* TI PPC NOTES *

v5N3, 1980.

NEWSLETTER OF THE TI PERSONAL PROGRAMMABLE CALCULATOR CLUB.

9213 Lanham Severn Road, LANHAM, Maryland, 20801.

Several things are new with this issue. We have a table of contents. Several members pointed out to me that every self-respecting newsletter has one. It will be practical if, at some time in the future, you are hunting for some specific article or reference.

We increased the number of pages from 12 to 16. I cannot guarantee this to be a permanent feature. It will largely depend on members' contributions in the form of programs and routines. Thanks to the many who send in a lot of good, usable material. Be patient, it takes time to send everything out to the referees and write it and paste listings in a publishable form. But most of it will appear in these pages.

I discontinued the practice if typing program sequences, except for short routines. The method is too error-prone. In the first issue alone I forgot two steps.(see p3) In its place we will have 50% reductions of actual PC100 listings. This takes about the same space as typed versions and has the added advantage of showing the program steps, the codes and the mnemonics, all in an unambiguous manner.

As a consequence of the foregoing, we went a step further and reduced four pages, text and program listings, to one single one. It looked so good, that we decided to publish two games this way, Mr. Magic on page 10 and Reaction Time Test on page 13. Which brings the actual total number of pages to 22!

To satisfy many requests I am announcing some large systems. One type is a translation from German articles on specialized subjects. The other type is "hardware": descriptions of the PC100, the TI-59 and several interface units. The prices quoted are for mailing inside the US, Canada and Mexico. Members living elsewhere please send a few dollars extra if you want it airmailed to you. Otherwise, indicate which type of mailing is OK. No need to send more money than indicated in that case.

The April fool joke was intended as a good-natured -but according to some members"mean" - introduction to the new NOTES. If it robbed you of some much-needed sleep, my sincere apologies. But remember, I said: "At this point you may go to bed. Sometime during the night..." Remember?

I hope this enlarged issue will bring us closer to our goal: have a newsletter we can be proud of. Please keep telling me what else you expect.

Maurice E.T. Swinnen.

TABLE OF CONTENTS.

How to crack a protected program 2
Errata, blind RTN, dot cleaner................................3
D-ring binders, programming systems, angle converters
TI-58 senser............4
Enhanced print routine, ME, 1 % resistor routine................5
Time dilation, firmware revisited...............................6
Decimal point trick, twin-T filter.........................7+8
Key board HIR...9
Mr. Magic, card trick program...............................10
Alpha data list..11+12
1*2,2*1 routines..12
Reaction time test program..................................13
Newcomer's corner, HIR print code loading...................14
Print code converter..15
Mathematical diversions: the pegboard problem...............16

The telecommunications experiment on the one hand has aroused a lively interest in some of the members to break the code on a protected card. On the other hand the experiment succeeded beyond my wildest dreams: I caught a lot of innocents! Moral of the story: a good scientist never accepts anything on face value. That goes also for a good programmer: question anything and everything until you are convinced that you discovered the truth for yourself. And even then check it out once in a while, in the light of new evidence.
The first doubting Thomas was Robert U. Myers of Lorton, Virginia. Early in March he sent me the correct sequence to trick the card in giving up its message: A R/S CLR STO 49 R/S. Later in the month Robert even sent me another sequence, the one I had intentionally left as a small window to be discovered by avid researchers: SBR 011 through SBR 014. Bill Kantrowitz in Brookline, Mass,wrote me the same day that his 15-year-old son had found that one too. Very clever, all you Sherlocks!
In the mean time I received at least one attempt to read the protected card itself. Dave Leising simply scraped off the magnetic coating in the small spot where the protection flag was recorded! He was very accurate, because only the last octet was messed up a little.But he got the whole message this way.
But nobody was using the "patended" Blachly-Swinnen code cracker routine: Pull off a length of Scotch tape, about twice the length of a mag card. Paste the very tip, about 2 mm, to the right side of the card. (the small black area) Now fold the tape carefully in a loop with the sticky side in and again paste the last 2 mm now to the magnetic side of the card, on the same spot opposite the small black area on the right side of the card. See drawing.

Now squeeze the tape loop shut, starting at the mag card. Then trim the tape loop, now a handle, to the same width as the card itself.
Turn on the TI-59 and read-in the card in the usual way, by first pressing CLR and letting the card slide into the slot on the right side of the calculator. But here comes the trickery and it will require a little practice on your side. When you see the card appearing on the left side , allow it to advance only so far that you see the first of the five blocks only, the one corresponding to A and A'. Then with a swift motion, pull back on the handle, against the movement of the motor. If you get a flashing zero, you are OK. Just press CLR , RST and LIST. The secret program will be listed. If, on the other hand, the swift pull back caused the motor to run away such that it cannot be stopped with R/S, don't dispair. Just read-in a blank card. That will make the motor stop, after the blank card has run through.Then proceed as explained above: CLR RST LIST.
If you don't get a flashing zero, but a -1 to -4 indication, the protection flag got into the calculator. This means you allowed the card to go in too far. Turn off the calculator and start all over again. So, practice, practice and practice again. Wasn't that the way you learned to ride a bike?
If you don't want your "opponent" to guess the sequence from the key board, you may write a mean program or a viscious one. Mine was a mean one: it used labels that couldn't be keyed from the key board and it had three flags set. If you did not set the correct flags, any sequence would dump you into a flashing routine. But I left a small window open, so that any persistent searcher would stand a fighting chance. But then I got a look at a viscious one: Robert Meyers returned my card with one. It required you to put 7^4 or 2401 into the t-register! Any other number and the only response you got was flashing or nonsense printing. To make your life even more miserable, Bob had the card recorded in 9 OP 17, to preclude anybody from writing a "HIR viewer" in bank 2 and so get at "the truth." The program succumbed, though, to the all-powerful Blachly-Swinnen code cracker!

SOME COMMENTS ON ISSUE 1 and 2.
Sharp-eyed members discovered a couple of typos: V5N1p3, first program, line 3 of the program itself: after the second RTN, add an = sign. Otherwise, the program will never arrive at the first of the two + signs and therefore will never flash with a nagative value.
V5N1p5, manual plotter-scaler, third line of the program itself: at the end of that line, add an R/S. The program plots alright, but does strange things to out-of-bounds-values.
In the second issue V5N2p6, Alphabetical Sort, step 002: change from decimal point (code 93) to CLR. (code 25) If you had a digit in the display when you pressed A, storing would begin at n + 1. With CLR in step 002 it always begins at 1, i.e. register 01. For the rest, the program is still "tops."
Paul Berg sent me his version of the access program on page 9: his is 12 steps shorter! Clever programming, Paul. But as a general philosophy about such excercises, if they are done to hone one's programming skills, they serve a very useful purpose. If, on the other hand, they are done to replace an already well-running routine and do not make the existing one run any faster, they are rather academic. The main consideration in these routines is that 1) they do what they are supposed to do and 2) they fit within 240 steps, such that two will fit on one mag card. The routines on pages 8 and 9 are shorter than or equal to 240 steps. Making them any shorter does not serve any further purpose.

Thomas Wysmuller reminds us of the fact that any RTN in a program is "blind", that is, any return encountered in a program will return program control back to the last point of departure. In other words, the RTN instruction that you rely upon to take you back to the address after the last subroutine call does not have to be a unique part of your subroutine. For example, the common print sequence OP 05 OP 00 ADV RTN appearing at the end of one sequence can serve as the endpoint of any number of subroutines which, instead of ending with their own RTN instruction can terminate with GTO xxx, where xxx is the location of the "print sequence." A variation of this approach will permit you to write LBL D in V5N1P3, first program, as LBL D SBR 068 (RCL 01 X 2 GTO 018, saving 7 steps in the process. Naturally, the direct addresses will have tobe changed.

Thomas provides a different printing dot cleaning routine he has been using:
OP 05 RST LBL D 11 LBL E x 101010101 = OP 01 OP 02 OP 03 OP 04 RST
LBL A 24 E LBL B 74 E LBL C 32 E
To use it, first press A and see a lot of I's printed. Then press R/S and press B for some II's. Next R/S and C for some square O's. Lastly press R/S and D to see a stream of 8 characters. What this routine does is selectively heat each of the print dots to maximum, i.e. seven times in less than .33 sec. Thomas says that it worked to clean rather dirty dots in all cases, except once, when a hair was stuck in a fresh roll of paper, courtesey , but no charge of TI. A tweezer worked wonders in that case to remove the hair from across four dots.

THINGS TO COME IN FUTURE ISSUES.- Emil Regelman and John Wortington of Bowie, MD, cooperated in developing a TELEPHONE RATE TIMER program. It allows you to find out <u>during</u> a long-distance call how much money you spent so far. The TI-59 runs a continuous display of that sum and the seconds remaining for the same money!!!! Prof Wilbur Widmer of the University of Connecticut wrote a RELATIVE PRIMALITY program for up to 56 integers. George Vogel sent me a MORTGAGE LOAN program that will run well on an SR-56. Bill Beebe of Lilburn, Georgia contributed a novel SR-52 LISTING PROGRAM that allows you to enter 45 steps at a time!!! And both Panos Galidas and John Wortington promised to have a well-documented program that actually takes out the labels and replaces them with direct addresses in an existing program!!! Both claim they can do 85 steps at a time. Other members contributed also programs, but, with my apologies, there is not enough space to mention them all.In any case, thanks to all who contributed so much.

For all you hole punchers: The Snow brothers are using the ideal ring binders, they tell me. Instead of being round, the rings in this binder are formed into an inverted letter D It holds more paper, about 25 % more, and it doesn't tear the holes as easily. You see, when you open the binder, the contents don't move, as in a round-ring type. They are made by Cardinal Products and called SUPER-COMBI binders. Cardinal Products is a division of Durand Manuf.Co. 939 West 35th Street, Chicago, IL 60609. Their tel.# is 312-254-9320. The binders are, of course, available in any office supply store worth its salt.

PROGRAMMING SYSTEMS. We have in our files several large articles which I translated from Display.(German) I would love to publish them, but they would take up an inordinate amount of space. Moreover, they are of interest only to certain people. Therefore, I think the best way to make them available is by request only, at a nominal copying and mailing fee. Here is a partial list. I will announce the rest once I have checked them for completeness and accuracy:

ASTRONOMY.- An article that appeared over two issues, written by Heinrich Schnepf and E. Lunnebach. Is a complete mini-course on astronomy. Contains programs for the TI-59 and the HP-67/97 to compute the ephemerides, rise and set and culmination of the planets, the sun, the moon, plus moon phases and eclipses. The complete text has been translated into English, but the alpha in the programs has not been translated. Should be easy, though, as all the equivalent words in the text have been done. The article consists of 46 pages on European format DIN-A4, which is longer than our 8.5 by 11. Therefore I will have to copy on 8.5 by 13 which can be trimmed to fit in a three-ring binder. Contains the original drawings on which I typed the translation of all the terms. Copy quality is excellent. A friend of mine who is employed at the Naval Observatory checked the whole thing for accuracy and has added some handwritten notes in the margins. Copying and mailing comes to $ 10.00 US.

PRIME NUMBER GENERATORS AND FACTOR FINDERS.- Also translated from Display. Contains 20 pages of articles and programs by several authors. Mostly for the SR-52, which can easily be written in 59-ese. One program each for the HP-67/97 and TI-59. For the 52 there is done some rather extensive research on the use of the pseudos, especially how to use p83 as an integer function. (no problem in the 59, though) Again the format is DIN-A4 size, slightly longer than 11 inches. Copying will be done on 8.5 by 13 inches paper. Copying and mailing comes to $ 6.00 US.

MASTER LIBRARY ACCESS ROUTINES.- As shown on Pages 8 and 9 of V5N1, these access routines will print descriptors in the margin. They work with the same instructions as the ones used in the ML manual. To reduce copying costs, no instructions are added. Only the programs listings are given for ML-04 through ML-25. The first 20 are written by Valentino Ducati, the last two by me. Each listing on 8.5 by 11 in. copy quality very good to excellent. Copying and mailing comes to $ 6.00 US.

ANGLE CONVERTERS.- When you examine surveying programs, occasionally you encounter terribly long routines intended to convert an angle into a standard one, say between 0 and 360. So, I wondered if anybody had developed some efficient and fast tricks to do this. Frank Blachly assured me they did exist. So, here are the ones he uses regularly: To convert an angle to $0 \le \theta \le 360$:

LBL A X:T 360 X:T P/R INV P/R CP GE STO + 360) LBL STO RTN

And to convert an angle to $-90 \le \theta \le 270$:

LBL B X:T 1 X:T P/R INV P/R RTN

Entry may be positive or negative. Fractional part should be in decimal degrees.

PGM 01 of the RPN-module contains this TI-58 check routine:

If the TI-58 is connected, just CMS, if TI-59 is connected, partition to 159.99 and then CMS.

269	06	6
270	69	OP
271	17	17
272	29	CP
273	59	INT
274	67	EQ
275	02	02
276	81	81
277	01	1
278	00	0
279	69	OP
280	17	17
281	47	CMS

Re the PRT routine in V5N1p2: J. Huntington Lewis, of Norfolk, VA, proposes the following enhancement:

LBL PRT STO AA RCL IND AA DIV 12 INV LOG + 1) HIR 08 X:T OP 06 RTN

so that the routine can be called as follows: DD X:T BB SBR PRT
where BB is the location of the desired print code. This will print the displayed value DD with the appropriate legend in the right hand margin.

LAMBDA FUNCTION. Robert Savage, 5628 Berkeley Road in Goleta CA, 93017, would like to see a program developed for the Lambda function.The program should have a 10-digit accuracy for arguments up to 50 or 100 and an order of at least 10. Please write Robert directly if you can help him.

HARDWARE.- I have several hardware articles available. These are too long to be published here.They may be ordered at a nominal fee for copying and mailing:
EXTERNAL COMMUNICATIONS FOR THE SR-52/56 CALCULATORS.Describes all the signal voltages and wave forms available in those calculators. Contains the bit patterns *for the printed characters, the instruction codes and the character codes. 12 pages, 8.5 by 11 inches. Dated Sept 27, 1976. $ 3.00 US.*
PC-100A, INTERFACE DESCRIPTION.Gives all the PC-100A I/O requirements, the operating instructions and the character codes for the printed characters. It also has the PROM codes for the SN74S287 IC circuit in the PC100A. It further has the Instruction Register (IRG) instructions, a schematic diagram of the interface and of the PC100A proper. The date is June 1, 1978. 12 pages. $ 3.00 US.
PC100 INTERFACE DESCRIPTION. Same as above but for the PC100 as opposed to the PC100A. Does not have a schematic of the interface, only a description of it. But contains a schematic diagram of the PC100. Dated Sept 30, 1976.7 pages. $ 2.00 US.
CRT INTERFACE FOR THE TI-59.Complete description to interface the TI-59 to a CRT-Display. The RF modulator schematic diagram is the only part that did not copy clearly. 8 Schematic diagrams for the complete interface. ROM map for the display characters. 24 pages. No date. $ 6.00 US.
TI-59 SCHEMATIC DIAGRAM.- 1 Page, free with any of the above. Otherwise, just send me a SASE.
It goes without saying that we cannot guarantee the above as to accuracy. We did not try them out, but the source we obtained them from is impeccable.

MECHANICAL ENGINEERING. Gus Raab, 806 Freedom Circle, Harleysville PA, 19438, would like to get in contact with any member who has developed programs in structural steel design and detailing or drafting. Also plate and sheet metal drafting and layout, machine shop math, template making and machine design. Please write Gus directly if you feel you can interchange programs and ideas with him, or call him at 215-368-4866.

---

1 % RESISTOR ROUTINE.- In connection with the Standard Resistor routine in V5N2, Paul E. Blair sends us this 1% resistor routine he wrote for the SR-52. It works well on the TI-58/59. The first two open parenthesis are not needed there, I suppose. It will produce the exact correct 1 % value for any odd entry:

LBL EE ((INV EE LOG X 96) FIX 0 EE FIX 2 DIV 96) INV LOG EE INV EE INV FIX RTN

A similar routine for 5% and 10% resistors does not agree with the commercially available values, they not being theoretically correct in some values. The alteration would be to change both 96's in 24's or 12's and Fix 2 in Fix 1.

TIME DILATION.- John Garza III offers a surprisingly short routine to compute the classic time dilation. As you might know, it is an interesting result of the theory of relativity, which says that time passes at different rates for objects moving relative to one another, as seen by an outside observer. This occurs even at the most minuscule velocities, but is only readily noticed at relativistic velocities, that is close to the speed of light. John assumes here the speed of light c = 1. The instructions are:
RST, enter velocity V and press X:T. Enter time and compute stationary time by pressing R/S.
In user memory: DIV X:T INV COS SIN = R/S
For all you astronomers, amateur and otherwise, John has also a STELLAR TRANSFORMATIONS program. It transforms the earth-origin coordinates of a star into it's coordinates using another star or the Shapley center as origin. It also computes the magnitude of the star as seen from the new origin.
If I receive enough requests, I'll publish it. Otherwise, I might add it to out Astronomy package. That is, if John agrees to it.

FIRMWARE REVISITED. The subject FIRMWARE refuses to die. In all the German newsletters not an issue goes by without a mention or a rehash of all the known facts. No NEW firmware locations are discovered, though. Assumptions as to how many firmware program locations there are are running amok. Ever since Walter Ulrich showed his discovery in GESPRO (see NOTES V5N1P7) in which the program counter was made to go up as high as 1374, several other attempts have surfaced. The latest one from TISOFT (Belgium) VII,N1. Their sequence is:
GTO 004 LRN PGM 02 SBR 240 RST RST RST LRN RST R/S LRN
Your program counter should now be at 6546 with a code 81. But if you go out of LRN and press LIST the listing starts at step 546. This strongly suggests that the program step 6546 is only a pseudo-step for the real step 546. The latter, in turn, can be found in Steffen Seitz's routine as 146. At least the listed code is identical there.
The editor offers the suggestion that the code is repeated nine times for a total of 8000 steps! To "prove" it,he has computed a formula to pre-determine the address. His recipe is as follows:

1. *In GTO 004 the number 004 may be replaced by any number 4 modulo 8.*
2. *Instead of SBR 240 you may also use SBR 239.*
3. *In RST RST RST, the first RST may be replaced by any code in the eighties: HIR, GTO IND, OP IND, +, IFF, D.MS, pi, GRAD, STF)*
4. *The last two in the row of three RST's constitute the address. It is always a four-digit address: 800a + 80b + 8c + d + 1*
   with one exception: when d = 9 then d = -1
   As an example, you enter RST √x) which makes the address 5434 (reading backwards) and the location then becomes 5 X 800 + 4 X 80 + 3 X 8 + 4 + 1 = 4349
   Another example: RST RST RST the last two RST are 8181 and the location is: 8 X 800 + 1 X 80 + 8 X 8 + 1 + 1 = 6546
   Now you try RST PRT PRT ! In LRN mode SST a few times and, lo and behold, the program counter goes to 000 and there is Steffen's firmware all over again, at the correct location!
   While ckecking out the sequence above and trying variants on it, I found that GTO 016 LRN PGM 02 SBR 240 31 31 RST LRN RST R/S *(synthesize the two "31" (= LRN)steps) will put the calculator in Don Laughery's curious "trace " state, but in fix 4 format! But this time, if you reset flag 9 you kill the "trace" state!*

THE DECIMAL-POINT TRICK. In V2N2p35 of Display, Heinrich Schnepf published a decimal-point trick, intended as an entry-sensing device. Since then I have seen several good programmers use it to produce really "friendly" programs, i.e. programs in which the amount of button pushing is reduced to an absolute minimum. But the great majority of people still struggle along with instructions as:" Of the three variables, enter only two through either key A, B or C and obtain the unknown through either key A', B' or C'." That means six keys for the user to remember, when three could do it. How? Let's look at Ohm's law.(Don't worry, I am NOT going to write a program about it!) R = E/I, E = IR and I = E/R. A rather simple interrelationship among three variables. You could enter your data as:

(Enter R) LBL A STO 01 R/S
(Enter E) LBL B STO 02 R/S
(Enter I) LBL C STO 03 R/S

And the computation of the three unknowns could be as follows:

(Compute R) LBL A' RCL 02 DIV RCL 03 = R/S
(Compute E) LBL B' RCL 01 X RCL 03 = R/S
(Compute I) LBL C' RCL 02 DIV RCL 01 = R/S

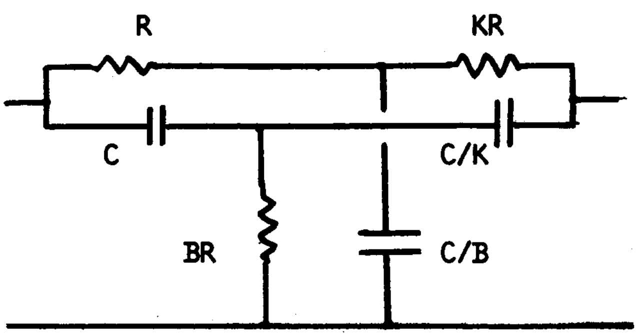
It works, but it is an unfriendly program, because of excessive button pushing. If we use the decimal-point trick at the very beginning of each entry label, we can reduce the number of keys to be remembered to the absolute minimum of three. Thus, each entry routine is written as:

LBL A . CP EQ A' STO 01 R/S LBL B . CP EQ B' STO 02 R/S LCL C . CP EQ C' STO 03 R/S

And the computation routines are written the same as above.

The decimal point will make any non-entry, that is a left over result from a former computation, zero. Then the CP command will store a zero in the t-register. That is the only thing CP will do in a program.(From the key board, CP will wipe out your program, because it means CANCEL PROGRAM) The next command is EQ or x=t if you will. (I prefer EQ, which is the same as the printer lists it. For x=t and its mate, x ≥ t, which I write GE, I have to do quite some gymnastics with the typing balls on the machine) This GE compares the display with the t-reg and if both are equal to zero the program will branch to A', B' or C' as the case may be. You old-time programmers, as opposed to newcomers, will say:"Why is he telling us all this. We knew this already a long time." Then why, pray tell, do I see it used so seldom?

My friend Robert Trost, in Soest, Holland, who is a professional market researcher and,as a part time job,invents electronic aids for the handicapped, recently sent me a schematic diagram of a frequency-selective infra-red amplifier. He requested me to write a quick-and-dirty program to compute the values of the twin-T filter feedback element used in such an amplifier. I would like to share the two versions of this program with you, to demonstrate what is meant by a reduction in button pushing. For those not familiar with a twin-T circuit (and for those who might have forgotten) let's examine the relationships between the different elements:



$$RC = \frac{1}{\omega o} = \frac{1}{2 \pi f}$$

$$B = \frac{K}{K + 1}$$

In which K is the bandwidth factor from 1 to 100 and B simply a constant as defined.

f is the frequency and R and C are the circuit elements as shown in the drawing.

For the widest band, K = 1 and for the narrowest band K = 100. (practical values) It is clear(people familiar with electronics) that for the wide band version we could easily get away with 5 % or even 10 % tolerances for the circuit elements. But for narrower bandwidths we will have to resort to much tighter tolerances, such

Decimal-point trick. (cont.)

1 % or better. Because of the rather simple relationship among the variables, it should be possible to write a program with less than 240 steps, even one with all the bells and whistles possible. With the TI-58 users in mind and also to please the TI-59 users who haven't a PC100 available, I wrote a simple calculator-only version. Then I wrote one for TI-59/PC100 use, with despriptor printing in the right-hand margin. Both will fit on one single mag card, each on its own track. The instructions are simple:

Enter two out of three variables through their respective keys: A, B or C.

To obtain the unknown, press the left-over key.

Enter the bandwidth factor K and press E.

With the printer, the four other circuit elements are printed with descriptors. Without the printer (the short version) the program stops each time to permit copying. Press R/S to continue.

R is entered in ohms and C in microfarads. If that bothers you, change the dividers in the program accordingly.(example step 060: DIV 6 INV LOG =)

```
000  76 LBL
001  99 PRT
002  55  ÷
003  01  1
004  02  2
005  22 INV
006  28 LOG
007  85  +
008  01  1
009  54  )
010  82 HIR
011  08  08
012  32 X:T
013  69 OP
014  06  06
015  22 INV
016  58 FIX
017  92 RTN
018  76 LBL
019  89  π
020  65  ×
021  02  2
022  65  ×
023  89  π
024  95  =
025  35 1/X
026  92 RTN
027  76 LBL
028  11  A
029  93  .
030  29 CP
031  67  EQ
032  16  A'
033  76 LBL
034  42 STO
035  32 X:T
036  03  3
037  05  5
038  58 FIX
039  00  00
040  71 SBR
041  99 PRT
042  42 STO
043  00  00
044  92 RTN
045  76 LBL
046  12  B
047  93  .
048  29 CP
049  67  EQ
050  17  B'
051  76 LBL
052  43 RCL
053  32 X:T
054  01  1
055  05  5
056  58 FIX
057  06  06
058  71 SBR
059  99 PRT
060  55  ÷
061  06  6
062  22 INV
063  28 LOG
064  95  =
065  42 STO
066  01  01
067  92 RTN
068  76 LBL
069  13  C
070  93  .
071  29 CP
072  67  EQ
073  18  C'
074  76 LBL
075  44 SUM
076  32 X:T
077  02  2
078  01  1
079  58 FIX
080  00  00
081  71 SBR
082  99 PRT
083  42 STO
084  02  02
085  92 RTN
086  76 LBL
087  16  A'
088  43 RCL
089  01  01
090  65  ×
091  43 RCL
092  02  02
093  71 SBR
094  89  π
095  71 SBR
096  42 STO
097  98 ADV
098  92 RTN
099  76 LBL
100  17  B'
101  43 RCL
102  00  00
103  65  ×
104  43 RCL
105  02  02
106  71 SBR
107  89  π
108  65  ×
109  06  6
110  22 INV
111  28 LOG
112  95  =
113  71 SBR
114  43 RCL
115  98 ADV
116  92 RTN
117  76 LBL
118  18  C'
119  43 RCL
120  00  00
121  65  ×
122  43 RCL
123  01  01
124  71 SBR
125  89  π
126  71 SBR
127  44 SUM
128  98 ADV
129  92 RTN
130  76 LBL
131  15  E
132  32 X:T
133  02  2
134  06  6
135  00  0
136  00  0
137  71 SBR
138  99 PRT
139  42 STO
140  03  03
141  55  ÷
142  53  (
143  24 CE
144  85  +
145  01  1
146  95  =
147  42 STO
148  04  04
149  43 RCL
150  00  00
151  65  ×
152  43 RCL
153  03  03
154  95  =
155  32 X:T
156  02  2
157  06  6
158  03  3
159  05  5
160  58 FIX
161  00  00
162  71 SBR
163  99 PRT
164  43 RCL
165  00  00
166  65  ×
167  43 RCL
168  04  04
169  95  =
170  32 X:T
171  01  1
172  04  4
173  03  3
174  05  5
175  58 FIX
176  00  00
177  71 SBR
178  99 PRT
179  43 RCL
180  01  01
181  55  ÷
182  43 RCL
183  03  03
184  65  ×
185  06  6
186  22 INV
187  28 LOG
188  95  =
189  32 X:T
190  01  1
191  05  5
192  06  6
193  03  3
194  02  2
195  06  6
196  58 FIX
197  06  06
198  71 SBR
199  99 PRT
200  43 RCL
201  01  01
202  55  ÷
203  43 RCL
204  04  04
205  65  ×
206  06  6
207  22 INV
208  28 LOG
209  95  =
210  32 X:T
211  01  1
212  05  5
213  06  6
214  03  3
215  01  1
216  04  4
217  58 FIX
218  06  06
219  71 SBR
220  99 PRT
221  98 ADV
222  92 RTN
```

```
001  99 PRT
019  89  π
028  11  A
034  42 STO
046  12  B
052  43 RCL
069  13  C
075  44 SUM
087  16  A'
100  17  B'
118  18  C'
131  15  E
```

TWIN-T FILTER.

EXECUTION SAMPLE:

10000	A
.05	B
	C
320	C
.05	B
	A
.05	B
250	C
	A
10	E

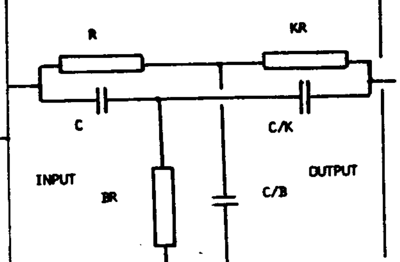
IN THE SHORT VERSION, PRESS R/S TO CONTINUE. VALUES WILL APPEAR IN THE SAME ORDER AS SHOWN IN THE PRINTER VERSION.

```
   10000.    R
0.050000     C
     318.    F

     320.    F
0.050000     C
    9947.    R

0.050000     C
     250.    F
   12732.    R

      10.    K
  127324.    KR
   11575.    BR
0.005000     C/K
0.055000     C/B
```




```
000  76 LBL
001  89  π
002  65  ×
003  02  2
004  65  ×
005  89  π
006  95  =
007  35 1/X
008  92 RTN
009  76 LBL
010  11  A
011  93  .
012  29 CP
013  67  EQ
014  16  A'
015  42 STO
016  00  00
017  91 R/S
018  76 LBL
019  12  B
020  93  .
021  29 CP
022  67  EQ
023  17  B'
024  55  ÷
025  06  6
026  22 INV
027  28 LOG
028  95  =
029  42 STO
030  01  01
031  91 R/S
032  76 LBL
033  13  C
034  93  .
035  29 CP
036  67  EQ
037  18  C'
038  42 STO
039  02  02
040  91 R/S
041  76 LBL
042  16  A'
043  43 RCL
044  01  01
045  65  ×
046  43 RCL
047  02  02
048  71 SBR
049  89  π
050  42 STO
051  00  00
052  91 R/S
053  76 LBL
054  17  B'
055  43 RCL
056  00  00
057  65  ×
058  43 RCL
059  02  02
060  71 SBR
061  89  π
062  65  ×
063  06  6
064  22 INV
065  28 LOG
066  95  =
067  42 STO
068  01  01
069  91 R/S
070  76 LBL
071  18  C'
072  43 RCL
073  01  01
074  65  ×
075  43 RCL
076  00  00
077  71 SBR
078  89  π
079  42 STO
080  02  02
081  91 R/S
082  76 LBL
083  15  E
084  42 STO
085  03  03
086  55  ÷
087  53  (
088  24 CE
089  85  +
090  01  1
091  95  =
092  42 STO
093  04  04
094  43 RCL
095  03  03
096  91 R/S
097  43 RCL
098  00  00
099  65  ×
100  43 RCL
101  03  03
102  95  =
103  91 R/S
104  43 RCL
105  04  04
106  65  ×
107  43 RCL
108  00  00
109  95  =
110  91 R/S
111  43 RCL
112  01  01
113  55  ÷
114  43 RCL
115  03  03
116  65  ×
117  06  6
118  22 INV
119  28 LOG
120  95  =
121  91 R/S
122  43 RCL
123  01  01
124  55  ÷
125  43 RCL
126  04  04
127  65  ×
128  06  6
129  22 INV
130  28 LOG
131  95  =
132  91 R/S
133  25 CLR
134  69 OP
135  40  40
136  91 R/S
137  91 R/S
138  91 R/S
139  91 R/S
```

```
001  89  π
010  11  A
019  12  B
033  13  C
042  16  A'
054  17  B'
071  18  C'
083  15  E
```

KEY BOARD HIR ! - *One disadvantage of the HIRs is that they have to be used in program memory. They cannot be keyed directly. However, if you just want to experiment with HIRs, this program by Dick Blayney will put all the known HIR codes in program memory. By calling a specified SBR you can execute each HIR command. The program itself will tell you which SBR to execute!*
Dick's original program did not contain print commands. I added them in the believe that it helps your memory to see the HIR and its corresponding SBR printed. This interferes a little with HIR 8. If you store something in it and then recall from HIR 8 you will see .0000003614 instead. This is the print code for the letters S and B from printing the word SBR. If that bothers you, just delete all the printing: steps 036 through 047 and steps 056 through 065 and you will obtain Dick's original routine. You could then use the table instead, in order to verify the SBR corresponding to a specified HIR command. Try it both ways.
The instructions are as follows: Execute LBL A, allow 70 sec for its execution. Routine A will put all known HIR commands in user memory, including the non-existing HIR 9 commands. But it never hurts to experiment with them. You never know you might become famous.... If you want to put in the other non-existent HIR, HIR 00, after execution of A, press GTO 957 LRN STO 82 BST BST DEL SST SST R/S. The calculator will go out of LRN automatically.
After execution of A you want, for example, to store 1234 in HIR 7. Enter 7 and press E. Without the printer only 901 is displayed, meaning that to execute HIR 07 you have to press SBR 901. With the printer both 7.HIR and 901. SBR are printed. Next enter 1234 and press SBR 901.

```
LBL A 10 OP 17  99 STO  00 RCL 00  EE 3 +/-  + 9.10082  = STO IND 00 CLR  DSZ 0 010
OP 17  R/S   LBL E  X:T  232435  OP 04  X:T  OP 06  X  8 - 957  =  ABS  X:T  361435
OP 04  X:T  OP 06  R/S
```

FUNCTION	HIR REGISTERS 0	1	2	3	4	5	6	7	8	9
0 = STO	957	949	941	933	925	917	909	901	893	885
1 = RCL	877	869	861	853	845	837	829	821	813	805
2	797	789	781	773	765	757	749	741	733	725
3 = SUM	717	709	701	693	685	677	669	661	653	645
4 = PRD	637	629	621	613	605	597	589	581	573	656
5 = INV SUM	557	549	541	533	525	517	509	501	493	485
6 = INV PROD	477	469	461	453	445	437	429	421	413	405
7 = INV PROD	397	389	381	373	365	357	349	341	333	325
8 = INV PROD	317	309	301	293	285	277	269	261	253	245
9 = INV PROD	237	229	221	213	205	197	189	181	173	165

A few reminders about the HIRs:
HIR 05 = OP 01, HIR 06 = op 02, HIR 07 = op 03 and HIR 08 = OP 04.
Nested arithmetic operands are pushed in HIR 1, HIR 2, ...8.
CMS, CLR nor CE will clear the HIRs, but OP 00 will clear HIR 5, 6, 7 and 8.
P/R uses first available HIRs and HIR 7 and 8. OP 11 uses first two available HIRs.
INV P/R uses first two available HIRs and HIR 7 and 8.OP 13: first four available HIRs.
DMS and INV DMS use first two available HIRs and HIR 7 and 8.
Sigma + and - and Σ+ use HIR 7 and 8. OP 15 uses first three available HIRs.
$\bar{x}$ uses first available HIR. OP 14 uses first three available HIRs and HIR 8.

• M R .   M A G I C •

A CARD TRICK PROGRAM, BY ROBERT ¢ RICHARD SNOW.   TI-59 / PC-100.

As with all card tricks, this illusion requires some sneaky sleight of hand. The sleight of hand is provided by the TI-59 itself. You announce that your calculator is going to guess any card. To the astonishment of your audience the calculator accomplishes this feat in about six guesses. What the poor "victim" doesn't realize is, that it is you who is steering the calculator onto the right track!
When you press B, for BEGIN, the printer prints the following message: READY? PICK A CARD. The "victim" picks one from a deck and places it, face up, on the table. Now you watch the display and see that it is flashing alternately 9999 and 9999. (with a decimal point) When you press R/S and hold that key down for a second or so, to allow a loop to complete itself, there are always two outcomes possible, depending on the presence or absence of a decimal point following the 9's. But only you know that! Your victim, even someone familiar with the TI-59, will not always notice this slight difference. You can easily predict the outcome, what the printer will print, by following, and hopefully learning by heart, the binary tree below. The large black dots indicate the presence of the decimal point. Thus, the first time you press R/S with the decimal point, you obtain BLACK. The next time SPADE results. From now on, the decimal point means the high card(s). With a little practice you will see that it is very easy to obtain the required result. But, if you make a mistake, and don't we all once in a while, don't despair! This develish program has a fantastic error recovery: If you make a boo-boo, and print the wrong color or card, start talking fast: "Funny, I still have to check this program. It sometimes goes off on a tangent! It has been known to reverse itself. Maybe, this time...let's see..." And, lo and behold, the printer prints I CHANGED MY MIND and then prints the opposite of its first answer! This error recovery happens about every 10 seconds and, for credibility's sake, should not be abused. This also means that you have only 10 seconds to produce the correct answer each time, so, practice and hurry up!
The audience is usually flabbergasted as to how the calculator is able to pull this off. They suspect some sort of trickery, of course, but few are able to pinpoint it. Some of my victims watched my feet: maybe they suspected that I secretely directed the TI-59 by means of foot pedals! Others even asked me not to repeat aloud the value and color of the card. Did they actually think the TI-59 could hear it?
Once you have become an expert at this trick, you might shorten the game somewhat and astound even more your audience, by using some extra sleight of hand. If the card is between 2 and 10, this extended trick will work to tell the card at once, after the initial selection of the color and the suit. It is done by pressing 2nd A', after the R/S key to halt the program. The table below tells you what to do to go directly to a specific card.

PROMPT	Decimal point absent	Decimal point present
2 THRU 8	2	3
2 3 4 OR 5	4	5
6 7 OR 8	6	7
9 OR MORE	9	10

For the benefit of our newcomers: Loading the print code registers is not an easy chore. These are typically for HIR loading (R30 through R50) The rest are for OP nn loading. One is even both: R55 !
The best way to explain how to store these large numbers into the respective registers is by means of an example. Let's take R37:
Store the integer part first : 25 +/- STO 37. Then count the number of digits to the right side of the decimal point: here eleven. This "11" will be the negative exponent, as follows: 3633131617 +/- EE 11 +/- SUM 37.
If the number doesn't have a negative sign, you have to leave off, of course, the +/- before STO and also the +/- before EE but keep the one before SUM.

INSTRUCTIONS.

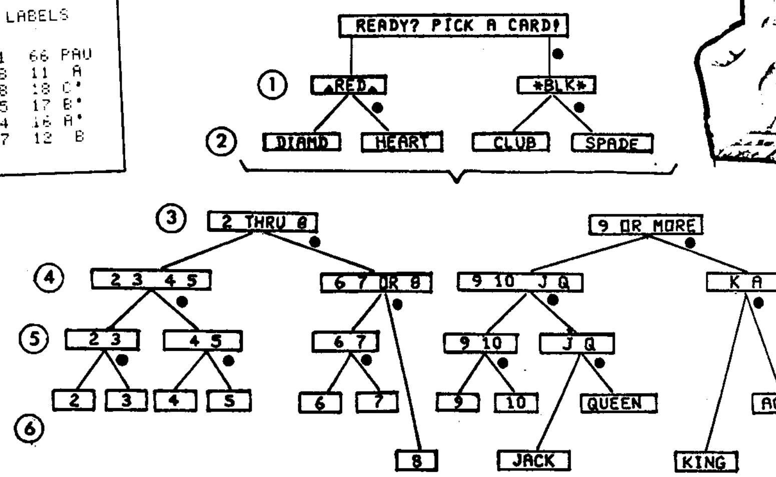
1. Press B to start. READY? PICK A CARD printed.
2. Victim selects a card.
3. Within 10 seconds press R/S with or without decimal point present.
4. Press A to continue and allow the PC100 to print the calculator's (or is it your?) choice.
5. For the "short cut" press 2nd A' instead of A.
   If you make a mistake, wait 10 seconds for the calculator to change its mind.
6. Go back to 3 until the complete card has been revealed.

NOTE: Registers 44 and 49 don't contain print code.
Program is recorded on one mag card, banks 1 and 3.

```
000 76 LBL      074 76 LBL      148 65 ×
001 66 PAU      075 17 B'       149 03 3
002 59 INT      076 44 SUM      150 95 =
003 09 9        077 09 09       151 17 B'
004 09 9        078 73 RC*      152 44 SUM
005 09 9        079 09 09       153 05 05
006 09 9        080 82 HIR      154 44 SUM
007 66 PAU      081 07 07       155 05 05
008 97 DSZ      082 69 OP       156 85 +
009 03 03       083 05 05       157 04 4
010 00 00       084 69 OP       158 03 3
011 01 01       085 00 00       159 75 -
012 43 RCL      086 44 SUM      160 67 EQ
013 55 55       087 00 00       161 01 01
014 82 HIR      088 42 STO      162 72 72
015 08 08       089 01 01       163 04 4
016 69 OP       090 69 OP       164 05 5
017 01 01       091 29 29       165 95 =
018 43 RCL      092 81 RST      166 17 B'
019 56 56       093 76 LBL      167 44 SUM
020 69 OP       094 16 A'       168 05 05
021 02 02       095 24 CE       169 01 1
022 43 RCL      096 69 OP       170 01 1
023 57 57       097 10 10       171 32 X:T
024 69 OP       098 85 +        172 43 RCL
025 03 03       099 43 RCL      173 05 05
026 69 OP       100 09 09       174 22 INV
027 05 05       101 75 -        175 77 GE
028 69 OP       102 03 3        176 01 01
029 00 00       103 08 8        177 05 05
030 43 RCL      104 95 =        178 04 4
031 01 01       105 92 RTN      179 00 0
032 22 INV      106 76 LBL      180 44 SUM
033 44 SUM      107 12 B        181 05 05
034 00 00       108 25 CLR      182 73 RC*
035 43 RCL      109 43 RCL      183 05 05
036 06 06       110 58 58       184 69 OP
037 42 STO      111 69 OP       185 03 03
038 05 05       112 01 01       186 69 OP
039 32 X:T      113 43 RCL      187 05 05
040 42 STO      114 59 59       188 12 B
041 09 09       115 69 OP
042 01 1        116 04 04
043 75 -        117 42 STO
044 43 RCL      118 03 03
045 04 04       119 03 3
046 54 )        120 00 0
047 76 LBL      121 42 STO
048 11 A        122 00 00
049 24 CE       123 42 STO
050 69 OP       124 09 09
051 10 10       125 18 C'
052 42 STO      126 65 ×
053 04 04       127 07 7
054 32 X:T      128 85 +
055 09 9        129 42 STO
056 42 STO      130 05 05
057 03 03       131 03 3
058 43 RCL      132 08 8
059 05 05       133 95 =
060 42 STO      134 42 STO
061 06 06       135 09 09
062 43 RCL      136 18 C'
063 09 09       137 65 ×
064 32 X:T      138 02 2
065 83 GO*      139 44 SUM
066 00 00       140 05 05
067 76 LBL      141 85 +
068 18 C'       142 44 SUM
069 73 RC*      143 09 09
070 09 09       144 95 =
071 82 HIR      145 44 SUM
072 06 06       146 05 05
073 01 1        147 18 C'
```

```
121.4571003324     30  Y? PI
118.15260013       31  CK A
3.007535171675     32  .RED.
-25.01624133016    33  DIAMD
-25.02317133537    34  HEART
3.005114272651     35  *BLK*
-25.00015274114    36  CLUB
-25.03633131617    37  SPADE
11.00003003723     38  2 TH
11.03541000973     39  RU 8?
15.000030004       40  2 3
15.000050006       41  4 5
15.000070008       42  6 7
15.032350009       43  OR 8
11.00012003235     45  9 OR
11.00030323517     46  MORE
15.00012000201     47  9 10
15.000250034       48  J Q
30.000260013       50  K A
25131526.          51  JACK
3441171731.        52  QUEEN
26243122.          53  KING
13151700.          54  ACE
2400.030243116     55  I MIND
1523133122.        56  CHANG
1716003045.        57  ED MY
35171316.          58  .READ
1513351673.        59  CARD?
```

LABELS		
001	66	PAU
048	11	A
068	18	C'
075	17	B'
094	16	A'
107	12	B



ALPHA DATA LIST.- In V4N3S35 of Display, Herbert Lauterbach published an intriguing program called RCL. It permitted to list on one line the alpha code contained in a register, the corresponding alpha and the register number. The program was written within less than 160 steps, such that the majority of the registers could be listed. I say MAJORITY, because, in spite of using only only R00 for IND functions and the HIRs for temporary storage, Herbert didn't include provisions in his program to also list R90 through R99. His program required 5 min 39 sec to list 11 test registers or almost 31 sec per register. I saw fit to speed it up somewhat (3 min 58 sec or about 21.6 sec per register) and to include provisions to list R90 through R99. I added the required impossibly-long words and dangling participles so loved by the practitioners of Goethe's beautiful language, and sent it off, Display-wards.
At the same time I distributed an English translation of both programs among the members of our local club. I didn't have to wait long to get several enhancements from the Snow brothers, from Norman Herzberg and from Bill Skillman. They were all faster and used less working registers. By using their tricks, I was able to write one more program which I submitted to PPX. Choosing "the best one" among so many versions is difficult. Each one has its peculiar qualities. The one by Bill Skillman is "friendly to the user." Although not the fastest of the pack: 24 sec per 10-digit register vs both Norman Herzberg's and Richard Snow's clocking in at 18 sec per register, it has the advantage that it does not require any special programming gymnastics to list register 90 through 99. It also does not require you to be present during a long listing to stop the program at appropriate places. You may pre-determine the starting and the stopping register.
Instructions:
1. *Enter the program with registers to be listed, banks 2, 3 and 4.*
2. *Enter this program, bank 1.*
3. *Enter number of lowest register to be listed and press A. Heading printed. Enter number of highest register to be listed and press R/S. This is for registers 05 through 89. Registers 00 through 04 used by program.*
4. *For registers 90 through 99, force bank 1 of program with registers to be listed into bank 2 by means of 2 +/-.*
5. *Repeat 2.*
6. *Repeat 3, but press B.*

```
ALPHA REG LIST:

  NUMERIC  REG ALPHA
1327332313. 90 ALPHA
  35172200. 91  REG
2724363762. 92 LIST:
1735241500. 93 ERIC
    314130. 94   NUM

000  76 LBL     036  00  00     068  52 EE      100  03  03     132  02  2      164  01  1
001  16 A'      037  42  42     069  92 RTN     101  43 RCL     133  85  +      165  00  0
002  65  x      038  93  .      070  76 LBL     102  94  94     134  04  4      166  95  =
003  00  0      039  02  2      071  11  A      103  69 OP      135  52 EE      167  16 A'
004  42 STO     040  44 SUM     072  42 STO     104  01  01     136  09  9      168  69 OP
005  02  02     041  02  02     073  00  00     105  43 RCL     137  22 INV     169  01  01
006  32 X:T     042  22 INV     074  32 X:T     106  93  93     138  87 IFF     170  69 OP
007  05  5      043  97 DSZ     075  22 INV     107  69 OP      139  05  05     171  05  05
008  42 STO     044  01  01     076  86 STF     108  02  02     140  01  01     172  69 OP
009  01  01     045  00  00     077  05  05     109  43 RCL     141  46  46     173  20  20
010  93  .      046  53  53     078  01  1      110  90  90     142  85  +      174  97 DSZ
011  01  1      047  43 RCL     079  00  0      111  69 OP      143  05  5      175  04  04
012  95  =      048  03  03     080  69 OP      112  04  04     144  52 EE      176  01  01
013  42 STO     049  22 INV     081  17  17     113  69 OP      145  04  4      177  28  28
014  03  03     050  67  EQ     082  43 RCL     114  05  05     146  95  =      178  06  6
015  75  -      051  00  00     083  90  90     115  09  9      147  22 INV     179  69 OP
016  59 INT     052  15  15     084  69 OP      116  69 OP      148  52 EE      180  17  17
017  85  +      053  09  9      085  00  00     117  17  17     149  69 OP      181  25 CLR
018  22 INV     054  75  -      086  69 OP      118  25 CLR     150  03  03     182  92 RTN
019  44 SUM     055  43 RCL     087  01  01     119  32 X:T     151  73 RC*     183  76 LBL
020  03  03     056  01  01     088  43 RCL     120  75  -      152  00  00     184  12  B
021  93  .      057  65  x      089  92  92     121  91 R/S     153  69 OP      185  42 STO
022  01  1      058  02  2      090  69 OP      122  95  =      154  04  04     186  00  00
023  49 PRD     059  95  =      091  03  03     123  94 +/-     155  16 A'      187  32 X:T
024  03  03     060  22 INV     092  43 RCL     124  42 STO     156  69 OP      188  03  3
025  49 PPD     061  28 LOG     093  91  91     125  04  04     157  02  02     189  00  0
026  02  02     062  65  x      094  69 OP      126  69 OP      158  43 RCL     190  94 +/-
027  49 PRD     063  43 RCL     095  02  02     127  24  24     159  03  03     191  44 SUM
028  02  02     064  02  02     096  69 OP      128  43 RCL     160  67  EQ     192  00  00
029  95  =      065  95  =      097  05  05     129  00  00     161  01  01     193  61 GTO
030  44 SUM     066  52 EE      098  98 ADV     130  16 A'      162  68  68     194  00  00
031  02  02     067  22 INV     099  69 OP      131  52 EE      163  65  x      195  76  76
032  22 INV
033  59 INT
034  22 INV
035  67  EQ
```

RECORDING INSTRUCTIONS:

PARTITION 10 OP 17. KEY IN PROGRAM WITH 196 STEPS, LOAD REGISTERS 90 - 94.

PARTITION 6 OP 17. RECORD SIDE 1 OF MAG CARD.

Continued on next page.

As you might have guessed, this was by no means the end of the story. Valentino Ducati saw my program in Display and found it still slow. So, one day he sent me his version called ALPHA SPEICHERAUFLISTING MARK II, an allusion to the name I had given it, without the Mark II. It was fast alright: 4 sec per register. But Valentino cheated to a certain extent: his program uses double the amount of paper, in that it prints the Alpha code and the Alpha itself on two separate lines, with the register number repeated on each line. But is is no ordinary program, not by a long shot. It has some of the cleverest program tricks I have seen in a long time. It had only one drawback in use: it always started the listing with R00. Now, you just offer such a challenge to Bill Skillman and he sacrifices a weekend on it to enhance it. It now starts with any register you want. Bill insisted on it, however, that I call this version ALPHA LIST MARK III.
The instructions for the Ducati-Skillman program are:

1. *Load the Alpha List program, side 1 only.*
2. *Select the proper partition, = the one of the program whose registers you want to list.*
3. *If that program has a side 2, force it into bank 2 with 2 +/-.*
4. *If that program has a side 3, force it into bank 3 with 3 +/-.*
5. *Enter starting register number and press A.*
   If you selected partition 0 OP 17, the display will flash 9999...99.
   If you selected 10 OP 17, the printer prints 1, asking for side 1.
   Insert side 1. If it missreads, force side 1 into bank 1 with 1 +/-. To stop the flashing press CLR and continue with R/S.
6. *Printer now prints 4, asking for side 4. Insert side 4. If it missreads, use the same manoeuver as in 5 above. If no side 4 is available, press R/S twice.*
   Now the listing starts, beginning with register n, which you entered in 5.
   If you want a repeat performance, the same list, press SBR 079. This time the listing will start with R01.

000	76	LBL	023	82	HIR	046	00	00	069	59	59	092	01	01	115	20	20	138	00	0
001	16	A'	024	18	18	047	74	74	070	43	RCL	093	16	16	116	73	RC*	139	00	0
002	53	(	025	29	CP	048	42	STO	071	00	00	094	48	EXC	117	00	00	140	16	A'
003	22	INV	026	22	INV	049	90	90	072	42	STO	095	00	00	118	67	EQ	141	95	=
004	57	ENG	027	67	EQ	050	01	1	073	90	90	096	67	EQ	119	01	01	142	69	OP
005	85	+	028	67	EQ	051	99	PRT	074	04	4	097	01	01	120	05	05	143	04	04
006	01	1	029	22	INV	052	04	4	075	99	PRT	098	05	05	121	43	RCL	144	73	RC*
007	85	+	030	58	FIX	053	94	+/-	076	94	+/-	099	69	OP	122	00	00	145	00	00
008	28	LOG	031	69	OP	054	22	INV	077	22	INV	100	06	06	123	52	EE	146	61	GTO
009	59	INT	032	16	16	055	96	WRT	078	96	WRT	101	69	OP	124	01	1	147	00	00
010	65	×	033	88	DMS	056	09	9	079	98	ADV	102	03	03	125	94	+/-	148	99	99
011	02	2	034	82	HIR	057	42	STO	080	01	1	103	69	OP	126	82	HIR	149	76	LBL
012	54	)	035	18	18	058	10	10	081	69	OP	104	05	05	127	08	08	150	67	EQ
013	92	RTN	036	82	HIR	059	73	RC*	082	04	04	105	43	RCL	128	59	INT	151	25	CLR
014	76	LBL	037	04	04	060	10	10	083	29	CP	106	00	00	129	16	A'	152	35	1/X
015	11	A	038	67	EQ	061	72	ST*	084	82	HIR	107	75	-	130	52	EE	153	52	EE
016	93	.	039	67	EQ	062	90	90	085	13	13	108	82	HIR	131	02	2	154	22	INV
017	82	HIR	040	32	X:T	063	97	DSZ	086	67	EQ	109	14	14	132	85	+	155	57	ENG
018	03	03	041	98	ADV	064	90	90	087	00	00	110	95	=	133	82	HIR	156	24	CE
019	01	1	042	09	9	065	51	BST	088	94	94	111	67	EQ	134	18	18	157	92	RTN
020	37	P/R	043	09	9	066	97	DSZ	089	42	STO	112	01	01	135	22	INV	158	98	ADV
021	69	OP	044	22	INV	067	10	10	090	00	00	113	58	58	136	59	INT	159	92	RTN
022	00	00	045	67	EQ	068	00	00	091	61	GTO	114	69	OP	137	52	EE			

<u>1 EXC 2, 2 EXC 1 ROUTINES.</u> In v1n2, 1979 of the PPX Exchange our club published an article called PROGRAMMER'S POINT OF VIEW. By means of twelve different routines we demonstrated how you might be able to guess an individual's profession by taking a look at his ideosincrasies in programming! PPX published a follow-up in v3n6, 1979.In it they published the best of many new categories I received in the mail. Here again are a few "jewels" sent to me by Bill Beebe from Lilburn, Georgia:
000: GTO 006 LBL A RST STO 01 1 STF IND 01 IFF 02 017 2 ABS RTN Which Bill calls a flagwaver's delight. LBL A √x + 1/X INT = INT R/S for people who cover their tracks, and LBL A 1/X INV LNX INT R/S which Bill calls his "cat-skinner" on the premise that there is more than one way to do that.
As most of you will remember, the idea is that your program change a "1" in the display into a "2" and a "2" into a "1".
PPX forgot to mention that the idea for this "cutie" comes from my friend Bill Kolb, the local HP club coordinator. This then to render to Caesar.....
Sorry for the omission, Bill.

REACTION TIME TEST.

In Personal Computing, Aug 1979, pp 58-66, John Walker published a TI-58/59 program called JUMP HOLE GUNNER. It had a rather unique feature I had never seen in a TI-58/59 program: the program generated random time clues, to which the user had to react within a certain time window. That meant, both reacting too fast or too slow got you in trouble.
The time duration of the window, nor the onset of the window were stated. One had to learn the hard way, from experience.
The clues were not only visual, but also audible! The author recommended to place a radio receiver close to the TI-59/PC100 and tune the radio to the static in-between two stations on AM. (This is no April joke this time. Honest. It works perfectly!) In a short while you are able to recognize parts of program execution by its sounds. In fact, this can be done with any program.
Then Karl-Joseph Meusch sent me a program called REAKTIONS TEST. It used about the same type of timing loop as Jump Hole Gunner:
The program runs through a sequence of 10000000000 x:t 20000000000....etc. up to 60000000000 x:t. You stop it somewhere along the line with R/S. In the display you will have a digit between 1 and 6 followed by from 0 to 10 zeros. If you now press E, starting at step 108, the program extracts the leading digit and the number of zeros. How that is done can be found out by doing LBL E manually from the key board. You will be surprised to find out how simple it is. It might be a good challenge among the members to produce a general routine for this kind of program trick. Besides in games you might find use for it in "serious" programs.
The instructions for this game are simple:

1. READ-IN TWO MAG CARDS, BANKS 1, 2, 3 AND 4.
2. PRESS A. SEE INSTRUCTIONS PRINTED. THE ANOTATION METS AND 9213 ARE MY INITIALS AND HOUSE NUMBER. THESE MAY BE CHANGED TO YOURS BY ALTERING THE CONTENTS OF REGS 09 AND 29.
3. AS SOON A YOU SEE THE WORDS -¤-STOP-¤-¤-STOP-¤- PRINTED ON THE TAPE, PRESS R/S AND HOLD THAT KEY DOWN FOR AT LEAST ONE FULL SECOND. THIS TO ALLOW A TIMING LOOP TO COMPLETE ITSELF.
4. TO OBTAIN YOUR REACTION TIME FOR TRIAL ONE, PRESS E. THE REACTION TIME WILL BE PRINTED AND THE GAME CONTINUES WITH 3, ABOVE.

   YOU WILL SEE REPEATED FLASHINGS OF .1111111 IN THE DISPLAY. THIS ONLY TO DISTRACT YOU AND WARN YOU OF AN IMMINENT PRINTING OF "STOP".
5. AFTER 10 TRIALS THE PROGRAM WILL PRINT OUT YOUR MEAN REACTION TIME, FOLLOWED BY A RATHER SUBJECTIVE VERBAL EVALUATION OF YOUR PERFORMANCE AS AN ALERT HUMAN BEING.

   A SAMPLE OF WHAT TO EXPECT IS PRINTED IN THE LOWER LEFT CORNER.

   THERE ARE TEN POSSIBLE VERBAL EVALUATION TERMS. THEY ARE STORED IN REGISTERS 40 THROUGH 59. YOU CAN, OF COURSE, ALTER ANY OR ALL OF THEM.

TO MAKE THE GAME MORE INTERESTING, PLACE A RADIO RECIVER CLOSE TO THE TI-59 AS EXPLAINED ABOVE.

This program borrows heavily from REAKTIONS TEST and parts of it are recognizable as pertaining to JUMP HOLE GUNNER. Thanks to both authors.

A few notes:

1. This program uses for timing a digit followed by only eight zeros, rather than the costumary ten. This to shorten the timing loop somewhat and improve timing accuracy.
2. LBL C' is undefined and produces a blinking zero. (steps 282-283)
3. Arrows indicate entrance and exit points in the program. They were added to make reading the program a little easier.

```
********************
**REACTION TEST*****
********************
******METS 9213*****
********************

AFTER EACH -?STOP?-
PRINT-OUT, PRESS R/S
QUICKLY. THEN PRESS E
FOR TIME EVALUATION.

-?-STOP-?--?-STOP-?-
        0.20     SEC
-?-STOP-?--?-STOP-?-
        0.20     SEC
-?-STOP-?--?-STOP-?-
        0.17     SEC
-?-STOP-?--?-STOP-?-
        0.17     SEC
-?-STOP-?--?-STOP-?-
        0.27     SEC
-?-STOP-?--?-STOP-?-
        0.23     SEC
-?-STOP-?--?-STOP-?-
        0.50     SEC
-?-STOP-?--?-STOP-?-
        0.27     SEC
-?-STOP-?--?-STOP-?-
        0.33     SEC
-?-STOP-?--?-STOP-?-
        0.33     SEC

      0.2667      %

********************
******RESULTS!******
******SPEEDY G.*****
********************
```

ALPHA REG LIST:

NUMERIC	REG	ALPHA
5938173736.	9	+METS
2132350037.	10	FOR T
2438170017.	11	IME E
4213274113.	12	VALUA
3724323940.	13	TION.
3441241526.	14	QUICK
2745403723.	15	LY.TH
1739003335.	16	EN PR
1736360017.	17	ESS E
3335243937.	18	PRINT
2032413757.	19	-OUT,
3335173636.	20	PRESS
35633600.	21	R/S
1321371735.	22	AFTER
17131523.	23	EACH
20733637.	24	-?ST
3233732000.	25	OP?-
5151351713.	26	**REA
1537243231.	27	CTION
37173637.	28	TEST
12030204.	29	9213
5151515151.	30	*****
37323200.	31	TOO
2113363773.	32	FAST?
38411523.	33	MUCH
37323200.	34	TOO
3627324373.	35	SLOW?
2073203637.	36	-?-ST
3233207320.	37	OP-?-
5935173641.	38	*RESU
2737366259.	39	LTS!*
4313261700.	40	WAKE
4133737373.	41	UP???
3335131537.	42	PRACT
2415177373.	43	ICE??
5937323537.	44	*TORT
3224361773.	45	OISE?
2713301700.	46	LAME
1641152673.	47	DUCK?
3627324300.	48	SLOW
3332261773.	49	POKE?
4317272757.	50	WELL,
4317272773.	51	WELL?
1315151733.	52	ACCEP
3713142717.	53	TABLE
5936331717.	54	*SPEE
1645002240.	55	DY G.
4217354500.	56	VERY
2232321673.	57	GOOD?
4532410035.	58	YOU R
1315173573.	59	ACER?

```
000 01 1        113 86 STF      224 97 DSZ      336 01 1
001 00 0        114 01 01       225 06 06       337 00 0
002 00 0        115 42 STO      226 02 02       338 42 STO
003 00 0        116 01 01       227 24 24       339 05 05
004 00 0        117 52 EE       228 43 RCL      340 44 SUM
005 00 0        118 95 =        229 36 36       341 08 08
006 00 0        119 52 EE       230 69 OP       342 42 STO
007 00 0        120 00 0        231 01 01       343 06 06
008 00 0        121 00 0        232 69 OP       344 44 SUM
009 32 X:T      122 48 EXC      233 03 03       345 06 06
010 02 2        123 01 01       234 43 RCL      346 16 A'
011 00 0        124 55 ÷        235 37 37       347 43 RCL
012 00 0        125 43 RCL      236 69 OP       348 26 26
013 00 0        126 01 01       237 02 02       349 69 OP
014 00 0        127 95 =        238 69 OP       350 01 01
015 00 0        128 28 LOG      239 04 04       351 43 RCL
016 00 0        129 42 STO      240 69 OP       352 27 27
017 00 0        130 02 02       241 05 05       353 69 OP
018 00 0        131 06 6        242 81 RST      354 02 02
019 32 X:T      132 32 X:T      243 86 STF      355 43 RCL
020 03 3        133 52 EE       244 04 04       356 28 28
021 00 0        134 95 =        245 98 ADV      357 69 OP
022 00 0        135 52 EE       246 06 6        358 03 03
023 00 0        136 00 0        247 07 7        359 69 OP
024 00 0        137 00 0        248 58 FIX      360 05 05
025 00 0        138 22 INV      249 04 04       361 16 A'
026 00 0        139 52 EE       250 69 OP       362 43 RCL
027 00 0        140 42 STO      251 04 04       363 09 09
028 00 0        141 03 03       252 43 RCL      364 69 OP
029 32 X:T      142 67 EQ       253 04 04       365 02 02
030 04 4        143 00 00       254 55 ÷        366 43 RCL
031 00 0        144 94 94       255 01 1        367 29 29
032 00 0        145 32 X:T      256 00 0        368 69 OP
033 00 0        146 43 RCL      257 95 =        369 03 03
034 00 0        147 01 01       258 69 OP       370 69 OP
035 00 0        148 32 X:T      259 06 06       371 05 05
036 00 0        149 77 GE       260 22 INV      372 16 A'
037 00 0        150 01 01       261 58 FIX      373 98 ADV
038 00 0        151 63 63       262 98 ADV      374 02 2
039 32 X:T      152 43 RCL      263 16 A'       375 05 5
040 05 5        153 01 01       264 43 RCL      376 42 STO
041 00 0        154 75 -        265 38 38       377 00 00
042 00 0        155 93 .        266 69 OP       378 04 4
043 00 0        156 09 9        267 02 02       379 42 STO
044 00 0        157 85 +        268 43 RCL      380 02 02
045 00 0        158 43 RCL      269 39 39       381 04 4
046 00 0        159 02 02       270 69 OP       382 42 STO
047 00 0        160 55 ÷        271 03 03       383 01 01
048 00 0        161 01 1        272 69 OP       384 73 RC*
049 32 X:T      162 00 0        273 05 05       385 00 00
050 06 6        163 95 =        274 43 RCL      386 84 OP*
051 00 0        164 55 ÷        275 04 04       387 01 01
052 00 0        165 03 3        276 32 X:T      388 69 OP
053 00 0        166 95 =        277 03 3        389 30 30
054 00 0        167 44 SUM      278 00 0        390 97 DSZ
055 00 0        168 04 04       279 77 GE       391 01 01
056 00 0        169 32 X:T      280 02 02       392 03 03
057 00 0        170 03 3        281 84 84       393 84 84
058 00 0        171 06 6        282 00 0        394 69 OP
059 32 X:T      172 01 1        283 18 C'       395 05 05
060 69 OP       173 07 7        284 32 X:T      396 97 DSZ
061 00 00       174 01 1        285 55 ÷        397 02 02
062 86 STF      175 05 5        286 32 X:T      398 03 03
063 03 03       176 69 OP       287 75 -        399 81 81
064 43 RCL      177 04 04       288 01 1        400 98 ADV
065 34 34       178 25 CLR      289 00 0        401 61 GTO
066 69 OP       179 32 X:T      290 03 3        402 02 02
067 02 02       180 58 FIX      291 35 1/X      403 02 02
068 43 RCL      181 02 02       292 95 =        404 87 IFF
069 35 35       182 69 OP       293 34 √X       405 04 04
070 69 OP       183 06 06       294 65 ×        406 02 02
071 03 03       184 22 INV      295 09 9        407 43 43
072 69 OP       185 58 FIX      296 75 -        408 22 INV
073 05 05       186 22 INV      297 01 1        409 87 IFF
074 86 STF      187 87 IFF      298 00 0        410 02 02
075 01 01       188 03 03       299 95 =        411 02 02
076 61 GTO      189 01 01       300 59 INT      412 02 02
077 01 01       190 94 94       301 50 I×I      413 06 6
078 86 86       191 06 6        302 65 ×        414 44 SUM
079 76 LBL      192 44 SUM      303 02 2        415 04 04
080 16 A'       193 04 04       304 85 +        416 43 RCL
081 43 RCL      194 22 INV      305 04 4        417 31 31
082 30 30       195 86 STF      306 00 0        418 69 OP
083 69 OP       196 03 03       307 95 =        419 00 00
084 01 01       197 22 INV      308 42 STO      420 69 OP
085 69 OP       198 97 DSZ      309 00 00       421 02 02
086 02 02       199 05 05       310 73 RC*      422 43 RCL
087 69 OP       200 02 02       311 00 00       423 32 32
088 03 03       201 43 43       312 69 OP       424 69 OP
089 69 OP       202 43 RCL      313 02 02       425 03 03
090 04 04       203 08 08       314 69 OP       426 69 OP
091 69 OP       204 50 I×I      315 20 20       427 05 05
092 05 05       205 23 LNX      316 73 RC*      428 61 GTO
093 92 RTN      206 50 I×I      317 00 00       429 01 01
094 69 OP       207 42 STO      318 69 OP       430 86 86
095 00 00       208 08 08       319 03 03
096 02 2        209 22 INV      320 69 OP
097 94 +/-      210 59 INT      321 05 05
098 44 SUM      211 65 ×        322 16 A'
099 04 04       212 03 3        323 25 CLR
100 43 RCL      213 01 1        324 22 INV
101 33 33       214 95 =        325 86 STF
102 69 OP       215 59 INT      326 04 04
103 01 01       216 42 STO      327 92 RTN
104 61 GTO      217 06 06       328 76 LBL
105 00 00       218 29 CP       329 11 A
106 62 62       219 86 STF      330 86 STF
107 76 LBL      220 02 02       331 01 01
108 15 E        221 09 9        332 98 ADV
109 87 IFF      222 35 1/X      333 25 CLR
110 01 01       223 66 PAU      334 42 STO
111 04 04                       335 04 04
112 04 04

080 16 A'
108 15 E
329 11 A
```

NEWCOMER'S CORNER. *In V5N1p12 I said that HIR 05 through HIR 08 is the same as OP 01 through OP 04. To a certain extent this is true. But if you want to load print code into the HIRs, you will have to follow certain rules. For print code loaded into HIRs, the three left-most digits are ignored. Thus, you just put "dummy" digits in those positions, such as 999. Stay away form zeros, because if they become leading ones, they will cease to be "dummy." Any other digit or combination of digits may be used, though. The next 10 digits following the three left-most ones will be recognized as print code in five pairs. This makes a total of 13 digits, the maximum you can store in any register, HIR or ordinary data register. In HIR loading, only the order of the digits is important, not where the decimal point is placed. Let's try to experiment with these ideas. Write in user memory*

LBL A HIR 06 R/S LBL B OP 05 R/S

Now we are going to create the print code to print five Y's (code 45). So we key in key board mode: 123454545 X 10000 = STO 00 4545 SUM 00 RCL 00
What we see in the display now is 1.2345455 12 which is what we entered 1234545454545 but because the display cannot show more than 10 digits, the calculator goes into EE mode automatically for numbers with more than 10 digits.
We now load that number into HIR 06 by pressing A, followed by printing with OP 05 by pressing B. As expected, we have five Y's in sector 2 of the tape.
Now, to show that the position of the decimal point has no influence we key RCL 00 DIV 1000 = and we see in the display 1234545455. The last 5 is a rounding of 4.545. Again we press A, followed by B. Same result, five Y's printed in the same sector of the tape. For our next experiment, let's add routine C:

LBL C OP 02 R/S *also to be written in user memory.*

We are now almost ready to do a two-word trick. The code for HI is 2324, while the code for LO is 2732. We load this into R00, so we will be able to recall it a few times if needed. We load this code as 2324.9992732
So we key 2324 STO 00 .9992732 SUM 00
If we now key RCL 00 INV INT A B we see the word LO printed in sector 2 of the tape. This, of course, from the fractional part .9992732 in which the three 9's were ignored. If, on the other hand, we key RCL 00 INT C B we see the word HI printed, from the integer part of our number, i.e. 2324.
This is a simple and effective way to store two words in one register and selectively print them once by means of HIR loading, once by means of regular OP loading. You might wonder why we couldn't load the print code as 2324.2732 and subsequently separate both words either by INT or by means of INV INT,as we did before and print both by a regular OP 02 loading. Yes, that would be possible, but in the INV INT case we obtain a fraction as print code. Before we can use it as such we will have to multiply it by 10000 or, a little shorter,by 4 INV LOG. But we still have to use an extra 4 steps. In the case of HIR loading, the HIR doesn't care where the decimal point is, it only requires three "dummy" left-most digits. So, which of the two methods is more economical in program steps?

Let us now examine how code stored with, for example, OP 02 is modified in HIR 6. Key in the following routine, after the R/S of routine C:

LBL D HIR 16 R/S *This routine, as you can see, will recall the contents of HIR 6. Now, enter the five Y's again as 4545454545 OP 02. Then press D. The display now shows .0045454545 which means that our entered number was divided by 10^{12}. The last two 45 digits are not visible in the display, but if you multiply the displayed number by 1000 you will prove to yourself that they are indeed in the display REGISTER. By examining the .0045454545 display we see that the calculator did what we said in this article: create three left-most dummy digits in front of the entered number. By imitating this rule we were able to load directly into HIR 6. Here the zeros are not leading ones and therefore alright. But to be on the safe side, always use a digit different from zero.*

PRINT CODE CONVERTER.- Robert Snow's original print code converter in VN10p2 and the modified one in V5N1p2 can only be used with positive integers smaller than 99999. Moreover, R09 has to be zeroed prior to calling SBR PRT. Bill Skillman sent me two modified versions that will work with -9999 ≤ N ≤ 99999. There is furthermore no requirement to zero R09. The routine uses R08 and R09, but no flags. It is 11 steps longer than the original Snow routine. Here follows the best one of the two:

Note to our newcomers: To test this routine, place a number between -9999 and 99999 in the display and press SBR PRT. When the display returns with the required print code, press OP 02 OP 05. If the printer is attached, you will see your entry printed in sector 2. Even an entered - sign will be there.
This SBR is especially handy to enable you to print any 5-digit number anywhere on the paper tape. It can be used as a subroutine in your main program.

000	76	LBL	014	09	09	028	85	+	042	95	=
001	99	PRT	015	32	X:T	029	28	LOG	043	65	×
002	29	CP	016	50	I×I	030	59	INT	044	01	1
003	67	EQ	017	55	÷	031	65	×	045	00	0
004	00	00	018	28	LOG	032	01	1	046	97	DSZ
005	27	27	019	59	INT	033	00	0	047	08	08
006	32	X:T	020	42	STO	034	00	0	048	00	00
007	22	INV	021	08	08	035	49	PRD	049	26	26
008	77	GE	022	69	OP	036	09	09	050	25	CLR
009	00	00	023	28	28	037	02	2	051	48	EXC
010	12	12	024	22	INV	038	75	-	052	09	09
011	02	2	025	28	LOG	039	59	INT	053	92	RTN
012	00	0	026	85	+	040	44	SUM			
013	42	STO	027	01	1	041	09	09			

Mathematical diversions.- On the last page you will find a contribution by Sam Allen on a program to solve the "pegboard problem." The instructions for running this program are:

1. *Load the program. (TI-58 or TI-59)*
2. *Reset the program counter.*
3. *Clear all memories.*
4. *Load N, the dimension of the array, into R14.*
5. *Press R/S to run.*

If a solution is found, the calculator will halt with N in the display. If the array has no solution the program eventually will crash. (Good luck, TI-58 users!) The array will be found in R01 through R08, where the number of the register corresponds to the row of the array. The row elements are packed in reverse order : Column N to the right of the decimal point , followed by column N-1, etc.
Solutions exist for N=5. It is not known whether or not there are solutions for N=8. A related program appeared recently in Byte (FEB 1979, p 146-148) called "The Eight Calculating Queens." It was written by Bill White for the SR-56. It is very easy to translate it for the TI-58/59. It could even be done for the SR-52. On the SR-56 the 100-step program runs about 14 hours to produce all 92 possible solutions of that chess problem.

MATHEMATICAL DIVERSIONS.- During the summer of 1979, Samuel G. Allen brought to my attention a nice problem called "The Pegboard Problem." It was once marketed commercially as a board drilled with an 8 X 8 array of holes, which were to be filled by 64 pegs of 8 different colors, in such a way that no row, column, or diagonal contained two pegs of the same color. The manufacturer even offered a $ 1000 prize to anyone who could come up with a solution!
To date only a pegboard of 5 X 5 is definitely known to have solutions. That is what Sam wrote me in the beginning. But then came the big surprise: after verifying that there is no solution to the 6 X 6 board, Sam triumphantly announced that he had found solutions to a 7 X 7 pegboard! Those solutions are reproduced here. That is, the only solution the calculator (a TI-58) came up with, is the one in the upper right corner. The others were derived from this one by transposition.

```
7 6 5 4 3 2 1
5 4 3 2 1 7 6
3 2 1 7 6 5 4
1 7 6 5 4 3 2
6 5 4 3 2 1 7
4 3 2 1 7 6 5
2 1 7 6 5 4 3
```

```
7 6 5 4 3 2 1
4 3 2 1 7 6 5
1 7 6 3 4 3 2
5 4 3 2 1 7 6
2 1 7 6 5 4 3
6 5 4 3 2 1 7
3 2 1 7 6 5 4
```

```
7 6 5 4 3 2 1
3 2 1 7 6 5 4
6 5 4 3 2 1 7
2 1 7 6 5 4 3
5 4 3 2 1 7 6
1 7 6 5 4 3 2
4 3 2 1 7 6 5
```

```
7 6 5 4 3 2 1
2 1 7 6 5 4 3
4 3 2 1 7 6 5
6 5 4 3 2 1 7
1 7 6 5 4 3 2
3 2 1 7 6 5 4
5 4 3 2 1 7 6
```

Then Sam devoted some 120 hours running time to finding solutions to an 8 X 8 board, but to no avail. I have a hunch that it will take an inordinate amount of time on a TI-58, or a TI-59 for that matter, to produce an 8 X 8 array solution. But then, with the right algorithm, who knows... Here then follows Sam's program for solving pegboards of N X N dimension in which he used some very clever packing of data registers.

```
000  01   1
001  44  SUM
002  10   10
003  25  CLR
004  42  STO
005  11   11
006  01   1
007  44  SUM
008  11   11
009  43  RCL
010  14   14
011  42  STO
012  12   12
013  22  INV
014  87  IFF
015  40  IND
016  12   12
017  00   00
018  71   71
019  97  DSZ
020  12   12
021  00   00
022  13   13
023  97  DSZ
024  11   11
025  00   00
026  51   51
027  01   1
028  22  INV
029  44  SUM
030  10   10
031  43  RCL
032  14   14
033  42  STO
034  11   11
035  86  STF
036  01   01
037  86  STF
038  02   02
039  86  STF
040  03   03
041  86  STF
042  04   04
043  86  STF
044  05   05
045  86  STF
046  06   06
047  86  STF
048  07   07
049  86  STF
050  08   08
051  73  RC*
052  10   10
053  65   ×
054  01   1
055  00   0
056  75   -
057  22  INV
058  59  INT
059  72  ST*
060  10   10
061  95   =
062  42  STO
063  12   12
064  22  INV
065  86  STF
066  40  IND
067  12   12
068  61  GTO
069  00   00
070  19   19
071  43  RCL
072  10   10
073  42  STO
074  00   00
075  22  INV
076  97  DSZ
077  00   00
078  01   01
079  40   40
080  73  RC*
081  00   00
082  42  STO
083  13   13
084  43  RCL
085  14   14
086  42  STO
087  09   09
088  43  RCL
089  12   12
090  32  X:T
091  61  GTO
092  00   00
093  99   99
094  22  INV
095  97  DSZ
096  09   09
097  00   00
098  75   75
099  01   1
100  00   0
101  49  PRD
102  13   13
103  43  RCL
104  13   13
105  75   -
106  22  INV
107  59  INT
108  42  STO
109  13   13
110  95   =
111  22  INV
112  67   EQ
113  00   00
114  94   94
115  43  RCL
116  09   09
117  32  X:T
118  43  RCL
119  11   11
120  67   EQ
121  00   00
122  19   19
123  75   -
124  32  X:T
125  95   =
126  50  I×I
127  32  X:T
128  43  RCL
129  10   10
130  75   -
131  43  RCL
132  00   00
133  95   =
134  67   EQ
135  00   00
136  19   19
137  61  GTO
138  00   00
139  75   75
140  43  RCL
141  12   12
142  74  SM*
143  10   10
144  86  STF
145  40  IND
146  12   12
147  01   1
148  00   0
149  22  INV
150  64  PD*
151  10   10
152  43  RCL
153  14   14
154  32  X:T
155  43  RCL
156  11   11
157  22  INV
158  67   EQ
159  00   00
160  06   06
161  43  RCL
162  10   10
163  77   GE
164  01   01
165  67   67
166  81  RST
167  91  R/S
168  81  RST
```

Remember to synthesize DSZ 11, for example, as DSZ A and DSZ 12 as DSZ B, and the address following this as GTO nnn, after which you go back and delete the GTO.

I have several of these mathematical diversional problems. If members show interest we could try solutions for them. In the mean time, please let me know what you think of this one and if you have a better solution.

See you next month. *Cordially,* Maurice