NEWSLETTER OF THE TI PERSONAL PROGRAMMABLE CALCULATOR CLUB,
9213 Lanham Severn Road, LANHAM, Maryland, 20801.

Yes, you are NOT seeing double. This IS a double issue: v5n4/5. I had so much good and "hot" material available, that it would have been a crime not to publish it very soon. This will, of course, allow me an extended "rest" period. I will still answer all your many letters and send programs to the reviewers. But don't expect the next issue within less than 10 weeks from now. I hope this enlarged issue, with its many good articles, will keep you sufficiently busy during that period.

The highlight of this issue is, without a doubt, the KEYCODE TRANSLATIONS discovery by John Mairs, on page 18. I hope it will stimulate a lot of research. Another high point, in my opinion, is Richard Snow's 13-DIGIT ALPHA REGISTER PRINT. It does everything what the Alpha programs in v5n3 do, only this one senses if your program intended HR or OP printing and selects automatically between the two! It works in 99% of the cases. As a bonus, it also is a good straightforward 13-digit printer-lister.

The two LABEL-TO-DIRECT-ADRESSES programs did not make it for this issue. There are still a few points to be ironed out. But fear not, they are working remarkably well.

To meet the many requests by surveyors-members, Frank Blachly presents his TRAVERSE program. It does (almost) everything a surveyor needs. Frank will also have an SR-52 version for next time.

Lots of members wondered how Richard Snow developed his ALPHABETICAL SORT program in v5n2p6. On page 5 and 6 Richard explains it. I had, accidently, ascribed the program to his brother Robert. My apologies to both of them.

Richard Nelson, editor of the HP PPC JOURNAL, gave our club a fantastic plug in the Feb-Mar 80 issue of the journal. The TI PPC CLUB thanks Richard for this. About fifteen bilinguals (RPN-AOS) have joined our ranks as a result of it. I had told Richard about the calendar printing contest we once had in 52-Notes and proposed this as a basis of some friendly competition; now that the RPN programmers have a more "seaworthy" machine in the HP-41C. Richard accepted the challenge. It will be interesting to see if anybody can beat Panos Galidas' record of 2 minutes 38.6 seconds to print one full year. The best program in RPN so far is by John Kennedy with a running time of slightly less than 6 1/2 minutes. But don't let that fool you. Those figures are likely to be coming down very soon. Their major effort in this area has been in memory reduction rather than speed. Get those TI-59's fired up and let's show them!!!

Two other editors are willing to help our club grow: Darrell Huff, who writes the Calcu-letter column in Popular Science and Jim Mc Dermott, a Special-Features editor with EDN, the prestigious Electronics Design News. Jim is going to mention our club in a special article on calculators in the June 20th issue. My most cordial thanks to both of them.

To satisfy many requests, we went to pre-punched paper. If it doesn't provoke any violent reactions, we will stay with that format in future.

Maurice E.T. Swinnen.
OTHER TI USERS CLUBS.- The Recreational Programmer, P.O. Box 2571, 3013 Cameron
Kalamazoo, MI 49003, has ceased to publish and is in the
process of refunding left-over contributions to its members.

The MIKRO-TASCHEN-COMPUTER-ANWENDER-CLUB (MICAC), formerly the German Chap-
ter of the HP-65 Users Club. Newsletter is called DISPLAY. Editor is Heinrich
Schnepp, Buchenweg 24, D-5000 Koeln 40, West Germany. (Koeln is German for
Cologne) Has been publishing for 6 years now. Appears about 6 times a year with
a giant issue of at least 100 pages. Last issue, July-Oct 79 had 120 pages.
Supports all programmables, HP and TI. Most programs in German but sometimes
Heinrich makes an exception and publishes a program I sent him for his own
enjoyment. (in English) The quality is about the highest you can expect.
Dues are $24.00 US/year. Well worth it if you are bilingual in both senses
of the word, German-English, RPN-AOS.

GESPRO. Renamed PPX. Published by GESPRO GMBH, Postfach 330112, Koblenz,
D-5400, West Germany. Editor is Dipl-Ing(FH)Wolfgang Bauer and his collaborators
Dr. Gerhard R. Eiden, & Ing(grad) Jochen Weber. Appears 8 times per year at
48.00 DM (about $24.00 US) plus air mail postage. Copyrighted. Last issue, Jan
80, had 48 pages. But nothing is reduced in size, such that many pages contain
long listings, as, e.g. pages 6 through 9 filled with a downloading of the TI-
58/59 firmware! Programs published are excellent with good documentation. Seems
to be supported somewhat by TI-Deutschland, judging by the ad on the last two
pages. Wolfgang has been promising his readers since last year that they would
get access to the "thousands of programs" from PPX in Lubbock. In the last issue
he asks his members to have a little more patience: "Eventually we will succeed." Has
a program exchange and sale at about 9.50 DM each. This is the 2nd volume.

TI SOFTWARE CLUB WALDKAPPEL.- P.O. Box 46, D-3445, Waldkappel, West Germany.
Should appear 8 times per year, but its editor, H. Roeske, has not yet been able
to accomplish this. Inexpensive program exchange at 1.50 to 6.00 DM each. Dues
are in total 42.00 DM/year. The newsletter itself is of less quality than both
Display and Gespro-PPX.

TI SOFTWARE CLUB PLEWNIA.- Editor is Peter Poloczek. Newsletter is higher
in quality than Waldkappel, but less than the first two. Some very good hardware
articles. Appears 6 times/year at 40.00 DM/year. Program exchange, but more expensive
than Waldkappel. Sells also paper, modules, etc. Address: Kalb, Hauptstrasse
72, D-6000, Frankfurt/Main, West Germany.

INTERESSENGEMEINSCHAFT PROGRAMMIERBAREN ANWENDER (IGPA) Editor is Thomas
Brettenger, Schillerstrasse 13, D-6452, Hainburg, West Germany. The least expensive
of them all. 20.00 DM/year and programs at 2.00 to 3.00 DM each. Has lots
of good programs and routines for the TI-58.

AG-59.- Editor Bernhard Fink, Argelanderstrasse 74, D-5300, Bonn, West Ger-
many. Will appear at irregular intervals at 10.00 DM each issue. Only the first
issue is out. Editor says that his goal is not hardware nor software but "brain-
ware." He also stresses the point that this is not a club, but a, what he calls
Arbeitsgemeinschaft TI-59, hence the AG-59. Only actively contributing members
are kept on the rolls.

ZEPPA.- Another "elite" group, this time without formal dues. Only active contributions in the form of programs, routines, algorithms. Prospective members are
chosen by the group for his/her past record as a programmer. It is very difficult
to become a member. Editor: H. Zupp, Subbelstrasse 30, D-5000, Koeln 30, West
Germany. Quality of programs and routines very high.

TI-59/PC100 TRICKS.- A series of compiled tricks edited by Harald M. Otto,
Bad Rothenfelde, West Germany. Some original but many from 52-Notes and from our
local Washington DC club. His three booklets are nicely printed. Appear whenever
Harald has enough material to justify a printing. Last issue was December 1979.

TI SOFTWARE EXCHANGE.- As opposed to all the above clubs in Germany who re-
quire programs and correspondence to be in German, this club in Belgium edits an
excellent newsletter in English. They also have a program exchange, because PPX
does not accept members outside the US, Canada and Mexico. The editor is Jean Verswijvelen, who is a physicist, aided by Robert Broeckx, Annie Debaere and Thomas Coppens, all three mathematicians. The address is Selstbaan 24, B-2080 Kapellen, Belgium. Dues are 425 Bfr or about $14.00 US. Their catalog contains about 270 good, mostly math, programs. Newsletter appears 4 times per year.

BRITISH TI USERS CLUB.- Editor Philip R. Rowley, 2 Woodside Crescent, Clayton, Newcastle-under-Lyme, Staffs ST5 4BW, Great Britain. Just as the Belgian club, this one seems to be heavily mathematically inclined. Their math programs are excellent and better screened than the ones from PPX. Their October 79 newsletter is not of the same quality as the Belgian one, but that might have changed since. I am waiting for more news from them. The editor wrote Richard Vanderburgh about an idea they had: what if we all get together and have Lubbock make us a special utility module. The editor is also trying to get Heinrich Schnepf warm to the idea. I'll keep you informed on the negotiations.

---

MOONLANDING PROGRAMS.- Last year in all the many German newsletters there was an outbreak of a rather serious "epidemic": moonlanding programs. You couldn't read one newsletter without encountering at least two of those programs in it, some of them very clever programming, but after a while boooooooring! This was also the sentiment of one of the editors, so he put his foot down and declared those things "programa non grata" starting with the October 1979 issue. Which made one of his members quip by means of a cartoon: The caption reads: "You children can come out now. Since October there are no more moonlandings."

---

BASS BOOSTER- Elsewhere in this issue you will find Terry Mickelson's program by this name. Because I run out of space, I could not type in the formulas Terry used in his program. So, here they are:

\[ R_1 = \frac{a}{(2 \pi f C)} \]
with \[ a = \frac{1}{2 \log_{10} \text{dB/20}} \]

\[ R_2 = \frac{1}{a} \]

\[ \text{with } \quad \text{dB} = \text{boost}, \quad C \text{ in Farads and } f \text{ in Hz.} \]

\[ \text{dB} = 20 \log \left( \frac{\sqrt{R_2/R_1}}{2} \right) \]

\[ f_0 = \sqrt{\left( \frac{\sqrt{R_2/R_1}}{2} \right)^2 / 2 \pi C R_1} \quad \left( \frac{2}{\sqrt{R_2/R_1}} \right) / 2 \pi C R_2 \]
TI PPC NOTES

BASS BOOSTER—Terry Nickelson is the author of this extremely practical program.

That is, if you are an EE you can make good use of it. Luckily, a full 70% of our members are EEs, so that I might be justified to bring you an EE program once in a while. Terry lives in Duncan, B.C. Canada.

The BASS BOOSTER is several things at the same time, depending on your specialty: an electronics designer would call it a second order Chebyshev high pass filter; sometimes he might refer to it as a Sallen and Key circuit; the audio designer calls it Thiele's auxiliary filter used to boost bass in loudspeaker boxes (Chebyshev alignment) that are purposely made too small, then corrected by this circuit; (and other means) to the audiophile, this is a very effective bass booster that not only sounds good (it is flat beyond 5 Hz!!!) but limits rumble and tone arm resonance because it is a high pass filter.

The max attainable boost is 16 dB. After this it begins to sharply peak the response. A very hot op-amp coupled with a buzzing sound in the speaker means oscillation. If you use the 741 op-amp, you will lose some treble, but otherwise it will work OK. The 318 has a tendency to oscillate. The LF356 is ideal.

About the program itself: PROGRAM WORKS WITH PRINTER ONLY.

Enter boost required in dB and press A.

Enter frequency of max boost in Hz and press B.

Enter value of C1 in microfarads and press C.

To obtain approximate and 1% values of R1 and R2, press D.

Now, if you enter 5 or 10% values for R1 and R2 respectively through A' and B', (and maybe also a new value for C1 in uF through C') pressing D' will give you the DB boost and the frequency.

If you enter an odd resistance value between 1 Ohms and 10 Megohms and press E', you will obtain the nearest 1% value. This is just a convenience routine not directly related to this filter program.

If you want to recall any of the values residing in A, B, C, A', B' or C', press E followed by one of the aforementioned keys. Thus E, A, E', C', E, B', etc.
ALPHABETICAL SORT. - Fortunately the regular PC-100 print code for letters of the alphabet increase in value in alphabetical order. A Shell sorting routine can then be modified to put short words or print code into alphabetical order.

Initialization The decimal point trick is used in LBL A so that a zero is normally stored in register 00. If a bad entry is made, enter the number of good entries and press A again. Register 00 will be incremented to display the next register number that your alpha print code will be stored. The contents of register 00 are also stored in HIR 08 for the Shell sort routine.

Print Code Modification The Shell sorting routine merely sorts numbers. The more digits the number has, the larger the number. An alphabet sort must disregard the number of characters in the sorting routine. One method is by dividing the print code down to a value less than ten but not less than one. The algorithm: "div LOG INT INV LOG =" can accomplish this but INV LOG needs to be rounded, thus EE is used at step 019. OP 03 at step 015 performs two functions. First it provides the integer function needed in the above algorithm. It then stores the mantissa of the logarithm as a single digit into HIR 07 in the form of \( N \times 10^{-12} \). This value is added to the converted print code and makes up a pseudo scientific notation to keep track where the decimal point goes in the original number.

The converted code is then stored indirectly. Register 00 is incremented to the next register and the CLR at step 007 takes the display out of EE mode. The process is repeated until the user enters all of the data to be alphabetized.

Shell Sort Routine LBL C is an optimized version of the Shell sort routine from the Math / Utilities library. If you have a Math / Utilities module, you can substitute "OP 30 PCM 06 B 1 HIR 38" in place of steps 031 to 111.

HIR 08 contains 1 more than the number of registers stored with print code. Half this value will be the first offset value or the difference in register numbers to be compared. HIR 05 contains the last low register number to be compared before picking a new offset value. Pending arithmetic is used to increment register 00 at steps 043 and 082. The contents of the lower register are entered into the t register at step 049. The offset value from HIR 07 is added to the pointer, register 00. The contents of the upper register is compared to that of the lower register. If the value is greater, then the lower register is incremented using pending arithmetic. \((1 + \text{STO 00})\) The next two registers are compared. If the contents of the upper register is less than that of the lower register, then the contents of the registers are swapped. Note how the contents are temporarily stored in the t register. \((\text{steps 059 to 068})\) The offset is again subtracted from register 00. \((\text{HIR 17} +/- \text{from the t register})\) Additional comparisons allow the smaller values to sink to the smaller register numbers.

When the lower register number exceeds the number of registers less the offset value \((\text{contents of HIR 05})\) then the pending arithmetic is cleared at step 088 which resets the lower register counter. A new offset value is chosen and a new lower register number limit is computed and entered into HIR 05.

When the sorting is finished, the contents of all the registers are in ascending order.

Reconstructing the Print Code The next step is to return the print code to its original form so that it may be used in a program or as the user sees fit. The display is rounded off with EE at step 121. This separates the print code which is stored back into the same register from the mantissa of the logarithm by subtraction. The mantissa in the form of \( N \times 10^{-12} \) is easily converted with EE 00. Steps 128 to 132 moves the decimal point back where it belongs. The print code is thus returned to its original value. Step 133 clears the EE mode and the DSZ loop steps down to the next register.
Printing the Results If all 99 registers need to be used, then a short printing routine can be entered from step 138 to 159. I seldom use this much data so decided to get a little fancy. With a little figuring, two columns can be printed out, thus saving some paper.

The contents of HIR 08 are divided by 2 and becomes a new offset value. Starting at step 152, the same penning arithmetic increment trick is used to enter the lower register number into register 00 and the t register. The contents of the lower register is loaded into OP 01. If the value in HIR 08 is greater than the lower register number, then the offset value is added to register 00 in step 144. The contents of the upper register are loaded into OP 03 and both are printed. The cycle is repeated until the lower register number is equal or greater than the offset value in HIR 08.

Remember, the original value stored in HIR 08 was 1 greater than the number of registers used to store print code data. Half of the HIR 08 value will leave an integer if an odd number of registers are used. A mixed number will result if an even number of registers are used. If HIR 18 is equal to the lower register number, (an integer) then an odd number of registers are used and a zero is entered into OP 03 before printing the last line. This prints only the low register contents in OP 01. Once the lower register number exceeds the value in HIR 08, the printing is stopped. The RST initializes the routine for a new list of print code to be sorted if desired.

Pre- Stored Data Occasionally I need to sort print code from pre-recorded registers. The print code in these registers is not in the correct format for the Shell sorter to alphabetize correctly. Label B was added to modify the print code in those registers. Just enter the number of registers to be alphabetically sorted beginning with register 01 and press B. Your number is incremented and entered into the t register. The contents of the registers are indirectly recalled and sent to a subroutine at step 013 to modify the print code. Register 00 is recalled at step 008 just before the RTN instruction. When this value is greater or equal to the value entered into the t register, the sorting will automatically start.

Program and description by Richard G Snow

Notes from the editor:
I am slowly learning to read other programmer's code and recognizing at the same time the person who wrote it. Everybody develops a peculiar style that stands out in the crowd. The Snow brothers pose a special problem: so long as their effort is individual (they live about 400 miles from each other) things are easy to me; I am able to distinguish Robert from Richard. But a few days after a holiday I usually get a super-program, proof that they got together again and produced something you need two heads for. The individual styles get mixed and the nice, recognizable, individual "handwriting" disappears. Is it surprising then that I sometimes have trouble crediting the legitimate author? This happened with the above program. The more, they sometimes simply sign "R C Snow" and that could be any of them. If the effort is pooled, credit is easily established, because the signature then is $R^2 \cdot C^2 \cdot \text{Snow}^2$. (Yes, you "purists" I know, it is mathematically incorrect, but it is cute.)

All the above, including Richard Snow's program description, as an addendum to ALPHABETICAL SORT in v5n2p6.

ASTRONOMY.- In v5n3p6 I told you about a STELLAR TRANSFORMATIONS program written by John Garza III. With John's permission I have now included that program with the Astronomy package consisting of articles on that subject I translated from Display. If those members who ordered the package would also like to have John's program, just send me a SASE. In future orders the program will be automatically included.
COMPILERS, INTERPRETERS, EDITORS, SIMULATORS AND SUCH.- In Display v4n5/6s67/73

Peter Klinghardt publishes a program called HP-59. I said "program" but should have said "programming system", for the whole thing consists of 9 card sides. The object of it is to enter and execute authentic RPN programs in means of a TI-59. To use it you first read in 3 card sides with an EDITOR program, which allows you to enter and edit your RPN program. Next you read in 3 different card sides with a TEST INTERPRETER program, which allows you to trace-execute your RPN program. And finally you read in a RUN INTERPRETER program on another 3 sides, which permits you to run the RPN program. It works alright, although it is SLOOOOOOW. The programming system supports an RPN program of up to 114 steps of merged code, DSZ on reg 0, 10 data registers, RPN-stack with 4 registers, 8 logic comparisons, 6 subroutine levels, 10 user-defined keys, 10 callable ordinary subroutine labels and indirect addressing of registers and labels through register 0, plus full printing. The system consists (in English translation) of 14 pages, too large for the newsletter. $4.00 US copying and mailing.

Edward G. Nilges, of Evanston IL, sent me another large system: MOUSE INTERPRETER. It runs on the 59 to same way, although slower, as a Basic program on a microprocessor described by Peter Grogon in Byte, v2n7p198-220. The programming system consists of a 640-step RUN-program and a 345-step LOAD-program. The whole system fits on three mag cards. Interested members may obtain a copy of this 11-page article-program for $3.00 US copying and mailing costs.

And finally, Robert J.K. Jacob brought to my attention the existence of a BASIC TO TI-58/59 COMPILER. Yes, this cross compiler written in Basic will translate, on a computer of course, other Basic programs into keystrokes for the TI-58/59. All you need for your computer to have at least 16 K RAM, numeric and string arrays and string functions such as MID$, CHR$ and ASC. The same company who sells this is also working on a cross compiler in Fortran. If my information is correct the cost of the software is $65.00 US. In any case, you might write SINGULAR SYSTEMS, 810 Stratford, Sidney, OH, 45365, USA.

PRINTING FOUR PRINT CODE REGISTERS ON ONE LINE.- In the discussion on the M/U module I offered the sequence

N STO 00 PGM 03 SBR 179 as a means to do the above.

J. Huntington Lewis says he has a simpler way that doesn't require the M/U module and is relocatable. He also claims that it needs less steps:

LBL PRT STO AA X:T STO BB OP 00 LBL IFF RCL IND AA OP IND BB OP 3AA DSZ BB IFF OP 05 RTN

Then enter into the print routine as follows:

....PN X:T RN SBR PRT ....

In which PN is the number of print registers to be used, such as 4, 3, 2 or 1. RN is the register location of the highest print code. AA is any register smaller than 10. BB is any register. OP 3AA is the OP 30 series code to decrement register AA.

HARDWARE.- In v3n7p6 (of 52-Notes) David Swindell (877) told about a hardware jump he had performed between two specific points on the printed circuit board of the SR-56. This way he was able to create pseudos. Several members have inquired about this trick. I don't know how to do it. I tried to get hold of David Swindell, but have not received any answer from him. Has Dave moved? Does anybody know his present address? Has anybody received info from Dave how to do it? Does it really work? Attila Voros is the latest member to inquire about it. Please write me if you know about this hardware trick. Thank you.
SR-52 LISTING ON A TI-59.

For those who still have a lot of 52 programs and would like to list them in a more readable form, this program by William Beeby comes in handy. PPX program 908068C seems to be an enhancement of an earlier program developed by TI. I have seen only the latter. It permits entry of maximum five steps. This program permits entry of 45 steps and is considerably faster. It also lists some of the pseudos, those mostly used, such as 83 (pseudo INT) and 63.

User Instructions: (TI-59/PC100A)

1. Read in program in 6 OP 16 (turn-on) and initialize by pressing E'.
2. Enter the SR-52 key codes in groups of five = 10 digits and press C.
   The display, which showed a 9, indicating 9 groups of five codes could be entered, now shows an 8. Same reasoning. Enter next group and press R/S.
   Repeat until all 9 groups have been entered. Processing starts automatically.
3. If you want to do single group processing, use key A.
4. If you want to list only a certain section, enter the first step in that section and press C'. This can be used as an error recovery.

I used this program to list Dean Athans PROPERTIES OF A POLYGON AREA program in this same issue.

SR-52 LISTING ON A TI-59.
PARTITION TO 10 OP 17 LOAD ALL ALPHA REGISTERS AND KEY IN PROGRAM STEPS.
PARTITION TO 6 OP 17 AND RECORD 2 ME cases, ALL 4 BANKS.

**TI-59 KEY CODES:** Jared Weinberger, in Bologna, Italy, contributes this routine which permits the generation of all the key codes on the TI-59. The program takes advantage of dynamic code modification to accomplish this feat. Press A to start.

000: LBL A 7 OP 17 100 STD 00 9.200760869 STD 60 LBL A' 6 OP 17
028: SBR 475 7 OP 17 .001 SUM 60 DSZ 0 A' R/S

All key codes are listed at step 478. Needless to say, you need the PC100A.
HIR Operations (G. Vogel)

Here is another HIR ops program for those who have not seen enough of them yet. But maybe this one is different: (1) It is very easy to operate, and (2) it works with any numbers. Normally, when a number smaller (in abs. value) than 1 is entered via any SUM or PRD (HIR) functions or their inverses, it is automatically changed to another number first: e.g. .002 is treated as 2E-03, changed to 2E+03, and so in fact 2000 is what operates, which is not wanted. But this problem is avoided if the pertinent operation is carried out in the scientific mode, and this trick is used in the program below, which is also rather easy to use:

E' clears all HIR registers.
To STO a number in HIR n, key it in, and press SBR and the n-th key from the top in the STO column (if it's a white one, prefix 2nd; if it's in the top row, SBR is not necessary).
To RCL the contents of HIR n, press SBR and the n-th key in the RCL column.
To SUM a number into HIR n, key it in, and press SBR and the n-th key in the SUM column (to subtract, follow number with +/-).
To PRD (multiply) a number into HIR n, key it in, and press SBR and the n-th key in the last column (to divide, follow number with 1/x).
To print all eight HIR registers, press A.

```
000 22 INV
001 21 S
002 35 92 HIP
003 76 LEL
004 74 07
005 70 16 S
006 82 HIP
007 01
008 39 97 HIP
009 76 LEL
010 75 LEL
011 74 06
012 75 LEL
013 22 INV
014 43 75 LEL
015 75 LEL
016 43 16
017 98 02 HIP
018 76 LEL
019 75 LEL
020 75 LEL
021 75 LEL
022 75 LEL
023 75 LEL
024 42 EE
025 82 HIP
026 05 05
027 82 HIP
028 76 LEL
029 76 13
030 76 LEL
031 76 06
032 82 HIP
```

Notes from the editor: What George Vogel is saying here is that you should use extreme caution when doing SUM, INV SUM, PRD and INV PRD functions on the HIRs with values comprised between 1 and -1. These values are changed by the HIRs: for example .002 is changed to 2000!!! The only way to do those functions is by entering those small values in EE or ENG format. Then, the HIRs do not change them. So, George put an EE command in those routines. But, again caution, this kind of lulls you to sleep when experimenting with HIRs. You might believe after a while that all is peaches and cream, until you start using the HIRs in a real program and discover that you didn't take into account the possibility of numbers between 1 and -1. So, to experiment more realistically I would remove all those EE commands. Then, where necessary you can enter your values in EE format.
SNOOPY.— About a year ago I got from one of the Snow brothers a program. In the same envelope I found a PC100-drawing of Snoopy. No explanation, just the drawing. I had to get even with them, so I returned a mag card with a short and (admittedly) rather clumsy program to draw Snoopy by just pressing A. I didn’t have to wait very long to receive an enhancement of my program. This time Snoopy could even speak, any four-letter word or his traditional CURSES. And you can draw the little critter by means of any of the alpha characters. Only his eye will always be a Q.

USER INSTRUCTIONS:
If you just want to draw Snoopy, enter the two-digit code of the character you want used for drawing and press B.
If you want Snoopy to say something, enter up to five characters, 10 digits, and press A. After a zero appears in the display, enter the two-digit drawing code and press B.
If you want Snoopy to say CURSES just press D.
And if you want Snoopy’s name printed on the bottom of the drawing, press C when the drawing stops.
To repeat a drawing, just press R/S.

RECORDING INSTRUCTIONS:
In 8 OP 17, load registers 61 through 79 with print code.
Key in the 318 program steps.
In 6 OP 17 record both sides 1 and 2 of a mag card.

LIQUID AND GAS FLOW THROUGH AN ORIFICE.— Mark A. Pelletier, 1213 E. Miller Street, Griffith, Indiana, 46319, has written some specialized programs to compute liquid and gas flow through an orifice, based on Spink IX. (flange taps) Mark will send copies of them to any interested member. As a general courtesy when requesting copies, I would suggest to always send a SASE, even when the author doesn’t specifically requires it. It saves him time and effort. The TI PPC CLUB thanks Mark for his generous offer.
The M/U Module - One of the better modules to appear is the Math/Utilities module. It is well programmed and has lots of practical routines. Especially PGM 05, the super-plotter, is tops. Here is short demonstration routine for that program. Although it doesn't utilize to the fullest all its capabilities, it produces a rather impressive curve representing the classic equation \( f(x) = \sin \frac{x}{x} \). Write in user memory the following short program:

```
LBL A' 40 ( X:T RAD STO 26 SIN DIV RCL 26 ) RTN
```

Then initialize by accessing PGM 05
Enter initial value of \( x \) as 4 +/- \( x \), \( \pi = \) and press A.
Enter \( x \)-increment as \( \pi \div 4 = \) and press B.
Enter \( y \)-min as 0.25 +/- and press C.
Enter \( y \)-max as 1 and press D.
Enter number of desired tapes as 1 and press E.
Enter number of functions as 1 and press D'.
Enter number of points to be plotted and press E' . (I suggest to start with 30)
Now sit back and watch the plotting.
Note that in the user program, the 40 is the code for the decimal point. May be replaced by any other code, of course.

Some other, non-scheduled tricks with the M/U module:

```
PGM 03 SBR 363 will print the contents of the display in OP 01 | Warning: these routines alter the content of reg 00.
PGM 03 SBR 369 same OP 02.
PGM 03 SBR 375 same OP 03.
PGM 03 SBR 381 same OP 04.
PGM 02 SBR 315 will print the word LOW.
PGM 02 SBR 313 will print the word SLOW.
PGM 05 SBR 262 is handy to end a program execution. It will print a line of decimal points, followed by 4 ADVs. Be careful, though, with Reg 24. It is being D SZed each time this routine is used.
Most people object to the prompting used in the M/U module as ENTER CARD, when it should have said ENTER BANK. If you want to change that, you can do your own prompting by using in your program part of the module:
PGM 02 SBR SBR 14 13 29 26 OP 02 PGM 02 SBR 033
```

One extremely practical routine of hidden gold buried in the module is the following:

```
N STO 00 PGM 03 SBR 179 which will print on one line four consecutive print code registers, \( N \) being the highest of the four. For example: 24 STO 00 PGM 03 SBR 179 will print the print code contents of registers 21, 22, 23 and 24 from left to right on the tape. It constitutes a handy, up to four, column printer.

These are only a few of the special tricks possible with the M/U module. I hope the members will examine more closely this handy module and send me their findings.
```

M U-03, ALPHA MESSAGES, is very handy if you want to store, for example, addresses and/or telephone numbers. It is, however, sometimes a chore to calculate what message goes into which register, especially if you have already information on a mag card and want to add some more. In that case, your fear is that you might destroy some of the previous data. So, I computed a list, reproduced on the next page, that will give you line numbers, registers involved, partition required and, for the sake of completeness, what to press. Robert and Richard Snow conveniently converted the list to a short program:

```
LBL A HIR 08 X 4.04 + .03 = RTN LBL B . INV EQ HIR HIR 18 LBL HIR X .4 + 1.3 )
OP 17 RTN LBL C OP 16 INV INT X 25 - .75 ) RTN
```

Instructions:
1. a. Which registers store a given line? Enter line # and press A. Display will show xx.yy in which xx is the lowest and yy the highest of four registers.
b. After this, what partition do I need? Press B. Display will show the correct partition and the calculator is partitioned automatically.

c. What is now the maximum number of lines that can be accommodated in this partition? Press C for the answer.

2. a. To accommodate a given number of lines, what partition do I need?
Enter number of lines and press B.
Again the partition is shown and the calculator partitioned automatically.
b. What is the maximum number of lines in this partition? Press C.

3. In the current partition, what is the maximum number of lines that can be printed and stored? Press C for the answer.

You have a choice of doing either 1, 2 or 3 in sequence.

<table>
<thead>
<tr>
<th>LINE #</th>
<th>REGISTERS</th>
<th>PARTITION</th>
<th>PRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 5 6 7</td>
<td>879.09</td>
<td>1 OP 17</td>
</tr>
<tr>
<td>2</td>
<td>8 9 10 11</td>
<td>799.19</td>
<td>2 OP 17</td>
</tr>
<tr>
<td>3</td>
<td>12 13 14 15</td>
<td>799.19</td>
<td>2 OP 17</td>
</tr>
<tr>
<td>4</td>
<td>16 17 18 19</td>
<td>799.19</td>
<td>2 OP 17</td>
</tr>
<tr>
<td>5</td>
<td>20 21 22 23</td>
<td>719.29</td>
<td>3 OP 17</td>
</tr>
<tr>
<td>6</td>
<td>24 25 26 27</td>
<td>719.29</td>
<td>3 OP 17</td>
</tr>
<tr>
<td>7</td>
<td>28 29 30 31</td>
<td>639.39</td>
<td>4 OP 17</td>
</tr>
<tr>
<td>8</td>
<td>32 33 34 35</td>
<td>639.39</td>
<td>4 OP 17</td>
</tr>
<tr>
<td>9</td>
<td>36 37 38 39</td>
<td>639.39</td>
<td>4 OP 17</td>
</tr>
<tr>
<td>10</td>
<td>40 41 42 43</td>
<td>559.49</td>
<td>5 OP 17</td>
</tr>
<tr>
<td>11</td>
<td>44 45 46 47</td>
<td>559.49</td>
<td>5 OP 17</td>
</tr>
<tr>
<td>12</td>
<td>48 49 50 51</td>
<td>479.59</td>
<td>6 OP 17</td>
</tr>
<tr>
<td>13</td>
<td>52 53 54 55</td>
<td>479.59</td>
<td>6 OP 17</td>
</tr>
<tr>
<td>14</td>
<td>56 57 58 59</td>
<td>479.59</td>
<td>6 OP 17</td>
</tr>
<tr>
<td>15</td>
<td>60 61 62 63</td>
<td>399.69</td>
<td>7 OP 17</td>
</tr>
<tr>
<td>16</td>
<td>64 65 66 67</td>
<td>399.69</td>
<td>7 OP 17</td>
</tr>
<tr>
<td>17</td>
<td>68 69 70 71</td>
<td>319.79</td>
<td>8 OP 17</td>
</tr>
<tr>
<td>18</td>
<td>72 73 74 75</td>
<td>319.79</td>
<td>8 OP 18</td>
</tr>
<tr>
<td>19</td>
<td>76 77 78 79</td>
<td>319.79</td>
<td>8 OP 17</td>
</tr>
<tr>
<td>20</td>
<td>80 81 82 83</td>
<td>239.89</td>
<td>9 OP 17</td>
</tr>
<tr>
<td>21</td>
<td>84 85 86 87</td>
<td>239.89</td>
<td>9 OP 17</td>
</tr>
<tr>
<td>22</td>
<td>88 89 90 91</td>
<td>159.99</td>
<td>10 OP 17</td>
</tr>
<tr>
<td>23</td>
<td>92 93 94 95</td>
<td>159.99</td>
<td>10 OP 17</td>
</tr>
<tr>
<td>24</td>
<td>96 97 98 99</td>
<td>159.99</td>
<td>10 OP 17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>MAX NUMBER OF LINES</th>
<th>BANK #</th>
<th>REGISTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 OP 17</td>
<td>1 1/2</td>
<td>1</td>
<td>90 TO 99 + 160 PGM STEPS</td>
</tr>
<tr>
<td>2 OP 17</td>
<td>4</td>
<td>2</td>
<td>60 TO 89</td>
</tr>
<tr>
<td>3 OP 17</td>
<td>6 1/2</td>
<td>3</td>
<td>30 TO 59</td>
</tr>
<tr>
<td>4 OP 17</td>
<td>9</td>
<td>4</td>
<td>00 TO 29</td>
</tr>
</tbody>
</table>

I made a photocopy of the above list and pasted it as an added page facing page 12 of the MU manual. It complements the instructions on page 12. (of the manual)

Additionally, the short utility program is recorded on a mag card and kept in a slot of the MU card and module carrying case.
DETERMINING A REMAINDER. - In v5n2pl2, in Nichomachus’s puzzle, Richard Snow offered the method - ( CE DIV 105 ) INT X 105 = . But if you look on the same page, the Fibonacci number routine by George Vogel, steps 013 through 024, don’t they present a remarkable resemblance to the algorithm above? Richard draw my attention to it.

QUADRATIC SOLUTION. Stuart Cox offers this “quick and dirty” solution to equations of the form Ax^2 + Bx + C, which he translated from an RPN program in the HP PPC JOURNAL, but doesn’t remember the particular issue. Sorry, Richard Nelson, we did our best. (For our newcomers: Richard Nelson is the editor of that journal)

This short program - only 45 steps - uses the P/R conversion for an imaginary root, uses no tests, and uses only two data registers. The program can be used on the TI-57, the 58 and the 59. On the 57, ignore the first two steps: LBL A.

Instructions:
Enter A, press A (58/59) or RST R/S (57). Enter B, press R/S. Enter C, press R/S. See imaginary portion of root. If zero is displayed, then there are two real roots. Else, if no zero displayed, there are two real roots and they are identical. Press R/S to see the first real root. Press R/S again to see the second real root.

000: LBL A STO 00 LT X R/S DIV 2 = STO 01 X^2 - R/S DIV RCL
017: 00 = X:T 0 INV P/R DIV 2 = X:T VX X:T P/R R/S X:T - X:T
034: RCL 01 = R/S RCL 01 +/- - X:T = R/S (last step : 044)

COSINE LAW. - Stuart Cox also offers the following short routine. Only 15 steps on the 58/59 and three less on the 57. The routine computes the cosine law, defined as c^2 = a^2 + b^2 - 2abCosC. (Source: PPC JOURNAL)

The program uses the P/R conversion and the t-register.


ENHANCED OP 07 - OP 07 normally prints only with an asterisk. If you want to print with any character, use this clever routine by Bill Beebe. There are no safeguards to prevent out of bounds entry, so make sure you enter a number between 0 and 19 only. Keep the alpha code for the character you want to print in register 01. Enter 0 to 19 and press or call A.

000: LBL A OP 00 DIV X:T 5 - INT STO 02 OP 22 = X 10 +/- + 8 =
021: INV LOG EE INV EE X RCL 01 = OP IND 02 OP 05 X:T R/S (last step : 035)

BRAIN TEASERS. Object of these “time wasters” as Stuart Cox calls them, is to try to devise a key sequence that will turn the display either into the number 197 or into 0.1415926536, each routine to have 13 steps, or if you are real clever, less than 13 steps. Stuart offers two routines to accomplish this and says that INV does not count as a step. This probably to be able to compare the number of steps to those on an RPN calculator and in order not to give them unfair advantage. These puzzles seem to be similar to the one we once had in 52-Notes and which required you to produce a 3 in the display. Needless to say, you cannot use any numerical key. And yes, all the other keys are fair game.

HIR REGISTERS CLEARING. Palmer O. Hansen reminds me of a statement in v5n1p2 that said something to the effect that no single key will clear all the HIR registers. So, he asks me if we have an efficient routine to do so. The best he achieved to date was a routine of 15 steps long. I thought it could be a little shorter. So, what about this one:
CLR RMS HIR 03 HIR 04 OP 00 in which CLR RMS cleans the first two HIRs, CLR HIR 03 HIR 04 does it to the next two and OP 00 wipes HIR 5 through 8.
Relative Primality of Up To 56 Integers (W. J. Widmer)

This program calculates the GCD of up to 56 integers; if GCD > 1, the relatively prime cognate set is output on pause (if printout is desired, be sure to correct direct address steps). With partitioning, up to 96 numbers may be tested (or up to 104 if the HIR registers are used per TI-PPC Notes V5N1Pl2).

<table>
<thead>
<tr>
<th>STP Code</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>76</td>
</tr>
<tr>
<td>1 11</td>
<td>A</td>
</tr>
<tr>
<td>2 58</td>
<td>FIX</td>
</tr>
<tr>
<td>3 00</td>
<td>0</td>
</tr>
<tr>
<td>4 42</td>
<td>STO</td>
</tr>
<tr>
<td>5 00</td>
<td>0</td>
</tr>
<tr>
<td>6 42</td>
<td>STO</td>
</tr>
<tr>
<td>7 57</td>
<td>57</td>
</tr>
<tr>
<td>8 91</td>
<td>R/S</td>
</tr>
<tr>
<td>9 72</td>
<td>STO</td>
</tr>
<tr>
<td>0 10</td>
<td>57</td>
</tr>
<tr>
<td>1 97</td>
<td>DSZ</td>
</tr>
<tr>
<td>2 57</td>
<td>57</td>
</tr>
<tr>
<td>3 00</td>
<td>0</td>
</tr>
<tr>
<td>4 08</td>
<td>0</td>
</tr>
<tr>
<td>5 00</td>
<td>0</td>
</tr>
<tr>
<td>6 32</td>
<td>XST</td>
</tr>
<tr>
<td>7 43</td>
<td>RCL</td>
</tr>
<tr>
<td>8 00</td>
<td>0</td>
</tr>
<tr>
<td>9 42</td>
<td>STO</td>
</tr>
<tr>
<td>0 20</td>
<td>57</td>
</tr>
<tr>
<td>1 73</td>
<td>RCL</td>
</tr>
<tr>
<td>2 57</td>
<td>57</td>
</tr>
<tr>
<td>3 42</td>
<td>STO</td>
</tr>
</tbody>
</table>

To get DSZ NN nnn (steps 11, 25, 45): STO NN BST BST DSZ SST (gives DSZ NN); then STO nn BST BST n SST SST SST (gives direct address follow-up)—see TI-PPC NOTES V5N1Pl2. Briefly: step 16 sets for test at 091; steps 4-7, 17-20, 36-39, 49-52 set counters for decrement cycles; steps 21-28 call GCD calculation; steps 69-93 are the GCD algorithm (compare steps 13-40 in TI-PPC NOTES, V5N2Pl2); steps 40-48 reduce N's to relatively prime cognates; steps 53-68 cycle the outputs and end the program.

Instructions
1. For m integers (m ≤ 56) input m and key [A]  
2. Input 1st N and key [R/S]  
3. Repeat step 2 for all N's  
4. Last [R/S] outputs GCD FIX8  
5. If GCD = 1, end of program  
6. If GCD > 1, [R/S] once to calc & output all n = N/GCD

Examples


See flow diagram on next page.
BIOMEDICAL USE OF A CALCULATOR. - A German firm, with a grant from the Deutches Blindenhilfswerk e.V. Duisberg (the German equivalent of the Lighthouse for the Blind) has developed the Braillotron. It is a cradle-like device in which fits one of the TI calculators: TI-2550 II, TI-30, SR-51, SR-52. The top part of the cradle contains a Braille "display" consisting of either 9 or 14 Braille characters, depending on the calculator used. The calculator itself is modified with a plug in the back, so that its display drives the Braille modules, arranged in the same configuration as the digits in the display of the calculator. Display time is .56 sec for 14 characters. Dimensions of the cradle are 8.3 in (210 mm) by 4.7 in (120 mm) by 2.8 in (70 mm). Weight is about 700 g or 1 1/2 lbs. No price given. Address: Dipl. Ing. K.P. Schönherr, Schloß Solitude 3, Technologieforschung in Medizin- und Rehabilitationstechnik, D-7000 Stuttgart-1, West Germany. Tel. 0711/694327.

TRAFFIC CONTROL: - In the journal of the Institute of Transportation Engineers, the ITE Journal, James P. Ruddon, P.E., published in the April 1980 issue a program called THE UN-TRAFFIC CONTROLLER. It is a TI-58/59 only program that simulates a traffic controller at two roads crossing each other. There are, of course, many microprocessor-controlled devices on the market that can do it much better, just because they have Input/Output ports, which the TI-58/59 lacks. But this is the first time I have encountered such an attempt. The program is well-documented with flowcharts.

HANDBOOK OF ELECTRONIC DESIGN AND ANALYSIS PROCEDURES USING PROGRAMMABLES CALCULATORS. Bruce K. Murdock. Van Nostrand Reinhold Electrical/Computer Science and Engineering series. 1979. § 26.59 US. The book is, as the title implies, of great value to electrical and electronic engineers. The main topics are: Network Analysis, Filter Design, Electromagnetic Component Design, High Frequency Circuit Design and Engineering Mathematics. This is by far the best book I have seen to enable the EE to write good, useful programs. Although most programs in the book are for the HP-67/97, and only a few of them, in the Network Analysis section, have been translated for the TI-59, the algorithms, formulas and flowcharts supplied for each program are so easy to follow, that it would be a cinch to write TI-59 programs from them. Furthermore, each program has a worked-out example, so that programs you might write can be checked out easily. This 525-page book is well worth its price.
MORTGAGE SCHEDULE ON THE SR-56.-- George Vogel.

The following program for the SR-56 (or TI-58/59) will print the complete schedule for a "direct reduction" (mortgage) loan. Press RST to initialize, and enter amount of loan, % annual interest, and number of monthly payments with R/S. The printout gives the amount of the (constant) monthly payment, then for each payment its number, interest part, principal part, and balance remaining. After the last payment, the program prints the total interest paid and stops, leaving you to marvel at the cost of borrowing. -- The program illustrates (twice) the use of the handy property of EE of truncating (as opposed to merely rounding) the display. For example, the monthly payment is calculated to be, say, $123,4567890.123. Merely rounding it by Fix 2 would print it as the better-looking $123,460, but the full number would be carried through the rest of the program, cumulating an error which could easily add up to several dollars. Truncation with EE (followed by INV EE to return to standard display mode) will keep everything accurate to the last cent.

CMs fix 2 STO 1 prt R/S prt DIV 1200 = STO 2 R/S prt pap STO 0 x:T RCL 1 x RCL 2 DIV 
( 1 - ( 1 + RCL 2 ) y^x RCL 0 +/- = EE INV EE STO 3 prt pap 1 SUM 5 RCL 5 fix 0prt RCL 1 x RCL 2 = fix 2 EE INV EE prt SUM 4 +/- + RCL 3 = prt INV SUM 1 RCL 1 prt pap RCL 5 EQ 92 GTO 49 RCL 4 prt pap pap pap pap pap R/S (100 steps; pap = Adv)

(EQ means x=t)

OP 07 ENHANCED SOME MORE.-- Just to prove that he is able to write a more "civilized" version of his enhanced OP 07, Bill Beebe sent me this one:

000: LBL A OP 00 X:T 19 GE 013 OP 99 RTN ( ( ( X:T DIV 5 - INT STO
022: 02 OP 22 ) X 10 - 8 ) ABS INV LOG X RCL 01 ) EE INV EE
042: OP IND 02 OP 05 RTN (last step 046)

As you can see, the program checks for out-of-bounds entry and, as a true SBR, is written with only parenthesis. No = sign is used, so no danger of prematurely completing pending operations in the main program.

As in the simple OP 07 enhanced, a call to A will execute the SBR.

CUBIC EQUATION: Samuel G. Allen offers these two solutions to a cubic equation of the form x^3 + ax^2 + bx + c = 0. Sam made these routines for the SR-56, which he prefers above the larger models because of its speed. I suppose they will run as well on the TI-57 and the TI-58/59.

When a cubic equation has an isolated real root (its absolute value is considerably larger or smaller than that of the two other roots) that root may be found by one of the following routines:

000: X ( CE + RCL 2 ) + RCL 1 = 1/X X RCL 0 = +/- PAUSE RST
000: 1/X X ( CE X RCL 0 + RCL 1 ) + RCL 2 = +/- PAUSE RST

The first routine will converge to a root of a small absolute value and the second will converge to a root of a large absolute value. Put"a" into reg 2,"b" into reg 1 and "c" into reg 0. Start either routine with a non-zero number in the display. An approximate root is better, if known.

SHORTEST USEFUL ROUTINE-- Samuel G. Allen sends me the shortest do-something-useful routine I have seen so far: 000: X:T + PRT RST

With the t-reg clear and 1 in the display, press RST R/S and it will print a listing of the Fibonacci Numbers. Now, somebody is likely to dispute the usefulness of the Fibonacci numbers per se. Useful or not, some people seem to get a kick out of them, judging by the existence of an entire journal devoted to them: The Fibonacci Quarterly.

CLEARING HIRS.-- On v5n3p9 I stated that OP 00 will clear HIR 5 through HIR 8. That is true only when the printer is attached, say Palmer O. Hanson. Prove it is: v5nlp2, the printer sensing routine: 1 P/R OP 00 HIR 18.
ALARM CLOCK.— John Wortington and Emil Regelman, both of Bowie, MD, are the authors of this program. The clock displays hours, minutes and seconds in an HH.MM SS format, the SS indicated by the "power of ten" digits. The display is continuous, except for the last second of each minute, to allow the program to compare actual time with alarm time.

The alarm buzzer is the program card itself being pulled through the card reader! It is possible to move the buzzer by placing one or more NOP cards with other commands that take more time, such as I=-I. This can cause your clock to run fast. If your clock runs slow, speed it up by replacing one or more pauses with NOP cards. In any case, do not insert nor delete, as this program has direct addresses.

User Instructions:
1. Select either 12-hour or 24-hour format.
   For 12-Hr: enter clock starting time in HH.MM and press C. For PM enter -HH.MM.
   For 24-Hr: enter clock starting time in HH.MM and press C.
2. Enter alarm time in HH.MM format and press A.
3. At actual time = starting time, press E to run program.
   Clock will indicate in format HH.MM SS. Seconds will be updated every second.
4. To arm the alarm, slide the program card, side 1, into the card reader slot.
   When actual = alarm time, the card will be pulled through, sounding the alarm.
   After that, the clock will continue to indicate the correct time.

| '000 43 RCL | 060 66 PNU | 120 66 PNU | 180 66 PNU |
| '011 01 01 | 121 06 2 | 181 04 4 | 241 05 5 |
| '02 32 X7 | 122 08 3 | 182 02 2 | 242 08 9 |
| '03 45 X | 123 09 4 | 183 06 2 | 243 04 4 |
| '04 32 X | 124 09 4 | 184 04 2 | 244 06 2 |
| '05 38 44 | 125 02 2 | 185 04 4 | 245 05 5 |
| '06 26 44 | 126 09 3 | 186 04 4 | 246 04 9 |
| '07 30 3 | 127 09 4 | 187 06 4 | 247 06 5 |
| '08 30 3 | 128 09 3 | 188 06 5 | 248 05 5 |
| '09 38 3 | 129 09 4 | 189 05 3 | 249 05 3 |
| '10 30 3 | 130 09 4 | 190 05 3 | 250 05 3 |
| '11 30 3 | 131 09 4 | 191 05 3 | 251 05 3 |
| '12 30 3 | 132 09 4 | 192 05 3 | 252 05 3 |
| '13 30 3 | 133 09 4 | 193 05 3 | 253 05 3 |
| '14 30 3 | 134 09 4 | 194 05 3 | 254 05 3 |
| '15 30 3 | 135 09 4 | 195 05 3 | 255 05 3 |
| '16 30 3 | 136 09 4 | 196 05 3 | 256 05 3 |
| '17 30 3 | 137 09 4 | 197 05 3 | 257 05 3 |
| '18 30 3 | 138 09 4 | 198 05 3 | 258 05 3 |
| '19 30 3 | 139 09 4 | 199 05 3 | 259 05 3 |
| '20 30 3 | 140 09 4 | 200 05 3 | 260 05 3 |

USE OF THE CALCULATOR IN MANAGEMENT.— Dr. John M. Cozzolino is an Associate Professor and the Director of the Business Risk Education Center at The Wharton School of the University of Pennsylvania. Dr. Cozzolino teaches seminars on Scientific Methods for Risk Management Decisions. Each attendee receives a TI MBA calculator as part of the course. (and instruction how to use it) If you bring an SR-52 or a TI-59, special programs have been developed to be used on those calculators.
KEYCODE TRANSLATION- In v5nlp0 Don Laughery talked about his unique TRACE state. As a comment I added that, if you entered a program in memory, the calculator insisted on reducing the codes of the entered commands by 10. Now, John Mairs of Springfield VA, has done some nice sleuthing and come up with a new "transitional" state that exhibits some similar behavior with respect to altering key codes. John writes:

I have discovered an interesting calculator quirk on my TI-58 (it works on the 59 as well) which might have some useful application to program security. (Richard Snow and I doubt that it can be done. But, one never knows.) From Turn-on and with the ME-module in place, key in the following: LRN Y LRN RST 100000MN PGM 1 SST (1) in which Y is any key whose keycode has a digit > 4 in the ones place. For example CLR, SUM and PRT are OK, while GTO and STO are not. Now enter LRN mode and press any key, BST and see that the keycode is MN higher than normal. Richard Snow says that it would constitute a nice way of entering merged code, especially HIRs. Suppose you entered as MN the digits 50. By simply pressing X:T (32 code) you would get 50 + 32 = 82 = HIR !!! The machine will execute a program with this "elevated" key code. When not in LRN mode, pressing CP or RST will disengage this new transition state, BUT IT WILL NOT ALTER THE MODIFIED KEY CODE IN PROGRAM MEMORY.

If in sequence (1) above 100000MNAB is pressed, after LRN mode is entered, the machine will be on (approximately) step 80*AB + 1. For example, if 100000MN06 is used, the program step counter will be at 481. Although, since I have a TI-58, it is not really on that step. (On a TI-59 it is. On the TI-58 the failure to get to step 481 is due to the partition limit, of course.) If AB>03, pressing BST will take the machine out of LRN mode and will make it impossible to re-enter LRN mode.

If 100000MNAB+CD is used to generate 100000MNAB.CD (although the display shows only the ten integer digits) the machine will also come up on a step approximately equal to 80*AB.CD+1.

Now, in reference to program security, I would like to see if the modified code can be written onto a mag card. (Both Richard Snow and I confirm: It can be done.) And also if is possible for the mag card to re-read into the machine the modified code translated back into the normal code, so the machine can execute it normally. That it doesn't do, sorry to say. It reads the modified code and executes as modified code. I invite you and your editors to research the subject further.

Any guess as to why the initial key code in step 000 must not have 0 to 3 in the ones location? Why does the machine alter the program step if AB.CD is used? We don't know. Does anyone out there?

Some remarks from Richard and me: As John has pointed out, practically any address may be accessed. If a GTO is used in the calculate mode, then N will not be added to the translated key code. If an address is accessed beyond the present partition, then a maximum of seven steps of translated code may be entered into a single register before the machine automatically goes out of LRN mode. The contents of that register may then be recalled.

Program steps even beyond register memory can be accessed up to step 7999. Doesn't that remind you of the Firmware REVISED in v5n3p6? Useful storage at these steps seems doubtful, as user RAM doesn't exist beyond step 359. But what if somebody would solder another RAM chip on top of the existing one, as we did in the SR-52? Wouldn't that provide a means of accessing those extra steps?

One should experiment with variations of John's method as the usual fractured digits, program crashes and maybe even strange trace modes are encountered. The key code translation routine doesn't have to be accessed from the machine just turned on. The routine may also be initiated at any step containing a key code with a units digit of 4 or more. The digits used in the display are transferred to a buffer register and seen as you SST through the seven steps in LRN mode. The digits can also be found back in the t-register, but with the decimal point relocated. The display is put into a FIX 10 mode, except when displaying a zero, which appears to be in FIX 9. Flag 4 seems to be the only flag that becomes set while accessing this transitional mode. HIRs 1 through 8 contain 10^-99.

The translated code is the sum of MN and the key code. BUT NEVER EXCEEDS TWO DIGITS. If 56, for example, is used as MN and the + key is pressed, one would expect to obtain 56 + 85 = 141, but, because of the two-digit maximum we obtain 41. This would give an alternate method to enter the SST (+41) code for the diagnostic routine in v5n3p9.
13-DIGIT ALPHA REGISTER PRINT.- This program, by Richard C. Snow, of Vallejo, CA, will list the contents, the alpha and the register number of data registers loaded for OP or for HIR printing. The program will automatically select between the two methods by a process explained later in this article. The method works in 99% of the possible cases.

A minimum of 15 digits is required to print 13 digits, a decimal point and a possible minus sign. There is consequently not enough space left on one print line to also print the register number and the alphanumerics. Therefore, alpha is printed on the second line.

Few programs use all the registers for HIR print code. Some also use OP print code. The way this program selects between the two is by assuming that OP data print code consists of only integers and having no more than ten digits. This should work in the majority of cases. A simple algorithm to do this could be:

```
CP RCL IND 00 - OP 02 EE INV EE = X=T PRT RCL IND 00 HIR 06 LBL PRT OP 05
```

The OP 02 leaves only the integer value and EE rounds the number to the value in the display. This value is subtracted from the original number. If the difference is NOT zero, then the data either contains a fraction or it has more than ten digits shown in the display. The data is then re-entered into HIR 06 and printed as HIR code.

How about listing registers which are not print code? LBL B sets flag 1, which is later tested to by-pass the alpha print routine. LBL A clears flag 1 and allows the alpha to be printed. The user determines which registers are to be listed with alpha.

The first part of LBL A clears any pending operations, such that a listing may be stopped at any time & another listing started at any register, without fear of having the program print out a lot of left-over garbage. OP 19 and flag 7 are used to stop the program after listing the last register in the present partition or when an error condition occurs. When the program stops automatically, there are no pending operations, the print registers are cleared, the t-register is zeroed, all flags are reset and the error condition is cleared. In short, you are ready to enter a new program, without being haunted with operations left over from this one.

The heart of the 13-DIGIT ALPHA REGISTER LIST program is a modified version of the (Robert Snow) PRINT CODE CONVERTER routine. The longer version used here preserves the least-significant digits of fractions being converted to print code. When an integer is added to a fraction, digits beyond the twelfth digit are truncated.

Register containing a zero are skipped. If the number to be converted is negative, the print code for the minus sign is entered into HIR 08. The alpha code (2000) is already multiplied by 100 to make room for the first converted digit. Is the number a fraction, (absolute value is less than one) the number is sent directly to the print code converter routine. Larger numbers are divided down to \( N \times 10^0 \) before entering the print code converter. Fractions less than .01 are sent to another part of the print code converter to have their leading zero eliminated. This allows another significant figure to be printed.

Two DSZ loops are used in the print code converter routine. DSZ 01 limits the number of digits to be converted before leaving the loop to load the print registers. (five digits max) DSZ 00 determines when the print code for a decimal point will be added to the converted code in HIR 08. Flag 0 is set to indicate that the decimal point code has been entered. In the remaining loops, if flag 0 is set, the decimal point print code (40) will be by-passed. HIR 01, a pending arithmetic register, contains the rest of the number to be converted. If this value is zero after flag 0 is set, the program exits the converter routine. This to prevent trailing zeros in fractions.

LBL A and LBL B append .1 to the register number to prevent exiting the print code converter to soon, when printing register number ending in zero, such as 10, 20, 30, etc.

Since the converter routine uses registers 00 and 01, the register number to be printed is stored in HIR 04 and it is incremented near the end of the program at step 231. The data to be printed is temporarily stored in HIR 03, so that the alphanumerics can be printed for register 01 after its contents have been destroyed.

(over)
A maximum of sixteen digits of data can be printed, so that thirteen significant digits can be properly printed, for numbers between .001 and 10^16. Smaller fractions will be rounded off in steps 167 and 173. Larger numbers will only be missing trailing zeros and a decimal point.

Normally there is a blank space between the printed data and the register number. A final decimal point check was added at step 183 to provide a decimal point for values of N x 10^15. Flag 0 is set to prevent a decimal point being produced while the two-digit register number is being converted.

LBL A' enters 100 into HR 08. This "100" can be any three digits. This to prepare the converted data to be used as HIR print code. This keeps the converted code left-justified in order to print the number as a continuous string of digits.

USER INSTRUCTIONS:
A : 13-Digit register list with alphanumerics.
B : 13-Digit register list only. No alpha.

Enter the number of the first register to be listed and press either A or B. If no register is specified, the listing will begin with register 01.

Register 01 through 89 may be listed by this program. The contents of register 01 should be restored if a second listing which includes register 01 is needed.

<table>
<thead>
<tr>
<th>CODE</th>
<th>DECIMAL</th>
<th>DIGIT CONVERTER 9 8 7 6 5 4 3 2 1 0</th>
<th>FRACTION 0 1 2</th>
<th>ALPHABETICS</th>
<th>FIXED DECIMAL</th>
<th>DATA</th>
<th>MP</th>
<th>REV</th>
<th>REG</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td>000</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td>001</td>
</tr>
<tr>
<td>002</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td></td>
<td></td>
<td></td>
<td>002</td>
</tr>
</tbody>
</table>

AREA OF A REGULAR N-GON (POLYGON) - John D. Garza III, from Texas City TX, is the author of this short, but handy routine. Good things always seem to arrive in pairs: No sooner had I typed in the SR-52 program on PROPERTIES OF A POLYGON by Dean Athans, when this one by John Garza arrived. It was a cinch for the same two reviewers to do also this one in a hurry. They had experience by now.

USER INSTRUCTIONS:
The program requires the Master Library Module but runs without the printer. Needless to say it works on both TI-58 and TI-59.

Enter the number of sides of the polygon and press A.
Enter the length of one side and press A again. (John uses a two-register stack)
To obtain the area, press R/S.

000: LBL A  EXC 08  EXC 09  R/S  RCL 08  DIV 2 ) PGM 12 A 90 PGM 12 B  RCL 09
022: RCL 09  ) PGM 12 C  PGM 12 A'  PGM 12 D  PGM
040: DIV 2 = RTN

__________________________

033  23  CE ENG
031  31  LBL
030  11
029  09
028  08
027  07
026  06
025  05
024  04
023  03
022  02
021  01
020  00
019  09
018  08
017  07
016  06
015  05
014  04
013  03
012  02
011  01
010  00
009  09
008  08
007  07
006  06
005  05
004  04
003  03
002  02
001  01
000  00

1.0135432195 02
0.000125753213 03
1.0002517222
0100234371759000.06
2724576007
1234567890.123
0.012345679012311
3.7159253753513
2.7152918258659
13
PROPERTIES OF A POLYGON AREA.- It is not often that I can bring you a well-written SR-52 program. Here is one, written by Dean Athans of Monrovia, California. The program computes total area, centroid location (X,Y), moments of inertia about indicated X and Y axis (I_x, I_y) and moments of inertia about centroidal X and Y axis (I_x, I_y) for a polygon. The program allows rapid solution of a possibly complex problem encountered in mechanical and civil engineering. A polygon is defined as a plane area having only straight-line edges. The only limitation of this program is that THE ENTIRE POLYGON HAS TO BE LOCATED IN THE FIRST QUADRANT. That means all coordinates must be zero or positive.

The program calculates incremental trapezoidal areas (between each edge and the X-axis) and moments of inertia about a common, X, axis between successive coordinates. The equations used are:

\[
A = \frac{1}{2} (Y_1 + Y_2) (X_2 - X_1)
\]

\[
\bar{Y} = \frac{(Y_1 + Y_2)^2 - Y_1 Y_2}{3(Y_1 + Y_2)}
\]

\[
I_x = \frac{(X_2 - X_1) (Y_1 + Y_2) (Y_1^2 + Y_2^2)}{12}
\]

\[
\bar{I}_x = I_x - A\bar{Y}^2
\]

Similar equations are applied for \(\bar{X}, \bar{I}_y\) and \(\bar{I}_y\).

To run the program, coordinates are entered sequentially as X, A, Y, RUN. An example below shows what to expect.

```
O, A, O, RUN
O, A, 6, RUN
4, A, 6, RUN
4, A, O, RUN
1, A, O, RUN
1, A, 2, RUN
3, A, 2, RUN
3, A, 5, RUN
1, A, 5, RUN
1, A, 0, run
O, A, 0, RUN
C............. 18 area
RUN ............. 2.83.. \(\bar{Y}\)
RUN ............. 210 \(\bar{I}_x\)
RUN ............. 65.5 \(\bar{I}_x\)
D............. 18 area
RUN ............. 2 \(X\)
RUN ............. 102 \(\bar{I}_y\)
RUN ............. 30 \(\bar{I}_y\)
```

(over)
As you will notice, Dean uses the stack in steps 063 through 074, the same sequence I have been promoting in v5nlp4. Only, Dean wrote this program back in 1978. For TI-59 users who never owned or came near an SR-52, and who would like to translate this program, a few remarks are in order:

The SR-52 allows to use the second function of the digits as labels, just as the TI-59 does. Only, in the SR-52 listing they appear as '0', '1', '2', etc.

DSZ in the SR-52 is on register 0 only. It is therefore not stated. Thus, a sequence such as DSZ E in the SR-52 should be in 59-ese DSZ 0 E...

In the 52, the IND command comes before the function, in the 59 it comes after the function. Thus, IND STO 0 0 becomes STO IND 00.

Register operations in the 52 require three steps, in the 59 only two.

Branching in the 52 is possible if the result of a computation is zero.

With the IFZ command it is assumed that the (non-accessible) t-register is always equal to zero. In the 59 you either replace IFZ with CP x=t (the latter listed as EQ) OR you re-write the program such that one value is dumped into the t-register and the other appears in the display, after which you simply say x=t. In the latter case you gain speed and program steps, but it requires a little more work on your part.

The sequence \( x^2 \sqrt{x} \) was used to synthesize the ABS function on the SR-52. It lacks such a command.

Watch your step (pun intended) with the LBL LBL C' trick on step 188 if you want the correct, intended result on the TI-59.

The unmodified SR-52 has only 20 data registers 00 through 19, but users manipulate the hierarchial stack 60 through 69 and the program memory 70 through 99. I modified my SR-52 by adding one more memory chip. This supplies the missing registers to 59 and allows me to have two programs simultaneously in the calculator: the regular one with 224 steps and an additional one with 160 steps. A small toggle switch on the top of the calculator permits switching between the two programs, back and forth. Each program can be recorded on a separate mag card.

The SR-52 does not allow for recording of data registers, but several programs have been developed to transfer data register contents to program memory, from where they may be recorded on a mag card. The card also records a re-transfer program sequence.

-----------------------------------------

FRACTION REDUCTION ON THE SR-56. Samuel G. Allen tells me that there are several models of Casio calculators that will work with proper and improper fractions and with mixed numbers. When doing so, they always leave the fraction in its lowest terms. This may be done by removing a common factor from the numerator and denominator. I am not absolutely sure that the method he recommends, which he says comes from Excursions in Number Theory by Ogilvy and Anderson, is generally accepted by mathematicians:

\[
\begin{align*}
\frac{1}{6} &= \frac{1}{4} \\
\frac{2}{5} &= \frac{2}{5} \\
\frac{1}{5} &= \frac{1}{5} \\
\frac{143}{85} &= \frac{170}{1056}
\end{align*}
\]

But he also offers a legitimate method in the form of an SR-56 program:

00: RCL 1 DIV RCL 2 STD 3 = INT STD 0 X RCL 2 = INV SUM 1 RCL 1
20: EXC 3 -( CE DIV RCL 3 ) INT X RCL 3 = INV X=I 20
38: RCL 3 INV PROD 1 INV PROD 2 CLR R/S RST

User instructions: Clear the t-register. Load the numerator in register 1 and the denominator into register 2. Start program execution at location 00. Numerator and denominator will be retained in R1 and R2, but will be reduced to their lowest terms. In the case of an improper fraction, the fraction will be returned as a mixed number, with the integer part in register 0.

Example: Reduce 1870/544 to its lowest terms. Load 1870 into R1 and 544 into R2. Find 3 in register 0, 7 in R1 and 16 in R2. The reduced fraction is 3 + 7/16.

The method used is Euclid's Algorithm for Greatest Common Divisor.

-----------------------------------------
KEYBOARD HIR revisited. - In response to the article by the same name in v5n3p9, Dave Leising of Grand Rapids, Michigan, sent me the following keyboard HIR program. It permits the direct set-up and execution of any Hierarchial Internal Register (HIR) instruction, op code 82, without having to "bit-fiddle" it into program memory. Use of the printer is required and provides user prompting. The program uses a self-altering technique to synthesize and then execute an indirect HIR instruction, impossible to accomplish with any other method. (Look how Richard Snow does an IND HIR in "more Hirconia.")

USER INSTRUCTIONS:
1. In 6 OA 17 key in this program and record it with the same partitioning.(side 1)
2. Initialize by pressing A. All subsequent operations are done with R/S.
3. Printer will prompt for entry of a two-digit code XY, representing the HIR code you want to use, followed by print out of current code residing in memory with a question mark. If you want to use the existing code press R/S. If you want to use another value, enter that and press R/S. For example, if you would like to do "STO HIR 7", enter 7. (the 0 in 07 is understood)
4. The entered value is printed, followed by a prompt for an operand. That means, you may now enter the value you want stored, for example. Enter and press R/S. Operands are used for all HIR operations, except, of course, for RCL HIR. If you enter one, it will be ignored. Just press R/S.
5. The entered operand will be printed, followed by the print out of the result of the chosen HIR operation. Press R/S again to return to the start position, in order to perform a new operation. Return to step 3, above.

Segment 000 to 031 saves the contents of pending operations stack, so that arithmetic used in program will not disturb results.
Segment 032 to 087 prompts for entry of HIR code XY, retrieves and prints current value of same from memory with a ?, halts for entry and prints entered value.
Segment 088 to 103 tests the entered HIR code XY for legal limits (00 through 99) returns to location 032 (entry prompt) if illegal value is entered. It also stores the valid entry in data memory location 09.
Segment 104 to 140 prompts for entry of operand, retrieves and prints current value of operand from memory with a ?, halts for entry, prints entered value, stores operand in data memory location 10.
Segment 141 to 145 clears the current instruction from program-segment-equivalent, data register 59 using present instruction mask in data register 00.
Segment 146 to 152 retrieves new HIR code XY from data register 09 and converts it to a new instruction mask.
Segment 153 to 154 updates present instruction mask with a new one.

(over)
Keyboard HIR, continued.
Segment 155 to 156 inserts new instruction in program-segment-equivalent data register 59.
Segment 157 to 188 restores pending operation stack.
Segment 189 to 191 repartitions memory so that program-segment-equivalent data register 59 becomes program-significant.
Segment 192 to 193 recalls previously entered operand to the display register.
Segment 194 to 195 calls and executes alterable program segment.
Segment 196 prints the result of an operation.
Segment 197 to 199 repartitions memory so that alterable program segment returns to data-significance.
Segment 200 to 202 clears, halts and returns for another program cycle.
Segment 205 to 206 sets floating point mode
Segment 207 to 209 insures correct initial partitioning.
Segment 210 clears all masks, operands, etc.
Segment 211 to 223 initializes program-segment-equivalent data register 59 with the basic HIR subroutine.
Segment 224 resets to begin program execution.

--------------------------------------------------------------------------------------------------------------------------

TELEPHONE RATE TIMER.- This program, written by Emil Regelman and John Worthington, and later enhanced by Robert Snow, computes and shows a running display of the cost of a non-operator-assisted telephone call, based on the rates listed in the Maryland Suburban telephone directory. The program can easily be adapted to rates elsewhere. The program assumes one rate for the first minute, and another rate for all subsequent minutes. It takes into account the time of day and thus the rate reduction (100 % day time, 60 % evening, 35 % weekend rates) and the distance in miles between the caller and the respondent. The program allows for entry of special rates, if known. A constantly changing exponent will show at the same time the number of seconds remaining at that particular charge.

Instructions:
1. Enter program, both sides of mag card. Program uses TI-59 only, no printer.
2. Enter estimated miles between you and your respondent.
3. Dial your call on the telephone.
4. Press A, B or C, depending on the applicable rate. (Day, evening, weekend)
   Or enter special rate and press D.
5. When your respondent picks up the phone on the other end, press E.
6. During your telephone call the display will show the running cost and the seconds remaining at that charge. Every minute the charge will change. When you finally stop talking and hang up, press R/S. The display will show the final charge and the seconds remaining. Ignore the latter.

NOTE: When you use the special rate and key D, mostly the first minute carries a different rate than the subsequent minutes. Thus, enter the starting rate and press D. At exactly one minute, while talking, press R/S, enter the new rate and press D again. Rates are always entered in dollars.

Adjustments:
The clock may be adjusted by substituting PAUses for NOPs. NOP = 500 msec, PAU = 17 msec. If the clock cycle is shorter than 1 minute, substitute PAU for NOP at one or more of the following locations: 021, 066, 107, 148, or 189. If the clock cycle is longer than 1 minute, substitute NOP for PAU at one of these locations: 008, 045, 086, 168, or 209. Additional trimming may be done by changing the NOPs at steps 251 to 255 to another command, such as IXI (35 msec), provided that the new command does not change the result in the display.

The program is absolute-addressed. To preserve the order, change PAU to NOPs, but DO NOT DELETE them.

The mileage data begins at step 275. The number shown is the maximum mileage at the rate shown, beginning at step 357. The rate is coded as XXYY, where XX is the first minute rate in cents and YY is the subsequent minute's rate, also in cents.

THE ZEROS AT STEPS 247-248 ARE ESSENTIAL, AND MUST NOT BE DELETED OR CHANGED.

Program on next page.
tanh anyone? In v2n11pl of 52-Notes, Fred Fish found that PGM 05 of the ML-module could be used to compute sinh and cosh. His routines were:

STD 02 PGM 05 C’ for sinh and STD 02 PGM 05 E’ for cosh. If you need Reg 02 for another purpose, you can also use:

PGM 05 SBR 110 and PGM 05 SBR 006 for sinh and cosh respectively, with the argument in the display. It runs a bit faster. One warning: E’ and SBR 006 leave the calculator in radian mode when done.

Now Paul Berg from Cedar Falls, Iowa, has carried these routines a little further, such that you may also generate the tanh.

User instructions: For sinh, enter the argument and press A.
For cosh, enter the argument and press B.
For tanh, enter the argument and press C.

### ANGLE CONVERTERS
Re the routine on v5n3p4, Ralph Donnelly, of Martinsburg, West Virginia, says it can be done shorter and faster. And the admonition to enter the angle in decimal degrees is unnecessary in both routines, he further claims. His routine runs as follows:

```plaintext
LBL A PRT DIV 360 = INV INT X 360 = PRT R/S
```
MORE HIRmania. - Re-v5n3p9. That there is a HIRmania (word coined by George Vogel) raging in the land should be no secret to you, witness the many HIR programs in this issue. Members seem to have a great desire to find out everything there is to the HIRs. Programs "just to fool around with them" are in great favor. One of them was Dick Blagney's in v5n3p9. Richard Snow provides us with a less elaborate method you can use from the keyboard. It even allows you to do an IND HIR !!!

At some spot in program memory (at the very beginning maybe?) put some HIRs. By this I mean: go into LRN and press STO 82 STO 82 STO 82, etc., ten times. Then go back and delete the STOs.

Now go out of LRN and press RST. We said already before that HIR 8 and OP 4 are the same register, only HIR 8 requires 3 left-most dummy digits. So, to store something in HIR 8, we can do: 4545454545 OP 04. If we now recall HIR 8 we should see 0.004545454545. (The last few digits rounded off, of course)

Now, here is the "RCL HIR" trick: press SST, followed by 18. Lo and behold, the display shows the contents of HIR 8.

To show the same thing by an IND means, do as follows: Enter 18, STO 00. Press SST IND 00 and the contents of HIR 8 is displayed. What you just did is an IND HIR 18, which means "RCL IND HIR 8", with reg 00 as the pointer register.

All other functions on the HIRs can be done this way, directly or indirectly. It is a dynamite way of checking contents of HIR registers while you are programming. Just put a block of HIRs somewhere in an empty part of your program and you can do anything you please.

To recapitulate: if you want to store in, say, HIR 7, the number 222222 : Enter 222222, press RST SST 7. Then to recall same: CLR SST 17. Then you want to add 5 to HIR 7: enter 5, SST 37. To recall the result: SST 17.

ERRATA:- Bill Skillman, who, like all EEs I have known, is a stickler for accuracy, sent me the following remarks:

v5n3p12: Oops, don't do DSZ on register 40. The program interprets it as DSZ IND.
v5n3p11: User instructions for recording: partition 6 OP 17, record bank 1.
v5n3p13: Instruction 5: if misreads, force side 1 into bank 4 with 4 +/-.
v5n3p13: Note 1: although there are only 8 zeros, the spacing of the numbers is 10 steps. So you are correct in dividing by 10 in step 161-162. A quick timing leads me to think step 165 should be 4, but I am not sure.
v5n3p15: Ah-ha! There is a requirement to zero reg 09, but only if the first input is zero! This truth may never be discovered by mortal man!
v5n3p9: Will we ever agree on the HIRs used by these operations? I rechecked it carefully and have constructed the attached table. The number of HIRs used depends not only on the parenthesis but on the OPs too.

<table>
<thead>
<tr>
<th>NESTED_OPS</th>
<th>HIRs_used</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP 01 ....</td>
<td>HIR 5</td>
</tr>
<tr>
<td>OP 02 ....</td>
<td>HIR 6</td>
</tr>
<tr>
<td>OP 03 ....</td>
<td>HIR 7</td>
</tr>
<tr>
<td>OP 04 ....</td>
<td>HIR 8</td>
</tr>
<tr>
<td>OP 12 ....</td>
<td>HIR 8 and first 3 available HIRs.</td>
</tr>
<tr>
<td>OP 13 ....</td>
<td>First 4 available HIRs.</td>
</tr>
<tr>
<td>OP 14 ....</td>
<td>HIR 8 and first 3 available HIRs.</td>
</tr>
<tr>
<td>OP 15 ....</td>
<td>HIR 8 and first 3 available HIRs.</td>
</tr>
<tr>
<td>P/R .......</td>
<td>First available HIR, r in HIR 7, r in HIR 8.</td>
</tr>
<tr>
<td>INV P/R .....</td>
<td>First 2 available HIRs, x in HIR 7, y in HIR 8.</td>
</tr>
<tr>
<td>INV Σ+ .....</td>
<td>x = 1 in HIR 7, -x = y in HIR 8</td>
</tr>
<tr>
<td>Σ+ ..........</td>
<td>x = 1 in HIR 7, y = x in HIR 8.</td>
</tr>
<tr>
<td>D.MS ........</td>
<td>First 2 available HIRs, 100 X frac(x) in HIR 8</td>
</tr>
<tr>
<td>INV D.MS ......</td>
<td>First 2 available HIRs, 100 X (Int (x) +.6 frac (x)) in HIR 8.</td>
</tr>
<tr>
<td>OP 11 ......</td>
<td>First 2 available HIRs.</td>
</tr>
<tr>
<td>INV X .......</td>
<td>First available HIR.</td>
</tr>
</tbody>
</table>
Putting a die is rather simple. I learned a good deal about using the Master Library modules, and especially Module 15, from Fred Brown's "Survival Guide" for the TRS-80/89 Master Library." Fred tells me that he still has about 30 copies of the manual, which I think is the way to go. He also recommends that you consult with other experienced programmers who are familiar with the use of the modules. In order to write an efficient "digi-station" routine, in which we probably could use a program to find a simple digit in some one of our programs, we could use an "direct estimator" which can be easily done using the following routine.

Enter a single digit between 0 and 9. Then write into program the following routine:

```
NEW E F

END
```

The next thing is a "digit selector." That means we should be able to extract a single digit from a randomly selected row. The selection for this operation should be the same as the selection for the simple digit. This digit should produce the printing of its corresponding message, as with "DIGEST ELEMENTS." There are five possible messages, whose alpha-code may be stored in 20 registers. Easily done on a TRS-80.

If all this seems formidable, let's go on, and see that in reality this can be a rather straightforward problem having several solutions. We will present one here. It's rather simple and is one of our own design. In order to write an efficient "digit extractor" routine, in which we probably could use a program to find a single digit in some one of our programs, we could use an "direct estimator" which can be easily done using the following routine.

Enter a single digit between 0 and 9. Then write into program the following routine:

```
NEW E F

END
```

The next thing is a "digit selector." That means we should be able to extract a single digit from a randomly selected row. The selection for this operation should be the same as the selection for the simple digit. This digit should produce the printing of its corresponding message, as with "DIGEST ELEMENTS." There are five possible messages, whose alpha-code may be stored in 20 registers. Easily done on a TRS-80.
3-D TIC-TAC-TOE

by Robert and Richard SNOW.

This game may be played on the TI-59, with or without the PC100.

Instructions:
1. Initialize, press E.
2. To print a 3-D TTT board, press D.
   This board may be printed anytime during the game, without disturbing the game itself.
3. If you want the machine to move first, enter zero or CLR and press A or R/S.
4. If you want to move first, enter a number 1 to 27 and press A or R/S.
   Machine returns with counter-move in the display. If that number is flashing, the machine wins.
   A steady display is OK.
   A flashing 999...99 means an illegal move.
   With the printer, pressing D anytime will produce a 3-D TTT board, on which the player's pieces are printed as "O"s and the machine's pieces as "X"s.

NOTE: Squares 1 through 9 are the top grid.
   Squares 10 through 18 are the middle grid.
   Squares 19 through 27 are the bottom grid.