A Collection of Information on The TI CC-40 Computer

by

Palmer O. Hanson, Jr.

Editor - TI PPC Notes

May 1986

This collection is a compilation of articles on the CC-40 and its peripherals which appeared in the 1985 issues of TI PPC Notes.

<u>A MATRIX INVERSION BENCHMARK</u> - P. Hanson. The paper "Mathematics Written in Sand" by W. Kahan in the 1983 <u>Proceedings of the Statistical Computing Section</u> of the American Statistical Association has been a rich resource for benchmark problems. See V9N2P15 and V9N4P6 for examples. Page 24 of the paper invites the reader to consider the 4x4 matrix A and its inverse:

$$A = \begin{vmatrix} 6 & -1 & -3 & 1 \\ -2 & 0 & 1 & 3 \\ 2 & -1 & 0 & 1 \\ -3 & 2 & -1 & 0 \end{vmatrix} \qquad A^{-1} = \begin{vmatrix} -5 & -6 & 23 & 9 \\ -11 & -13 & 50 & 20 \\ -7 & -8 & 31 & 12 \\ -1 & -1 & 5 & 2 \end{vmatrix}$$

Kahan presents the solution for the inverse from the HP-15C on page 25 of the paper. Those results are reproduced column by column below, together with results from the ML-02 program of the TI-59 Master Library, from the Matrix Inversion routine in the Mathematics module of the CC-40, and from the ML-02 program modified to include the double-divide workaround for the non-commutative multiply on pages 3-4 of this issue. Appropriate readout techniques were used to obtain thirteen significant digits for the TI-59 and CC-40 solutions. Without those techniques, the rounding to the display features of those devices makes it appear that they had all arrived at exact solutions.

Exact	HP-15C	ML-02	CC-40	ML-02 Plus
-5	-5.000000049	-5.000000000099	-4.999999999961	-4.999999999973
-11	-11.00000011	-11.00000000026	-10.99999999992	-10.99999999995
-7	-7.000000067	-7.000000000168	-6.999999999947	-6.999999999973
-1	-1.000000011	-1.000000000030	-0.9999999999925	-0.9999999999955
-6	-6.000000059	-6.000000000122	-5.999999999956	-5.999999999973
-13	-13.00000013	-13.00000000031	-12.99999999991	-12.99999999995
-8	-8.000000080	-8.000000000201	-7.999999999942	-7.999999999973
-1	-1.000000013	-1.000000000035	-0.9999999999917	-0.9999999999955
23	23.00000022	23.00000000044	22.99999999982	22.99999999989
50	50.00000048	50.00000000115	49.99999999962	49.99999999980
31	31.00000030	31.00000000075	30.99999999976	30.99999999988
5	5.000000048	5.000000000130	4.999999999966	4.999999999980
9	9.000000085	9.000000000155	8.999999999930	8.999999999957
20	20.00000019	20.00000000042	19.99999999985	19.99999999993
12	12.00000012	12.00000000027	11.99999999991	11.99999999995
2	2.000000019	2.000000000050	1.999999999987	1.999999999993

Relative error:

RMS	1.00e-08	2.41e-11	7.49e-12	4.29e-12
Max	1.30e-08	3.50e-11	8.30e-12	5.40e-12
Relative Inaccuracy	2331	5.6	1.7	1

where the relative inaccuracy was determined by the ratio of the RMS of the relative errors for the HP-15C, ML-02, and CC-40 routines to the relative error for the ML-02 Plus routine. For this problem the use of the double-divide technique provides a performance improvement by a factor of over five.

MATRIX INVERSION ON THE CC-40 - Palmer Hanson.

In the discussion of the matrix operations of the CC-40 Mathematics module in V8N5P14 I noted a an important deficiency in running the routines from the keyboard -- the output values are brought to the display in a manner such that the user cannot perform chain calculations on the result without reentering the displayed value. Any digits not displayed are lost. Subsequent pages in V8N5 contained demonstration programs calling the solution for simultaneous equations in the Mathematics module as subroutines from a user program. A note on V8N5P16 explained how to modify the demonstration programs to solve for the inverse of an input matrix. The HX-1000 Printer/Plotter was not available at the time, so the demonstration programs only provided for return of the solution values to the display.

I wanted to include the CC-40 results in the benchmark matrix inversion exercise on page 5 of this issue. When I used the matrix inversion program from the keyboard with the Printer/Plotter attached I encountered the same situation often encountered with the TI-59. The contamination of the solution due to roundoff errors did not extend into the displayed digits, so the printed solution appeared exact. The printout is at the upper right. I could verify that it was not exact using the techniques on V8N5P16, but I needed a way to print all the digits of the solution. I decided to write a matrix inversion program which would emulate the options of the operation from the keyboard, and which would also make maximum use of the routines from the module. The program listed on page 7 is the result. The printout from the program for the matrix inversion problem on page 5 is at the lower right.

```
MATRICES

0-Menu   1-Add    2-Mult...
3-Det.   4-1/A    5-AX=B...
0-Exit.

***** Inverse Matrix *****
Order= 4

A(1,1)= 0
A(1,2)=-1
A(1,3)=-3
A(1,4)= 1
A(2,1)=-2
A(2,2)= 0
A(2,3)= 1
A(2,4)= 3
A(3,1)= 2
A(3,2)=-1
A(3,3)= 0
A(3,4)= 1
A(4,1)=-3
A(4,2)= 2
A(4,3)=-1
A(4,4)= 0

Determinant= 1.

X(1,1)=-5.
X(1,2)=-6.
X(1,3)= 23.
X(1,4)= 9.
X(2,1)=-11.
X(2,2)=-13.
X(2,3)= 50.
X(2,4)= 20.
X(3,1)=-7.
X(3,2)=-8.
X(3,3)= 31.
X(3,4)= 12.
X(4,1)=-1.
X(4,2)=-1.
X(4,3)= 5.
X(4,4)= 2.
```

```
***** Matrix Inversion *****
Order =        4
A(1,1)= 0
A(1,2)=-1
A(1,3)=-3
A(1,4)= 1
A(2,1)=-2
A(2,2)= 0
A(2,3)= 1
A(2,4)= 3
A(3,1)= 2
A(3,2)=-1
A(3,3)= 0
A(3,4)= 1
A(4,1)=-3
A(4,2)= 2
A(4,3)=-1
A(4,4)= 0

C(1,1) =  -.4999999999901E+01
C(1,2) =  -.5999999999950E+01
C(1,3) =   .2299999999982E+02
C(1,4) =   .8999999999930E+01
C(2,1) =  -.1099999999992E+02
C(2,2) =  -.1299999999991E+02
C(2,3) =   .4999999999962E+02
C(2,4) =   .1999999999985E+02
C(3,1) =  -.6999999999947E+01
C(3,2) =  -.7999999999942E+01
C(3,3) =   .3099999999970E+02
C(3,4) =   .1199999999991E+02
C(4,1) =  -.9999999999925E+00
C(4,2) =  -.9999999999917E+00
C(4,3) =   .4999999999960E+01
C(4,4) =   .1999999999987E+01
```

Program Description:

The program has a full set of prompts which emulates operation from the keyboard.

Line 105 - The IMAGE statement sets up a nine character string field for annotation of the output, and an exponential field to display thirteen significant digits and the exponent. Line 230 prints the output using the format set by the IMAGE statement.

Line 115 - The UP routine from pages 100-101 of the Mathematics module manual is used to select either the printer or the display for output. For output to the display the routine sets PN = 0. For output to the printer file #1 is opened for the device number entered by the user (10 for the HX-1000) and PN = 1. The value of PN is used as a branching control through the remainder of the program.

Matrix Inversion on the CC-40 (cont)

Line 120 - The WR routine from page 101 of the manual is called to print the message "Matrix Inversion" with five asterisks at either side.

Line 150 - The MI routine from pages 95-96 is used to enter and edit the elements of the input matrix. Prompting is provided exactly like that which is available when running the matrix routines from the keyboard. If PN = 1 the elements are printed with appropriate annotation. Note that the CC-40 accepts the elements row by row, not column by column as with the TI-59.

Line 170 - The MATS routine from page 94 of the manual is used to solve for the inverse. As explained on V8N5P14 the inverse appears in matrix C, properly located for readout in sequence.

Lines 200-250 - These statements control the output of the inverse to either the printer or the display depending on the value of PN. As with the routine when run from the keyboard, the output is row by row. It would have been convenient to be able to use the output routine from the module, but I have not yet been able to identify an appropriate call.

```
100 DIM A(8,8),B(8
),C(8,8)
105 IMAGE ########
# #.############^
^^^
110 PRINT "Matrix
Inversion":PAUSE 2
115 CALL UP("MI",P
N)
120 IF PN=1 THEN C
ALL WR("Matrix Inv
ersion",1)
130 INPUT "Enter o
rder of matrix:  "
;N
140 IF PN=1 THEN P
RINT #1,"Order = "
,N
150 CALL MI("A",A(
,),1,N,N,PN)
160 PRINT "Solving
"
170 CALL MATS(A(,)
,C(,),B(),1,1,4,0,
N,0,R)
200 FOR I=1 TO N
210 FOR J=1 TO N
220 A$="C("&STR$(I
)&","&STR$(J)&") =
 "
230 PRINT #PN,USIN
G 105,A$,C(I,J)
240 IF PN=1 THEN 2
50 ELSE PAUSE
250 NEXT J:NEXT I
900 CLOSE #1
999 END
```

A NEW ANOMALY WITH THE CC-40 MATHEMATICS MODULE

Examine the upper printout on page 6. The determinant for the matrix is shown as positive 1. The determinant output from ML-02 on the TI-59 is negative 1, which agrees with my hand determination of the determinant. So far I have not found another determinant with a wrong sign, but it seems that we must remain unsure about the sign of any output of the determinant from the matrix routines of the CC-40.

ERRATA:

Sum of Log(x) Formula for the Geometric Mean - The equation for the geometric mean on page 23 of the Statistics cartridge manual for the CC-40 is incorrect. The correct formula is

$$\bar{x}_g = 10^{\left(\sum_{i=1}^{n} f_i \cdot \log(x_i)\right)/N}$$

THE LOGARITHM ALGORITHM FOR THE CC-40 - Louis Krumpleman of Richmond, Kentucky writes that he has disassembled about eighty per cent of the internal code of the CC-40. An example is the Ln algorithm:

1. The input argument is converted to the form $m \times 10^n$ where $0.1 < m < 1.0$. In CC-40 BASIC this may be accomplished by finding n = INT(Log(X)) + 1, and then m = X/(10^N).

2. $m' = \sqrt{10} \times m$, where

   $\sqrt{10} = 3.162277660168$

3. $t = (m' - 1)/(m' + 1)$

4. $z' = t \times SA / SB$, where

   $SA = ((A1 \cdot t^2 + A2)t^2 + A3)t^2 + A4$

   $SB = (((B1 \cdot t^2 + B2)t^2 + B3)t^2 + B4)t^2 + B5$

   and the constants are

   A1 = -22.764761571152
   A2 = 197.6446297035
   A3 = -429.4834828658
   A4 = 265.5224908516
   B1 = 1.0
   B2 = -31.416484482822
   B3 = 158.6018962727
   B4 = -258.9954899200
   B5 = 132.7612454259

5. $Ln(X) = Ln(10) \times (n - 0.5) + z'$, and

   $Ln(10) = 2.302585092994$

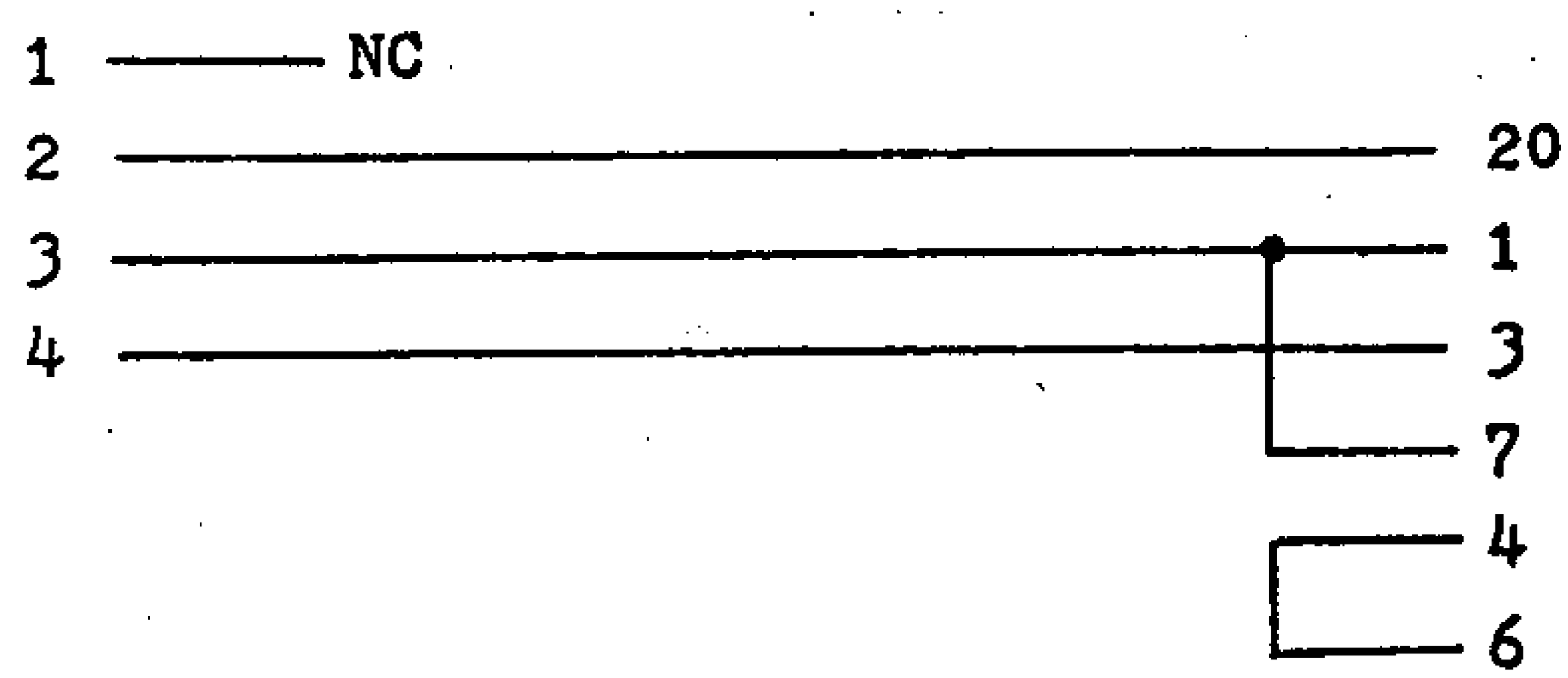
```
100 OPEN #1,"10,S=0",OUTPUT
105 A1=-22.764761571152
110 A2=197.6446297035
115 A3=-429.4834828658
120 A4=265.5224908516
125 B1=1
130 B2=-31.416484482822
135 B3=158.6018962727
140 B4=-258.99548992
145 B5=132.7612454259
150 S10=3.162277660168
155 E10=2.302585092994
190 X=0
200 X=X+.1
205 N=INT(LOG(X))+1
210 M=X/(10^N)
215 MP=S10*M
220 T=(MP-1)/(MP+1)
225 T2=T*T
230 SA=((A1*T2+A2)*T2+A3)*T2+A4
235 SB=(((B1*T2+B2)*T2+B3)*T2+B4)*T2
+B5
240 ZP=T*SA/SB
245 Y=E10*(N-.5)+ZP
250 REM PRINT USING"#.############^^^^"
;Y:PAUSE
255 YB=LN(X)
260 REM PRINT USING"#.############^^^^"
;YB:PAUSE
270 IF Y=YB THEN GOTO 200
280 PRINT #1,X,Y-YB
300 GOTO 200
390 CLOSE #1
999 END
```

Editor's Note: Louis observes that this looks like a Hastings or Abramowitz approximation. To test whether it is truly the internal Ln algorithm I wrote the little CC-40 program at the right above. It calculates Ln(X) using Louis' algorithm in BASIC, compares the result with Ln(X) using the internal algorithm, and prints the result if the two values are not equal. For the range of x from 0.1 through 100 in 0.1 step, only 94 of the 1000 values did not match exactly, and the largest difference was 1.E-12 . If you wish to view the values calculated remove the REM at steps 250 and 260. Then the program will stop with each calculated value in the display, and you press CLEAR to continue.

ANY PC-200'S OUT THERE? - In response to a telephone call in June a TI representative indicated that PC-200's, the printers for use with the BA-55 and TI-66, should start appearing on retail shelves in limited quantities at mid-year. So far I haven't seen any in the Tampa Bay area. Has anyone seen them elsewhere?

LOAN PAYMENT SCHEDULE FOR THE CC-40 - This little BASIC program was originally written by my long time friend Merle Lundeen for the Radio Shack Model III. I had previously modified the program for use with the Radio Shack Model 100. The program on the next page is a conversion for use with the CC-40, the HX-3000 RS-232 peripheral, and the Radio Shack Model VII line printer. With no experience in RS-232 interfacing I tried to use the the connections described on page 2 of the Jan/Feb/Mar 1984 issue of TISOFT for connecting the Radio Shack CGP-115 four color plotter to the TI-99/4A RS-232 output. The manuals for the Model VII and the CGP-115 suggest that their interfaces are similar, but I could not obtain any printing. I then added a jumper between pins 4 and 6 at the output of the HX-3000 and obtained some printing, but there was still a problem. If two PRINT #1 statements occurred too closely together in time then the second statement would not perform properly. After some experimenting I found a workaround (I love workarounds, see V9N2P17 for another). I simply added a delay loop after each PRINT #1 statement. The interconnect cabling between the Model VII and the HX-3000 was:

DIN Plug (Model VII) DB-25 Plug (HX-3000)



Program Description:

The program provides a full set of prompts at the CC-40 display.

Line 10 - Assigns a file number for output to the HX-3000, and matches the RS-232 output from the HX-3000 with the input to the Model VII.

Line 11 - Selects double width printing on the Model VII.

Line 12 - Prints the heading in double width letters.

Line 13 - Returns the Model VII to normal printing.

Lines 20-24 - Prompts are provided for the user to enter the principal, the interest, and the number of monthly payments.

Lines 30-34 - Print out of the input data with annotation.

Lines 40-50 - Calculate and print the monthly payment.

Lines 52-72 - Print the table of payments.

Lines 74-80 - Calculate and print the total payments.

Line 90 - The delay loop used to obtain proper printing. The subroutine is called immediately after each PRINT #1 statement; for example, see lines 11 through 13.

Lines 95-98 - IMAGE statements for use with with the PRINT #1 Using statements at lines 30, 50, 80, and 70.

A reduced printout for a sample problem appears on the next page.

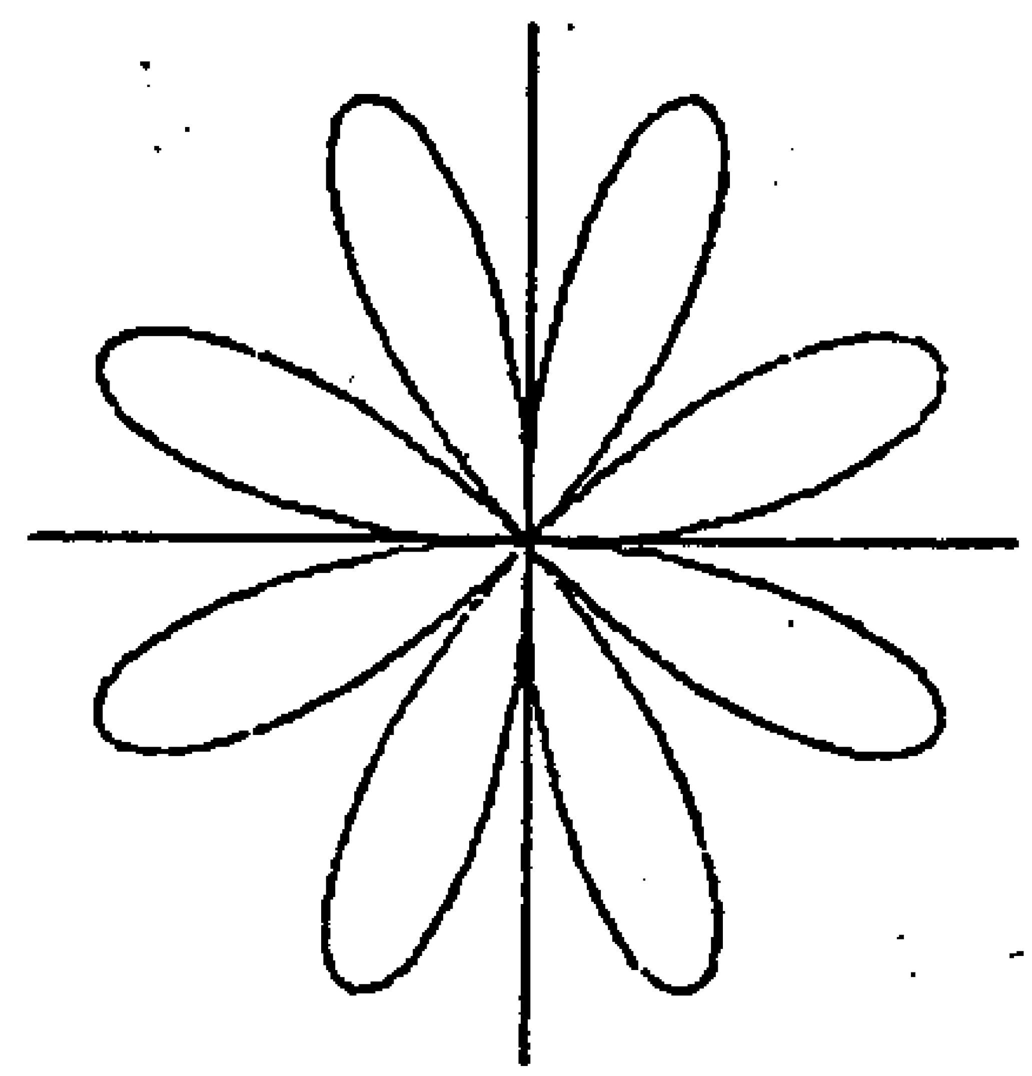
Loan Payment Schedule for the CC-40 (cont)

```
10 OPEN #1,"20.B=600,P=N,E=N,S=2,N=99,D=7",OUTPUT
11 PRINT #1,CHR$(31):GOSUB 90
12 PRINT #1,"  ***   LOAN PAYMENT SCHEDULE   ***":GOSUB 90
13 PRINT #1,CHR$(30):GOSUB 90
20 INPUT "Principal ? ";P
22 INPUT "Rate (%) ? ";R
24 INPUT "Number of Months ? ";N
30 PRINT #1,USING 95;P:GOSUB 90
32 PRINT #1,"  RATE (%)            ";R:GOSUB 90
34 PRINT #1,"  TERM (Months)       ";N:GOSUB 90
40 RP=R/1200
42 W=(1+RP)^N
44 M=P*RP*W/(W-1)
46 M=INT(100*M)/100
50 PRINT #1,USING 96;M:GOSUB 90
52 PRINT #1:GOSUB 90
56 PRINT #1,"  PAYMENT        BALANCE        INTEREST        ACCRUED INTEREST"
58 GOSUB 90
60 PRINT #1:GOSUB 90
62 FOR I=1 TO N
64 IN=INT(P*RP*100)/100
66 IS=IS+IN
68 P=P-(M-IN)
70 PRINT #1,USING 98,I,P,IN,IS:GOSUB 90
72 NEXT I
74 TP=M*N+P
76 PRINT #1:GOSUB 90
80 PRINT #1,USING 97,TP:GOSUB 90
85 CLOSE #1:STOP
90 FOR J=1 TO 1000:NEXT J:RETURN
95 IMAGE "  PRINCIPAL       $########.##"
96 IMAGE "  MONTHLY PAYMENT    $#####.##"
97 IMAGE "  TOTAL PAYMENTS    $########.##"
98 IMAGE "    ##      $########.##      $#####.##          $#######.##"
99 END
```

```
***      LOAN PAYMENT SCHEDULE      ***

PRINCIPAL        $    1000.00

RATE (%)               12.5

TERM (Months)          18

MONTHLY PAYMENT    $  61.21


PAYMENT        BALANCE        INTEREST        ACCRUED INTEREST

   1       $     949.20      $  10.41        $      10.41
   2       $     897.87      $   9.88        $      20.29
   3       $     846.01      $   9.35        $      29.64
   4       $     793.61      $   8.81        $      38.45
   5       $     740.66      $   8.26        $      46.71
   6       $     687.16      $   7.71        $      54.42
   7       $     633.10      $   7.15        $      61.57
   8       $     578.48      $   6.59        $      68.16
   9       $     523.29      $   6.02        $      74.18
  10       $     467.53      $   5.45        $      79.63
  11       $     411.19      $   4.87        $      84.50
  12       $     354.26      $   4.28        $      88.78
  13       $     296.74      $   3.69        $      92.47
  14       $     238.62      $   3.09        $      95.56
  15       $     179.89      $   2.48        $      98.04
  16       $     120.55      $   1.87        $      99.91
  17       $      60.59      $   1.25        $     101.16
  18       $        .01      $    .63        $     101.79

TOTAL PAYMENTS     $    1101.79
```

POLAR COORDINATE PLOTTING WITH THE CC-40/HX-1000 - Palmer Hanson

Pages 12 and 13 illustrate the kind of techniques which must be used for plotting on a unidirectional printer if the coordinates to be plotted along the direction of paper motion do not increase monotonically. Printer-plotters such as the HX-1000 or the Radio Shack CGP-115 permit the paper to be moved both forward and back under computer control. One result of the increased capability is that functions defined by polar coordinates can be plotted directly without intermediate storage to sort the along paper coordinates. The plots at the right on page 1 are examples of the kind of plots which can be obtained with the CC-40/HX-1000 using the graphics mode. An enlarged copy of one of the plots and the program used to obtain it appears at the right.



```
100 OPEN #1,"10",OUTPUT
110 PRINT #1,CHR$(19)
120 PRINT #1,"L(0,-107):1,(215,-107)
"
130 PRINT #1,"L(107,0),(107,-215)"
140 PRINT #1,"M(107,-107)"
150 PRINT #1,"O"
160 PRINT #1,"C4"
190 X1=0:Y1=0
200 FOR THETA=0 TO 360 STEP 2
210 RHO=100*SIN(4*THETA)
220 X2=INT(RHO*COS(THETA))
230 Y2=INT(RHO*SIN(THETA))
240 P$="L("&STR$(X1)&","&STR$(Y1)&")
,("&STR$(X2)&","&STR$(Y2)&")"
250 PRINT #1,P$
260 X1=X2:Y1=Y2
270 NEXT THETA
900 CLOSE #1
999 END
```

Program Comments:

Line 110 - CHR$(19) sets graphics mode.

Lines 120-130 draw the x and y axes.

Lines 140-150 move the pen to the intersection of the x and y axes and define that point as the origin for further plotting.

Line 160 changes the pen color to red.

Line 190 defines the origin as the first point for the draw command.

Lines 210-230 calculate the second point for the draw command. Since $\rho = \sin(4\theta)$ this will be an eight leaved rose.

Line 240 converts the first and second points for the first segment of the function plot into a string command which can be used by the printer. There are two important considerations when generating the string command: (1) the coordinates of the first and second points must be integers. If these arguments are not integers the computer will return the message "I/O error 80 #1". The integer functions in lines 220 and 230 ensure that requirement is met; and (2) this method of generating a string command is the only way that variable points can be plotted.

Line 250 delivers the assembled string command to the printer. The printer responds by drawing a line segment between the first and second points in the string command.

Line 260 changes the end point of the first line segment into the beginning point for the second line segment.

Line 270 sends the computer back to calculate the next end point. With this technique a four leaved rose can be plotted in about three minutes.

PARANOIA - George Thomson writes "Did you see Karpinski's article in BYTE about a program 'Paranoia'? I sent away as directed and got the IEEE Draft of the FP standards and one of the most horrendous programs you could imagine. The disk is called 'Paranoia' and it is all about finding out more than you'll ever know about internal computer arithmetic and whether there are 'Serious Flaws' or 'Defects'......"

The article George mentions, "Paranoia: A Floating-point Benchmark" by Richard Karpinski, appeared on pages 223 through 235 of the February 1985 issue of BYTE. The author of the program is William Kahan, who also wrote the "Mathematics Written in Sand ..." paper which has been discussed at length in earlier issues of TI PPC Notes (for example, see V9N2P15). Once again, the emphasis is on IEEE arithmetic and the proper use of guard digits. The complete Paranoia program is some 700 lines of BASIC, and the article contains instructions for obtaining a copy. The article also includes a limited version which tests for the use of a guard digit in addition and subtraction, what George Thomson calls "an itsy-bit of Paranoia".

The printout and program at the right are the implementation of the "itsy-bit" on the CC-40, where the only changes were extensive censoring of the comments. The output was much as we would have expected from our previous discussions of the number representation in the CC-40 (see V9N5P6), seven radix 100 digits. The program indicates that the CC-40 does have an add/subtract guard digit.

Testing the other computers that I have available was not quite so straightforward. Consider statement 30 at the right. BASIC implementations such as in the Model 100, Commodore 64, etc., do not permit the use of the ON combination of letters in a variable name. Other BASIC implementations on other computers have other lists of reserved words. Now consider statements 480 and 790 in the listing at the right which define the variables RADIX and RADIXMINUS. Many versions of BASIC allow the user to use several letters in the variable name, but only the first two characters are used by the computer. In such implementations RADIX and RADIXMINUS are seen as the same variable. Not surprisingly, the listing at the right will not operate properly in those computers.

```
Radix =  100
Precision =  7
Fpwidth =  1.E+14
Ulpone =  1.E-14
Add/subtract has a
 guard digit
```

```
10 OPEN #1,"10",OUTPUT
30 ONE=1
40 HALF=.5
50 ZERO=0
60 MINUSONE=-1
290 WIDE=ONE
310 WIDE=WIDE+WIDE
320 X=WIDE+ONE
340 Y=X-WIDE
350 Z=Y-ONE
370 IF (MINUSONE+ABS(Z))<ZERO THEN 3
10
400 Y=ONE
480 RADIX=WIDE+Y
490 Y=Y+Y
500 RADIX=RADIX-WIDE
520 IF RADIX=ZERO THEN 480
540 PRINT #1,"Radix = ";RADIX
590 PRECISION=ZERO
600 FPWIDTH=ONE
620 PRECISION=PRECISION+ONE
630 FPWIDTH=FPWIDTH*RADIX
640 Y=FPWIDTH+ONE
660 IF (Y-FPWIDTH)=ONE THEN 620
680 PRINT #1,"Precision = ";PRECISIO
N
700 PRINT #1,"Fpwidth = ";FPWIDTH
720 ULPONE=ONE/FPWIDTH
740 PRINT #1,"Ulpone = ";ULPONE
760 ONEMINUS=(HALF-ULPONE)+HALF
770 ULPRADIX=RADIX*ULPONE
790 RADIXMINUS=RADIX-ONE
800 RADIXMINUS=(RADIXMINUS-ULPRADIX)
+ONE
820 X=ONE-ULPONE
830 Y=ONE-ONEMINUS
840 Z=ONE-X
860 S=RADIX-ULPRADIX
870 T=RADIX-RADIXMINUS
880 U=RADIX-S
900 IF Y=ULPONE THEN 920
910 GOTO 960
920 IF T=ULPRADIX AND U=ULPRADIX THE
N 940
930 GOTO 960
940 PRINT #1,"Add/subtract has a gua
rd digit"
950 GOTO 980
960 PRINT #1,"Add/subtract lacks gua
rd digit"
980 CLOSE #1
990 END
```

Paranoia - (cont)

It was my unhappy experience with this kind of thing with FORTRAN in the mid-1960's that led to the formulation of Hanson's First Law of Higher Order Languages:

"NO PROGRAM WRITTEN FOR ONE COMPUTER WILL RUN AS IS ON ANY OTHER"

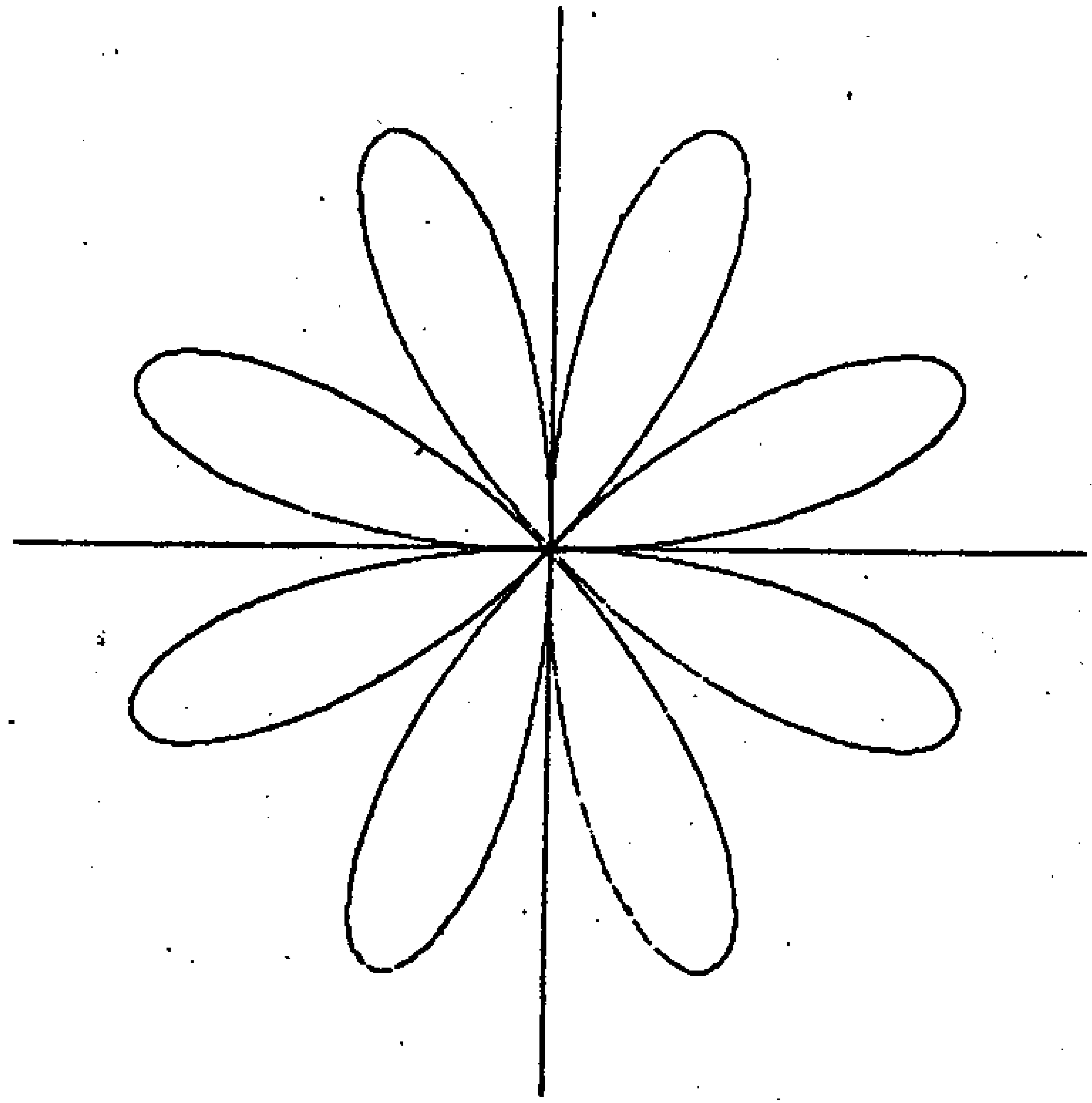
While it is possible to write programs which will travel well, very few such programs are written. The listing below makes the required changes in variable names to permit the "itsy-bit of Paranoia" to run on the Radio Shack Model 100.

```
Radix =  10
Precision =  14
fpwidth =  1E+14
Closest relative separation found is UlpOne =  1E-14
Add/subtract has a guard digit as it should.


30 WUN =        1.0
40 HALF =       0.5
50 ZERO =       0.0
60 NEGWUN = -1.0
290 WIDE = WUN
310 WIDE = WIDE + WIDE
320 X = WIDE + WUN
340 Y = X - WIDE
350 Z = Y - WUN
370 IF (NEGWUN + ABS(Z)) ( ZERO THEN 310
460 Y = WUN
480 RADIX = WIDE + Y
490 Y = Y + Y
500 RADIX = RADIX - WIDE
520 IF RADIX = ZERO THEN 480
540 LPRINT "Radix = ";RADIX
590 PRECIS = ZERO
600 FPWIDT = WUN
620 PRECIS = PRECIS + WUN
630 FPWIDT = FPWIDT * RADIX
640 Y = FPWIDT + WUN
660 IF (Y-FPWIDT) = WUN THEN 620
680 LPRINT "Precision = ";PRECIS
700 LPRINT "fpwidth = ";FPWIDT
720 ULPWUN = WUN/FPWIDT
740 LPRINT "Closest relative separation found is UlpOne = ";ULPWUN
760 MINWUN = (HALF - ULPWUN) + HALF
770 URADIX = RADIX * ULPWUN
790 MRADIX = RADIX - WUN
800 MRADIX = (MRADIX - URADIX) + WUN
820 X = WUN - ULPWUN
830 Y = WUN - MINWUN
840 Z = WUN - X
860 S = RADIX - URADIX
870 T = RADIX - MRADIX
880 U = RADIX - S
900 IF Y = ULPWUN THEN 920
910 GOTO 960
920 IF T = URADIX AND U = URADIX THEN 940
930 GOTO 960
940 LPRINT "Add/subtract has a guard digit as it should."
950 GOTO 980
960 LPRINT "Add/subtract lacks guard digit, cancellation obscured."
980 END
```

Again, as we expected, the program finds that the Model 100 uses fourteen radix 10 digits, and has an add/subtract guard digit. The listing above travels well, because I carefully chose the variable names. With appropriate changes in the output commands (You ALWAYS have to change the output commands when going from one computer to the next.) the program has been run on the Radio Shack Color Computer, the Commodore 64, and the Sharp EL-5500II. The Color Computer and the Commodore 64 use 32 radix 2 digits. That probably explains why the Color Computer generates least significant digit "garbage" when one tries to strip off the higher order digits in the manner that we use to view the guard digits on the TI-59 (see V8N5P10). My tests show that the Color computer has a gurad digit but the Commodore does not.

THE RADIO SHACK CGP-115 AND THE HX-3000 RS-232 INTERFACE FOR THE CC-40

You may wonder at my interest in the CGP-115. The full size figure at the right should help you understand. The mode of operation is very similar to that of the HX-1000 Printer/Plotter, but the paper is twice as wide. Maurice Swinnen and the editors of TISOFT have some marvelous programs for the 99 operating with the CGP-115; but so far neither Maurice or I have been able to get the CC-40 and HX-3000 RS-232 Interface to operate satisfactorily with the CGP-115. I tried the delay trick that worked for the Radio Shack Model 7 Printer (V10N1P8), but it didn't help. Before the next issue I hope to have the HX-3000 modified for the parallel interface, and will try that way to connect the CC-40 to the CGP-115.



Any ideas would be appreciated.

A USEFUL INPUT DATA EFFECT WITH SOME COMPUTERS - Larry Leeds. V9N5P7 reported that if numbers which exceeded the length of the mantissa of the storage word were entered into CC-40, either from the keyboard or from a program, then the computer would examine the extra digits to see if the lowest digit of the mantissa to be stored should be rounded. Appropriate adjustment of the exponent would also occur. Similar effects were found with other computers.

Larry reports that his Radio Shack Model 100 will also accept blanks in the sequence of figures in the display, or in program statements. This permits the use of the blanks to make the entry of numbers with many digits more readable. For example, the statement:

```
X = 111 222 333 + 5
```

will yield x = 111222338, and the statement

```
y = 0.000 000 000 000 123 456
```

will yield x = 1.23456E-13. Tests show that the Radio Shack Color Computer and the Commocore 128 respond similarly. An attempt to introduce blanks with the CC-40 leads to the "Illegal Syntax" diagnostic message.

SORTING ON THE CC-40 - V8N6P21 discussed the Shell sort subprogram in the Statistics cartridge for the CC-40. A sample program provided prompting for data entry, callup of the subprogram, and display of the results. The program would sort 60 random numbers in 31 seconds. The TI-59 using the Shell sort from the Math/Utilities module takes 295 seconds.

There are other sorting routines than Shell sort. An examination of some of them was triggered by Albert Nijenhuis' article "A Confusion of Sorts" in the June 1985 issue of Creative Computing. For comparison I wrote a program which has five options for sorting:

1. An old "bubblesort" routine which I wrote many years ago while learning BASIC programming.

2. An "insertion sort" routine adapted from an earlier Nijenhuis article in the August 1980 issue of Creative Computing.

3. The Shell sort which is included with the Statistics cartridge for the CC-40.

4. A "heapsort" routine adapted from the program in a Nijenhuis' article in the September 1980 issue of Creative Computing.

5. An "address sort" routine which doesn't use comparison techniques, but rather uses array processing to order a set of input integers.

As expected the bubblesort program was by far the slowest--three times slower than the insertion sort and five times slower than the heapsort. Bubblesort and insertion sort run about a factor of two slower for the "worst case" set of an inverted list. Shell sort actually runs substantially faster with the inverted list than with random numbers. Heapsort execution time is essentially the same for random numbers or an inverted list. The address sort is by far the fastest, but the user must live with some limitations:

* Only integers can be sorted.

* Two one dimensional arrays are needed. Thus for a given memory size the address sort can only accept about half as many elements as the remaining methods.

Sorting on the CC-40 - (cont)

Program Comments:

Line 110 - The call to the UP subprogram with the arguments listed will display the message in quotations for about three seconds and then display the prompt "Use Printer?". A response of "N" will set the variable PN to zero, and proceed to the next line of the program. A response of "Y" will set the variable PN to one and display the prompt "Enter Filename:". If you enter a "10" for the HX-1000 the subprogram will open file #1 for output, set the printer to the 32 character per line option, print the message in the first part of the argument of the UP subroutine, and proceed to the next program line.

Lines 120 through 170 provide selection from three options for date entry.

Lines 200 through 230 provide prompts and controls for data entry from the keyboard. The use of the IF NOT NUMERIC function in line 222 provides an easy way to respond if the input string is not a valid number.

Lines 235 through 280 provide prompts and controls for data input from DATA statements. The data statements must have been previously entered. The ON WARNING and CALL ERROR statements provide automatic sensing of the end of data. As written the program will read all the DATA statements in the program. The idea for the use of DATA statements for data entry was obtained from Codeworks (see page 4 of this issue).

Lines 285 either prints or displays the number of input data points depending on the value of PN set in line 110.

Lines 290 through 298 provide an option of printing (or displaying without a printer) the input data points.

Lines 300 through 340 provide options for the method of sorting.

```
100 DIM X(300),Y(300)
110 CALL UP("Sorting Demonstration",
PN)
120 A$="Data Options: "
130 PRINT A$&"1 = Keyboard":PAUSE 1
140 PRINT A$&"2 = Data File":PAUSE 1
150 PRINT A$&"3 = Test Numbers":PAUS
E 1
160 INPUT "Which Input Option (1-3)?
 ";OP
170 ON OP GOTO 200,235,1000
200 REM Input from Keyboard
205 PRINT "Press <C> to End Input":P
AUSE 2
210 N=0
215 INPUT "X("&STR$(N+1)&") = ";X$
218 IF X$="C"OR X$="c"THEN 285
222 IF NOT NUMERIC(X$)THEN PRINT "In
valid Entry: ";:GOTO 215
225 X(N+1)=VAL(X$)
230 N=N+1:GOTO 215
235 REM Input from Data File
240 ON WARNING ERROR:ON ERROR 265
245 N=0
250 READ X(N+1)
255 N=N+1
260 GOTO 250
265 ON ERROR 990
270 CALL ERR(E,T,F,F1)
275 IF E<>43 THEN 990
280 RETURN 285
285 PRINT #PN,"Number of Data Points
 = ";N:PAUSE 2
290 INPUT "Print the Input Data (Y/N
)? ";Q$
295 IF Q$="N"OR Q$="n"THEN 300
298 GOSUB 1100
300 A$="Options: "
305 PRINT A$&"1 = Bubble Sort":PAUSE
 1
310 PRINT A$&"2 = Insertion Sort":PA
USE 1
315 PRINT A$&"3 = Shell Sort":PAUSE
1
320 PRINT A$&"4 = Heap Sort":PAUSE 1
325 PRINT A$&"5 = Address Sort":PAUS
E 1
330 INPUT "Which Sorting Option (1-5
)? ";OP
335 PRINT "Sorting"
340 ON OP GOSUB 400,500,600,700,900
345 DISPLAY BEEP
350 INPUT "Print the Sorted Data (Y/
N)? ";Q$
355 IF Q$="N"OR Q$="n"THEN 900
360 GOSUB 1100
370 GOTO 900
400 REM Bubble Sort
410 FOR I=1 TO N-1
420 IF X(I+1)>=X(I)THEN 480
430 A=X(I+1)
440 X(I+1)=X(I)
450 X(I)=A
460 I=I-2
470 IF I<0 THEN I=0
480 NEXT I
490 RETURN
500 REM Insertion Sort
510 FOR J=1 TO N-1
520 B=X(J+1)
530 FOR I=J TO 1 STEP -1
540 IF B>=X(I)THEN 580
550 X(I+1)=X(I)
560 NEXT I
570 I=0
580 X(I+1)=B
590 NEXT J
595 RETURN
```

Sorting with the CC-40 - (cont)

Lines 345 through 370 indicate that the sorting is complete with a "beep" and provide an option of printing (or displaying without a printer) the sorted data points.

Lines 400 through 490 are an elementary bubble sort subroutine that I wrote in an old programming class.

Lines 500 through 595 are an insertion sort subroutine adapted from the program on page 37 of the August 1980 issue of Creative Computing.

Lines 600 through 620 call the Shell sort subroutine of the Statistics cartridge.

Lines 700 through 796 are a heapsort subroutine adapted from the program on page 137 of the September 1980 issue of Creative Computing.

Lines 800 through 895 are a subroutine for sorting using indirect address techniques.

Lines 900 through 999 close file #1 if it was opened at line 110, and provide program ending.

Lines 1000 through 1040 provide two options for generating test numbers. The random option provides 100 random integers in the range from 1 to 100. The inverse option provides the integers from 1 to 100 in reverse order.

Lines 1100 through 1175 provide a subroutine for display or printout of the input and output data. Use of PRINT #PN statements with PN set by the UP subroutine in line 110 in line provides automatic control of whether the output is to the display or the printer.

Lines 5000 through 5030 contain the DATA statements for the 26 test integers used to test the heapsort in the September 1980 issue of Creative Computing.

```
600 REM Shell Sort with Statistics M
odule
610 CALL SORT(X(),N)
620 RETURN
700 REM Heap Sort
705 M=N
710 FOR L=INT(N/2)TO 1 STEP -1
715 B=X(L)
720 GOSUB 760
725 NEXT L
730 L=1
735 FOR M=N-1 TO 1 STEP -1
740 B=X(M+1):X(M+1)=X(1)
745 GOSUB 760
750 NEXT M
755 RETURN
760 I=L
765 J=I+I
770 IF J>M THEN 794
775 IF J=M THEN 785
780 IF X(J+1)>X(J)THEN J=J+1
785 IF B>=X(J)THEN 794
790 X(I)=X(J):I=J
792 GOTO 765
794 X(I)=B
796 RETURN
800 REM Address Sort
805 M=N:IM=0
810 FOR I=1 TO M
815 Y(X(I))=Y(X(I))+1
820 IF X(I)>IM THEN IM=X(I)
830 NEXT I
840 M=N:L=1
850 FOR I=1 TO IM
860 FOR J=1 TO Y(I)
870 IF Y(I)=0 THEN 890
875 X(L)=I
880 L=L+1
885 NEXT J
890 NEXT I
895 RETURN
900 IF PN=1 THEN CLOSE #1
998 STOP
999 END
1000 PRINT "Options for 100 Test Num
bers:":PAUSE 2
1005 PRINT "Press 1 for Random Integ
ers, or":PAUSE 2
1010 INPUT "Press 2 for an Inverted
List:";Q
1015 FOR I=1 TO 100
1020 IF Q=1 THEN X(I)=INTRND(100)
1025 IF Q=2 THEN X(I)=101-I
1030 NEXT I
1035 N=100
1040 GOTO 285
1100 REM Printing Subroutine
1105 PRINT #PN:IF PN=0 THEN 1160
1110 I=1
1115 FOR J=0 TO 4
1120 PRINT #PN,X(I+J);
1125 IF I+J=N THEN 1150
1130 NEXT J
1135 IF PN=0 THEN PAUSE
1140 PRINT #PN
1145 I=I+5:GOTO 1115
1150 IF PN=0 THEN PAUSE
1155 PRINT #PN:RETURN
1160 FOR I=1 TO N
1165 PRINT X(I):PAUSE
1170 NEXT I
1175 RETURN
5000 DATA 19,8,18,15,14,25
5010 DATA 16,10,12,13,3,26
5020 DATA 24,17,20,9,4,7
5030 DATA 11,6,5,2,1,21,23,22
```

Sorting on the CC-40 - (cont)

```
Number of Data Points =  26

19  8  18  15  14
25  16  10  12  13
3  20  24  17  26
9  4  7  11  6
5  2  1  21  23
22

1  2  3  4  5
6  7  8  9  10
11  12  13  14  15
16  17  18  19  20
21  22  23  24  25
26
```

I did not include the commentary in the program through the use of REM statements to conserve memory in the CC-40. The program does include fairly extensive prompting to the display. A user should be able to run the program with very little reference to other instructions. Simply enter RUN in the display, press <ENTER> and follow the prompts. A sample printout which results from selection of the second input option, from DATA statements, appears at the right.

The execution times in seconds for the five sorting techniques for different numbers of random integers are:

	Number of Integers				
Method	10	30	100	300	1000
Bubble	2	24	252	2435	26685
Insertion	1	9	86	735	
Shell	3	13	73	390	
Heap	2	11	51	190	780
Address	1	3	10	30	

The table shows that execution times for the bubble sort grow as the square of N, the execution times for the heap sort grow as N LogN, and the execution times for the address sort grow linearly with N. Clearly, one should use some version of the addres sort if execution time is important and the problem permits.

For those times the random number input option (Lines 1000 through 1040) was modified such that the range of the positive random integers was the same as the number of data items to be sorted. That gives an advantage to the address sort. To check on how much an advantage that was I reran the 100 number case, but with the range of the integers raised to 300. The execution times for the five sorting methods were then 269, 89, 73, 55 and 15 seconds respectively.

The execution times for an input of 100 integers in reverse order were 490, 160, 39, 50, and 10 seconds respectively.

A NEW PORTABLE FROM TI? - CHHU is a new HP users group headed by Richard Nelson, the long-time editor of the PPC Calculator Journal. PPC is now apparently under other leadership. CHHU has a telephone bulletin board which will provide a recorded message (call 714-754-4557). Charlie Williamson reports that a recent bulletin stated "... rumor has it that TI will reenter the handheld marlet with new products, ala the Sharp 5100 and the TI-59. Will the TI -88 be reborn? ..."

A CC-40 DISC MEMORY

Maurice Swinnen sends this information on an external memory for the CC-40. See the ad at the right which is from a TI brochure on the CC-40 from Germany. The Quickdisk is made in Germany by:

MECHATRONIC GMBH
Dresdner strasse 21
D-7032 SINDELFINGEN

but Maurice says they will not sell except in large quantities. The German address where everything is available is:

REISS, Program Service
Bergstrasse 80
D-5584 BULLAY
West Germany

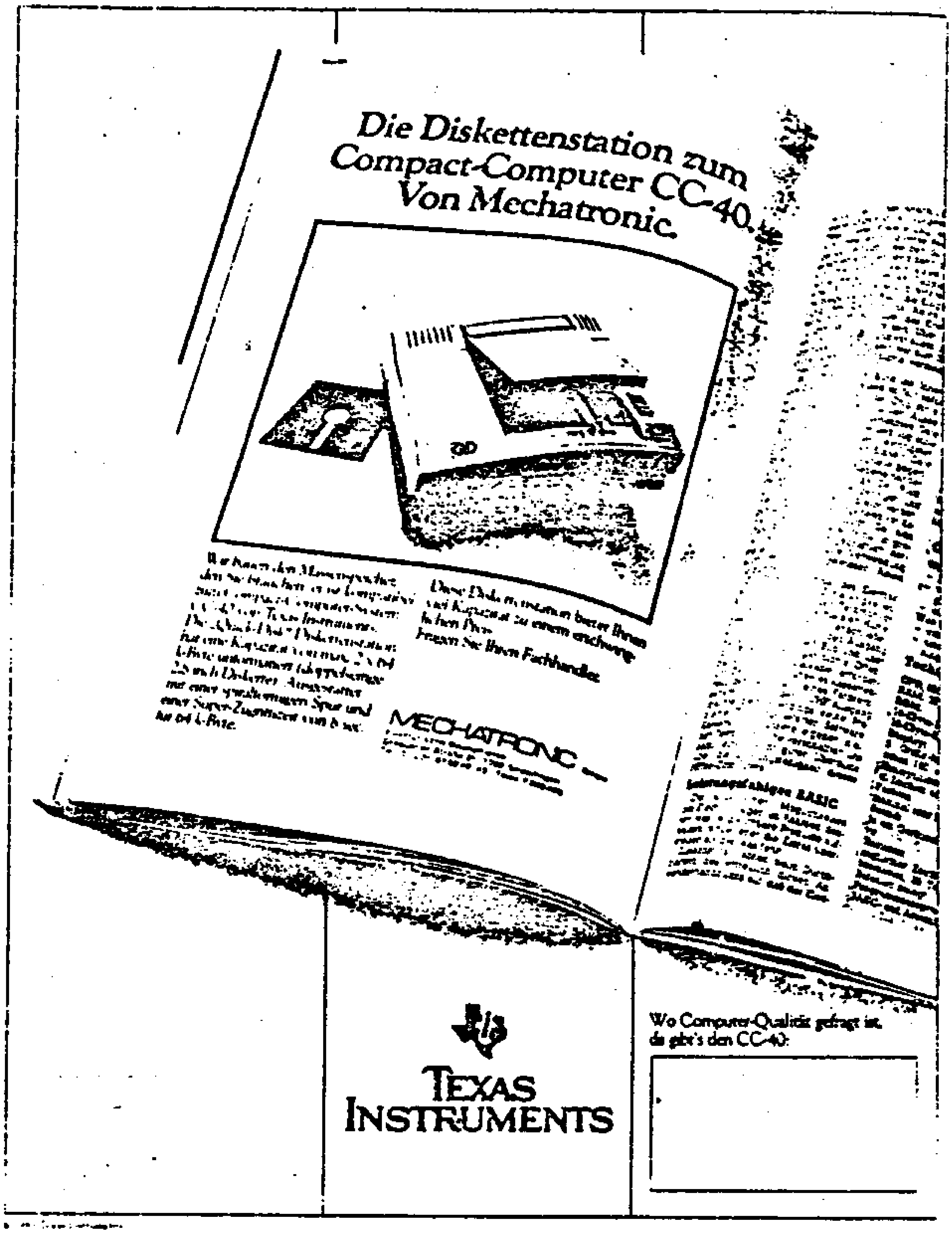
In the United States the unit is available from:

DIGITAL MATRIX SYSTEMS
1761 International Pkwy
Richardson TX 75081

You can call Frederick Winters at

(214)-997-0000

for prices and for more details. The storage capability is 128 KBytes. The unit runs on 4 size D batteries which will last about two hours on continuous use.



Digital Matrix Systems also has CC-40's and some peripherals for sale. I suggest that you call or write for current status.

CC-40 REPAIR - Maurice also writes that he found someone who repairs and modifies CC-40's. Write to MICROREP, 4413 Cornell Drive, Garland TX 75042. Maurice reports that they repaired his RS-232 Interface so that it now works!

A MINI-PUZZLE - Larry Leeds. Express 2^{2503} in scientific notation using the TI-59 to perform the calculations. You should be able to show that the answer is $3.006\ 624\ 187\ 161 \times 10^{753}$.

SCRAMBLING IN BASIC - Larry Leeds writes "Your new scramble program (V10N2P23) in fast mode is an excellent example of absolutely ingenious programming which was necessary to circumvent the limitations of the TI-59. It might be of interest to the members to note that when the same algorithm is used in a program in BASIC, the programming is very elementary."

Larry's program for the Radio Shack Model 100 scrambles the integers from 1 to 100 in just six seconds. The conversion for the CC-40 will scramble the integers from 1 to 100 in twelve seconds. A sample printout appears at the right. The CC-40 program appears below.

Scramble Program

Seed = 234

16	70	60	20	34
17	1	10	76	72
36	82	59	22	91
71	93	88	84	24
73	42	7	6	23
11	94	80	14	12
30	62	83	26	2
86	39	53	47	65
37	38	97	8	18
13	40	58	43	50
35	5	45	55	31
89	25	32	49	87
54	96	61	36	27
95	52	67	41	81
4	92	57	78	64
51	9	85	68	63
66	98	29	74	28
79	33	46	60	100
44	21	77	48	19
65	75	18	90	3

```
10 DIM A(100)
20 Q=ATN(1.E+14)
30 CALL UP("Scramble Program",PN)
40 INPUT "Seed ? ";S
45 PRINT #PN,"Seed = ";S:PRINT #PN
50 FOR H=1 TO 100:A(H)=H:NEXT H
60 FOR H=100 TO 1 STEP -1
70 X=S*Q
80 S=100*(X-INT(X)):Z=INT(S)
85 IF Z=0 THEN 70
90 T=A(A(Z)):A(A(Z))=A(H):A(H)=T
95 NEXT H
99 DISPLAY BEEP
100 FOR I=0 TO 95 STEP 5
110 FOR J=1 TO 5
120 PRINT #PN,A(I+J);
130 NEXT J
140 IF PN=0 THEN PAUSE
150 PRINT #PN
160 NEXT I
170 PRINT #PN
180 STOP
190 END
```

Program Comments:

Line 20 generates the value pi/2 which is used in the scrambler.

Lines 30 uses the UP routine from the cartridge to set up the control of printing or displaying the results. In answer to the prompt "Enter Filename:" respond with a "10". See page 14 for a more complete description of the UP subroutine.

Lines 50 enters the integers 1 through 100 in ascending order into the array.

Lines 60 through 99 scramble the integers. Larry uses his own random number generator rather than the one in the machine.

Lines 100 through 160 print or display the scrambled integers in groups of five.

UP SUBROUTINE AVAILABILITY - In both the sorting and scrambling programs in this issue I used the UP routine from the cartridge to establish printer/display control. Examination of the manuals shows, and tests verify, that the UP routine is in the three cartridges I own, Mathematica, Statistics and Finance. Is the UP routine available in all the cartridges?

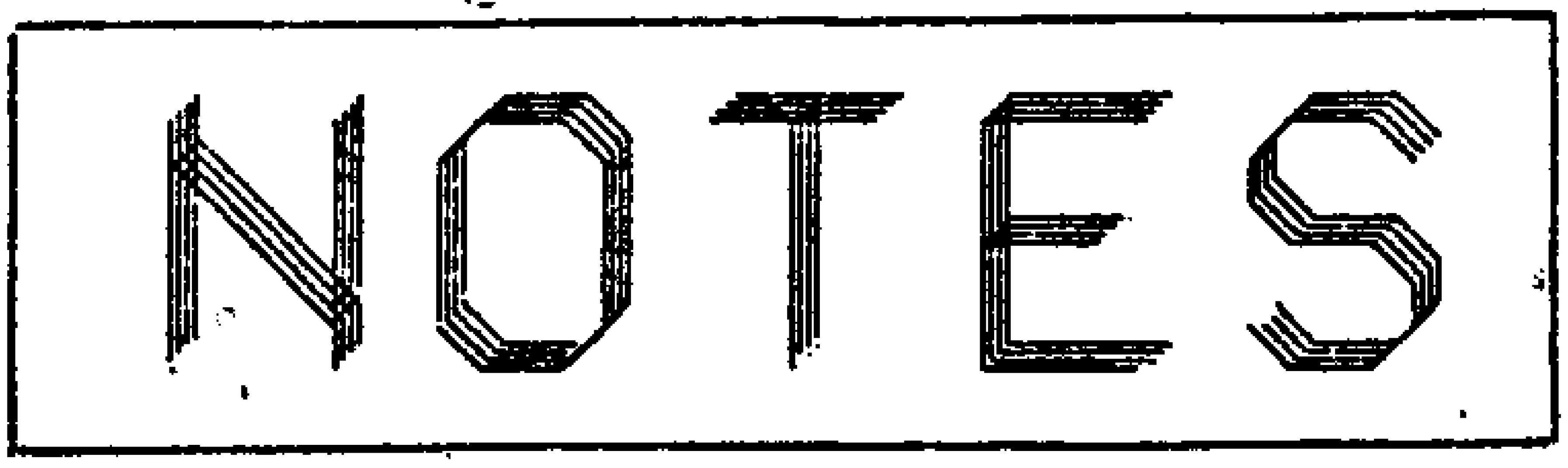
RTN: AN UNEXPECTED RESULT - Charlie Williamson writes: "A running program named EE encounters SBR X⇌T (for example): it saves a return address, and branches to execute X⇌T. Within X⇌T, our program hits R/S and waits for 'operator action' as they say. One then pushes either A or B, both of which contain a RTN. The program runs and encounters that RTN, whereupon it doesn't branch back to EE, but stops.

What's wrong? Obviously an address was saved, and an authentic RTN was present. SBR A (rather than A alone) doesn't help. I don't understand what the 59 does with an A-push in this situation.

One functional solution is to push GTO A R/S. The program then does branch back to EE. This subject is the basis of a George Vogel programming puzzle (V5N7P5). Presumedly his answer is similar.

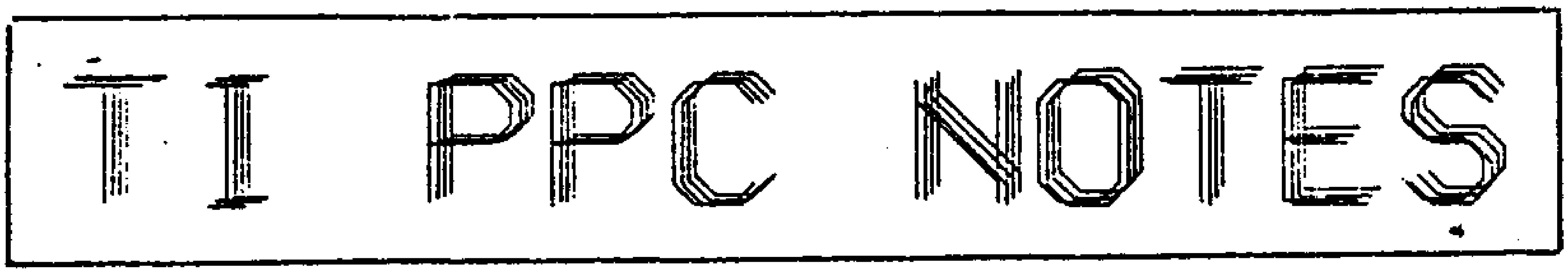
BANNER PROGRAM FOR THE CC-40/HX-1000

This program accepts a string of characters and prints a banner of one-half inch high letters. The program was used to generate the "TI PPC NOTES" logo on the first page of this issue. The letters in the banner can be enlarged without losing their sharpness as is illustrated by the word "NOTES" from the logo.




```
100 INPUT "Enter the banner characte
rs: ";B$
110 OPEN #1,"10",OUTPUT
120 PRINT #1,CHR$(19)
130 PRINT #1,"A1"
140 PRINT #1,"S9"
150 PRINT #1,"M(80,-80)"
160 PRINT #1,"O"
200 FOR J=1 TO LEN(B$)
210 PRINT #1,"M(0,-65)"
220 PRINT #1,"O"
300 FOR I=0 TO 6 STEP 2
310 P$="M("&STR$(I)&","&STR$(-I)&")"
320 PRINT #1,P$
330 P$="T("&SEG$(B$,J,1)&")"
340 PRINT #1,P$
350 NEXT I
360 NEXT J
370 PRINT #1,"M(0,-120)"
900 CLOSE #1
999 END
```

If you watch the banner making process you will see that all of the strokes are completed for one letter before proceeding to the next letter. An alternate process would be to complete single lines for all of the characters before offsetting the printing for subsequent printing; however, this places a requirement on the retracing characteristics of the printer/plotter which is not adequate for well formed letters in the banner. The banner below illustrates the loss of sharpness which results.



NO TEA PLEASE - Charlie Williamson. This is another of Charlie's programming challenges for the TI-58/59. As with his earlier max/min sorter challenge (See V7N1/2P9) t register comparisons are not allowed in the routine. Here's the challenge:

> Given integers X, A and B where B is greater than or equal to A and the integers are smaller in absolute value than $L = 5*10^{12}$ and have been stored in R0, R1, and R2 respectively, write a program with no direct comparisons that returns F(X) as

$$F(X) = \begin{cases} -1 & \text{if } X < A \\ 0 & \text{if } A \le X \le B \\ +1 & \text{if } B < X \end{cases}$$

Charlie believes he has a program which also works for numbers other than integers as well. He also asks for a search of your program for numbers where the program fails. For computers/calculators with fewer digits it may be necessary to decrease the value of L .

THE SOCIAL SECURITY NUMBER PUZZLE

Social security numbers in the US always consist of 9 digits.
My friend has an unusual social security number:
The first two digits on the left are evenly divisible by 2.
And the first three digits are evenly divisible by 3.
As you have guessed, this goes on up to the nineth digit,
such that the entire number is evenly divisible by 9.
What is his social security number?
Note: His number does not contain zeroes, nor are any digits repeated in it. Good luck!

Maurice E.T. Swinnen

A possible, but slow, solution on the CC-40 would run as follows:

```
100 OPEN #1,"16",OUTPUT
110 FOR N=121111119 TO 989999991
120 N$=STR$(N)
130 FOR I=9 TO 2 STEP -1
140 IF VAL(SEG$(N$,1,I))/I<>INT(VAL(SEG$(N$,1,I))/I)THEN 170
150 NEXT I
160 PRINT #1,N
170 NEXT N
180 CLOSE #1
190 END
```

A slightly faster solution, this time requiring only 3 years running time would read as follows:

```
100 OPEN #1,"16",OUTPUT
110 FOR N=121111119 TO 989999991
120 FOR I=0 TO 7
130 A(I)=INT(N/10^I)
140 IF A(I)/9-I<>INT(A(I)/9-I)THEN 170
150 NEXT I
160 PRINT #1,N
170 NEXT N
180 CLOSE #1:END
```

Can you write a FASTER solution, either in calculator language or in any dialect of Basic?

By the way, this entire article was writtten on a CC-40 and printed on a TI HX-3030 companion printer.

A BIT OF HISTORY - Hal Halvorsen. I wonder if the group knows that Asimov's first "Foundation" story (later novel) had the future mentor Hari Seldon using a little wizard calculator with bright red display digits. This was around 1941 in the old Astounding Science Fiction magazine. Asimov once wrote that that was one of the few science fiction predictions he knew that came out right on the money, now dated of course.

PLUMBING DESIGN - D. H. Berry. I have many programs available for the TI-59 which deal with HVAC, piping, and plumbing design. For a free catalog, send a large self-addressed envelope with two stamps to: D. H. Berry, 7693 Ceres Drive, Orlando FL 32822

FINDING PI - L. Leeds. V10N3P4 described a method for finding pi which involved dividing 2143 by 22 and taking the square root two times. The result was correct to nine digits. Larry used his Model 100 to search for a fraction which would yield more correct digits with the fourth root technique, but did not find any. For a single square root technique he found three fractions which would give thirteen correct digits on the TI-59; 3044467/308469, 17007401/1723210, and 26140802/2648617. For a simple division he found three fractions which would give thirteen digits of pi on a TI-59; 4272943/1360120, 5419351/1725033, and 61905677/19705189. Of course, those are all much harder to remember than the old standby 355/113 which yields seven correct digits.

How does one mechanize a search for fractions which will work? One method which was used by Larry is to use a decimal to fraction converter, and compare the result to a preselected error. Another method is to simply test the decimal equivalent of fractions against the value of pi loaded to the accuracy of the computer. If the value of the fraction is greater than pi then the denominator is increased by one and the new fraction is tested. If the value of the fraction is less than pi then the numerator is increased by one and the new fraction is tested. The previous best solution may be saved for comparison with the newly generated result so that the program only prints improved solutions. Sample programs written for the CC-40 and HX-1000 are listed below.

```
100 PRINT "Decimal to Fraction":PAUS
E 2
110 INPUT "Allowable Error ? ";E
120 INPUT "Decimal Number ? ";N
150 A=1:B=1:C=1:D=1
160 IF 1>=N THEN 180
170 D=0:GOTO 190
180 A=0
190 F=A+C:G=B+D
200 P=F/G:T=ABS(P-N)
220 IF E>=T THEN 270
240 IF N>=P THEN 260
250 C=F:D=G:GOTO 190
260 A=F:B=G:GOTO 190
270 PRINT F;"/";G
280 PAUSE
290 GOTO 110
```

```
100 PRINT "Search for Pi Fractions"
110 PAUSE 2
120 N=PI
130 OPEN #1,"10.*=0",OUTPUT
140 E=1
150 A=1:B=1:C=1:D=1
160 IF 1>=N THEN 180
170 D=0:GOTO 190
180 A=0
190 F=A+C:G=B+D
200 P=F/G:T=ABS(P-N)
220 IF E>=T THEN 270
240 IF N>=P THEN 260
250 C=F:D=G:GOTO 190
260 A=F:B=G:GOTO 190
270 PRINT #1,F;"/";G
280 E=.1*E
290 IF E>1.E-13 THEN 150
300 CLOSE #1
310 END
```

```
100 PRINT "Pi-Finder":PAUSE 2
110 OPEN #1,"10.*=0",OUTPUT
120 P=3.14159265359
130 M=1:N=1
140 D=ABS(P-M/N)
150 IF ABS(P-M/N)<D THEN 180
160 IF M/N>P THEN N=N+1 ELSE M=M+1
170 GOTO 150
180 PRINT #1,STR$(M)&"/"&STR$(N);TAB
(22);
190 PRINT #1,USING 900,M/N
200 GOTO 140
900 IMAGE #.#############


2/1                   2.0000000000000
3/1                   3.0000000000000
13/4                  3.2500000000000
16/5                  3.2000000000000
19/6                  3.1666666666670
22/7                  3.1428571428570
```

TI PPC NOTES V10N3P4

MY HOME COMPUTER - My home computer is a tool
That causes friends to stare and drool.
I now have power to explore
Those worlds I never knew before.

It costs a lot, but that's OK
"I'm worth it", I've been heard to say
Who wouldn't spend a couple of grand
To have such power at his hand?

I stare in awe at this machine
They say can do most anything
But somehow all that awe gets missed
When I type in my grocery list.

GROCERY LIST ON THE CC-40/HX-1000

Coffee
Milk
Sausage
Eggs
Potatoes
Bananas

E. E. Bard
