanoseconds for TRS-80 RAM.

accumulator—the main register(s) in a microprocessor used for arithmetic, shifting, logical, and other operations.

accuracy—generally, the quality or freedom from mistake or error; the extent to which the results of a calculation or a measurement approach the true value of the actual quantities.

acoustic coupler—a connection to a modem allowing signals to be transmitted through a regular telephone handset.

active elements—any generators of voltage or current in an impedance network; also known as active components.

adaptor—a device for connecting parts that will not mate; a device designed to provide a compatible connection between systems or subsystems.

A/D converter—analog to digital converter. See D/A converter.

add with carry—a machine-language instruction in which one operand is added to another, along with a possible carry from the previous (lower-order) add.

address—a code that specifies a register, memory location, or other data source or destination.

ALGOL—an acronym for ALGOrithmic Language. A very high-level language used in scientific applications, generally on large-scale computers.

algorithm—a predetermined process for the solution of a problem or completion of a task in a finite number of steps.

alignment—the process of adjusting components of a system for proper interrelationships, including adjustments and synchronization for the components in a system. For the TRS-80, this usually applies to cassette heads and disk drives.

alphanumerics—refer to the letters of the alphabet and digits of the number system, specifically omitting the characters of punctuation and syntax.

alternating current—ac. Electric current that reverses direction periodically, usually many times per second.

ALU—Arithmetic Logic Unit.

Ampere—the unit of electric current in the meter-kilogram-second system of units; defined in terms of the force of attraction between two parallel current conductors; 1 coulomb/second.

Ampere-turn—a unit of magnetomotive force defined as the force of a closed loop of one turn with a current of one ampere flowing through the loop.

analog—the representation of a physical variable by another variable insofar as the proportional relationships are the same over some specified range.

analog input module—an I/O rack module which converts an analog signal from a user device to a digital signal which may be processed by the processor.

analog output module—an I/O rack module which converts a digital signal from the processor into an analog output signal for use by a user device.

AND—a Boolean logic function. Two operators are tested and, if both are true, the answer is true. Truth is indicated by a high bit, or 1 in machine language, or a positive value in BASIC. If the operators are bytes or words, each element is tested separately. A bit-by-bit logical operation which produces a one in the result bit only if both operand bits are ones.

anode—in a semiconductor diode, the terminal toward which electrons flow from an external circuit; the positive terminal.

APL—a programming language; a popular and powerful high-level mathematical language with extensive symbol manipulation.

argument—any of the independent variables accompanying a command.

Arithmetic Logic Unit—ALU. The section of a microprocessor which performs arithmetic functions such as addition or subtraction and logic functions such as ANDing.

arithmetic shift—a type of shift in which an operand is shifted right or left with the sign bit being extended (right shift) or maintained (left shift).

array—a collection of data items arranged in a meaningful pattern such as rows and columns which allow the collection and retrieval of data.

ASCII—American Standard Code for Information Interchange. An almost universally accepted code (at least for punctuation and capital letters) where characters and printer commands are represented by numbers between 0 and 255 (base 10). The number is referred to as an ASCII code.

assembler—software that translates operational codes into their binary equivalents on a statement-for-statement basis.

assembly language—a symbolic computer language that is translated by an assembler program into machine language, the numeric codes that are equivalent to microprocessor instructions.

asynchronous—not related through repeating time patterns.

asynchronous shift register—a shift register which does not require a clock. Register segments are loaded and shifted only at data entry.

B

backup—1) refers to making copies of all software and data stored externally 2) having duplicate hardware available.

base—the starting point for representation of a number in written form, where numbers are expressed as multiples of powers of the base value.

BASIC—an acronym for Beginner's All-purpose Symbolic Instruction Code. Developed at Dartmouth College and similar to FORTRAN. The standard, high-level, interactive language for microcomputers.

batch processing—a method of computing in which many of the same types of jobs or programs are done in one machine run. For example, a programming class may type programs on data cards and turn them over to the computer operator. All the cards are put into the card reader, and the results of each person's program are returned later. This is contrasted with interactive computing.

baud—1) a unit of data transmission speed equal to the number of code elements (bits) per second 2) a unit of signaling speed equal to the number of discrete conditions or signal events per second.

baud rate—a measure of the speed at which serial data is transmitted electronically. The equivalent of bits per second (bps) in microcomputing.

benchmark—to test performance against a known standard.

BCD—binary coded decimal. The 4-bit binary notation in which individual decimal digits (0 through 9) are represented by 4-bit binary numerals; e.g., the number 23 is represented by 0010 0011 in the BCD notation.

bias—a dc voltage applied to a transistor control electrode to establish the desired operating point.

bidirectional bus—a bus structure used for the two-way transmission of signals, that is, both input and output.

bidirectional printer—a printer capable of printing both left-to-right and right-to-left. Data is prestored in a fixed-size buffer.

binary—a number system which uses only 0 and 1 as digits. It is the equivalent of base 2. Used in microcomputing because it is easy to represent 1s and 0s by high and low electrical signals.

binary digit—the two digits, zero and one, used in binary notation. Often shortened to bit.

binary point—the point, analagous to a decimal point, that separates the integer and fractional portions of a binary mixed number.

bipolar device—a device whose operation depends on the transport of holes and electrons, usually made of layers of silicon with differing electrical characteristics.

bi-stable—two-state

bit—an abbreviation for binary digit. A 0 or 1 in the binary number system. A single high or low signal in a computer.

bit position—the position of a binary digit within a byte or larger group of binary digits. Bit positions in the Model I, II, III, and Color Computer are numbered from right to left, zero through N. This number corresponds to the power of two represented.

Boolean algebra—a mathematical system of logic first identified by George Boole, a 19th century English mathematician. Routines are described by combinations of ANDs, ORs, XORs, NOTs, and IF-THENs. All computer functions are based upon these operators.

boot—short for bootstrap loader or the use of one. The bootstrap loader is a very short routine whose purpose is to load a more sophisticated system into the computer when it is first turned on. On some machines it is keyed in, and on others it is in read only memory (ROM). Using this program is called booting or cold-starting the system.

borrow—one bit subtracted from the next higher bit position.

bps—bits per second.

breakdown—a large, abrupt rise in electric current due to decreased resistance in a semiconductor device caused by a small increase in voltage.

buffer—memory set aside temporarily for use by the program. Particularly refers to memory used to make up differences in the data transfer rates of the computer and external devices such as printers and disks.

bug—an error in software or hardware.

bump contact—a large area contact used for alloying directly to the substrate of a chip for mounting or interconnecting purposes.

bus—an ordered collection of all address, data, timing, and status lines in the computer.

byte—eight bits that are read simultaneously as a single code.

C

CAI—an acronym for Computer Aided Instruction.

card—a specially designed sheet of cardboard with holes punched in specific columns. The placement of the holes represents machine-readable data. Also a term referring to a printed circuit board.

card reader—a device for reading information from punched cards.

carrier—a steady signal that can be slightly modified (modulated) continuously. These modulations can be interpreted as data. In microcomputers the technique is used primarily in modern communications and tape input/output (I/O).

carry—a one bit added to the next higher bit position or to the carry flag.

carry flag—a bit in the microprocessor used to record the carry "off the end" as a result of a machine-language instruction.

cassette recorder—a magnetic tape recording and playback device for entering or storing programs.

cathode—in a semiconductor diode, the terminal from which electrons flow to an external circuit; the negative terminal.

character—a single symbol that is represented inside the computer by a specific code.

charge—a basic property of elementary particles of matter. The charge, measured in coulombs, is the algebraic sum of the electric charge of its constituents.

checksum—a method of detecting errors in a block of data by adding each piece of data in the block to a sum and comparing the final result to a predetermined result for the block of data.

chip—the shaped and processed semiconductor die mounted on a substrate to form a transistor or other semiconductor device.

circuit—a conductor or system of conductors through which an electric current may flow.

circuit card—a printed circuit board containing electronic components.

clear—to return a memory to a non-programmed state, usually represented as 0 or OFF (empty).

clobber—to destroy the contents of memory or a register.

clock—a simple circuit that generates the synchronization signals for the microprocessor. The speed or frequency of this clock directly affects the speed at which the computer can perform, regardless of the speed of which the individual chips are capable.

COBOL—COmmon Business-Oriented Language. A language used primarily for data processing. Allows programming statements that are very similar to English sentences.

Colossus—a British computer used to crack German Enigma codes during World War II.

common carrier—a communications transmission medium, such as the Direct Distance Dialing (DDD) network of the Bell System.

compiler—software that will convert a program written in a high-level language to binary code, on a many-for-one basis.

complement—a mathematical calculation. In computers it specifically refers to inverting a binary number. Any 1 is replaced by a 0, and vice versa.

complementary functions—two driving point functions whose sum is a positive constant.

complementary metal oxide semiconductor—CMOS. A signal inverting device formed by the combination of a p channel with an n channel device usually connected in series across the power supply.

complementary transistors—two transistors of opposite conductivity (pnp and npn) in the same functional unit.

computer interface—a device designed for data communication between a central computer and another unit such as a programmable controller processor.

concatenate—to put two things, each complete by itself, together to make a larger complete thing. In computers this refers to strings of characters or programs.

conditional jump—a machine-language instruction that jumps if a specified flag (or flags) is set or reset.

conductor—a substance, body, or other medium that is suitable to carry an electric current.

constant—a value that doesn't change.

control block—a storage area of a microprocessor containing the information required for control of a task, function, operation, or quantity of information.

coulomb—the unit of electric charge in SI units (International System of Units); the quantity of electric charge that passes any cross section of a conductor in one second when current is maintained constant at one ampere.

counter—in relay-panel hardware, an electro-mechanical device which can be wired and preset to control other devices according to the total cycles of one ON and OFF function. A counter is internal to the processor; i.e., it is controlled by a user-programmed instruction. A counter instruction has greater capability than any hardware counter.

CPU—central processing unit. The circuitry that actually performs the functions of the instruction set.

CRT—cathode ray tube. In computing this is just the screen the data appears on. A TV has a CRT.

cue—refers to positioning the tape on a cassette unit so that it is set up to a read/write section of tape.

current—the net transfer or electric charge per unit of time by free electrons; 1 ampere = 1 coulomb/second.

current mode logic—CML. Integrated circuit logic in which transistors are paralleled so as to eliminate current hogging.

cursor—a visual movable pointer used on a CRT by the programmer to indicate where an instruction is to be added to the program. The cursor is also used during editing functions.

cycle—a specific period of time, marked in the computer by the clock.

D

D/A converter—digital to analog converter. Common in interfacing computers to the outside world.

daisy chain—a bus line which interconnects devices for serial operation.

daisy wheel—a printer type which has a splined character wheel.

data—general term for numbers, letters, symbols, and analog quantities that serve as information for computer processing.

data base—refers to a series of programs each having a different function, but all using the same data. The data is stored in one location or file and each program uses it in a fashion that still allows the other program to use it.

data entry—the practice of entering data into the computer or onto a storage device. Knowledge of operating or programming a computer is not necessary for a data entry operator.

data link—equipment, especially transmission cables and interface modules, which permits the transmission of information.

debug—to remove bugs from a program.

decrement—to decrease the value of a number. In computers the number is in memory or a register, and the amount it is decremented is usually one.

dedicated—in computer terminology, a system set up to perform a single task.

default—that which is assumed if no specific information is given.

degauss—to demagnetize. Must be done periodically to tape and disk heads for reliable data transfer.

diagnostic program—a test program to help isolate hardware malfunctions in the programmable controller and application equipment.

die bond—a process in which chips are joined to a substrate.

differential discriminator—a circuit that passes only pulses whose amplitudes are between two predetermined values, neither of which are zero.

digital—the representation of data in binary code. In microcomputers, a high electrical signal is a 1 and a low signal is a 0.

digital circuit—an electronic network designed to respond at input voltages at one level, and similarly, to produce output voltages at one level.

diode—a device with an anode and a cathode which permits current flow in one direction and inhibits current flow in the other direction.

diode transistor logic—a circuit that uses diodes, transistors, and resistors to provide logic functions.

direct current—dc. Electric current which flows in only one direction; the term designates a practically non-pulsating current.

disassembly—remaking an assembly source program from a machine-code program.

disk—an oxide-coated, circular, flat object, in a variety of sizes and containers, on which computer data can be stored.

disk controller—an interface between the computer and the disk drive.

disk drive—a piece of hardware that rotates the disk and performs data transfer to and from the disk.

disk operating system—DOS. The system software that manipulates the data to be sent to the disk controller.

displacement—a signed value in machine language used in defining a memory address.

dividend—the number that is divided by the divisor. In A/B, A is the dividend.

divisor—the number that "goes into" the dividend in a divide operation. In A/B, B is the divisor.

DMA—direct memory access. A process where the CPU is disabled or bypassed temporarily and memory is read or written to directly.

documentation—a collection of written instructions necessary to use a piece of hardware, software, or a system.

domain—a region in a solid within which elementary atomic, molecular, magnetic, or electric moments are uniformly arrayed.

doping—the addition of impurities to a semiconductor to achieve a desired characteristic.

dot-matrix printer—instead of each letter having a separate type head (like a typewriter), a single print head makes the characters by printing groups of dots. The print is not as easy to read, but such printers are less expensive to manufacture.

double-dabble—a method of converting from binary to decimal representation by doubling the leftmost bit, adding the next bit, and continuing until the rightmost bit has been processed.

downtime—the time when a system is not available for production due to required maintenance.

driver—a small piece of system software used to control an external device such as a keyboard or printer.

dump—to write data from memory to an external storage device.

duplex—refers to two-way communications taking place independently, but simultaneously.

dynamic memory—circuits that require a periodic (every few milliseconds) recharge so that the stored data is not lost.

E

EAROM—an acronym for Electrical Alterable Read Only Memory. The chip can be read at normal speed, but must be written to with a slower process. Once written to, it is used like a ROM, but can be completely erased if necessary.

editor—a program that allows text to be entered into memory. Interactive languages usually have their own editors.

electron—a stable elementary particle with a negative electric charge of about -1.602×10^{-19} coulomb.

emitter-coupled logic—a form of current mode logic in which the emitters of two transistors are connected to a single current-carrying resistor in a way that only one transistor conducts at a time.

enhancement mode—operation of a field effect transistor in which no current flows when zero gate voltage is applied, and increasing the gate voltage increases the current.

EOF—End Of File.

EOL—End Of Line (of text).

EPROM—Erasable Programmable Read Only Memory. A read only memory in which stored data can be erased by ultraviolet light or other means and reprogrammed bit-by-bit with appropriate voltage pulses.

Exclusive OR—a bit-by-bit logical operation which produces a one bit in the result only if one or the other (but not both) operand bits is a one.

execution—the performance of a specific operation such as would be accomplished through processing one instruction, a series of instructions, or a complete program.

execution cycle—a cycle during which a single instruction of one specific operation.

execution time—the total time required for the execution to actually occur.

expansion interface—a device attached to the computer that allows a greater amount of memory or attachment of other peripherals.

exponent—the power of two of a floating-point number.

F

feedback—the signal or data fed back to the programmable controller from a controlled machine or process to denote its response to the command signal.

fetch cycle—a cycle during which the next instruction to be performed is read from memory.

Fibonacci series—the sequence of number 1, 1, 2, 3, 5, 8, 13, 21, 34, . . . in which each term is computed by addition of the two previous terms.

field-effect transistor—FET. A transistor in which the resistance of the current path from the source to drain is modulated by applying a transverse electric field between grid or gate electrodes; the electric field varies the thickness of depletion layers between the gates, thereby reducing the conductance.

file—a set of data, specifically arranged, that is treated as a single entity by the software or storage device.

filter—electrical device used to suppress undesirable electrical noise.

firmware—software that is made semi-permanent by putting it into some type of ROM.

flag—a single bit that is high (set) or low (reset), used to indicate whether or not certain conditions exist or have occurred.

flip chip—a tiny semiconductor die having terminations all on one side in the form of solder pads or bump contacts; after the surface of the chip has been passivated or otherwise treated, it is flipped over for attaching to a matching substrate.

flip-flop—a bi-stable device that assumes either of two possible states such that the transition between the states must be accomplished by electronic switching.

floating-point number—a standard way of representing any size of number in computers. Floating-point numbers contain a fractional portion (mantissa) and power of two (exponent) in a form similar to scientific notation.

flowcharting—a method of graphically displaying program steps, used to develop and define an algorithm before writing the actual code.

FORTRAN—FORmula TRANslator. One of the first high-level languages, written specifically to allow easy entry of mathematical problems.

full duplex—a mode of data transmission that is the equivalent of two paths—one in each direction simultaneously.

G

game theory—see von Neumann.

garbage—computer term for useless data.

gate—a circuit that performs a single Boolean function. A circuit having an output and a multiplicity of inputs, so designed that the output is energized only when a certain combination of pulses is present at the inputs.

GIGO—Garbage In, Garbage Out. One of the rules of computing. If the data going into the computer is bad, the data coming out will be bad also.

graphics—information displayed pictorially as opposed to alphanumerically.

ground—a conducting path, intentional or accidental, between an electric circuit or equipment and the earth, or some conducting body serving in place of the earth.

H

H—a suffix for hexadecimal, e.g., 4FFFH.

half duplex—data can flow in both directions, but not simultaneously. See duplex.

Hall effect—the development of a transverse electric field in a current-carrying conductor placed in a magnetic field; ordinarily the conductor is positioned so that the magnetic field is perpendicular to the direction of current flow and the electric field is perpendicular to both.

Hall generator—a generator using the Hall effect to give an output voltage proportional to magnetic field strength.

handshaking—a term used in data transfer. Indicates that beside the data lines there are also signal lines so both devices know precisely when to send or receive data. Handshaking requires clocking pulses on both ends of the communications line. Contrast with buffer.

hangup—the computer has ceased processing inexplicably.

hard copy—a printout; any form of printed document such as a ladder diagram, program listing, paper tape, or punched cards.

hard magnetic—a term describing a metal having a high coercive force which gives a high magnetic hysteresis; usually a permanent magnetic material.

hard wired—having a fixed wired program or control system built in by the manufacturer and not subject to change by programming.

hardware—refers to any physical piece of equipment in a computer system.

hex—hexadecimal.

hexa-dabble—conversion from hexadecimal to decimal by multiplying each hex digit by sixteen and adding the next hex digit until the last (rightmost) hex digit has been reached.

hexadecimal—representation of numbers in base sixteen by use of the hexadecimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

high—a signal line logic level. The computer senses this level and treats it as a binary 1.

high-level language—a programming language which is CPU-independent and closely resembles English.

high order—see most significant bit.

HIT—acronym for Hash Index Table. A section of the directory on a TRS-80 disk.

hole—a mobile vacancy having an energy state near the top of the energy band of a solid; behaves as though it were a positively charged particle.

host computer—the primary computer in a multi-computer or terminal hookup.

human engineering—usually refers to designing hardware and software with ease of use in mind.

hysteresis—an oscillator effect wherein a given value of an operating parameter may result in multiple values of output power or frequency.

I

IC—integrated circuit.

immediate—addressing mode in which the address of the information that an operation is supposed to act upon immediately follows the operation code.

inclusive OR—a bit-by-bit logical operation which produces a one-bit result if one or the other operand bits, or both is a one.

increment—to increase, usually by one. See decrement.

indexed—addressing mode where the information is addressed by a specified value, or by the value in a specified register.

indirect—addressing mode in which the address given points to another address, and the second address is where the information actually is.

input devices—devices such as limit switches, pressure switches, push buttons, etc., that supply data to a programmable controller. These discrete inputs are two types: those with common return, and those with individual returns (referred to as isolated inputs). Other inputs include analog devices and digital encoders.

instruction—a command or order that will cause a computer to perform one certain prescribed operation.

insulator—a nonconducting material used for supporting or separating conductors to prevent undesired current flow to other objects.

integer variable—a BASIC variable type. It can hold values of −32,768 through +32,767 in two-byte two's complement notation.

integrated circuit—IC. An interconnected array of active and passive elements integrated with a single semiconductor substrate or deposited on the substrate and capable of performing at least one electronic circuit function. See chip.

integrated injection logic—I^2L. Integrated circuit logic which uses a simple and compact bipolar transistor gate structure which makes possible large scale integration on silicon for logic arrays and other analog and digital applications.

intelligent terminal—a terminal with a CPU and a certain amount of memory that can organize the data it receives and thus achieve a high level of handshaking with the host computer.

interactive computing—refers to the appearance of a one-to-one human-computer relationship.

interface—a piece of hardware, specifically designed to hook two other devices together. Usually some software is also required.

interpreter—a piece of system software that executes a program written in a high-level language directly. While useful for interactive computing, this system is too slow for most serious programming. Contrast with compiler.

interrupt—a signal that tells the CPU that a task must be done immediately. The registers are pushed to the stack, and a routine for the interrupt is branched to. When finished, the registers are popped from the stack and the main program continues.

I/O—acronym for input/output. Refers to the transfer of data.

I/O module—the printed circuit board that is the termination for field wiring of I/O devices.

I/O rack—a chassis which contains I/O modules.

I/O scan—the time required for the programmable controller processor to monitor all inputs and control all outputs. The I/O scan repeats continuously.

isolated I/O module—a module which has each input or output electrically isolated from every other input or output on that module. That is to say, each input or output has a separate return wire.

iteration—one pass through a given set of instructions.

J

jack—a socket, usually mounted on a device, which will receive a plug (generally mounted on a wire).

Josephson effect—the tunneling of electron pairs through a thin insulating barrier between two superconducting materials.

K

K—abbreviation for kilo. In computer terms 1024, in loose terms 1000.

Karnaugh map—a truth table that shows a geometrical pattern of functional relationships for gating configurations; with this map, essential gating requirements can be recognized in their simplest form.

L

ladder diagrams—an industry standard for representing control logic relay systems.

language—a set of symbols and rules for representing and communicating information (data) among people, or between people and machines.

large scale integration—LSI. Any integrated circuit which has more than 100 equivalent gates manufactured simultaneously on a single slice of semiconductor material.

latching relay—a relay with 2 separate coils, one of which must be energized to change the state of the relay; it will remain in either state without power.

leakage current—in general, the undesirable flow of current through or over the surface of an insulating material or insulator; the alternating current that passes through a rectifier without being rectified.

leakage flux—magnetic lines of force that go beyond their intended space.

least significant bit—the rightmost bit in a binary value, representing 2^0.

least significant byte—refers to the lowest position digit of a number. The rightmost byte of a number or character string.

LIFO—acronym for Last In First Out. Most CPUs maintain a "stack" of memory. The last data pushed onto the stack is the first popped out.

light emitting diode—LED. A semiconductor diode that converts electric energy efficiently into spontaneous and noncoherent electromagnetic radiation at visible and near infrared wavelengths of electroluminescence at a forward biased pn junction.

light pen—a device that senses light, interfaced to the computer for the purpose of drawing on the CRT screen.

line—in communications, describes cables, telephone lines, etc., over which data is transmitted to and received from the terminal.

line driver—an integrated circuit specifically designed to transmit digital information over long lines—that is, extended distances.

line printer—a high-speed printing device that prints an entire line at one time.

linear circuit—a network in which the parameters of resistance, inductance, and capacitance are constant with respect to current or voltage, and in which the voltage or current of sources is independent of or directly proportional to the outputs.

linearity—the relationship that exists between two quantities when a change in one of them produces a direct proportional change in the other.

location—a storage position in memory.

logic—a means of solving complex problems through the repeated use of simple functions which define basic concepts. Three basic logic functions are AND, OR, and NOT.

logic diagram—a drawing which represents the logic functions AND, OR, NOT, etc.

logic level—the voltage magnitude associated with signal pulses representing ones and zeroes (1s and 0s) in binary computation.

logical shift—a type of shift in which an operand is shifted right or left, with a zero filling the vacated bit position.

loop—a set of instructions that executes itself continuously. If the programmer has the presence of mind to provide for a test, the loop is discontinued when the test is met, otherwise it goes on until the machine is shut down.

loop counter—one way to test a loop. The counter is incremented at each pass through the loop. When it reaches a certain value, the loop is terminated.

low—a logic signal voltage. The computer senses this as a binary 0.

lsb—see least significant bit.

LSI—acronym for Large Scale Integration. An integrated circuit with a large number of circuits such as a CPU. See chip.

M

machine code—refers to programming instructions that are stored in binary and can be executed directly by the CPU without any compilation, interpretation, or assembly.

machine language—the primary instructions that were designed into the CPU by the manufacturer. These instructions move data between memory and registers, perform simple adding in registers, and allow branching based on values in registers.

macro—a routine that can be separately programmed, given a name, and executed from another program. The macro can perform functions on variables in the program that called it without disturbing anything else and then return control to the calling program.

magnetoresistor—magnetic field controlled variable resistor.

magnitude—the absolute value, independent of direction.

mainframe—refers to the CPU of a computer. This term is usually confined to larger computers.

mantissa—the fractional portion of a floating-point number.

matrix—a two-dimensional array of circuit elements, such as wires, diodes, etc., which can transform a digital code from one type to another.

memory—the hardware that stores data for use by the CPU. Each piece of data (bit) is represented by some type of electrical charge. Memory can be anything from tiny magnetic doughnuts to bubbles in a fluid. Most microcomputers have chips that contain many microscopic capacitors, each capable of storing a tiny electrical charge.

memory module—a processor module consisting of memory storage and capable of storing a finite number of words (e.g., 4096 words in a 4K memory module). Storage capacity is usually rounded off and abbreviated with K representing each 1024 words.

metal oxide semiconductor—MOS. A metal insulator semiconductor structure in which the insulating layer is an oxide of the substrate material; for a silicon substrate the insulator is silicon oxide.

micro electronics—refers to circuits built from miniaturized components and includes integrated circuits.

microprocessor—an electronic computer processor section implemented in relatively few IC chips (typically LSI) which contain arithmetic, logic, register, control, and memory functions.

microsecond—μs. One millionth of a second: 1×10^{-6} or 0.000001 second.

millisecond—μs. One thousandth of a second: 10^{-3} or 0.001 second.

minuend—the number from which the subtrahend is subtracted.

mixed number—a number consisting of an integer and fraction as, for example, 4.35 or (binary) 1010.1011.

mnemonic—a short, alphanumeric abbreviation used to represent a machine-language code. An assembler will take a program written in these mnemonics and convert it to machine code.

modem—MOdulator/DEModulator. An I/O device that allows communication over telephone lines.

module—an interchangeable plug-in item containing electronic components which may be combined with other interchangeable items to form a complete unit.

monitor—1) a CRT 2) a short program that displays the contents of registers and memory locations and allows them to be changed. Monitors can also allow another program to execute one instruction at a time, saving programs and disassembling them.

MOS—see metal oxide semiconductor.

MOSFET—metal oxide semiconductor field effect transistor.

most significant bit—the leftmost bit in a binary value, representing the highest-order power of two. In two's complement notation, this bit is the sign bit.

most significant byte—the highest-order byte. In the multiple-precision number A13EF122H, A1H is the most significant byte.

msb—see most significant byte.

multiple-precision numbers—multiple-byte numbers that allow extended precision.

multiplexing—a method allowing several sets of data to be sent at different times over the same communication lines, yet all of the data can be used simultaneously after the final set is received. For example, several LED displays, each requiring four data lines, can all be written to with only one group of four data lines. The same concept is used with communication lines.

multiplicand—the number to be multiplied by the multiplier.

multiplicand register—the register used to hold the multiplicand in a machine-language multiply.

multiplier—the number that is multiplied against the multiplicand. The number "on the bottom."

N

NAND—an acronym for NOT AND. A Boolean logic expression. AND is performed, then NOT is performed to the result.

n-channel—a conduction channel formed by electrons in an n-type semiconductor, as in an n-type field-effect transistor.

negation—changing a negative value to a positive value, or vice versa. Taking the two's complement by changing all ones to zeros, all zeros to ones, and adding one.

nesting—putting one loop inside another. Some computers limit the number of loops that can be nested.

network—a collection of electric elements, such as resistors, coils, capacitors, and sources of energy, connected together to form several interrelated circuits. A collection of computer terminals interconnected to a host CPU.

noise—extraneous signals; any disturbance which causes interference with the desired signal or operation.

non-volatile memory—a memory that does not lose its information while its power supply is turned off.

normalization—converting data to a standard format for processing. In floating-point format, converting a number so that a significant bit (or hex digit) is the first bit (or four bits) of the fraction.

NOT—a Boolean operator that reverses outputs (1 becomes 0, 0 becomes 1). This is the one's complement.

NPN transistor—a junction transistor having a p-type base between an n-type emitter and an n-type collector; the emitter should then be negative with respect to the base and the collector should be positive with respect to the base.

n-type semiconductor—an extrinsic semiconductor in which the conduction electron density exceeds the hole density.

O

object code—all of the machine code that is generated by a compiler or assembler. Once object code is loaded into memory it is called machine code.

octal—refers to the base 8 number system, using digits 0–7.

octal-dabble—conversion of an octal number to decimal by multiplying by eight and adding the next octal digit, continuing until the last (rightmost) digit has been added.

OEM—Original Equipment Manufacturer.

off-line—describes equipment or devices which are not connected to the communications line.

offset value—a value that can be added to an address. Most addressing modes allow an offset value.

off-the-shelf—a term referring to software. A generalized program that can be used by a greater number of computer owners, so that it can be mass produced and bought off-the-shelf.

Ohm—the unit of resistance of a conductor such that a constant current of one ampere in it produces a voltage of one volt between its ends.

Ohm's law—a fundamental rule of electricity; states that the current in an electric circuit is inversely proportional to the resistance of the circuit and is directly proportional to the electromotive force in the circuit. In its strictest sense, Ohm's law applies only to linear constant-current circuits.

on-line—a term describing a situation where one computer is connected to another, with full handshake, over a modem line.

on-line operation—operations where the programmable controller is directly controlling the machine or process.

operands—the numeric values used in the add, subtract, or other operation.

OR—a Boolean logic function. If at least one of the lines tested is high (binary 1), the answer is high.

oscillation—any effect that varies periodically back and forth between two values, as in the amplitude of an alternating current.

output—the current, voltage, power, driving force, or information which a circuit or a device delivers. The terminals or other places where a circuit or device can deliver energy.

output devices—devices such as solenoids, motor starters, etc., that receive data from the programmable controller.

overflow—a condition that exists when the result of an add, subtract, or other arithmetic operation is too large to be held in the number of bits allotted.

overflow flag—a bit in the microprocessor used to record an overflow condition for machine-language operation.

overlay—a method of decreasing the amount of memory a program uses by allowing sections that are not in use simultaneously to load into the same area of memory. The new routine destroys the first routine, but it can always be loaded again if needed. Usually used in system programs.

oxide—an iron compound coating on tapes and disks that allows them to be magnetized so that they can be read by electrical devices and the information converted back to machine code.

P

padding—filling bit positions to the left with zeros to make a total of eight or sixteen bits.

page—refers to a 256 (2 to the 8th power) word block of memory. How large a word depends on the computer. Most micros are eight-bit word machines. Many chips do special indexed and offset addressing on the page where the program counter is pointing and/or on the first page of memory.

parallel—describes a method of data transfer where each bit of a word has its own data line, and all are transferred simultaneously.

parallel circuit—an electric circuit in which the elements, branches (having elements in series), or components are connected between two points, with one of the two ends of each component connected to each point.

parallel operation—type of information transfer whereby all digits of a word are handled simultaneously.

parallel output—simultaneous availability of two or more bits, channels, or digits.

parameter—a variable or constant that can be defined by the user and usually has a default value.

parity—a method of checking accuracy. The parity is found by adding all the bits of a word together. If the answer is even, the parity is 0 or even. If odd, the parity is 1 or odd. The bit sometimes replaces the most significant bit and usually sets a flag.

parity bit—an additional bit added to a memory word to make the sum of the number of 1s in a word always even or odd as required.

parity check—a check that tests whether the number of 1s in an array of binary digits is odd or even.

partial product—the intermediate results of a multiply. At the end, the partial product becomes the whole product.

partial product register—the register used to hold the partial results of a machine-language multiply.

passivation—growth of an oxide layer on the surface of a semiconductor to provide electrical stability by isolating the transistor surface from electrical and chemical conditions in the environment; this reduces reverse-current leakage, increases breakdown voltage, and raises power dissipation rating.

passive element—an element of an electric circuit that is not the source of energy, such as a resistor, inductor, or capacitor.

PC—see programmable controller.

PC board—see printed circuit board.

p-channel—a conduction channel formed by holes in a p-type semiconductor, as in a p-type field effect transistor.

peripheral devices—a generic term for equipment attached to a computer, such as keyboards, disk drives, cassette tapes, printers, plotters, speech synthesizers.

permeability—a factor, characteristic of a material, that is proportional to the magnetic induction produced in a material divided by the magnetic field strength given by the equation:

$$m = \frac{\text{magnetic induction (gauss)}}{\text{magnetizing field (oersteds)}}$$

permutation—arrangements of things in definite order. Two binary digits have four permutations: 00, 01, 10, and 11.

PILOT—a simple language for handling English sentences and strings of alphanumeric characters. Generally used for CAI.

PL/1—an acronym for programming language 1. A programming language used by very large computers. It incorporates most of the better features from other programming languages. Its power comes from the fact that bits can be manipulated from the high-level language.

plotter—a device that can draw graphs and curves and is controlled by the computer through an interface.

port—a single addressable channel used for communications.

P-N junction—a region of transition between p-type emitter and n-type semiconducting regions in a semiconductor device.

PNP transistor—a junction type transistor having an n-type base between a p-type emitter and a p-type collector.

positional notation—representation of a number where each digit position represents an increasingly higher power of the base.

precision—the number of significant digits that a variable or number format may contain.

print buffer—a portion of memory dedicated to holding the string of characters to be printed.

printed circuit board—a piece of plastic board with lines of a conductive material deposited on it to connect the components. The lines act like wires. These can be manufactured quickly and are easy to assemble the components on.

processor—a unit in the programmable controller which scans all the inputs and outputs in a predetermined order. The processor monitors the status of the inputs and outputs in response to the user-programmed instructions in memory, and it energizes or de-energizes outputs as a result of the logical comparisons made through these instructions.

product—the result of a multiply.

program—a sequence of instructions to be executed by the processor to control a machine or process.

program panel—a device for inserting, monitoring, and editing a program in a programmable controller.

program scan—the time required for the programmable controller processor to execute all instructions in the program once. The program scan repeats continuously. The program monitors inputs and controls outputs through the input and output image tables.

programmable controller—PC. A solid state control system which has a user-programmable memory for storage of instructions to implement specific functions such as I/O control logic, timing, counting, arithmetic, and data manipulation. A PC consists of the central processor, input/output interface, memory, and programming device which typically uses relay-equivalent symbols. The PC is purposely designed as an industrial control system which can perform functions equivalent to a relay panel or a wired solid state logic control system.

PROM—Programmable Read Only Memory. A memory device that is written to once and from then on acts like a ROM.

protocol—a defined means of establishing criteria for receiving and transmitting data through communication channels.

pseudo code—a mnemonic used by assemblers that is not a command to the CPU, but a command to the assembler itself.

p-type semiconductor—an extrinsic semiconductor in which the hole density exceeds the conduction electron density.

punched-card equipment—peripheral devices that enable punching or reading paper punched cards that hold character or binary data.

Q

quotient—the result of a divide operation.

R

RAM—acronym for Random Access Memory. An addressable LSI device used to store information in microscopic flip-flops or capacitors. Each may be set to an ON or OFF state, representing logical 1 or 0. This type of memory is volatile, that is to say, memory is lost while power is off, unless battery backup is used.

read—to sense the presence of information in some type of storage, which includes RAM memory, magnetic tape, punched tape, etc.

real time clock—a clock in the sense that we normally think of one, interfaced to the computer.

record—a file is divided into records, each of which is organized in the same manner.

register—a fast-access memory location in the microprocessor. Used for holding intermediate results and for computation in machine language.

relative addressing—an address that is dependent upon where the program counter is presently pointing.

remainder—the amount of divident remaining after a divide has been completed.

residue—the amount of dividend remaining, part way through a divide.

resistor-transistor logic—RTL. One of the simplest logic circuits, having several resistors, a transistor, and a diode.

resolution—a measure of the smallest possible increment of change in the variable output of a device.

restoring divide—a divide in which the divisor is restored if the divide "does not go" for any iteration. A common microcomputer divide technique.

ROM—an acronym for Read Only Memory. Memory that is addressed by the bus, but can only be read from. If you tell the CPU to write to it, the machine will try, but the data is not remembered.

rotate—a type of shift in which data is recirculated right or left back into the operand from the opposite end.

rounding—the process of truncating bits to the right of a bit position and adding zero or one to the next higher bit position based on the value to the right. rounding the binary fraction 1011.1011 to two fractional bits, for example, results in 1011.11.

RPG—an acronym for Report Program Generator. A language for business that primarily reads data from cards and prints reports containing that data.

RS-232—an interface that converts parallel data to serial data for communications purposes. The output is universally standard.

rung—a grouping of PC instructions which controls one output. This is represented as one section of a logic ladder diagram.

S

scaled up—referring to a number which has been multiplied by a scale factor for processing.

scaling—multiplying a number by a fixed amount so that a fraction can be processed as an integer value.

scan time—the time necessary to completely execute the entire programmable controller program one time.

scientific notation—a standard form for representing any size number by a mantissa and power of ten.

self-diagnostic—the hardware and firmware within a controller which allows it to continuously monitor its own status and indicate any fault which might occur within.

semiconductor—a compound that can be made to vary its resistance to electricity by mixing it differently. Layers of this material can be used to make circuits that do the same things tubes do, but using much less electricity. Transistors and integrated circuits are made from semiconductive material and are called semiconductors.

semiconductor device—an electronic device in which the characteristic distinguishing electronic conduction takes place within a semiconductor.

sensor—a sensing element, a device which senses either the absolute value or the change in a physical quantity, and converts that change into a useful signal for an information-gathering system.

serial—a way of sending data, one bit at a time, between two devices. The bits are rejoined into bytes by the receiving device. contrast with parallel.

serial operation—type of information transfer within a programmable controller whereby the bits are handled sequentially rather than simultaneously, as they are in parallel operation. Serial operation is slower than parallel operation for equivalent clock rates. However, only one channel is required for serial operation.

series circuit—a circuit in which all parts are connected end to end to provide a single path for current.

shift and add—a multiply method in which the multiply is achieved by shifting of and addition of the multiplicand.

shift register—a program, entered by the user into the memory of a programmable controller, in which the information data (usually single bits) is shifted one or more positions on a continual basis. There are two types of shift registers: asynchronous and synchronous.

sign bit—sometimes the most significant bit is used to indicate the sign of the number it represents. 1 is negative (−) and 0 is positive (+).

sign extension—extending the sign bit of a two's complement number to the left by a duplication.

sign flag—a bit in the microprocessor used to record the sign of the result of a machine-language operation.

sign-magnitude—a nonstandard way of representing positive and negative numbers in microcomputers.

signed numbers—numbers that may be either positive or negative.

significant bits—the number of bits in a binary value after leading zeros have been removed.

significant digit—a digit that contributes to the precision of a number. The number of significant digits is counted beginning with the digit contributing the most value, called the most significant digit, and ending with one contributing the least value, called the least significant digit.

silicon controlled rectifier—SCR. A semiconductor rectifier that can be controlled; it is a pnpn four-layer semiconductor device that normally acts as an open circuit, but switches rapidly to a conducting state when an appropriate gate signal is applied to the gate terminal.

simulator—a computer that is programmed to mimic the action and functions of another piece of machinery, usually for training purposes. A computer is usually employed because it is cheaper to have the computer simulate these actions than to use the real thing. Airplane and power plant trainers are excellent examples.

sink—a device that drains energy off a system; a device that switches a load to an absorbing material, such as a ground.

software—refers to the programs that can be run on a computer.

solid state devices (semiconductors)—electronic components that control electron flow through solid materials such as crystals; e.g., transistors, diodes, integrated circuits.

SOS—silicon on sapphire. A semiconductor manufacturing technology in which metal oxide semiconductor devices are constructed in a thin single-crystal silicon film grown on an electrically insulating synthetic sapphire substrate.

source program—the program written in a language or mnemonics that is converted to machine code. The source program as well as the object code generated from it can be saved in mass storage devices.

special purpose logic—proprietary features of a programmable controller which allow it to perform logic not normally found in relay ladder logic.

SPOOL—acronym for Simultaneous Peripheral Output, On-Line. Used to overlap processing, typically, with printing.

stack—an area of memory used by the CPU and the programmer particularly for storage of register values during interrupt routines. See LIFO.

start-up—the time between equipment installation and the full operation of the system.

state—the logic 0 or 1 condition in programmable controller memory or at a circuit's input or output.

status register—the register that contains the status flags set and tested by the CPU operations.

stepper motor—a special motor in a disk drive that moves the read/write head a specific distance each time power is applied. That distance defines the tracks on a disk.

storage—see memory.

strip printer—a peripheral device used with a programmable controller to provide a hard copy of process numbers, status, and functions.

subroutine—a routine within a program that ends with an instruction to return program flow to where it was before the routine began. This routine is used many times from many different places in the program, and the subroutine allows you to write the code for that routine only once. Similar to a macro.

substrate—the physical material on which a microcircuit is fabricated; used primarily for mechanical support and insulating purposes; however, semiconductor and ferrite substrates may also provide useful electric functions.

subtract with carry—a machine-language instruction in which one operand is subtracted from another, along with a possible borrow from the next lower byte.

subtrahend—the number that is subtracted from the minuend.

successive addition—a multiplication method in which the multiplicand is added a number of times equal to the multiplier to find the product.

surge—a transient variation in the current and/or potential at a point in the circuit.

synchronous shift register—shift register which uses a clock for timing of a system operation and where only one state change per clock pulse occurs.

syntax—the term is used exactly as it is used in English composition. Every language has its own syntax.

system—a collection of units combined to work as a larger integrated unit having the capabilities of all the separate units.

system software—software that the computer must have loaded and running to work properly.

T

table—an ordered collection of variables and/or values, indexed in such a way that finding a particular one can be done quickly.

tape reader—a unit which is capable of sensing data from punched tape.

Teletype™—a peripheral electromechanical device for entering or outputting a program or data in either a punched paper tape or printed format.

termination—1) the load connected to the output end of a transmission line 2) the provisions for ending a transmission line and connecting to a bus bar or other terminating device.

text editor—see word processor.

thumbwheel switch—a rotating numeric switch used to input numeric information to a controller.

timer—in relay-panel hardware, an electromechanical device which can be wired and preset to control the operating interval of other devices. In the programmable controller a timer is internal to the processor, which is to say it is controlled by a user-programmed instruction. A timer instruction has greater capability than any hardware timer. Therefore, programmable controller applications do not require hardware timers.

time sharing—refers to systems which allow several people to use the computer at the same time.

track—a concentric area on a disk where data is stored in microscopic magnetized areas.

transducer—a device used to convert physical parameters, such as temperature, pressure, and weight into electrical signals.

translator package—a computer program which allows a user program (in binary) to be converted into a usable form for computer manipulation.

transistor—an active component of an electronic circuit consisting of a small block of semiconducting material to which at least three electrical contacts are made, usually two closely spaced rectifying contacts and one ohmic (non-rectifying) contact; it may be used as an amplifier, detector, or switch.

transistor-transistor logic—TTL. A logic circuit containing two transistors, for driving large output capacitances at high speed. A family of integrated circuit logic. (Usually 5 volts is high or 1 and 0 volts is low or 0; 5V = 1, 0V = 0).

Triac^{TM}—a General Electric trademark for a gate controlled semiconductor switch designed for alternating current power control; with phase control of the gate signal, load current can be varied over a range from 5 percent to 95 percent of full power.

truncation—the process of dropping bits to the right of a bit position. Truncating the binary fraction 1011.1011 to a number with fraction of two bits, for example, results in 1011.10.

truth table—a table defining the results for several different variables and containing all possible states of the variables.

TTL—see transistor-transistor logic.

TTY—an abbreviation for Teletype.

two's complement—a standard way of representing positive and negative numbers in microcomputers.

U

unsigned numbers—numbers that may be only positive; absolute numbers.

utility—a program designed to aid the programmer in developing other software.

UV erasable PROM—an ultraviolet erasable PROM is a programmable read-only memory which can be cleared (set to 0) by exposure to intense ultraviolet light. After being cleared, it may be reprogrammed.

V

variable—a labeled entity that can take on any value.

volatile memory—a memory that loses its information if the power is removed from it.

volt—the unit of potential difference or electromotive force in the meter-kilogram-second system, equal to the potential difference between two points for which 1 coulomb of electricity will do 1 joule of work in going from one point to the other.

voltage—potential difference or electromotive force capable of producing a current; measured in volts.

voltage drop—the voltage developed across a component or a conductor by the flow of current through the resistance or impedance of the component or conductor.

von Neumann, John (1903–1957)—Mathemetician. He put the concept of games, winning strategy, and different types of games into mathematical formulae. He also advanced the concept of storing the program in memory as opposed to having it on tape.

W

weighted value—the numerical value assigned to any single bit as a function of its position in the code word.

word—a grouping or a number of bits in a sequence that is treated as a unit and is stored in one memory location. If the CPU works with 8 bits, then the word length is 8 bits. Common word sizes are 4, 8, 12, 16, and 32. Some are as large as 128 bits.

word processor—a computer system dedicated to editing text and printing it in various controllable formats. See editor.

write—to store in memory or on a mass storage device.

X

XOR—a Boolean function. Acronym for eXclusive OR. Similar to OR but answer is high (1) if and only if one line is high.

Z

zero flag—a bit in the microprocessor used to record the zero/non-zero status of the result of a machine-language instruction.

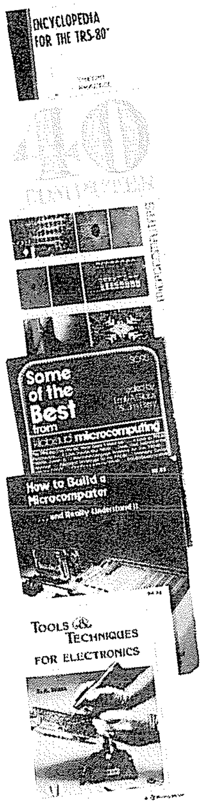
zero page—refers to the first page of memory.

INDEX

INDEX

INDEX COMPILED BY NAN MCCARTHY

WAYNE GREEN BOOKS



Encyclopedia for The TRS-80*—A ten-volume series to be issued every two months starting July 1981. The Encyclopedia contains the most up-to-date information on how to use your TRS-80*.

40 Computer Games from Kilobaud Microcomputing—Games in nine different categories for large and small systems, including a section on calculator games.

Understanding and Programming Microcomputers—A well-structured introductory text on the hardware and software aspects of microcomputing.

Some of the Best from Kilobaud Microcomputing—A collection of articles focusing on programming techniques and hardcore hardware construction projects.

How to Build a Microcomputer and Really Understand It—A technical manual and programming guide that takes the hobbyist step-by-step through the design, construction, testing and debugging of a complete microcomputer system (6502 chip).

Tools and Techniques for Electronics—Describes the safe and correct ways to use basic and specialized tools for electronic projects as well as specialized metal working tools and the chemical aids which are used in repair shops.

Annotated BASIC—A New Technique for Neophytes—Two volumes explaining the complexities of modern BASIC, including complete TRS-80* Level II BASIC programs. Each program is annotated and flowcharted to explain the workings of the program. By following the programs and annotation, you can develop new techniques to use in your own programs—or in modifying commercial programs for your specific use.

Kilobaud Klassroom—A Practical Course in Digital Electronics.—This popular series, first published in *Kilobaud Microcomputing*, combines theory with practice. It starts out with very simple electronics projects and, by the end of the course, you'll construct your own working microcomputer!

The New Weather Satellite Handbook—This handbook contains all the information on the most sophisticated spacecraft now in orbit. It is written to serve both the experienced amateur satellite enthusiast and the newcomer. The book is an introduction to satellite watching that tells you how to construct a complete ground station. An entire chapter is devoted to microcomputers and the Weather Satellite Station.

To order call Toll Free 800-258-5473.

*TRS-80 is a trademark of Radio Shack Division of Tandy Corp.

The real value of your computer lies in your ability to use it. The capabilities of the TRS-80* are incredible if you have the information which will help you get the most from it. Little of this information is available in your instruction books.

The *Encyclopedia for the TRS-80* will teach you how to get the most from your computer. In addition to a wealth of programs which are ready for you to use, reviews of accessories and commercially available programs, you will also learn how to write your own programs or even modify commercial programs for your own specific use.

The *Encyclopedia* is packed with practical information, written and edited for the average TRS-80 owner, not the computer scientist. You will find it interesting and valuable.

Wayne Green
Publisher

$10.95 ISBN 0-88006-041-7