* TI PPC NOTES * v5N4/5, 1980

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 HIR 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 already 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 ALPHABETHICAL 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 anybdoy 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.

TABLE OF CONTENTS

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 Chapter of the HP-65 Users Club. Newsletter is called DISPLAY. Editor is Heinrich Schnepf, 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 colaborators 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 succeeed." 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 Brettinger, 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 Germany. 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 "brainware." 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!

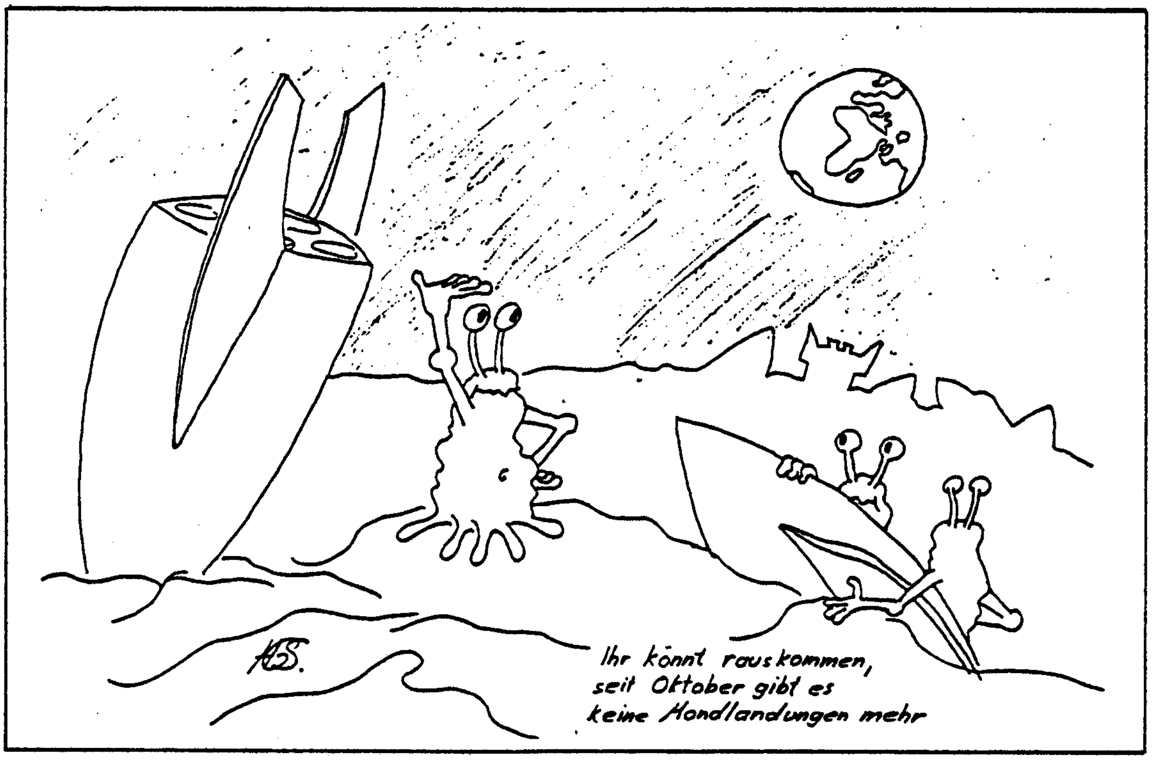
ZEPRA.- Another "elite" group, this time without formal dues. Only active contributions in the form of programs, routines, algorithms. Prospective members are chosen by the group for his/her past record as a programmer. It is very difficult to become a member. Editor: H. Zupp, Subbelstrasse 30, D-5000,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

TI-59/PC100 TRICKS.- A series of compilated 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 require 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:

$R1 = (a) / (2 \pi f C)$ $\qquad$ $R2 = (1/a) / 2 \pi f C$

with

$a = 1 / 2 \ \text{alog}\ dB/20)$

with dB = boost, C in Farads and f in Hz.

$dB = 20 \log (\sqrt{R2/R1}) / 2$

$$fo = \sqrt{ ((\sqrt{R2/R1}/2) / 2 \pi C R1) ((2 / \sqrt{R2/R1}) / 2 \pi C R2) }$$

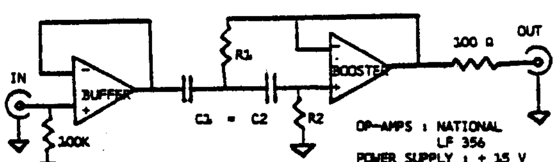
BASS BOOSTER- Terry Mickelson is the author of this extremely practical program. That is, if you are an EE you can make good use of it. Luckily, a full 70 % of our members are EEs, so that I might be justified to bring you an EE program once in a while. Terry lives in Duncan, B.C. Canada.

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 Mhz!!!) 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.

BASS BOOSTER- 2ND ORDER SALLEN & KEY, CHEBYSHEV HIGH PASS.




```
      2 × π =     62
0000001614  DB    63
0000002346  HZ    64
0000004121  UF    65
0000003502  R1    66
0000003503  R2    67
0000000261  1%    68
2020202020  ----- 69
```

```
000  91 R/S
001  76 LBL
002  15 E
003  86 STF
004  03  03
005  91 R/S
006  76 LBL
007  10 E'
008  28 LOG
009  42 STO
010  03  03
011  53 (
012  59 INT
013  22 INV
014  28 LOG
015  65 ×
016  53 (
017  09  9
018  06  6
019  35 1/X
020  22 INV
021  28 LOG
022  45 Y^X
023  53 (
024  43 RCL
025  03  03
026  22 INV
027  59 INT
028  65 ×
029  09  9
030  06  6
031  54 )
032  58 FIX
033  00  00
034  88 DMS
035  54 )
036  58 FIX
037  02  02
038  52 EE
039  22 INV
040  52 EE
041  54 )
042  22 INV
043  58 FIX
044  92 RTN
045  76 LBL
046  11 A
047  87 IFF
048  03  03
049  00  00
050  53  53
051  42 STO
052  11  11
053  43 RCL
054  11  11
055  81 RST
056  76 LBL
057  16 A'
058  87 IFF
059  03  03
060  00  00
061  64  64
062  42 STO
063  14  14
064  43 RCL
065  14  14
066  81 RST
067  76 LBL
068  12 B
069  87 IFF
070  03  03
071  00  00
072  75  75
073  42 STO
074  12  12
075  43 RCL
076  12  12
077  81 RST
078  76 LBL
079  17 B'
080  87 IFF
081  03  03
082  00  00
083  86  86
084  42 STO
085  15  15
086  43 RCL
087  15  15
088  81 RST
089  76 LBL
090  13 C
091  76 LBL
092  18 C'
093  87 IFF
094  03  03
095  00  00
096  99  99
097  42 STO
098  13  13
099  43 RCL
100  13  13
101  81 RST
102  76 LBL
103  14 D
104  07  7
105  69 OP
106  17  17
107  43 RCL
108  63  63
109  69 OP
110  04  04
111  43 RCL
112  11  11
113  69 OP
114  06  06
115  43 RCL
116  64  64
117  69 OP
118  04  04
119  43 RCL
120  12  12
121  69 OP
122  06  06
123  43 RCL
124  65  65
125  69 OP
126  04  04
127  43 RCL
128  13  13
129  69 OP
130  06  06
131  98 ADV
132  43 RCL
133  11  11
134  55 ÷
135  02  2
136  00  0
137  95 =
138  22 INV
139  28 LOG
140  35 1/X
141  55 ÷
142  02  2
143  95 =
144  42 STO
145  01  01
146  35 1/X
147  42 STO
148  02  02
149  43 RCL
150  12  12
151  65 ×
152  43 RCL
153  13  13
154  65 ×
155  43 RCL
156  62  62
157  95 =
158  42 STO
159  00  00
160  35 1/X
161  65 ×
162  43 RCL
163  01  01
164  95 =
165  42 STO
166  14  14
167  43 RCL
168  02  02
169  55 ÷
170  43 RCL
171  00  00
172  95 =
173  42 STO
174  15  15
175  43 RCL
176  14  14
177  10 E'
178  48 EXC
179  14  14
180  32 X:T
181  43 RCL
182  66  66
183  69 OP
184  04  04
185  32 X:T
186  69 OP
187  06  06
188  43 RCL
189  68  68
190  69 OP
191  04  04
192  43 RCL
193  14  14
194  69 OP
195  06  06
196  43 RCL
197  67  67
198  69 OP
199  04  04
200  32 X:T
201  43 RCL
202  15  15
203  10 E'
204  48 EXC
205  15  15
206  32 X:T
207  69 OP
208  04  04
209  32 X:T
210  69 OP
211  06  06
212  43 RCL
213  68  68
214  69 OP
215  04  04
216  43 RCL
217  15  15
218  61 GTO
219  03  03
220  16  16
221  76 LBL
222  19 D'
223  07  7
224  69 OP
225  17  17
226  43 RCL
227  65  65
228  69 OP
229  04  04
230  43 RCL
231  13  13
232  69 OP
233  06  06
234  65 ×
235  43 RCL
236  62  62
237  95 =
238  42 STO
239  02  02
240  43 RCL
241  66  66
242  69 OP
243  04  04
244  43 RCL
245  14  14
246  69 OP
247  06  06
248  35 1/X
249  32 X:T
250  43 RCL
251  67  67
252  69 OP
253  04  04
254  43 RCL
255  15  15
256  69 OP
257  06  06
258  65 ×
259  32 X:T
260  95 =
261  34 ΓX
262  35 1/X
263  65 ×
264  02  2
265  95 =
266  42 STO
267  03  03
268  35 1/X
269  42 STO
270  04  04
271  28 LOG
272  65 ×
273  02  2
274  00  0
275  95 =
276  98 ADV
277  32 X:T
278  43 RCL
279  63  63
280  69 OP
281  04  04
282  32 X:T
283  58 FIX
284  02  02
285  69 OP
286  06  06
287  22 INV
288  58 FIX
289  43 RCL
290  04  04
291  55 ÷
292  43 RCL
293  02  02
294  55 ÷
295  43 RCL
296  15  15
297  65 ×
298  43 RCL
299  03  03
300  55 ÷
301  43 RCL
302  02  02
303  55 ÷
304  43 RCL
305  14  14
306  95 =
307  34 ΓX
308  32 X:T
309  43 RCL
310  64  64
311  69 OP
312  04  04
313  32 X:T
314  58 FIX
315  02  02
316  69 OP
317  06  06
318  22 INV
319  58 FIX
320  43 RCL
321  69  69
322  69 OP
323  01  01
324  69 OP
325  02  02
326  69 OP
327  03  03
328  69 OP
329  04  04
330  69 OP
331  05  05
332  69 OP
333  00  00
334  25 CLR
335  98 ADV
336  98 ADV
337  98 ADV
338  81 RST
```

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 PGM 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 + 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. (steps 059 to 068) The offset is again subtracted from register 00. (HIR 17 +/- 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 (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 0 0. 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 pending 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 developes 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 ususally 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 C^2 *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 ALPHABETHICAL 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 my 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 adressing 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 Grogono 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 lastly, 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 3<u>AA</u> *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.

<u>SR-52 LISTING ON A TI-59</u>.- *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 MAG CARDS, ALL 4 BANKS.

```
ALPHA REG LIST:

   NUMERIC  REG ALPHA
1755000000.  10 E'
  13000000.  11  A
  14000000.  12  B
  15000000.  13  C
  16000000.  14  D
  17000000.  15  E
1365000000.  16 A'
1465000000.  17 B'
1565000000.  18 C'
1665000000.  19 D'
 263440000.  20 1/X
         0.  21
2431420000.  22 INV
2731440000.  23 LNX
1517000000.  24 CE
1527350000.  25 CLR
          .  26
2431420000.  27 INV
2732220000.  28 LOG
4473000000.  29 X!
5344000000.  30 √X
3351040200.  31 P*31
3624310000.  32 SIN
1532360000.  33 COS
3713310000.  34 TAN
4452450000.  35 X√Y
2431160000.  36 IND
1630360000.  37 DMS
1663350000.  38 D/R
3363350000.  39 P/R
4470000000.  40 X²
2237320000.  41 GTO
3637320000.  42 STO
3515270000.  43 RCL
3641300000.  44 SUM
4566000000.  45 YX
2714270000.  46 LBL
1530360000.  47 CMS
1744150000.  48 EXC
3335160000.  49 PRD
3637210000.  50 STF
3614350000.  51 SBR
1717000000.  52 EE
  55000000.  53  (
  56000000.  54  )
  72000000.  55  ÷
3537310000.  56 RTN
2124440000.  57 FIX
1636460000.  58 DSZ
  53000000.  59  π
2421210000.  60 IFF
3351070200.  61 P*61
3351070300.  62 P*62
3351070400.  63 P*63
3351070500.  64 P*64
  50000000.  65  ×
          .  66
1065000000.  67 7'
1165000000.  68 8'
1265000000.  69 9'
2421170000.  70 IFE
3351080200.  71 P*71
3351080300.  72 P*72
3351080400.  73 P*73
3351080500.  74 P*74
  20000000.  75  -
3351080700.  76 P*76
 665000000.  77 5'
 665000000.  78 5'
 765000000.  79 6'
2421470000.  80 IF+
2327370000.  81 HLT
3351090300.  82 P*82
3351090400.  83 P*83
3351090500.  84 P*84
  47000000.  85  +
3536370000.  86 RST
 265000000.  87 1'
 365000000.  88 2'
 465000000.  89 3'
2421460000.  90 IFZ
3541310000.  91 RUN
         0.  92
  40000000.  93  .
4763200000.  94 +/-
  64000000.  95  =
3517131600.  96 READ
2736370000.  97 LST
3335370000.  98 PRT
3313330000.  99 PAP
```

```
000  09   9      052  06   6      104  32  X:T
001  42  STO     053  22  INV     105  43  RCL
002  00   00     054  52  EE      106  66   66
003  91  R/S     055  69  OP      107  77   GE
004  76  LBL     056  02   02     108  10  E'
005  16  A'      057  01   1      109  97  DSZ
006  53   (      058  44  SUM     110  26   26
007  53   (      059  66   66     111  00   00
008  24  CE      060  01   1      112  48   48
009  55   ÷      061  00   0      113  92  RTN
010  01   1      062  00   0      114  76  LBL
011  00   0      063  49  PRD     115  10  E'
012  75   -      064  21   21     116  01   1
013  59  INT     065  34  √X      117  69  OP
014  42  STO     066  32  X:T     118  17   17
015  92   92     067  43  RCL     119  47  CMS
016  85   +      068  21   21     120  69  OP
017  93   .      069  59  INT     121  17   17
018  01   1      070  85   +      122  25  CLR
019  85   +      071  77   GE      123  76  LBL
020  59  INT     072  00   00     124  18  C'
021  65   ×      073  82   82     125  42  STO
022  93   .      074  16  A'      126  66   66
023  02   2      075  52  EE      127  69  OP
024  54   )      076  08   8      128  00   00
025  55   ÷      077  22  INV     129  98  ADV
026  01   1      078  52  EE      130  81  RST
027  00   0      079  61  GTO     131  43  RCL
028  85   +      080  00   00     132  00   00
029  43  RCL     081  84   84     133  91  R/S
030  92   92.    082  73  RC*     134  76  LBL
031  54   )      083  21   21     135  13   C
032  92  RTN     084  69  OP      136  72  ST*
033  76  LBL     085  04   04     137  00   00
034  11   A      086  00   0      138  97  DSZ
035  55   ÷      087  95   =      139  00   00
036  01   1      088  22  INV     140  01   01
037  00   0      089  44  SUM     141  31   31
038  22  INV     090  21   21     142  76  LBL
039  28  LOG     091  16  A'      143  15   E
040  95   =      092  16  A'      144  09   9
041  42  STO     093  52  EE      145  42  STO
042  21   21     094  06   6      146  00   00
043  05   5      095  22  INV     147  25  CLR
044  42  STO     096  52  EE      148  63  EX*
045  26   26.    097  69  OP      149  00   00
046  43  RCL     098  03   03     150  11   A
047  66   66.    099  69  OP      151  97  DSZ
048  16  A'      100  05   05     152  00   00
049  16  A'      101  02   2      153  01   01
050  16  A'      102  02   2      154  47   47
051  52  EE      103  04   4      155  81  RST
```

<u>TI-59 KEY CODES</u>.- 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 STO 00 9.200760869 STO 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., 0.002 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     033  76 LBL     066  15  15     099  33  33     132  15  E      165  46  46     198  99 PRT
001  52 EE      034  77 GE      067  81 RST     100  81 RST     133  52 EE      166  81 RST     199  82 HIR
002  91 R/S     035  82 HIR     068  76 LBL     101  76 LBL     134  82 HIR     167  76 LBL     200  17  17
003  76 LBL     036  07  07     069  68 NOP     102  44 SUM     135  41  41     168  75  -      201  99 PRT
004  12  B      037  81 RST     070  82 HIR     103  52 EE      136  81 RST     169  52 EE      202  82 HIR
005  82 HIR     038  76 LBL     071  16  16     104  82 HIR     137  76 LBL     170  82 HIR     203  18  18
006  01  01     039  87 IFF     072  81 RST     105  34  34     138  25 CLR     171  47  47     204  99 PRT
007  81 RST     040  82 HIR     073  76 LBL     106  81 RST     139  52 EE      172  81 RST     205  98 ADV
008  76 LBL     041  08  08     074  78 Σ+      107  76 LBL     140  82 HIR     173  76 LBL     206  91 R/S
009  22 INV     042  81 RST     075  82 HIR     108  54  )      141  42  42     174  85  +      207  76 LBL
010  82 HIR     043  76 LBL     076  17  17     109  52 EE      142  81 RST     175  52 EE      208  10 E'
011  02  02     044  13  C      077  81 RST     110  82 HIR     143  76 LBL     176  82 HIR     209  25 CLR
012  81 RST     045  82 HIR     078  76 LBL     111  35  35     144  35 1/X     177  48  48     210  82 HIR
013  76 LBL     046  11  11     079  88 DMS     112  81 RST     145  52 EE      178  81 RST     211  01  01
014  32 X:T     047  81 RST     080  82 HIR     113  76 LBL     146  82 HIR     179  76 LBL     212  82 HIR
015  82 HIR     048  76 LBL     081  18  18     114  69 OP      147  43  43     180  11  A      213  02  02
016  03  03     049  23 LNX     082  81 RST     115  52 EE      148  81 RST     181  82 HIR     214  82 HIR
017  81 RST     050  82 HIR     083  76 LBL     116  82 HIR     149  76 LBL     182  11  11     215  03  03
018  76 LBL     051  12  12     084  14  D      117  36  36     150  45 YX      183  99 PRT     216  82 HIR
019  42 STO     052  81 RST     085  52 EE      118  81 RST     151  52 EE      184  82 HIR     217  04  04
020  82 HIR     053  76 LBL     086  82 HIR     119  76 LBL     152  82 HIR     185  12  12     218  82 HIR
021  04  04     054  33 X²      087  31  31     120  79  X̄      153  44  44     186  99 PRT     219  05  05
022  81 RST     055  82 HIR     088  81 RST     121  52 EE      154  81 RST     187  82 HIR     220  82 HIR
023  76 LBL     056  13  13     089  76 LBL     122  82 HIR     155  76 LBL     188  13  13     221  06  06
024  52 EE      057  81 RST     090  24 CE      123  37  37     156  55  ÷      189  99 PRT     222  82 HIR
025  82 HIR     058  76 LBL     091  52 EE      124  81 RST     157  52 EE      190  82 HIR     223  07  07
026  05  05     059  43 RCL     092  82 HIR     125  76 LBL     158  82 HIR     191  14  14     224  82 HIR
027  81 RST     060  82 HIR     093  32  32     126  89  π      159  45  45     192  99 PRT     225  08  08
028  76 LBL     061  14  14     094  81 RST     127  52 EE      160  81 RST     193  82 HIR     226  91 R/S
029  67 EQ      062  81 RST     095  76 LBL     128  82 HIR     161  76 LBL     194  15  15
030  82 HIR     063  76 LBL     096  34 √X      129  38  38     162  65  ×      195  99 PRT
031  06  06     064  53  (      097  52 EE      130  81 RST     163  52 EE      196  82 HIR
032  81 RST     065  82 HIR     098  82 HIR     131  76 LBL     164  82 HIR     197  16  16
```

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 a 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.

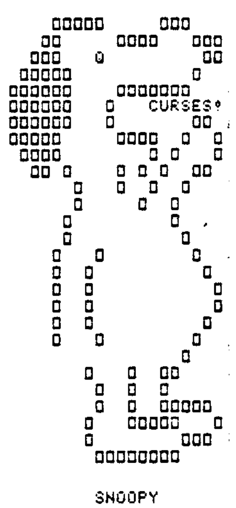
To repeat a drawing, just press R/S.

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.



REGISTERS

100.	61
101.	62
10000.	63
10100.	64
10101.	65
1000000.	66
1000001.	67
1010100.	68
1010101.3345	69
100000000.1541	70
100000001.	71
100000100.	72
100010001.	73
101000000.	74
101010000.	75
101010100.	76
101010101.	77
3536173673.	78
3400.	79

000	76	LBL
001	15	E
002	59	INT
003	65	×
004	43	RCL
005	60	60
006	95	=
007	92	RTN
008	76	LBL
009	16	A'
010	15	E
011	69	OP
012	01	01
013	92	RTN
014	76	LBL
015	10	E'
016	19	D'
017	43	RCL
018	63	63
019	76	LBL
020	17	B'
021	15	E
022	69	OP
023	02	02
024	92	RTN
025	76	LBL
026	18	C'
027	15	E
028	69	OP
029	03	03
030	92	RTN
031	76	LBL
032	19	D'
033	15	E
034	69	OP
035	04	04
036	69	OP
037	05	05
038	69	OP
039	00	00
040	92	RTN
041	76	LBL
042	14	D
043	22	INV
044	76	LBL
045	12	B
046	86	STF
047	01	01
048	32	X:T
049	08	8
050	69	OP
051	17	17
052	00	0
053	67	EQ
054	00	00
055	59	59
056	32	X:T
057	42	STO
058	60	60
059	01	1
060	16	A'
061	69	OP
062	03	03
063	43	RCL
064	76	76
065	17	B'
066	43	RCL
067	74	74
068	19	D'
069	43	RCL
070	62	62
071	16	A'
072	43	RCL
073	76	76
074	18	C'
075	43	RCL
076	65	65
077	19	D'
078	43	RCL
079	65	65
080	16	A'
081	43	RCL
082	79	79
083	69	OP
084	02	02
085	43	RCL
086	62	62
087	19	D'
088	43	RCL
089	69	69
090	16	A'
091	43	RCL
092	70	70
093	17	B'
094	43	RCL
095	63	63
096	19	D'
097	43	RCL
098	77	77
099	16	A'
100	69	OP
101	03	03
102	43	RCL
103	70	70
104	17	B'
105	43	RCL
106	74	74
107	19	D'
108	43	RCL
109	77	77
110	16	A'
111	43	RCL
112	71	71
113	17	B'
114	82	HIR
115	14	14
116	87	IFF
117	01	01
118	01	01
119	26	26
120	43	RCL
121	70	70
122	82	HIR
123	07	07
124	43	RCL
125	78	78
126	69	OP
127	04	04
128	69	OP
129	05	05
130	43	RCL
131	77	77
132	16	A'
133	43	RCL
134	64	64
135	82	HIR
136	07	07
137	19	D'
138	43	RCL
139	77	77
140	16	A'
141	43	RCL
142	76	76
143	18	C'
144	43	RCL
145	67	67
146	19	D'
147	43	RCL
148	69	69
149	16	A'
150	43	RCL
151	61	61
152	18	C'
153	69	OP
154	03	03
155	43	RCL
156	71	71
157	19	D'
158	43	RCL
159	64	64
160	16	A'
161	43	RCL
162	70	70
163	17	B'
164	43	RCL
165	73	73
166	18	C'
167	43	RCL
168	64	64
169	19	D'
170	43	RCL
171	66	66
172	17	B'
173	43	RCL
174	72	72
175	18	C'
176	43	RCL
177	66	66
178	19	D'
179	69	OP
180	02	02
181	43	RCL
182	63	63
183	18	C'
184	43	RCL
185	70	70
186	19	D'
187	69	OP
188	02	02
189	69	OP
190	04	04
191	69	OP
192	05	05
193	43	RCL
194	66	66
195	19	D'
196	01	1
197	16	A'
198	43	RCL
199	61	61
200	17	B'
201	43	RCL
202	63	63
203	19	D'
204	69	OP
205	02	02
206	01	1
207	16	A'
208	43	RCL
209	61	61
210	10	E'
211	01	1
212	16	A'
213	69	OP
214	04	04
215	69	OP
216	05	05
217	69	OP
218	05	05
219	43	RCL
220	61	61
221	19	D'
222	69	OP
223	02	02
224	01	1
225	16	A'
226	43	RCL
227	63	63
228	19	D'
229	43	RCL
230	75	75
231	17	B'
232	43	RCL
233	66	66
234	10	E'
235	43	RCL
236	67	67
237	18	C'
238	43	RCL
239	70	70
240	19	D'
241	43	RCL
242	61	61
243	17	B'
244	43	RCL
245	67	67
246	18	C'
247	69	OP
248	05	05
249	43	RCL
250	76	76
251	10	E'
252	43	RCL
253	69	69
254	82	HIR
255	03	03
256	18	C'
257	43	RCL
258	71	71
259	10	E'
260	43	RCL
261	68	68
262	19	D'
263	43	RCL
264	62	62
265	17	B'
266	43	RCL
267	77	77
268	18	C'
269	43	RCL
270	70	70
271	19	D'
272	98	ADV
273	06	6
274	69	OP
275	17	17
276	25	CLR
277	82	HIR
278	04	04
279	92	RTN
280	87	IFF
281	01	01
282	12	B
283	61	GTO
284	14	D
285	76	LBL
286	13	C
287	82	HIR
288	13	13
289	82	HIR
290	07	07
291	03	3
292	06	6
293	03	3
294	01	1
295	69	OP
296	02	02
297	69	OP
298	05	05
299	00	0
300	98	ADV
301	61	GTO
302	02	02
303	79	79
304	76	LBL
305	11	A
306	42	STO
307	01	01
308	82	HIR
309	04	04
310	00	0
311	91	R/S
312	12	B
313	43	RCL
314	01	01
315	92	RTN
316	61	GTO
317	11	A

LABELS

001	15	E
009	16	A'
015	10	E'
020	17	B'
026	18	C'
032	19	D'
042	14	D
045	12	B
286	13	C
305	11	A

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 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 π = and press A.
Enter x-increment as π div 4 = and press B.
Enter y-min as .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*
PGM 03 SBR 369 *same OP 02.*
PGM 03 SBR 375 *same OP 03.*
PGM 03 SBR 381 *same OP 04.*

Warning: these routines alter the content of reg 00.

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 DSZed 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.

MU-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.*

OVER.

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 accomodated in this partition? Press C for the answer.

2. a. To accomodate 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.

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

PARTITION	MAX NUMBER OF LINES
1 OP 17	1 1/2
2 OP 17	4
3 OP 17	6 1/2
4 OP 17	9
5 OP 17	11 1/2
6 OP 17	14
7 OP 17	16 1/2
8 OP 17	19
9 OP 17	21 1/2
10 OP 17	24

BANK #	REGISTERS
1	90 TO 99 + 160 PGM STEPS
2	60 TO 89
3	30 TO 59
4	00 TO 29

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 v5n2p12, 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 1/X X R/S DIV 2 = STO 01 X² - R/S DIV RCL
017: 00 = X:T 0 INV P/R DIV 2 = X:T √X 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)
Instructions: Enter C, press A. (RST R/S on the 57) Enter a, press R/S. Enter b, press R/S. See c displayed.
The program uses the P/R conversion and the t-register.

```
000: LBL A X:T R/S X:T P/R X:T - R/S = X:T INV P/R X:T RTN
```

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 with 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 v1n1p2 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 DMS HIR 03 HIR 04 OP 00 in which CLR DMS 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 V5N1P12).

STP	Code	Key	STP	Code	Key	STP	Code	Key	STP	Code	Key
000	76	LBL	024	59	59	049	43	RCL	073	43	RCL
1	11	A	5	97	DSZ	050	00	00	4	59	59
2	58	FIX	6	57	57	1	42	STO	5	75	-
3	00	0	7	00	0	2	57	57	6	53	(
4	42	STO	8	69	69	3	73	RCL	7	24	CE
5	00	00	9	58	FIX			IND	8	55	÷
6	42	STO	030	08	8	4	57	57	9	43	RCL
7	57	57	1	43	RCL	5	66	Pse	080	58	58
8	91	R/S	2	58	58	6	66	Pse	1	42	STO
9	72	STO	3	91	R/S	7	66	Pse	2	59	59
		IND	4	58	Fix	8	66	Pse	3	54	)
010	57	57	5	00	0	9	66	Pse	4	59	INT
1	97	DSZ	6	43	RCL	060	97	DSZ	5	65	X
2	57	57	7	00	00	1	57	57	6	43	RCL
3	00	0	8	42	STO	2	00	0	7	58	58
4	08	08	9	57	57	3	53	53	8	54	)
5	00	0	040	43	RCL	4	43	RCL	9	95	=
6	32	X:t	1	58	58	5	58	58	090	22	INV
7	43	RCL	2	22	INV	6	58	FIX	1	67	X=t
8	00	00	3	64	Prd	7	08	8	2	00	0
9	42	STO			IND	8	91	R/S	3	71	71
020	57	57	4	57	57	9	73	RCL	4	43	RCL
1	73	RCL	5	97	DSZ			IND	5	58	58
		IND	6	57	57	070	57	57	6	61	GTO
2	57	57	7	00	0	1	42	STO	7	00	0
3	42	STO	8	40	40	2	58	58	8	23	23

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 (gives direct address follow-up)--see TI-PPC NOTES V5N1P12. 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, V5N2P12); 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

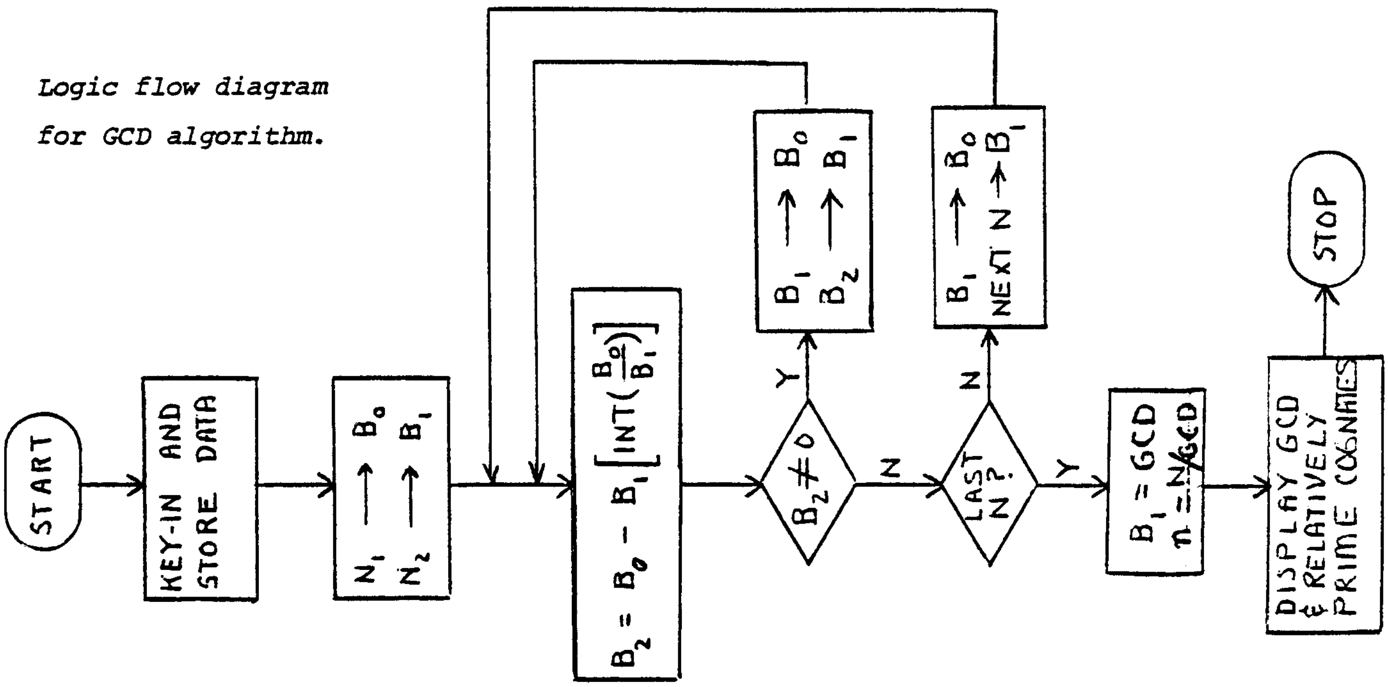
a) Given three N's 81, 63, 27: key 3 [A] 81 [R/S] 63 [R/S] 27 [R/S] ⟶ 9.00000000 (8 sec); then [R/S] ⟶ 9 ⟶ 7 ⟶ 3 ⟶ GCD

b) Given 224, 220, ··· (-4)··· 4 (56 terms in all): key 56 [A] 224 [R/S] 220 [R/S] ··· 4 [R/S] ⟶ 4.00000000 (2 min 20 sec); then [R/S] ⟶ 56 (25 sec) ⟶ 55 ··· ⟶ 1 ⟶ 4.00000000

Reference: Uspensky & Heaslet, "Elementary Number Theory" (McGraw-Hill Book Co.; my copy is 1939), pages 26-28.

See flow diagram on next page.

Logic flow diagram for GCD algorithm.



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. Schoenherr, Schlosz 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. Rudden, 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 flow charts.

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, $123.4567890123. Merely rounding it by Fix 2 would print it as the better-looking $123.46, 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 ) yx RCL 0 +/- = EE INV EE STO 3 prt pap 1 SUM 5 RCL 5 fix 0 prt
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 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, says Palmer O. Hanson. Prove is: v5n1p2, 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 tweek the clock by replacing one or more NOPs with other commands that take more time, such as IxI.. This in case your clock runs fast. If your clock runs slow, speed it up by replacing one our more pauses with NOPs. 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 to 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 PAU    120 66 PAU    180 66 PAU    240 66 PAU    300 43 RCL    360 50 I×I
001 55 55     061 01 1      121 02 2      181 04 4      241 05 5      301 57 57     361 29 CP
002 32 X:T    062 03 3      122 08 8      182 03 3      242 08 8      302 22 INV    362 65 ×
003 71 SBR    063 66 PAU    123 66 PAU    183 66 PAU    243 66 PAU    303 44 SUM    363 02 2
004 03 03     064 66 PAU    124 66 PAU    184 66 PAU    244 66 PAU    304 50 50     364 04 4
005 44 44     065 01 1      125 02 2      185 04 4      245 05 5      305 81 RST    365 42 STO
006 67 EQ     066 04 4      126 09 9      186 04 4      246 09 9      306 43 RCL    366 56 56
007 03 03     067 66 PAU    127 66 PAU    187 66 PAU    247 66 PAU    307 58 58     367 42 STO
008 90 90     068 66 PAU    128 66 PAU    188 66 PAU    248 66 PAU    308 49 PRD    368 57 57
009 76 LBL    069 01 1      129 03 3      189 04 4      249 68 NOP    309 53 53     369 01 1
010 15 E      070 05 5      130 00 0      190 05 5      250 68 NOP    310 61 GTO    370 42 STO
011 52 EE     071 66 PAU    131 66 PAU    191 66 PAU    251 68 NOP    311 02 02     371 58 58
012 66 PAU    072 66 PAU    132 66 PAU    192 66 PAU    252 68 NOP    312 91 91     372 43 RCL
013 66 PAU    073 01 1      133 03 3      193 04 4      253 68 NOP    313 76 LBL    373 55 55
014 01 1      074 06 6      134 01 1      194 06 6      254 68 NOP    314 13 C      374 22 INV
015 66 PAU    075 66 PAU    135 66 PAU    195 66 PAU    255 68 NOP    315 65 ×      375 67 EQ
016 66 PAU    076 66 PAU    136 66 PAU    196 66 PAU    256 68 NOP    316 01 1      376 03 03
017 00 0      077 01 1      137 03 3      197 04 4      257 68 NOP    317 03 3      377 28 28
018 02 2      078 07 7      138 02 2      198 07 7      258 68 NOP    318 42 STO    378 93 .
019 66 PAU    079 66 PAU    139 66 PAU    199 66 PAU    259 68 NOP    319 56 56     379 06 6
020 66 PAU    080 66 PAU    140 66 PAU    200 66 PAU    260 68 NOP    320 01 1      380 05 5
021 00 0      081 01 1      141 03 3      201 04 4      261 68 NOP    321 02 2      381 42 STO
022 03 3      082 08 8      142 03 3      202 08 8      262 25 CLR    322 42 STO    382 55 55
023 66 PAU    083 66 PAU    143 66 PAU    203 66 PAU    263 93 .      323 57 57     383 61 GTO
024 66 PAU    084 66 PAU    144 66 PAU    204 66 PAU    264 00 0      324 01 1      384 03 03
025 00 0      085 01 1      145 03 3      205 04 4      265 01 1      325 94 +/-    385 28 28
026 04 4      086 09 9      146 04 4      206 09 9      266 44 SUM    326 42 STO    386 32 X:T
027 66 PAU    087 66 PAU    147 66 PAU    207 66 PAU    267 51 51     327 58 58     387 93 .
028 66 PAU    088 66 PAU    148 66 PAU    208 66 PAU    268 93 .      328 29 CP     388 06 6
029 00 0      089 02 2      149 02 2      209 05 5      269 06 6      329 01 1      389 05 5
030 05 5      090 00 0      150 05 5      210 00 0      270 32 X:T    330 58 FIX    390 42 STO
031 66 PAU    091 66 PAU    151 66 PAU    211 66 PAU    271 43 RCL    331 02 02     391 55 55
032 66 PAU    092 66 PAU    152 66 PAU    212 66 PAU    272 51 51     332 65 ×      392 04 4
033 00 0      093 02 2      153 03 3      213 05 5      273 22 INV    333 69 OP     393 94 +/-
034 06 6      094 01 1      154 06 6      214 01 1      274 77 GE     334 10 10     394 22 INV
035 66 PAU    095 66 PAU    155 66 PAU    215 66 PAU    275 00 00     335 42 STO    395 58 FIX
036 66 PAU    096 66 PAU    156 66 PAU    216 66 PAU    276 00 00     336 53 53     396 22 INV
037 00 0      097 02 2      157 03 3      217 05 5      277 25 CLR    337 75 -      397 96 WRT
038 07 7      098 02 2      158 07 7      218 02 2      278 42 STO    338 59 INT    398 25 CLR
039 66 PAU    099 66 PAU    159 66 PAU    219 66 PAU    279 51 51     339 42 STO    399 58 FIX
040 66 PAU    100 66 PAU    160 66 PAU    220 66 PAU    280 01 1      340 50 50     400 02 02
041 00 0      101 02 2      161 03 3      221 05 5      281 44 SUM    341 95 =      401 32 X:T
042 08 8      102 03 3      162 08 8      222 03 3      282 50 50     342 42 STO    402 61 GTO
043 66 PAU    103 66 PAU    163 66 PAU    223 66 PAU    283 43 RCL    343 51 51     403 00 00
044 66 PAU    104 66 PAU    164 66 PAU    224 66 PAU    284 57 57     344 43 RCL    404 11 11
045 00 0      105 02 2      165 03 3      225 05 5      285 32 X:T    345 50 50     405 76 LBL
046 09 9      106 04 4      166 09 9      226 04 4      286 43 RCL    346 85 +      406 11 A
047 66 PAU    107 66 PAU    167 66 PAU    227 66 PAU    287 50 50     347 43 RCL    407 42 STO
048 66 PAU    108 66 PAU    168 66 PAU    228 66 PAU    288 67 EQ     348 51 51     408 55 55
049 01 1      109 02 2      169 04 4      229 05 5      289 03 03     349 95 =      409 61 GTO
050 00 0      110 05 5      170 00 0      230 05 5      290 06 06     350 65 ×      410 03 03
051 66 PAU    111 66 PAU    171 66 PAU    231 66 PAU    291 43 RCL    351 43 RCL    411 44 44
052 66 PAU    112 66 PAU    172 66 PAU    232 66 PAU    292 56 56     352 53 53
053 01 1      113 02 2      173 04 4      233 05 5      293 32 X:T    353 95 =
054 01 1      114 06 6      174 01 1      234 06 6      294 43 RCL    354 92 RTN    010 15 E
055 66 PAU    115 66 PAU    175 66 PAU    235 66 PAU    295 50 50     355 61 GTO    314 13 C
056 66 PAU    116 66 PAU    176 66 PAU    236 66 PAU    296 22 INV    356 00 00     359 18 C'
057 01 1      117 02 2      177 04 4      237 05 5      297 77 GE     357 11 11     406 11 A
058 02 2      118 07 7      178 02 2      238 07 7      298 00 00     358 76 LBL
059 66 PAU    119 66 PAU    179 66 PAU    239 66 PAU    299 00 00     359 18 C'
```

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 attendant 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 v5n1p10 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 exibits 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 ML-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 REVISITED in v5n3p6? Useful storage at these steps seems doubtful, as user RAM doesn't exist beyond step 959. 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 v5n2p9.

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 minumum 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.

Registers 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 X 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 registers 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 HIR 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.

```
000  24 CE      END
001  91 R/S
002  76 LBL     PREPARE HIR PRT CODE
003  16 A'
004  01  1
005  00  0
006  00  0
007  82 HIR
008  08  08
009  05  5
010  76 LBL     CONVERTER ROUTINE
011  18 C'
012  42 STO
013  01  01
014  01  1
015  00  0
016  00  0
017  82 HIR
018  48  48
019  97 DSZ
020  00  00
021  00  00
022  41  41
023  87 IFF     DECIMAL POINT
024  00  00
025  00  00
026  36  36
027  36 STF
028  00  00
029  04  4
030  00  0
031  82 HIR
032  38  38
033  61 GTO
034  00  00
035  61  61
036  82 HIR     EXIT LOOP 7
037  11  11
038  67 EQ
039  00  00

040  65  65
041  01  1      DIGIT CONVERTER
042  00  0
043  85  +
044  32 X:T
045  08  8
046  77 GE
047  00  00
048  51  51
049  02  2
050  85  +
051  01  1
052  54  )
053  59 INT
054  82 HIR
055  38  38
056  00  0
057  32 X:T     NEXT DIGIT
058  22 INV
059  59 INT
060  65  x
061  97 DSZ
062  01  01
063  00  00
064  14  14     PRT CODE
065  82 HIR
066  13  13
067  92 RTN
068  81 RST     13-DIGIT REG LIST
069  76 LBL
070  12  B
071  93  .
072  01  1
073  61 GTO
074  00  00
075  81  81
076  76 LBL
077  11  A
078  93  .
079  01  1

080  22 INV     REG #
081  86 STF
082  01  01
083  82 HIR
084  04  04
085  29 CP
086  69 OP
087  00  00
088  42 STO
089  00  00
090  25 CLR     DATA
091  73 RC*
092  00  00
093  82 HIR
094  03  03
095  69 OP
096  19  19
097  87 IFF     SKIP IF ZERO
098  07  07
099  00  00
100  68  68
101  67  EQ
102  02  02
103  31  31
104  32 X:T
105  22 INV
106  86 STF
107  00  00
108  22 INV     MINUS SIGN
109  77  GE
110  01  01
111  18  18
112  02  2
113  00  0
114  00  0
115  00  0
116  82 HIR     LOAD DSZ REGS
117  08  08
118  04  4
119  42 STO

120  01  01
121  01  1
122  42 STO
123  00  00
124  32 X:T
125  50 IxI
126  77  GE
127  01  01
128  44  44
129  32 X:T     FRACTION
130  93  .
131  00  0
132  01  1
133  32 X:T
134  77  GE
135  01  01
136  54  54     NO LEADING ZERO
137  65  x
138  71 SBR
139  00  00
140  27  27
141  61 GTO
142  01  01
143  57  57
144  55  -      GENERATE PRT CODE
145  28 LOG
146  59 INT
147  44 SUM
148  00  00
149  22 INV
150  28 LOG
151  52 EE
152  22 INV
153  52 EE
154  71 SBR
155  00  00
156  43  43
157  69 OP
158  01  01
159  16 A'

160  82 HIR     SPACE
161  06  06
162  16 A'
163  82 HIR
164  07  07
165  82 HIR
166  58  58
167  01  1      ROUND OFF FRACTION
168  54  )
169  58 FIX
170  02  02
171  52 EE
172  22 INV
173  58 FIX
174  65  x
175  02  2
176  18 C'
177  01  1
178  00  0
179  00  0
180  82 HIR
181  48  48
182  25 CLR
183  87 IFF     FINAL CHECK
184  00  00
185  01  01
186  97  97
187  97 DSZ
188  00  00
189  01  01     DECIMAL POINT
190  95  95
191  04  4
192  00  0
193  82 HIR
194  38  38
195  86 STF
196  00  00
197  82 HIR     REG NUMBER
198  14  14
199  55  ÷

200  02  2      APLHANUMERICS
201  18 C'
202  69 OP
203  04  04
204  69 OP
205  05  05
206  87 IFF
207  01  01
208  02  02
209  31  31
210  25 CLR
211  69 OP
212  00  00
213  82 HIR
214  13  13
215  75  -
216  69 OP
217  02  02
218  52 EE
219  22 INV

220  52 EE      OP CODE OR HIR CODE
221  95  =
222  67  EQ
223  02  02
224  29  29
225  82 HIR
226  13  13
227  82 HIR
228  06  06
229  69 OP
230  05  05
231  01  1      NEXT REG
232  82 HIR
233  34  34
234  82 HIR
235  14  14
236  61 GTO
237  00  00
238  86  86
```

```
13.01624222437     02
     DIGIT
.001001327332313  03
     ALPHA
-1.003517222       05
     REG-
100243637173500 0.06
     ISTER
27243637.          07
     LIST

1234567890.123     09
0.01234567890123   11
3.14159265359      12
2.718281828459     13
```

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:  -  2  )  X  90  )  DIV  RCL 09  )  PGM  12  C  PGM  12  A'  PGM  12  D  PGM
040:  12  D  PGM 12  E  X  RCL 09  X  RCL 08  DIV  2  =  RTN
```

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 $(\overline{X},\overline{Y})$, moments of inertia about indicated X and Y axis (I_x, I_y) and moments of enertia about centroidal X and Y axis $(\overline{I}_x, \overline{I}_y)$ 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 POLIGON 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:

$$A = 1/2\ (Y_1 + Y_2)\ (X_2 - X_1)$$

$$\overline{Y} = ((Y_1 + Y_2)^2 - Y_1Y_2)\ /\ 3(Y_1 + Y_2)$$

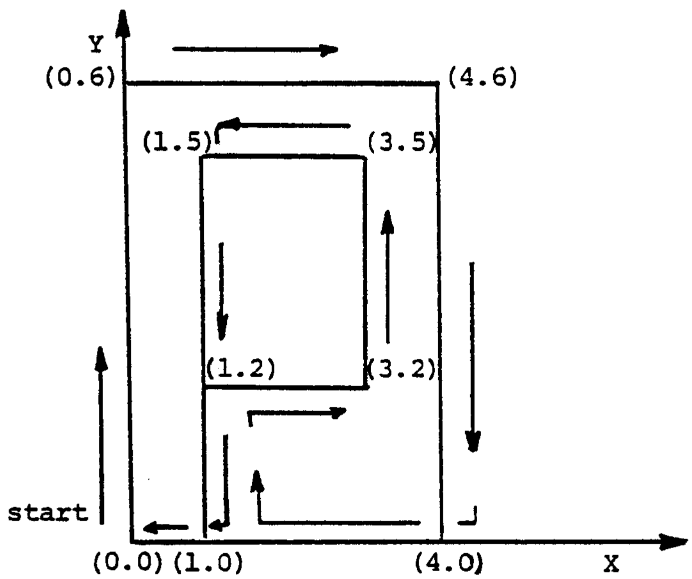
$$I_x = (X_2 - X_1)\ (Y_1 + Y_2)\ (Y_1^2 + Y_2^2)\ /\ 12$$

$$\overline{I}_x = I_x - A\overline{Y}^2$$

Similar equations are applied for $\overline{X}$, I_y and $\overline{I}_y$.

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

0, A, 0, RUN
0, A, 6, RUN
4, A, 6, RUN
4, A, 0, RUN
1, A, 0, RUN
1, A, 2, RUN
3, A, 2, RUN
3, A, 5, RUN
1, A, 5, RUN
1, A, 0, run
0, A, 0, RUN
C........... 18 area
RUN .... 2.83.. $\overline{Y}$
RUN .... 210 I_x
RUN .... 65.5 $\overline{I}_x$
D .... 18 area
RUN .... 2 $\overline{X}$
RUN .... 102 I_y
RUN .... 30 $\overline{I}_y$



000	46	LBL	032	02	2	064	18	C'	096	01	1	128	42	STO	160	09	9	192	40	X²
001	19	D'	033	65	×	065	48	EXC	097	00	0	129	00	0	161	94	+/-	193	30	√X
002	95	=	034	43	RCL	066	01	1	098	65	×	130	09	9	162	42	STO	194	81	HLT
003	36	IND	035	01	1	067	09	9	099	43	RCL	131	18	C'	163	00	0	195	36	IND
004	44	SUM	036	03	3	068	48	EXC	100	00	0	132	18	C'	164	08	8	196	43	RCL
005	00	0	037	54	)	069	01	1	101	08	8	133	42	STO	165	46	LBL	197	01	1
006	00	0	038	55	÷	070	08	8	102	55	÷	134	01	1	166	15	E	198	06	6
007	22	INV	039	03	3	071	48	EXC	103	02	2	135	00	0	167	01	1	199	55	÷
008	58	DSZ	040	55	÷	072	01	1	104	19	D'	136	42	STO	168	01	1	200	17	B'
009	15	E	041	43	RCL	073	06	6	105	55	÷	137	01	1	169	42	STO	201	95	=
010	56	RTN	042	00	0	074	56	RTN	106	18	C'	138	03	3	170	00	0	202	81	HLT
011	46	LBL	043	09	9	075	42	STO	107	06	6	139	94	+/-	171	00	0	203	18	C'
012	17	B'	044	46	LBL	076	01	1	108	65	×	140	42	STO	172	81	HLT	204	17	B'
013	36	IND	045	89	3'	077	03	3	109	53	(	141	01	1	173	46	LBL	205	40	X²
014	43	RCL	046	19	D'	078	42	STO	110	43	RCL	142	01	1	174	14	D	206	30	√X
015	01	1	047	56	RTN	079	01	1	111	01	1	143	94	+/-	175	02	2	207	81	HLT
016	08	8	048	46	LBL	080	07	7	112	08	8	144	48	EXC	176	18	C'	208	17	B'
017	56	RTN	049	10	E'	081	19	D'	113	40	X²	145	01	1	177	06	6	209	75	-
018	46	LBL	050	43	RCL	082	19	D'	114	85	+	146	05	5	178	18	C'	210	36	IND
019	16	A'	051	00	0	083	40	X²	115	43	RCL	147	42	STO	179	05	5	211	43	RCL
020	65	×	052	09	9	084	85	+	116	01	1	148	01	1	180	41	GTO	212	01	1
021	53	(	053	65	×	085	43	RCL	117	05	5	149	02	2	181	46	LBL	213	06	6
022	53	(	054	43	RCL	086	01	1	118	40	X²	150	18	C'	182	13	C	214	65	×
023	43	RCL	055	01	1	087	02	2	119	54	)	151	16	A'	183	03	3	215	18	C'
024	00	0	056	01	1	088	40	X²	120	19	D'	152	43	RCL	184	18	C'	216	18	C'
025	09	9	057	55	÷	089	95	=	121	10	E'	153	01	1	185	04	4	217	18	C'
026	40	X²	058	56	RTN	090	65	×	122	02	2	154	07	7	186	18	C'	218	40	X²
027	90	IFZ	059	46	LBL	091	10	E'	123	19	D'	155	42	STO	187	07	7	219	95	=
028	89	3'	060	11	A	092	01	1	124	16	A'	156	01	1	188	46	LBL	220	40	X²
029	75	-	061	19	D'	093	02	2	125	43	RCL	157	02	2	189	46	LBL	221	30	√X
030	43	RCL	062	19	D'	094	19	D'	126	01	1	158	42	STO	190	18	C'	222	81	HLT
031	01	1	063	46	LBL	095	43	RCL	127	00	0	159	00	0	191	17	B'			

(over)

(properties of a polygon.(continued)

As you will notice, Dean uses the stack in steps 063 through 074, the same sequence I have been promoting in v5n1p4. 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 ummodified 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 20 through 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:

$$\frac{16}{64}=\frac{1}{4} \qquad \frac{26}{65}=\frac{2}{5} \qquad \frac{19}{95}=\frac{1}{5} \qquad \frac{14385}{170856}=\frac{1435}{17056}$$

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

```
00: RCL 1 DIV RCL 2  STO 3  = INT STO 0 X RCL 2  =  INV SUM  1  RCL 1
20: EXC 3  - ( CE DIV RCL 3  )  INT  X  RCL 3  =  INV  X=T 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 Hirmania.")

USER INSTRUCTIONS:

1. *In 6 OP 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 one 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.*

```
000  82 HIR      035  01  1      070  00  0      105  69 OP       140  10  10     175  82 HIR      210  47 CMS
001  11  11      036  03  3      071  00  0      106  04  04      141  43 RCL     176  05  05      211  09  9
002  42 STO      037  07  7      072  00  0      107  03  3       142  00  00     177  43 RCL      212  93  .
003  01  01      038  01  1      073  00  0      108  02  2       143  22 INV     178  06  06      213  02  2
004  82 HIR      039  07  7      074  00  0      109  03  3       144  44 SUM     179  82 HIR      214  00  0
005  12  12      040  03  3      075  00  0      110  03  3       145  59  59     180  06  06      215  00  0
006  42 STO      041  05  5      076  69 OP      111  01  1       146  43 RCL     181  43 RCL      216  08  8
007  02  02      042  69 OP      077  04  04     112  07  7       147  09  09     182  07  07      217  02  2
008  82 HIR      043  01  01     078  69 OP      113  03  3       148  55  ÷      183  82 HIR      218  08  8
009  13  13      044  02  2      079  05  05     114  05  5       149  03  3      184  07  07      219  02  2
010  42 STO      045  03  3      080  43 RCL     115  69 OP       150  22 INV     185  43 RCL      220  07  7
011  03  03      046  02  2      081  09  09     116  02  02      151  28 LOG     186  08  08      221  06  6
012  82 HIR      047  04  4      082  69 OP      117  01  1       152  95  =      187  82 HIR      222  42 STO
013  14  14      048  03  3      083  40  40     118  03  3       153  42 STO     188  08  08      223  59  59
014  42 STO      049  05  5      084  99 PRT     119  03  3       154  00  00     189  05  5       224  81 RST
015  04  04      050  00  0      085  24 CE      120  01  1       155  44 SUM     190  69 OP
016  82 HIR      051  00  0      086  91 R/S     121  01  1       156  59  59     191  17  17
017  15  15      052  69 OP      087  99 PRT     122  06  6       157  43 RCL     192  43 RCL
018  42 STO      053  02  02     088  59 INT     123  00  0       158  01  01     193  10  10
019  05  05      054  01  1      089  29 CP      124  00  0       159  82 HIR     194  71 SBR
020  82 HIR      055  05  5      090  22 INV     125  00  0       160  01  01     195  82 HIR
021  16  16      056  03  3      091  77 GE      126  00  0       161  43 RCL     196  99 PRT
022  42 STO      057  02  2      092  00  00     127  69 OP       162  02  02     197  06  6
023  06  06      058  01  1      093  32  32     128  03  03      163  82 HIR     198  69 OP
024  82 HIR      059  06  6      094  32 X:T     129  69 OP       164  02  02     199  17  17
025  17  17      060  01  1      095  01  1      130  05  05      165  43 RCL     200  25 CLR
026  42 STO      061  07  7      096  00  0      131  43 RCL      166  03  03     201  91 R/S
027  07  07      062  00  0      097  00  0      132  10  10      167  82 HIR     202  81 RST
028  82 HIR      063  00  0      098  32 X:T     133  69 OP       168  03  03     203  76 LBL
029  18  18      064  69 OP      099  77 GE      134  40  40      169  43 RCL     204  11  A
030  42 STO      065  03  03     100  00  00     135  99 PRT      170  04  04     205  22 INV
031  08  08      066  04  4      101  32  32     136  24 CE       171  82 HIR     206  58 FIX
032  01  1       067  04  4      102  42 STO     137  91 R/S      172  04  04     207  06  6
033  07  7       068  04  4      103  09  09     138  99 PRT      173  43 RCL     208  69 OP
034  03  3       069  05  5      104  25 CLR     139  42 STO      174  05  05     209  17  17
```

THIS IS THE RESULT OF PRESSING KEY A AND MAKING REG 59 PROGRAM-SIGNIFICANT

```
480  00  0
481  00  0
482  00  0
483  76 LBL
484  82 HIR
485  82 HIR
486  00  00
487  92 RTN
```

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 Wortington, 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 in 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.

```
000 76 LBL
001 15 E
002 32 X:T
003 52 EE
004 05 5
005 09 9
006 66 PAU
007 66 PAU
008 66 PAU
009 05 5
010 08 8
011 66 PAU
012 66 PAU
013 05 5
014 07 7
015 66 PAU
016 66 PAU
017 05 5
018 06 6
019 66 PAU
020 66 PAU
021 68 NOP
022 05 5
023 05 5
024 66 PAU
025 66 PAU
026 04 4
027 66 PAU
028 66 PAU
029 05 5
030 03 3
031 66 PAU
032 66 PAU
033 05 5
034 02 2
035 66 PAU
036 66 PAU
037 05 5
038 01 1
039 66 PAU
040 66 PAU
041 05 5
042 00 0
043 66 PAU
044 66 PAU
045 66 PAU
046 04 4
047 09 9
048 66 PAU
049 66 PAU
050 04 4
051 08 8
052 66 PAU
053 66 PAU
054 04 4
055 07 7
056 66 PAU
057 66 PAU
058 04 4
059 06 6
060 66 PAU
061 66 PAU
062 04 4
063 05 5
064 66 PAU
065 66 PAU
066 68 NOP
067 04 4
068 04 4
069 66 PAU
070 66 PAU
071 03 3
072 66 PAU
073 66 PAU
074 04 4
075 02 2
076 66 PAU
077 66 PAU
078 04 4
079 01 1
080 66 PAU
081 66 PAU
082 04 4
083 00 0
084 66 PAU
085 66 PAU
086 66 PAU
087 03 3
088 09 9
089 66 PAU
090 66 PAU
091 03 3
092 08 8
093 66 PAU
094 66 PAU
095 03 3
096 07 7
097 66 PAU
098 66 PAU
099 03 3
100 06 6
101 66 PAU
102 66 PAU
103 03 3
104 05 5
105 66 PAU
106 66 PAU
107 68 NOP
108 03 3
109 04 4
110 66 PAU
111 66 PAU
112 03 3
113 03 3
114 66 PAU
115 66 PAU
116 02 2
117 66 PAU
118 66 PAU
119 03 3
120 01 1
121 66 PAU
122 66 PAU
123 03 3
124 00 0
125 66 PAU
126 66 PAU
127 66 PAU
128 02 2
129 09 9
130 66 PAU
131 66 PAU
132 02 2
133 08 8
134 66 PAU
135 66 PAU
136 02 2
137 07 7
138 66 PAU
139 66 PAU
140 02 2
141 06 6
142 66 PAU
143 66 PAU
144 02 2
145 05 5
146 66 PAU
147 66 PAU
148 68 NOP
149 02 2
150 04 4
151 66 PAU
152 66 PAU
153 02 2
154 03 3
155 66 PAU
156 66 PAU
157 02 2
158 02 2
159 66 PAU
160 66 PAU
161 01 1
162 66 PAU
163 66 PAU
164 02 2
165 00 0
166 66 PAU
167 66 PAU
168 66 PAU
169 01 1
170 09 9
171 66 PAU
172 66 PAU
173 01 1
174 08 8
175 66 PAU
176 66 PAU
177 01 1
178 07 7
179 66 PAU
180 66 PAU
181 01 1
182 06 6
183 66 PAU
184 66 PAU
185 01 1
186 05 5
187 66 PAU
188 66 PAU
189 68 NOP
190 01 1
191 04 4
192 66 PAU
193 66 PAU
194 01 1
195 03 3
196 66 PAU
197 66 PAU
198 01 1
199 02 2
200 66 PAU
201 66 PAU
202 01 1
203 01 1
204 66 PAU
205 66 PAU
206 00 0
207 66 PAU
208 66 PAU
209 66 PAU
210 00 0
211 09 9
212 66 PAU
213 66 PAU
214 00 0
215 08 8
216 66 PAU
217 66 PAU
218 00 0
219 07 7
220 66 PAU
221 66 PAU
222 00 0
223 06 6
224 66 PAU
225 66 PAU
226 00 0
227 05 5
228 66 PAU
229 66 PAU
230 00 0
231 04 4
232 66 PAU
233 66 PAU
234 00 0
235 03 3
236 66 PAU
237 66 PAU
238 00 0
239 02 2
240 66 PAU
241 66 PAU
242 00 0
243 01 1
244 66 PAU
245 66 PAU
246 66 PAU
247 00 0
248 00 0
249 22 INV
250 52 EE
251 68 NOP
252 68 NOP
253 68 NOP
254 68 NOP
255 68 NOP
256 85 +
257 43 RCL
258 00 00
259 95 =
260 61 GTO
261 00 00
262 03 03
263 76 LBL
264 16 A'
265 65 ×
266 93 .
267 00 0
268 01 1
269 95 =
270 92 RTN
271 76 LBL
272 17 B'
273 42 STO
274 02 02
275 01 1
276 00 0
277 77 GE
278 04 04
279 09 09
280 01 1
281 06 6
282 77 GE
283 04 04
284 05 05
285 02 2
286 02 2
287 77 GE
288 04 04
289 01 01
290 03 3
291 00 0
292 77 GE
293 03 03
294 97 97
295 04 4
296 00 0
297 77 GE
298 03 03
299 93 93
300 05 5
301 05 5
302 77 GE
303 03 03
304 89 89
305 07 7
306 00 0
307 77 GE
308 03 03
309 85 85
310 01 1
311 02 2
312 04 4
313 77 GE
314 03 03
315 81 81
316 01 1
317 09 9
318 06 6
319 77 GE
320 03 03
321 77 77
322 02 2
323 09 9
324 02 2
325 77 GE
326 03 03
327 73 73
328 04 4
329 03 3
330 00 0
331 77 GE
332 03 03
333 69 69
334 09 9
335 02 2
336 05 5
337 77 GE
338 03 03
339 65 65
340 01 1
341 09 9
342 01 1
343 00 0
344 77 GE
345 03 03
346 61 61
347 03 3
348 00 0
349 00 0
350 00 0
351 77 GE
352 03 03
353 57 57
354 43 RCL
355 99 99
356 91 R/S
357 02 2
358 00 0
359 02 2
360 85 +
361 02 2
362 00 0
363 02 2
364 85 +
365 02 2
366 00 0
367 00 0
368 85 +
369 02 2
370 00 0
371 02 2
372 85 +
373 02 2
374 00 0
375 02 2
376 85 +
377 02 2
378 00 0
379 01 1
380 85 +
381 01 1
382 00 0
383 02 2
384 85 +
385 02 2
386 00 0
387 02 2
388 85 +
389 04 4
390 00 0
391 04 4
392 85 +
393 04 4
394 00 0
395 03 3
396 85 +
397 04 4
398 00 0
399 04 4
400 85 +
401 04 4
402 00 0
403 02 2
404 85 +
405 04 4
406 00 0
407 03 3
408 85 +
409 01 1
410 09 9
411 00 0
412 09 9
413 95 =
414 16 A'
415 42 STO
416 01 01
417 22 INV
418 59 INT
419 42 STO
420 00 00
421 43 RCL
422 02 02
423 49 PRD
424 00 00
425 65 ×
426 53 (
427 43 RCL
428 01 01
429 59 INT
430 16 A'
431 32 X:T
432 58 FIX
433 02 02
434 25 CLR
435 91 R/S
436 76 LBL
437 14 D
438 16 A'
439 32 X:T
440 01 1
441 95 =
442 91 R/S
443 16 A'
444 42 STO
445 00 00
446 61 GTO
447 04 04
448 32 32
449 76 LBL
450 13 C
451 32 X:T
452 93 .
453 04 4
454 17 B'
455 76 LBL
456 12 B
457 32 X:T
458 93 .
459 06 6
460 05 5
461 17 B'
462 76 LBL
463 11 A
464 32 X:T
465 01 1
466 17 B'
                END
```

---

tanh anyone? In v2n11p1 of 52-Notes, Fred Fish found that PGM 05 of the ML-module could be used to compute sinh and cosh. His routines were:
STO 02 PGM 05 C' *for sinh and* STO 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.

000: LBL A PGM 15 SBR 110 RTN LBL B PGM 05 SBR 006 RTN LBL C STO 00 A DIV

022: ((RCL 00 B) RTN

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:*
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 Blayney's in v5n3p9. Richard Snow provides us with a less elaborate method you can use from the key board. 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:

v5n1p12: Oops, don't do DSZ on register 40. The program interpretes 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.

NESTED OPs	HIRs used.
OP 01	HIR 5
OP 02	HIR 6
OP 03	HIR 7
OP 04	HIR 8
OP 12	HIR 8 and first 3 available HIRs.
OP 13	First 4 available HIRs.
OP 14	HIR 8 and first 3 available HIRs.
OP 15	HIR 8 and first 3 available HIRs.
P/R	First available HIR,r in HIR 7, θ in HIR 8.
INV P/R	First 2 available HIRs, x in HIR 7, y in HIR 8.
INV Σ+.........	x-1 in HIR 7, -x.y in HIR 8
Σ+	x+1 in HIR 7, x.y in HIR 8.
D.MS	First 2 available HIRs, 100 X frac(x) in HIR 8
INV D.MS	First 2 available HIRs, 100 X (Int (x) + .6 frac (x)) in HIR 8.
OP 11	First 2 available HIRs.
INV $\bar{x}$	First available HIR.

****

☆ TRAVERSE ☆

Frank C Blachly

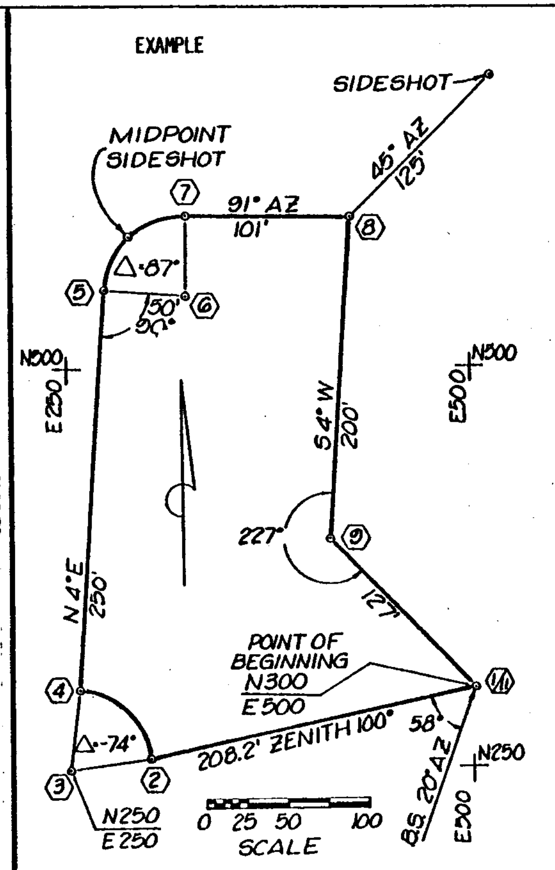
TRAVERSE WILL DO ALL OF THE BASIC SURVEYING ROUTINES: AZIMUTH/BEARING TRAVERSE, HORIZONTAL OR SLOPE DISTANCES, FIELD ANGLE TRAVERSE, INVERSING (COMPUTING BEARING AND DISTANCE BETWEEN KNOWN PAIRS OF COORDINATES) CURVE COMPUTATIONS WITH CURVE DATA, SIDESHOTS, AND FIND ERROR OF CLOSURE AND AREA.

WHEN TRAVERSING A CLOSED LOOP CLOCKWISE, THE SECTOR AREA WILL BE ADDED IF THE CENTRAL ANGLE, Δ, IS POSITIVE OR SUBTRACTED IF Δ IS NEGATIVE. WHEN TRAVERSING COUNTER CLOCKWISE, THE OPPOSITE IS TRUE AND THE AREA WILL BE NEGATIVE.

NOTE: DO NOT PRESS CMS, AS THIS WILL CLEAR PRESTORED CONSTANTS (MEMORIES 21, 22 & 23) AND PRINT CODES. (MEMORIES 24 THROUGH 36)
DO NOT PRESS RST, AS THIS WILL CLEAR ALL FLAGS (FLAG 7 USED FOR PRINTER SENSING)
IF AN ERROR IS MADE DURING STEPS 1, 2 OR 4, REPEAT THAT STEP.
IF AN ERROR IS MADE DURING STEPS 3, 6 OR 7, FOLLOW ALL THE WAY THROUGH, COMPUTING THE COORDINATES AS A SIDESHOT. (STEP 5B OR LBL B')

USER INSTRUCTIONS:	ENTER	PRESS
1. ENTER BEGINNING COORDINATES	NORTHING EASTING	2ND E' R/S
DO STEP 2A, OR 2B OR 6.		
2A. ENTER QUADRANT AND BEARING AS Q.DDMMSS	Q.DDMMSS	A
2B. ENTER AZIMUTH AS DD.MMSSss	DD.MMSS	2ND A'
IF STEP 2 IS THE INITIAL BACKSIDE FOR A FIELD ANGLE, DIRECT TOWARDS BEGINNING COORDINATES		
DO STEP 3 IF NEEDED.		
3. ENTER FIELD ANGLE (ANGLE RIGHT IS POSITIVE, LEFT IS NEGATIVE)	DD.MMSS	B
4. ENTER DISTANCE	DISTANCE	D
IF DISTANCE ENTERED WAS A SLOPE DISTANCE, ENTER ZENITH ANGLE	DD.MMSS	R/S
DO STEP 5A OR 5B.		
5A. COMPUTE COORDINATES AND OCCUPY		E
** IF NO PC100, PRESS R/S FOR EASTING **		
5B. COMPUTE COORDINATES AS A SIDESHOT		2ND B'
** IF NO PC100, PRESS R/S FOR EASTING **		
6. INVERSE TO GIVEN COORDINATES ENTER NORTHING ENTER EASTING	NORTHING EASTING	C R/S
	BEARING IS DISPLAYED	
** IF NO PC100, PRESS R/S FOR QUADRANT ** ** IF NO PC100, PRESS R/S FOR DISTANCE **		
GO TO STEP 5A OR 5B		
7. CURVE COMPUTATIONS IF AT CENTER OF CURVE, ENTER CENTRAL ANGLE	± Δ	2ND D'
NOTE: FOR CENTRAL ANGLES NOT IN THE RANGE -180 ≤Δ≤ 180 BREAK INTO TWO SECTIONS	RADIUS IS DISPLAYED	
** IF NO PC100, PRESS R/S FOR ARC LENGTH ** ** IF NO PC100, PRESS R/S FOR TANGENT ** ** IF NO PC100, PRESS R/S FOR CHORD BEARING ** ** IF NO PC100, PRESS R/S FOR QUADRANT ** ** IF NO PC100, PRESS R/S FOR CHORD DISTANCE **		
GO TO STEP 5A OR 5B		
8. ERROR OF CLOSURE AND AREA OF CLOSED LOOP:		
ENTER BEGINNING COORDINATES	NORTHING EASTING	2ND C' R/S
	ERROR BEARING IS DISPLAYED	
** IF NO PC100, PRESS R/S FOR QUADRANT ** ** IF NO PC100, PRESS R/S FOR ERROR DISTANCE ** ** IF NO PC100, PRESS R/S FOR NORTHING ** ** IF NO PC100, PRESS R/S FOR EASTING ** ** IF NO PC100, PRESS R/S FOR SQUARE FEET ** ** IF NO PC100, PRESS R/S FOR ACRES **		

EXAMPLE



DATA REGISTERS

00	NORTHING
01	EASTING
02	QUADRANT
03	BEARING (DD.MMSS)
04	AZIMUTH (DD.dddd)
05	DISTANCE
06	NORTHING
07	EASTING
08	AZIMUTH (DD.dddd)
09	N
10	DOUBLE AREA }
11	D.M.D. } AREA BY D.M.D.
12	DEPARTURE }
13	CENTRAL ANGLE (DD.dddd)
14	ARC LENGTH
15	SECTOR AREA
16	RC* QUADRANT PRINT CODE
17	USED

FLAGS: 7 PC100

```
[300][2ND][E']

            1.
      300.000   N
[500][R/S]
      500.000   E
[20][2ND][A']

      20.0000   AZ
[58][B]

      58.0000   ANGL
[208.2][D]
      208.200   D
[100][R/S]
     100.0000   ZA
      205.037   D
```

```
[E]

            2.
      257.370   N
      299.444   E
[250][C]

[250][R/S]
      81.3117   SW
       49.990   D
[E]

            3.
      250.000   N
      250.000   E
[74][+/-][2ND][D']

   CURVE DATA

     -74.0000   Δ
       49.990   R
       64.564   A
       37.670   T

      45.2843   NW
       60.169   CD
[E]

            4.
      299.560   N
      256.544   E
[1.04][A]

       4.0000   NE
[250][D]
      250.000   D
[E]

            5.
      548.951   N
      273.983   E
[90][+/-][B]

     -90.0000   ANGL
[50][D]
       50.000   D
[E]

            6.
      545.463   N
      323.861   E
[43.3][2ND][D']

   CURVE DATA

      43.3000   Δ
       50.000   R
       37.961   A
       19.943   T

      25.4500   NE
       37.056   CD
[2ND][B']

        SIDE
      582.327   N
      290.082   E
```

```
[87][2ND][D']

   CURVE DATA

      87.0000   Δ
       50.000   R
       75.922   A
       47.448   T

      47.3000   NE
       68.835   CD
[E]

            7.
      595.455   N
      324.734   E
[91][2ND][A']

      91.0000   AZ
[101][D]
      101.000   D
[E]

            8.
      593.693   N
      425.718   E
[45][2ND][A']

      45.0000   AZ
[125][D]
      125.000   D
[2ND][B']

        SIDE
      682.081   N
      514.107   E
[3.04][A]

       4.0000   SW
[200][D]
      200.000   D
[E]

            9.
      394.180   N
      411.767   E
[227][+/-][B]

    -227.0000   ANGL
[127][D]
      127.000   D
[E]

           10.
      301.298   N
      498.381   E
[300][2ND][C']

[500][R/S]
      51.1715   SE
        2.075   D

           11.
      300.000   N
      500.000   E

     52594.85   SF
      1.20741   AC
```

OFF/ON
REPARTITION
3 OP 17
LRN

```
000 76 LBL
001 15 E
002 61 GTO
003 02 02
004 47 47
005 76 LBL
006 17 B'
007 61 GTO
008 03 03
009 06 06
010 76 LBL
011 13 C
012 61 GTO
013 03 03
014 42 42
015 76 LBL
016 19 D'
017 61 GTO
018 04 04
019 46 46
020 76 LBL
021 18 C'
022 61 GTO
023 06 06
024 05 05
025 76 LBL
026 99 PRT
027 55 ÷
028 01 1
029 02 2
030 22 INV
031 28 LOG
032 85 +
033 01 1
034 54 )
035 82 HIR
036 08 08
037 32 X:T
038 69 OP
039 06 06
040 92 RTN
041 76 LBL
042 87 IFF
043 87 IFF
044 07 07
045 00 00
046 48 48
047 91 R/S
048 92 RTN
049 76 LBL
050 10 E'
051 32 X:T
052 02 2
053 22 INV
054 58 FIX
055 69 OP
056 17 17
057 47 CMS
058 04 4
059 69 OP
060 17 17
061 02 2
062 00 0
063 69 OP
064 07 07
065 69 OP
066 19 19
067 25 CLR
068 32 X:T
069 42 STO
070 00 00
071 42 STO
072 06 06
073 69 OP
074 29 29
075 43 RCL
076 09 09
077 98 ADV
078 22 INV
079 58 FIX
080 99 PRT
081 43 RCL
082 06 06
083 32 X:T
084 02 2
085 09 9
086 58 FIX
087 03 03
088 71 SBR
089 99 PRT
090 92 RTN
091 42 STO
092 01 01
093 42 STO
094 07 07
095 32 X:T
096 09 9
097 07 7
098 71 SBR
099 99 PRT
100 92 RTN
101 76 LBL
102 11 A
103 98 ADV
104 42 STO
105 02 02
106 22 INV
107 58 FIX
108 59 INT
109 48 EXC
110 02 02
111 88 DMS
112 82 HIR
113 18 18
114 42 STO
115 03 03
116 32 X:T
117 43 RCL
118 02 02
119 85 +
120 02 2
121 09 9
122 95 =
123 42 STO
124 16 16
125 73 RC*
126 16 16
127 58 FIX
128 04 04
129 71 SBR
130 99 PRT
131 09 9
132 00 0
133 65 ×
134 43 RCL
135 02 02
136 75 -
137 53 (
138 24 CE
139 85 +
140 43 RCL
141 03 03
142 88 DMS
143 54 )
144 38 SIN
145 22 INV
146 38 SIN
147 50 I×I
148 95 =
149 61 GTO
150 01 01
151 65 65
152 76 LBL
153 16 A'
154 98 ADV
155 32 X:T
156 43 RCL
157 34 34
158 58 FIX
159 04 04
160 71 SBR
161 99 PRT
162 22 INV
163 58 FIX
164 88 DMS
165 42 STO
166 08 08
167 22 INV
168 88 DMS
169 58 FIX
170 04 04
171 92 RTN
172 76 LBL
173 12 B
174 98 ADV
175 32 X:T
176 43 RCL
177 28 28
178 58 FIX
179 04 04
180 71 SBR
181 99 PRT
182 22 INV
183 58 FIX
184 88 DMS
185 75 -
186 43 RCL
187 22 22
188 85 +
189 43 RCL
190 08 08
191 95 =
192 71 SBR
193 04 04
194 30 30
195 48 EXC
196 08 08
197 42 STO
198 04 04
199 43 RCL
200 08 08
201 22 INV
202 88 DMS
203 58 FIX
204 04 04
205 92 RTN
206 76 LBL
207 14 D
208 42 STO
209 05 05
210 32 X:T
211 01 1
212 06 6
213 58 FIX
214 03 03
215 71 SBR
216 99 PRT
217 92 RTN
218 32 X:T
219 43 RCL
220 29 29
221 58 FIX
222 04 04
223 71 SBR
224 99 PRT
225 22 INV
226 58 FIX
227 88 DMS
228 38 SIN
229 49 PRD
230 05 05
231 43 RCL
232 05 05
233 61 GTO
234 14 D
235 43 RCL
236 05 05
237 32 X:T
238 43 RCL
239 08 08
240 37 P/R
241 44 SUM
242 07 07
243 32 X:T
244 44 SUM
245 06 06
246 92 RTN
247 71 SBR
248 02 02
249 35 35
250 53 (
251 53 (
252 43 RCL
253 11 11
254 85 +
255 53 (
256 53 (
257 43 RCL
258 07 07
259 75 -
260 43 RCL
261 01 01
262 54 )
263 85 +
264 48 EXC
265 12 12
266 54 )
267 54 )
268 42 STO
269 11 11
270 65 ×
271 53 (
272 43 RCL
273 06 06
274 75 -
275 43 RCL
276 00 00
277 54 )
278 54 )
279 22 INV
280 44 SUM
281 10 10
282 25 CLR
283 48 EXC
284 15 15
285 44 SUM
286 10 10
287 43 RCL
288 08 08
289 42 STO
290 04 04
291 71 SBR
292 00 00
293 73 73
294 42 STO
295 00 00
296 71 SBR
297 87 IFF
298 43 RCL
299 07 07
300 42 STO
301 01 01
302 71 SBR
303 00 00
304 95 95
305 92 RTN
306 71 SBR
307 02 02
308 35 35
309 43 RCL
310 24 24
311 98 ADV
312 69 OP
313 00 00
314 69 OP
315 03 03
316 69 OP
317 05 05
318 43 RCL
319 04 04
320 42 STO
321 08 08
322 43 RCL
323 00 00
324 48 EXC
325 06 06
326 71 SBR
327 00 00
328 83 83
329 71 SBR
330 87 IFF
331 25 CLR
332 42 STO
333 15 15
334 43 RCL
335 01 01
336 48 EXC
337 07 07
338 71 SBR
339 00 00
340 95 95
341 92 RTN
342 98 ADV
343 42 STO
344 06 06
345 91 R/S
346 42 STO
347 07 07
348 43 RCL
349 06 06
350 75 -
351 43 RCL
352 00 00
353 42 STO
354 06 06
355 95 =
356 32 X:T
357 43 RCL
358 07 07
359 75 -
360 43 RCL
361 01 01
362 42 STO
363 07 07
364 95 =
365 22 INV
366 37 P/R
367 32 X:T
368 42 STO
369 05 05
370 32 X:T
371 71 SBR
372 03 03
373 78 78
374 43 RCL
375 05 05
376 14 D
377 92 RTN
378 71 SBR
379 04 04
380 30 30
381 42 STO
382 08 08
383 55 ÷
384 09 9
385 00 0
386 85 +
387 01 1
388 95 =
389 59 INT
390 42 STO
391 02 02
392 85 +
393 02 2
394 09 9
395 95 =
396 42 STO
397 16 16
398 43 RCL
399 08 08
400 22 INV
401 58 FIX
402 52 EE
403 22 INV
404 52 EE
405 38 SIN
406 22 INV
407 38 SIN
408 50 I×I
409 22 INV
410 88 DMS
411 32 X:T
412 73 RC*
413 16 16
414 58 FIX
415 04 04
416 71 SBR
417 99 PRT
418 71 SBR
419 87 IFF
420 87 IFF
421 07 07
422 04 04
423 29 29
424 43 RCL
425 02 02
426 22 INV
427 58 FIX
428 91 R/S
429 92 RTN
430 32 X:T
431 43 RCL
432 21 21
433 32 X:T
434 37 P/R
435 22 INV
436 37 P/R
437 29 CP
438 77 GE
439 04 04
440 45 45
441 85 +
442 43 RCL
443 21 21
444 95 =
445 92 RTN
446 32 X:T
447 22 INV
448 58 FIX
449 69 OP
450 00 00
451 43 RCL
452 25 25
453 69 OP
454 02 02
455 43 RCL
456 26 26
457 69 OP
458 03 03
459 98 ADV
460 69 OP
461 05 05
462 98 ADV
463 07 7
464 05 5
465 58 FIX
466 04 04
467 71 SBR
468 99 PRT
469 22 INV
470 58 FIX
471 88 DMS
472 42 STO
473 13 13
474 43 RCL
475 05 05
476 32 X:T
477 03 3
478 05 5
479 58 FIX
480 03 03
481 71 SBR
482 99 PRT
483 71 SBR
484 87 IFF
485 43 RCL
486 05 05
487 65 ×
488 89 π
489 65 ×
490 43 RCL
491 13 13
492 50 I×I
493 55 ÷
494 43 RCL
495 22 22
496 95 =
497 42 STO
498 14 14
499 32 X:T
500 09 9
501 03 3
502 71 SBR
503 99 PRT
504 71 SBR
505 87 IFF
506 43 RCL
507 14 14
508 65 ×
509 43 RCL
510 05 05
511 65 ×
512 43 RCL
513 13 13
514 69 OP
515 10 10
516 95 =
517 42 STO
518 15 15
519 43 RCL
520 13 13
521 50 I×I
522 55 ÷
523 02 2
524 95 =
525 30 TAN
526 65 ×
527 43 RCL
528 05 05
529 95 =
530 32 X:T
531 03 3
532 07 7
533 71 SBR
534 99 PRT
535 71 SBR
536 87 IFF
537 43 RCL
538 22 22
539 85 +
540 09 9
541 00 0
542 65 ×
543 43 RCL
544 13 13
545 69 OP
546 10 10
547 85 +
548 43 RCL
549 08 08
550 42 STO
551 17 17
552 85 +
553 43 RCL
554 13 13
555 55 ÷
556 02 2
557 95 =
558 98 ADV
559 71 SBR
560 03 03
561 78 78
562 43 RCL
563 17 17
564 75 -
565 43 RCL
566 22 22
567 85 +
568 43 RCL
569 13 13
570 95 =
571 71 SBR
572 04 04
573 30 30
574 42 STO
575 08 08
576 43 RCL
577 13 13
578 50 I×I
579 55 ÷
580 02 2
581 95 =
582 38 SIN
583 65 ×
584 02 2
585 65 ×
586 43 RCL
587 05 05
588 95 =
589 32 X:T
590 43 RCL
591 27 27
592 58 FIX
593 03 03
594 71 SBR
595 99 PRT
596 71 SBR
597 87 IFF
598 43 RCL
599 13 13
600 22 INV
601 88 DMS
602 58 FIX
603 04 04
604 92 RTN
605 13 C
606 71 SBR
607 87 IFF
608 15 E
609 71 SBR
610 87 IFF
611 98 ADV
612 43 RCL
613 10 10
614 55 ÷
615 02 2
616 95 =
617 32 X:T
618 43 RCL
619 35 35
620 58 FIX
621 02 02
622 71 SBR
623 99 PRT
624 71 SBR
625 87 IFF
626 43 RCL
627 10 10
628 55 ÷
629 43 RCL
630 23 23
631 95 =
632 32 X:T
633 43 RCL
634 36 36
635 58 FIX
636 05 05
637 71 SBR
638 99 PRT
639 92 RTN
```

LABELS

```
001 15 E
006 17 B'
011 13 C
016 19 D'
021 18 C'
026 99 PRT
042 87 IFF
050 10 E'
102 11 A
153 16 A'
173 12 B
207 14 D
```

LRN
REPARTITION
6 OP 17
STORE CONSTANTS IN
MEMORIES 21 THRU 36
THEN
RECORD SIDES 1, 2, 3+4

PRESTORED DATA

360.	21
180.	22
97120.	23

ALPHA REG LIST:

NUMERIC	REG	ALPHA
36249617.	24	SIDE
9541354217.	25	CURVE
16133713.	26	DATA
1516.	27	CD
13312227.	28	ANGL
4693.	29	ZA
3117.	30	NE
3617.	31	SE
3643.	32	SJ
3143.	33	NJ
9346.	34	AZ
3621.	35	SF
1315.	36	AC

WAR GAMES.- My friend and colleague Al Skillicorn, besides being a microwave engineer, is also known as a war games buff. So, recently he announced his intention to merge two hobbys: play war games on his TI-59. War games, for those who are unfamiliar with this term, are complex board games to recreate to the minutest detail famous battles from history. Most strategists from history were accomplished chess players. Now lots of chess players are getting interested in re-playing famous battles. The outcome of this battle-on-a-board is determined by a mixture of skill (strategy) and a judicious amount of simple "dumb luck." But then, weren't most famous battles in history fought on these premises?

War games usually come with "combat results tables", which serve as "arbiters" in confrontations, the outcome of which will be decided upon the roll of a die (the dumb luck part) and the attacker-to-defender-ratio.(the strategy part) The following table is an example, taken from "the battle of Waterloo." In it Ae means "attacker eliminated", Ar = "attacker retreats", De and Dr mean the same thing for the "defender" and Ee finally means "equally eliminated".

COMBAT RATIOS i.e. attacker-to-defender ratios

DIE	1-5	1-4	1-3	1-2	1-1	2-1	3-1	4-1	5-1	6-1
1	Ae	Ar	Ar	Dr	Dr	Dr	De	De	De	De
2	Ae	Ae	Ar	Ar	Dr	Dr	Dr	De	De	De
3	Ae	Ae	Ae	Ar	Dr	Dr	Dr	Dr	De	De
4	Ae	Ae	Ae	Ar	Ar	Dr	Dr	Dr	De	De
5	Ae	Ae	Ae	Ar	Ar	Ee	Dr	Ee	Ee	De
6	Ae	Ae	Ae	Ae	Ar	Ee	Ee	Ee	Ee	De

Attacks executed at "worse" than 1 to 5 are treated as 1 to 5.
Attacks executed at greater than 6 to 1 are treated as 6 to 1.

The interpretation of these tables and having to constantly check if your opponent is not cheating, either inadvertently or by design, is a rather tedious task, which all war games buffs would gladly have the TI-59 do for them.

The attempt to "mechanize" the above table by means of a program is not a trivial exercise, as the same technique, if we come up with an acceptable one, would have application in lots of other, more "serious" fields, such as engineering, physics, mathematics, just to name a few.

Rolling a die is rather simple. I learned a good deal about using the Master Library module, and especially PGM 15, from Fred Fish's "Survival Guide for the TI-58/59 Master Library." Fred tells me that he still has about 50 copies of it available at $ 7.95 post paid, book rate. I highly recommend it for serious programmers who want to get the ultimate out of the ML module. Write Fred at 4902 E.Wiletta, Apt. # 28, Phoenix AZ, 85008 or phone him at 602-275-1489.

The sequence LBL A PGM 15 SBR DMS X 6 + 1 = INT R/S will roll a digit between 1 and 6 included, if we pre-load R09 with a seed. In order to completely eliminate bias with respect to this "seed entering" we could write the initialization routine as 000: LBL E OP 29 RST Press E, go drink a cup of coffee, then press R/S. You will find R09 preloaded with a random seed.

By now storing the random number, 1 through 6, generated by routine A, into R00 we can RCL IND 00 to bring the contents of one the registers 1 through 6 in the display. Subsequently we assign a value 0 through 4 to the table evaluations Ar, Ae, Dr, De and Ee. Then each register 1 through 6 may now contain a 10-digit number representing each a row in the table. Only the digits 0 through 4 may be used. Leading zeros are not seen but have to be taken into account. Thus, a row as the one below could be represented as the digits below it:

Ae	Ar	Ar	Dr	Dr	Dr	De	De	De	Ee
0	1	1	2	2	2	3	3	3	4

The next thing we need is a "digit selector." That means, we should be able to extract a single digit from a randomly selected row. The criterium for this selection should be the combat ratio entered. Once the digit is extracted, each digit should provoke the printing of its corresponding message, like "DEFENDER RETREATS or "EQUALLY ELIMINATED." There are five possible messages, whose alpha code may be stored in 20 registers. Easily done on a TI-59.

If all this seems formidable, let's go on, and see that in reality this is a rather straightforward problem having several solutions. We will present one here, in the hope that it will stimulate you to write one of your own design.

In order to write an efficient "digit extraction" routine, in which we probably could use the sequence DIV x:t INV LOG = with a single digit in the t-reg, our "digit extractor" should work from the entry of one single digit. What we enter, however, is a "combat ratio", ranging from 1-5 to 6-1. A simple scheme for entering this combat ratio would be 1-5 entered as "1.5" and 6-1 entered as "6.1'. If we subsequently in the program multiply the entered value by 10, we end up with numbers ranging from 15, 14, 13......to 51, 61. Storing these numbers in, say,R10 and then RCL IND 10, we would obtain the contents of registers 15, 14, 13....to 51, 61. So, it is obvious that we now have to pre-load registers 15, 14, 13 ....to 51, 61 with the digits 9, 8, 7.....to 1, 0.

Thus RCL IND 10 will result in a single digit 9 through 0 appearing in the display. If this seems involved, let me defend it by pointing out its speed and its economy in program steps, versus a computing routine.

To demonstrate how it works, load a 10-digit number, composed of the digits 0 through 4 only, into R00. Then write into program memory the following routine. Enter a single digit between 9 and 0 and press A. See how it extracts one single digit from R00.

LBL A X:T RCL 00 DIV X:T INV LOG = INT DIV 10 = INV INT X 10 = R/S

We now have solved almost all of our problems. We have chosen a register 1 through 6 by means of a random roll of a die. We have entered the combat ratio and obtained a digit from from 9 through 0. By means of this one we have extracted one single digit from the random selected register. Rests now to make that one single digit, 0 through 4, print its corresponding message.

That we can do by the "multiply and add a bias" routine. It works as follows: Suppose we make each printing routine, there will be five of them, exactly 16 steps long: RCL 27 OP 01 RCL 28 OP 02 RCL 14 OP 03 RCL 15 GTO PRT

Registers 27, 28, 14 and 15 contain, of course, alpha code. And we define PRT as: LBL PRT OP 04 OP 05 OP 00 CLR RTN

And let's also suppose that the "selected out" digit is "0" and that our "digit selection routine " ends on step 052. We now write, continuing on step 053:

X 16 + YY = STO 08 GTO IND 08

The "0" we obtained as the selected digit, multiplied by 16 still gives as result zero. But adding YY to it will furnish us with a GTO address. That address should be the step IMMEDIATELY FOLLOWING THE 08 of GTO IND 08. So, to compute YY we count up and find that YY has to be equal to 65 Thus, a "0" will GTO 065. A "1" will send the program counter to 1 X 16 + 65 = 081. We can compute all the GTO IND addresses this way, obtaining for the last one 4 X 16 + 65 = 129.

These then are the program steps at which each of the five printing routines have to be written. That is all folks and as you can see, there was no need to insert NOPs at strategic places, as I have seen recommended sometimes.

We now load registers 1 through 6 with table coding, registers 15, 14,...61 with the digits 9, 8,..0 and 24 registers of our choice with alpha code.

Writing the program itself now is a cinch. You probably will recognize in it all the elements we have discussed in the above.

To run the program, enter the combat ratio and press A. The printer-arbitor will print one of five possible messages.

Record two mag cards with 6 OP 17. The program automatically partitions to 7 OP 17. This was made necessary because a possible RCL 61, although it contains only a zero. It would cause an error otherwise.

WAR GAMES.- Arbiter program.

```
000  76 LBL    036  42 STO    072  02  02    108  03  03
001  15  E     037  00  00    073  43 RCL    109  43 RCL
002  07  7     038  73 RC*    074  16  16    110  19  19
003  69 OP     039  00  00    075  69 OP     111  61 GTO
004  17  17    040  55  ÷     076  03  03    112  99 PRT
005  69 OP     041  73 RC*    077  43 RCL    113  43 RCL
006  29  29    042  10  10    078  17  17    114  24  24
007  81 RST    043  22 INV    079  61 GTO    115  69 OP
008  76 LBL    044  28 LOG    080  99 PRT    116  01  01
009  99 PRT    045  95  =     081  43 RCL    117  43 RCL
010  69 OP     046  59 INT    082  22  22    118  25  25
011  04  04    047  55  ÷     083  69 OP     119  69 OP
012  69 OP     048  01  1     084  01  01    120  02  02
013  05  05    049  00  0     085  43 RCL    121  43 RCL
014  69 OP     050  95  =     086  23  23    122  16  16
015  00  00    051  22 INV    087  69 OP     123  69 OP
016  25 CLR    052  59 INT    088  02  02    124  03  03
017  92 RTN    053  65  ×     089  43 RCL    125  43 RCL
018  76 LBL    054  01  1     090  18  18    126  17  17
019  11  A     055  06  6     091  69 OP     127  61 GTO
020  65  ×     056  00  0     092  03  03    128  99 PRT
021  01  1     057  85  +     093  43 RCL    129  43 RCL
022  00  0     058  06  6     094  19  19    130  26  26
023  95  =     059  05  5     095  61 GTO    131  69 OP
024  42 STO    060  95  =     096  99 PRT    132  01  01
025  10  10    061  42 STO    097  43 RCL    133  43 RCL
026  36 PGM    062  08  08    098  24  24    134  27  27
027  15  15    063  83 GO*    099  69 OP     135  69 OP
028  71 SBR    064  08  08    100  01  01    136  02  02
029  88 DMS    065  43 RCL    101  43 RCL    137  43 RCL
030  65  ×     066  22  22    102  25  25    138  16  16
031  06  6     067  69 OP     103  69 OP     139  69 OP
032  85  +     068  01  01    104  02  02    140  03  03
033  01  1     069  43 RCL    105  43 RCL    141  43 RCL
034  95  =     070  23  23    106  18  18    142  17  17
035  59 INT    071  69 OP     107  69 OP     143  61 GTO
                                             144  99 PRT
```

```
REGISTERS

  112223333              01
   11222333              02
    1222333              03
    1122333              04
    1142443              05
     114443              06
          5              11
          6              12
          7              13
          8              14
          9              15
 1727242824  ELIMI       16
 2913371716  NATED       17
 3517373517  RETRE       18
 1337360000  ATS         19
          4              21
   13373713   ATTA       22
 1526173500  CKER        23
   16172117   DEFE       24
 3916173500  NDER        25
     173441    EQU       26
 1327274500  ALLY        27
          3              31
          2              41
          1              51
```

```
ATTACKER ELIMINATED
ATTACKER RETREATS
DEFENDER RETREATS
 EQUALLY ELIMINATED
DEFENDER ELIMINATED
ATTACKER RETREATS
```

The two referees agreed that this program was straigthforward and was written in a logical sequence. The first referee, Richard Snow, only suggested to bring the OP 04 steps within the PRT subroutine, which I originally did not have. That saved eight steps without slowing down the execution. But the second referee, his brother Robert, objected to two things: 1) the random number generator in the ML module uses registers 9 and 7; one cannot use CMS without destroying the seed, and 2) why I used nine registers to store just one single digit, when I had already a digit unpacker on board. (steps 043 and following)

So, Robert wrote a separate SBR, A', as the unpacking routine and used only two registers, 1 and 2, to store all nine digits. Next he used a different random number generator that doesn't require the ML module. The only drawback of his program is, that, although he uses less program steps, he requires quite some repetition in his print code registers. (he uses 20 registers loaded with print code versus my program only 10)

His instructions are:

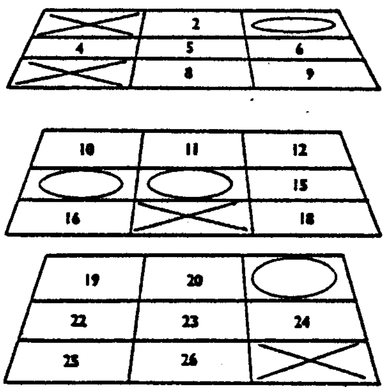
Enter a seed and press E. Enter the combat ratio and press A or R/S.
Robert's program should be recorded on one card, banks 1 and 4.
Mine requires two cards, banks 1, 3 and 4.

Maurice E.T.. Swinnen

```
000  76 LBL    029  00  00    060  73 RC*          12345.      01
001  16 A'     030  38  38    061  00  00           6789.      02
002  55  ÷     031  42 STO    062  16 A'              22.      03
003  32 X:T    032  00  00    063  65  ×       112223333.      04
004  22 INV    033  32 X:T    064  04  4        11222333.      05
005  28 LOG    034  59 INT    065  42 STO        1222333.      06
006  95  =     035  61 GTO    066  00  00        1122333.      07
007  22 INV    036  00  00    067  85  +         1142443.      08
008  59 INT    037  42  42    068  01  1          114443.      09
009  65  ×     038  32 X:T    069  00  0     2913371716 NATED  10
010  01  1     039  71 SBR    070  95  =     1727242824 ELIMI  11
011  00  0     040  00  00    071  42 STO    1526173500 CKEP   12
012  95  =     041  07  07    072  03  03      13373713  ATTA  13
013  59 INT    042  32 X:T    073  73 RC*    1337360000 ATS    14
014  92 RTN    043  73 RC*    074  03  03    3517373517 RETRE  15
015  76 LBL    044  00  00    075  84 OP*    1526173500 CKEP   16
016  15  E     045  16 A'     076  00  00      13373713  ATTA  17
017  38 SIN    046  32 X:T    077  69 OP     1337360000 ATS    18
018  82 HIR    047  06  6     078  23  23    3517373517 RETRE  19
019  04  04    048  82 HIR    079  97 DSZ    3916173500 NDER   20
020  25 CLR    049  44  44    080  00  00      16172117  DEFE  21
021  91 R/S    050  82 HIR    081  00  00    2913371716 NATED  22
022  76 LBL    051  14  14    082  73  73    1727242824 ELIMI  23
023  11  A     052  59 INT    083  69 OP     3916173500 NDER   24
024  42 STO    053  82 HIR    084  05  05      16172117  DEFE  25
025  00  00    054  54  54    085  61 GTO    2913371716 NATED  26
026  32 X:T    055  85  +     086  00  00    1727242824 ELIMI  27
027  02  2     056  04  4     087  20  20    1327274500 ALLY   28
028  77 GE     057  95  =                        173441   EQU  29
               058  42 STO
               059  00  00
```

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 "0"'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.

```
000  92 RTN
001  61 GTO
002  11 A
003  76 LBL
004  19 D'
005  42 STO
006  30 30
007  61 GTO
008  00 00
009  18 18
010  43 RCL
011  31 31
012  44 SUM
013  28 28
014  44 SUM
015  29 29
016  44 SUM
017  30 30
018  73 RC*
019  28 28
020  85 +
021  73 RC*
022  29 29
023  85 +
024  73 RC*
025  30 30
026  95 =
027  42 STO
028  34 34
029  32 X:T
030  87 IFF
031  40 IND
032  34 34
033  03 03
034  24 24
035  02 2
036  67 EQ
037  03 03
038  01 01
039  08 8
040  67 EQ
041  02 02
042  99 99
043  04 4
044  22 INV
045  77 GE
046  03 03
047  24 24
048  67 EQ
049  03 03
050  03 03
051  01 1
052  67 EQ
053  03 03
054  05 05
055  00 0
056  67 EQ
057  03 03
058  07 07
059  03 3      R
060  05 5
061  01 1      A
062  03 3
063  03 3      T
064  07 7
065  03 3

066  06 6      S
067  07 7      !
068  03 3
069  14 D
070  00 0
071  76 LBL
072  15 E
073  47 CMS
074  81 RST
075  76 LBL
076  11 A
077  42 STO
078  32 32
079  25 CLR
080  29 CP
081  73 RC*
082  32 32
083  67 EQ
084  00 00
085  89 89
086  00 0
087  35 1/X
088  81 RST
089  01 1
090  04 4
091  42 STO
092  28 28
093  02 2
094  07 7
095  42 STO
096  29 29
097  01 1
098  42 STO
099  31 31
100  72 ST*
101  32 32
102  42 STO
103  00 00
104  19 D'
105  03 3
106  42 STO
107  00 00
108  05 5
109  42 STO
110  28 28
111  01 1
112  42 STO
113  29 29
114  09 9
115  42 STO
116  31 31
117  19 D'
118  03 3
119  42 STO
120  00 00
121  42 STO
122  29 29
123  05 5
124  42 STO
125  28 28
126  07 7
127  19 D'
128  01 1
129  04 4
130  42 STO
131  28 28

132  09 9
133  42 STO
134  29 29
135  01 1
136  09 9
137  19 D'
138  03 3
139  42 STO
140  29 29
141  02 2
142  05 5
143  19 D'
144  02 2
145  01 1
146  42 STO
147  29 29
148  07 7
149  19 D'
150  03 3
151  42 STO
152  00 00
153  42 STO
154  31 31
155  01 1
156  42 STO
157  28 28
158  01 1
159  01 1
160  19 D'
161  03 3
162  42 STO
163  00 00
164  42 STO
165  28 28
166  01 1
167  09 9
168  42 STO
169  29 29
170  01 1
171  01 1
172  19 D'
173  03 3
174  42 STO
175  00 00
176  01 1
177  42 STO
178  28 28
179  42 STO
180  31 31
181  01 1
182  03 3
183  19 D'
184  03 3
185  42 STO
186  00 00
187  07 7
188  42 STO
189  28 28
190  01 1
191  09 9
192  42 STO
193  29 29
194  01 1
195  03 3
196  19 D'
197  09 9

198  42 STO
199  00 00
200  01 1
201  42 STO
202  28 28
203  01 1
204  09 9
205  42 STO
206  29 29
207  01 1
208  00 0
209  19 D'
210  09 9
211  42 STO
212  00 00
213  01 1
214  42 STO
215  28 28
216  03 3
217  42 STO
218  29 29
219  42 STO
220  31 31
221  02 2
222  19 D'
223  03 3
224  42 STO
225  00 00
226  01 1
227  42 STO
228  28 28
229  42 STO
230  31 31
231  07 7
232  42 STO
233  29 29
234  04 4
235  19 D'
236  03 3
237  42 STO
238  00 00
239  01 1
240  00 0
241  42 STO
242  28 28
243  01 1
244  06 6
245  42 STO
246  29 29
247  01 1
248  03 3
249  19 D'
250  03 3
251  42 STO
252  00 00
253  01 1
254  09 9
255  42 STO
256  28 28
257  02 2
258  05 5
259  42 STO
260  29 29
261  02 2
262  02 2
263  19 D'

264  87 IFF
265  00 00
266  02 02
267  77 77
268  01 1      C
269  05 5
270  01 1      A
271  03 3
272  03 3      T
273  07 7
274  14 D
275  00 0
276  15 E
277  04 4
278  72 ST*
279  32 32
280  87 IFF
281  08 08
282  02 02
283  87 87
284  43 RCL
285  32 32
286  81 RST
287  02 2      I
288  04 4
289  00 0
290  00 0
291  04 4      W
292  03 3
293  02 2      I
294  04 4
295  03 3      N
296  01 1
297  14 D
298  15 E
299  86 STF
300  08 08
301  86 STF
302  02 02
303  86 STF
304  04 04
305  86 STF
306  01 01
307  86 STF
308  00 00
309  29 CP
310  73 RC*
311  28 28
312  67 EQ
313  03 03
314  29 29
315  73 RC*
316  29 29
317  67 EQ
318  03 03
319  34 34
320  43 RCL
321  30 30
322  42 STO
323  32 32
324  97 DSZ
325  00 00
326  00 00
327  10 10
328  92 RTN

329  43 RCL
330  28 28
331  61 GTO
332  03 03
333  22 22
334  43 RCL
335  29 29
336  61 GTO
337  03 03
338  22 22
339  06 6
340  02 2
341  00 0
342  00 0
343  00 0
344  00 0
345  85 +
346  69 OP
347  20 20
348  73 RC*
349  00 00
350  77 GE
351  03 03
352  54 54
353  92 RTN
354  05 5
355  00 0
356  92 RTN
357  76 LBL
358  14 D
359  93 .
360  69 OP
361  03 03
362  00 0
363  42 STO
364  00 00
365  03 3
366  42 STO
367  35 35
368  32 X:T
369  01 1
370  42 STO
371  36 36
372  71 SBR
373  04 04
374  08 08
375  71 SBR
376  04 04
377  08 08
378  61 GTO
379  04 04
380  08 08
381  02 2      -
382  00 0
383  04 4      +
384  07 7
385  02 2      -
386  00 0
387  02 2      -
388  00 0
389  00 0
390  00 0
391  84 OP*
392  36 36
393  97 DSZ
394  36 36

395  51 BST
396  02 2      -
397  00 0
398  02 2      -
399  00 0
400  04 4      +
401  07 7
402  02 2      -
403  00 0
404  02 2
405  00 0
406  84 OP*
407  36 36
408  69 OP
409  05 05
410  69 OP
411  00 00
412  71 SBR
413  03 03
414  46 46
415  52 EE
416  08 8
417  22 INV
418  52 EE
419  85 +
420  71 SBR
421  03 03
422  39 39
423  95 =
424  84 OP*
425  36 36
426  01 1
427  44 SUM
428  36 36
429  71 SBR
430  03 03
431  39 39
432  95 =
433  58 FIX
434  02 02
435  84 OP*
436  36 36
437  22 INV
438  58 FIX
439  69 OP
440  05 05
441  97 DSZ
442  35 35
443  03 03
444  81 81
445  03 3
446  42 STO
447  35 35
448  69 OP
449  00 00
450  43 RCL
451  32 32
452  92 RTN
453  61 GTO
454  11 A

LABELS.
004  19 D'
072  15 E
076  11 A
358  14 D
```