

original

A Collection of Information
on
The TI-74 and CC-40 Computers

by
Palmer O. Hanson, Jr.
Editor - TI PPC Notes
January 1990

This collection is a compilation of articles on the TI-74 and CC-40 and peripherals which appeared in the Volume 13 (1988-1989) issues of TI PPC Notes.

This material is not copyrighted and may be reproduced for personal use. When the material is used elsewhere we ask as a matter of courtesy that TI PPC Notes be mentioned. The use of material in this compilation is entirely at the user's risk. No responsibility as to the accuracy and the consequences due to the lack of it will be borne by either the club or the editor.

HARDWARE FOR SALE - In past issues members have offered used hardware for sale; for example, the V12N2 issue included four such offers. Several of the members have reported that they have sold the hardware in question. The following listings summarize the current availability of the used hardware. Since the material may be sold at any time it will be best to write first to check on terms and availability and to send money later.

Memo Processor Card for the CC-40	\$ 50.00
Hex-bus RS-232 Interface for the CC-40	125.00
Hex-bus Video Interface for the CC-40	125.00
Hex-bus Wafer Tape Drive for the CC-40	125.00

All hardware sold as is. He will sell the whole package for \$300.00. Write to Dan Eicher, P.O. Box 17401, Indianapolis, IN 46217. You can call at nights at 317-241-9942.

HARDWARE WANTED

B. V. Takach at P.O. Box 114, Wahroonga 2076, New South Wales, Australia would like to buy a CC-40 Assembler Cartridge and manual.

Lars Herold Andersen at 5 Ravnsbjerg Hegn, DK 7400 Herning, Denmark would like to buy an SR-50A, an SR-52 and an SR-56.

Michael Sperber at Birkenallee 67, 8526 Bubenreuth, Federal Republic of Germany would like to buy a SR-52.

PC to TI-74 INTERFACE CABLE

Georges Leonard wrote to call my attention to page 66 of the #44 issue of the EduCALC catalog which contains the entry which appears at the right. I called EduCALC for more details:

1. The device is not available now.
2. It is manufactured by TI.
3. It will both download and upload.

4. It is not expected to work with other computers such as the ATARI 1040ST.


I also called 1-800-TI-CARES. I was told that the device would not be available until the third or fourth quarter.

PC-Interface Cable
Brand New

Download data from your TI-74 to a TI or IBM-compatible PC—manipulate the data with PC programs, save to hard disc, floppies or printed hardcopies.

High-speed parallel transfer through 4' cable; connector is 1.5 x .75 inches.

Stock #U-2153 [PC-Interface Cable] \$54.95



A GAME FOR THE TI-74 - Stephen Gutknecht. In the "Arrow Game" an asterisk (*) is placed at a random position in the TI-74 display. An arrow, (->) or (<-), appears at either end of the display and begins moving toward the target. The object of the game is to press a key at the exact moment the moving arrow hits the target. Any key except ON, BREAK, RESET, SHIFT, FN or CTL may be used.

At the beginning of the game the player enters a number from zero to 999 in response to the prompt "Random Seed?" to randomize the sequence of target locations. In response to the prompt "Delay for game?" the player enters a number between 0 and 99 to control the speed at which the arrow moves across the display. Higher numbers make the arrow move slower. The first game begins immediately after the delay value is entered. The program stops with either Win!!! or Lose.. at the left end of the display. The remainder of the display shows the cumulative number of wins and games played. To start another game press any key.

To start a new series of games with a different delay press BREAK quickly. To exit the program press and hold BREAK.

Program Listing:

```

10 REM Arrow Game
20 REM By: Stephen Gutknecht
30 REM 1/07/87
1000 WHERE_START=5! define target limit
2000 DISPLAY ERASE ALL,"Random Seed? 000";
2010 ACCEPT AT(14)SIZE(-3)NULL(0)VALIDATE(DIGIT),SEED
3000 DISPLAY ERASE ALL,"Delay for game?";
3010 PAUSE .5
3050 ON ERROR STOP
3060 ON BREAK STOP
3100 ACCEPT AT(17)SIZE(2)NULL(2)VALIDATE(DIGIT),GAME_DELAY
3200 ON ERROR 3000
3210 ON BREAK ERROR
3500 GAME_SCORE=0
3510 GAME_NUMBER=0
4000 SEED=SEED+1! Create different seed for each game
4100 RANDOMIZE SEED
4500 KEYPRESS=0
4510 TARGET_LOCATION=INT(((31-WHERE_START)*RND)+WHERE_START)+1
4600 REVERSE=INT(2*RND)
4800 ON (REVERSE+1)GOSUB 7000,7200
4900 DISPLAY ERASE ALL AT(TARGET_LOCATION),*";
5000 DISPLAY AT(ARROW_LOCATION),ARROWS;
5010 PAUSE .5
5100 IF GAME_DELAY=0 THEN 5500
5200 LOOP=FIRST_PLACE
5210 DISPLAY AT(LOOP-LOOP_CHANGE),ARROW_FULL$;
5300 DELAY_LOOP=1
5310 CALL KEY(KEY,KEYPRESS);IF KEYPRESS THEN 6000
5320 IF DELAY_LOOP=GAME_DELAY THEN 5400
5330 DELAY_LOOP=DELAY_LOOP+1
5340 GOTO 5310
5400 LOOP=LOOP+LOOP_CHANGE
5410 IF LOOP=(TARGET_LOCATION+LOOP_CHANGE-REVERSE)THEN 6100
5420 GOTO 5210
5500 LOOP=FIRST_PLACE
5510 DISPLAY AT(LOOP-LOOP_CHANGE),ARROW_FULL$;
5520 CALL KEY(KEY,KEYPRESS);IF KEYPRESS THEN 6000
5530 LOOP=LOOP+LOOP_CHANGE
5540 IF LOOP=(TARGET_LOCATION+LOOP_CHANGE-REVERSE)THEN 6100
5590 GOTO 5510
6000 IF LOOP=(TARGET_LOCATION-REVERSE)THEN 6200
6100 DISPLAY ERASE ALL,"Lose..";
6120 GOTO 6500
6200 DISPLAY ERASE ALL,"Win!!!";
6210 GAME_SCORE=GAME_SCORE+1
6500 PAUSE .3
6510 GAME_NUMBER=GAME_NUMBER+1
6520 DISPLAY AT(10),"Score:";GAME_SCORE;
6530 DISPLAY AT(22),"Games:";GAME_NUMBER;
6600 CALL KEY(KEY,KEYPRESS)
6610 IF KEYPRESS=0 THEN 6600
6990 GOTO 4500
7000 FIRST_PLACE=2
7010 LOOP_CHANGE=1
7020 ARROW_LOCATION=1
7030 ARROWS=">"
7040 ARROW_FULL$="->"
7190 RETURN
7200 FIRST_PLACE=29
7210 LOOP_CHANGE=-1
7220 ARROW_LOCATION=31
7230 ARROWS="<"
7240 ARROW_FULL$="<-"
7300 TARGET_LOCATION=(31-TARGET_LOCATION)+1
7390 RETURN

```

A Game for the TI-74 - (cont)

Author's Note: The section of the program starting at line 5500 is only for the purpose of making the game faster. This is a replacement for the main loop that starts at line 5200. This will only be called when the game delay is set to zero.

Editor's Notes: This program incorporates several capabilities of the TI-74 not previously used in programs in our newsletter, namely,

- * Use of fifteen character variable names including the underline which is obtained with CTL 5. (See page 1-4 of the TI-74 Programming Reference Guide.) V10N2P16 contrasted the long variable names available with the TI-74 with other versions of BASIC which permit several letters in the variable name but only use the first two letters to defining variables. With those machines the variable names GAME_SCORE and GAME_NUMBER from lines 3500 and 3510 are seen as the same variable GA.
- * Use of the KEY subprogram to test whether a key has been pressed. See page 2-60 of the TI-74 Programming Reference Guide.

The very long variable names including the underline were apparently selected for compatibility with program development on another machine; for example, a portion of the program listing submitted by Mr. Gutknecht actually looked like

```

10 | "Arrow game" 4.02
20 | By: Stephen Gutknecht
30 | TI-74 BASICALC
40 | standard BASIC, 6K
50 | 01/07/87 at 09:30pm *EST*
98 |
99 |
1000 | where_start = 5 | define target limit
2000 | display erase all , "Random seed? 000" ;
2010 | accept at ( 14 ) size ( -3 ) null ( 0 ) validate ( digit ) , seed
3000 | display erase all , "Delay for game?" ;
3010 | pause 0.5
3050 | on error stop
3060 | on break stop
3100 | accept at ( 17 ) size ( 2 ) null ( 2 ) validate ( digit ) , game_delay
3200 | on error 3000
3210 | on break error
3500 | game_score = 0
3510 | game_number = 0
4000 | seed = seed + 1 | create different seed for each game
4100 | randomize seed
4499 |
4500 | keypress = 0
4510 | target_location = int ( ( ( 31 - where_start ) * rnd ) + where_start ) + 1
4600 | reverse = int ( 2 * rnd )
4800 | on ( reverse + 1 ) gosub 7000 , 7200
4900 | display erase all at ( target_location ) , "s" ;
4999 |
5000 | display at ( arrow_location ) , arrow0 ;
5010 | pause 0.7
5100 | if game_delay = 0 then 5500
5200 | loop = first_place
5210 | display at ( loop - loop_change ) , arrow_full0 ;
5300 | delay_loop = 1
5310 | call key ( key , keypress ) ; if keypress then 6000
5320 | if delay_loop = game_delay then 5400
5330 | delay_loop = delay_loop + 1
5340 | goto 5310
5400 | loop = loop + loop_change
5410 | if loop = ( target_location + loop_change - reverse ) then 6100

```

It occurred to me that the time for a member to enter the program in his TI-74 could be reduced and the amount of memory usage could be reduced by shortening the variable names. For example, "game_delay" was replaced by "GD", "target_location" was replaced by "TL", etc. The program which resulted appears on the next page.

A Game for the TI-74 - (cont)

```

10 REM Arrow game
20 REM By: Stephen Gutknecht
30 REM 1/07/87
1000 WS=5! define target limit
2000 DISPLAY ERASE ALL,"Random Seed? 000";
2010 ACCEPT AT(14)SIZE(-3)NULL(0)VALIDATE(DIGIT),S
3000 DISPLAY ERASE ALL,"Delay for game?";
3010 PAUSE .5
3050 DN ERROR STDP
3060 DN BREAK STDP
3100 ACCEPT AT(17)SIZE(2)NULL(2)VALIDATE(DIGIT),GD
3200 DN ERROR 3000
3210 DN BREAK ERROR
3500 GS=0
3510 GN=0
4000 S=S+1! Create different seed for each game
4100 RANDOMIZE S
4500 KP=0
4510 TL=INT(((31-WS)*RND)+WS)+1
4600 RV=INT(2*RND)
4800 DN (RV+1)GDSUB 7000,7200
4900 DISPLAY ERASE ALL AT(TL),"*";
5000 DISPLAY AT(AL),ARS;
5010 PAUSE .7
5100 IF GD=0 THEN 5500
5200 L=FP
5210 DISPLAY AT(L-LC),AF$;
5300 DL=1
5310 CALL KEY(KEY,KP):IF KP THEN 6000
5320 IF DL=GD THEN 5400
5330 DL=DL+1
5340 GOTO 5310
5400 L=L+LC
5410 IF L=(TL+LC-RV)THEN 6100
5420 GOTO 5210
5500 L=FP
5510 DISPLAY AT(L-LC),AF$;
5520 CALL KEY(KEY,KP):IF KP THEN 6000
5530 L=L+LC
5540 IF L=(TL+LC-RV)THEN 6100
5590 GOTO 5510
6000 IF L=(TL-RV)THEN 6200
6100 DISPLAY ERASE ALL,"Lose..";
6120 GOTO 6500
6200 DISPLAY ERASE ALL,"Win!!!";
6210 GS=GS+1
6500 PAUSE .3
6510 GN=GN+1
6520 DISPLAY AT(10),"Score:";GS;
6530 DISPLAY AT(22),"Games:";GN;
6600 CALL KEY(KEY,KP)
6610 IF KP=0 THEN 6600
6990 GOTO 4500
7000 FP=2
7010 LC=1
7020 AL=1
7030 ARS=">"
7040 AFS="<->"
7190 RETURN
7200 FP=29
7210 LC=-1
7220 AL=31
7230 ARS="<"
7240 AFS="<->"
7300 TL=(31-TL)+1
7390 RETURN

```

DETERMINING MEMORY USEAGE - P. Hanson and Douglas Elliot

A FRE(1) command had indicated that Mr. Gutknecht's program on page 12 used 1103 bytes. A FRE(1) after the modification to use shorter variable names indicated that the number of bytes used had actually increased to 1150 bytes. The increase was due my failure to delete the unused variable names from the system. I looked for a "CALL CLEANUP" capability such as that which was available with the CC-40. When I failed to find an equivalent capability for the TI-74 I saved the program to magnetic tape, pressed NEW ALL, and reloaded to program from tape. A FRE(1) now indicated that the condensed program on page 13 used 990 bytes.

In a conversation with Douglas Elliot I commented that there were several useful CC-40 capabilities which had not been provided with the TI-74. I mentioned the CHAR command in which the user could define up to seven special display characters and the the CALL CLEANUP capability. Douglas noted that the equivalent of the CALL CLEANUP could be obtained on the TI-74 by the SAVE command, where page 2-113 of the TI-74 Programming Reference Guide states that "Before storing the program, SAVE removes any variables from the system that are not used in the program. He also noted that it was not necessary to actually save the program. A user can simply enter the command SAVE "1" in the display, press ENTER, and ignore the error message "EO I/O error 31 "1" which appears.

SUPPLIES FOR THE TI-74 AND TI-95 - One member has reported difficulty in obtaining accessories and supplies for the TI-95. I have found that the Service Merchandise outlets, the only support for those devices in the Tampa Bay area, had clearance sales on TI-74 and TI-95 material and no longer carry either equipment or supplies. The latest catalogs from EduCALC (Volume 41) and Elek-Tek (Volume 16) continue to list TI-74 and TI-95 material. The addresses and telephone numbers are:

Addresses:	EduCALC Mail Store 27953 Cabot Road Laguna Niguel CA 92677	Elek-Tek 6557 North Lincoln Avenue Chicago IL 60645-3986
Telephone:	(714)-582-2637	1-800-621-1269

Both firms will accept Master Card or VISA for telephone orders. If you order please mention our club.

I have successfully used Radio Shack's Thermal Paper for use with the PC-3 Printer (Radio Shack Catalog No. 26-3592) in my PC-324. A box of five rolls is priced at \$2.49 plus tax. The paper seems to provide better contrast than the TI paper, but it's inconvenient to use since the rolls are so small -- each roll contains only 86 inches of paper. Both the TI paper and the Radio Shack paper are made in Japan. In V11N4P7 I commented on the high price of the PC-324 paper -- about 97 cents per cubic inch. The Radio Shack paper is only slightly less expensive, about 80 cents per cubic inch.

EXPERIENCE WITH PC-3 PAPER USED WITH THE PC-324 - P. Hanson. In V13N1P3 I reported that I had successfully used Radio Shack's thermal paper for the PC-3 in my PC-324. As I used more of the paper in subsequent months I have observed that catalog number 26-3592B consistently provides better contrast than catalog number 26-3592 or the TI paper. I do not know the difference between the two catalog numbers. In several stores I found the packages with the two catalog numbers intermixed on the store shelf, but there typically is more of the 26-3592 than of the 26-3592B available.

A 32K RAM FOR THE TI-95 - Scott Garver writes: I am now the owner of a fully working 32K RAM cartridge for my TI-95. The conversion is rather straightforward. The greatest problem was in trying to obtain the needed RAM chip. I currently have two additional chips on hand and can order more. I am offering to supply converted cartridges to club members at a cost of \$65.00 each for the conversion and the new chip. I will either (1) modify a cartridge that they send to me or (2) purchase a new cartridge and modify it, passing the additional cartridge cost on to them. Since I hate things that fail, I will guarantee the conversion of each cartridge with a moneyback guarantee for 9 months from the date of purchase. What I cannot guarantee though is the life of the existing battery. For those who want it I will install Robert Prins' TI-95 Utilities Cartridge programs onto the 32K cartridge free of charge. It has not been tested but the 32K should also work with the TI-74. Write to Scott L. Garver, 1113 Woodlawn Ct., Pekin, IL 61554 if you are interested.

Editor's Note: Cartridge battery life and the Robert Prins Utility Cartridge programs are discussed elsewhere in this issue.

SMALLEST CIRCLE TO ENCLOSE FOUR OR MORE POINTS - This programming challenge was proposed by Don Laughery in V12N4P26. He stated that a solution would have application in the field of positional tolerancing. Larry Leeds and Peter Messer responded with assistance from the editor.

The Editor's Note which accompanied the challenge stated that the challenge was clearly for more than three points since algebraic solutions for a circle through three points were well known. Our first insight was recognition that in many cases the circle through three points may not be the smallest circle to enclose the three points. Consider the case of three points where the third point is inside the circle with a diameter equal to the distance between the other two, and with a center at the mid-point between the two points. This is then the minimum diameter circle for the three points.

It was evident that the first phase in any solution would involve finding the pair of points which were furthest from each other (call them P1 and P2), and testing the remaining points (call them P3, P4, etc.) to determine if they were inside the minimum radius circle through P1 and P2. If all other points were inside the defined circle the problem was solved.

Larry and Peter then proposed algorithms for solution when any of the remaining points (P3, P4, etc.) were outside the minimum circle defined by P1 and P2. The proposed algorithms involved testing the distance of the remaining points from either P1 or P2, or from the center of the minimum circle through P1 and P2, and then solving for the circle through P1, P2 and a selected third point. A little reflection will reveal that the circles so defined will have their centers on the perpendicular bisector of the line segment between P1 and P2.

The editor programmed the algorithms for the TI-74. The solution was ready for publication when it was found that there were some cases where a smaller diameter circle could be found to enclose all of the points by using only one of the most distant points (P1 or P2) and two other points. One example is the set of points (4,4), (0,-6), (-5,2), and (5,3). In this case the most distant points are (4,4) and (0,-6) but the circle through the last three points yields a smaller diameter which will enclose the four points.

At this point the editor wrote a new program for the TI-74 which finds the minimum circle defined by P1 and P2, and also finds the parameters (center and radius) for each circle defined by any combination of three points in the ensemble. The program also examines each circle so defined to determine whether or not the remaining points are within the circle. If any point is not, the radius of the circle is increased until all points are included. Finally, the circle with minimum radius is selected as the solution. The program on page 29 mechanizes that algorithm. An example solution is also provided on page 29, where it is yet to be proved that even this solution is the one which yields the minimum radius which will include all of the points.

Lines 10 through 70 provide for selection of the display, the PC-324 or the HX-1000 for output.

Lines 100 through 190 provide for data entry using a modified form of the string input techniques used in Maurice Swinnen's books. In this case the end of input data is signalled by pressing ENTER with a null string (no input) in the display, rather than by entering an "E" or an "e" and pressing ENTER. User's will find this is more convenient.

Lines 200 through 245 identify the pair of most distant points (P1 and P2) and find the center of the minimum circle through the two points.

Smallest Circle - (cont)

Lines 300 through 335 examine each of the remaining points (P3, P4, etc.). If a point is outside the minimum circle the radius is increased to include that point. The center of the circle is not changed. If none of the remaining points is outside the minimum circle through P1 and P2 the program proceeds to print the solution.

Lines 345 through 760 examines each possible set of three points from the ensemble in order, find the parameters which define the circle through the three points, examine the remaining points to determine whether they are inside the defined circle or not and increases the radius (actually the square of the radius) to include all points. If the resulting circle is smaller than the previously selected circle then its parameters are used as the basis for examining subsequent candidates.

Lines 400 through 460 use the solution for a circle through three points from lines 200 through 295 of the program on page 26.

Lines 800 through 850 print the solution, or display the solution if a printer is not used.

<pre> 10 AS="Smallest Circle": PRINT AS:PAUSE 1 20 DIM X(10),Y(10) 25 INPUT "Use Printer? Y /N ";Z\$ 30 IF Z\$="Y"OR Z\$="y"THE N PN=1 ELSE 100 35 PRINT "Device Numbers ":PAUSE 1 40 PRINT "For the HX-100 0 enter 10":PAUSE 1 45 PRINT "For the PC-324 enter 12":PAUSE 1 50 INPUT "Enter device n umber ";DS 55 OPEN #1,DS,OUTPUT 60 IF DS="10"THEN PRINT #1,CHR\$(18) 65 PRINT #1:PRINT #1,AS 70 PRINT #1 100 PRINT "End Input by Entering "&CHR\$(255):PAU SE 2 110 N=1 120 XS="X = " 125 YS="Y = " 130 INPUT XS:XXS:IF XXS= ""THEN 190 135 INPUT YS:YYs:IF YYs= ""THEN 190 140 X(N)=VAL(XXS) 145 Y(N)=VAL(YYs) 150 IF PN=0 THEN 180 155 PRINT #PN,XS:X(N) 160 PRINT #PN,YS:Y(N) 170 PRINT #PN 180 N=N+1:GOTO 130 190 N=N-1 195 PRINT "Solving" 200 M1=0 205 FOR I=1 TO N 210 FOR J=2 TO N-1 </pre>	<pre> 215 D2=(X(I)-X(J))^2+(Y(I)-Y(J))^2 220 IF D2>M1 THEN M1=D2: S=I:T=J 225 NEXT J 230 NEXT I 235 HH=(X(S)+X(T))/2 240 KK=(Y(S)+Y(T))/2 245 M2=M1/4 300 Z=0 305 FOR I=1 TO N 310 IF I=S OR I=T THEN 3 25 315 D2=(X(I)-HH)^2+(Y(I) -KK)^2 320 IF D2>M2 THEN M2=D2: Z=1 325 NEXT I 335 IF Z=0 THEN 800 345 FOR I=N TO 3 STEP -1 350 FOR J=(I-1)TO 2 STEP -1 355 FOR L=(J-1)TO 1 STEP -1 365 A=X(I):B=Y(I) 370 C=X(J):D=Y(J) 375 E=X(L):F=Y(L) 400 A=A-E:C=C-E 405 B=B-F:D=D-F 410 G=C*B-A*D:G=C+G 415 H1=C*C+D*D 420 X1=B:B=H1 425 B=X1*B:H1=A*H1 430 X1=X1*X1+A*A 435 D=X1*D:C=X1*C 440 B=B-D:C=C-H1 445 B=B/G:C=C/G 450 E=E+B:F=F+C 455 R2=B*B+C*C 460 H=E:K=F 705 FOR M=1 TO N 710 IF M=L OR M=J OR M=I THEN 725 </pre>	<pre> 715 D2=(X(M)-H)^2+(Y(M)- K)^2 720 IF D2>R2 THEN R2=D2 725 NEXT M 730 IF R2<M2 THEN M2=R2: HH=H:KK=K 735 NEXT L 755 NEXT J 760 NEXT I 800 PAUSE ALL 810 PRINT #PN," h = ";HH 820 PRINT #PN," k = ";KK 830 PRINT #PN," r = ";SQ R(M2) 840 IF PN=1 THEN PRINT # 1 850 PAUSE 0 999 END </pre> <p>Smallest Circle</p> <pre> X = -7 Y = 0 X = 0 Y = 6 X = 3.5 Y = 5 X = 0 Y = -6 X = -2 Y = 5 h = -.6913265306 k = .2767857143 r = 6.314742376 </pre>
---	---	--

THE SMALLEST CIRCLE PROBLEM REVISITED - In V12N4P26 Don Laughery proposed the finding of the smallest circle which would just enclose n random points as a programming challenge. V13N1P28-29 presented a solution which was the result of the combined effort of Larry Leeds, Peter Messer and the editor. The publication of the program elicited interest from other members. The details of the subsequent work on the solution are presented below to show the kind of results that can be achieved when we get several members working on a problem. One result is that I have a file of notes that is nearly an inch thick.

Carl Rabe, George Thomson and Larry Leeds all found that a divide by zero error will be generated when three of the points are on a straight line, or when the same point is entered twice, which is the same thing as saying that three points are in a straight line. The problem is caused by the divides by G at line 445, where the value of G calculated at line 410 will be equal to zero for the conditions mentioned above. The problem was circumvented by testing for G = 0 and replacing the zero by a very small number; that is, by inserting a new line

```
412 IF G=0 THEN G=1.E-06
```

Carl Rabe then found that the smallest circle will not be found for the sequence of points

```
(5,7) (9,10) (12,13) (16,18) (21,25) and (25,21)
```

but would be found if the sequence was rearranged. The set of points involves the case where the two most distant points are the first and last points, and the remaining points are within the circle with its diameter equal to the distance between the two points, and its center midway between the two points. There was a fundamental flaw in the program at lines 205 and 210 which were

```
205 FOR I=1 TO N
210 FOR J=2 TO N-1
```

If you think about those two statements you will realize that the distance will never be calculated between the first and last points in the input sequence. There will be other cases which will not be tested as well. The deficiency can be eliminated for programs on the TI-74 or CC-40 by changing line 210 to

```
210 FOR J=(I+1) TO N
```

but that will not work for computers which use Microsoft BASIC such as the Radio Shack Color Computer or Model 100. A routine which seems to work with all the BASIC implementations is

```
205 FOR I=2 TO N
210 FOR J=1 TO I-1
```

The program was modified in the second way to avoid potential problems if the program were to be used on other machines. With the changes described above the program appeared to operate satisfactorily. Carl Rabe found that if he entered the set of points

```
(8,11) (21,2) (22,18) (34,26) (33,38) and (29,31)
```

Smallest Circle - (cont)

the program would yield the correct answer with the center at (24.4375, 20.845166...) and a radius of 19.1649... . But if the point (22,18) was duplicated at the end of the set an incorrect answer with the center at (22,18) and a radius of 16.0312... would result. At first we were really puzzled because the routine from lines 705 to 725 was expressly put into the program to ensure that we couldn't have this sort of problem. Finally, we realized that line 710 prevents the testing of the three points which were used to define the circle being tested by the routine. We deleted line 710 and received a correct solution with the duplicate points. (Incidentally, line 710 is not very important--it was added to provide some small saving in execution time by not testing the points used to define the circle). After thinking about the problem some more we found that there was another solution to the difficulty with duplicate points. It is to change the branch at line 712 when $G = 0$ from

```
IF G=0 THEN G=0.000001      to      IF G=0 THEN 735
```

where we rely on the idea that if $G = 0$ then there are either all three points in a straight line, or duplicate points and we really don't want to examine the resulting "circle" which has an infinite radius anyway. Again, we received correct results with duplicate points.

The program now appears to work satisfactorily, but not necessarily efficiently. The inefficiency results from having to examine the circles defined by every combination of three points, a result of our inability to find a methodology for finding the correct combination more directly. The inefficiency was emphasized when Don Laughery wrote that the solution could sometimes be required to examine twenty or more points. The following table compares solution times for various numbers of points for the program on the TI-74, for the equivalent program on the Radio Shack Model 100 and for a program by Carl Rabe in True BASIC on the ATARI 1040ST processor.

Execution Time (seconds)				
n	nC3	TI-74	M 100	ATARI
---	-----	-----	-----	-----
3	1	1	1	< 1
4	4	3	3	
5	10	8	6	
6	20	17	13	
9	84	92	71	
12	220	301	239	
15	455	752	608	
18	816	1558	1305	
21	1330	2940	2444	78

Smallest Circle - (cont)

Those execution times are for the condition where all the points are not within the circle defined by the two most distant points. If all the points are within the circle defined by the most distant points the execution times would be much shorter. $nC3$ in the table is the number of combinations of n points taken three at a time. Not surprisingly, the execution time appears to approximately follow $nC3$ since the major portion of the calculations are involved with testing each combination of three points to determine whether it defines the smallest circle. Actually, we would expect that the execution time would be proportional to the product of n times $nC3$ since for each combination of three points all of the remaining points are tested to determine whether they are inside the circle defined by the three points.

Now, purists will object that we haven't demonstrated that we always find the smallest circle. Larry Leeds has pointed out that, if one of the circles defined by one of the sets of three points encloses all of the remaining points, then the circle is of minimum radius. This follows from the fact that the center of the circle is equidistant from each of the three points, and if the center is displaced in any direction the distance to at least one of the three points will be increased. But, we have never proved that there cannot be sets of points for which none of the circles defined by any three points of the set will enclose all of the remaining points. Lines 705 through 725 of the present program account for that possibility by expanding the circle radius to enclose any point outside the circle defined by a set of three points, but it seems obvious that this will not yield the smallest circle. For example, for a single point outside the circle defined by three points, a smaller enclosing circle could probably be obtained by moving the center toward the outside point, and increasing the radius by a smaller amount. Larry has also found the following example which controverts our early assumptions that the smallest circle must go through at least one of the two points which are furthest apart:

(0,6) (3.5,5) (0,-6) (-7,0) (3.5,-5)

where the smallest circle passes through the second, fourth and fifth points, and encloses the first and third points which are the furthest apart. Meanwhile, on February 23 Don Laughery wrote:

"I had occasion to used the smallest circle program under fire this week. ... I collected the data and entered it into the program. While it was running (3 1/2 minutes) I plotted the data and checked it with the overlay - BINGO!! I even caught a few reversed-sign errors I made in plotting. Out of 25 runs no anomalies showed up, so it looks like maybe it's the answer. My thanks to you and all the others who worked on this. ..."

A printout of the current version of the program for the TI-74 appears on page 15. Members are invited to examine the remaining issues of improved efficiency and proof of minimum radius for all cases. Any input will be reported in subsequent issues.

Smallest Circle - (cont)

TI-74 Program Listing

```

10 AS="Smallest Circle":
PRINT AS:PAUSE 1
20 DIM X(30),Y(30)
25 INPUT "Use Printer? Y
/N ";Z$
30 IF Z$="Y"DR Z$="y"THE
N PN=1 ELSE 100
35 PRINT "Device Numbers
:":PAUSE 1
40 PRINT "For the HX-100
0 enter 10":PAUSE 1
45 PRINT "For the PC-324
enter 12":PAUSE 1
50 INPUT "Enter device n
umber ";DS
55 OPEN #1,DS,OUTPUT
60 IF DS="10"THEN PRINT
#1,CHR$(18)
65 PRINT #1:PRINT #1,AS
70 PRINT #1
100 PRINT "End Input by
Entering "&CHR$(255):PAU
SE 2
110 N=1
120 XS="X = "
125 YS="Y = "
130 INPUT XS:XXS:IF XXS=
""THEN 190
135 INPUT YS:YYS:IF YYS=
""THEN 190
140 X(N)=VAL(XXS)
145 Y(N)=VAL(YYS)
150 IF PN=0 THEN 180
155 PRINT #PN,XS:X(N)
160 PRINT #PN,YS:Y(N)
170 PRINT #PN
180 N=N+1:GOTO 130
190 N=N-1
195 PRINT "Solving"
200 M1=0
205 FOR I=2 TO N
210 FOR J=1 TO I-1
215 D2=(X(I)-X(J))^2+(Y(
I)-Y(J))^2
220 IF D2>M1 THEN M1=D2:
S=I:T=J
225 NEXT J
230 NEXT I
235 HH=(X(S)+X(T))/2
240 KK=(Y(S)+Y(T))/2
245 M2=M1/4
300 Z=0
305 FOR I=1 TO N
310 IF I=S DR I=T THEN 3
25
315 D2=(X(I)-HH)^2+(Y(I)
-KK)^2
320 IF D2>M2 THEN M2=D2:
Z=1
325 NEXT I
335 IF Z=0 THEN 800
345 FOR I=N TO 3 STEP -1
350 FOR J=(I-1)TO 2 STEP
-1
355 FOR L=(J-1)TO 1 STEP
-1
365 A=X(I):B=Y(I)
370 C=X(J):D=Y(J)
375 E=X(L):F=Y(L)
400 A=A-E:C=C-E
405 B=B-F:D=D-F
410 G=C*B-A*D:G=G+G
412 IF G=0 THEN 735
415 H1=C*C+D*D
420 X1=B:B=H1
425 B=X1*B:H1=A*H1
430 X1=X1*X1+A*A
435 D=X1*D:C=X1*C
440 B=B-D:C=C-H1
445 B=B/G:C=C/G
450 E=E+B:F=F+C
455 R2=B*B+C*C
460 H=E:K=F
705 FOR M=1 TO N
710 IF M=L DR M=J DR M=I
THEN 725
715 D2=(X(M)-H)^2+(Y(M)-
K)^2
720 IF D2>R2 THEN R2=D2
725 NEXT M
730 IF R2<M2 THEN M2=R2:
HH=H:KK=K
735 NEXT L
755 NEXT J
760 NEXT I
800 PAUSE ALL
810 PRINT #PN," h = ";HH
820 PRINT #PN," k = ";KK
830 PRINT #PN," r = ";SQ
R(M2)
840 IF PN=1 THEN PRINT #
1
850 PAUSE 0
999 END

```

ERRATA

TI-74/PC-324 Calendar - V12N2P6. Wayne Spyker entered this program as listed and found that the printout for a month would not terminate. An example appears at the right, where the printing was terminated by pressing BREAK. The problem was traced to the second statement in line 270. That statement reads IF I>Q(M) THEN ... where Q(M) is the number of days in the month being printed. The statement should be IF I>Q(M) THEN The error was introduced by a flaw in the printout on the PC-324. Once again, we see that the only sure way to publish programs which will run is to key the program in from the printed page. Unfortunately, that is sometimes not compatible with publication schedules.

APRIL 1989						
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	32	33	34	35	36
37	38	39	40	41	42	43
44	45	46	47	48	49	50
51	52	53	54	55	56	57

A FASTER SMALLEST CIRCLE PROGRAM - Carl Rabe. The problem of finding the smallest circle which will just enclose n random points was proposed by Don Laughery in V12N4P26. A workable solution was published in V13N2P12; however, for large numbers of points the program was painfully slow. Carl Rabe tackled the problem of reducing the solution time. His method involved identifying a subset of four points which are near the periphery of the set of points. A solution is found which will enclose the four points. That solutions is checked against the remaining points. Any points which fall outside the circle defined by the subset are added to the subset and the process is repeated. Don Laughery has noted that this process is similar to the one he uses graphically, that is, he ignores the points which are obviously not on the perimeter.

Carl reports the following improvement in solution times on the TI-74:

Pairs	V13N2P12	This Program
-----	-----	-----
12	301 sec	9 to 44 sec
21	2940 sec	11 to 47 sec

```

SMALLEST CIRCLE
# 1 X= 9.8 Y= 3.9
# 2 X= 6.1 Y=-9.8
# 3 X= 11.3 Y= 2.1
# 4 X= 8.4 Y=-8.3
# 5 X=-9.8 Y= 7.5
# 6 X= 5.5 Y= 13.9
# 7 X= 5 Y= 8.7
# 8 X= 9.6 Y= 8.1
# 9 X= 16.5 Y= 1.1
# 10 X= 11.8 Y= 12.5
# 11 X=-9.8 Y=-7.2
H = 1.84275743
K = 1.725961397
R = 14.67060283

```

where the variation in the solution time with this program depends upon the difficulty of solving a particular case. The shorter times are for cases which require a single cycle through the compute and check-for-outside of circle sequence. The longer times are for cases which require three cycles. A sample printout appears at the right.

Carl has noted that the program as written sometimes uses more cycles than would be expected due to the interaction of truncation errors in comparisons such as those at lines 675, 685, 725 and 745. We will work on that and try to publish a modification in the next issue. Meanwhile, the program on page 15 provides by far the fastest smallest circle solutions obtained to date.

Note: The constants in lines 725 and 745 must be selected to match the input data resolution.

Program Description:

Lines 10-290: This is the input section. It is similar to the input section of the program on V13N2P15. Lines 45-55 have been made inoperative and the device code number is set to that for the PC-324 in line 60. This may be changed to provide a choice of printers if that is desired. Additions provide a for optional recheck of inputs or a change of inputs if desired (PC-324 printer paper is at a premium due to the cost and the small amount of paper in each roll.)

Lines 295-440: This section sorts out four pairs which will fall near the periphery of the plot of all the input pairs. It places the four pairs in an array for use by the main program. If the number of input pairs is 4 or less then no sorting is done.

Lines 445-700. These lines are from V13N2P15 lines 200-760 with minor changes.

Lines 705-760. This section checks all points to see if any are outside the circle obtained from the reduced set of points. If any points are found outside the circle they are added to the reduced array and a new cycle is initiated. The process is continued until no points are found to be outside the computed circle, which is then the smallest circle.

Lines 765-815 provide for display and/or printout of the results.

Lines 820-865 are a subroutine used by the sorting process.

A Faster Smallest Circle Program - (cont)

Program Listing

```

10 AS="SMALLEST CIRCLE"
PRINT AS:PAUSE 1
15 !FROM PPC NOTES V13N2
P15
20 !MODIFIED 10/15/89 C.
RABE
25 DIM X(52),Y(52),U(52)
,V(52)
30 INPUT "USE PRINTER? Y
/N ":Z8
35 IF Z8="Y"DR Z8="V"THE
N PN=1 ELSE 80
40 !PRINT "PRINTER NUMBE
RS":PAUSE 1
45 !PRINT "FDR THE MX-10
00 ENTER 10":PAUSE 1
50 !PRINT "FDR THE PC-32
4 ENTER 12":PAUSE 2
55 !INPUT "ENTER PRINTER
NUMBER ":DS
60 DS="12"
65 OPEN #1,DS,OUTPUT
70 IF DS="10"THEN PRINT
#1,CHRS(18)
75 PRINT #1:PRINT #1,AS
80 INPUT "USE CANNED DAT
A? Y/N ":Z28
85 IF Z28="Y"DR Z28="V"TH
EN 90 ELSE 155
90 PRINT "(DATA STATEMEN
T BEGINS LINE 125)":PAUS
E 1
95 FOR I=1 TO 50
100 READ X(I),Y(I)
105 IF X(I)=999 THEN 215
110 M=M+1
115 NEXT I
120 PRINT "NUMBER OF DAT
A PAIRS IS:"M:PAUSE 1
125 DATA 9.8,3.9,6.1,-9.
8,11.3,2.1
130 DATA 8.4,-8.3,-9.8,7
.5,5.5,13.9
135 DATA 5.8,7.9,6.8,1.1
6.5,1.1
140 DATA 11.8,12.5,-9.8,
-7.2
145 DATA 999,999
150 GOTO 215
155 PRINT "ENTER INPUT X
& Y VALUES":PAUSE 1
160 PRINT "(TO END INPUT
ENTER A NULL)"&CHRS(255)
):PAUSE 1
165 M=1
170 XS="X = "
175 YS="Y = "
180 INPUT XS:XXS:IF XXS=
""THEN 210
185 INPUT YS:YYS:IF YYS=
""THEN 210
190 X(M)=VAL(XXS)
195 Y(M)=VAL(YYS)
200 IF PN=0 THEN 205
205 M=M+1:GOTO 180
210 M=M-1
215 PRINT "NUMBER OF DAT
A PAIRS IS:"M:PAUSE 2
220 INPUT "WISH TO RECHE
CK INPUT? Y/N":Z228
225 IF Z228="Y"DR Z228="
V"THEN 235 ELSE 275
230 PRINT "TO CHK INPUTS
PRESS:ENTER ":PAUSE
235 FOR I=1 TO M
240 PRINT "PAIR#":I:" X
I":X(I):" Y":Y(I):PAUSE
245 NEXT I
250 INPUT "CHANGE AN INP
UT? Y/N":Z258
255 IF Z258="Y"DR Z258="V"
THEN 260 ELSE 220
260 INPUT "WHICH PAIR #?
I ":I
265 INPUT "ENTER NEW X:"
IX(I)
270 INPUT "ENTER NEW Y:"
IY(I):GOTO 250
275 IF Z8="Y"DR Z8="V"TH
EN 280 ELSE 295
280 FOR I=1 TO M
285 PRINT #PN,"#":I:"X="
IX(I):"Y="IY(I):PAUSE
290 NEXT I
295 PRINT "NOW SORTING..
PLEASE WAIT":PAUSE 1
300 DIM XX(21),YY(21),XX
X(21),YYY(21)
305 FOR J=1 TO M:U(I)=X(
I):V(I)=Y(I)
310 NEXT I
315 TOTAL_M=M
320 FAC=4
325 IF FAC=M THEN FAC=M
:GOTO 430
330 PSN,I=1:LDUX=X(I):PT
R=0
335 FOR I=1 TO M
340 IF X(I)<LDUX THEN LD
UX=X(I):PSN=I
345 NEXT I
350 CALL PLACE(X),Y),P
SN,PTR)
355 PSN,I=2:HX=X(I):PTR
=1
360 FOR I=1 TO M
365 IF X(I)>HX THEN HX
=X(I):PSN=I
370 NEXT I
375 CALL PLACE(X),Y),P
SN,PTR)
380 PSN,I=3:LDVY=Y(I):PT
R=2
385 FOR I=1 TO M
390 IF Y(I)<LDVY THEN LD
VY=Y(I):PSN=I
395 NEXT I
400 CALL PLACE(X),Y),P
SN,PTR)
405 PSN,I=4:HIY=Y(I):PTR
=3
410 FOR I=1 TO M
415 IF Y(I)>HIY THEN HIY
=Y(I):PSN=I
420 NEXT I
425 CALL PLACE(X),Y),P
SN,PTR)
430 FOR I=1 TO FAC
435 XX(I)=X(I):YY(I)=Y(
I)
440 NEXT I
445 PRINT "NOW SOLVING..
PLEASE WAIT":
450 CYCLES=CYCLES+1
455 FOR I=1 TO FAC
460 X(I)=XX(I):Y(I)=YY(
I)
465 NEXT I
470 M=FAC
475 M1=0
480 FOR I=2 TO M
485 FOR J=1 TO I-1
490 B2=(X(I)-X(J))+X(I)
-X(J))+Y(I)-Y(J))+Y(I)
-Y(J))
495 IF B2>M1 THEN M1=B2:
S=I:T=J
500 NEXT J
505 NEXT I
510 MH=(X(S)+X(T))/2
515 KK=(Y(S)+Y(T))/2
520 M2=M1/4
525 Z=0
530 FOR I=1 TO M
535 IF I=S DR I=T THEN 5
50
540 B2=(X(I)-MH)+X(I)-H
M)+Y(I)-KK)+Y(I)-KK)
545 IF B2>M2 THEN M2=B2:
Z=1
550 NEXT I
555 IF Z=0 THEN 765
560 FOR I=M TO 3 STEP -1
565 FOR J=(I-1)TO 2 STEP
-1
570 FOR L=(J-1)TO 1 STEP
-1
575 A=X(I):B=Y(I)
580 C=X(J):D=Y(J)
585 E=X(L):F=Y(L)
590 A=A-E:C=C-E
595 B=B-F:D=D-F
600 G=C*B-A*D:G=C+G
605 IF G=0 THEN 690
610 M1=C*D+B*D
615 X1=B*B=M1
620 B=X1*B:M1=A*M1
625 X1=X1*X1+A*A
630 B=X1*B:C=X1*C
635 B=B-D:C=C-M1
640 B=B/G:C=C/G
645 E=E+B:F=F+C
650 R2=B*B+C*C
655 M=E:K=F
660 FOR M=1 TO M
665 IF M=L DR M=J DR M=I
THEN 680
670 D2=(X(M)-H)+(X(M)-H)
+(Y(M)-K)+(Y(M)-K)
675 IF D2>R2 THEN R2=D2
680 NEXT M
685 IF R2<M2 THEN M2=R2:
MH=M:KK=K
690 NEXT L
695 NEXT J
700 NEXT I
705 M=TOTAL_M
710 FOR I=1 TO M
715 X(I)=U(I):Y(I)=V(I)
720 NEXT I
725 M2=INT(M2+1.E+07+.5)
/1.E+07
730 CNT=0
735 FOR I=1 TO M
740 B2=(X(I)-MH)+X(I)-M
H)+Y(I)-KK)+Y(I)-KK)
745 B2=INT(B2+1.E+07+.5)
/1.E+07
750 IF B2>M2 THEN CNT=CN
T+1:FAC=FAC+1:XX(FAC)=X(
I):YY(FAC)=Y(I)
755 NEXT I
760 IF CNT<0 THEN 450
765 PRINT "SOLUTION COMP
LETED.":PAUSE 2
770 PRINT "NUMBER OF CYC
LES:":CYCLES:PAUSE
775 PRINT "PAIRS FOR FIN
AL COMPUTE:":FAC:PAUSE
780 PRINT "CIRCLE DATA F
OLLOW":PAUSE 2
785 PAUSE ALL
790 PRINT #PN," M = ":MH
795 PRINT #PN," K = ":KK
800 PRINT #PN," R = ":SQ
R(M2)
805 IF PN=1 THEN PRINT #
1
810 PAUSE 0
815 PRINT "PROBLEM COMPL
ETED":PAUSE 1
820 SUB PLACE(X),Y),PS
N,PTR)
825 PTR=PTR+1
830 TMPX=X(PTR)
835 TMPY=Y(PTR)
840 X(PTR)=X(PSN)
845 Y(PTR)=Y(PSN)
850 X(PSN)=TMPX
855 Y(PSN)=TMPY
860 SUBEND
865 END

```

Editor's Comments: Carl implemented some other features in his program which are worthy of mention:

1. The definition of lines 40 through 55 as comments saves the user some time if he is only using the PC-324. Note that line 60 sets the device code number to that for the PC-324. If the option to select the printer from the keyboard is to be implemented the exclamation points in lines 40 through 55 must be removed and line 60 must be deleted or changed to comment status by the insertion of an exclamation point.

A Faster Smallest Circle Program - (cont)

2. Lines 80 through 145 provide for entry of data pairs from DATA statements rather than from the keyboard. Lines 125 through 140 provide a sample problem which can be used to check operation of the program. Line 105 operating with the 999's in the DATA statement in line 145 provide for termination of the input routine. The printout on page 14 was obtained using the problem in the data statements. Other methods of accomplishing the same effect are discussed elsewhere in this issue.

3. Line 315 illustrates the use of an underline in a variable name. This was previously demonstrated in Stephen Gutknecht's arrow game program in V13N3P12.

4. The first statement in line 330 illustrates a capability which is unique to TI BASIC, where

PSN,I=1 in TI BASIC is equivalent to the statements PSN=1:I=1

which must be used in most other BASIC implementations. The technique is described in the discussion of the LET keyword on page 2-63 of the TI-74 Programming Reference Guide. Similar statements appear in lines 380 and 405.

5. Several statements in this program illustrate a feature of the IF THEN ELSE statement that is not widely used, namely that actions can include groups of statements separated by colons. Thus, for line 495:

495 IF D2>M1 THEN M1=D2:S=I:T=J

if D2>M1 then M1, S and T are set to new values. If D2<=M1 no actions are taken and control passes directly to line 500. Several examples illustrating the effect are discussed on page 2-46 of the TI-74 Programming Reference Guide.

ERROR CODE DIFFERENCES BETWEEN THE CC-40 AND TI-74 - P. Hanson. While trying to demonstrate an alternate method for determining the end of the items in DATA statements I tried to translate a routine which had previously been used successfully with the CC-40. I found that my program wouldn't work when directly translated. I eventually traced the problem to a difference in the error code assigned for the error message "DATA error" for the two machines. I proceeded to compare the lists of error codes on pages K-11 and K-12 of the Compact Computer 40 User's Guide and page A-27 of the TI-74 Programming Reference Guide and found additional differences in error code assignments. I then compared the detailed descriptions of the error codes in pages K-2 through K-11 of the Compact Computer 40 User's Guide and pages A-16 through A-26 of the TI-74 Programming Reference Guide. The diagram on page 17 presents the results of my work. Note that:

- * Some error codes and error messages are the same; for example, "Bad Value" (code 04) and "Stack Underflow" (code 05).
- * Some error codes are different even though the error codes and the detailed descriptions are the same; for example, "Division by zero" which is code 34 with the CC-40 but code 26 with the TI-74.
- * In some cases, several error codes and error messages from the CC-40 have been combined into a single error code and message on the TI-74; for example, the "FOR/NEXT error" (code 6) on the TI-74 is the composite of three separate errors on the CC-40, "NEXT without FOR" (code 06), "Illegal FOR/NEXT nesting" (code 13) and "FOR without NEXT" (code 30). The obvious result of such combination is that the error indications from the TI-74 will not be as definitive as from the CC-40.

Error Code Differences between the CC-40 and TI-74 - (cont)

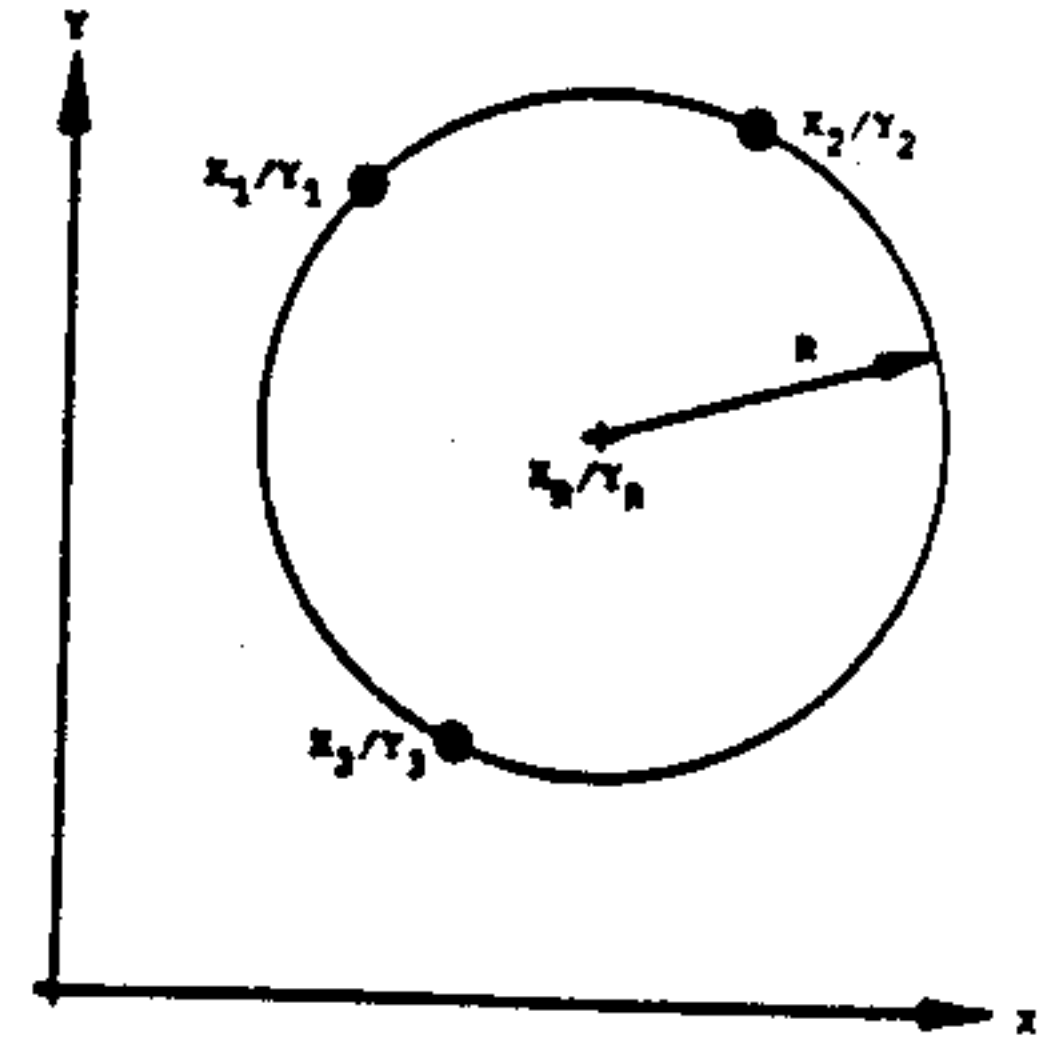
CORRESPONDENCE BETWEEN CC-40 AND TI-74 ERROR CODES

CC-40		TI-74	
Message	Code	Code	Message
I/O error	00	0	I/O error
Illegal syntax	01	1	Syntax
Expression too complex	02	2	Complex
String-number mismatch	03	3	Mismatch
Bad value	04	4	Bad value
Stack underflow	05	5	Stack underflow
NEXT without FOR	06	6	FOR/NEXT error
Bad INPUT data	07	7	Bad data
Invalid dimension	08	8	Bad dimension
Variable previously defined	09	9	Previously defined
Can't do that	10	10	Can't do that
Illegal after SUBEND	11	11	Line number error
Line reference out of range	12	12	Missing statement
Illegal FOR/NEXT nesting	13	13	Not found
Missing RETURN from error	14	14	Bad program type
Program not found	15	15	Protection error
Line not found	16	16	In use
Bad line number	17	17	Not defined
Bad program type	18	18	Image error
Illegal in program	19	19	File error
Protection violation	20	20	Name Table Full
Subprogram in use	21	21	Parenthesis
Variable not defined	22	22	Too long
Error in image	23	23	Bad Argument
File error	24	24	Extension missing
Name table full	25	25	Overflow
Unmatched parenthesis	26	26	Division by zero
Line too long	27	27	Contents may be lost
Name too long	28	28	Truncation
Bad argument	29	29	Break
FOR without NEXT	30	30	Initialized
BASIC extension missing	31	31	No RAM
Bad subscript	32	32	DATA error
Overflow	33		
Division by zero	34		
Memory contents may be lost	35		
String truncation	36		
Break	37		
System initialized	38		
Must be in subprogram	39		
No RAM in cartridge	40		
Statement must be first on line	41		
Missing SUBEND	42		
Data Error	43		
Must be in program	44		
RETURN without GOSUB	45		
System error	126	126	System error
Memory full	127	127	Memory full

The I/O Error Codes are the same for the two machines with the exception that the ILLEGAL IN SLAVE MODE (code 254) is not listed for the TI-74. See pages K-13 through K-16 of the Compact Computer 40 User's Guide and pages A-28 through A-31 of the TI-74 Programming Reference Guide.

CIRCLE THROUGH THREE POINTS - P. Hanson.

When work with the programming challenge on V12N4F26 required a routine for finding the circle which passes through three points I looked through past issues of our newsletter for an existing program. I found one for the TI-57 on V7N1/2P27. It was written by Ingvar Magnusson, and was originally published in the Swedish newsletter Programbiten. The program listing appears at the right. To run:



1. Enter X1 and press STO 0.
2. Enter Y1 and press STO 1.
3. Enter X2 and press STO 2.
4. Enter Y2 and press STO 3.
5. Enter X3 and press STO 4.
6. Enter Y3 and press STO 5.
7. Press RST R/S and see R in the display.
8. Press RCL 4 and see XR in the display.
9. Press RCL 5 and see YR in the display.

00	33	4	RCL	4	25	23	X ²	
01	-34	0	INV	SUM	0	26	75	+
02	-34	2	INV	SUM	2	27	33	0
03	33	5	RCL	5	28	39	7	PRD
04	-34	1	INV	SUM	1	29	25	X ²
05	-34	3	INV	SUM	3	30	85	=
06	33	2	RCL	2	31	39	3	PRD
07	95		X		32	39	2	PRD
08	33	1	RCL	1	33	33	3	RCL
09	65		-		34	-34	1	INV
10	33	0	RCL	0	35	33	7	RCL
11	95		X		36	-34	2	INV
12	33	3	RCL	3	37	33	6	RCL
13	95		X		38	-39	1	INV
14	32	6	STO	6	39	-39	2	INV
15	34	6	SUM	6	40	33	1	RCL
16	32	2	RCL	2	41	34	4	SUM
17	23		X ²		42	23		X ²
18	75		+		43	75		+
19	23	3	RCL	3	44	33	2	RCL
20	23		X ²		45	34	5	SUM
21	95		=		46	23		X ²
22	32	7	STO	7	47	95		=
23	38	1	EXC	1	48	24		FX
24	39	1	PRD	1	49	81		R/S

Conversion for the TI-59 is straightforward since there are corresponding TI-59 commands for each TI-57 command. The listing which follows provides for data entry with the User Defined keys and printout with the PC-100. A printout for a sample problem appears at the right. To use the program:

1. Enter X1 and press x t. Enter Y1 and press A.
2. Enter X2 and press x t. Enter Y2 and press B.
3. Enter X3 and press x t. Enter Y3 and press C.
4. To solve press D.

-6.
2.
4.
4.
0.
-6.
- .3913049478
- .0434782609
5.969360964

H
K
R

000	76	LBL	027	32	X:T	054	03	03	081	48	EXC	108	06	06	135	69	DP
001	11	H	028	99	PRT	055	43	RCL	082	01	01	109	22	INV	136	06	06
002	98	ADV	029	42	STO	056	02	02	083	49	PRD	110	49	PRD	137	02	2
003	32	X:T	030	04	04	057	65	7	084	01	01	111	01	01	138	06	6
004	99	PRT	031	32	X:T	058	43	RCL	085	33	X ²	112	22	INV	139	69	DP
005	42	STO	032	99	PRT	059	01	01	086	85	+	113	49	PRD	140	04	04
006	00	00	033	42	STO	060	75	-	087	43	RCL	114	02	02	141	43	RCL
007	32	X:T	034	05	05	061	43	RCL	088	00	00	115	43	RCL	142	05	05
008	99	PRT	035	91	R/S	062	00	00	089	49	PRD	116	01	01	143	69	DP
009	42	STO	036	76	LBL	063	65	7	090	07	07	117	44	SUM	144	06	06
010	01	01	037	14	D	064	43	RCL	091	33	X ²	118	04	04	145	03	3
011	91	R/S	038	98	ADV	065	03	03	092	95	=	119	33	X ²	146	05	5
012	76	LBL	039	43	RCL	066	95	=	093	49	PRD	120	85	+	147	69	DP
013	12	B	040	04	04	067	42	STO	094	03	03	121	43	RCL	148	04	04
014	98	ADV	041	22	INV	068	06	06	095	49	PRD	122	02	02	149	32	X:T
015	32	X:T	042	44	SUM	069	44	SUM	096	02	02	123	44	SUM	150	69	DP
016	99	PRT	043	00	00	070	06	06	097	43	RCL	124	05	05	151	06	06
017	42	STO	044	22	INV	071	43	RCL	098	03	03	125	33	X ²	152	98	ADV
018	02	02	045	44	SUM	072	02	02	099	22	INV	126	95	=	153	91	R/S
019	32	X:T	046	02	02	073	33	X ²	100	44	SUM	127	34	FX	154	00	0
020	99	PRT	047	43	RCL	074	85	+	101	01	01	128	32	X:T	155	00	0
021	42	STO	048	05	05	075	43	RCL	102	43	RCL	129	02	2	156	00	0
022	03	03	049	22	INV	076	03	03	103	07	07	130	03	3	157	00	0
023	91	R/S	050	44	SUM	077	33	X ²	104	22	INV	131	69	DP	158	00	0
024	76	LBL	051	01	01	078	95	=	105	44	SUM	132	04	04	159	00	0
025	13	C	052	22	INV	079	42	STO	106	02	02	133	43	RCL			
026	98	ADV	053	44	SUM	080	07	07	107	43	RCL	134	04	04			

Circle through three points - (cont)

When operating without a PC-100 the calculator will stop with the radius of the circle in the display. Then press RCL 04 to see the x coordinate of the center of the circle (H) in the display, and press RCL 05 to see the y coordinate of the center of the circle (K).

For a conversion for the TI-74 the data input and data output routines must be changed to accommodate the TI-74 and the associated peripherals. For example, in the listing on the next page lines 10 through 70 are similar to the output selection routines used in TI-74 programs in earlier issues. A "brute force" translation of the solution sequence can be obtained by designating variables A through H to correspond to data registers R00 through R07 of the TI-59. It will also be helpful to designate a variable X to correspond to the display register of the TI-59. Some example translations follow:

a. Steps 055-070 of the TI-59 program which calculate two products ($R02 \times R01$ and $R00 \times R03$) and place twice the difference in R06 can be translated into the statements on lines 220 and 225 of the BASIC program for the TI-74.

b. Translation of TI-59 commands such as the EXC 01 at steps 081-082, which exchange the contents of the display register with data register R01, cannot be directly translated since there is no equivalent command in BASIC. One alternative is to use yet another variable in the BASIC program, say register Y, and write a sequence of statements $Y = X$, $X = B$ and $B = Y$. In this translation I substituted equivalent commands based on more thorough analysis of what was actually being done in the program.

The reader is invited to compare other segments of the BASIC program below with the TI-59 program listing on page 24 and satisfy himself of the equivalence of the two programs.

10 AS="Circle Thru 3 Poi nts":PRINT AS:PAUSE 1	\$:A:PRINT @PN,X\$:A	230 H=C+C+D*D
20 INPUT "Use printer? Y /N ":Z\$	105 X\$="Y(1) = ":INPUT X	235 X=B:B=H
30 IF Z\$="Y"DR Z\$="y"THE N PN=1 ELSE 100	\$:B:PRINT @PN,X\$:B	240 B=X*B
35 PRINT "Device Numbers ":PAUSE 1	110 PRINT @PN	245 H=A*H
40 PRINT "For the HX-100 0 enter 10":PAUSE 1	115 X\$="X(2) = ":INPUT X	250 X=X*X+A*A
45 PRINT "For the PC-324 enter 12":PAUSE 1	\$:C:PRINT @PN,X\$:C	255 D=X*D
50 INPUT "Enter device n umber ":D\$	120 X\$="Y(2) = ":INPUT X	260 C=X*C
55 OPEN #1,D\$,OUTPUT	\$:D:PRINT @PN,X\$:D	265 B=B-D
60 IF D\$="10"THEN PRINT #1,CHR\$(18)	125 PRINT @PN	270 C=C-H
65 PRINT #1:PRINT #1,AS	130 X\$="X(3) = ":INPUT X	275 B=B/G
70 PRINT #1	\$:E:PRINT @PN,X\$:E	280 C=C/G
100 X\$="X(1) = ":INPUT X	135 X\$="Y(3) = ":INPUT X	285 E=E+B
	\$:F:PRINT @PN,X\$:F	290 F=F+C
	140 PRINT @PN	295 R=SQR(B*B+C*C)
	200 A=A-E	400 PAUSE ALL
	205 C=C-E	410 PRINT @PN," h = ":E
	210 B=B-F	420 PRINT @PN," k = ":F
	215 D=D-F	430 PRINT @PN," r = ":R
	220 G=C+B-A*D	440 PAUSE 0:PRINT @PN
	225 G=G+G	999 END

Lines 200-295 will be used in the solution of the programming challenge to find the smallest circle to enclose 4 or more points.

BEST FIT CIRCLE - At one time it appeared that we might need a routine for finding the best fit circle through four or more points in the solution to the programming challenge on V12N4P26. Appropriate equations are defined in William Kolb's Curve Fitting for Programmable Calculators. The "Circle Best Fit" program in Maurice Swinnen's Statistical Library book (see V12N3P4) provides an equivalent program in Sharp BASIC which is readily translatable for TI-74 BASIC. The resulting program listing is on page 27. A complete set of prompts are provided. Sample problems are illustrated in the right hand column on page 27.

Lines 10 through 70 provide for selection of the display, the PC-324 or the HX-1000 for output.

Lines 100 through 190 provide for data entry using the string input techniques used in Maurice Swinnen's books.

Lines 200 through 370 are a direct translation from Maurice's book.

Lines 400 through 440 provide for output of the solution to the display or to the printer based on the value of the variable FN which was defined in line 30. The PAUSE ALL statement in line 400 combined with a value of FN of zero causes calculation to stop as each parameter is displayed when a printer is not used.

Lines 500 through 690 provide an ability to calculate points on the defined circle.

Lines 700-770 provide for adding points to the input array for a revised solution.

DIFFERENT PRINT AND DISPLAY FOR SOME TI-74/PC-324 CHARACTERS

In line 355 of the taper bore check portion of the Menu and Module program on V13N2P21 Don Laughery wanted to use the right arrow (ASCII 126, CTL 4) in the prompts to the user. To his surprise he found that while the code displays as a right arrow, it prints as a tilde (~). He asked if I was aware of other situations such as this. While working on another problem I stumbled on the paragraph at the bottom of page 3-22 of the TI-74 Programming Reference Guide which indicates that there are four ASCII codes for which the PC-324 printout differs from the TI-74 display:

ASCII Code	Keystrokes	Display	Printout
92	CTL /	∨	\
124	CTL 1		!
126	CTL 4	→	~
127		←	space

Best Fit Circle - (cont)Program ListingExamples

```

10 AS="Best Fit Circle":
PRINT AS:PAUSE 1
20 DIM X(50),Y(50)
25 INPUT "Use printer? Y
/N ";Z$
30 IF Z$="Y"OR Z$="y"THE
N PN=1 ELSE 100
35 PRINT "Device Numbers
:":PAUSE 1
40 PRINT "For the HX-100
0 enter 10":PAUSE 1
45 PRINT "For the PC-324
enter 12":PAUSE 1
50 INPUT "Enter device n
umber ";DS
55 OPEN #1,DS,OUTPUT
60 IF DS="10"THEN PRINT
#1,CHR$(18)
65 PRINT #1:PRINT #1,AS
70 PRINT #1
100 PRINT "End Input by
Entering E":PAUSE 1
110 N=1
120 XS="X = "
125 YS="Y = "
130 INPUT XS;XXS:IF XXS=
"E"OR XXS="e"THEN 190
135 INPUT YS;YYS:IF YYS=
"E"OR YYS="e"THEN 190
140 X(N)=VAL(XXS)
145 Y(N)=VAL(YYS)
150 IF PN=0 THEN 180
155 PRINT #PN,XS;X(N)
160 PRINT #PN,YS;Y(N)
170 PRINT #PN
180 N=N+1:GOTO 130
190 N=N-1
200 S6=0:S7=0:S8=0:S9=0:
T0=0:T1=0:U6=0:U9=0:V0=0
:V2=0
210 FOR I=1 TO N
220 U=X(I):V=Y(I)
230 S6=S6+U:S7=S7+U*U:S8
=S8+V:S9=S9+V*V:T0=T0+U*
V:T1=T1+1
240 U6=U6+V*U:U9=U9+U*
V*V:V0=V0+U*U:V2=V2+V*
V*V
250 NEXT I
300 R5=T0*T1-S6*S8:R6=(S
7+S9)*S8-(U6+V2)*T1
310 R7=S8*S8-S9*T1:R8=(S
7+S9)*S6-(U9+V0)*T1:R9=S
6*S6-S7*T1
320 IF R8=0 THEN PRINT "

```

```

Solution Undefined":STOP
330 S0=R7+R9-R5*R5
340 H=(R5*R6+R7*R8)/(2*S
0)
350 K=(R5*R8+R6*R9)/(2*S
0)
360 AA=2*(H*S6+K*S8):BB=
(H*H+K*K)*T1
370 R=SQR((S7+S9-AA+BB)/
T1)
400 PAUSE ALL
410 PRINT #PN," h = ";H
420 PRINT #PN," k = ";K
430 PRINT #PN," r = ";R
440 PAUSE 0:PRINT #PN
500 PS="Predict y for x"
505 PRINT #PN,PS:PRINT #
PN
510 INPUT PS;" = ";XXS
520 IF XXS="E"OR XXS="e"
THEN 600
530 XX=VAL(XXS)
540 YY=SQR(ABS(R*R-(XX-H
)^2))+K
550 PAUSE ALL
560 PRINT #PN,XS;XX
570 PRINT #PN,YS;YY
580 PAUSE 0:PRINT #PN
590 GOTO 510
600 PS="Predict x for y"
605 PRINT #PN,PS:PRINT #
PN
610 INPUT PS;" = ";YYS
620 IF YYS="E"OR YYS="e"
THEN 700
630 YY=VAL(YYS)
640 XX=SQR(ABS(R*R-(YY-K
)^2))+H
650 PAUSE ALL
660 PRINT #PN,YS;YY
670 PRINT #PN,XS;XX
680 PAUSE 0:PRINT #PN
690 GOTO 610
700 PS="Add more data (Y
/N) ? "
710 INPUT PS;Z$
720 IF Z$="N"OR Z$="n"TH
EN 999
730 PRINT #PN,"Additiona
l Input Data"
740 PRINT #PN
750 N=N+1
760 GOTO 120
770 GOTO 120
999 END

```

Best Fit Circle

X = 12
Y = 1

X = 14
Y = 3

X = 15
Y = 6

X = 14
Y = 9

h = 9.357142857
k = 5.948979592
r = 5.577611132

Predict y for x

X = 4
Y = 7.501643866

X = 5
Y = 9.431084137

Predict x for y

Y = 9
X = 14.02629914

Additional Input Data

X = 12
Y = 11

X = 9
Y = 12

h = 8.976331361
k = 6.035502959
r = 5.89669835

Predict y for x

X = 4
Y = 9.198915297

Predict x for y

Y = 9
X = 14.07366485

BIORHYTHMS ON THE TI-74 - A biorhythms program for the TI-95 was published in the Volume 2, Number 2 issue of Programmable Calculator News. Page 2 of this issue notes that there were errors in the listing at lines 210 and 338. When members asked about a similar program for the TI-74 I decided that the conversion would provide another good example of translation from the TI-59 or TI-95 to the TI-74. The methodology I selected was the same as for the translation of the program for a circle through three points on pages 25 and 26. The task was easier in this case because the TI-95 program was already using alphabetic addressing for the data registers and the TI-74 translation simply assigned a letter variable to match each TI-95 data register. The variable X was assigned to simulate the display register of the TI-95. A listing of the translation for the TI-74 and some sample printouts follow.

Program Listing

<pre> 1000 AS="Biorhythms":PRINT AS:PAUSE 1 1010 INPUT "Use printer? Y/N ";Z\$ 1020 IF Z\$="Y"OR Z\$="y"THEN PN=1 ELSE 1100 1030 PRINT "Device Numbers:":PAUSE 1 1040 PRINT "For the HX-1000 enter 10":PAUSE 1 1050 PRINT "For the PC-324 enter 12":PAUSE 1 1060 INPUT "Enter device number ";DS 1070 OPEN #1,DS,OUTPUT 1080 IF DS="10"THEN PRINT #1,CHR\$(18) 1090 PRINT #1:PRINT #1,A\$:PRINT #1 1100 BS="Birthday ";TS="Today ";IS="(mdd.yyyy)?" ":IF PN=0 THEN PAUSE ALL 1105 DEG:Z=1:INPUT BS&IS:DS 1110 GOTO 1800 1140 PRINT #PN,BS&" ":DS:IF PN=1 THEN PRINT #1 1145 E=X 1150 Z=2:INPUT TS&IS:DS 1155 GOTO 1800 1190 PRINT #PN,TS&" ":DS:IF PN=1 THEN PRINT #1 1195 F=X 1200 G=F-E 1210 PRINT #PN,G;"days old" 1220 IF PN=1 THEN PRINT #1 1230 DS="Days into cycle" </pre>	<pre> e " 1240 PS=" % through cycle" e " 1250 MS=" Amplitude" " 1300 PRINT #PN,"Physical Cycle:" 1305 X=G/23:X=X-INT(X):M=X:X=23*X 1310 X=X+.1:X=INT(X) 1315 H=X:X=X/23:X=1000*X 1320 X=INT(X):X=X/10:T=X 1325 GDSUB 1700 1400 PRINT #PN,"Emotional Cycle:" 1405 X=G/28:X=X-INT(X):M=X:X=28*X 1410 X=X+.1:X=INT(X) 1415 H=X:X=X/28:X=1000*X 1420 X=INT(X):X=X/10:T=X 1425 GDSUB 1700 1500 PRINT #PN,"Intellectual Cycle:" 1505 X=G/33:X=X-INT(X):M=X:X=33*X 1510 X=X+.1:X=INT(X) 1515 H=X:X=X/33:X=1000*X 1520 X=INT(X):X=X/10:T=X 1525 GDSUB 1700 1600 INPUT "Another solution (Y/N)? ";Z\$ 1605 IF Z\$="Y"OR Z\$="y"THEN 1610 ELSE END 1610 IF PN=1 THEN 1090 ELSE 1100 1700 IF PN=1 THEN PRINT #1 1705 PRINT #PN,DS:H 1710 IF PN=1 THEN PRINT #1 1715 PRINT #PN,PS:T 1720 IF PN=1 THEN PRINT </pre>	<pre> #1 1725 M=INT(1000*SIN(360*M))/1000 1730 PRINT #PN,MS:M 1735 IF PN=1 THEN PRINT #1 1740 RETURN 1800 X=VAL(D\$):B=X:GDSUB 1950:X=0 1805 GDSUB 1950:IF X<A THEN 1900 1810 X=X-INT(X):B=B-X:X=10000*X 1815 D=X:GDSUB 1950:X=15 181:IF X>A THEN 1900 1820 X=32:GDSUB 1950:X=B :X=X/100:B=X 1825 X=X-INT(X):B=B-X:X=100*X:C=X:IF X>A THEN 1900 1830 X=13:GDSUB 1950:X=B :IF X>A THEN 1900 1835 X1=365*D+C+31*B-31 1840 X=3:GDSUB 1950:X=B :IF X>A THEN 1960 1845 D=D-1 1850 X=X1+INT(D/4) 1855 X1=INT(D/100) 1860 X1=.75+.75*X1 1865 X1=INT(X1) 1870 X=X-X1 1895 DN Z GOTO 1140,1190 1900 ES="Entry error: try again (Y/N)?" 1910 INPUT ES:ES 1920 IF ES="N"OR ES="n"THEN 1999 1930 DN Z GOTO 1100,1150 1950 T=X:X=A:A=T:RETURN 1960 X1=X1-INT(.4*B+2.3) :GOTO 1850 1999 END </pre>
---	--	---

Biorhythms for the TI-74 - (cont)Sample Printouts

Biorhythms		Biorhythms		Biorhythms	
Birthday	912.1951	Birthday	913.1946	Birthday	222.1954
Today	608.1988	Today	423.1949	Today	901.1977
13419 days old		953 days old		8592 days old	
Physical Cycle:		Physical Cycle:		Physical Cycle:	
Days into cycle	10	Days into cycle	10	Days into cycle	13
% through cycle	43.4	% through cycle	43.4	% through cycle	56.5
Amplitude	.398	Amplitude	.398	Amplitude	-.399
Emotional Cycle:		Emotional Cycle:		Emotional Cycle:	
Days into cycle	7	Days into cycle	1	Days into cycle	24
% through cycle	25	% through cycle	3.5	% through cycle	85.7
Amplitude	.999	Amplitude	.222	Amplitude	-.782
Intellectual Cycle:		Intellectual Cycle:		Intellectual Cycle:	
Days into cycle	21	Days into cycle	29	Days into cycle	12
% through cycle	63.6	% through cycle	87.8	% through cycle	36.3
Amplitude	-.756	Amplitude	-.691	Amplitude	.755

Some comments on the translation:

I have not provided a copy of the TI-95 program. If you have not yet subscribed to Programmable Calculator News you should do so as soon as possible.

a. Lines 1000 through 1090 provide for selection of the display, the PC-324 or the HX-1000 for output.

b. Line 1100 defines some string variables which are used more than once in the program.

c. Lines 1105-1195 are the equivalent of the LBL AA portion of the TI-95 program (steps 0270 ff).

d. Lines 1800-1895 are the equivalent of the LBL E' portion of the TI-95 program. In this routine I tried to make a true "brute force" translation. For example, with a value in the display register of the TI-95 a STO B command is translated into $B = X$ for the TI-74. Similarly, a FRC command in the TI-95 is translated into $X = X - \text{INT}(X)$ for the TI-74. Where nesting of parenthesis was involved it was necessary to define additional variables, e.g., X1, to hold intermediate values as in line 1835 of the TI-74 program. This is equivalent to the storage of nested values in the hierarchy registers of the TI-59 or TI-95.

Biorhythms for the TI-74 - (cont)

e. Lines 1900-1930 are the equivalent of the LBL M2 portion of the TI-95 program and provide for input error recovery.

f. Line 1950 provides the equivalent of the EXC A command which is used many times in the TI-95 program.

g. Line 1960 is the equivalent of the LBL M1 portion of the TI-95 program. If you are going to pursue this brute force method of conversion you should be sure you understand the conditions at the call and return for this subroutine.

h. Lines 1700-1740 are the equivalent of the LBL BB portion of the TI-95 program. Lines 1725-1730 are not in the TI-95 program. They were added to provide the "Amplitude" output for each cycle. I have found that parameter to be of interest to people to study biorhythms. The mechanization is equivalent to that in the Biorhythms program (LE-21) in the Leisure Library module for the TI-59.

i. Lines 1600-1610 provide the capability to obtain a second solution without going back through the output device selection routine.

TI PC NOTES

V13N4P3

MAKE YOUR OWN PAPER FOR THE PC-324 - Carl Rabe writes: I recently purchased a PC-324 printer and immediately noticed the paper problems that you wrote of in V13N1 (short rolls and high cost). So, I took a large roll of TP-30250 paper for the PC-100 and put it in the vise. With a new fine toothed hack saw blade I carefully cut off the end of the roll to reduce it to a width of 2 1/4 inches. After cutting it off I rubbed the new edge on a cloth to remove dust and smooth it a little. I used a small hairpin to start a roll. I rolled the paper tightly into a roll about the right size to just fit in the back of the PC-324. I was able to put about fifteen feet on a roll. So far it has worked fine in the PC-324.

Editor's Notes: The contrast on some sample printouts received from Carl is about the same as that obtained with normal PC-324 paper. The contrast is decidedly inferior to that obtained with Radio Shack Catalog No. 26-3592B paper. As noted in V13N3P24 the improved contrast only occurs with the catalog number with a "B" suffix. There is still some "no suffix" paper and some "A" suffix paper still on the shelves of Radio Shack stores, so be sure to look for and get the "B" suffix. The price is the same.

Carl also wrote that he was going to try to suspend the rest of the PC-100 paper roll horizontally in a small cardboard box to put behind the PC-324. He will feed the paper into the back of the printer with the cover slightly ajar. He hasn't reported on the results yet. Of course, that approach is similar in concept to the retractable spindle provided with the HX-1000 Printer/Plotter.

I have searched for larger rolls of 2 1/4 inch wide thermal paper, but so far have only been able to find the blue printing version. Has any member found a source of black printing 2 1/4 inch wide paper on larger rolls?

A "MENU AND MODULE" PROGRAM FOR THE TI-74 - D. Laughery and P. Hanson

One of the attractive features of the TI-95 is the built-in file system capability which permits the user to store programs and data files in either the calculator's file space or in a Constant Memory Cartridge. The result is that the user can build a library of programs which can be easily accessed. The Radio Shack Model 100 offers a similar capability where one can maintain up to nineteen data files or BASIC programs in the user RAM at one time. A menu screen provides immediate access to any of the files or programs. The software cartridges provided for the TI-74 provide a similar kind of menu driven access to a variety of programs, but no such capability is built-in for user-generated BASIC programs. How might we develop a capability to store a several BASIC programs simultaneously in user RAM of the TI-74? We could simply write the various routines in sequence in user RAM and access the desired routine by

- * using RUN statements with line numbers to enter the program at the appropriate location, or by
- * providing a menu in user RAM with access to the appropriate routine through the use of ON-GOTO or ON-GOSUB statements.

Both of those methods have the disadvantage that the user must use care to keep track of the variables and the dimensioning of variables in the various routines to avoid conflicts. A better solution is to use menu driven subprograms. A sample program was written to demonstrate the concept. The sample program:

- * Displays or prints a directory of the subprograms available which was modeled on that used in the software cartridges.
- * Demonstrates the use of argument lists with CALL statements and parameter lists with SUB statements to provide transfer of data between several subprograms.
- * Demonstrates the call of one subprogram from within another.
- * Runs Don Laughery's confidence limits and taper programs which are described elsewhere in this issue.
- * Provides subprograms to enter a series of numbers, calculate some statistics on the numbers, and sort the numbers using a heap sort technique.

The program listing appears on page 21. Comments on the details of the program follow:

Lines 100-195 mechanize the menu function. The dimension statement at line 100 is needed to permit transfer of array data between three of the module programs. The accept statement at line 120 converts all alphanumeric input to uppercase to ensure compatibility with the call statements to follow. The call statements for the SORT, STATS and INPUT subprogram modules include argument lists to permit transfer of data between the subprograms. Additional modules can be added by adding similar IF...THEN CALL... statements between lines 120 and line 195. A printout from the DIR subprogram appears at the right.

CONF	Confidence Bounds
TAPER	Bore check w balls
AREA	Find polygon area
PRINT	Printer Control
INPUT	Vector Entry
STATS	Vector Statistics
SORT	Vector Sorting
DIR	Directory

"Menu and Module" Program for the TI-74 - (cont)

Lines 200-295 are a subprogram which calculates confidence bounds based on an article on page 59 of the February 1989 issue of Quality. The routine is described in detail with a sample printout on page 26. Note that the variable X can be used in the subprogram even though the variable X was defined as an array in the main program. This could not be done if the module had been implemented with ON GOTO or ON GOSUB statements.

Lines 300-450 are a subprogram by Don Laughery for performing taper bore check calculations. The routine is described in more detail on page 18.

Lines 500-580 are a subprogram which calculates the area enclosed by a number of vertices. The routine is essentially the same as that on V11N2P22.

Lines 600-650 are a subprogram which is called by five other subprograms (CONF, SORT, STATS, INPUT and DIR) to provide printer setup if desired. The argument PN is needed to pass the variable which defines whether to print or display between the calling subprograms and the PRINT subprogram.

Lines 700-790 are a subprogram which provides the ability to build a one dimensional array which can be accessed by other subprograms. Note that it would not be necessary to use the same array name as in the main program.

Lines 800-895 are a subprogram which calculates elementary statistics for an one dimensional array. The program was adapted from the one on V11N3P21. The elements of the array must be provided by another subprogram, in this demonstration by the INPUT subprogram. It is not necessary to use the same array name as in the main program or as in the INPUT subprogram.

Lines 900-990 are a subprogram which performs a heap sort on a one dimensional array. The routine is essentially the same as that on V10N3P13-16 which was adapted from a program on page 137 of the September 1980 issue of Creative Computing. Again, it is not necessary to use the same array name as in the main program or as in the INPUT subprogram.

The sample printout at the right illustrates the output from use of the INPUT, STATS and SORT subprograms.

Lines 1000-1099 are a subprogram which either prints or displays a directory of the subprograms in the menu. Additional DATA statements should be added between lines 1040 and 1098 to match any subprograms which are added. The null string for the data in line 1098 is used by line 1020 to detect the end of the directory.

X(1)	.8225408469
X(2)	.163164479
X(3)	.4352797351
X(4)	.3233738121
X(5)	.1733442624
X(6)	.5765405918
X(7)	.6272445458
X(8)	.4822732445
X(9)	.9196005246
X(10)	.29828705

Min	= .163164479
Max	= .9196005246
Mean	= .4821649092
S.D.	= .2572322652
RMS	= .5404022521
RSS	= 1.708901969

Sorted List:

.163164479
.1733442624
.29828705
.3233738121
.4352797351
.4822732445
.5765405918
.6272445458
.8225408469
.9196005246

"Menu and Module" Program for the TI-74 - (cont)Program Listing

```

100 DIM X(100)
105 DISPLAY "Enter DIR t
o display directory"
110 PAUSE 3
115 DISPLAY "Program Nam
e?":
120 ACCEPT AT(15)VALIDAT
E(URLPHA),AS
125 IF AS="TAPER"THEN CA
LL TAPER
130 IF AS="CONF"THEN CAL
L CONF
135 IF AS="AREA"THEN CAL
L AREA
140 IF AS="SDRT"THEN CAL
L SDRT(N),X()
145 IF AS="STATS"THEN CA
LL STATS(N),X()
150 IF AS="INPUT"THEN CA
LL INPUT(N),X()
155 IF AS="PRINT"THEN CA
LL PRINT
190 IF AS="DIR"THEN CALL
DIR
195 GOTO 115
200 SUB CONF
205 REM Calculation of c
onfidence, D. Laushery,
2/8/89
210 REM Adapted from pag
e 59 of Feb 1989 Quality
215 CALL PRINT(PN):PAUSE
ALL
220 INPUT "Pop. % bounde
d by sample? "IP
225 IF PN=1 THEN PRINT #
1:P:" percent confidence
"IPRINT #1
230 IF PN=1 THEN PRINT #
1:"Sample Conf"
IPRINT #1
240 P=F/100
245 X=1-P
250 FOR N=3 TO 100
260 IF N>15 THEN N=N+4
270 CONF2=(INT(1000*(1-(
P^N+N*X*(P^(N-1)))))/10
00
280 IF PN=0 THEN DISPLAY
:"Sample = "IN:"Conf = "
:CONF2:GOTO 290
285 PRINT #1,N,CONF2
290 NEXT N
295 SUBEND
300 SUB TAPER
305 REM Taper bore check
: Don Laushery, 2/16/89
310 DEG:PAUSE ALL
315 INPUT "Large Ball Di
a.=? "D1:R1=D1/2
320 INPUT "Small ball di
a.=? "D2:R2=D2/2
325 INPUT "Dim from face
to large ball=? "A:A=A
+R1
330 INPUT "Dim from face
to small ball=? "B:B=B
+R2
335 N=B-A:ANG=ASIN((R1-R
2)/N)
340 DF=2*((A+R1+SIN(ANG)
)+TAN(ANG)+R1+COS(ANG))
345 DISPLAY USING 410:DF
350 DISPLAY USING 420:2*
ANG
355 DISPLAY "1=Dia-Depth
/2=Depth-Dia/3=New "
360 ACCEPT VALIDATE("123
")SIZE(1),X
365 DN X GOTO 370,390,32
5
370 INPUT "Enter Specifi
ed Depth "IT
375 DT=DF-2*(T*TAN(ANG))
380 DISPLAY USING 430:DT
IT
385 GOTO 355
390 INPUT "Enter Specifi
ed Diameter "DT
395 T=((DF-DT)/2)/TAN(AN
G)
400 DISPLAY USING 440:T,
DT
405 GOTO 355
410 IMAGE FACE DIA=###.#
###
420 IMAGE INCL ANGLE= ##
#.### DEG
430 IMAGE DIA=###.### A
T ###.### DEPTH
440 IMAGE DEPTH= ###.###
@ AT ###.### DIA
450 SUBEND
500 SUB AREA
505 REM Area Finder from
V11N2P22, TI PPC Notes
510 DIM X(31),Y(31)
515 INPUT "Number of ver
tices? "IN
520 IF N<3 OR N>30 THEN
515
525 FOR I=1 TO N
530 INPUT "X("&STRS(I)&"
)=? "X(I)
535 INPUT "Y("&STRS(I)&"
)=? "Y(I)
540 NEXT I
545 X(N+1)=X(1):Y(N+1)=Y
(1)
550 S=0
555 FOR I=1 TO N
560 S=S+X(I)*Y(I+1)-X(I+
1)*Y(I)
565 NEXT I
570 PRINT "Area = "S/2
575 PAUSE
580 SUBEND
600 SUB PRINT(PN)
605 REM Printer Control
Routine
610 DISPLAY "Use printer
? ":
615 ACCEPT AT(14)VALIDAT
E(URLPHA),AS
620 IF AS="Y"THEN PN=1 E
LSE 650
625 PRINT "PC324, use 12
: HX1000, use 10":PAUSE
2
630 INPUT "Enter device
number "IDS
635 OPEN #1,DS,OUTPUT
640 IF DS="10"THEN PRINT
#1,CHRS(18)
645 PRINT #1
650 SUBEND
700 SUB INPUT(N,X())
705 REM Vector Entry Rou
tine
710 CALL PRINT(PN)
715 INPUT "Number of ele
ments? "IN
720 IF N>100 THEN 710
725 FOR I=1 TO N
730 AS="X("&STRS(I)&" "
735 INPUT AS&"=? "X(I)
740 IF PN=1 THEN PRINT #
1:AS:X(I)
745 NEXT I
750 IF PN=1 THEN PRINT #
1:CLOSE #1
755 PAUSE 0
790 SUBEND
800 SUB STATS(N,X())
805 REM statistics for V
ector Elements - adapted
from V11N3P21 of TI PPC
Notes
810 CALL PRINT(PN)
815 MIN=X(1):MAX=X(1)
820 FOR I=1 TO N
825 S1=S1+X(I)
830 S2=S2+X(I)*X(I)
835 IF X(I)<MIN THEN MIN
=X(I)
840 IF X(I)>MAX THEN MAX
=X(I)
845 NEXT I
850 PAUSE ALL
855 PRINT #PN,"Min = ":
MIN
860 PRINT #PN,"Max = ":
MAX
865 PRINT #PN,"Mean = ":
S1/N
870 PRINT #PN,"S.D. = ":
SOR((S2-S1*S1/N)/(N-1))
875 PRINT #PN,"RMS = ":
SOR(S2/N)
880 PRINT #PN,"RSS = ":
SOR(S2)
885 IF PN=1 THEN PRINT #
1:CLOSE #1
890 PAUSE 0
895 SUBEND
900 SUB SDRT(N,X())
902 REM Heap sort adapte
d from page 137 of Sept
1980 Creative Computing
904 REM Also see V10N3P1
3-16 of TI PPC Notes
906 CALL PRINT(PN)
908 M=N
910 FOR L=INT(N/2)TO 1 S
TEP -1
912 B=X(L)
914 GDSUB 960
916 NEXT L
918 L=1
920 FOR M=N-1 TO 1 STEP
-1
922 B=X(M+1):X(M+1)=X(L)
924 GDSUB 960
926 NEXT M
928 PAUSE ALL
930 PRINT #PN,"Sorted Li
st:"
932 IF PN=1 THEN PRINT #
1
934 FOR I=1 TO N
936 PRINT #PN,X(I)
938 NEXT I
940 IF PN=1 THEN PRINT #
1:CLOSE #1
942 PAUSE 0
944 SUBEXIT
960 REM Subrout:ne
962 I=L
964 J=I+1
966 IF J>M THEN 978
968 IF J=M THEN 972
970 IF X(J+1)>X(J)THEN J
=J+1
972 IF B>X(J)THEN 978
974 X(I)=X(J):I=J
976 GOTO 964
978 X(I)=B
980 RETURN
990 SUBEND
1000 SUB DIR
1005 REM Directory of Pr
ograms
1010 CALL PRINT(PN)
1015 PAUSE ALL
1020 READ AS:IF AS="THE
N 1030
1025 PRINT #PN,AS:GOTO 1
020
1030 IF PN=1 THEN PRINT
#1:CLOSE #1
1035 PAUSE 0:SUBEXIT
1040 REM Listing of Subr
outines
1051 DATA CONF Confiden
ce Bounds
1052 DATA TAPER Bore che
ck w balls
1053 DATA AREA Find pol
ygon area
1054 DATA PRINT Printer
Control
1055 DATA INPUT Vector E
ntry
1056 DATA STATS Vector S
tatistics
1057 DATA SDRT Vector S
orting
1097 DATA DIR Director
y
1098 DATA ""
1099 SUBEND

```

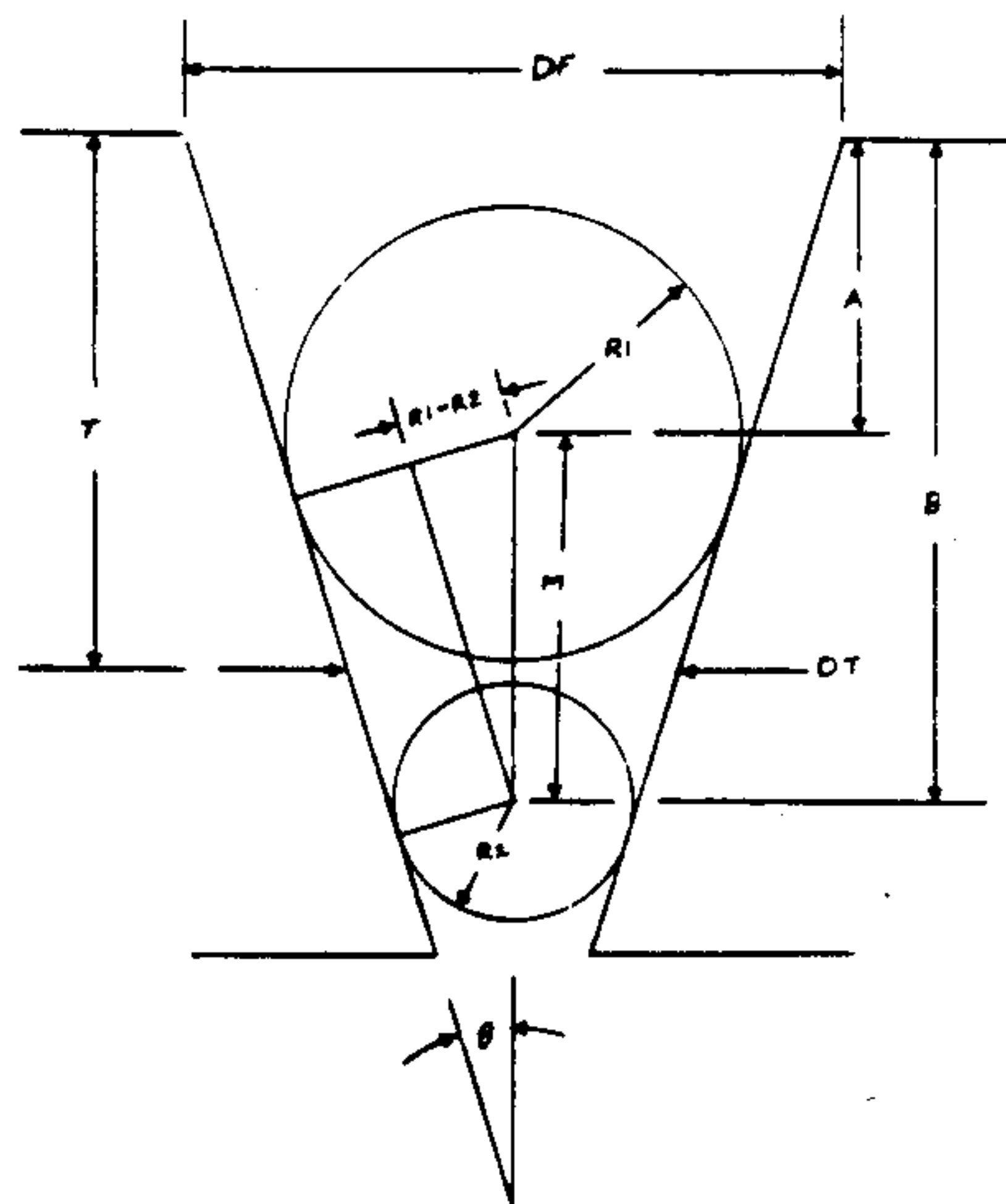
Real Estate/Investment Library Module for the TI-59. The pocket-sized Quick Reference Guide is available. The full-sized book is not. Send five dollars to TI PPC Notes.

TAPER BORE CHECK - Don Laughery.

This is one of the options in the "menu and module" TI-74 demonstration program on pages 19-21. The program listing is lines 300 through 450 on page 21. The program solves for taper bore parameters given input of the diameters of two precision balls and the dimensions from the face to the top of the balls when placed in the taper bore. See the diagram at the right. The larger ball's diameter must not exceed the face diameter times the secant of the half-angle. The smaller ball's diameter must be larger than the exit diameter times the secant of the half-angle. The dimension from the face to the top of the larger ball is negative if the ball projects above the face. To check your entry of the program use the following sample problem:

Large Ball Diameter = 1.0625
 Small Ball Diameter = 1.000
 Face to Large Ball = 0.410
 Face to Small Ball = 2.950

Then, the Face Diameter is 1.0860 and the Included Angle is 1.4274 degrees.



TI PPC NOTES

V13N1P4

A SURVEYING MODULE FOR THE TI-74 AND TI-95 - The Survey-6 module for the TI-74 is an integrated coordinate geometry program designed for land survey calculations. It performs traverse, inverse, intersections and resection. The integrated program has the ability to store and recall coordinate pairs by your chosen name/number or automatic consecutive numbers. More than one coordinate pair with the same name, such as 'TANK' can be resident in memory because only points stored with the automatically attached current job code are normally visible.

Field staking calculations include stationed horizontal curve (offset from centerline, tangent, chord, secant), stationed vertical curve, stationed straight grade, vertically offset EDM reduction, and triangle solutions (sss, sas, asa, ssa). Printer output and cassette tape storage of named/numbered coordinates is also possible.

The program is currently available on a module for the TI-74 for ninety dollars. It is expected to be available on a module for the TI-95 in the near future at the same price. For further information write to Ron W. Hardy, 2879 Lassen Ct., Las Vegas NV 89122, or call (702)-459-4312.

Editor's Note: This announcement does not constitute an endorsement by our club. It is provided for the possible use of members who are surveyors. If you decide to purchase the program please consider writing an evaluation for publication in a future issue.

CALCULATION OF CONFIDENCE BOUNDS - Don Laughery

This program is one of the options in the "menu and module" program for the TI-74 on page 19. The program is an adaptation of a program in Gerald Hurayt's article "Bounded by Extreme Values" which appeared on page 59 of the February 1989 issue of Quality. The program generates a table of sample size and confidence interval for a given confidence level. A listing of the adaptation of the program for the TI-74 appears as lines 200-295 of the "menu and module" program. The print-out at the right is for a 95% confidence level.

Editor's Note: This program is of particular interest for the demonstration of two BASIC language features. The first feature is the changing of the increment in a FOR-NEXT loop while the program is in operation. You might think that one could simply do this by writing a statement FOR N = 3 TO 100 STEP J with J set at one at the start and changed to another value at an selected point in the loop. Unfortunately, the typical BASIC implementation does not respond to that sequence. One method of accomplishing the same thing is illustrated in line 260 of this program which advances the value of N an additional four digits each cycle when N is greater than 15.

95 percent confidence	
Sample	Conf
3	.007
4	.014
5	.022
6	.032
7	.044
8	.057
9	.071
10	.086
11	.101
12	.118
13	.135
14	.152
15	.17
20	.264
25	.357
30	.446
35	.527
40	.6
45	.665
50	.72
55	.766
60	.808
65	.842
70	.87
75	.894
80	.913
85	.93
90	.943
95	.954
100	.962

The second feature is unique to the BASIC implemented in the TI-74 and the CC-40. Don had observed that the execution time seemed to increase as the program proceeded through the table. The effect can be readily observed during a the printout of the table, where the execution time for the rows with a sample size of 60 or greater is substantially longer than for rows with the sample size less than 60. At first I thought that a some sort of sneak loop had been mechanized which caused a time delay; however, once I had convinced myself that was not the case I proceeded to identify an interesting idiosyncrasy of the TI-74 and CC-40 BASIC implementation--the method of calculating a number raised to an integer power is different for integers of 59 and above than for integers of 58 and below! A similar characteristic is evident with the TI-95. Details appear on the next page.

 TI PPC NOTES

V13N2P10

ENTRY OF NULL STRINGS - V12N3P4-6 reviewed three books co-authored by Maurice Swinnen which provide a wide range of BASIC programs which are easily converted for the TI-74. The programs which require multiple input accept the input as string values and convert the string values to numerics internally using VAL commands. This permits termination of the input by entering the letter "E". In V13N1P28 the editor reported that the input of a null string to signify the end of input data was more convenient than the input of "E" as in the Swinnen programs. Maurice writes that he considered that method for his books, but the version of BASIC used in the Sharp machines will not accept a null string input.

AN ANOMALY IN EXPONENTIATION WITH THE TI-74 AND CC-40 - Palmer Hanson. I suspected that the change in execution time of the confidence limits program at higher values of sample size might be associated with the exponentiation function. I wrote the short program at the upper right to determine the execution time for 100 calculations of an argument raised to different powers. I ran the program with arguments of 0.95 and 2 with the TI-74, CC-40 and Radio Shack Model 100. I also ran a similar program with the TI-95. The execution times in seconds were:

N	TI-74		CC-40		Model 100		TI-95	
	2	.95	2	.95	2	.95	2	.95
2.5	30	34	48	48	33	34	34	38
10	3	4	3	5	2	4	5	7
30	5	11	7	16	3	6	8	14
50	10	18	13	26	5	7	12	21
58	11	21	16	30	5	8	13	23
59	31	36	48	48	5	9	14	24
70	30	36	47	49	5	9	16	28
100	30	36	46	46	6	9	34	38
200	30	35	45	47	7	10	35	39
300	31	36	45	47	0V	11	37	39

```

1 INPUT "N=" :N
2 FOR I=1 TO 100
3 P=.95^N
4 NEXT I
5 GOTO 1
6 END

```

```

1 INPUT "N=" :N
2 P=1
3 FOR I=1 TO N
4 P=.95*P
5 NEXT I
6 PRINT P
7 PAUSE
8 GOTO 1

```

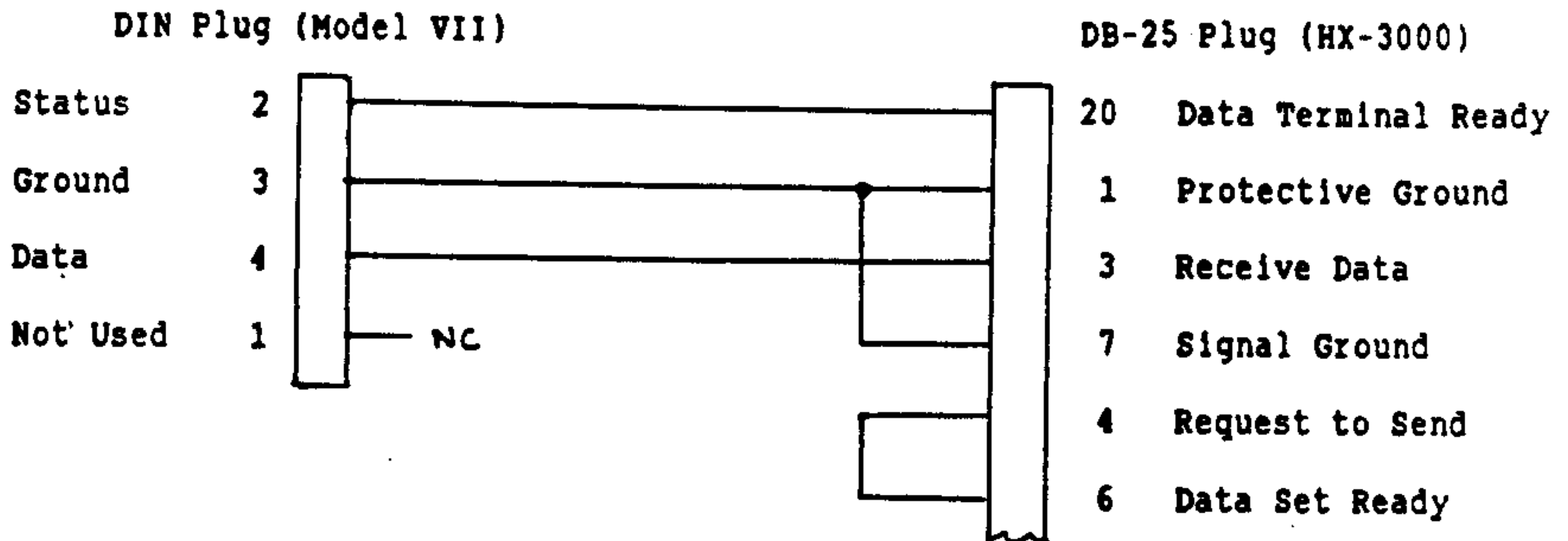
For integer exponents the execution time on the TI-74 and CC-40 increases more or less (within my timing errors) linearly over the range from 10 to 58. The execution time for non-integer exponents (2.5) and for integer exponents above 59 is a constant. By contrast, the execution time for the Model 100 and integer exponents increases linearly over the range shown (and continued over extended ranges as well), but showed a much longer execution time for non-integer exponents. The execution time for the TI-95 increases linearly up to an exponent of 100.

I then compared the values obtained with three methods of exponentiation, 0.95^N , $\text{EXP}(\text{LN}(0.95)*N)$, and the short program at the lower right which would find the product of N values of 0.95. I found that for integers of N of 58 and below the value of 0.95^N agreed with the product of N values of 0.95, but for integers of 59 and above the values of 0.95^N agreed with $\text{EXP}(\text{LN}(0.95)*N)$. These results support the idea that for integer values of N the calculation of A^N is calculated differently depending on the value of N. The product method of evaluating A^N where N is an integer was apparently used for small N to improve execution time. But, why does the change in method occur between N of 58 and 59? The table above shows that execution time is longer for an argument of 0.95 than for an argument of 2. With an argument of .987654321 the execution time with N of 58 or 59 are the same, 39 seconds with the TI-74 and 53 seconds with the CC-40. Thus, the change in calculation method corresponds with the point at which the $\text{EXP}(\text{LN}(A)*N)$ solution becomes the faster method.

The Model 100 calculations do not follow the same pattern. The calculation of A^N by the three methods typically yields small differences between the three methods. For the TI-95 the situation is even more complex with agreement between various methods of calculation seeming to depend on the range of values used for the argument. I will try to cover that in more detail in the next issue.

In the process of this investigation I also found that for negative arguments and integer exponents the TI-74, the CC-40, and the TI-95 all calculate A^N properly. VBN3P16 reported that Charlie Williamson had noted that the HP calculators properly evaluated a negative number raised to an integer power. V11N4P19 discussed the different responses of the TI-59 and TI-95 for powers and roots with negative arguments, but only hinted at the capability to raise a negative number to an integer power. Somehow, I failed to make that observation in over five years of experience with the CC-40.

MORE ON INTERFACING WITH PERIPHERALS - P. Hanson. Scott Garver also asked if anyone had successfully used the CC-40 hex-bus RS-232 interface with the TI-74 to obtain an 80 column printout capability. I replied that Maurice Swinnen had told me that he had done so. To verify the capability I used the interconnecting cable that Maurice defined in V12N3P23 to connect my TI-74 to the HX-3000. I used a serial cable based on information in the Jan/Feb/Mar 1984 issue of the TISOFT newsletter to connect the HX-3000 to an old Radio Shack Model VII printer. The serial cable was



where V10N1P8 noted that the connection between pins 4 and 6 was not in the TISOFT documentation, but was found by trial and error to be needed to enable printing with the CC-40, the HX-3000 and the Model VII. V10N1P8 also reported that I had to include a programmed delay loop after every print statement to provide a proper printout. I discussed the problem at that time at some length with Maurice. He could only speculate that my HX-3000 had an internal problem. I did not return it for repair. With the TI-74, the HX-3000 and the Model VII I obtained the same printout of a 68 column loan payment schedule that I had used to demonstrate the CC-40 and the HX-3000, but again, only by including the programmed delay loops.

While I was examining CC-40 and TI-74 interface compatibility I also connected the CC-40 to the PC-324 using gender changing pins at the TI-74 end of the interface cable defined by Maurice Swinnen in V12N3P23. The PC-324 operated satisfactorily with the CC-40. I jumped to the conclusion that I could use the modified cable with the CI-7 Cassette Interface to provide a long-desired recorder capability for my CC-40. I couldn't make it work, and I don't know why.

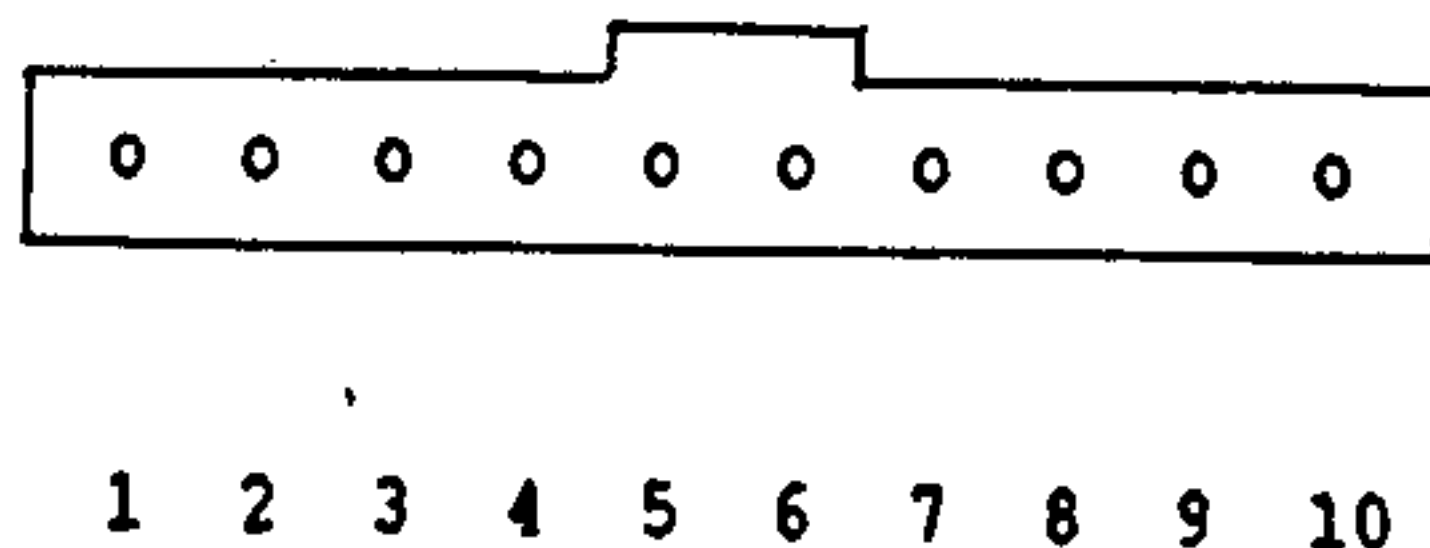
In V12N3P23 I reported that I had used Maurice's cable from V12N3P23 to connect the TI-95 to the HX-1000 Printer/Plotter. I successfully demonstrated printing in both the normal and compressed mode. It appeared to set graphics mode (software control code 019) but I could not successfully send the graphics mode commands. I could send software control code 017 and return to the Text mode.

All of this raises some questions on the various uses of peripherals:

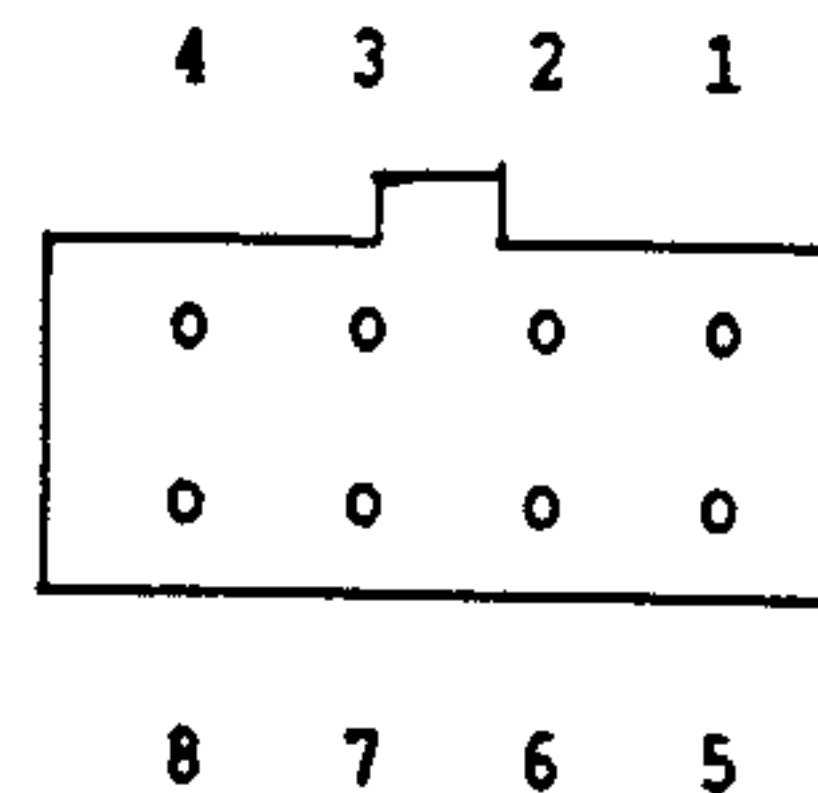
1. Page 34 of the RS232 Users Manual describes a parallel port cabling option for the HX-3000. My HX-3000 does not have that option installed. Has anyone used an HX-3000 with that option?
 2. When I opened my HX-3000 I noted an "edge connector" on the board which looked like it would accept an external connector. Has anyone modified his HX-3000 on his own to implement the parallel port option?
 3. Has anyone successfully used a TI-95 to plot on the HX-1000?
 4. Has anyone implemented a magnetic recorder capability with the CC-40?
-

INTERFACE INFORMATION FOR THE TI-74 AND CC-40 - B. V. Tackach of Wahrenonga, New South Wales, Australia. The enlightening discussion of cabling to connect the TI-74 with the CC-40 peripherals in V12N3P13 was a little confusing. The numbering and lettering of the TI-74 and CC-40 hex bus ports was not consistent with the pin numbering used by TI in its various applications notes and user manuals. In addition some additional information is needed for successful use of the CC-40 peripherals with the TI-74, or with the CC-40 for that matter.

Relative Pin Position Diagrams



TI-74 DOCK-BUS Interface



CC-40 HEX-BUS Interface

DOCK-BUS/HEX-BUS Interface Cable Connections

TI-74 DOCK-BUS			CC-40 HEX-BUS	
Description	Signal	Pin	Signal	Pin
System power distribution - output	PO	1*		
System power distribution - input	PI	2*		
Data bit - least significant bit	D0	3	D0 Data-LSB	1
Data bit	D1	4	D1 Data	2
Data bit	D2	5	D2 Data	7
Data bit - most significant bit	D3	6	D3 Data-MSB	8
Handshake - I/O timing control line	HSK	7	HSK	5
Bus Available - I/O traffic control line	BAV	8	BAV	3
System reset line	RESET	9*		
Common ground line	GND	10	GND	4
			Protect Gnd	6*

Notes:

1. Pin numbers marked with an asterisk are not connected when DOCK-BUS and HEX-BUS are interfaced.
2. CAUTION! Pin #1 of the DOCK-BUS is at +6 volts, supplied by the TI-74. Pin #2 is used to supply all DOCK-BUS peripherals from a common power supply. This is not supported by the CC-40 system.
3. The Protective Ground (pin #6 of the CC-40 HEX-BUS should not be joined to the floating reference power line (pin #10 of the TI-74 DOCK-BUS).

Interface Information for the TI-74 and CC-40 - (cont)Device Numbers for CC-40 Peripherals.

Model	Device	Remarks	Device Numbers
HX-1000	Printer Plotter	Switch Selectable	10 to 11
HX-1010	Printer 80	Internally changed	16 or 17
HX-2000	Wafertape	Was not released	1 to 7
HX-3000	RS232 & Parallel	RS232 - Internally selectable Parallel - Internally selectable	20 to 23
QD-01	Disk Drive	Parallel - Internally selectable by Mechatronic	50 to 53 8

Note: If the TI-99/4 Impact Printer Model PHP2500 is used on the parallel port of the HX-3000, the RS-232 board in the printer must be removed. Refer to Appendix D of the printer manual.

Device Numbers for TI-74 Peripherals

Model	Device	Remarks	Device Numbers
PC-324	Printer	Reserved for RESET-ALL	0
QD-02	Disk Drive	by Mechatronic	12 8

Editor's Note: The TI-74 DOCK-BUS pin assignments are consistent with page 17 of my TI-74 BASICALC Technical Data Manual. The CC-40 HEX-BUS pin assignments are consistent with those in my HX-3000 RS232 Users Manual. Some of the confusion resulted from Maurice Swinnen's definition of the hex-bus interface by "looking into the cable" not by "looking into the device". Other information which is helpful in the construction of a cable between the the TI-74 and a CC-40 peripheral includes the color coding of the wiring in a Hex-bus cable:

Pin	Color	Pin	Color
1	grey	5	brown
2	yellow	6	green
3	red	7	black
4	orange	8	blue

I still have four of the sixteen inch cables with a hex-bus connector at one end. Send one dollar if you want one.

Finally, I remind our members that the connection of the TI-74 to CC-40 peripherals has not been approved by TI, and may result in loss of warranty.

EXTENDED PRECISION PROGRAMS FOR THE TI-59 - V11N2P25 announced that Robert Prins had compiled a 98 page treatment on extended precision programs for the TI-58 and TI-59. Robert reports that he has three copies remaining. He will sell each copy for ten dollars including shipping. He suggests that you write first to confirm that copies are available. His address is in the item at the bottom of page 17. If you are into extended precision calculations for the TI-59 you should get a copy of this.

THE USE OF DATA STATEMENTS TO STORE INPUT DATA

Carl Rabe included a routine for reading input data from DATA statements in his latest smallest circle program. This technique is valuable if user is planning to process the input data many times, or as a means to provide a "canned" test problem with a program. The first program at the right contains essentially the same routine. The routine senses the end of data by recognizing the 999's in line 520.

Another way of accomplishing the same thing is with an error processing routine such as that used to provide a test case in the CC-40 sorting program on V10N3P14. The second program at the right is similar to the routine in the sorting program. The program uses error code 32 (DATA Error) to sense the end of DATA statements. It was during the of conversion of the routine from the CC-40 to the TI-74 that I discovered the differences between error codes described in more detail on pages 16 and 17. The idea for this method originally came from an article in Codeworks.

While working on the review of the error codes I recognized a fundamental deficiency in this method. It assumes that error code 32 is the result of the "Out of data" condition, but page A-18 of the TI-74 Programming Reference Guide shows that other problems in DATA statements can generate a code 32. If you change the DATA statement in line 510 by inserting an alphabetic character after the first 5, say such that the statement reads "DATA 5A,8.7,99.6, ..." then code 32 will be generated, the input from DATA statements will terminate, and the routine will indicate that only six of the eleven data pairs will be read. Thus, the termination of input from DATA statements by use of the error processing routine can result in the omission of desired data if there are faults in the format of the data. This would seem to allow the use of the method for "canned data" for test problems; but, since there are better methods for reading input data from DATA statements there doesn't seem to be any good reason to use the error processing technique.

A third way of terminating the input from DATA statements is to read the data as string expressions, convert to numeric variables using the VAL function, and sense the end of the data with null strings in the DATA statements. A similar technique was used to display the directory for the "Menu and Module" program on V13N2P21. The third program at the right presents a conversion of that routine for the input of numerics. The advantage over Carl's method (the first program at the right) is that the user does not have to check the input for values which are equal to the termination value; for example, in Carl's routine a X value of 999 would cause the input to omit any subsequent data.

The bottom routine at the right adds the number of data pairs as the first value in the DATA statements (see the 11 as the first value in line 500). The single READ statement at line 105 finds that value, and assigns it to the variable N used to terminate the input of data via the following FOR-NEXT loop. The user instructions must emphasize that the first item in the data list is the number of input pairs.

```

100 DIM X(50),Y(50)
105 FOR I=1 TO 50
110 READ X(I),Y(I)
115 IF X(I)=999 THEN 200
120 N=N+1
125 NEXT I
200 PRINT "Number of Dat
& Pairs is:";N;PAUSE 2
205 END
500 DATA 9.8,3.9,6.1,-9.
8,11.3,2.1
505 DATA 8.4,-8.3,-9.8,7
.5,5.5,13.9
510 DATA 5.8,7.99,6.8,1.
16.5,1.1
515 DATA 11.8,12.5,-9.8,
-7.2
520 DATA 999,999

```

```

100 DIM X(50),Y(50)
105 ON ERROR 150
110 READ X(N+1),Y(N+1)
115 N=N+1
120 GOTO 110
150 CALL ERR(E,F)
155 IF E=32 THEN 200
160 RETURN 900
200 PRINT "Number of Dat
& Pairs is:";N;PAUSE 2
205 END
500 DATA 9.8,3.9,6.1,-9.
8,11.3,2.1
505 DATA 8.4,-8.3,-9.8,7
.5,5.5,13.9
510 DATA 5.8,7.99,6.8,1.
16.5,1.1
515 DATA 11.8,12.5,-9.8,
-7.2
900 PRINT "Error Code =
";E;PAUSE
910 END

```

```

100 DIM X(50),Y(50)
105 FOR I=1 TO 50
110 READ X$,Y$
115 IF X$="" OR Y$="" THEN
200
120 X(I)=VAL(X$);Y(I)=VA
L(Y$);N=N+1
125 NEXT I
200 PRINT "Number of Dat
& Pairs is:";N;PAUSE 2
205 END
500 DATA 9.8,3.9,6.1,-9.
8,11.3,2.1
505 DATA 8.4,-8.3,-9.8,7
.5,5.5,13.9
510 DATA 5.8,7.99,6.8,1.
16.5,1.1
515 DATA 11.8,12.5,-9.8,
-7.2
520 DATA "", ""

```

```

100 DIM X(50),Y(50)
105 READ N
110 FOR I=1 TO N
115 READ X(I),Y(I)
120 NEXT I
200 PRINT "Number of Dat
& Pairs is:";N;PAUSE 2
205 END
500 DATA 11,9.8,3.9,6.1,
-9.8,11.3,2.1
505 DATA 8.4,-8.3,-9.8,7
.5,5.5,13.9
510 DATA 5.8,7.99,6.8,1.
16.5,1.1
515 DATA 11.8,12.5,-9.8,
-7.2

```

STORAGE OF DATA IN THE TI-74 - Jim Nugent noted that the TI-74 implementation of the RAM cartridge leaves much to be desired. It would have been nice to have multiple program files and data files in RAM, similar to that in the TI-95 or in the Radio Shack Model 100. That would have allowed a user to go out into the field and collect data with only the hand-held computer, and then come in to the office and print it out, save it on tape or transfer it to a PC.

Editor's Note: Other users have echoed Jim's complaint about the inability to store data files in the TI-74's memory or in an installed RAM. That issue also affects the projected use of a TI-74 with a PC using the interface cable.

One solution is to "bite the bullet" and violate TI's instructions for use of the PC-324 Printer or the CI-7 Cassette Interface with the TI-74. You simply leave the calculator running in a BASIC program at the end of the field site note taking. You have to be sure to leave the computer in a mode where the automatic power down can not come in, or you can set up the block of the automatic power down function as described on page 1-7 of the TI-74 User's Guide. Then when you get to where a peripheral is located, you simply hook up the printer and continue on. You can do this by leaving the TI-74 in the BREAK mode (remember not to clear the display after the break or you will be subject to automatic shutdown), and then doing a Continue (CON) to proceed after you have made the connection. You can open the port to the printer after you have made the connection with an entry from the keyboard. Of course, you would want to make a substantial number of "dry runs" with this technique before you tried to use it in a case where it was critical to retain the data. You also have to recognize that you could be susceptible to loss of warranty if you use this procedure since instructions on pages 8 and 9 of the PC-324 User's Guide and on the instruction card for the CI-7 are very specific that the first step in connecting with the peripheral is to ensure that the calculator is off.

One way to store data which will survive exit from BASIC or shutdown is to put the data in DATA statements. The consideration of that option led to the examination of the various methods of retrieving data from DATA statements on page 18 of this issue.

If you have a limited number of data items to store through a shutdown you can use the "Assigning Text to a Key" idea from pages 3-22 and 3-23 of the TI-74 User's Guide. What you can do is bring the information to the display followed by a command which permits data entry such as an INPUT statement. Then make keyboard entries to match the displayed data and press SHIFT FN and one of the ten number keys. After turning off and back on again, and again setting up with an INPUT statement, you can retrieve the information stored in the SHIFT FN definitions by pressing FN and the appropriate number key. A bit cumbersome, perhaps, but it works.

Finally, there may be hope for a better solution. A contact at TI tells me that a program has been written which permits the establishment of data files in the TI-74 memory. So far, I haven't been able to obtain the program.

THE PC TO TI-74 INTERFACE CABLE ALSO WORKS WITH THE TI-95 - V13N3P10 reported the coming availability of a cable which will provide the interface between a TI-74 and a TI or IBM compatible PC. I received an engineering model from TI and immediately found that the cable will also provide the interface between a TI-95 and a PC. Scott Garver currently has the cable. He promises to provide an evaluation for the first issue in the coming year.

DOUBLE EXPONENTIAL EVALUATION OF DATA ON THE TI-74 - P. Hanson. V12N3P19 showed the ease with which the program for linear regression with user defined functions from V12N1P14 could be modified to solve linear regression problems with two independent variables W and X. Some additional changes to the program are needed to permit the solution of double exponential problems. In addition, I changed the independent variables from W and X to X and Z for consistency with Marcel's TI-59 program on pages 15-17. The resulting changes relative to the listing on V12N1P14 are:

1. Add Z(50) in line 100.
2. Change "Pairs" to "Points" in the prompt in line 130.
3. Change the comma between A\$ and X(I) to a semicolon in line 160, and change the comma between A\$ and Y(I) to a semicolon in line 180.

4. Add lines 181 and 183:

```
181 A$ = "Z"&STR$(I)&" = ":input A$;Z(I)
```

```
183 IF PN<>0 THEN PRINT #PN,A$;Z(I)
```

5. Change lines 310 and 560 to permit modification of the dependent variable in the user defined functions:

```
310 B(I)=B(I)+F(I)*F(0):NEXT I
```

```
560 D = F(0)-YF
```

A complete set of prompts are provided. Steps 800-860 provide the user defined functions for the double exponential evaluation. To obtain a standard linear regression with two variables such as in V12N3P19 the user defined functions should be:

```
810 F(0) = Y(L)
820 F(1) = 1
820 F(2) = X(L)
830 F(3) = Z(L)
860 RETURN
```

A printout from the program for the same problem as on page 16 appears at the right. Note that in this mechanization that A1 = ln(a), A2 = b and A3 = c.

The program listing appears on page 19.

X1 =	21
Y1 =	.201064
Z1 =	2
X2 =	40
Y2 =	1.43996
Z2 =	7
X3 =	100
Y3 =	4.34868
Z3 =	5
X4 =	500
Y4 =	32.3097
Z4 =	3
X5 =	50
Y5 =	2.67693
Z5 =	10
A1 =	-6.725432649
A2 =	1.499999478
A3 =	.8000012245
d1 =	-2.4272E-08
d2 =	2.278683E-07
d3 =	-1.74774E-07
d4 =	6.555E-08
d5 =	-9.438459E-08
Mean =	-2.458E-12
S.E. =	2.193904E-07

MORE ON THE LINEAR/HYPERBOLIC PROGRAM INADEQUACY IN THE KOLB BOOK

V13N2P4 reported on a long standing inadequacy with the set of TI-59 programs in the second edition of William Kolb's Curve Fitting for Programmable Calculators. Marcel Bogart reports that the third edition of the book, available from EduCALC as stock number M-135 for \$13.95, does NOT have the suggested corrections in place.

HP-35 WANTED - William West, 731 Monroe Street, Apt. 303, Rockville MD 20850 .

Double Exponential Evaluation of Data on the TI-74 - (cont)Program Listing

```

10 REM Multiple Linear R      230 B(I)=0:F(I)=0:NEXT I      560 D=F(0)-YF
expression
20 REM with User Defined     240 FOR L=1 TO K                570 IF AS="N"OR AS="n"TH
Functions
30 REM 17 July 1989          250 GOSUB 800                    EN 610
100 DIM A(8,8),B(8),F(8)     300 FOR I=1 TO N:FOR J=1      580 PS="d"&STR$(L)&" = "
,X(50),Y(50),Z(50)          TO N
105 INPUT "Use Printer <    305 A(I,J)=A(I,J)+F(I)*F    590 PRINT #PN,PS;D
Y/N)? ":AS                   (J):NEXT J
110 IF AS="Y"OR AS="y"TH     310 B(I)=B(I)+F(I)*F(0):      600 IF PN=0 THEN PAUSE
EN PN=1 ELSE 125
115 INPUT "Device Code ?    315 NEXT I
":PS
120 OPEN #1,PS,OUTPUT        320 FOR L=1 TO N
125 PRINT "Are the funct     325 P=A(L,L)
ions correct?":PAUSE 2
130 INPUT "Number of Dat    330 FOR J=L TO N
a Points? ":K
140 FOR I=1 TO K
150 AS="X"&STR$(I)&" = "
:INPUT AS:X(I)
160 IF PN<>0 THEN PRINT
#PN,AS;X(I)
170 AS="Y"&STR$(I)&" = "
:INPUT AS:Y(I)
180 IF PN<>0 THEN PRINT
#PN,AS;Y(I)
181 AS="Z"&STR$(I)&" = "
:INPUT AS:Z(I)
183 IF PN<>0 THEN PRINT
#PN,AS;Z(I)
185 PRINT #PN:IF ES<>"T
HEN 700
190 NEXT I
200 INPUT "Order of the
solution? ":N
210 FOR I=1 TO N:FOR J=1
TO N
220 A(I,J)=0:NEXT J
230 B(I)=0:F(I)=0:NEXT I
240 FOR L=1 TO K
250 GOSUB 800
300 FOR I=1 TO N:FOR J=1
TO N
305 A(I,J)=A(I,J)+F(I)*F
(J):NEXT J
310 B(I)=B(I)+F(I)*F(0):
NEXT I
315 NEXT L
320 FOR L=1 TO N
325 P=A(L,L)
330 FOR J=L TO N
335 A(L,J)=A(L,J)/P:NEXT
J
340 B(L)=B(L)/P
345 FOR I=1 TO N
350 IF I=L THEN 375
355 G=A(I,L)
360 FOR J=L TO N
365 A(I,J)=A(I,J)-G*A(L,
J):NEXT J
370 B(I)=B(I)-G*B(L)
375 NEXT I
380 NEXT L
400 FOR I=1 TO N
410 XS="A"&STR$(I)&" = "
420 PRINT #PN,XS;B(I)
430 IF PN=0 THEN PAUSE
440 NEXT I
450 PRINT #PN
500 INPUT "Display Resid
uals <Y/N)? ":AS
510 S1=0:S2=0
520 FOR L=1 TO K
530 GOSUB 800
540 YF=0:FOR J=1 TO N
550 YF=YF+B(J)*F(J):NEXT
J
560 D=F(0)-YF
570 IF AS="N"OR AS="n"TH
EN 610
580 PS="d"&STR$(L)&" = "
590 PRINT #PN,PS;D
600 IF PN=0 THEN PAUSE
610 S1=S1+D:S2=S2+D*D:NE
XT L
620 PRINT #PN
630 PRINT #PN,"Mean = ":
S1/K
640 IF PN=0 THEN PAUSE
650 PRINT #PN
660 PRINT #PN,"S.E. = ":
SQR(S2/(K-N))
670 IF PN=0 THEN PAUSE
680 PRINT #PN
700 INPUT "Edit Input Da
ta <Y/N)? ":ES
710 IF ES="N"OR ES="n"TH
EN 780
720 INPUT "Which Data Pa
ir to Edit? ":I
730 IF I<1 OR I>K THEN 7
00
740 GOTO 150
780 INPUT "New Solution
<Y/N)? ":AS
790 IF AS="Y"OR AS="y"TH
EN 200
799 STOP
800 REM USER DEFINED FUN
CTIONS
810 F(0)=LN(Y(L))
820 F(1)=1
830 F(2)=LN(X(L))
840 F(3)=LN(Z(L))
860 RETURN

```

ERRATA

Double Exponential Evaluation of Data on the TI-59 - V13N3P15. Some readers have pointed out that that previous issues of our newsletter have typically defined equations of the form $y = Ae^{Bx}$ as exponential curves and equations of the form $y = Ax^B$ as power curves. This is consistent with references such as the documentation for the Real Estate/Investment and Statistics modules for the TI-59, the TI-95 Statistics Library, the fx-7000G Owner's Manual, and Kolb's Curve Fitting for Programmable Calculators. (Some references such as Spiegel's Statistics call the $y = Ax^B$ equation a geometric curve.) If we had followed the typical usage then the solution in V13N3P15-17 should have been called a double power evaluation. Hopefully, anyone who read the documentation accompanying the program understood what we meant.

REGRESSION WITH 2 OR 3 VARIABLES AND USER DEFINED FUNCTIONS - P. Hanson

This program is another iteration in the development of a TI-74 regression program which provides versatility through the use of user defined functions for the regression. Earlier versions in the development appeared in V11N4P12-15, V12N1P14, V12N3P19 and V13N3P18/19. The impetus for this iteration was support of the discussion of curve fitting with George Thomson and Marcel Bogart as documented on page 25 of this issue. Changes relative to the program in V13N3P18/19 provide:

- * User selection of two or three variables (one or two independent variables).
- * The capability to run the solution unweighted, or to apply a user defined weighting as part of the user defined functions.
- * The capability to use one function for the solution and another for the calculation of residuals.

Program Listing:

<pre> 10 REM Linear Regression with 2 or 3 Variables 20 REM and User Defined Functions 30 REM 4 September 1989 100 DIM A(8,8),B(8),F(8) ,X(50),Y(50),Z(50) 105 INPUT "Use Printer < Y/N>? ";AS 110 IF AS="Y"DR AS="y"TH EN PN=1 ELSE 123 115 INPUT "Device Code ? ";PS 120 OPEN #1,PS,OUTPUT 123 INPUT "Are functions correct <Y/N>? ";AS 127 IF AS="N"DR AS="n"TH EN STDP 130 INPUT "Number of Dat a Points? ";K 133 INPUT "2 or 3 Variab les? ";KK 137 IF KK=2 DR KK=3 THEN 140 ELSE 133 140 FOR I=1 TO K 150 AS="X"&STR\$(I)&" = " :INPUT AS;X(I) 160 IF PN<>0 THEN PRINT @PN,AS;X(I) 170 AS="Y"&STR\$(I)&" = " :INPUT AS;Y(I) 175 IF PN<>0 THEN PRINT @PN,AS;Y(I) 180 IF KK=2 THEN 185 181 AS="Z"&STR\$(I)&" = " :INPUT AS;Z(I) 183 IF PN<>0 THEN PRINT @PN,AS;Z(I) 185 PRINT @PN:IF ES<>"T HEN 700 190 NEXT I 200 INPUT "Order of the solution? ";N </pre>	<pre> 205 M=1 210 FOR I=1 TO N:FOR J=1 TO N 220 A(I,J)=0:NEXT J 230 B(I)=0:F(I)=0:NEXT I 240 FOR L=1 TO K 250 GOSUB 800 300 FOR I=1 TO N:FOR J=1 TO N 305 A(I,J)=A(I,J)+F(I)*F (J)*M:NEXT J 310 B(I)=B(I)+F(I)*F(0)* M:NEXT I 315 NEXT L 320 FOR L=1 TO N 325 P=A(L,L) 330 FOR J=L TO N 335 A(L,J)=A(L,J)/P:NEXT J 340 B(L)=B(L)/P 345 FOR I=1 TO N 350 IF I=L THEN 375 355 G=A(I,L) 360 FOR J=L TO N 365 A(I,J)=A(I,J)-G*A(L, J):NEXT J 370 B(I)=B(I)-G*B(L) 375 NEXT I 380 NEXT L 400 FOR I=1 TO N 410 XS="A"&STR\$(I)&" = " 420 PRINT @PN,XS;B(I) 430 IF PN=0 THEN PAUSE 440 NEXT I 450 PRINT @PN 500 INPUT "Display Resid uals <Y/N>? ";AS 510 S1=0:S2=0 520 FOR L=1 TO K 530 GOSUB 900 570 IF AS="N"DR AS="n"TH EN 610 </pre>	<pre> 580 PS="d"&STR\$(L)&" = " 590 PRINT @PN,PS;D 600 IF PN=0 THEN PAUSE 610 S1=S1+D:S2=S2+D*D:NE XT L 620 PRINT @PN 630 PRINT @PN,"Mean = "; S1/K 640 IF PN=0 THEN PAUSE 650 PRINT @PN 660 PRINT @PN,"S.E. = "; SOR(S2/(K-N)) 670 IF PN=0 THEN PAUSE 680 PRINT @PN 700 INPUT "Edit Input Da ta <Y/N>? ";ES 710 IF ES="N"DR ES="n"TH EN 780 720 INPUT "Which Data Pa ir to Edit? ";I 730 IF I<1 DR I>K THEN 7 00 740 GOTO 150 780 INPUT "New Solution <Y/N>? ";AS 790 IF AS="Y"DR AS="y"TH EN 200 799 STDP 800 REM USER DEFINED FUN CTIONS 810 F(0)=LN(Y(L)) 820 F(1)=1 830 F(2)=LN(X(L)) 840 F(3)=LN(Z(L)) 850 M=Y(L)*Y(L) 860 RETURN 900 REM RESIDUAL CALCULA TIONS 910 YF=EXP(B(1))*X(L)^B(2)*Z(L)^B(3) 920 D=Y(L)-YF 930 RETURN </pre>
---	--	--

Regression with 2 or 3 Variables and User Defined Functions - (cont)

The changes in the program relative to the listing on V13N3P19 are:

The Else address in line 110 is changed to agree with other changes in the program.

Line 125 is replaced by lines 123 and 127 to provide for an operator input to indicate that the functions are correct.

Lines 133, 137 and 180 are added to provide for the input of 2 or 3 variables.

Line 205 is added to provide a default value for uniform weighting.

Lines 305 and 310 are changed to include the weighting factor W.

Lines 530 to 560 are replaced by a new line 530 to provide for use of a different function for the calculation of residuals.

Lines 900-930 are added to provide the functions for the calculation of residuals.

Sample User Defined Functions and Residual Calculations

Lines 800 to 860 and 900 to 930 provide for a solution for a double power function evaluation similar to that in the program on V13N3P18/19. The name was changed from "double exponential" to "double power" for consistency with other solutions in TI PPC Notes. The difference in calculations is that the residuals are in the coordinate system before the logarithmic transformation.

Two additional listings for lines 800 ff appear below. The left hand listing can be used for a single exponential solution with the residuals in the coordinate system before transformation. The right hand listing can be used for solution of polynomials with one independent variable in a manner similar to that in the program in V11N4P12-15.

```

800 REM USER DEFINED FUN
CTIONS
810 F(0)=LN(Y(L))
820 F(1)=1
830 F(2)=X(L)
840 W=Y(L)*Y(L)
850 RETURN
900 REM RESIDUAL CALCULA
TIONS
910 YF=EXP(B(1))*EXP(B(2)
)*X(L))
920 D=Y(L)-YF
930 RETURN

```

```

800 REM USER DEFINED FUN
CTIONS
810 F(0)=Y(L)
820 F(1)=1
830 FOR NN=2 TO N
840 F(NN)=F(NN-1)*X(L)
850 NEXT NN
860 RETURN
900 REM RESIDUAL CALCULA
TIONS
910 GDSUB 800
920 YF=0:FOR NN=1 TO N
930 YF=YF+B(NN)*F(NN):NE
XT NN
940 D=Y(L)-YF
950 RETURN

```

The important things to remember when generating other user defined functions and residual calculations are that F(0) in the user defined function section is the independent variable (with or without transformation), W in the user defined function section is the weighting to be used (a default of W = 1 is provided elsewhere), and the equations for the residual calculations which follow line 900 must find the variable D as the difference between the dependent variable (or a transformation thereof) and the calculated value.

Regression with 2 or 3 Variables and User Defined Functions - (cont)

The pair of printouts at the left below are the solutions with and without y-squared weighting for the same double power function problem as on V13N3P18. For the unweighted solution the coefficients are the same as in V13N3P18, but the residuals are different since they are calculated in the coordinate system before transformation. Thus, the mean of the residuals and the standard error are larger. For the weighted solution, both the mean error and the standard error are reduced relative to the unweighted solution.

The pair of printouts at the right below are the solutions with and without y-squared weighting for an exponential fit to the same data used for the demonstration of the five function curve fit in V12N4P24/25. In this case the first coefficient of the solution is actually $\ln(a)$.

W = 1

W = Y²

W = 1

W = Y²

```

X1 = 21
Y1 = .201064
Z1 = 2

X2 = 40
Y2 = 1.43996
Z2 = 7

X3 = 100
Y3 = 4.34868
Z3 = 5

X4 = 500
Y4 = 32.3097
Z4 = 3

X5 = 50
Y5 = 2.67693
Z5 = 10

R1 = -6.725432649
R2 = 1.499999478
R3 = .8000012245

d1 = -4.88015E-09
d2 = 3.28122E-07
d3 = -7.60033E-07
d4 = 2.117956E-06
d5 = -2.52659E-07

Mean = 2.857012E-07
S.E. = 1.617855E-06

```

```

X1 = 21
Y1 = .201064
Z1 = 2

X2 = 40
Y2 = 1.43996
Z2 = 7

X3 = 100
Y3 = 4.34868
Z3 = 5

X4 = 500
Y4 = 32.3097
Z4 = 3

X5 = 50
Y5 = 2.67693
Z5 = 10

R1 = -6.725433786
R2 = 1.499999639
R3 = .8000014045

d1 = 9.969297E-08
d2 = 6.02523E-07
d3 = -3.11868E-07
d4 = 1.5396E-08
d5 = -1.0859E-08

Mean = 7.897699E-08
S.E. = 4.850718E-07

```

```

X1 = 1
Y1 = 3.2

X2 = 2
Y2 = 7.4

X3 = 3
Y3 = 12

X4 = 4
Y4 = 16.8

X5 = 5
Y5 = 22

R1 = .9096871079
R2 = .4675682173

d1 = -.7640067736
d2 = 1.073016751
d3 = 1.901450901
d4 = .6816241667
d5 = -3.726669937

Mean = -.1669169783
S.E. = 2.562763297

```

```

X1 = 1
Y1 = 3.2

X2 = 2
Y2 = 7.4

X3 = 3
Y3 = 12

X4 = 4
Y4 = 16.8

X5 = 5
Y5 = 22

R1 = 1.364904852
R2 = .3513039132

d1 = -2.36339629
d2 = -.5051360299
d3 = .7674412557
d4 = .8394427791
d5 = -.6786605439

Mean = -.3880617659
S.E. = 1.591122751

```

CAN YOU USE A TI-99/4? - A set of used TI-99/4 has been donated to our club.

It does have documentation, a TV interface and a cassette interface but does not have a disc drive. Write for details if you are interested.

TRUNCATION EFFECTS IN EXTENDED PRECISION

In the errata section on page 2 Carl Rabe pointed out that the last lines of digits in several extended precision examples were incorrect. This is a direct result of the truncation that occurs beyond the last digit in the sum carried by the calculations. If all the terms of the summation are of the same sign then one might reasonably expect to find that, on the average, the truncation observed will be one half of the last digit times the number of terms in the sum. One example of such an effect occurred in Patrik Johnasson's program for calculation e to 480 digits which appeared in V9N4P24. The last line of the printout from the program is 6445490479. A 1300 digit calculation of e provided by Robert Frins in V9N5P4 will show that the correct last line for the 480 digit calculation would be 6445490598. Patrik's program required 245 iterations, so the expected truncation would be of the order of 122. If that value is added to 6445490479 the sum is 6445490601 which is within 3 of the correct last line. A nice explanation of the effect, albeit in Swedish, appears in the Programbiten article which first published Patrik's program. Send a SASE and an extra stamp if you would like a copy.

If you use the rule to check the accuracy of the solutions for $\ln(2)$ and $\ln(3)$ in V11N2P24 you will find that the rule doesn't seem to work. The reason is that the \ln program already provided an automatic correction to account for truncation.

A copy of the program from V11N2P24 translated to calculate $\ln(3)$ on the TI-74 appears at the right. Lines 500-510 provide the automatic truncation correction by adding 1 at the least significant digit every other cycle, for an average of 0.5 per cycle. Without this correction the error in the last line would be substantially larger.

```

100 DIM A(120),B(120),C(
120)
110 IMAGE #####
120 CALL UP("Logarithm o
f 3",Z)
130 S=1.E+10
140 INPUT "Number of 10
Digit Blocks ? ";N
150 FOR I=1 TO N
160 A(I)=6666666666
170 NEXT I
300 I=1
400 R=0
410 FOR J=1 TO N
420 B(J)=A(J)+R*S
430 R=B(J)-I*INT(B(J)/I)
440 B(J)=INT(B(J)/I)
450 NEXT J
500 M=(I+1)/4
510 R=2*(M-INT(M))
520 FOR J=N TO 1 STEP -1
530 C(J)=C(J)+B(J)+R
540 R=INT(C(J)/S)
550 C(J)=C(J)-R*S
560 NEXT J
600 R=0
610 FOR J=1 TO N
620 T=INT((A(J)+R*S)/9)
630 R=A(J)+R*S-9*T
640 A(J)=T
650 NEXT J
700 IF B(N)<>0 THEN I=I+
2:GOTO 400
800 PRINT #2,10*N;" Digi
ts"
810 PRINT #2
820 FOR J=1 TO N
830 PRINT #2,USING 110,C
(J)
840 IF Z=0 THEN PAUSE
850 NEXT J
860 PRINT #2
870 PRINT #2,"I = ";I
900 IF Z=1 THEN CLOSE #1
999 STOP

```

USING X*X INSTEAD OF X^2 TO INCREASE SPEED - Carl Rabe was investigating ways to speed up his BASIC program for solving the smallest circle problem from V13N2P12. Replacing all of the X^2 functions with $X*X$ expressions provided a 25 per cent decrease in execution time on his ATARI 1040ST using True BASIC. He asked if we would see a comparable improvement with other machines. I set up routines to time 1000 iterations of either $X*X$ or X^2 on my TI-74, Model 100, and fx-7000G. The times in seconds for the various machines were:

	TI-74	Model 100	fx-7000G
X*X	17	13	16
X^2	19	28	14

The very limited improvement in the speed of the TI-74 with the $X*X$ expression seems consistent with the observations in V13N2P27. It seems likely that the TI-74 automatically implements an exponentiation to the second power as a product.

MORE ERRATA

Regression with User Defined Functions - V12N4P12. The original user defined function regression program in V11N4P12 used the linear equation solution from the Mathematics library of the TI-74. V12N1P14 published a modification which did not depend on the Mathematics module. The original program also printed the input data with the input value on one line and annotation on another line. The modification in V12N1P14 also included an attempt to provide input values on the same line as the annotation; however, the modification incorrectly used commas between the annotation string and the input value. The result is that variables which use more than eight spaces will spill over and be printed on the next line. The reduced printouts which follow illustrate the problem, where the first and third columns were printed with semicolons between the annotation string and the value, and the second and fourth columns with commas.

X1 = 21 Y1 = .201064 Z1 = 2	X1 = 21 Y1 = .201064 Z1 = 2	X1 = 3.141592654 Y1 = 1.23456E+16 Z1 = 2	X1 = 3.141592654 Y1 = 1.23456E+16 Z1 = 2
X2 = 40 Y2 = 1.43996 Z2 = 7	X2 = 40 Y2 = 1.43996 Z2 = 7	X2 = .123456 Y2 = .1234567 Z2 = .12345678	X2 = .123456 Y2 = .1234567 Z2 = .12345678

So, lines 160 and 180 of the program listing on V12N1P14 should be changed accordingly. That is the same change as in item 3 on page 18 of this issue.

STORAGE LIFE FOR THE RAM CARTRIDGES - Scott Garver found that various information on the external 8K RAM suggests a limited storage life. For example, page 2 of the 8K Constant Memory User's Guide states that the device:

"... ensures that all contents of the cartridge memory are maintained--even when the cartridge is removed from the calculator. An internal battery gives the cartridge a typical service life of three years or more."

Similarly, Page 12 of the TI-74 BASICALC Technical Data Manual states:

"... The cartridge houses a single HM6264 8K CMOS RAM and a 3 volt lithium battery. When the cartridge is separated from the console, the battery powers the RAM, retaining data for as long as five years. ..."

Scott called 1-800-TI-CARES and was informed that the battery in the RAM cartridge is disconnected until it is used the first time. After the first useage the life span is expected to be 3 to 5 years. Obvious additional questions are:

Is the life extended when the RAM is in the calculator?

Does the calculator recharge the RAM?

As of May 1 Scott had not received answers to those questions. He opened a RAM and verified that the battery is is a 3 volt lithium cell (C?2430). Lithium cells are not considered to be rechargeable. The remaining issue is whether pin 24 shown in Table 2 on page 11 of the manual, labeled as "+5 Volt for RAM retention" actually provides the hold-up power when the RAM is installed in the TI-74 or TI-95, thus extending the life of the battery. If so, it would seem that a logical accessory would be a storage device which provides the holdup power for spare RAMs.

A 32K RAM FOR THE TI-74 FROM TI - Page 12 of this issue describes a 32K RAM capability for the TI-95 as developed by Scott Garver. A recent brochure from TI on TI-74 applications indicated that a 32K CRAM Module is available for the TI-74. I called TI for information and was told that the module had been delivered for use in some special applications but was not yet available to the public. For most TI-74's in circulation it will be necessary to load a special MEMADD subprogram prior to use of the 32K Module. TI will send an engineering model for evaluation. I will report on it in the first issue of the coming year.

Other literature received from TI indicates that they been successful in marketing sizeable numbers of TI-74's for use special field applications with activities such as the Farmers Home Administration, Cummins Engine Company, Kodak and the Canadian Department of Revenue. Those applications should lend assurance to the continued availability of the TI-74/TI-94 peripherals such as the PC-324 and CI-7.

STORAGE LIFE FOR THE 8K RAM CARTRIDGE - V13N3P22 presented Scott Garver's review of the documentation on the life of the battery in the 8K RAM cartridge, and noted that there were some unanswered questions. We have obtained some answers from TI:

1. When the RAM is installed in TI-74 or TI-95 the power to support the RAM comes from the batteries in the parent machine. Thus, RAM storage life will be extended if the RAM is in a calculator or computer.
2. The battery in the RAM is not recharged by the calculator or computer.

Manufacturers of carbon-zinc and alkaline cells provide a "Best If Installed by ..." note on the package to alert users to shelf life for the cells. There are also well established degradation modes for lead-acid and nickel-cadmium cells. So another question on the RAM battery is:

Doesn't the RAM battery age and loses capability gradually even when the RAM is installed in a calculator or computer?

THE PARALLEL PORT OPTION FOR THE HX-3000 - The discussion of interfacing with peripherals in V13N3P23 asked if anyone had modified an HX-3000 to implement the parallel port option. Jim Nugent of Peoria, Illinois wrote: "I opened up my HX-3000 back in 1983 and installed a short (12 inch) cable that ran out of the case over or under the RS-232 connector. When I wanted to print I took the cable off the back of my IBM PC and connected it to the cable stub hanging out of the back of the HX-3000.

To add a parallel port to an HX-3000 you need a 20 contact ribbon header socket (IDM20 from JDR - see page 327 in the August 1989 issue of BYTE), a foot or two of 20 conductor ribbon cable, and a male D-subminiature 25 pin connector. I hand wired my D-sub connector, so I must have used the circuit diagram in the HX-3000 data sheet that came with the unit. As I remember it, the 20 pins on the HX-3000 were half grounds. All of the pins on the bottom row, I think, were grounds giving you a ribbon cable with every other wire a ground. My cable out the back was crude and not very portable. A better solution would be to bite the bullet and mount the D-sub connector on the side or top of the HX-3000 case."

Jim also addressed the other interface issue raised in V13N3P23. He tried to use the CI-7 with the CC-40 without success (the same experience as the editor). He agreed that the CC-40 had some great features which were not carried forward to the TI-74 such as the CHAR subprogram, the DEBUG Monitor subprogram, and a better RAM file system.

MORE ON SORTING - P. Hanson. V10N3P13-16 compared five sorting algorithms which were mechanized on the CC-40. Execution times were presented for various numbers of random integers. As a result of correspondence with Robert Prins I re-examined the comparison between the two faster general purpose algorithms, a Shell sort algorithm available as a subroutine of the Statistics cartridge and a heap sort algorithm adapted from a program on page 137 of the September 1980 issue of Creative Computing. For various numbers of random integers the execution times in seconds were:

Method	Number of Integers			
	30	100	300	1000
Shell	13	71	390	1875
Heap	11	52	194	797
Ratio	1.18	1.36	2.01	2.35

where the heap sort becomes relatively more time efficient as the number of integers increases. For integers in reverse order the results were:

Method	Number of Integers			
	30	100	300	1000
Shell	9	39	157	703
Heap	10	48	180	747
Ratio	0.90	0.81	0.87	0.94

where the Shell sort has a slight advantage, at least for the range tested. Finally, I tested the performance of the two methods where the numbers were already sorted. For the 300 integer case the Shell sort algorithm declares the list to be sorted in 55 seconds. In contrast, the heap sort algorithm does not declare an already sorted list to be sorted until 209 seconds have passed. That is slightly longer than the time to sort 300 random integers or 300 integers in inverse order! I conclude that the execution time of the heap sort is largely independent of the condition of the unsorted list, a characteristic I had not recognized before.

GAME PROGRAMS FOR THE TI-74 - Steve Greenspon of Montreal, Quebec sent in a tape with several TI-74 game programs. The games include blackjack, crazy eights, hangman, and monopoly, where the monopoly program requires an 8K RAM. The tape also includes a matrix manipulation program which requires the additional RAM. If you would like to obtain these programs send one dollar (no checks, please) to cover postage and packaging. I will send Steve's tape to you. You can copy the program as you see fit and promptly return the tape to me for use by other members.

REPAIR SUPPORT FOR THE CC-40 AND TI-74 - P. Hanson. Earlier this year my CC-40 developed a problem with the display in which many of the dots would not turn on. I called 1-800-TI-CARES to see if CC-40 repair was still available. I was told that defective units were replaced rather than repaired. The replacement cost was \$66.50. I shipped the defective unit on April 17 and had a replacement by April 28. Unfortunately, the repair facility had replaced my 18K CC-40 with an 8K unit. I called the repair facility and had an 18K unit in hand by May 8.

In mid-May my TI-74 developed an intermittent problem during entry of BASIC statements. The entries into the display would be different from the keys which had been pressed. Again, I called 1-800-TI-CARES. The repair/replacement cost was \$63.00. I shipped the defective unit on May 11 and had a replacement in hand on May 25.

EXPANDED ERRATA SHEETS FOR THE CC-40 - I received a new manual with the replacement CC-40 mentioned above. The manual seems to be identical to the manual I received several years ago. The errata sheet is completely different with the exception of the addition to page 1-7 identifying the AC adapter as the AC9201. Both errata sheets are copyrighted in 1983. The one I received originally carries the nomenclature 1055825-1. The one received this spring carries the nomenclature 10055825-4. For the benefit of CC-40 owners who do not have the later errata sheet I have reproduced the three pages below.

ADDENDUM

Texas Instruments Compact Computer 40 User's Guide

Caution: Read the information on static electricity on page 1-4 before working with your computer.

The following notes provide additional information about using and programming your Texas Instruments Compact Computer 40.

Page 1-7

The optional AC adapter referred to is the Texas Instruments model AC9201; use only the AC9201 with your CC-40.

Page 5-34

Add the following sentence to the end of the third paragraph.
I/O error-type 255 is returned as 0.

Page 5-43

The following paragraph provides more information on GETMEM.

The memory reserved by GETMEM can be released during program execution by a call to RELMEM. Any of the following actions cause the reserved memory to be released.

- Editing the program or subprogram.
- Entering a NEW, OLD, RENUMBER, RUN, SAVE, or VERIFY command.
- Listing the program to a peripheral device.
- Calling the ADDMEM or CLEANUP subprogram.
- Turning the system off or pressing the reset key.

Page 5-48

The example at line 290 should be as shown below.

```
290 IF AS = "Y" THEN COUNT = COUNT + 1: DISPLAY
    AT(4), "Enter value: "; GOTO 400
```

Page 5-65

The following paragraph provides more information on status-variable.

When CALL KEY is executed, the keyboard is scanned for input. Status-variable is used to store a value that represents the status of the scan. A value of 0 means no key was pressed. A value of 1 means a different key was pressed since the last time the keyboard was scanned for input (e.g., since CALL KEY, KEYS, INPUT, or ACCEPT was last executed). A value of -1 means the same key was pressed.

(continued)

To avoid this problem, use one of the following methods to be sure that the stack pointer does not point to register 9 when a breakpoint is executed.

- Begin the register file stack at register A₁₆ instead of at register 1.
- If the position of the stack cannot be altered, add PUSH and POP instructions to the code to ensure that the stack does not use register 9 within the section of code being debugged. The added instructions can be removed after the code is debugged.
- Write the assembly language software such that the stack pointer stays on even byte boundaries.

Page I-5

The last paragraph should begin as shown below.

Then B can be typed to continue program execution...

Page K-6

The fifth error message paragraph should be changed as shown below.

- Invalid character in statement. For example "%", "?", "[", "]", "{", etc., are valid only within quoted strings or in an IMAGE or REM statement.

The following program segment provides more information on the use of the CALL KEY subprogram. This segment prompts twice for a key to be pressed. To determine that the responses are distinct, the status variable is compared to 1 (S<>1) in lines 520 and 560.

```
500 PRINT "MORE ENTRIES? (Y OR N)"
510 CALL KEY(K,S)
520 IF S<>1 THEN 510
530 IF K = ASC("Y") OR K = ASC("y") THEN 400
540 PRINT "END SESSION? (Y OR N)"
550 CALL KEY(K,S)
560 IF S<>1 THEN 550
570 IF K = ASC("Y") OR K = ASC("y") THEN STOP
```

Page 5-133

The example for line 330 should be as follows.

```
330 SUB PAYCHECK( DATE, Q, SSN, PAYRATE, TABLE(,))
```

Marks the beginning of a subprogram. The variables DATE, Q, SSN, PAYRATE, and the array TABLE with two dimensions may be used and/or have their values changed in the subprogram and their corresponding arguments in the calling statement changed. However, if the corresponding argument of DATE, Q, SSN, or PAYRATE is enclosed in parentheses in the CALL statement, the value of that argument cannot be changed. The corresponding array argument of TABLE *must* be passed by reference in the CALL statement and therefore any of its values can be changed in the subprogram.

Page I-4 (Appendix)

In the fifth line from the top, the address is the second breakpoint set, as shown below.

where *nnnn* is the address of the second breakpoint set.

Pages I-3,4

The following problem can occur when a breakpoint is set in assembly language software by either the breakpoint command or the single step command. If the register file stack is at register 9 and a breakpoint is executed, the program counter is destroyed. The most significant byte of the program counter is changed to the least significant byte plus one.

For example, if the breakpoint is set at address 1235₁₆, the breakpoint message is 3635 at 09:, where at is the appropriate status register value. The PC command can be used to change the program counter back to the correct address and program execution can continue.