TI PPC NOTES

NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB

P.O. Box 1421, Largo, FL 34294

Volume 8, Number 1 January/February 1983

Welcome back for another year for the TI Programmable Calculator Club, and for its newsletter, TI PPC Notes. For newcomers I should explain that the club is a loosely organized group of people who use programmable calculators produced by Texas Instruments, Inc. The club is not sanctioned by, nor is it supported in any way, by Texas Instruments. Club members live all over the United States, and around the world from Australia to Yugoslavia. Therefore, club activity centers around the newsletter, which will be published six times a year in 1983, with 24 to 30 pages per issue. Although the club and newsletter begin their fourth year in 1983, their origins go back to mid 1976 through a predecessor newsletter, 52 Notes. Maurice Swinnen, who had been editor of TI PPC Notes since its beginning, retired as editor at the end of 1982. As the new editor it seems appropriate to restate the goals of the newsletter.

The newsletter will continue to deal mainly with hand-held programmable calculators manufactured by Texas Instruments. Emphasis will continue to be on the TI-58, the TI-58C and the TI-59. If reader interest is apparent the newsletter will also publish articles on the good old SR-52 and SR-56, on the TI-57, and on the newer hand-held programmables such as the TI-55II and the TI-57II. We will also expand coverage to the hand-held computers such as the CC-40 as they become available.

The highest priority is given to discussion of any conditions where a calculator does not perform as indicated by the applicable instructions, or as might reasonably be expected. The articles in this issue on the statistics quirks of the TI-55II and the INV Σ+ quirk of the TI-58/59 are examples.

Calculator "quirks", that is any unpublished behavioral trick of a programmable, say in response to keyboard or program sequences not defined by the manufacturer, are next on the list. Two of the more useful "quirks"--fast mode and high resolution graphics--have been extensively covered in past issues of TI PPC Notes, and will continue to be covered in the future.

The newsletter publishes lots of programs. How many, and on what subjects, depends upon reader participation. The programs range from games to electronics engineering, finance, statistics, navigation and the like.

The newsletter also publishes book reviews (Page 16 in this issue) and describes accessories which are not manufactured by Texas Instruments. The description of the "beeper" developed by Lester Tibbets is an example in this issue.

As to programming style I will probably bring a different set of prejudices to the newsletter as compared with Maurice. Past subscribers have surely understood that "hell hath no fury like Maurice Swinnen seeing a unnecessary Nop". That is a prejudice I do not share. I do not even object to other "waste" steps such as closing parentheses immediately preceding an equals sign and the like, particularly if such non-essential steps aid in following the program flow. When we don't have the luxury of comment cards then it is often helpful to make the program easy to follow. I do support the elimination of waste steps which will keep a program from spilling over onto an additional magnetic card side. My attitude is quite similar to that expressed by Charles Coe in the article "What is Optimization" in the January/February 1981 issue of PPX Exchange.

There are areas of programming where I wholeheartedly support effort to reduce program steps. The first is in the contests and programming puzzles which have been run frequently in TI PPC Notes in the past and which will continue to be run in the future. For that area the primary idea is to hone programming skills. Another such area is the optimization of utility routines which can be transferred in total to a users program. The print code converter programs are an example.

My strongest prejudice is against publishing programs which will not run as listed. I hate to do inverse coding to try and find out why a program won't run as advertised, won't properly handle some cases, and the like. I hope to avoid that sort of problem by liberal use of reviewers, and by performing an independent key-in and test of published programs from the final copy which goes to the printer. I will, in the process, try to ensure that published programs can be run satisfactorily in accordance with the instructions.

I will give preference to programs which are "friendly to the user". I have always detested waiting for trains, airplanes, elevators or for other machines. Humans should control machines, not the other way around. I particularly object to waiting on computers, or on programmable calculators for that matter. Perhaps that is why I was so fascinated with fast mode for the TI-58/59. Not surprisingly, this first issue reflects my preferences and includes several fast mode programs which illustrate new methods for fast mode entry which minimize operator actions.

Some waiting on computers is not so directly related to execution speed, but rather is a result of the design of input and output routines. The curve-fitting program on pages 15 through 20 of V7N1 provides a convenient example. As each data pair is entered the pair is printed and then the accumulated sums to be used in the curve fitting process are updated. Thus, after each y entry the user must wait about ten seconds before entering the next data pair. The delay is too short to permit the user to do anything else, and long enough to be tedious. I find that objectionable. A better

approach is to build an input file in a way such that the data may be entered as fast as the user can press the appropriate keys. Then the user should be able to list the input data, if he has a printer. The solve mode should read the input file, process the input data as required, and determine the solution. Two programs which illustrate that technique are included in this issue. Now you may object that these programs limit the number of data pairs that may be processed. But how many times do you perform curve fits to more than thirty points anyway? Even that sort of limitation can be circumvented, say by entering a set of data points, accumulating the appropriate sums, and iterating the process until all the points have been entered. That concept was incorporated in my program Polynomial Curve Fit with Errors (PPX 208059).

There is another advantage to entering files of data for curve fitting programs. The program can calculate residuals for each input point without reentering the data, and use the residuals to calculate the standard error of the fit. As George Thomson observed in V7N6P12 the standard error of the fit is a much more useful measure of "goodness of fit" than the correlation coefficient (Op 13) for most engineering and scientific work. Output of the residuals is also helpful in identifying wild data points. Of course one doesn't go to that sort of programming effort for a program one expects to use only a few times. But for "workhorse" programs that one expects to use over and over again, it pays to spend some time in careful design of the input and output routines.

While I am on the subject of my prejudices, I also dislike the endless entering of keystrokes for a program that I would like to use. Panos Galidas recognized that sort of aversion when he offered to provide magnetic cards for his checkbook program at a nominal cost (V7N7/8P28). In a similar vein an organization called CCN Magna-zine Service provides cassettes for the programs in Color Computer News. On a trial basis, and disregarding the horror stories I have heard about magnetic card reader incompatability, I have decided to offer a similar service for programs published in TI PPC Notes. The details are presented elsewhere in this issue.

Finally, I would like to thank club members for their patience while I go through the learning process of getting out a newsletter. I had fully intended to mail the first issue in early February. A bout with that bugaboo or men over fifty, prostate problems, made the meeting of that goal an impossibility.

Palmer O. Hanson, Jr.

IN MEMORIUM - I am sorry to report the death of Paul E. Blair last August due to heart disease. We all knew Paul through contributions such as a 1% resistor program (V5N3P5) and one of the better solutions to Zimmerman's puzzle (V7N6P15). The club has lost a clever programmer and a good friend.

THE CLUB MODULE - The club module didn't get much attention last year, probably as a result of anticipation of the TI-88. However, several members indicated renewed interest in a club module as part of the interest survey which accompanied the subscription for for 1983. Lars Hedlund of the Swedish newsletter Programbiten has continued to work on the idea. You may remember that Lars presented a list of proposed routines in V6N4/5P2. If you are interested in a club module, please let me know. And make your preferences known as to what routines should be included.

ERRATUM - Jorge Valencia of Lima, Peru wrote that he had problems with the Aitken listing for Henrik Klein's integration program (V7N10P13). After a cursory look I couldn't make any sense out of it either. Not having Henrik's address I wrote to Maurice Swinnen for an explanation. Maurice had no explanation for the listing and forwarded the comment to Henrik. Henrik's explanation follows: "...I must certainly have been taking a nap setting up that listing. I have simply mixed up the proper column with a special TI-58 version of 'Romberg'. Fortunately the fault is quite easy to correct, as the 'Romberg' program and the 'Aitken' program are identical from step 147 through 239, so all you need to do to correct the error is to transfer steps 195 through 239 from the 'Romberg' to the 'Aitken' program."

ERRATUM - Robert Prins of De Bilt, Netherlands writes with some corrections to his pie chart program which appeared in V7N10P6. He states "I'm sure nobody will ever get in trouble using the program, but there are a few small errors in it". His changes involve the insertion of four steps between the existing locations 366 and 383. Those changes then require that a number of addresses be changed. Since the changes are defined with respect to the published program it is adviseable to make the changes to the absolute addresses in the program first, and then perform the insertions:

Step 279 from 72 to 76
Step 289 from 72 to 76
Step 346 from 72 to 76
Step 354 from 4 to 5
Step 355 from 7 to 1
Step 389 from 0 to 4
Step 423 from 39 to 43
Step 440 from 4 to 5
Step 441 from 7 to 1
Step 446 from 72 to 76
Step 524 from 93 to 97
Step 528 from 44 to 48
Step 531 from 41 to 45
Step 537 from 44 to 48

Now make the following insertions:

Insert a CP between steps 366 and 367
Insert a INT between steps 373 and 374, between steps 378 and 379, and between steps 382 and 383. Remember that those locations are with respect to the original program. As a cross check, each of the inserted INT commands should occur immediately after a RC*03 sequence.

Finally, change steps 451 and 452 of bank 2.1 from 5 and 99 respectively to 6 and 03, that is, change the absolute address from 599 to 603. Happy Pie-charting!

MIN-MAX SORTER - Henrik Klein. This programming puzzle was proposed by Charlie Williamson as "Place a in the display register (or x register if you prefer) and b in the t register. Devise a routine which will place max(a,b) in the display register and min(a,b) in the t register." An obvious routine would be LBL A GE 006 x⇄t RTN, but for this problem the use of t register comparisons was not permitted. Unfortunately, the description of the problem in V7N1/2P9 omitted the words "in the display register and min(a,b)" making the solution somewhat easier. V7N3P12 and V7N4/5P23 carried solutions to the reduced problem. Henrik provides a twenty-one step solution to the original problem:

LBL A (CE + (x⇄t - x≠t - |x| x x⇄t 1) DIV 2 + x≠t) RTN

which seems to work equally as well as Charlie's 40 step routine which appeared in V7N3P12.

POWERS OF MINUS ONE - Robert Prins. Charlie Williamson also proposed this puzzle in V7N1/2P9 as "Given an integer n in the display, then devise a routine which will return (-1) to the nth power to the display. Solutions appeared in V7N3P12/13 and in V7N4/5P24, but most of the solutions would not work over the full range of input integers. One solution which would work for the full range up to 9,999,999,999,999 was

LBL A |x| - (CE DIV 2) INT x 2 = x 2 - 1 = +/- RTN

which requires twenty steps. Robert's solution which requires only seventeen steps and still provides operation over the full range is

LBL A Rad DIV 10 = INV INT x 10 x π = COS RTN

Can anyone provide an even shorter routine which will work over the full range?

DIGIT REVERSER PUZZLE - Charlie Williamson observes that several of the digit reverser routines (see V7N10P3/4/5) which use the LOG INT sequence will not handle large numbers properly. Charlie provided test cases such as 10E+12, 10E+12 + 1, and 9,999,999,999,981. (Readers of earlier issues have gathered that I am always interested in routines which will handle all cases - Ed) Myer Boland has provided an all purpose routine which is a modification of an earlier program which appeared in 52 Notes. His program does digit reversal and digit summing, and includes the ability to handle decimal points. The program also provides correct reversal for very large numbers:

```
CP R/S
LBL B Stflg 1
LBL A STO 00
LBL D RCL 00 x 10 ) STO 00 INV INT INV x⇄t D RCL 00
LBL π IND DIV 10 INV Ifflg 1 lnx
LBL log ) - INV INT SUM 01 ) INV x=t π ( EXC 01 x 10 ) RST
LBL lnx PRD 01 GTO log
```

Enter any number. Press A to reverse the digits. Press B to sum the digits.

```
000   76 LBL
001   11  A
002   43 RCL
003   00  00
004   32 X:T
005   04  4
006   04  4
007   71 SBR
008   42 STO
009   43 RCL
010   01  01
011   32 X:T
012   04  4
013   05  5
014   71 SBR
015   42 STO
016   98 ADV
017   91 R/S
018   00  0
019   00  0
020   00  0
```

Listing # 1

```
469   76 LBL
470   42 STO
471   69 OP
472   04  04
473   32 X:T
474   69 OP
475   06  06
476   69 OP
477   08  08
478   91 R/S
479   92 RTN
```

Listing # 2

```
468   76 LBL
469   42 STO
470   69 OP
471   04  04
472   32 X:T
473   69 OP
474   06  06
475   69 OP
476   08  08
477   91 R/S
478   92 RTN
479   00  0
```

Listing # 3

HAND-HELD STOP.- Maurice Swinnen. Recently I noticed (for the first time, strangely, after all these years) that Bill Skillman uses a "funny" sequence at the end of a lot of his programs. (see listing # 2.) At first glance, the OP 08 nor the R/S make any sense, if you surmize this thing to be a subroutine.

So, the obvious solution is to call Bill himself and ask questions, which I did. As expected, Bill was genuinely surprised that the great guru himself did not know what this sequence could do. He even told me it came straight out of the 52-Notes, "years ago." I reminded Bill that my advancing age (lately a lot) and the resultant possible early onset of senility could account for this oversight. So, Bill told me it was a "hand-held stop," that is, with the printer attached it will print all, hand-held it will stop to let you copy results by hand.

To test this thing out, let's put both listings # 1 and 2 in user-memory. Press LRN and key in LBL A routine. Then LRN again and GTO 468 (NOT 469 as you might think. You will have trouble getting the RTN in at the very last step) and key in the LBL STO routine. Follow it with a LRN, to get out of LRN mode, of course and press GTO 100 LRN INS LRN. That will shift all the steps of LBL STO one step up and the RTN is where we want it. You could also change the partitioning to 5 OP 17, key in LBL STO and then go back to 6 OP 17 partitioning, but that is more button pushing than the former method.

Now store a couple of different numbers in both R00 and R01 and press A. If your calculator is mounted on the print cradle, both numbers will be printed in sequence, followed by a space.

Record the program on a mag card , banks 1 and 2 and then take the calculator off the printer. Now read in the program, both banks, store a couple of numbers in R00 and R01 and press A. The calculator will now stop with the contents of R00 in the display. Press R/S and the contents of R01 will appear. Hence the name "hand-held stop." Neat huh? But beware:

With either the calculator hand-held or mounted on the print cradle, press from the keyboard GTO 100 LRN DEL LRN. This will shift the LBL STO routine steps one step down, as shown in listing # 3. Now try again by pressing A, followed by R/S. It doesn't work anymore!

Moral of the story? THE RTN HAS TO BE ON THE LAST STEP OF THE PARTITIONING for this trick to work.

Editor's Note: The appropriate references from 52 Notes are V2N10P3 by A. B. Winston and V3N6P5 by Jared Weinberger.

THE INV-Σ+ QUIRK - The article "Hard-wired Functions" by Don O'Grady in the March/April 1981 issue of PPX Exchange cautions users of the TI-58 and TI-59 against difficulty when using INV-Σ+ with the data entered using scientific notation. The problem results from a +/- command which is the first instruction encountered in the firmware when the INV-Σ+ function is called. The instruction was intended to change the sign of the entered value, or the sign of the mantissa when in scientific notation. However, if the INV-Σ+ function is called immediately after completion of an entry in scientific notation then the +/- changes the sign of the exponent not the sign of the mantissa. To observe the problem, initialize the calculator for statistics entry with the sequence 2nd-Pgm-01-SBR-CLR. Enter 1 EE 3 (1000) and press 2nd-Σ+. Enter 1 EE 3 again, but this time press INV-2nd-Σ+. Recall the sum of the two entries, which should be zero, with a RCL-01. Instead you will see 1.000001 03 in the display, which is the sum of 1000 and 0.001.

The problem occurs only with the "live entry" state which results from entry of a number in scientific mode. It does not occur with the "live entry" state established by entering the sequence EE-INV-EE with a dead entry state in the display. Even in scientific mode, the EE-INV-EE sequence produces a "live entry" state for the sign of the mantissa, not for the sign of the exponent. An easy way to return sign control to the mantissa, not the exponent, after entry of a number in scientific notation is to follow the entry with a decimal point.

This quirk does not occur with the TI-57, the TI-55-II, or even with the TI-35.

AN ALARM FOR THE TI-58/59 - If you use your calculator for lengthy, unattended calculations then you have probably wished for an audible alarm to indicate that calculations were complete. With the PC-100 printer attached the sound of a printout could be used as a cue, but from just a few feet away you would probably miss the cue. Member Lester Tibbetts has devised a better solution. His alarm is a 1¼" x 3" x 3/4" black box which contains a photocell, battery, alarm and associated circuitry. You can place the device with the photocell down and over the right side of the display. Even a single character such as the initial zero will cause the alarm to sound. When a program is running the sound will stop. When program execution stops the alarm sounds again. If you end your program with a deliberate keying error to cause a flashing display, for example the sequence times, equals, R/S the alarm becomes a beeper, pulsating as the result is flashed. Better still, place the device with the photocell to the left where it cannot see the initial zero. Store or print your final result and end the program with the sequence CLR-1/X-R/S to flash an overflow. The alarm"s loud, insistent beep will then occur only at program completion. The use of coded pauses, say during subroutines, will let you keep tabs on program execution while you are busy with other things. The net result is much like having a faster calculator since time freedom is provided for the operator. You can obtain the device, or further information, from Lester Tibbets, RD-1, Box 294A, Emporium, PA 15834. The price is $19.95 plus $1.00 shipping for both domestic and foreign mailing, or $2.00 shipping if overseas airmail is desired. Lester provides a money back satisfaction guarantee. Nine volt battery not included.

A COMPLETE INDEX FOR TI PPC NOTES - Robert Fruit - Again this year I am offering an index for TI PPC Notes. I found last year to be a never ending experience with Murphy's Law. First there was the copying problem with last year's index. Then, few people took me up on the index offer. The final blow this year was getting kicked off the computer I had been using. I did find a micro computer I could use and there were many problems in converting the data, and in writing a new program. After all that, there is good news. I redesigned the index, and it's going to cost less than last year.

Let me tell you about the index. It is an index by subject (including author). There are a number of subject categories and under each is an alphabetical list of the applicable titles. This approach leads to each title appearing several times in the subject index, once for each subject that corresponds to it. For most subject categories it is an easy matter to find the articles that relate to your interest. For instance if you remembered that someone had a maze building program, how long would it take you to find the article without this index? This index shows 33 titles under the category GAMES and from there it can be narrowed down to two titles. How long do you think it would take you now? I mention the example to emphasize to you that if you use your TI-58/59 calculator as much as I do, there is a continuing need to check back issues of the TI PPC Notes to keep yourself from reinventing the wheel. The subject categories are:

April Fools	Firmware	Plotting
Articles Vol 5	Games	Programs
Articles Vol 6	Graphics Mode	Sources of Information
Articles Vol 7	Help Requests	Speed Programs
Authors	HIR Codes	SR-52
Bench Marks	Mathematics	SR-56
Brain Teasers	Modules	TI-57
Club Challenges	Newcomers Corner	TI-59
Club Module	Other Clubs	TI-88
Diagnostics	Patent Information	Utility Programs
Fast Mode	Pictures	

After making the point above about how short most of the categories are, I must confess that there are a few big lists; ARTICLES, AUTHORS and PROGRAMS (a list of every title that has a program associated with it). These subject categories are really catch all ones that list every related title. Authors is the largest with 501 titles. The strongest feature of this index is that it cross-references every article. There are not many articles with errors in them, but when there are, the cross-reference makes it easy to ascertain why you are having problems. Remember the Check Sum program? Even better, check the cross-references beforehand and avoid problems. If there is anything that sets my index apart from the one that comes with the TI PPC Notes, it is the cross-references.

Anyone who wants an index should write to Robert Fruit, 100 Fuller Road, Hinsdale, IL 60521. The 14 page index comes on 8½ x 11 inch paper copied on both sides. It will be sent first class mail. The cost is $5.00 (U.S. currency). Make checks or money orders payable to Robert Fruit. See a full size copy of a page on the facing page.

TIPPC NOTES SUBJECT INDEX PAGE 8

MORE ON INDEXES - If you wonder why I use more than two pages on the subject of indexes, it is because I think that we need a really good one. Bob's index is the best I have seen. He failed to mention what I think is the most important characteristic, namely that it is a multi-year index. What we really need is a multi-year,multi-periodical index, say one which would include the articles from PPX Exchange, from 52 Notes, and from TI PPC Notes. Perhaps if we support Bob on his index this year we can induce him to expand the coverage next year.

While we are on the subject of indexes, Maurice Swinnen compiled an index of 52 Notes in the same fashion as the indexes for TI PPC Notes; that is, there is an index by article and by author for each of the years from 1976 through 1979. Maurice has given me permission to offer it to club members for a nominal fee. I can provide this six page index for $2.00 to cover the costs of printing and mailing. You may send cash, stamps, or what have you? I, and my banker as well, would prefer not to be deluged with two dollar checks. But if you must use a check, then so be it, and make it payable to PPC Publications.

PROGRAMMING WORKBOOKS - Maurice Swinnen has available about 500 TI-59 workbooks. These books were used in TI-59 seminars to allow students to do exercises in class. They can also be used for self-study, as they contain nice exercises and references to specific pages in the manual Personal Programming. This book is normally listed for $4.95. Maurice can make the books available ABSOLUTELY FREE for the asking. You only pay the postage of $1.00 per book. The books will be mailed at BOOK RATE to you, as the 100+ page book weighs over 13 ounces. A real bargain for a beginner on the TI-58 or TI-59. Send your dollar and a mailing label to Maurice Swinnen, 9213 Lanham Severn Road, Lanham, MD 20706. Order as many books as you wish.

FAST MODE SFF FOR 11 OR 12 DIGITS - Robert Prins - Patrick Acosta's modulo 210 Speedy Factor Finder (see V6N4/5P13 and V7N4/5P4) could use fast mode for integers up to ten digits, but required normal mode for integers of 11 or 12 digits. That limitation is a result of the characteristics of the Pgm 02 SBR 239 09 method of fast mode entry, namely that in general algebra from the keyboard is not permitted while in fast mode. Robert Prins of the Netherlands has found a way to use the first five steps of Patrick's program (not previously used) to provide a fast mode capability for Patrick's program for 11 or 12 digits. The only change to Patrick's program, starting at location 000 is:

PAU R/S GTO 00 27 Pgm 02 SBR 02 39 .....

With this modification you initialize fast mode with SBR 005 rather than RST R/S, re-enter all four cards in the same manner previously defined, and test integers up to ten digits using the same procedures. For 11 or 12 digit input integers you use a revised procedure. For example, for a 12 digit number enter n.nnnnnnn EE 11 INV EE SBR 0 (only one zero please) + nnnn = . When you press = the program starts, and subsequent operation is as with Patrick's original program. The only deficiency with this method is that the printout of the input integer will not show all 11 or 12 digits. However, there is a modulo 210 SFF program using the h12 method for fast mode entry which will indicate all the digits of the input integer.

LET ME COUNT THE WAYS - Myer Boland proposed this puzzle in V7N7/8P27 as "Press A and the calculator flashes 1,2,3, 4,5 ... ad infinitum. No numerical keys are to be used with the exception of the PI key. That one may be used." Solutions appeared in V7N10P2. Myer comments that some of the solutions as published did not obey the rules; for example, the editors solution

OP 10 + PAU LBL A PI RST

involves the use of the numerical keys to enter the OP 10. Myer notes that the rule against the use of the numerical keys may be circumvented by using the key-in sequence

OP E' + PAU LBL A PI RST

Similarly, one of Bjorn Gustavsson's solutions was

OP 20 RCL 00 PAU RST LBL A Cms RST

where the numerical keys would normally be used to enter the OP 20 and the RCL 00. The rule against using the numerical keys may be circumvented with the key-in sequence

OP 2nd CLR RCL SST PAU RST LBL A Cms RST

In an extension of that spirit Kevin Pray proposed a solution where the key-in sequence starting from the turn-on condition would be

LRN SST COS + PAU LBL A RST LRN

which enters a program

0 COS + PAU LBL A RST

The solution obviously depends on the zero at location 000, but no numerical key was used to enter the program. The use of unusual key-in sequences was illustrated in problem 71 of the "TI-59 Test" in the May/June issue of PPX Exchange. The problem asked "How could the program instructions DSZ 10 SIN be keyed in?" The correct solution was DSZ E' SIN. The general rule is that when the first part of a multi-part command sequence is followed by any keystroke other than one of the numerical keys 0 through 9, then the merging process is terminated, and the subsequent keystrokes are entered into the program as if the merged code sequence had not been initiated. John Szablya described such techniques for editing of transfer addresses in the November 1979 issue of PPX Exchange, but he cautioned that the procedure may be confusing (and dangerous) for beginning programmers. The technique is certainly helpful in circumventing the rules for solving puzzles.

A CHANCE TO PUBLISH - Member Clifford Lieberman is an Associate Editor of COMPUTERS IN BIOLOGY AND MEDICINE. The Journal is interested in areas such as (1) Analysis of Biomedical Systems; Solutions of Equations; (2) Synthesis of Biomedical Systems; Simulations; (3) Special Medical Data Processing Methods; (4) Special Purpose Computers and Clinical Data Processing for Real Time, Clinical and Experimental Use; and (5) Medical Diagnosis and Medical Record Processing. As with most journals there are formal rules for submission of manuscripts. If you believe you have a program which may be of interest to the journal write to: Clifford Lieberman, D.D.S., 1020 N. Quincy St., STE 1006, Arlington, Virginia 22201. To obtain the Aims and Scope of the journal, write to COMPUTERS IN BIOLOGY AND MEDICINE, Georgetown University Medical Center, 3900 Reservoir RD. NW, Washington, DC 20007

A CALL FOR PROGRAMS USING OTHER MODULES - One of my reasons for accepting the position as editor of TI PPC Notes was the opportunity to expand the coverage into areas not well covered previously. One area that has always intrigued me is the use of the other memory modules. To see if there was any potential there I asked Maurice to include the interest survey as part of the subscription form. The results of the first 250 subscriptions were:

Module Number	Module Name	Number Interested
2	Applied Statistics	87
3	Real Estate and Investment	42
4	Surveying	23
5	Navigation	30
6	Aviation	18
7	Leisure Library	74
8	Securities Analysis	49
9	Business Decisions	55
10	Math/Utilities	141
11	Electrical Engineering	49
12	Agriculture	10
13	RPN Simulator	62

With that amount of interest it remains a mystery to me why there has been so little mention of the other modules in 52 Notes, PPX Exchange or TI PPC Notes. For example, reference to Bob Fruit's index shows only seven citations for the twelve modules. The only articles were on the M/U and RPN modules. In the case of TI PPC Notes that lack of coverage isn't because Maurice somehow elected not to publish such material. I have made a once through review of his unpublished submissions, and came up empty-handed.

There are some really useful programs in those other modules. I am confident that the readers have some valuable interfacing programs to share, if only they will. In this issue I have included some programs illustrating the kind of material that I think should be available.

INV LIST PRT ALL - FAST MODE (V5N9/10P15) - Jorge Valencia of Lima, Peru writes: Funny that two years after that issue no indication of an error has been made. He gets the program to work only by performing step 7 before step 6. Review of the program shows that the problem is the CLR at location 025. Replace the CLR with a Pause (code 66) and operation in accordance with the instructions can be obtained.

BRAIN TEASER - This one is from TISOFT, Volume 4, Number 2. Start with a zero in the display and try to obtain 0.1 in the display. You cannot use the number keys, but any other key is fair game. One solution is INV lnx DIV INV log = which is six steps counting the INV's. Can you do the problem in fewer steps? Can you do the problem without disturbing any pending operations? This problem is relatively easy. There are more difficult challenges in subsequent pages.

NEWCOMER'S CORNER. Maurice E.T. Swinnen. People often ask me, especially at the TI seminars: "How do you start writing a program?" My answer usually is: "You first have to formulate in your mind what exactly your program has to do. The rest comes easy."

Let us go through this "thinking process" and actually build a program, to see what's involved. Suppose we want to write a program to keep our checkbook straight. We have to provide a means to:
1. Accumulate all deposits. Each deposit increases the total amount of money we have in the bank.
2. Subtract the amount of each check we write. Each check diminishes the total amount we have in the bank.
3. Show the total amount we have in the bank.
4. Record the program and the total amount on a magnetic card, such that we can use it again, next time we either make deposits or withdrawels.

Additionally, we agree to abide by the following conditions and rules:
1. The program will be short, as very few things have to be done. So, we have room to add bells and whistles.
2. We are not going to write an economy version, for calculator only. Instead we are going to print all deposits, withdrawals and totals, with the necessary descriptors, as it should be.
3. The recording of the program and the total, at the end of each session, will be as automatic as humanly possible. One single key punch allowed.
4. Totals should be shown either a) on demand, by one single key punch or b) automatic, following a deposit or a withdrawal.
5. Recording at the end of each session should be on one side of a mag card only. Requiring to record two card sides, and later reading in two sides, for such a short and simple program is a "crime" and should be punished by the full force of the law.

For the time being, that is all I can think of. Maybe in the course of the actual writing of the program some "niceties" might present themselves.

If we want to record everything, program AND register which contains the total on one side of a mag card, then both have to lie within the same bank. If we use the first 160 steps for the program, and we partition to 10 OP 17 (automatically, please, at initialization) we have at our disposal registers 90 through 99 to keep the "total" in. (see manual VII-1) This entails that when we place the calculator in such a mode that we only have to slide the card in the slot to automatically record the program and that one register (say 90), it better be in 6 OP 17 partitioning, also to be programmed, as opposed to keyed in from the keyboard. That will make our life easy when we later want to read in the program. We will only have to turn on the calculator and slide the card into the slot, no key punch required.

All the above is what I mean by "formulate in our minds" what the program has to and will do. Even if it takes a little more time to sit down and think about it, it will save you considerable time later writing the program. It will avoid time-comsuming mistakes, sometimes requiring complete rewriting of large portions of code.

So, the first thing we should write is a routine to partition automatically to 10 OP 17, such that we have register 90 available as the SUM-register. That routine will become our initialization, to be executed each time after we read in the program. It is simple enough: LBL A 10 OP 17 CLR R/S

We could now use LBL B to define the withdrawal routine. But I have found that it is so much handier to do that through key R/S. We usually have many more withdrawals than we have deposits. As we have already initialized by pressing key A, which ends in an R/S, we could simply tack on at this spot the withdrawal routine, which will, of necessity, have to end in a GTO A. That way we will always place the program pointer at the same R/S step at the end of the A-routine. Of course, a requirement will also be that the deposit routine, still to be written, will also have to end in a GTO A. And so will the "total" routine, and any other routine we might devise, and which might be used alternately with the withdrawal routine. So we write after the R/S of the A-segment: STO 00 INV SUM 90 15 23 17 34 OP 04 FIX 2 RCL 00 OP 06 INV FIX GTO A

Try this out. It will subtract from register 90 the amount of each check you enter and follow by an R/S press. It

will print that amount along with the descriptor CHEQ. Nothing very fancy, I admit, just plain, but practical programming.

We now do the same for the deposit routine: LBL D (why not D? D for Deposit will be easy to remember) STO 01 SUM 90 16 33 01 36 OP 04 RCL 01 FIX 2 OP 06 INV FIX GTO A. Here each amount entered and followed by a D press will be added to the amount already in register 90. The entered amount will be printed with the descriptor DPOS. Again, simple and straightforward.

At once we see a candidate for a subroutine: OP 04 ..... FIX 2 OP 06 INV FIX. We only need to put X:T where the "....." are and insert in both the deposit and the withdrawal routine another X:T following STO 00 or STO 01. We could, admittedly, also use only register 00 and thus put RCL 00 where the "....." are. No gain in number of steps, though, as both require two extra steps.

We could now write the "total" routine as: LBL E 37 01 37 OP 04 RCL 90 FIX 2 OP 06 INV FIX GTO A. Again, we are reminded that a subroutine would save us several steps: LBL A' OP 04 X:T FIX 2 OP 06 INV FIX RTN, which we place at the head of the program, of course. The next thing we attack is the automatic recording routine, to be used at the end of each session, when we are finished with all deposits and withdrawels. The first order of business is to partition to 6 OP 17, for reasons explained above. Then we want to print the words RECORD BANK 1, or something to that effect. We know how to do that. The only precaution I would recommend, and which I sometimes see omitted with disastrous results in some programs, is an OP 00 at the beginning and at the end of such a single printing. OP 00 will clear completely all print registers, OP 01 through OP 04. If you don't use OP 00 you might suddenly find some unwanted print-out in other parts of your program and you might spend some time tracking down where it came from.

Now, we add 1 WRT, at which the display will go blank and wait for the user to slide the card in the slot. That should not necessarily be the end of that routine. You might still want to reassure the user that this is the end. To this end you print "END", then put a few ADV's, so that the paper shoots out of the printer, making it easy to tear it off and throw it in the waste basket!

So far, all this should work. But we promised bells and whistles. When you are writing those monthly checks for the mortgage, the electricty, the gas, the telephone, the car payment and depositing the pay check, you want to write down the amounts in your checkbook, along with entering them in your calculator to let IT do all the figuring. Now, there are two schools of thought: one just enters deposit and amounts of each check and at the very end presses E to get the final total. The other one wants to know all the subtotals. That means the total has to be printed (automatically please. No extra key punch allowed) after each change, be it deposit or withdrawal. Wouldn't it be nice if we could satisfy both sides? Suppose we provided one key to be pressed to set a flag (and reset it, if we press a second time) to later check in the program if a print-out of the total is required or not. Of course, that routine should also end in GTO A, as we don't know when the user is going to press the corresponding user-defined key. And any routine that can be used alternately with the withdrawal routine has to end up at the R/S at the end of the A-routine, so that the calculator is ready to enter a withdrawal by pressing R/S. So we write: LBL C INV IFF 1 E' INV LBL E' STF 1 GTO A. Try it out: the first time you press C, and suppose flag 1 is not set, the program will transfer to LBL E' and so set the flag. The second time you press C the test "INV IFF 1" will not be true and so the program will fall through to INV LBL E' STF 1. This means that the flag will be reset, as the two keystrokes "LBL E'" will simply be ignored and only INV STF 1 will be executed. The third time you press C it will be a repetition of the first case, and so on.

It will be now rather simple to use a test "IFF 1" at the end of both deposit and withdrawal routines to see if the user wants "total" print-out or not. We now simply give the "total" routine another LBL (D' here) and end it in a RTN, so that it can become a subroutine.

And LBL E, the former "total" routine? Well, it simply becomes LBL E D' GTO A. This allows you to selectively

Newcomers corner, Swinnen, cont.2

call the "total" routine from the keyboard, by pressing E, or to tack it automatically to the end of both the deposit and the withdrawal routines,by pressing key C, which sets flag 1.

The example given is self-explanatory, I think. The top print-out is without flag 1 set. The bottom one was obtained after key C was pressed, and thus flag 1 set. Both were started by pressing A to initialize after which all registers were cleared by pressing CMs. Then two deposits, one of $ 5200, another one of $ 1200 were made. (If it is not my money I am always very generous) Then five checks for various amounts were written. You can see that both methods yield the same result.

I don't claim anything world shaking with this program, just a nice exercise in simple, bread-and-butter programming. And the result is a program that is handy to use and reduces the number of mistakes in keeping your checkbook up to date. I use it and I find it much handier than to have to fire up the computer, as some of my friends claim they do. You just don't use a sledge hammer to kill a flee, do you now?

USER INSTRUCTIONS: a) First time only:

1. Key in the program in 6 OP 17 partitioning.
2. Press A to initialize. Display will show a zero.
3. Press CMs to clear any registers. Then press E. A total "0.00 TOT" should be printed.
4. Now record the program on a mag card as follows: Press B. See "RECORD BANK 1" printed. Slide the mag card into the slot. The card will be pulled through automatically. See "END" printed.
   Your card now contains the program and a balance of zero in register 90.

b) Normal user instructions:

1. Turn on calculator and printer and slide the mag card into the slot. A steady "1" should appear in the display. If the display blinks, press CLR and try again. If it still refuses your program, rub the card, black side down, on the side of your pants or skirt, as the case may be, to clean off the oils left by your fingers. Now press CLR and try again. If it still refuses call Lubbock or throw that caclculator in the trash.
2. Press A to initialize. A steady 0 should be in the display.
3. Enter either a deposit and press D or a withdrawal and press R/S. They may be intermixed at will. Repeat as many times as needed.
4. If you want a total (balance) printed automatically after each entry in 3, above, press C once. To undo at any time, press C again, etc.
5. If at any time you want a total, manually, press E.
6. When you are finished, record the program and the balance (total) by pressing B. See "RECORD BANK 1" printed. Slide the card in the slot. See "END" printed. Your card now contains the program and the total (balance), for future use. For a new session go back to 1 of the normal user instructions.

DISPLAYING THIRTEEN DIGITS - George Wm. Thomson, Detroit, Michigan.

R. W. Snyder included in his program for reversing digits (V7N10P4) a method for displaying integers with over ten digits. For thirteen digits: TRACE DIV 1000 = INV INT TRACE. Unfortunately this sequence may give errors in the tenth significant digit due to the effect of rounding to the display. For example, 4953148324 624 gives as the output 4953148325 624 with a digit error in the tenth place. The correct answer is obtained with the alternative sequence: TRACE DIV 1000 - INT = TRACE. The format is the same with the first ten digits shown as an integer and the others as thousandths. Once again we are reminded of the extreme care needed in stretching calculations to the limit of the TI-59 through use of the guard digits.

BOOK REVIEW.

PROBLEM SOLVING WITH THE PROGRAMMABLE CALCULATOR, David L. Dunlop & Thomas F. Sigmund. Prentice-Hall, Englewood Cliffs NJ 07632, 1982, 227 pages, paper back, $ 10.95.

As if two new books by Peter Zehna on the use of the TI-59 in Statistics and Probability weren't enough, Prentice-Hall has outdone itself and given us a third book, this one on general problem solving with the TI-59. This is the kind of book that makes me excited. Both authors teach at the University of Pittsburgh at Johnstown. I really would love to meet them. They are able to make the use of a programmable calculator fun not only in the games they explain to the most minute detail, but also in their more serious applications in math, business and science in general.

Under the general heading of CHALLENGES, they have:

1. Number curiosities.
2. Figurate numbers.
3. Prime and composite numbers.
4. Diophantine equations.
5. Probability.
6. Maximum and minimum.
7. Geometry.

And under the general heading of SIMULATIONS, they have:

1. Probability.
2. Biological science.
3. Physical science.
4. Chemistry.
5. Automobile.
6. General.

And lastly there are games such as FOOTBALL, RACE CAR DRIVER, MISTERY NUMBER and CALCULASER, exciting all of them and well thought out. They are real fun to play and will hold your attention from the beginning to the end.

If you think the SOURCEBOOK from TI is good, this book is excellent.

If I ever meet these two gentlemen personally, though, I will let them in on the secret that two closing parenthesis are not needed BEFORE an equal sign. They must not have read the TI PPC NOTES lately and don't known yet about my pet peeve. (Maurice E.T. Swinnen.)

Newcomers corner, Swinnen, cont.3

```
     0.00    TOT
  5200.00    DPOS
  1200.00    DPOS
  6400.00    TOT
    23.45    CHEQ
    45.56    CHEQ
    78.89    CHEQ
    10.10    CHEQ
    13.59    CHEQ
  6228.41    TOT

     0.00    TOT
  5200.00    DPOS
  5200.00    TOT
  1200.00    DPOS
  6400.00    TOT
    23.45    CHEQ
  6376.55    TOT
    45.56    CHEQ
  6330.99    TOT
    78.89    CHEQ
  6252.10    TOT
    10.10    CHEQ
  6242.00    TOT
    13.59    CHEQ
  6228.41    TOT

RECORD BANK 1

    END
```

```
000 76 LBL
001 16 A'
002 69 DP
003 04  04
004 32 X:T
005 58 FIX
006 02  02
007 69 DP
008 06  06
009 22 INV
010 58 FIX
011 92 RTN
012 76 LBL
013 19 D'
014 43 RCL
015 90  90
016 32 X:T
017 03  3
018 07  7
019 00  0
020 01  1
021 03  3
022 07  7
023 16 A'
024 92 RTN
025 76 LBL
026 11  A
027 01  1
028 00  0
029 69 DP
030 17  17
031 25 CLR
032 91 R/S
033 42 STD
034 00  00
035 22 INV
036 44 SUM
037 90  90
038 32 X:T
039 01  1
```

```
040 05  5
041 02  2
042 03  3
043 01  1
044 07  7
045 03  3
046 04  4
047 16 A'
048 22 INV
049 87 IFF
050 01  01
051 11  A
052 19 D'
053 61 GTD
054 11  A
055 76 LBL
056 14  D
057 42 STD
058 01  01
059 44 SUM
060 90  90
061 32 X:T
062 01  1
063 06  6
064 03  3
065 03  3
066 00  0
067 01  1
068 03  3
069 06  6
070 16 A'
071 22 INV
072 87 IFF
073 01  01
074 11  A
075 19 D'
076 61 GTD
077 11  A
078 76 LBL
079 15  E
```

```
080 19 D'
081 61 GTD
082 11  A
083 76 LBL
084 12  B
085 06  6
086 69 DP
087 17  17
088 98 ADV
089 69 DP
090 00  00
091 03  3
092 05  5
093 01  1
094 07  7
095 01  1
096 05  5
097 69 DP
098 01  01
099 03  3
100 02  2
101 03  3
102 05  5
103 01  1
104 06  6
105 00  0
106 00  0
107 01  1
108 04  4
109 69 DP
110 02  02
111 01  1
112 03  3
113 02  2
114 09  9
115 02  2
116 06  6
117 00  0
118 00  0
119 00  0
```

```
120 02  2
121 69 DP
122 03  03
123 69 DP
124 05  05
125 69 DP
126 00  00
127 01  1
128 96 WRT
129 98 ADV
130 01  1
131 07  7
132 02  2
133 09  9
134 01  1
135 06  6
136 69 DP
137 02  02
138 69 DP
139 05  05
140 25 CLR
141 98 ADV
142 98 ADV
143 98 ADV
144 98 ADV
145 91 R/S
146 76 LBL
147 13  C
148 22 INV
149 87 IFF
150 01  01
151 10 E'
152 22 INV
153 76 LBL
154 10 E'
155 86 STF
156 01  01
157 61 GTD
158 11  A
159 00  0
```

h24 REVISITED - In V6N9/10P15 Patrick Acosta wrote: "h24 is the strangest hex-code of all. Create it at step 000 by pressing INS once and return to user memory and write 7 program steps in 001 to 007 so that you have: 000: h24 AB CD EF GH IJ KL MN If you then press RST R/S the h24 will change the order in which the keycodes are interpreted to: 000: DA FC HE JG LI NK 0L 00 . This gives the equivalent of 15 program steps packed into 8 steps. SBR 001 will give a 7 step program, and SBR 000 will give an entirely different 8 step program. All the programmer has to do is to write his program so that either way the 8 keycodes are interpreted will be useful. An IQ of 160 and nerves of steel are recommended."

The editor doesn't pretend to have an IQ of 160 or nerves of steel. Neverless, I made several determined tries to write some code which would do something useful both ways, and run it with the h24 technique, but with no success when using my TI-59. During the recovery period from my recent illness I tried again. But this time I had a TI-58C available. Voila!!! Instant success, almost. After some experimenting I concluded that there was a misprint in V6N9/10P15. My tests show that the reverse sequence should be: 000: DA FC HE JG LI NK 0M 00 where the difference is in the next to the last term being 0M not 0L. My sample test sequence is:

h24	99	28	34	34	34	54	98	95	91
	Prt	log	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	)	Adv	=	R/S

As expected, if one starts with 100 in the display, the result will be 1.090507733 in the display, which when squared three times gives an answer of 2. Reversing, per my revised rules gives:

h24	89	42	43	43	43	85	09	00	95	91
	π	STO	43	RCL	43	+	9	0	=	R/S

Then if one uses SBR 000 instead of SBR 001 one gets 93.14159265 as the result. If one changes the ADV (code 98) at location 007 to a Nop (code 68) which changes M from 9 to 6, then the answer from SBR 000 becomes 63.141459265. The other supporting evidence for the use of 0M not 0L is that it would be more consistent with the pattern used to generate the rest of the reverse code. I have written to Patrick Acosta for confirmation of all of this. With the above sequence one will get π in R43 if the partitioning is set properly, say 5 Op 17. If not, then the final answer is the same, but with a flashing display, and π doesn't go to R43. All as expected. Rather than using SBR 000 one can use RST R/S and get the same effects.

For a sample program that does something other than demonstrate the 0M versus 0L problem I chose the 1 EXC 2, 2 EXC 1 Routine from V5N3P12 in which the program is required to change a "1" in the display into a "2", and vice versa. The following program, called by SBR 010 will perform the required functions:

000: h24 1 RTN 2 2 2 2 2 2 R/S STO 01 Op 31 GT* 01

You cannot insert a LBL A before the STO 01 since the h24, or any other hexadecimal code at location 000, prevents the search mechanism from recognizing labels. There are other idiosyncrasies to investigate; for example, a RTN generated by the reverse sequence does not seem to be recognized. An h24 acts entirely different in a 59! Who will write a really useful h24 routine for the TI-58C?

CHALLENGE - Ralph W. Snyder, Indianapolis, Indiana. (Editor's Note: Ralph wants club members to work this problem so much that he has posted a $25.00 reward for the best solution. I took to heart Maurice's comment in V7N10P3 about provoking Ralph's "holy wrath" by making errors in transcription--so the challenge below is copied directly from Ralph's submission.)

Here's a chance for 59ers to work their algebraic know-how in melding algebraic manipulations with programming art to optimize program steps and execution time, i.e., to get there fastest with the least.

Newton's tangent h-correction method in root-finding as applied to annuity-type functions may be stated as: (1) $h = -f(i)/f'(i)$. There is also a secant method which doesn't require the formal derivative f'(i) and which is written: (2) $h = d/[1 - f(i+d)/f(i)]$, where d is a small increment, say 10-5. The secant method can also be put into the form of the method of false position or "regula falsi" as: (3) $h = df(i)/[f(i) - f(i+d)]$. The secant/false position method for a correction is quite comparable to the tangent method in results, but it isn't as efficient, i.e., not as fast.

In determining an unknown rate i, a good, mathematically sound, formula for an initial guess is vital for minimizing the number of iterations of the correction formula. Here is one from a series expansion of $i/(1-v^n)$, the reciprocal of the annuity function, (used in my PPX #198054, "Solving All Types of Annuities by Bond Formula"), and beside it is a secant h-correction based on equation (2):

$$(4)\ i = \frac{\frac{C}{P-F} - \frac{1}{n}}{\frac{F}{P-F} - \frac{C}{P-F} + \frac{1}{2n} + \frac{1}{2} + \frac{i_0 n}{12}}, \text{ where } i_0 = \frac{\frac{C}{P-F} - \frac{1}{n}}{\frac{F}{P-F} - \frac{C}{P-F} + \frac{1}{2n} + \frac{1}{2}}; \quad (5)\ h = \frac{d}{1 - \frac{\frac{i+d}{v^n_{(i+d)} - 1} - \frac{F-C}{P-F}(i+d) + \frac{C}{P-F}}{\frac{i}{v^n_{(i)} - 1} - \frac{F-C}{P-F} i + \frac{C}{P-F}}}.$$

The challenge is to produce the h of formula (5) by recasting it, to meet the goal of optimization set out above. Hint: Consider putting formula (5) in the form of equation (3).

Here is a list of symbols used, and data for two problems (both are "ordinary annuity" types, hence do not require the setting of Flag 4, which is for "annuity due" types; accordingly (F-C)/(P-F) = F/(P-F)):

	Sto.in Reg.no.	Problem (A) Bond	Problem (B) Sink Fd (Ord An)		Sto.in Reg.no.	Problem (A) Bond	Problem (B) Sink Fd (Ord An)
C = periodic pmt.	03	450	-450	P-F	08	679.51631725	-13400.1353591
F = future val.	05	10000	13400.1353591	F/(P-F)	06	14.71635007	-1
P = present val.	04	10679.51631725	0	True i		.04	.04
n = no. periods	01	20	20	1st i	computed	.0399937152	.0393353142
1/n	16	.05	.05	h_1	"	.000006248	.0006628510
1/2n + 1/2	10	.525	.525	h_2	"		.0000018348
n/12	00	1.666666667	1.666666667	i (dec.)	"	.04	.0400000001

My first approach was to set up a subroutine which was used to compute both f(i) and f(i+d). Although the number of steps seemed the fewest, the double use of the subroutine resulted in 76 effective steps for h, which compares with 43 steps by my Newton's method h-correction. Note: Do not expect any arrangements of the secant/false position formula to match the Newton, but you should do better than 76 steps.

Here is a listing of the steps for 1st i from my PPX program, plus the iteration control and increment d. The h-correction should of course start with step 047. Also shown is the iteration-test and closing routine, which should be tacked on at the end of your h routine.

000	43 RCL	012	32 X↕T	024	85 +	036	95 = 1st i		99 PRT
001	03 03 C	013	53 (	025	42 STO	037	99 PRT		44 SUM
002	55 ÷	014	43 RCL	026	09 09 $\frac{F-C}{P-F}$	038	42 STO		12 12
003	43 RCL	015	06 06 $\frac{F}{P-F}$	027	43 RCL	039	12 12 i		50 I×I
004	08 08 P-F	016	22 INV	028	10 10 $\frac{1}{2n}+\frac{1}{2}$	040	05 5		77 GE
005	75 -	017	87 IFF	029	85 +	041	94 +/-		00 00
006	42 STO	018	04 04	030	35 1/X	042	22 INV		47 47
007	11 11 $\frac{C}{P-F}$	019	00 00	031	65 ×	043	28 LOG		43 RCL
008	43 RCL	020	24 24	032	32 X↕T	044	42 STO		12 12
009	16 16 1/n	021	75 -	033	65 ×	045	17 17 d		29 CP
010	95 =	022	43 RCL	034	43 RCL	046	32 X↕T		91 R/S
011	55 ÷	023	11 11	035	00 00 n/12				

EDUCALC - The catalog for this mail order distributor lists over seventy books on calculators and calculator applications. The applications include such fields as real estate, chemistry, navigation, solar energy, statistics, astronomy, and the like. For information write to:

EduCALC Mail Store
27963 Cabot Road
South Laguna, CA 92677

ANOTHER GAME - Dave Lane. For those of you who have enjoyed my game of "Misadventure" I offer another one of logic. This was written for one of my friends who loves challenges, but did not have a PC-100 printer. The game can also be used to keep teen-agers quiet during long car trips--so that Mom and Dad can turn off the rock music. Here is the nasty little item:

WHAT! NO * RULES?

Yes--the name of the game tells it all--your task is to find the rules of the game! Here's how it works: You start with a total of 1. You must get 100 to win. You input only numbers, and the calculator will follow precise rules to come up with a new total. It may add, subtract, multiply or divide. It may ignore the number. It may dislike the number completely, and put you back to 1. When you hit 100, the calculator will flash the number of turns you took. After this, you can restart the game, and the calculator will initialize itself with another set of operations and input checks.

Some Instructions:
Read in Side 1 of the card.
Press A to initialize--it takes about 15 seconds.
Key in a number and press R/S. The calculator will use your input to calculate and display a new total.
Repeat keying in numbers and pressing R/S until you reach 100. When you do the calculator will flash the number of turns you took to get to 100.
Press CLR, then A to start a new game--with different rules!

Some Hints:
Initially, input a series of 2's and observe what happens. Record your input and the calculator's response on paper. Figure out the sequence which is constant during any one game. Use the sequence (of +, -, *, / and null) to get to 100. BEWARE, there are other rules than just the sequence.

The Challenge:
You can set up a specific sequence by seeding the random number generator. Key in a positive number and press B. Try to find the sequences which give you the biggest problems--largest number of turns to solve. Let one of your cohorts try to beat the number of turns. Let me know, c/o TI PPC Notes, which seed you find hardest, and the number of turns it took you to get to 100.

User Benefit:
To realize that you must learn all the rules before you can play the game effectively.

P.S. My wife wouldn't even try this game since it was so nebulous. So for my next game I'll come down to earth (or sea) and present a competitive game without so much logic.

The program is on the next page.

ANOTHER GAME (cont)

000	76	LBL	037	32	X:T	074	16	A'	111	42	STO	148	95	=
001	34	fX	038	69	OP	075	76	LBL	112	59	59	149	59	INT
002	43	RCL	039	26	26	076	75	-	113	76	LBL	150	72	ST*
003	04	04	040	69	OP	077	43	RCL	114	11	A	151	00	00
004	42	STO	041	27	27	078	08	08	115	85	+	152	69	OP
005	05	05	042	43	RCL	079	22	INV	116	08	8	153	20	20
006	09	9	043	04	04	080	44	SUM	117	95	=	154	97	DSZ
007	42	STO	044	22	INV	081	09	09	118	44	SUM	155	05	05
008	06	06	045	77	GE	082	16	A'	119	59	59	156	39	COS
009	76	LBL	046	45	Y^x	083	76	LBL	120	03	3	157	01	1
010	16	A'	047	00	0	084	65	×	121	69	OP	158	42	STO
011	22	INV	048	77	GE	085	43	RCL	122	17	17	159	09	09
012	97	DSZ	049	45	Y^x	086	08	08	123	47	CMS	160	61	GTO
013	05	05	050	73	RC*	087	49	PRD	124	06	6	161	34	fX
014	34	fX	051	06	06	088	09	09	125	69	OP	162	76	LBL
015	43	RCL	052	32	X:T	089	16	A'	126	17	17	163	18	C'
016	09	09	053	42	STO	090	76	LBL	127	18	C'	164	07	7
017	59	INT	054	08	08	091	55	÷	128	07	7	165	22	INV
018	32	X:T	055	00	0	092	43	RCL	129	85	+	166	23	LNX
019	01	1	056	67	EQ	093	08	08	130	05	5	167	65	×
020	00	0	057	85	+	094	22	INV	131	95	=	168	43	RCL
021	00	0	058	01	1	095	49	PRD	132	59	INT	169	59	59
022	67	EQ	059	67	EQ	096	09	09	133	42	STO	170	95	=
023	52	EE	060	75	-	097	16	A'	134	04	04	171	22	INV
024	22	INV	061	02	2	098	76	LBL	135	75	-	172	59	INT
025	77	GE	062	67	EQ	099	33	X²	136	02	2	173	42	STO
026	33	X²	063	65	×	100	01	1	137	95	=	174	59	59
027	00	0	064	03	3	101	61	GTO	138	42	STO	175	65	×
028	77	GE	065	67	EQ	102	42	STO	139	05	05	176	92	RTN
029	33	X²	066	55	÷	103	76	LBL	140	01	1	177	76	LBL
030	32	X:T	067	16	A'	104	45	Y^x	141	00	0	178	52	EE
031	76	LBL	068	76	LBL	105	01	1	142	42	STO	179	43	RCL
032	42	STO	069	85	+	106	42	STO	143	00	00	180	07	07
033	42	STO	070	43	RCL	107	09	09	144	76	LBL	181	61	GTO
034	09	09	071	08	08	108	16	A'	145	39	COS	182	61	GTO
035	91	R/S	072	44	SUM	109	76	LBL	146	18	C'	183	11	A
036	59	INT	073	09	09	110	12	B	147	05	5	184	00	0

KEYBOARD INTERRUPT - Ken Ward and Dave Leising. On the SR-52 the D/R switch provided a means to control branching of a running program from the keyboard. There is no similar switch on the TI-59. There is a way to mechanize a halt on interrupt. All you do is place a call to a RTN instruction in a CROM (Solid State module) where you want the interrupt to occur. The required code is Pgm MM SBR nnn where nnn is the address of a RTN in program MM. Then, with the program running, when you press and hold the RST key nothing will happen so long as the program is executing from user memory. But when the CROM call is executed the program will return to location 000 and halt. You must remember that a real RST occurs clearing the subroutine register, resetting flags, and the like.

Editor's Note: Old timers will remember that this technique was described in 52 Notes, by George Hartwig in V3N2P5 and by Roy Chardon in V3N3P6. Chardon wrote: "...Make the CROM call to PGM 01 SBR 021, since every CROM appears to have a RTN at that step." That idea was correct at the time, but later modules it will not work. For example, the M/U module has the equivalent RTN at Pgm 01 SBR 013 as a result of deleting the diagnostic portion of Pgm 01. That is also true in other modules as well.

For more complex techniques which provide branching from the keyboard without stopping the program see the articles by Martin Neef (v5N7P11) and Dejan Ristanovic (V6N9/10P31).

1188 DIGITS OF PI - Jovan Puzovic of Yugoslavia. This program was reviewed by Dejan Ristanovic who is well known to TI PPC Notes. One of the problems proposed for the ~~HP-41~~ "friendly competition" between the HP-41 and the TI-59 was the calculation of pi to large numbers of places. V7N5P9 of the PPC Calculator Journal reported in June 1980 that an HP-41 program would find 1160 places in 15¼ hours. Bob Fruit responded with a TI-59 program which would find 460 digits in 6 hours 18 minutes. Once again TI-59 users found that the HP-41 users had used their extra memory to advantage. But now Jovan Puzovic has found a way to circumvent the memory limitation and find 1188 digits of pi, but even with his program running in fast mode it requires eighty hours. His program mechanizes the same basic equation as that of Bob Fruit:

$$\pi = 16 * \arctan(1/5) - 4 * \arctan(1/239)$$

The calculation proceeds in three stages. First the arctan(1/5) portion is found and stored on magnetic cards. This requires about 62 hours. Then the arctan(1/239) portion is found, and is again dumped to magnetic cards. This requires an additional 18 hours. Finally a short routine combines the two intermediate solutions. The fast mode entry is accomplished in as few key-strokes as I have seen. The only discrepancy I can find is that the CLR at location 000 is not seen and the calculator is running in an error condition until the CE at location 016 is encountered. Of course this doesn't cause any problem. That effect was defined by Patrick Acosta in V6N8P4 at the top of the page.

User Instructions:

1. Enter program A first. There is a gap between steps 156 and 202 which is required since the Stflg Ind must be at the end of the partition for the fast mode entry technique to work. Press A and see a flashing display. Do not clear the flashing, but press 7 EE.
2. After about 62 hours the display reads zero. Press RST to get out of fast mode and record banks 1, 2, 3, and 4, marking the cards A1 through A4.
3. Enter program B. Note that program B is identical to program A for steps 000 through 156. Press B and see a flashing display. Press 7 EE.
4. After about 18 hours the display reads zero. Press RST to get out of fast mode and record banks 1 through 4 marking the cards B1 through B4.
5. Enter Program C. Press A. The display will read -3. Enter magnetic card side A4. Press R/S. The display will read -2. Enter magnetic card side B4. Press R/S. after a few minutes the printer will begin printing. The digits obtained are not the first ones, but are digits 840 through 1188. The calculator stops with a flashing display.
6. Press B and see -3 in the display. Proceed as in the rest of step 5, but this time enter card sides A3 and B3. You will have the next 360 higher digits of pi printed. Don't press A after B.
7. Press C and see -3 in the display. Again, proceed as in step 5, this time using card sides A2 and B2. 360 more digits of pi appear.
8. Press D and see -3 in the display. One more time, proceed as in step 5, this time using card sides A1 and B1. The highest 120 digits of pi will be printed starting with the 141592 ... sequence at the top. The leading 3 is not displayed.

1188 DIGITS OF PI (cont)

PROGRAM A

000	25	CLR	080	09	9	160	00	0
001	01	1	081	42	STO	161	00	0
002	00	0	082	00	00	162	00	0
003	69	DP	083	25	CLR	163	00	0
004	17	17	084	85	+	164	00	0
005	47	CMS	085	73	RC*	165	00	0
006	01	1	086	00	00	166	00	0
007	32	X:T	087	59	INT	167	00	0
008	09	9	088	75	-	168	00	0
009	09	9	089	53	(	169	00	0
010	42	STO	090	24	CE	170	00	0
011	00	00	091	55	÷	171	00	0
012	82	HIR	092	82	HIR	172	00	0
013	18	18	093	16	16	173	00	0
014	75	-	094	54	)	174	00	0
015	53	(	095	59	INT	175	00	0
016	24	CE	096	82	HIR	176	00	0
017	55	÷	097	05	05	177	00	0
018	82	HIR	098	65	×	178	00	0
019	17	17	099	82	HIR	179	00	0
020	54	)	100	16	16	180	00	0
021	59	INT	101	85	+	181	00	0
022	74	SM*	102	73	RC*	182	00	0
023	00	00	103	00	00	183	00	0
024	65	×	104	22	INV	184	00	0
025	82	HIR	105	59	INT	185	00	0
026	17	17	106	95	=	186	00	0
027	95	=	107	65	×	187	00	0
028	65	×	108	82	HIR	188	00	0
029	82	HIR	109	18	18	189	00	0
030	18	18	110	75	-	190	00	0
031	75	-	111	53	(	191	00	0
032	53	(	112	24	CE	192	00	0
033	53	(	113	55	÷	193	00	0
034	24	CE	114	82	HIR	194	00	0
035	55	÷	115	16	16	195	00	0
036	82	HIR	116	54	)	196	00	0
037	17	17	117	59	INT	197	00	0
038	54	)	118	72	ST*	198	00	0
039	59	INT	119	00	00	199	00	0
040	55	÷	120	65	×	200	00	0
041	82	HIR	121	82	HIR	201	00	0
042	18	18	122	16	16	202	76	LBL
043	54	)	123	95	=	203	11	A
044	74	SM*	124	65	×	204	25	CLR
045	00	00	125	82	HIR	205	09	9
046	65	×	126	18	18	206	69	DP
047	82	HIR	127	22	INV	207	17	17
048	17	17	128	64	PD*	208	01	1
049	65	×	129	00	00	209	07	7
050	82	HIR	130	82	HIR	210	00	0
051	18	18	131	15	15	211	05	5
052	95	=	132	74	SM*	212	82	HIR
053	65	×	133	00	00	213	07	07
054	82	HIR	134	82	HIR	214	02	2
055	18	18	135	18	18	215	05	5
056	97	DSZ	136	97	DSZ	216	82	HIR
057	00	00	137	00	00	217	06	06
058	00	00	138	00	00	218	01	1
059	14	14	139	84	84	219	52	EE
060	25	CLR	140	25	CLR	220	06	6
061	82	HIR	141	22	INV	221	82	HIR
062	17	17	142	87	IFF	222	08	08
063	50	I×I	143	01	01	223	68	NOP
064	67	EQ	144	00	00	224	25	CLR
065	01	01	145	08	08	225	02	2
066	47	47	146	91	R/S	226	52	EE
067	75	-	147	82	HIR	227	01	1
068	02	2	148	16	16	228	02	2
069	95	=	149	34	FX	229	94	+/-
070	65	×	150	82	HIR	230	85	+
071	82	HIR	151	06	06	231	01	1
072	17	17	152	86	STF	232	95	=
073	69	DP	153	01	01	233	22	INV
074	10	10	154	61	GTO	234	52	EE
075	94	+/-	155	00	00	235	58	FIX
076	95	=	156	79	79	236	00	00
077	82	HIR	157	00	0	237	60	DEG
078	07	07	158	00	0	238	86	STF
079	09	9	159	00	0	239	40	IND

PROGRAM B

202	76	LBL	242	59	INT	280	43	RCL
203	12	B	243	65	×	281	06	06
204	25	CLR	244	01	1	282	75	-
205	09	9	245	06	6	283	01	1
206	69	DP	246	75	-	284	95	=
207	17	17	247	73	RC*	285	42	STO
208	05	5	248	02	02	286	01	01
209	00	0	249	59	INT	287	73	RC*
210	01	1	250	04	4	288	01	01
211	82	HIR	251	85	+	289	59	INT
212	07	07	252	82	HIR	290	99	PRT
213	02	2	253	17	17	291	73	RC*
214	03	3	254	55	÷	292	01	01
215	09	9	255	82	HIR	293	22	INV
216	33	X²	256	18	18	294	59	INT
217	82	HIR	257	75	-	295	65	×
218	06	06	258	59	INT	296	82	HIR
219	01	1	259	82	HIR	297	18	18
220	52	EE	260	07	07	298	95	=
221	06	6	261	95	=	299	99	PRT
222	82	HIR	262	72	ST*	300	69	DP
223	08	08	263	01	01	301	31	31
224	25	CLR	264	82	HIR	302	97	DSZ
225	02	2	265	07	07	303	00	00
226	52	EE	266	42	STO	304	01	01
227	01	1	267	02	2	305	67	67
228	02	2	268	09	9	306	25	CLR
229	94	+/-	269	59	INT	307	35	1/X
230	85	+	270	65	×	308	91	R/S
231	01	1	271	01	1	309	02	2
232	95	=	272	06	6	310	00	0
233	22	INV	273	75	-	311	94	+/-
234	52	EE	274	73	RC*	312	82	HIR
235	58	FIX	275	02	02	313	07	07
236	00	00	276	59	INT	314	97	DSZ
237	60	DEG	277	65	×	315	00	00
238	86	STF	278	04	4	316	00	00
239	40	IND	279	85	+	317	14	14

PROGRAM C

000	76	LBL	080	59	INT	160	43	RCL
001	10	E'	081	65	×	161	06	06
002	25	CLR	082	01	1	162	75	-
003	09	9	083	06	6	163	01	1
004	69	DP	084	75	-	164	95	=
005	17	17	085	73	RC*	165	42	STO
006	03	3	086	02	02	166	01	01
007	94	+/-	087	22	INV	167	73	RC*
008	91	R/S	088	59	INT	168	01	01
009	02	2	089	65	×	169	59	INT
010	94	+/-	090	04	4	170	99	PRT
011	91	R/S	091	85	+	171	73	RC*
012	92	RTN	092	82	HIR	172	01	01
013	76	LBL	093	17	17	173	22	INV
014	11	A	094	55	÷	174	59	INT
015	10	E'	095	82	HIR	175	65	×
016	29	CP	096	18	18	176	82	HIR
017	01	1	097	75	-	177	18	18
018	52	EE	098	59	INT	178	95	=
019	06	6	099	82	HIR	179	99	PRT
020	82	HIR	100	07	07	180	69	DP
021	08	08	101	95	=	181	31	31
022	25	CLR	102	72	ST*	182	97	DSZ
023	82	HIR	103	01	01	183	00	00
024	07	07	104	42	STO	184	01	01
025	02	2	105	08	08	185	67	67
026	09	9	106	59	INT	186	25	CLR
027	42	STO	107	65	×	187	35	1/X
028	07	07	108	01	1	188	91	R/S
029	03	3	109	06	6	189	02	2
030	01	1	110	75	-	190	00	0
031	42	STO	111	73	RC*	191	94	+/-
032	06	06	112	02	02	192	82	HIR
033	61	GTO	113	59	INT	193	07	07
034	19	D'	114	65	×	194	65	×
035	76	LBL	115	04	4	195	82	HIR
036	12	B	116	85	+	196	18	18
037	76	LBL	117	82	HIR	197	95	=
038	13	C	118	17	17	198	22	INV
039	10	E'	119	95	=	199	44	SUM
040	03	3	120	42	STO	200	08	08
041	00	0	121	08	08	201	61	GTO
042	42	STO	122	22	INV	202	01	01
043	07	07	123	77	GE	203	29	29
044	42	STO	124	01	01			
045	06	06	125	89	89			
046	61	GTO	126	25	CLR			
047	19	D'	127	82	HIR			
048	76	LBL	128	07	07			
049	14	D	129	43	RCL			
050	10	E'	130	08	08			
051	01	1	131	75	-			
052	00	0	132	53	(			
053	42	STO	133	24	CE			
054	07	07	134	55	÷			
055	03	3	135	82	HIR			
056	00	0	136	18	18			
057	42	STO	137	54	)			
058	06	06	138	59	INT			
059	76	LBL	139	82	HIR			
060	19	D'	140	37	37			
061	43	RCL	141	65	×			
062	07	07	142	82	HIR			
063	42	STO	143	18	18			
064	00	00	144	95	=			
065	43	RCL	145	74	SM*			
066	06	06	146	01	01			
067	42	STO	147	69	DP			
068	01	01	148	21	21			
069	85	+	149	69	DP			
070	03	3	150	22	22			
071	00	0	151	97	DSZ			
072	95	=	152	00	00			
073	42	STO	153	00	00			
074	02	02	154	75	75			
075	73	RC*	155	43	RCL			
076	01	01	156	07	07			
077	42	STO	157	42	STO			
078	08	08	158	00	00			
079	22	INV	159	85	+			

MU-19 CONTROLLER - Palmer Hanson - In my work I had reason to use the Discrete Fourier Series (MU-19) program regularly. The program as written provides rapid assembly of the input data into a file, but requires the user to individually enter the number for each harmonic solution desired. In my application I typically wanted all the coefficients up to a certain harmonic. This little program, when operated with a printer, provides titles for input and output, and annotation of the Fourier coefficients. Operator selection of the highest harmonic is provided. The program uses Karl Gailer's print code converter program (V5N6P10) since the highest value to convert was less than 89. By limiting the length of the program to 160 steps the maximum number of input values is permitted. The printout numbers the input data starting at 01. The data is actually stored in memory starting at R16. The sample problem is from page 73 or the Math/Utilities manual. The instructions for running the program are:

1. With the Math/Utilities module in place, load the program.

2. Press A to initialize and set up for data input. The headings will be printed and 31 will appear in the display.

3. Enter the data in sequence. Press R/S for data entry. The data will be printed and the calculator will stop with the number of data entries in the display.

4. To solve, enter the number of the highest harmonic desired and Press D. New headings will be printed for the solution, and the harmonics will be printed with appropriate annotation.

```
000  76 LBL    040  98 ADV    080  43 RCL    120  10 E'
001  10 E'     041  01  1     081  01  01    121  85  +
002  55  ÷      042  06  6     082  61 GTO    122  01  1
003  01  1     043  42 STO    083  00  00    123  03  3
004  00  0     044  00  00    084  64  64    124  00  0
005  85  +     045  69 DP     085  76 LBL    125  00  0
006  01  1     046  00  00    086  14  D     126  00  0
007  93  .     047  01  1     087  98 ADV    127  00  0
008  01  1     048  06  6     088  42 STO    128  95  =
009  85  +     049  01  1     089  00  00    129  69 DP
010  59 INT    050  03  3     090  69 DP     130  04  04
011  65  ×     051  03  3     091  20  20    131  43 RCL
012  09  9     052  07  7     092  69 DP     132  04  04
013  09  9     053  01  1     093  00  00    133  36 PGM
014  85  +     054  03  3     094  02  2     134  19  19
015  53  (     055  69 DP     095  01  1     135  14  D
016  22 INV    056  02  02    096  05  5     136  69 DP
017  59 INT    057  03  3     097  05  5     137  06  06
018  65  ×     058  01  1     098  03  3     138  01  1
019  01  1     059  69 DP     099  07  7     139  85  +
020  00  0     060  04  04    100  05  5     140  52 EE
021  85  +     061  69 DP     101  06  6     141  08  8
022  69 DP     062  05  05    102  69 DP     142  94 +/-
023  10  10    063  98 ADV    103  02  02    143  95  =
024  65  ×     064  91 R/S    104  69 DP     144  82 HIR
025  08  8     065  72 ST*    105  05  05    145  38  38
026  08  8     066  00  00    106  98 ADV    146  25 CLR
027  75  -     067  69 DP     107  43 RCL    147  32 X:T
028  08  8     068  21  21    108  01  01    148  69 DP
029  08  8     069  43 RCL    109  42 STO    149  06  06
030  95  =     070  01  01    110  10  10    150  69 DP
031  59 INT    071  10 E'     111  01  1     151  24  24
032  92 RTN    072  69 DP     112  06  6     152  98 ADV
033  76 LBL    073  04  04    113  42 STO    153  97 DSZ
034  11  A     074  73 RC*    114  11  11    154  00  00
035  01  1     075  00  00    115  00  0     155  01  01
036  00  0     076  69 DP     116  42 STO    156  18  18
037  69 DP     077  06  06    117  04  04    157  98 ADV
038  17  17    078  69 DP     118  43 RCL    158  60 DEG
039  47 CMS    079  20  20    119  04  04    159  91 R/S
```

DATA	R
2.	01
3.2	02
3.7	03
2.5	04
1.2	05
1.5	06
2.7	07
2.4	08
0.	09
-3.3	10
-3.3	11
0.	12

F(T)	
2.1	A00
0.	B00
-1.417222017	A01
1.885961277	B01
-0.55	A02
1.905255888	B02
.5833333333	A03
.0166666667	B03
0.25	A04
.0577350269	B04
.0838886833	A05
-.0192946108	B05
0.	A06
0.	B06

PROFILE PLOT - A fast mode program for the TI-58C by David Lobbestael of Blenheim, Ontario, Canada. This program uses the h12 method for transferring in and out of fast mode during program execution. It uses the Math/Utilities module and runs under control of the MU-05 Superplotter program. The program returns to user memory to run the function routine in fast mode. Several unusual fast mode techniques are incorporated in this single program:

1. Hexadecimal code 9C which prints and displays as 02 is used for fast mode entry. That causes the unusual sequence of locations during listing of 007, 009, 009, 010 -- note the two 009's. Patrick Acosta had recognized that fast mode entry could be obtained with any hex code which ended with a 2, but had recommended that h12 be used to avoid undesireable side effects. There are no obvious bad effects here.

2. Other fast mode users have experienced the problem of dropping into TRACE mode when leaving fast mode (see bottom of V6N8P4). This program recognizes that the problem is caused by the setting of Flag 9 at fast mode entry. Remember that flag 9 sets trace when using a printer? See page V-65 of Personal Programming. The program as written solves the problem with an INV Stflg 9 just before fast mode exit (locations 234 to 236). The TRACE effect can also be eliminated by using a different fast mode entry constant, say with 2 EE-12 + 1 rather than the 2 EE-12 + 3 used in the program.

3. The program uses the technique described in the middle of V6N8P4 and the RTN at location 237 to leave fast mode and return to MU-05 control.

User Instructions:

1. Generate the h(9C) at location 008 with the following sequence with the Master Library module installed: GTO 8 LRN 0 BST LRN Pgm 1 2 SBR 4 4 4 R/S P-R LRN Ins LRN RST CLR. This sequence will not work with the Math/Utilities module installed. Having generated the hex code turn off the calculator and replace the Master Library module with the Math/Utilities module. Note that one of the glories of doing hex codes with the TI-58C is that they survive turnoff!

2. Key in the rest of the program. Be sure to remember that you may not perform either a delete or an insert with the program counter lower than 009 or you will destroy the hex code.

3. Store the number 13 in R05, and the number 2 EE 12 +/- + 3 = in R03.

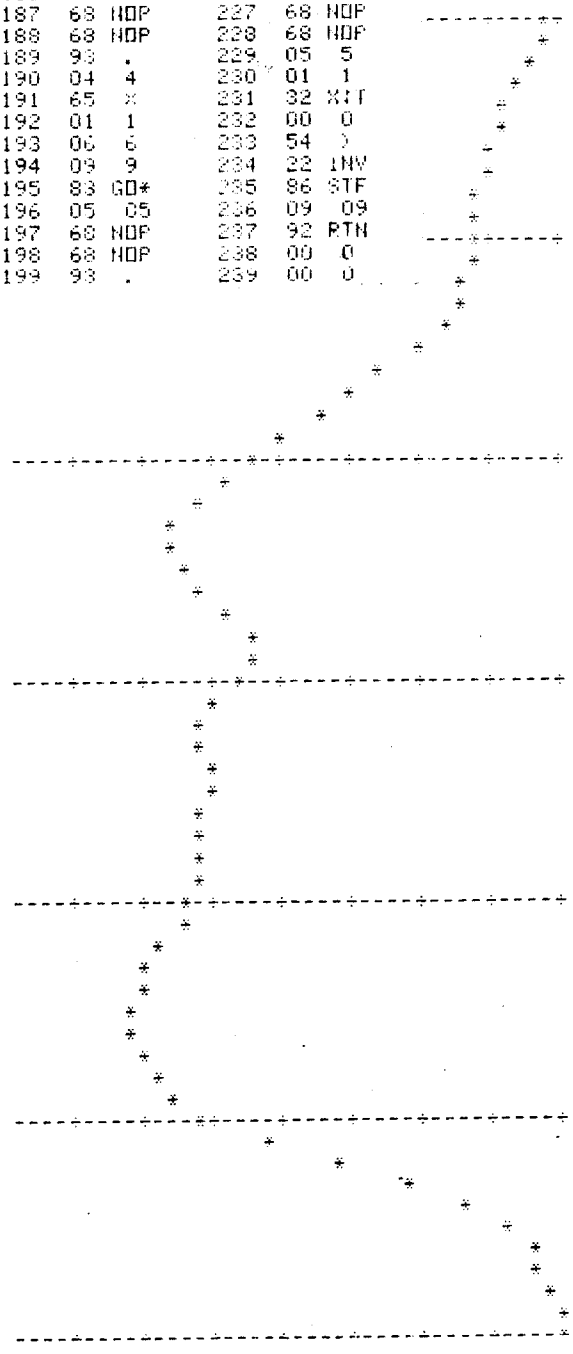
4. Select the MU-05 program and set up the initialization:

a. 2nd Pgm 05	
b. Enter 0 and press A	X_0
c. Enter 2 x π DIV 61 = and press B	delta-X
d. Enter 61 +/- and press C	Y_{min}
e. Enter 80 and press D	Y_{max}
f. Enter 2 and press E	Number of tapes
g. Enter 1 and press D'	Number of functions
h. Enter 61 and press E'	Number of points

5. Run time is about 55 minutes. The asymmetric values in steps 4.d and 4.e are not true min/max values but they produce the best proportioned graph.

```
000  76 LBL    040  00  0     080  93  .     120  06  6     160  05  5     200  05  5
001  16 A'     041  55  ÷     081  01  1     121  65  ×     161  65  ×     201  65  ×
002  58 FIX    042  89  π     082  65  ×     122  02  2     162  02  2     202  02  2
003  09  09    043  54  )     083  02  2     123  04  4     163  02  2     203  03  3
004  43 RCL    044  42 STO    084  01  1     124  02  2     164  09  9     204  00  0
005  03  03    045  00  00    085  05  5     125  83 GO*    165  83 GO*    205  83 GO*
006  86 STF    046  42 STO    086  83 GO*    126  05  05    166  05  05    206  05  05
007  40 IND    047  01  01    087  05  05    127  68 NOP    167  68 NOP    207  68 NOP
009  02  02    048  53  (     088  68 NOP    128  68 NOP    168  68 NOP    208  68 NOP
009  68 NOP    049  04  4     089  04  4     129  01  1     169  93  .     209  93  .
010  61 GTO    050  09  9     090  93  .     130  93  .     170  07  7     210  05  5
011  00  00    051  93  .     091  05  5     131  06  6     171  65  ×     211  65  ×
012  31  31    052  06  6     092  65  ×     132  65  ×     172  01  1     212  02  2
013  53  (     053  65  ×     093  08  8     133  03  3     173  00  0     213  00  0
014  24 CE     054  03  3     094  00  0     134  03  3     174  03  3     214  07  7
015  85  +     055  00  0     095  83 GO*    135  01  1     175  83 GO*    215  83 GO*
016  43 RCL    056  02  2     096  05  05    136  83 GO*    176  05  05    216  05  05
017  01  01    057  83 GO*    097  68 NOP    137  05  05    177  68 NOP    217  68 NOP
018  54  )     058  05  05    098  68 NOP    138  68 NOP    178  68 NOP    218  68 NOP
019  38 SIN    059  01  1     099  93  .     139  01  1     179  93  .     219  93  .
020  85  +     060  07  7     100  06  6     140  93  .     180  03  3     220  04  4
021  43 RCL    061  93  .     101  65  ×     141  03  3     181  65  ×     221  65  ×
022  00  00    062  04  4     102  01  1     142  65  ×     182  03  3     222  06  6
023  44 SUM    063  65  ×     103  07  7     143  02  2     183  00  0     223  04  4
024  01  01    064  02  2     104  01  1     144  00  0     184  05  5     224  83 GO*
025  01  1     065  09  9     105  83 GO*    145  08  8     185  83 GO*    225  05  05
026  00  0     066  08  8     106  05  05    146  83 GO*    186  05  05    226  68 NOP
027  44 SUM    067  83 GO*    107  68 NOP    147  05  05    187  68 NOP    227  68 NOP
028  02  02    068  05  05    108  68 NOP    148  68 NOP    188  68 NOP    228  68 NOP
029  83 GO*    069  01  1     109  02  2     149  93  .     189  93  .     229  05  5
030  02  02    070  03  3     110  93  .     150  03  3     190  04  4     230  01  1
031  04  4     071  93  .     111  07  7     151  65  ×     191  65  ×     231  32 X:T
032  09  9     072  08  8     112  65  ×     152  08  8     192  01  1     232  00  0
033  42 STO    073  65  ×     113  03  3     153  09  9     193  06  6     233  54  )
034  02  02    074  01  1     114  04  4     154  83 GO*    194  09  9     234  22 INV
035  53  (     075  09  9     115  83 GO*    155  05  05    195  83 GO*    235  86 STF
036  32 X:T    076  05  5     116  05  05    156  68 NOP    196  05  05    236  09  09
037  65  ×     077  83 GO*    117  68 NOP    157  68 NOP    197  68 NOP    237  92 RTN
038  01  1     078  05  05    118  68 NOP    158  68 NOP    198  68 NOP    238  00  0
039  08  8     079  07  7     119  93  .     159  93  .     199  93  .     239  00  0
```

The program in user memory is simply the profile of a person reduced to Fourier coefficients and the required housekeeping to enter and exit from fast mode. David writes that he wrote the program as a test of his agility in programming coupling this with the peculiar "quirks" of the TI-58C. The program can be optimized somewhat. After some extensive correspondence we had not achieved but a few minutes improvement in speed. The original program shown here stands as an excellent demonstrator of fast mode techniques on the TI-58C. Most of the techniques are transferrable to the TI-59. A superb program!!!



STATISTICS QUIRKS IN THE TI-55II - While testing other calculators for the presence of the INV Σ+ quirk of the TI-58 and TI-59, I found another, more serious quirk in the statistics routines of the TI-55II. Following the example on page 8-1 of the Calculator Decision-Making Sourcebook which is provided as the manual for the TI-55II, but omitting the use of the Fix-2 mode, press ON/C ON/C 2nd CSR to clear the calculator for statistics operations. Press 4 Σ+ and see a 1 in the display, indicating one entry has been made. Continue with the example by pressing 5 Σ+ 6 Σ+ 7 Σ+ and 8 Σ+, and see a 5 in the display indicating that 5 entries have been made. Press 2nd Mean and find the value 6. in the display as the calculated population mean. Now press - 6 = and find -1 -10 in the display, indicating that the calculated mean for the five input integers 4, 5, 6, 7, and 8 is not quite exactly 6. Other TI calculators with statistics functions such as the TI-57, TI-58C and TI-59 obtain the exact value of 6. Even the TI-35 will not produce the small residual error.

There is still more mischief in the statistics algorithm of the TI-55II. Let's test the Σ- function. Continuing on with the problem above, enter 40000000 Σ+ and see a 6 in the display. Now assume that was an erroneous entry. To remove it, enter 40000000 2nd Σ- and see a 5 in the display verifying removal of one entry. Press 2nd Mean, expecting to find a 6 in the display. Instead, you will find 5.9996. Inserting and removing the large value at different positions in the sequence will yield different results. For example, go back to the beginning and use the following sequence: ON/C ON/C 2nd CSR 4 Σ+ 5 Σ+ 6 Σ+ 40000000 Σ+ 40000000 2nd Σ- 7 Σ+ and 8 Σ+. See a five in the display as expected. Press 2nd Mean and see 6.0006 in the display. Try a different sequence, splitting the entry and removal of the large value: ON/C ON/C 2nd CSR 4 Σ+ 40000000 Σ+ 5 Σ+ 6 Σ+ 7 Σ+ 40000000 2nd Σ- and 8 Σ+ and see the expected 5 in the display. Press 2nd Mean and see 6.00104 in the display. Clearly, some unexpected things are happening. With the same sequences with the TI-57, TI-58C, TI-59, and TI-35 the answer is always the expeoted answer of exactly 6. Doing the equivalent arithmetic without use of the statistics keys of the TI-55II yields the expected answers. Use of other values than 40000000 in the Σ+ and Σ- sequence will yield other displayed results. If 400000 had been used instead of 40000000 in the first example above the displayed mean would have been 5.999996. Experimentation will show that the erroneous residual is divided by ten for each factor of ten by which the large simulated erroneous entry and removal value is reduced.

After some extensive experimenting I zeroed in on a function which will yield the same results as the statistics routine of the TI-55II. Using the TI-55II and starting at location 00 write the program:

- RCL 1 = EXC 0 + 1 = EXC 0 x RCL 0 1/X + RCL 1 = STO 1 R/S

- RCL 1 = EXC 0 - 1 = EXC 0 +/- x RCL 0 1/X + RCL 1 = STO 1 R/S

With that program you must clear locations 0 and 1, say with a 2nd CM. Then you can do a Σ+ by entering the value and pressing RST R/S. The latest mean value will be displayed. You can do a Σ-, but only immediately after a Σ+, by entering the value and pressing only R/S. Use of this little program seems to duplicate the calculation of means from the statistics functions of the TI-55II.

STATISTICS QUIRKS (cont) Analysis of the program reveals two features. The sum of the input values is not maintained; rather the current mean and the number of entries is maintained. The idea to try that method came from George Thomson who says the technique has been around for many years. This allows the calculator to handle smaller numbers, since in the full-up version the information to be used for the standard deviation calculations is maintained as the squares of deviations from the mean. The use of the x RCL 0 1/X sequence rather than the more obvious DIV RCL 0 seems to indicate that the programmer was obeying the old rule to avoid dividing whereever possible.

In future issues I will explore the effect of similar sequences on the standard deviation calculations. There, all calculators I have tested have had the potential to yield misleading results, if only the user has the misfortune to have used the particular values to which the algorithm is sensitive. All in this case means TI-55II, TI-35, TI-57, TI-58C, TI-59, HP-35, HP-67, HP-11, and even the HP-41. The really safe thing to do is not use the Σ- function.

THOUGHTS ABOUT CURVE FITTING - George Wm. Thomson. (Editor's note: Just as you can get into trouble by not understanding how the built-in statistics routines work, you can also misinterpret your results if you use "canned" curve fitting programs like those in V7N1/2F15 or RE-10/RE-11 in the Real Estate/ Investment module. The problem comes from the linearization techniques used to obtain the solutions. This article is highly recommended reading for users of those programs.)

Linear Relations

Suppose that it is assumed that some variable Y can be predicted from another variable X using the equation

$$(1) \qquad Y = A + BX$$

Once A and B are established, the deviations between the observed values of Y and the calculated values (A + BX) are given by:

$$(2) \qquad Y - A - BX$$

The Method of Least Squares is one way of determining a good set of values of A and B in many senses. The set of data points X, Y are taken to be fixed and A and B are adjusted until the Sum of the Squares of the Deviations is a minimum. That is the two partial derivatives of the sum of the deviations must be zero:

$$(3) \qquad \frac{\partial}{\partial A} \Sigma(Y - A - BX)^2 = 0$$

$$(4) \qquad \frac{\partial}{\partial B} \Sigma(Y - A - BX)^2 = 0$$

The familiar "least squares" solutions for fitting Y to X with all points of equal weight and X free from error follow immediately from these relations.

THOUGHTS ABOUT CURVE FITTING (cont)

Non-linear Relations

If data are assumed to fit relations where the X and/or Y no longer retain linearity, then complications arise. Suppose we have:

(5) $Y = A' X^{B'}$

Clearly

(6) $\frac{\partial}{\partial A'} \Sigma(Y - A' X^{B'})^2 = 0$

and

(7) $\frac{\partial}{\partial B'} \Sigma(Y - A' X^{B'})^2 = 0$

do not have neat solutions; an awkward non-linear solution is needed for A' and B'. On the other hand, the use of a transformation such as

(8) $\ln Y = \ln A' + B' \ln X$

$= A + B X$

no longer minimizes error in Y when the Method of Least Squares is used. Instead, it minimizes errors in the logarithm of Y, that is, proportionate or percentage errors in Y. For many problems, this may be the right thing to do, but the user should be aware that the standard process does not minimize absolute errors in Y.

There is a way of adjustment using the logarithmic transformation which retains the simplicity of the Method of Least Squares. In the usual procedure, all points are of equal weight. To compensate for the log transformation each data pair must be weighted by Y^2. That is, we must minimize

(9) $\Sigma w (\ln Y - A - B \ln X)^2$

This is no more difficult than the standard minimization and leads to analogous equations in terms of six cross-product sums. The sums can be stored in R01 to R06 and manipulated with Op 22:

The lower case symbols x, y at the right are used for the logarithmic transforms.

R 01	Σwy
R 02	Σwy^2
R 03	Σw
R 04	Σwx
R 05	Σwx^2
R 06	Σwxy

Let $\bar{x}$ and $\bar{y}$ be the weighted means of $\ln x$ and $\ln y$, that is

(10) $\bar{x} = R04/R03 = \Sigma wx/\Sigma w$

(11) $\bar{y} = R01/R03 = \Sigma wy/\Sigma w$

The least squares equation passes through this mean with slope

(12) $b = \Sigma w (x - \bar{x})(y - \bar{y}) / \Sigma w (x - \bar{x})^2$

THOUGHTS ABOUT CURVE FITTING (cOnt)

that is,

$$b = (\Sigma wxy - \Sigma wx \, \Sigma wy/\Sigma w) / (\Sigma wx^2 - (\Sigma wx)^2/\Sigma w)$$

The formulas above for handling weighted linear least squares problems are quite general. A standard problem of fitting Y to X can be handled if the points are of unequal weight by generating cross-products as shown above. (Note that only Op 12 and its x⇄t, Op 14 and Op 15 are applicable since R03 contains Σw, not $\bar{x}$.) Also, the weights can be scaled up or down at your convenience if the same factor is applied to all.

Example

The artificial example below shows a moderate range of Y values, 2 to 85. (Real life examples where log transforms are used may be 13 to 2000, as in barometric measurements for vapor pressure.) The hope is to fit an equation to within a few tenths in Y.

		"Y"		"ln Y"	
X	Y	Calc	Diff	Calc	Diff
1	1.9	1.66	+0.24	1.82	+0.08
2	5.7	5.44	+0.26	5.75	-0.05
3	10.8	10.89	-0.09	11.27	-0.47
4	17.6	17.82	-0.22	18.16	-0.56
5	26.0	26.11	-0.11	26.28	-0.28
6	35.7	35.67	+0.03	35.56	+0.14
8	58.4	58.37	+0.03	57.29	+1.11
10	85.4	85.53	-0.13	82.94	+2.46
Mean Difference			0.00		0.30
SE Fit			0.1872		1.1467
Solution		$Y = 1.660\,X^{1.712}$		$Y = 1.823\,X^{1.658}$	

The "Y" solution in the table above was obtained using TI-59 Program GWT-60, a straightforward application of the principles enumerated above. The deviations in Y are fairly evenly divided over the range, and average zero. The "ln Y" solution can be obtained from the use of the statistical keys on the TI-59 using logarithmic transforms. The same solution can be obtained with the curve fitting program in V7N1P15, or with the curve fitting programs RE-10/RE-11 in the Real Estate/Investment module. With no weighting involved the statistical cross-products are generated automatically so that each X,Y pair has the same unit weight. Because of the equal emphasis of ln Y on the low end and the high end, the fit at the high end suffers badly.

In conclusion, we should think hard about our data, their inherent accuracy, as well as precision, and be able to answer correctly the question about the correctness and adequacy of the whole methodology used to relate our data to the relationships we are investigating. And finally, can we tie our equation to some aspect of the real world around us?

THOUGHTS ABOUT CURVE FITTING (cont)

References

Many texts on statistics pay scant attention to the handling of data transforms. A fine reference for the entire field of least squares analysis is "Statistical Adjustment of Data", by W. Edwards Deming, Wiley, 1943. In case you think this is old-fashioned, bear in mind that hardly anything new has come along in numerical analysis since the days of Gauss who enunciated the principle of least squares in 1809 and applied it to many complex problems in astronomy, engineering and in hs monumental survey of all Germany. The weighting of ln Y in proportion to Y^2 is mentioned many times. Exercise 18, Section 67, provides a graphic example of the large error possible when weighting is ignored,

GWT-60 - User instructions on the following page.

```
000  76 LBL
001  16 A'
002  32 X:T
003  01  1
004  22 INV
005  44 SUM
006  10  10
007  32 X:T
008  61 GTO
009  10 E'
010  76 LBL
011  11  A
012  47 CMS
013  02  2
014  00  0
015  76 LBL
016  10 E'
017  42 STO
018  07  07
019  85  +
020  02  2
021  05  5
022  95  =
023  42 STO
024  08  08
025  92 RTN
026  98 ADV
027  99 PRT
028  72 ST*
029  07  07
030  69 OP
031  27  27
032  91 R/S
033  99 PRT
034  72 ST*
035  08  08
036  01  1
037  44 SUM
038  10  10
039  43 RCL
040  07  07
041  61 GTO
042  10 E'
043  76 LBL
044  15  E
045  36 PGM
046  01  01
047  71 SBR
048  25 CLR
049  42 STO
050  00  00
051  43 RCL
052  10  10
053  42 STO
054  09  09
055  02  2
056  00  0
057  10 E'
058  76 LBL
059  30 TAN
060  73 RC*
061  08  08
062  33 X²
063  44 SUM
064  00  00
065  69 OP
066  28  28
067  97 DSZ
068  09  09
069  30 TAN
070  43 RCL
071  10  10
072  55  ÷
073  43 RCL
074  00  00
075  95  =
076  42 STO
077  18  18
078  43 RCL
079  10  10
080  42 STO
081  09  09
082  02  2
083  00  0
084  42 STO
085  07  07
086  76 LBL
087  38 SIN
088  43 RCL
089  07  07
090  10 E'
091  73 RC*
092  08  08
093  33 X²
094  65  ×
095  43 RCL
096  18  18
097  95  =
098  42 STO
099  11  11
100  44 SUM
101  03  03
102  73 RC*
103  08  08
104  23 LNX
105  42 STO
106  13  13
107  65  ×
108  43 RCL
109  11  11
110  95  =
111  44 SUM
112  01  01
113  73 RC*
114  07  07
115  23 LNX
116  42 STO
117  12  12
118  65  ×
119  43 RCL
120  11  11
121  95  =
122  44 SUM
123  04  04
124  65  ×
125  43 RCL
126  13  13
127  95  =
128  44 SUM
129  06  06
130  43 RCL
131  11  11
132  65  ×
133  43 RCL
134  12  12
135  33 X²
136  95  =
137  44 SUM
138  05  05
139  43 RCL
140  11  11
141  65  ×
142  43 RCL
143  13  13
144  33 X²
145  95  =
146  44 SUM
147  02  02
148  69 OP
149  27  27
150  97 DSZ
151  09  09
152  38 SIN
153  98 ADV
154  69 OP
155  12  12
156  22 INV
157  23 LNX
158  91 R/S
159  42 STO
160  14  14
161  99 PRT
162  32 X:T
163  91 R/S
164  42 STO
165  15  15
166  99 PRT
167  98 ADV
168  76 LBL
169  14  D
170  58 FIX
171  02  02
172  98 ADV
173  43 RCL
174  10  10
175  42 STO
176  09  09
177  02  2
178  00  0
179  42 STO
180  07  07
181  25 CLR
182  42 STO
183  16  16
184  42 STO
185  17  17
186  76 LBL
187  39 COS
188  43 RCL
189  07  07
190  10 E'
191  73 RC*
192  07  07
193  99 PRT
194  23 LNX
195  42 STO
196  12  12
197  73 RC*
198  08  08
199  99 PRT
200  73 RC*
201  07  07
202  45 Y^x
203  43 RCL
204  15  15
205  65  ×
206  43 RCL
207  14  14
208  95  =
209  99 PRT
210  94 +/-
211  85  +
212  73 RC*
213  08  08
214  95  =
215  99 PRT
216  44 SUM
217  16  16
218  33 X²
219  44 SUM
220  17  17
221  69 OP
222  27  27
223  98 ADV
224  97 DSZ
225  09  09
226  39 COS
227  43 RCL
228  16  16
229  55  ÷
230  43 RCL
231  10  10
232  95  =
233  99 PRT
234  58 FIX
235  04  04
236  43 RCL
237  17  17
238  55  ÷
239  53  (
240  43 RCL
241  10  10
242  75  -
243  02  2
244  95  =
245  34 fX
246  99 PRT
247  22 INV
248  58 FIX
249  61 GTO
250  03  03
251  99  99
252  76 LBL
253  12  B
254  42 STO
255  14  14
256  91 R/S
257  42 STO
258  15  15
259  61 GTO
260  14  D
261  76 LBL
262  13  C
263  42 STO
264  14  14
265  91 R/S
266  42 STO
267  15  15
268  76 LBL
269  18  C'
270  91 R/S
271  99 PRT
272  45 Y^x
273  43 RCL
274  15  15
275  65  ×
276  43 RCL
277  14  14
278  95  =
279  99 PRT
280  98 ADV
281  61 GTO
282  18  C'
```

User Instructions for GWT-60:

GWT-60 provides a least squares curve fit for $Y = aX^b$, minimizing errors in Y. The X values are assumed error free. The data pairs are assumed of equal weight. The program runs with partitioning set at 7 Op 17. It stores the X values in locations 20, 21, 22.... and the Y values in locations 45, 46, 47..... . Up to 25 data pairs can be accepted. Unrounded values of the constants a and b are displayed at the completion of the least squares solution. An option is provided to use the unrounded values as is, or to use rounded values. An output table shows X input, y input, Y calculated, and the difference. At the end the average difference and the standard error of the fit are printed.

1. To input data, press A and see 45 in the display which is the address of the Y storage location to be used. Enter X, press R/S. The X value is printed and displayed. Enter Y, press R/S. The Y value is printed, and the calculator stops with the address of the next Y storage location to be used. Enter additional pairs.

2. For data correction, enter the storage location for the X value of the data pair to be corrected. Press A'. The address of the companion Y storage location will be displayed. Re-enter both the X and Y values as in step 1.

3. To add data pairs to the list. You must know the address of the first unused storage location for X. You may obtain this with the sequence RCL 10 + 25 =. Then press E' and see the address of the first unused Y storage location in the display. Proceed as in step 1.

4. To solve press E. The calculator will stop with an unrounded value for a in the display. You have the option to round it or not. Then, press R/S. The value you selected for a will be printed and the calculator will stop with the unrounded solution for b in the display. Again, you change it or not. Then, press R/S. The value you selected for b will be printed and the calculator will proceed to print out the table of the X value, the Y value, the calculated Y value, and the difference for each data pair. At the end the the mean error and standard error of the fit are printed.

5. You may evaluate the residuals for other coefficients by entering a and pressing B. B. The calculator stops with a in the display. Enter b and press R/S. The calculator proceeds with the printout of the residuals as in step 4.

6. To do single value calculations, but not residuals, enter a and press C. Enter b and press R/S. Enter X and press R/S. The calculator displays and prints $Y = aX^b$. For additional solutions with the same a and b, enter a new X and press C'.

7. You may want to select a different Fix value for the residual printout--see locations 170/171.

8. Note: The program does not check for zero or negative input values.

MAGNETIC CARD SERVICE - On page 3 I stated that I would try a magnetic card service for TI PPC Notes programs on a trial basis. I will provide cards for programs for one dollar per card plus a stamped and self addressed envelope. At that price (1) I cannot guarantee results due to the well known incompatibility of some calculators--if the card you doesn't work it will cost another dollar and another self-addressed stamped envelope for another try, and (2) I will not test each program to see that it operates properly. What I will do is load my calculator from cards which have provided satisfactory results in the past, write new cards, turn the calculator off and on again and load the new cards. My experience has shown that those procedures are satisfactory with properly operating calculators. I will not combine programs on cards. If you ask for a program of less than 240 steps I will load the program on both sides to improve the chances of success. In addition to cards for the programs in this issue I can also provide the following programs:

Fast Mode Calendar	V5N7P7	2 cards
Inverse List Print All	V5N9/10P15	1 card
Plot 60	V6N4/5P5	1 card
Modulo 210 SFF	V6N4/5P13	2 cards
Fitting Data to Curves	V7N1/2P15	3 cards
13 Digit Printer	V7N4/5P11	1 card
Supertest TI-59	V7N9P11	1 card
Pie Chart	V7N10P6	2 cards

A PROGRAMMING CHALLENGE - Gary Brown has taught TI-59 programming for several years. One problem which he assigned to students was a Sliding Average Filter. The problem:

Given a set of data points taken at even increments (ΔX), smooth the data by averaging a set number (n) of data points about each point.

Keyboard Inputs:

1. n = number of points to be averaged at each X, where X is greater than or equal to n. Your program should work over a range for n from 3 through 10.
2. The initial value of X (X_0).
3. The uniform increment of X (ΔX).
4. Each data point Y_0, Y_1, Y_2, ... that corresponds to each X_i.

Outputs from the Printer:

Note: These outputs should occur after the first n entries of Y_i values have been entered. Thereafter, a new average Y will be printed for each new Y_i entry.

1. The midpoint of X_n that corresponds to the specific n interval being computed.
2. The average Y over the n data points.

You will recognize the problem as similar to that solved by the ML-17 program. ML-17 used 116 program locations. Gary's students have solved the problem in under 50 locations. What can club members do?

The end of V8N1

Pork