TI PPC NOTES

NEWSLETTER OF THE TI PERSONAL PROGRAMMABLE CALCULATOR CLUB

P. O. Box 1421, Largo, FL 34649

Volume 13, Number 3 Third Quarter 1988/1989

The "star" article in this issue is on Frank Ferguson's spreadsheet program for the TI-95. But there is something for almost everyone, with two TI-59 programs, a game for the TI-74, and an expanded list of errata for the CC-40 manuals.

I particularly want to call your attention to the discussion of interfacing with peripherals on page 23. Hardly a month goes by without some member writing to describe interface problems or limitations. So, I would like to gather your collective experience and publish it in the next issue.

In the next issue I also hope to cover several of the more extensive programming efforts which have been in progress by our members. One is a set of navigation related programs for the TI-95 which has been put together by Hewlett Ladd and Al Mackenzie. The documentation presently runs to 36 pages. A second is a TI-95 utilities package developed by Robert Prins in machine language.

Enclosed with this issue you will also find a promotion sheet for the BYTE Information Exchange which offers a discount for club members. I don't have the capability to participate so you will have to make your own judgement on whether to join or not.

Finally, I note that some of the mystery and magic of the computer technology may be disappearing. The magazine rack in the Waldenbooks bookstore in Sunshine Mall, Clearwater, Florida displays the following notice: "COMPUTER MAGAZINES HAVE MOVED TO RACK NEXT TO COMICS".

TABLE OF CONTENTS

Magnetic card service continues to be available for TI-59 programs in this issue, and for the TI-59 programs in earlier issues back to 1983.

Send one dollar per card plus a stamped and self addressed envelope.

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 1>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
```

System Register Tests and the Unprotect Mode - V13N2P8. Scott Garver and Robert Prins wrote to explain that there is no need to remove system protection in order for system functions to work in programs. They called my attention to the note at the bottom of page A-2 of the *TI-95 Programming Guide*:

> "You can use any of the system functions from the keyboard or from within a program. To store a system function as a program instruction, however, you must remove the system protection before entering the learn mode".

This means that there is no need to use SF 49 to control entry to the system-unprotected mode as described in V13N2P8/9, and the SF 49 and RF 49 commands in the program listings on V13N2P10 are useless. The following listing reflects the removal of those commands. The only other change is that the address for the GTO at line 163 must be changed to 0177.

```
0000 CE `SQUARE ROOT N`
0014 ADV PRT CMS CE `1<`
0020 `N<100 ?` BRK ADV
0029 PRT STO 049 CE `# `
0036 `BLOCKS? <41:` BRK
0049 ADV STO 002 10
0055 STO 000 INV LOG
0060 STO 001 1 STO 008
0067 STO 089
0070 LBL 01 90 STO 003
0078 STO 005 50 STO 004
0086 STO 006 RCL 008
0092 STO 007 +/- ST+ 003
0099 ST+ 004
0102 LBL 02 RCL IND 003
0109 INC 003 x<>t
0113 RCL IND 004 INC 004
0120 IF= 2079 GTL 02
0126 IF< 2079 GTL 05 0
0133 x<>t 0
0135 LBL 03 INV INC 005
0142 INV INC 006
0146 ST- IND 006
0150 RCL IND 005
0154 ST- IND 006
0158 RCL IND 006 INV
0163 IF< 2079 GTO 0177
0169 RCL 001 ST+ IND 006
0176 1. DSZ 007 GTL 03
0184 INC 009 2
0188 LBL 04 ST+ 089
0194 GTL 01
0197 LBL 05 DSZ 000
0203 GTL 06 10 EXC 000
0211 EXC 009 PRT INC 008
0218 DSZ 002 GTL 06 CE
0225 `MORE BLOCKS?` Y/N
0238 GTL 11 GTO 0000
0244 LBL 06 RCL 007
0250 STO 003 10 ST* 009
0258 LBL 07 INV INC 005
0265 ST* IND 005 DSZ 003
0272 GTL 07 x^2
0276 LBL 08 INV INC 006
0283 ST* IND 006 DSZ 007
0290 GTL 08 2 STO 007 50
0299 LBL 09 STO 006
0305 RCL 008 EXC 003 x<>t
0312 0 x<>t
0314 LBL 10 ST- IND 006
0321 INV INC 006 x<>t
0326 ST+ IND 006 (
0331 RCL IND 006 /
0336 RCL 001 )( INT *
0343 x<>t RCL 001 )
0348 DSZ 003 GTL 10 90
0356 DSZ 007 GTL 09 9
0363 +/- GTL 04
0367 LBL 11 CE `# BLOCK`
0378 `S?` BRK STO 002
0384 GTL 06
```

```
10 INPUT "N = ";N
20 INPUT "D = ";D
30 X=N MOD D
40 R=X-D*INT(X/ABS
(D))*SGN(D)
50 Q=INT(N/ABS(D))
*SGN(D)
60 PRINT
65 PRINT "N = ";N
70 PRINT "D = ";D
75 PRINT "Q = ";Q
80 PRINT "R = ";R
90 GOTO 10
99 END
```

CHARLIE'S CHALLENGE - Myer Boland provided the first response to Charlie Williamson's challenge to write a program to divide two numbers such that the remainder will be a non-negative number less than the absolute value of the divisor, but without the use of direct comparison logic. Myer's BASIC program which appears at the right obtains the correct answers for the sample problems in V13N2P25.

Editor's Note: The MOD command in line 30 is available with Myer's computer and with my Model 100. It is not available with my Color Computer or with the TI-74 and CC-40. For those machines it is necessary to synthesize a MOD command. That appeared somewhat difficult to do without using comparisons, but then I discovered that all three machines, and the Model 100 as well, mechanize an integer function (INT) which is really a "floor" function such that

INT(2.3) = 2 and INT(-2.3) = -3

For those machines line 30 of Myer's program can then be easily replaced with:

```
30 X = N - D*INT(N/D)
```

So, once again we find that solving programming challenges provides additional insight into how our machines operate. In contrast the integer function of progrmmable calculators such as the TI-59, TI-95, HP-11 and fx-7000G return -2 for an input of -2.3 . That result can be obtained on the Model 100 with the FIX function.

GRA LISTINGS - The abstracts for three additional documents offered by Government Reports Announcements in 1988 are:

825,320
PB88-148101/GAR PC A13/MF A01
Hittman Associates, Inc., Columbia, MD.
Selected Hand-Held Calculator Codes for the Evaluation of the Cumulative Hydrologic Impacts of Mining, .
R. Sandberg, R. B. Scheibach, D. Koch, and T. A. Prickett. Sep 82, 298p OSM/TR-8a/3
Portions of this document are not fully legible. Sponsored by Office of Surface Mining Reclamation and Enforcement (DI), Washington, DC.

The objectives of the investigation were: to present a selected group of analytical models that will enable quick evaluation of both individual and cumulative impacts of surface mining on water resources; to consider the needs of the Office of Surface Mining relative to the capabilities of presently available programmable hand-held calculators; to describe techniques that fill the gap that now exists between ordinary paper-and-pencil evaluations and full-sized digital computer evaluations; to allow quick evaluation of permit applications without the need for complicated time-consuming computer analysis; to design calculator programs for the needs Office of Surface Mining (OSM), Region V. They are, however, applicable to hydrologic problems in other areas of the United States as well, since the basic hydrologic theories used are not dependent upon site-specific conditions.

834,123
PB89-182274/GAR PC E03/MF A01
Foersvarets Forskningsanstalt, Umea (Sweden).
Radiakraeknedosa: Omprogrammerad Version (Radiac Calculator),
G. Hulten, G. Jonsson, and T. Ulvsand. Sep 87, 34p FOA-E-40035
Text in Swedish.

A calculator, HP41CV, has been modified for radiac calculations. The radiac calculator has fixed programs for calculation of intensities, doses, nuclear fallout fields, weapon yield, and radii for damage. The report shows the equations used, but it is primarily a users' guide.

907,562
PB89-118194/GAR PC E04/MF E04
Institut Franco-Allemand de Recherches, Saint-Louis (France).
Sur l'Usage des Petits Calculateurs de Poche Comme Calculateurs de tir pour les Canons et les Obusiers (Use of Pocket Calculators When Firing Guns and Howitzers) (Verwendung von Taschenrechnern Beim Schiessen mit Geschuetzen und Haubitzen),
A. Carriere. 2 Sep 85, 47p ISL-N-610/85
Text in French; summary in German and English. Sponsored by Direction des Recherches, Etudes et Techniques, Paris (France). Centre de Documentation de l'Armement.

The possible fields of application are described in the report for programmable pocket calculators during the firing of howitzers. The simple programs lend themselves very well to flight path calculations in view of the adjustment of these artillery weapons.

AN UNUSUAL TI-59 MALFUNCTION - P. Hanson. In the past few months I found two TI-59's which operated satisfactorily with the battery installed but which would exhibit intermittent problems when mounted on a PC-100. In each case I could induce a loss of power by twisting the calculator on its mount on the printer. Even the twist induced by normal keystroking could cause the problem. The source of the problem was traced to the 22 uf capacitor C7, the small red capacitor mounted near the negative battery contact of the calculator. In the defective calculators the capacitor was located such that when the contact was depressed it touched the body of the capacitor. With each calculator I could see evidence of repeated scraping of the edge of the contact against the capacitor. The solution is to move the body of the capacitor. Since the problem did not occur with the battery installed it seems likely that the PC-100 mount depresses the battery contacts further than the battery.

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

SPREADSHEET FOR THE TI-95 - Frank Ferguson. The next four pages provide the documentation for a clever program which operates like a limited range spreadsheet. The program accepts input data into one or two columns, and then provides up to four output columns using functions of the input data which are mechanized through a manipulator program entered by the user. The program illustrates the saving and recalling of data files under program control. The input column entries are stored starting at data register 050.

One idiosyncrasy not mentioned in the instructions: If you call an output column (C1 through C4) at step 3.6 and a corresponding program is not in place in the manipulator program then the error flag will be set and the display will read "LABEL NOT FOUND". You will have to press CLEAR to return to program control.

1.0 INTRODUCTION

The **SPREAD SHEET** program is designed for applications wherein 1 or 2 columns of data can be manipulated or combined for various reasons. The user enters the data and the program will generate an output data table consisting up to 4 columns. Each column of the output data is a function of the input data determined by a manipulator program written by the user.

2.0 RUNNING THE PROGRAM

The **SPREAD SHEET** program must be stored in the memory file or an 8k RAM cartridge, under the file name SPD. Make sure that the content of program memory is saved because you will be using it to write the manipulator program.

2.1 Now begin the program, select (SPD) form memory file or 8k RAM.

2.2 The program displays a menu to let you choose the source of data.

<NEW> To enter NEW data from the keyboard.

<OLD> If you have previously saved data in program file or 8k RAM, OLD allows you to retrieve it.

<ESC> Exit the program. See step 4.0

2.3 Select <NEW> to begin entering data.

The program displays:

<CL1> Lets you enter one column of data.

<CL2> Lets you enter two columns of data.

<ESC> Exits the program. See step 4.0

2.4 Select <CL2> to enter two columns of data labeled A and B.

The program will respond by asking for the number of data in a single column.

2.5 Enter 5, then press <GO>.

The program will briefly display "ENTER LIST A" and then will prompt for the data with "n=?" .

2.6 Entering data.

Start with data list A enter 3 press <GO>, enter 7 press <GO> and so on. When all data in list A has been entered the program will briefly display "ENTER LIST B". Next, proceed as you did for list A to enter List B.

2.7 Input data.

LIST A	LIST B
3	20
7	13
11	15
5	9
1	3

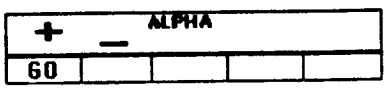
2.8 After all data has been entered the program will display:

<YES> Allows you to save data in memory file or 8k RAM.

<NO> Continues to the DATA TABLE. The data will be lost when you exit the program

2.9 Press <YES> to save data as a memory file.

The program briefly displays "Data Name ?" then will prompt you for the name of the data. (The required "+" is provided for you.)



Note: The calculator is already in alpha mode.
DO NOT press ALPHA to enter name.

2.10 Enter DT then press <GO>. This will save data as +DT.

The program will prompt you to "Select File" MEM or RAM.

Select files				
MEM	RAM			RPT

<MEM> Will save data in memory file.

<RAM> Will save data in the 8k RAM. If the 8k RAM is not installed the Calculator will display error message "INVALID DIR/FILE" and the data will be lost.

<RPT> Returns to step 2.8 to repeat filing process if desired.

2.11 Select <MEM>

The program will briefly display "FILE +DT SAVED" and continue to "DATA TABLE"

DATA TABLE				
C1	C2	C3	C4	ESC

<C1> thru <C4> Allows you to view the data table.

* <ESC> Exits program. See step 4.0

3. MANIPULATOR PROGRAM

The Manipulator Program is to be entered into the "Program Memory". It can be entered at this point in the program or before you begin running SPREAD SHEET.

3.1 Press LEARN. To put calculator in learn mode.

3.2 Press <lst>. See "Programming Guide" if you are not familiar with using the program memory.

Each column of the data table calls a subroutine in program memory. Column C1 calls LBL C1, column C2 calls LBL C2 and so on.

3.3 Now write Manipulator Program as follows:

```
LBL C1 RCL A * RCL B = RTN
LBL C2 RCL A / RCL B = RTN
LBL C3 RCL A - RCL B = RTN
LBL C4 RCL A + RCL B = RTN
```

3.4 Press LEARN to leave learn mode.

3.5 Press OLD to return to "DATA TABLE"

The program should display the following:

DATA TABLE				
C1	C2	C3	C4	ESC

3.6 Now press <C1> to display column 1 of the data table.

The program will display the product of data A and data B as follows:

```
Column 1
( 1 )      60.00
( 2 )      91.00
( 3 )     165.00
( 4 )      45.00
( 5 )       3.00
END
```

The number in parenthesis is the row number of each column.

3.7 To see the input data A in the parenthesis go to learn mode and insert STO G after RCL A in subroutine LBL C1.

```
LBL C1 RCL A STO G * RCL B = RTN
```

3.8 Return to data table as in step 3.4 and 3.5.

3.9 Press <C1>. The program should display as follows:

```
Column 1
( 3 )      60.00
( 7 )      91.00
( 11 )    165.00
( 5 )      45.00
( 1 )       3.00
END
```

The number in parenthesis resides in register G. You can store any number in G as long as it done in the Manipulator Program.

3.9 Continue with columns <C2> thru <C4>.

To temporarily stop the display from changing press and hold any key except BRK until you want the listing to continue.

4.0 QUITTING SPREAD SHEET

After you have finished using SPREAD SHEET you have the option to delete any data file you wish.

4.1 Press <ESC>. This will return to the beginning of SPREAD SHEET. See step 2.2 .

Now you can either enter new or old data and run the program again, or escape from the program.

4.2 Press <ESC> again

At this point the program will ask you Delete File ? yes or no. If you choose <YES> the program will ask you to name the file as you did in steps 2.9 and 2.10, when you saved a file

If you select <NO> the program will clear the display and reset normal mode.

4.3 Select <NO>. This will end the running of the program.

5.0 RETRIEVING OLD DATA

At this point you should have the input data that you saved in memory as +DT and the manipulator program stored in Program Memory.

You may choose to save the Manipulator Program for later use under any name you desire.

5.1 Repeat step 2.1 and 2.2 but this time select OLD.

The program briefly displays "DATA NAME" then prompts you for the name of the data.

5.2 Enter DT then press <GO>.

The program will prompt you to "Select File" MEM or RAM.

5.3 Select <MEM>.

The program will briefly display "FILE +DT READ" and continue to "DATA TABLE".

Now you are ready to use the previously entered data for output data columns.

6.0 CONCLUSION

If you wish to use a previously written Manipulator Program, it must be stored in "Program Memory" before you began running SPREAD SHEET.

```
TI-95 PROGRAM   (FERGUSON 11-05-87)          (LAST REVISION 12-14-88)
SPREAD SHEET                  MAIN PROGRAM

          KEYSTROKE                          COMMENTS              page 1
-------------------------------------------------------------------------
LBL AA DFN CLR CF6 CE              Clears menus, flags, and display.
       50 STO E 64 STO A           Initialize registers.
       '   SPREAD SHEET'           Displays title.
       DFN F1:NEW@A7               Defines menu windows for data
       DFN F2:OLD@A8               input options.
       DFN F5:ESC@QQ
       RTN
LBL A7 CE 'NO. of data col.'       Prompts for the number of input
       DFN F1:CS1@A5               data columns.
       DFN F2:CS2@A6
       RTN
LBL A6 SBL IN                      Calls enter data subroutine for
       SBL 01                      first data list.
       RCL D STO C SF 00           Reset loop counter and set flag.
LBL A5 SBL IN                      Calls enter data subroutine for
       SBL 01                      second data list.
LBL B2 CE 'File Data ?'            Display file data option.
       Y/N TF 02
       GTL DT                      If no go to DATA TABLE.
       DFN F5:RPT@B2               Define repeat 'file data' option.
       SBL FL DFN CLR              If yes call routine to name data.
       SBL PT                      Call PUT file subroutine.
LBL DT RCL D STO C 0 STO B         Initialize registers.
       50 STO E
       CE '   DATA TABLE'          Display 'DATA TABLE' and define
       DFN F1:C1 @A1               column identifiers.
       DFN F2:C2 @A2
       DFN F3:C3 @A3
       DFN F4:C4 @A4
LBL ES DFN F5:ESC@AA               Option to escape to beginning.
       RTN
LBL A1 CE 'C1' STA H 49 GTL A9     Enables columns to run.
LBL A2 CE 'C2' STA H 50 GTL A9
LBL A3 CE 'C3' STA H 51 GTL A9
LBL A4 CE 'C4' STA H 52
LBL A9 SBL CL                      Calls column display subroutine.
LBL B1 SBL S1                      Calls data locating subroutine.
       SBL AE INC E DSZ C          Calls sub. to format results .
       GTL B1                      Operates on next data bit.
       'END' PRT                   Display END if last data bit.
       GTL DT                      Returns Program to DATA TABLE.
LBL A8 SBL ES                      If using OLD data give ESC option.
       SBL FL                      Calls name data subroutine.
       SBL GT                      Calls subroutine to GET data.
       GTL DT                      Returns program to DATA TABLE.
LBL QQ CE 'Delete Files ?'         Delete file option.
       Y/N TF 02
       GTL NO                      If no go to stop routine.
       SBL FL                      If yes calls name data subroutine
       SBL ES                      Defines F5 as ESC .
       SBL DF                      Call delete file subroutine.
       GTL QQ                      Returns program to 'Delete File ?'.
LBL NO 0 DFN CLR HLP               Clears display, menu window, resets
       HLT                         normal mode, and halts program.
```

SUBROUTINES

KEYSTROKE	COMMENTS	page 2
LBL IN DFN CLR CE INC A TF 00 GTL 02 ADV 'No. of data' BRK STO C STO D LBL 02 CE 'ENTER LIST' CHR IND A PRT RTN	Subroutine IN allows you to enter data. It prompts for the number of data to be entered, then displays which list you are to enter.	
LBL 01 CE 'n=' SBL PI STO IND E INC E DSZ C GTL 01 RTN	Subroutine 01 is called by subroutine IN to store the data as you enter it.	
LBL PT RCL D STO 049 * 2 + .049 = TF 11 INV PUT IND H ADV PRT RTN	Subroutine PT (PUT data) stores data registers, starting with register 49, into main memory. of 8K ram.	
LBL CL STO B CE 'Column ' CHR IND B PRT RTN	Subroutine CL displays which column of the DATA TABLE is being selected by user.	
LBL S1 RCL IND E STO A RCL E STO B RCL D ST+B RCL IND B STO B INC B RTN	Subroutine S1 resets registers and loop caunters being used.	
LBL AE SF 10 SBL PH CE 'PGM' RUN SBL IND H STO A RCL B SBL CT 16 STO B TF 44 INC B TF 44 INC V 3 ST+V '(' COL IND V MRG B ')' RF 67 RF 44 RCL A ABS STO F .01 INV IF < F EE 900 IF < F EE RCL A FIX 2 RND OLD TF 44 INC B COL IND B MRG A SF 56 PRT SBA 27C RF 43 RF 44 RF 67 SBL PP RTN	Subroutine AE formats the display for output data	
LBL ST 49 TF 11 INV GET IND H PRT RCL 049 STO D RTN	Subroutine ST GET's data stored in main memory and stores it into data registers.	
LBL FL CE 'Data Name ?' PRT PAU CE SF 60 '+ ' BRK STA H RF 60 CE 'Select File' DFN F1:MEMORY DFN F2:RAM&RM MLT LBL RM SF 11 LBL ME RTN	Subroutine FL allowes the user to name the data file to be stored it into main memory or 8K ram.	

KEYSTROKE	COMMENTS	page 3
LBL DF TF 11 INV DF IND H PRT RTN	Subroutine DF deletes data files in memory or 8K ram.	
LBL PI BRK OLD COL 16 MRG= TF 74 PRT RTN	Subroutine PI is a input prompt utility	
LBL PH 8 PUT +ss RTN	Subroutine PH (PUSH) presurves the contents of the first 8 registers.	
LBL PP 0 GET +ss RTN	Subroutine PP (POP) restores the contents of the first 8 registers.	
LBL CT ABS STO U 0 STO Z STO X STO V LBL 10 INC X 10 ST/U RCL U INT IF = Z GTL 11 5 TF 10 IF > X GTL 10 RCL X STO V EE FIX 2 RTN LBL 11 INC V 10 ST*U RCL U FRC IF = Z GTL 12 5 TF 10 IF > V GTL 11 EE FIX 2 RTN LBL 12 RCL X IF < V RTN RCL X STO V RTN	Subroutine CT (COUNT) is a utility that counts the number of digits in the display register.	

NOTE: A PC-324 listing with line numbers appears on page 9

TI-95 Spreadsheet - (cont)

```
0000 LBL AA DFN CLR CFG
0006 CE 50 STO E 64
0013 STO A `  SPREAD SH`
0026 `EET` DFN F1:NEW@A7
0036 DFN F2:OLD@A8
0043 DFN F5:ESC@QQ RTN
0051 LBL A7 CE `No. of `
0062 `data col.`
0071 DFN F1:CS1@A5
0078 DFN F2:CS2@A6 RTN
0086 LBL A6 SBL IN
0092 SBL 01 RCL D STO C
0099 SF 00
0101 LBL A5 SBL IN
0107 SBL 01
0110 LBL B2 CE `File Da`
0121 `ta ?` Y/N TF 02
0128 GTL DT
0131 DFN F5:RPT@B2
0138 SBL FL DFN CLR
0143 SBL PT
0146 LBL DT RCL D STO C
0153 0 STO G 50 STO E CE
0161 `   DATA TABLE`
0174 DFN F1:C1 @A1
0181 DFN F2:C2 @A2
0188 DFN F3:C3 @A3
0195 DFN F4:C4 @A4
0202 LBL ES
0205 DFN F5:ESC@AA RTN
0213 LBL A1 CE `C1`
0219 STO H 49 GTL A9
0226 LBL A2 CE `C2`
0232 STO H 50 GTL A9
0239 LBL A3 CE `C3`
0245 STO H 51 GTL A9
0252 LBL A4 CE `C4`
0258 STO H 52
0262 LBL A9 SBL CL
0268 LBL B1 SBL SI
0274 SBL AE INC E DSZ C
0281 GTL B1 `END` PRT
0288 GTL DT
0291 LBL A8 SBL ES
0297 SBL FL SBL GT
0303 GTL DT
0306 LBL QQ CE `Delete `
0317 `File ?` Y/N TF 02
0326 GTL NO SBL FL
0332 SBL ES SBL DF
0338 GTL QQ
0341 LBL NO 0 DFN CLR
0347 HLP HLT
0349 LBL IN DFN CLR CE
0355 INC A TF 00 GTL 02
0362 ADV `No of data`
0373 BRK STO C STO D
0378 LBL 02 CE `ENTER L`
0389 `IST ` CHR IND A
0396 PRT RTN
0398 LBL 01 CE `n=?`
0405 SBL PI STO IND E
0411 INC E DSZ C GTL 01
0418 RTN
0419 LBL PT RCL D
0424 STO 049 #24.049=
0435 TF 11 INV PUT IND H
0441 ADV PRT RTN
0444 LBL CL STO D CE `C`
0451 `olumn ` CHR IND D
0460 PRT RTN
0462 LBL SI RCL IND E
0468 STO A RCL E STO D
0474 RCL D ST+ D
0478 RCL IND D STO D
0483 INC G RTN
0486 LBL AE SF 10 SBL PH
0494 CE `PGM` RUN
0499 SBL IND H STO A
0504 RCL G SBL CT 16
0511 STO D TF 44 INC D
0517 TF 44 INC V 3 ST+ V
0524 `(` CDL IND V MRG G
0530 `)` RF 67 RF 44
0535 RCL A ABS STO F .01
0543 INV IF< F EE 900
0550 IF< F EE RCL A
0555 FIX 2 RND OLD TF 44
0561 INC D CDL IND D
0566 MRG A SF 56 PRT
0571 SBA 27C RF 43 RF 44
0578 RF 67 SBL PP RTN
0584 LBL GT 49 TF 11 INV
0592 GET IND H PRT
0596 RCL 049 STO D RTN
0602 LBL FL CE `Data Na`
0613 `me` PRT PAU CE
0618 SF 60 `+` BRK STA H
0624 RF 60 CE `Select F`
0635 `ile` DFN F1:MEM@ME
0645 DFN F2:RAM@RM HLT
0653 LBL RM SF 11
0658 LBL ME RTN
0662 LBL DF TF 11 INV
0668 DF IND H PRT RTN
0673 LBL PI BRK OLD
0678 CDL 16 MRG = TF 74
0684 PRT RTN
0686 LBL PH 8 PUT +$$
0694 RTN
0695 LBL PP 0 GET +$$
0703 RTN
0704 LBL CT ABS STO U 0
0711 STO Z STO X STO V
0717 LBL 10 INC X 10
0724 ST/ U RCL U INT
0729 IF= Z GTL 11 5
0735 TF 10 IF> X GTL 10
0742 RCL X STO V EE
0747 FIX 2 RTN
0750 LBL 11 INC V 10
0757 ST* U RCL U FRC
0762 IF= Z GTL 12 5
0768 TF 10 IF> V GTL 11
0775 EE FIX 2 RTN
0779 LBL 12 RCL X IF< V
0786 RTN RCL X STO V RTN
```

MORE PPX PROGRAMS - A. E. Mackenzie (code 7 in our exchange, see V13N1P10) has updated his list of PPX programs. The following list includes six programs not previously listed and other previously listed programs for which there were less than three other sources.

```
7   198011 - General Annuities
79  338001 - Roots of a Polynomial (Newton's Method)
79  338005 - Quartic, Cubic, Quadratic Solutions
679 338009 - Lin-Bairstow Roots of a Polynomial
79  348004 - Adaptive Simpson's Method (Integration)
79  398049 - Rectangular/Spherical Coordinate Conversion
79  398050 - Ellipse and Circle Plotter
7   398056 - Hermite Polynomials
79Q 398096 - Polar Graphing Program
7   398099 - Two Dimensional Vector Computer
7   398167 - Vector Addition and Components
47  398194 - Properties of Circles
7   398196 - Addition of Fractions
7   668108 - PV-09 Pressure Vessels
7GQ 698006 - Axonometric Projection
79Q 698007 - Isometric Projection
79Q 698008 - Oblique Projection
79K 788015 - Moon Phase for Any Date
7   948005 - Almanac Date for Stars
279 948014 - Sunrise/Sunset
```

The PV-09 Pressure Vessels program (PPX 668108) was incorrectly listed on V13N1P6 as 668105. A1 also has the Astrology, 59-Fun, 3D Graphics, Mathematics, Programming Aids, Securities, and Printer Utility specialty pakettes.

THE TI-68 - The "EEs' Tools & Toys" column of the June 1989 issue of *IEEE SPECTRUM* included a favorable description of a new programmable calculator, the TI-68. The suggested retail price is $65.00. Capabilities of the device mentioned in the article include:

* 250 functions including 40 complex number functions.

* Solves up to five simultaneous equations with real or complex coefficients.

* Finds real and complex roots for quadratic, cubic and quartic equations.

* 36 memory registers and 440 program steps.

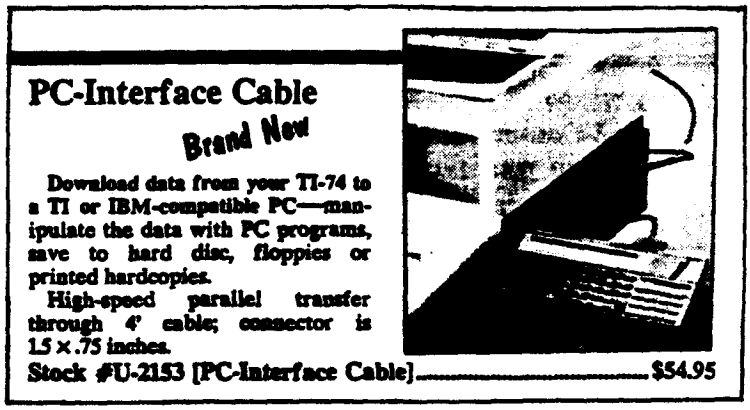
* Integration by Simpson's rule.

* Display of twelve digits.

Page 99 of the latest Elek-Tek catalog (Volume 18) offers the device for $42.00 plus $4.00 for shipping and handling. When I called Elek-Tek I was told that it was not yet in stock. The device was also not available from EduCALC. I called 1-800-TI-CARES on 26 June. I was told that the device was not yet released. It is expected to be available later in the year, perhaps as late as the fourth quarter.

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.

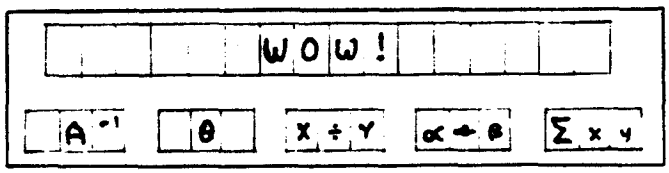
I also called 1-800-TI-CARES. I was told that the device would not be available until the third or fourth quarter.

Editor's Comment: Those who object to the tendency of the computer industry to announce products well in advance of their actual availability will enjoy Kenneth Sheldon's article "Moby Dick 2.1" on page 344 of the July 1989 issue of *BYTE*. The article starts by speculating:

> "As a fan of great literature, I found myself wondering the other day, 'What if novels were published the way software is?' If they were the process might go something like this: Herman Melvile would announce the publication of *Moby Dick* a year before you could actually buy it. Revievers would praise it, and several literary magazines would select it as 'Editor's Choice' for best novel of the year--all before it ever appeared on bookstore shelves. ... "

A TI-95 PROGRAMMING CHALLENGE FROM SCOTT GARVER

Write a program which will yield the following display:



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:

	Number of Integers			
Method	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:

	Number of Integers			
Method	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.

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 Gutkn
echt
30 REM 1/07/87
1000 WHERE_START=5! defi
ne 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 A
T(TARGET_LOCATION),"*";
5000 DISPLAY AT(ARROW_LO
CATION),ARROW$;
5010 PAUSE .5
5100 IF GAME_DELAY=0 THE
N 5500
5200 LOOP=FIRST_PLACE
5210 DISPLAY AT(LOOP-LOO
P_CHANGE),ARROW_FULL$;
5300 DELAY_LOOP=1
5310 CALL KEY(KEY,KEYPRE
SS):IF KEYPRESS THEN 600
0
5320 IF DELAY_LOOP=GAME_
DELAY THEN 5400
5330 DELAY_LOOP=DELAY_LO
OP+1
5340 GOTO 5310
5400 LOOP=LOOP+LOOP_CHAN
GE
5410 IF LOOP=(TARGET_LOC
ATION+LOOP_CHANGE-REVERS
E)THEN 6100
5420 GOTO 5210
5500 LOOP=FIRST_PLACE
5510 DISPLAY AT(LOOP-LOO
P_CHANGE),ARROW_FULL$;
5520 CALL KEY(KEY,KEYPRE
SS):IF KEYPRESS THEN 600
0
5530 LOOP=LOOP+LOOP_CHAN
GE
5540 IF LOOP=(TARGET_LOC
ATION+LOOP_CHANGE-REVERS
E)THEN 6100
5590 GOTO 5510
6000 IF LOOP=(TARGET_LOC
ATION-REVERSE)THEN 6200
6100 DISPLAY ERASE ALL,"
Lose..";
6120 GOTO 6500
6200 DISPLAY ERASE ALL,"
Win!!!";
6210 GAME_SCORE=GAME_SCO
RE+1
6500 PAUSE .3
6510 GAME_NUMBER=GAME_NU
MBER+1
6520 DISPLAY AT(10),"Sco
re:";GAME_SCORE;
6530 DISPLAY AT(22),"Gam
es:";GAME_NUMBER;
6600 CALL KEY(KEY,KEYPRE
SS)
6610 IF KEYPRESS=0 THEN
6600
6990 GOTO 4500
7000 FIRST_PLACE=2
7010 LOOP_CHANGE=1
7020 ARROW_LOCATION=1
7030 ARROW$=">"
7040 ARROW_FULL$="->"
7190 RETURN
7200 FIRST_PLACE=29
7210 LOOP_CHANGE=-1
7220 ARROW_LOCATION=31
7230 ARROW$="<"
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,  8K
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 ) , "*" ;
4999 !
5000 display at ( arrow_location ) , arrow$ ;
5010 pause 0.7
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
```

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 useage 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 Gutkn
echt
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 ON ERROR STOP
3060 ON BREAK STOP
3100 ACCEPT AT(17)SIZE(2
)NULL(2)VALIDATE(DIGIT),
GD
3200 ON ERROR 3000
3210 ON BREAK ERROR
3500 GS=0
3510 GN=0
4000 S=S+1! Create diffe
rent seed for each game
4100 RANDOMIZE S
4500 KP=0
4510 TL=INT(((31-WS)*RND
)+WS)+1
4600 RV=INT(2*RND)
4800 ON (RV+1)GOSUB 7000
,7200
4900 DISPLAY ERASE ALL A
T(TL),"*";
5000 DISPLAY AT(AL),AR$;
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
$;
5520 CALL KEY(KEY,KP):IF
 KP THEN 6000
5530 L=L+LC
5540 IF L=(TL+LC-RV)THEN
 6100
5590 GOTO 5510
6000 IF L=(TL-RV)THEN 62
00
6100 DISPLAY ERASE ALL,"
Lose..";
6120 GOTO 6500
6200 DISPLAY ERASE ALL,"
Win!!!";
6210 GS=GS+1
6500 PAUSE .3
6510 GN=GN+1
6520 DISPLAY AT(10),"Sco
re:";GS;
6530 DISPLAY AT(22),"Gam
es:";GN;
6600 CALL KEY(KEY,KP)
6610 IF KP=0 THEN 6600
6990 GOTO 4500
7000 FP=2
7010 LC=1
7020 AL=1
7030 AR$=">"
7040 AF$="->"
7190 RETURN
7200 FP=29
7210 LC=-1
7220 AL=31
7230 AR$="<"
7240 AF$="<-"
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 "E0 I/O error 31 "1" which appears.

RECENT TI-95 PRICES

Page 35 EduCALC Catalog #44: Order a TI-95 for $89.95 plus $1.00 for shipping and handling and you may order a Mathematics or Statistics cartridge for $5.00 more. Additional cartridges are #39.95 each. Call 1-800-633-2252, ext 353 for credit card orders.

Page 99 of Volume 18 of the Elek-Tek catalog offers a TI-95 for $75.00 plus $4.00 for shipping and handling. Call 1-800-621-1269 for credit card orders.

Page 50 of the August 1989 DAMARK International catalog: Order a TI-95 for $79.00 plus $6.00 for shipping and handling and get the Mathematics and Statistics cartridges at no extra charge. Call 1-800-729-9000 for credit card orders. Ask for Item No. B-447-122359.

DAMARK is known for selling factory closeouts, distributor overstocks, discontinued merchandise, etc. The availability of the TI-95 through DAMARK would seem to confirm the feeling that the product line is being discontinued or deemphasized. A call to 1-800-TI-CARES confirms that TI is pursuing no new development in support of the TI-95. They are continuing to support repairs. In contrast, as noted on page 10, they are continuing development of peripherals for the TI-74.

COMMENTS ON THE TI-60 - Gene Friel writes "I recently bought a TI-60 for quick calculations and found that, unlike the TI-55II which I recently sold, RST in the TI-60 goes to step 0 and continues the program. R/S had to be pressed for each loop in the TI-55II. Also, backstepping from step 0 in the TI-60 goes to the end of the program, unlike the TI-55II."

DOUBLE EXPONENTIAL EVALUATION OF DATA ON THE TI-59 - M. Bogart

Consider a set of data that might be expected to conform to a function of the type $Y = aX^bZ^c$. For the purposes of curve fitting the function can be linearized by taking the logarithm of both sides yielding

$$\ln(Y) = \ln(a) + b*\ln(X) + c*\ln(Z)$$

or

$$y = A + Bx + Cz$$

where y = ln(Y), A = ln(a), etc. One then can fit the linearized data in the least squares sense using well known techniques for multiple linear regression with two independent variables. For example, see pages 78-79 of the second edition of Kolb's *Curve Fitting for Programmable Calculators* or chapter 15 of Spiegel's *Theory and Problems of Statistics* in the Schaum's Outline Series. A solution for the TI-74 was discussed oon V12N3P19.

The TI-59 program on page 17 not only provides the least squares solution, but also provides for replacement of erroneous entries, evaluation of the resulting function at other points, and for input of user-selected solution constants.

Double Exponential Evaluation of Data on the TI-59 - (cont)

User Instructions

1. Press E to initialize by clearing the summation registers.

2. Press A to start data entry. See a 1 in the display indicating that the first x value is to be entered.

3. Enter the X value and press R/S. See a 2 in the display.

4. Enter the Y value and press R/S. See a 3 in the display. Note that in this program the dependent variable is entered before the second independent variable.

5. Enter the Z value and press R/S. After a short delay to complete the summations, see a number flashed in the display indicating the number of data points entered, and then see a 1 as a prompt for the X value of the next data point.

6. Repeat steps 3 through 5 as required.

7, Press B to solve. After about 45 seconds the value for a will appear in the display.. Press R/S to see the value for b, and press R/S again to see the value for c. If you press R/S again you will see .1111111111 in the display indicating that the output of the solution is complete.

8. To evaluate the function for values of Xi and Zi, enter the value for Xi and press C. The value of ln(Xi) will appear in the display. Enter the value of Zi and press R/S. The value of Yi will appear in the display.

9. To enter a set of user selected values for a, b, and c, press D and see 20 in the display. Enter the value for a and press R/S. See 21 in the display. Enter the value for b and press R/S. See 22 in the display. Enter the value for c and press R/S. See a zero in the display. If you want the selected coefficients printed press R/S. To evaluate the function with the selected coefficients go to step 8.

10. Printout with annotation is provided if a PC-100 is available. The printout for a sample problem appears at the right. You can avoid stopping for readout in the display by changing the R/S instructions to Nop's at steps 169 and 178.

11. Removal of incorrectly entered points. If you detect the incorrect entry immediately after the completion of the entry with the program ready to accept the next point as at step 3, you can remove the bad point from the summation by pressing 2nd A'. Then press A to resume data entry. An example appears after the third entry in the printout at the right.

If you do not detect the incorrect entry until after another data point has been entered you will have to re-enter the bad point, press 2nd A' two times (first to remove the bad point just entered, and then to remove the bad point previously entered). An example appears in the printout where the sixth point was entered to permit correction for the bad entry at point 4.

```
            21.        X
     0.201064          Y
            2.          Z
            1.          N

           40.        X
      1.43996          Y
            7.          Z
            2.          N

          100.        X
      4.34868          Y
          500.          Z
            3.          N

DELETED LAST POINT

          100.        X
      4.34868          Y
            5.          Z
            3.          N

          500.        X
      42.3097          Y
            3.          Z
            4.          N

           50.        X
      2.67693          Y
           10.          Z
            5.          N

          500.        X
      42.3097          Y
            3.          Z
            6.          N

DELETED LAST POINT
DELETED LAST POINT

          500.        X
      32.3097          Y
            3.          Z
            5.          N

            3.

            5.
21.46520318
7.649692624
21.46520318
98.00981875
32.53547628
7.649692624
32.53547628
13.36615679

48.23935118

4.690393783
28.68031952
8.048646705

.0012000013        A
1.499999478        B
.8000012246        C

           50.        X'
           10.        Z'
2.676930253           Y'

  SELECTED A, B, C

          0.7         A
          1.2         B
          0.3         C

           50.        X'
           10.        Z'
      152.70809       Y'
```

Double Exponential Evaluation of Data on the TI-59 - (cont)

Program Listing:

```
000 76 LBL   080 65 ×     160 03 3     240 69 DP    320 01 1     400 03 03
001 18 C'    081 43 RCL   161 69 DP    241 04 04    321 03 3     401 05 5
002 36 PGM   082 13 13    162 04 04    242 32 X:T   322 03 3     402 07 7
003 02 02    083 95 =     163 43 RCL   243 69 DP    323 06 6     403 01 1
004 91 R/S   084 44 SUM   164 20 20    244 06 06    324 03 3     404 05 5
005 92 RTN   085 31 31    165 22 INV   245 91 R/S   325 07 7     405 00 0
006 76 LBL   086 01 1     166 23 LNX   246 00 0     326 00 0     406 00 0
007 17 B'    087 44 SUM   167 69 DP    247 91 R/S   327 00 0     407 00 0
008 65 ×     088 23 23    168 06 06    248 76 LBL   328 03 3     408 00 0
009 43 RCL   089 03 3     169 91 R/S   249 16 A'    329 03 3     409 00 0
010 11 11    090 01 1     170 01 1     250 43 RCL   330 69 DP    410 00 0
011 95 =     091 69 DP    171 04 4     251 11 11    331 03 03    411 69 DP
012 92 RTN   092 04 04    172 69 DP    252 22 INV   332 03 3     412 04 04
013 76 LBL   093 43 RCL   173 04 04    253 44 SUM   333 02 2     413 69 DP
014 11 A     094 23 23    174 43 RCL   254 24 24    334 02 2     414 05 05
015 98 ADV   095 69 DP    175 21 21    255 33 X²    335 04 4     415 98 ADV
016 01 1     096 06 06    176 69 DP    256 22 INV   336 03 3     416 61 GTO
017 59 INT   097 66 PAU   177 06 06    257 44 SUM   337 01 1     417 10 E'
018 91 R/S   098 61 GTO   178 91 R/S   258 27 27    338 03 3     418 76 LBL
019 32 X:T   099 11 A     179 01 1     259 43 RCL   339 07 7     419 15 E
020 04 4     100 76 LBL   180 05 5     260 12 12    340 00 0     420 00 0
021 04 4     101 12 B     181 69 DP    261 17 B'    341 00 0     421 42 STO
022 69 DP    102 03 3     182 04 04    262 22 INV   342 69 DP    422 23 23
023 04 04    103 36 PGM   183 43 RCL   263 44 SUM   343 04 04    423 42 STO
024 32 X:T   104 02 02    184 22 22    264 29 29    344 69 DP    424 24 24
025 69 DP    105 11 A     185 69 DP    265 43 RCL   345 05 05    425 42 STO
026 06 06    106 01 1     186 06 06    266 13 13    346 91 R/S   426 25 25
027 23 LNX   107 36 PGM   187 98 ADV   267 17 B'    347 76 LBL   427 42 STO
028 42 STO   108 02 02    188 91 R/S   268 22 INV   348 14 D     428 26 26
029 11 11    109 12 B     189 09 9     269 44 SUM   349 02 2     429 42 STO
030 44 SUM   110 43 RCL   190 35 1/X   270 30 30    350 00 0     430 27 27
031 24 24    111 23 23    191 91 R/S   271 43 RCL   351 59 INT   431 42 STO
032 33 X²    112 18 C'    192 76 LBL   272 12 12    352 91 R/S   432 28 28
033 44 SUM   113 43 RCL   193 13 C     273 22 INV   353 23 LNX   433 42 STO
034 27 27    114 24 24    194 32 X:T   274 44 SUM   354 42 STO   434 29 29
035 02 2     115 18 C'    195 04 4     275 25 25    355 20 20    435 42 STO
036 59 INT   116 43 RCL   196 04 4     276 65 ×     356 02 2     436 30 30
037 91 R/S   117 26 26    197 06 6     277 43 RCL   357 01 1     437 42 STO
038 32 X:T   118 18 C'    198 05 5     278 13 13    358 59 INT   438 31 31
039 04 4     119 43 RCL   199 69 DP    279 95 =     359 91 R/S   439 91 R/S
040 05 5     120 24 24    200 04 04    280 22 INV   360 42 STO
041 69 DP    121 18 C'    201 32 X:T   281 44 SUM   361 21 21
042 04 04    122 43 RCL   202 69 DP    282 31 31    362 02 2
043 32 X:T   123 27 27    203 06 06    283 43 RCL   363 02 2
044 69 DP    124 18 C'    204 23 LNX   284 13 13    364 59 INT
045 06 06    125 43 RCL   205 42 STO   285 22 INV   365 91 R/S
046 23 LNX   126 30 30    206 11 11    286 44 SUM   366 42 STO
047 42 STO   127 18 C'    207 91 R/S   287 26 26    367 22 22
048 12 12    128 43 RCL   208 32 X:T   288 33 X²    368 00 0
049 44 SUM   129 26 26    209 04 4     289 22 INV   369 91 R/S
050 25 25    130 18 C'    210 06 6     290 44 SUM   370 98 ADV
051 17 B'    131 43 RCL   211 06 6     291 28 28    371 03 3
052 44 SUM   132 30 30    212 05 5     292 01 1     372 06 6
053 29 29    133 18 C'    213 69 DP    293 22 INV   373 01 1
054 03 3     134 43 RCL   214 04 04    294 44 SUM   374 07 7
055 59 INT   135 28 28    215 32 X:T   295 23 23    375 69 DP
056 91 R/S   136 18 C'    216 69 DP    296 43 RCL   376 01 01
057 32 X:T   137 36 PGM   217 06 06    297 23 23    377 02 2
058 04 4     138 02 02    218 23 LNX   298 01 1     378 07 7
059 06 6     139 13 C     219 42 STO   299 06 6     379 01 1
060 69 DP    140 01 1     220 13 13    300 01 1     380 07 7
061 04 04    141 36 PGM   221 65 ×     301 07 7     381 01 1
062 32 X:T   142 02 02    222 43 RCL   302 02 2     382 05 5
063 69 DP    143 14 D     223 22 22    303 07 7     383 03 3
064 06 06    144 43 RCL   224 85 +     304 01 1     384 07 7
065 23 LNX   145 25 25    225 43 RCL   305 07 7     385 01 1
066 42 STO   146 18 C'    226 20 20    306 69 DP    386 07 7
067 13 13    147 43 RCL   227 85 +     307 01 01    387 69 DP
068 44 SUM   148 29 29    228 43 RCL   308 03 3     388 02 02
069 26 26    149 18 C'    229 21 21    309 07 7     389 01 1
070 33 X²    150 43 RCL   230 17 B'    310 01 1     390 06 6
071 44 SUM   151 31 31    231 42 STO   311 07 7     391 00 0
072 28 28    152 18 C'    232 12 12    312 01 1     392 00 0
073 43 RCL   153 25 CLR   233 22 INV   313 06 6     393 01 1
074 13 13    154 36 PGM   234 23 LNX   314 00 0     394 03 3
075 17 B'    155 02 02    235 32 X:T   315 00 0     395 05 5
076 44 SUM   156 15 E     236 04 4     316 02 2     396 07 7
077 30 30    157 76 LBL   237 05 5     317 07 7     397 01 1
078 43 RCL   158 10 E'    238 06 6     318 69 DP    398 04 4
079 12 12    159 01 1     239 05 5     319 02 02    399 69 DP
```

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 A2 = c.

```
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
```

The program listing appears on page 19.

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
egression
20 REM with User Defined
 Functions
30 REM 17 July 1989
100 DIM A(8,8),B(8),F(8)
,X(50),Y(50),Z(50)
105 INPUT "Use Printer <
Y/N>? ";A$
110 IF A$="Y"OR A$="y"TH
EN PN=1 ELSE 125
115 INPUT "Device Code ?
 ";P$
120 OPEN #1,P$,OUTPUT
125 PRINT "Are the funct
ions correct?":PAUSE 2
130 INPUT "Number of Dat
a Points? ";K
140 FOR I=1 TO K
150 A$="X"&STR$(I)&" = "
:INPUT A$;X(I)
160 IF PN<>0 THEN PRINT
#PN,A$;X(I)
170 A$="Y"&STR$(I)&" = "
:INPUT A$;Y(I)
180 IF PN<>0 THEN PRINT
#PN,A$;Y(I)
181 A$="Z"&STR$(I)&" = "
:INPUT A$;Z(I)
183 IF PN<>0 THEN PRINT
#PN,A$;Z(I)
185 PRINT #PN:IF E$<>""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 X$="A"&STR$(I)&" = "
420 PRINT #PN,X$;B(I)
430 IF PN=0 THEN PAUSE
440 NEXT I
450 PRINT #PN
500 INPUT "Display Resid
uals <Y/N>? ";A$
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 A$="N"OR A$="n"TH
EN 610
580 P$="d"&STR$(L)&" = "
590 PRINT #PN,P$;D
600 IF PN=0 THEN PAUSE
610 S1=S1+D:S2=S2+D*D:NE
XT L
620 PRINT #PN
630 PRINT #PN,"Mean = ";
S1/K
640 IF PN=0 THEN PAUSE
650 PRINT #PN
660 PRINT #PN,"S.E. = ";
SQR(S2/(K-N))
670 IF PN=0 THEN PAUSE
680 PRINT #PN
700 INPUT "Edit Input Da
ta <Y/N>? ";E$
710 IF E$="N"OR E$="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>? ";A$
790 IF A$="Y"OR A$="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
```

MAILBAG

"The programming idea in 'Menu and Module' in V13N2P19 was one of the more useful I have seen in a long time. Thanks. ... The approach helped me put about a dozen subroutines together. ..." V.C.

> Subscribers to TI's *Programmable Calculator News* will remember that a similar concept was described on page 3 of the Volume 2, Number 1 issue of that newsletter. I should also give credit to member James Nugent who submitted a menu-driven system in mid-1988.

"I wish to express a belated but sincere thank you for the very personal way you handled my initial contact with you and your organization. Your attention to my individual needs and wants was a warm surprise! B.R.

> We would like to believe that the "personal" response is what distinguishes our club and newsletter. We also try to encourage the establishment of networks of club members who share specific interests. As an example, over the past six months there has been a lively correspondence between Carl Rabe, Larry Leeds, Peter Messer and Don Laughery on the smallest circle problem.

ALTERING THE UNITS OF POLYNOMIAL CONSTANTS ON THE TI-59 - M. Bogart.

The constants in many empirical engineering equations are a function of the units used for the independent and dependent variables. This program will convert the constants from one set of units to another for polynomials up to degree three in the following manner:

Given that $Y = A + Bx + Cx^2 + Dx^3$ for one set of units.

Then the problem is to find another set of coefficients such that

$$Y' = \gamma Y = A' + B'x' + C'(x')^2 + D'(x')^3$$

when $x = \alpha x' + \epsilon$ defines a linear relationship between the independent variables and γ is the conversion factor which changes the units of the dependent variable from the first system of units to the second.

User Instructions:

1. Press A to initialize and see a 1 in the display.

2. Enter the coefficients of the polynomial to be converted (A to D):

   a. Enter the value for A and press R/S. See a 2 in the display.

   b. Enter the value for B and press R/S. See a 3 in the display.

   c. Enter the value for C and press R/S. See a 4 in the display.

   d. Enter the value for D and press R/S. See a 5 in the display.

3. Enter the coefficients which define the relationship between the dependent variables:

   a. Enter the value for α and press R/S. See a 6 in the display.

   b. Enter the value for ϵ and press R/S. See a 7 in the display.

4. Enter the ratio of the dependent variables (γ) and press R/S. See .1111111111 in the display indicating that the input is complete.

5. Press B to calculate and display the coefficients of the converted equation. After about seven seconds the coefficient A' will appear in the display. Press R/S three more times to display the remaining coefficients of the converted equations. Pressing D a fourth time will yield .2222222222 in the display indicating that all of the converted coefficients have been displayed.

6. To calculate using the converted equation enter the independent variable and press C. After about three seconds the dependent variable will appear in the display. If you press R/S again .3333333333 will appear in the display.

7. The program as written is for use without a printer. To print the input values insert a PRT command after the ST* 09 command at steps 151/152. To print the output coefficients change the R/S at step 129 to PRT.

Altering the units of Polynomial Constants on the TI-59 - (cont)

8. A sample problem will demonstrate a change from °F to °C for the independent variable and a change from BTU/lb to joules/gram for the dependent variable for a third degree polynomial relating enthalpy of pure liquid ammonia (heat content above 32 °F) in BTU/lb to temperature in °F as derived from US National Bureau of Standards data. The coefficients of the original polynomial are:

A =	-34.8154	C =	3.18288e-04
B =	1.07658	D =	1.33145e-06

Since $F = 1.8C + 32$, $\alpha = 1.8$ and $\epsilon = 32$, and

1 BTU = 1054.6 joules and 1 lb = 453.6 grams, then

$$1 \frac{\text{BTU}}{\text{lb}} \times \frac{1054.6 \text{ joules}}{1 \text{ BTU}} \times \frac{1 \text{ lb}}{453.6 \text{ gm}} = 2.326 \text{ joules/gram}$$

Enter the seven values as defined in steps 1 through 4 in the user instructions. Press B and read the following coefficients for the converted equation:

A' =	0.0109691034	C' =	3.3619709e-03
B' =	4.609836978	D' =	1.8061428e-05

Enter a temperature in degrees C of -67.72 , press C and read an enthalpy of -302.358 joules/gram.

Editor's Note: The routine at steps 115 through 126 avoids one of the annoying idiosyncrasies of the TI-59 where a number such as 3.33e-10 will be displayed .0000000003 . In essence the routine guarantees that at least eight significant digits of a value will be displayed, but avoids the scientific notation display format if more digits can be displayed in the standard display format.

000	76	LBL	032	95	=	064	95	=	096	24	CE	128	09	09	160	76	LBL
001	17	B'	033	92	RTN	065	65	×	097	33	X²	129	91	R/S	161	10	E'
002	73	RC*	034	76	LBL	066	43	RCL	098	65	×	130	69	DP	162	55	÷
003	09	09	035	12	B	067	06	06	099	43	RCL	131	29	29	163	09	9
004	69	DP	036	04	4	068	85	+	100	04	04	132	97	DSZ	164	95	=
005	39	39	037	42	STO	069	43	RCL	101	65	×	133	00	00	165	18	C'
006	92	RTN	038	09	09	070	02	02	102	43	RCL	134	01	01	166	91	R/S
007	76	LBL	039	43	RCL	071	95	=	103	07	07	135	15	15	167	00	0
008	18	C'	040	06	06	072	65	×	104	95	=	136	02	2	168	91	R/S
009	22	INV	041	16	A'	073	43	RCL	105	42	STO	137	10	E'	169	76	LBL
010	52	EE	042	65	×	074	05	05	106	14	14	138	76	LBL	170	13	C
011	22	INV	043	43	RCL	075	65	×	107	01	1	139	11	A	171	68	NOP
012	58	FIX	044	07	07	076	43	RCL	108	01	1	140	01	1	172	68	NOP
013	92	RTN	045	95	=	077	07	07	109	42	STO	141	42	STO	173	42	STO
014	76	LBL	046	42	STO	078	95	=	110	09	09	142	09	09	174	15	15
015	16	A'	047	11	11	079	42	STO	111	29	CP	143	29	CP	175	01	1
016	65	×	048	43	RCL	080	12	12	112	04	4	144	07	7	176	04	4
017	53	(	049	04	04	081	43	RCL	113	42	STO	145	42	STO	177	42	STO
018	24	CE	050	65	×	082	05	05	114	00	00	146	00	00	178	09	09
019	65	×	051	43	RCL	083	33	X²	115	93	.	147	18	C'	179	43	RCL
020	53	(	052	06	06	084	65	×	116	00	0	148	43	RCL	180	15	15
021	24	CE	053	65	×	085	43	RCL	117	01	1	149	09	09	181	16	A'
022	65	×	054	03	3	086	07	07	118	32	X:T	150	91	R/S	182	68	NOP
023	17	B'	055	85	+	087	65	×	119	18	C'	151	72	ST*	183	68	NOP
024	85	+	056	43	RCL	088	43	RCL	120	73	RC*	152	09	09	184	42	STO
025	17	B'	057	03	03	089	08	08	121	09	09	153	69	DP	185	16	16
026	54	)	058	95	=	090	95	=	122	50	I×I	154	29	29	186	91	R/S
027	85	+	059	42	STO	091	42	STO	123	77	GE	155	97	DSZ	187	03	3
028	17	B'	060	08	08	092	13	13	124	01	01	156	00	00	188	10	E'
029	54	)	061	85	+	093	43	RCL	125	27	27	157	01	01			
030	85	+	062	43	RCL	094	05	05	126	52	EE	158	47	47			
031	17	B'	063	03	03	095	65	×	127	73	RC*	159	01	1			

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

X2 = 40
Y2 = 1.43996
Z2 = 7
```

```
X1 =          21
Y1 =          .201064
Z1 =          2

X2 =          40
Y2 =          1.43996
Z2 =          7
```

```
X1 = 3.141592654
Y1 = 1.23456E+16
Z1 = 2

X2 = .123456
Y2 = .1234567
Z2 = .12345678
```

```
X1 =
 3.141592654
Y1 =
 1.23456E+16
Z1 =                2

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:

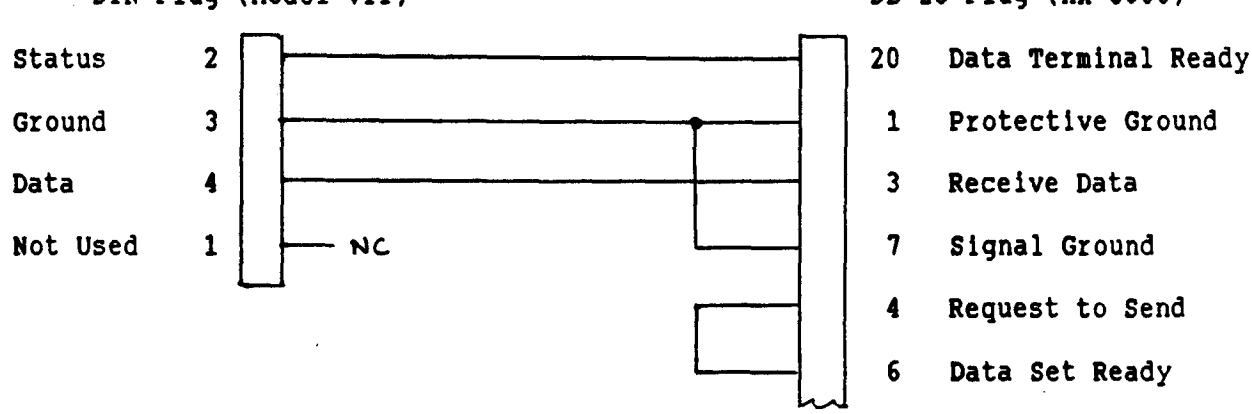
> "... The cartridge houses a single HM6264 8K CMOS RAM and a 3 volt lithium battery. When the cartridge is separated from the console, the battery powers the RAM, retaining data for as long as five years. ..."

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.

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 withe 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?

EVEN MORE ERRATA

Five Function Curve Fit for the TI-95 - V13N2P16. Don Laughery noted that erratic output formats will occur if the calculator is not in the FIX 9 mode. He suggests and Scott Garver agrees that user's should add a FIX 9 command in the initialization sequence, say between the DEC and CFG commands in the first line of the program listing on V13N2P17.

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.

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.

The faster fx-7000G execution time was obtained with the x^2 function. When the Ax^2 function was used the execution time was 76 seconds.

So, for some computers, but not for the TI-74 or CC-40, it does make sense to implement the square function as a product if speed is important.

TI-59 FOR SALE - with a one year old battery pack, charger, ML module, 42 blank magnetic cards, complete manuals, and two Specialty pakettes. One hundred dollars for the lot. Write to Joseph Williford, 895 Rushmeade, Jackson TN 38305. He will include a non-working PC-100C if you want it. Mr. Williford is not a member. You should send a copy of our diagnostics if you want them run before shipment.

PC-100C FOR SALE - It does NOT need the eraser fix for paper to advance properly. $50.00 plus shipping. Send $65.00 to TI PPC NOTES. If shipping is less than $15.00 I will return the remainder.

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 wqs $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 A$="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, KEY$, 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 A_{16} 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.

Printed in the U.S.A. 10055825-4

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 1235_{16}, the breakpoint message is `3635 st 09:`, where `st` 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.