* TI PPC NOTES *

****

v5N6 , 1980

NEWSLETTER OF THE TI PERSONAL PROGRAMMABLE CALCULATOR CLUB.

9213 Lanham Severn Road, Lanham, Maryland, 20801.

In v5n4/5p2 we gave you all the names and addresses of other TI PPC clubs. Since then we heard of two more clubs, one in Sweden and another one in Denmark. Of the latter one I don't have enough information as yet, but from Stockholm I received an excellent newsletter, comparable in quality to Display, 64 pages of the most expensive printing you have ever seen. If you read Swedish and want to subscribe, write(in English if you like) to Gunnar Svanborg, Box 39103, Norra Hamnvägen 3, 100 54 Stockholm, Sweden. How much? 120 Svenska Kronor / year. Tell him you want to subscribe to Föreningen Programbiten.

One of our members, Al Parker, of Solar Electronics Co, 901 North Highland Ave in Hollywood, CA, 90038, has developed and compiled a large electronics program library for the TI-59, which he wants to share with interested members. The list is far too long to be published here, so send Al a SASE with enough postage, to obtain that list. Al tells me the programs are free, but please , only genuinely interested requestors. The supply is not inexhaustable.

The Science Accessories Corp manufactures the DC-59 interface. It is a device that gets connected between an SR-52 or a TI-59 and any other electronic instrument so that digital information from the latter enters the calculator directly, bypassing the key board. A suitable program in the calculator can then work on those data. It requires, of course, modification of your calculator, which SAC will do. A practical example would be the connection to an electronic scale. A program could then compute the actual postage needed for letters or packages placed on the scale. A different user-defined key could be assigned to each rate.A flyer is enclosed with this newsletter. Please don't write to me but write directly to SAC, mentioning TI PPC NOTES, of course. If enough interest, they tell me, they might even consider offering the device in kit form to our members, at a lower price, now about $ 250.00.

Many service stations are advertising the price of gasoline in liters. I have seen otherwise good programmers struggle with longwinded conversions on their machines. I thought I had a short one, until Richard Snow sent me this one: With the ML-module attached, enter price per liter and press PGM 25 C and, voilà, the price per gallon will be displayed!

PGM 02 SBR 239 is the highlight of this issue.(see Quirks, p4) It not only puts the calculator in a transisional state where all the key codes are changed, but it also sometimes provokes a FAST mode, whereby execution speed is doubled. Besides those quirks it sometimes exibits fractured digits, non-resettable TRACE mode and even a FIX 10 state. It constitutes an unending source of experimentation.

The various LABELS-TO-ADDRESSES CONVERSION programs have just arrived from final review. There are three good ones, which I plan to publish in the next issue. This way the members can decide for themselves which one is best suited for their particular needs.

In Scientific American and in articles by Isaac Asimov there is often a reference to the "Journal of Recreational Mathematics." Dirk Dykstra provided me with the address: Baywood Publishing Co Inc. 120 Marine Street, Box D, Farmingdale, NY, 11735, USA. The subscription rate for an individual is $ 12.50 a year. An institution pays double that rate. There are four issue per year.

The June 20, 1980, issue of EDN, the well-known magazine for electronics designers, contains a 15-page feature article on hand-held personal calculators, by Jim McDermott. This very well written and researched article is a must reading.

TABLE OF CONTENTS.

REHASH.- According to Elmer Clausen, the print routine in v5n1 does not allow pending operations. Although this might seem trivial, occasions might arise where it would be convenient to operate through pending operations. Elmer offers this one as a substitute for those occasions:

```
LBL PRT  P/R  12 INV LOG  HIR 38  HIR 17  RTN
```

Similar to the SNOW routine on v5n1p2, this routine also requires you to add "9" to all the print codes, because HIR 38 means "add to HIR 8" and 12 INV LOG is 9 short of 10^{12}.

On the same page, the flag routine in the printer-sensor should be rearranged according to Elmer, as successful operation without the printer does not require that flag 1 be reset before calling the sequence. And flag 1 may have been left set by a previous program.

```
... CP  EQ  B  INV  LBL  B  STF  1  ...
```

BIOMEDICINE.- In v5n1p10 I said I thought I had seen it all. Well... Dr. Barry Tepperman, Toronto, Canada, sent me another one of that type of programs. The article in question is almost a book! It contains 43 pages. The complete reference is: Fabiato, Alexandre and Francoise Fabiato., Calculator Programs for Computing the Composition of the Solutions Containing Multiple Metals and Ligands Used for Experiments in Skinned Muscle Cells, J. Physiol., Paris, 1979, 463-505.
Address for reprints: Dr. Alexandre Fabiato, Professor of Physiology, Box 551, Medical College of Virginia, Richmond, VA, 23298, USA.

Barry himself is an accomplished bilingual (RPN-AOS) and his latest one is a jewel! It is called "RPN-59". It is an RPN simulation system for the TI-59 operated from the keyboard, using the subroutines of the RPN simulator module to emulate the stack and the arithmetic operations of a four-level calculator, in the manner of the keyboard operations of the HP-67. The program has been submitted to PPX and will probably be available in the next addendum. I highly recommend it.

BIOCHEMISTRY.- A new book on the use of calculators in biochemistry has appeared: POCKET PROGRAMMABLE CALCULATORS IN BIOCHEMISTRY, by John E. Barnes and Alan J. Waring, Wiley-Interscience, 377 pages, 6" by 9" softbound, $ 15.00 US. (Hardbound $ 25.00 US) It contains 60 programs for the HP-67/97 and for the TI-59. In an appendix it provides ways how to use also the HP-41C. The chapters are:

1. *Aqueous solutions of small molecules.*
2. *Macromolecules in solution.*
3. *Sedimentation*
4. *Ligand binding and kinetics.*
5. *Thermodynamics in biochemistry.*
6. *Spectroscopy.*
7. *Isotopes in biochemistry.*
   It further has appendices: 1) Least square fit to a straight line, 2) Physical constants and conversion factors, 3) Selected thermodynamic dissociation constants, 4) other sources of calculator information and 5) HP-41C compatibility.

A NEW CALCULATOR-COMPUTER.- The Wall Street Journal of Monday May 5, 1980, announces a new device soon to be marketed in this country. It is the first hand-held device that clearly crosses the line between a calculator and a computer. The Sharp PC1210 has been test-marketed in Japan and will go on sale soon in Europe. It can store up to 400 lines of program instructions in Basic. It has 26 memories and a casette I/O port. It will sell for about $ 125.00 US.
A second machine, capable of storing 1000 lines of Basic, the PC1211, will sell for about $ 175.00 US. As a comparison, the Radio Shack TRS-80 computer has a capacity of about 4000 bytes, while the HP41C can store 2200 bytes.
A companion low-cost printer will also be available soon.

CE substitute.- Using CE as a dummy operator works, of course, but it also clears any existing error condition. Philip Rowley, the editor of the British TI newsletter, says that IND works instead of CE in every case, but does not clear error states.

LETTERS TO THE EDITOR.- Several members have asked me to institute a "Letters to the editor" column. The latest one to ask is Joseph Suckiel. I agree, but only on the condition that we will not be bogged down in endless tirades about insignificant and uninteresting points, taking up precious real estate in our newsletter. So, let's agree on a couple of simple rules:

1. *Make it short and come to the point in fifteen lines or less.*
2. *Type it yourself, not more than 6 1/2 inches or 16,5 cm wide. It will get reduced to 50% of original size. This way we can fit two letters side by side.*

TI-59 PROGRAMMING INSTRUCTION.- The Department of Engineering and Applied Sciences of the University of Wisconsin will conduct three seminars between July 7 and July 11, 1980, a one-day, followed by two two-day ones. Prices run from $ 95.00 for the one-day, over $ 185.00 for the two-day to $ 450.00 for the full-week seminar. The first three days will familiarize you with the calculator (TI-59, TI-58, HP-67/97, HP-41C) while the last two days are dedicated to specific Industrial Engineering applications. Courses will be conducted in Milwaukee. If there is still time left when you read this, you may call (414) 224-1833 for enrollment. You may also buy a TI calculator or printer from them at special discount prices.

The TIPPP program will conduct three separate two-day seminars on programming the TI-58/59. Admission will be on first-come-first-serve basis, with a maximum of 50 students per seminar. For the Washington DC area the date is July 17-18 at the Sheraton in Lanham. In the Pittsburgh area it will be on September 11-12, while for the Philadelphia area the seminar will be held on September 15-16. Yours truly will be the instructor. Please DO NOT call me, but call Robert Burns in Lubbock, TX at (806) 741-3242.

SST DIAGNOSTIC.- Palmer O. Hanson reports the first documented success with the SST diagnostic.(v5n2p9) Calculator S/N 2365787 yielded erratic results with PG 02 of the ML module, but passed all the diagnostic tests recommended by TI: Page VII-9 of Personal Programming, the 598-test-1 of page 43 of the TI-58/59 Service Manual. It even passed Bill Skillman's "Magnetic Card Comparator", PPX 908119.
But it bit the dust when subjected to the SST diagnostic! It showed 04 at location 793 instead 41. Voilà!

CALCULATOR CALCULUS.- Maurice D. Weir, Associate Professor at the Naval Postgraduate School in Monterey, California, and a TI PPC Club member, wrote a book by this title and uses it in his teaching of mathematics. He sent me a typewritten copy of this fascinating work. It seems that he first started writing the book for use with the RPN calculators and later adapted it to the TI-59. He is under contract with Prentice -Hall publishers to expand the book for a possible publication in early 1982. Another of his books, which I haven't seen yet, will be published by the same publisher early 1981. It is called CALCULATOR CLOUT: METHODS OF PROGRAMMABLE CALCULATORS. Although the programs in CALCULATOR CALCULUS could be optimized somewhat, they are solid and work perfectly. They are very practical to teach someone in a non-dry manner the rudiments of calculus. I wish I 'd had those programs and a TI-59 at the time I wrestled with those difficult concepts. I took time out and re-taught myself the whole subject the "Maurice Weir" way. By means of these programs you can experiment endlessly, something you would never do if one calculation took you the whole afternoon. That's what happened when I was in college and that is why those experiments were never done up to now. I am happy I made myself go back to school.

BIOSTATISTICS.- Barry Tepperman wrote an article soon to be published in BYTE. It is called SURVIVORSHIP ANALYSIS AND THE TI-59. I deals with the analysis of survivor patterns in desease, essential to the development of new therapies. The TI-59 programs, four in total, are well written. They may be used with or without the printer. Look for them in BYTE in the very near future.

QUIRKS.- Since a few months a particular sequence cropped up in several newsletters, mostly from readers asking what it meant, rather than from readers-researchers explaining what it might mean. The sequence is: in user memory place: PGM 02 SBR 240 or SBR 239. Both SBRs works equally. Then press RST R/S to execute this sequence. Now, depending on, if you started writing the sequence on step 000, 001....through 007, the results will be different.

For example, if you start writing at 000, the program you write after you have executed this sequence, will have its key codes "elevated" by 40. That is, if you press STO (42) you will end up with HIR (82). This is handy to fill up some steps with HIRs and use Richard Snow's key board HIR trick.(v5n4/5p26)

If you want to cancel this special state, press RST or CP, which points towards a module dependency.

The same phenomena repeat themselves in each octet. Thus starting at 000 has the same effect as starting at 007, 015...

Richard Snow has investigated this thing and written a three-page compilation on it. Fanatics might obtain a copy of it by sending me SASE.

Another practical use for this sequence is pressing repeatedly the 1 key, which results in SST (41). If you fill up a card side with it, it can be used a diagnostic card, as explained in v5n2p9. Palmer O. Hanson submitted a program to do just that to the Honeywell PPX club, an internal club for Honeywell employees.

Then Martin Neef of the ZEPRA club (Germany) found another exciting use for the sequence: a FAST mode. Programs placed in user memory execute at about twice normal speed!

Starting at step 005: PGM 02 SBR 239 9 ; execute this sequence, but stay in LRN. Then BST to get to step 000.

Now place in user memory:

```
CMS  CP  200  x:t  OP 20  RCL 00  INV EQ  006  EE INV EE  R/S GTO 000
```

Still in LRN,BST a couple of times to get to the R/S step. Then go out of LRN. Now Press R/S. The DSZing of reg 00 should take about 35 sec.

Now, cancel the special state by pressing RST. Press R/S again and this time execution will take 70 sec!

It seems that, in order to make the program halt, the R/S step is not sufficient. You always need EE INV EE in front of it.

In the FAST mode you can't use user-defined keys, nor subroutine calls. Neither can you call subroutines or programs from the module. But you can start a SBR from the key board by pressing SBR xxx.

Palmer Hanson thinks Fast mode puts the calculator in the same mode as when it is accessing RAM memory. Take, for example, the INV SUM+ function. If you program 24 iterations of the function, it will take 22 sec to complete. If you use the RAM program which mechanizes that same function (loc. 213 to 248 of downloaded RAM) and iterate that one 24 times, it will take 38 sec.

A second clue that it is not the module but RAM is, that RST will NOT interrupt FAST mode program execution.

To return to Don Lauhgery's strange TRACE mode, J. Huntington Lewis asks us to try the following: LRN CLR LRN RST 2020202 PGM 1 SST LRN and the printer is in TRACE mode. Each key pressed will result again in an elevated code: +20, with a two-digit maximum. After you key in a program in user memory,you may BST and see the result!!! It will trace your program in reverse, without mnemonics!!!

But in key board mode, things are even stranger. The calculator stays in TRACE mode, but in FIX 9. (ever seen a FIX 9 ?) Normally, FIX 9 is the same as INV FIX. But if you press FIX 9 in this state, you get, as expected, INV FIX.

Pressing RST or CP will kill this state.

COPYING THE DISPLAY IN THE T-REG.- You sometimes want to store the display value in the t-reg, while keeping the same value in the x-reg. A rather clumsy approach is ...STO 00 X:T RCL 00 ... But v3n3p4 of 52-Notes did it this way, without involving an extra register: ...(CP + X:T) It can be used in a program, but don't try it from the key board. The CP will wipe out your user program. J. Huntington Lewis of Norfolk, VA, sends me this gem:...Y^X X:T 1) ... which is usable in a program and from the key board.

STO.- David Borton, of Walnut Creek, California, reworked a code converter by Karl Gailer, such that a very interesting application resulted. Suppose you want to write a program that stores sums of money in particular data memories. The number of the register could then become your costumer identification number. Each time you store an amount of money, you want confirmation by a suitable print-out. This routine does it in anelegant and efficient way.

```
LBL A  STO 00  DIV  10  +  1.1  +  INT  X  99  +  (  INV INT  X  10  +  OP 10
X  88  -  88  =  OP 04  R/S  STO IND  00  OP 06  R/S
```

Instructions are: Enter register number 01 through 89 (make sure you have the correct partition) and press A. Enter amount and press R/S. See printout of amount and register number on one line.

Then, as an afterthought, he said that it could become an enhancement of my DATA LIST.(PPX # 908031B) I added a few "niceties" to his program, such as automatic partitioning, automatic stop after arriving at register 90 , etc.

```
000:  RTN  LBL  A  INV  STF  1  STO  00  DIV  10  +  1.1  +  INT  X  99  +  (
022:  INV  INT  X  10  +  OP  10  X  88  -  88  =  OP  04  IFF  2  087  IFF  1
045:  060  R/S  STO IND  00  OP  06  RST  LBL  B  STF  1  0  STO  00  RCL IND  00
062:  CP  EQ  073  STF 2  RCL 00  SBR  008  OP 20  90  X:T  RCL  00  EQ  100
083:  GTO  060  RCL IND 00  OP 06  GTO 073  LBL E  9  OP  17  CLR  R/S  ADV  R/S
```

(Last step used is 101)

Instructions are: Initialize, press E
Enter register (=costumer) number 1 through 89, press A.
Enter amount, press R/S. Repeat as many times as needed.
To list all memories that do not contain zero, press B.

To record this program, AND THE DATA if you like, press 6 OP 17 1 Write, 2 write, etc. , four card sides if with data, one single card side if program only. You could make your life easier, in case you have to store data and record them each day, by adding an auxiliary routine:

```
LBL  D  6  OP 17  1  PRT WRITE  2  PRT  WRITE  3  PRT  WRITE  4  PRT  WRITE  R/S
```

The recording of the four card sides now becomes a cinch:
Press D. See 1 printed, asking for card side 1. Slide side one in slot.
See 2 printed, asking for card side 2. Slide side two in slot.
See 3 printed, asking for card side 3. Slide side three in slot.
See 4 printed, asking for card side 4. Slide side four in slot.
Calculator and printer halt.

RE-QUIRKS- Elsewhere in this issue we dedicated almost an entire page to the strange PGM 02 SBR 239 sequence. Palmer O. Hansen has been the most active member investigating it. As in the particular location there in an unintended LRN (code 31) it is obvious one should look for the same in other modules. Of the many investigated, Palmer found a very promising one in the RPN module:
GTO nnn (000 to 007) LRN PGM 01 SBR 564 GTO 500 LRN RST R/S DEL DEL (you will see fractured digits in the process) Now, depending on nnn your results will differ. But most of the time you will get a translation state, i.e. the code of a program entered in user memory will be either too high or too low by a certain amount.
The strangest one of them all, however, is this one:
GTO 006 LRN PGM 01 SBR 564 LRN RST R/S DEL DEL 5 X 4 = PRT (look at the printing with the leading zeros) Try also to put the calculator in TRACE mode.

You might have asked yourself why I am so interested in these, seemingly, useless quirks. May I remind you, however, that thanks to the PGM 02 SBR 239 quirk we are able to fill one bank easily with SSTs, to be used as the best diagnostic we ever had. Also, it permist us to write a row of HIRs anywhere in a program and, using Richard Snow's trick, perform any HIR function from the key board, using the SST key as the HIR button. Even an IND HIR is possible now. So, keep on searching!

ZEROS OF FUNCTIONS.- This program, written by John Wortington and Emil Regelman of Bowie, MD, uses the Secant method to search for the value of any one of ten variables that will make a given function equal to zero. After a solution is found, all variables are printed along with a descriptor in the right-hand margin. Default descriptors are A through E, through A' through E'. At entry of the variables a different descriptor may be entered, up to three characters long, through the x:t key.

The Secant method uses the following equation to generate its estimate:

$$E_2 = E_1 - \frac{(E_1 - E_o)}{(V_1 - V_o)} \cdot V_1$$

where... E_o = previous estimate
E_1 = current estimate
E_2 = next estimate
V_o = value of f(x) with E_o
V_1 = value of f(x) with E_1

The search is terminated when $E_1 = E_2$. The initial values of E_o and V_o are arbitrarily chosen to be 0 and 0.02 respectively. Therefore, the initial estimate for the unknown cannot be zero, since it would immediately satisfy this relationship, and terminate the search.

Even though the Secant method provides rapid search with minimal initial parameters, it will generally not find more than one root. Also, since the range of the search is uncontrolled, it is more prone to testing areas of discontinuity and failing. Using a different initial estimate, however, will alter the search pattern and may avoid such difficulties.

The complete program will fit on one card side. Some of the commands will have to be synthesized, such as the HIRs. Steps 088 and 089 are no mistakes: they read IFF 20. This has to be synthesized, of course,by keing IFF STO 20 after which you delete the STO command. This command will normally set flag 0, but if a direct-address call is made to step 089 (which is done at step 127) the command will do CLR.(2nd CLR = code 20) This is a clever way of killing two birds with one stone.

Note also the extreme economy in program steps for an IND STOrage of entry data through keys A through E and A' through E'.

The GTO IND 00 (listed as GO* 00) is used here because it is shorter than GTO 000, if you make sure to have a zero at that particular time in reg 00. RST was not possible, because it would also reset flags.

EX* 32 means, of course, EXC IND 32, something not often used in programs. Those commands, as well as other INDs can be keyed in directly from the keyboard and require no synthesizing techniques.

Note also the HIR loading done in LBL RCL.(steps 132 and following)

INSTRUCTIONS:

After you key in the program, or load it from one card side, press GTO SBR LRN and key in a subroutine, ending in RTN, describing the function you want to search. Variables in your SBR are keyed in as A, B, C, etc. up to E'. Thus, if you want, for example, to search 3x + 3y + z - 15 = 0 , you key in, starting at step 240:

(3 X A) + (3 X B) + C − 15 = RTN in which A stands for x, B stands for Y, etc.

Now go out of LRN mode and enter the values for the variables. (or approximate guesses) Thus, you might enter 2 A 4 B 10 C

If you now press SBR RCL , you will see a list of all the entered variables with A through C in the right-hand margin.

Suppose you would like to call the first entry X instead of A, etc., you enter as follows: 44 x:t 2 A 45 x:t 4 B 46 x:t 10 C in which 44, 45 and 46 are the codes for X, Y and Z respectively. Follow again by SBR RCL to list the variables.

If you press SBR LIST you may list the SBR you entered.

To start the search press SBR R/S. Some searches take time, so be patient. If you want all the intermediate values printed, rather than pause-displayed, replace step 116 with a PRT command. It slows down program execution somewhat, however.

When the search is finished the printer will print a list of all the variables entered and of the unknown computed.

The "fixed" mode will limit the accuracy of the search to the number of decimal places and will provide a print-out in the fixed format.The value computed is automatically fixed, rounded to the number of places requested, and stored in its appropriate memory. The displayed value, however, is also fixed, but has NOT been rounded off.

(over)

(Zeros of functions. Cont.)

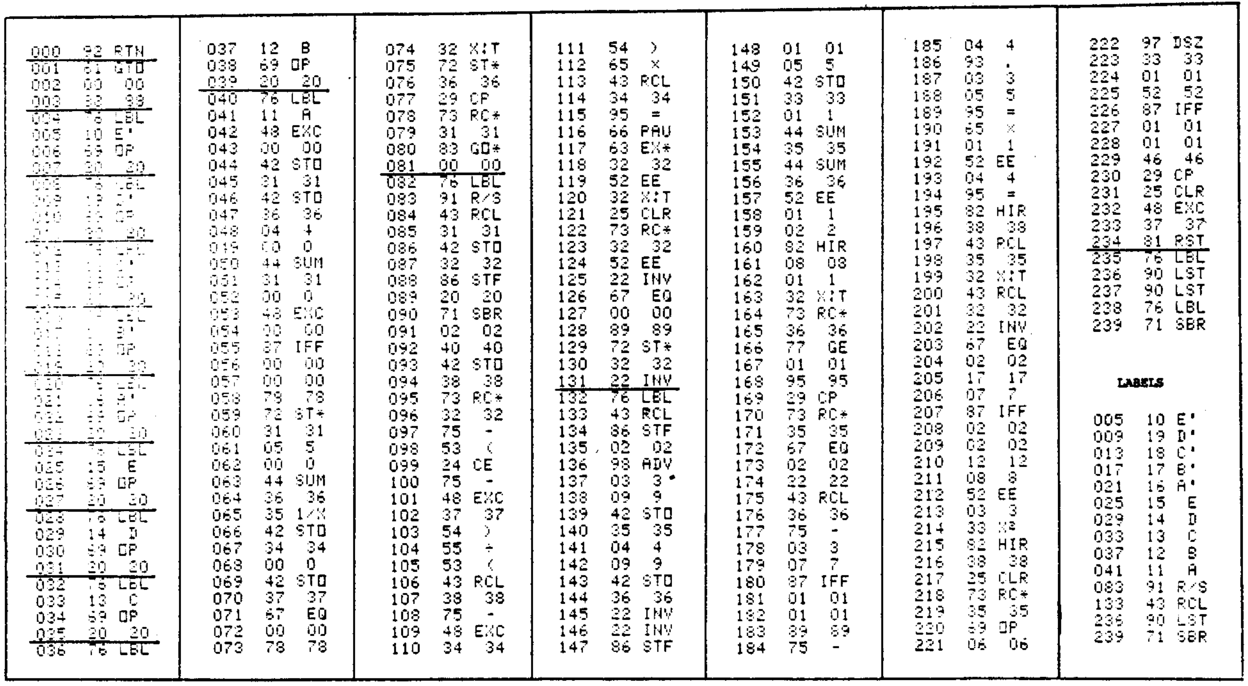
To delete a title in the right-hand margin, enter a value smaller than 1, BUT NOT EQUAL TO ZERO, into the t-register and press A through E', as appropriate.

Memories 00 and <u>30 through 59</u> are used. Memories 00 and <u>10 through 39</u> may be used by making the following changes:

Location	048	061	137	141	178
change	4	5	3	4	3
to	1	2	0	1	0

The search for the same variable may be repeated, with different values for the known variables, by storing the new value through the appropriate key A through E' and starting a new search by pressing SBR R/S.

To search for a <u>different</u> variable, store the new estimate and press SBR R/S.



```
000  92 RTN
001  61 GTO
002  00  00
003  33  33
004  76 LBL
005  10 E'
006  69 OP
007  20  20
008  76 LBL
009  19 D'
010  69 OP
011  20  20
012  76 LBL
013  18 C'
014  69 OP
015  20  20
016  76 LBL
017  17 B'
018  69 OP
019  20  20
020  76 LBL
021  16 A'
022  69 OP
023  20  20
024  76 LBL
025  15  E
026  69 OP
027  20  20
028  76 LBL
029  14  D
030  69 OP
031  20  20
032  76 LBL
033  13  C
034  69 OP
035  20  20
036  76 LBL

037  12  B
038  69 OP
039  20  20
040  76 LBL
041  11  A
042  48 EXC
043  00  00
044  42 STO
045  31  31
046  42 STO
047  36  36
048  04  4
049  00  0
050  44 SUM
051  31  31
052  00  0
053  48 EXC
054  00  00
055  87 IFF
056  00  00
057  00  00
058  78  78
059  72 ST*
060  31  31
061  05  5
062  00  0
063  44 SUM
064  36  36
065  35 1/X
066  42 STO
067  34  34
068  00  0
069  42 STO
070  37  37
071  67  EQ
072  00  00
073  78  78

074  32 X:T
075  72 ST*
076  36  36
077  29 CP
078  73 RC*
079  31  31
080  83 GO*
081  00  00
082  76 LBL
083  91 R/S
084  43 RCL
085  31  31
086  42 STO
087  32  32
088  86 STF
089  20  20
090  71 SBR
091  02  02
092  40  40
093  42 STO
094  38  38
095  73 RC*
096  32  32
097  75  -
098  53  (
099  24 CE
100  75  -
101  48 EXC
102  37  37
103  54  )
104  55  ÷
105  53  (
106  43 RCL
107  38  38
108  75  -
109  48 EXC
110  34  34

111  54  )
112  65  ×
113  43 RCL
114  34  34
115  95  =
116  66 PAU
117  63 EX*
118  32  32
119  52 EE
120  32 X:T
121  25 CLR
122  73 RC*
123  32  32
124  52 EE
125  22 INV
126  67  EQ
127  00  00
128  89  89
129  72 ST*
130  32  32
131  22 INV
132  76 LBL
133  43 RCL
134  86 STF
135  02  02
136  98 ADV
137  03  3
138  09  9
139  42 STO
140  35  35
141  04  4
142  09  9
143  42 STO
144  36  36
145  22 INV
146  22 INV
147  86 STF

148  01  01
149  05  5
150  42 STO
151  33  33
152  01  1
153  44 SUM
154  35  35
155  44 SUM
156  36  36
157  52 EE
158  01  1
159  02  2
160  82 HIR
161  08  08
162  01  1
163  32 X:T
164  73 RC*
165  36  36
166  77  GE
167  01  01
168  95  95
169  29 CP
170  73 RC*
171  35  35
172  67  EQ
173  02  02
174  22  22
175  43 RCL
176  36  36
177  75  -
178  03  3
179  07  7
180  87 IFF
181  01  01
182  01  01
183  89  89
184  75  -

185  04  4
186  93  .
187  03  3
188  05  5
189  95  =
190  65  ×
191  01  1
192  52 EE
193  04  4
194  95  =
195  82 HIR
196  38  38
197  43 RCL
198  35  35
199  32 X:T
200  43 RCL
201  32  32
202  22 INV
203  67  EQ
204  02  02
205  17  17
206  07  7
207  87 IFF
208  02  02
209  02  02
210  12  12
211  08  8
212  52 EE
213  03  3
214  33 X²
215  82 HIR
216  38  38
217  25 CLR
218  73 RC*
219  35  35
220  69 OP
221  06  06

222  97 DSZ
223  33  33
224  01  01
225  52  52
226  87 IFF
227  01  01
228  01  01
229  46  46
230  29 CP
231  25 CLR
232  48 EXC
233  37  37
234  81 RST
235  76 LBL
236  90 LST
237  90 LST
238  76 LBL
239  71 SBR

      LABELS

005  10 E'
009  19 D'
013  18 C'
017  17 B'
021  16 A'
025  15  E
029  14  D
033  13  C
037  12  B
041  11  A
083  91 R/S
133  43 RCL
236  90 LST
239  71 SBR
```

<u>SPEEDY FACTOR FINDER.</u> The Math/Utility module contains a program (MU 09) to find all the prime factors of any positive integer. It is, sadly, not the greatest of programs. Thanks to it being in firmware, it runs 3 min 30 sec to factor the number 987654321. If you download it, however, it runs in user memory about 7 min 37 sec for the same number. Mike Louder wrote a speedy factor finder once,intended for the SR-52. Dick Vanderburgh adapted it for the TI-59 in V3N11p4. Now Bill Skillman sped it up some more, such that for the number above it runs only 2 min 26 sec. In firmware, in a module, that would mean only 1 min 35 sec !

Instructions: 1. Enter program in bank 1.

2. Enter the number, N, and press E. Factors will be printed.

No printer ? Replace the PRT instructions with either Pauses or R/S.

The program doesn't use any module, so SR-52 and SR-56 users, here is your chance.

```
000: R/S LBL D' 2 LBL D SUM 02 RCL 01 DIV RCL 02 = INV INT EQ 020 RTN RCL 01 DIV
023: X:T RCL 02 EQ 078 PRT = STO 01 STO 03 CP GTO 008 LBL E CP CMS STO 01 PRT ADV
047: STO 03 D' 1 D D' CP D' 6 D 4 D D' 4 D D' 4 D 6 D RCL 02 X:T RCL 03 √X  GE 053
076: RCL 01 PRT ADV CLR RST
```

Labels: at 002: 19 D' , at 005: 14 D , at 040: 15 E

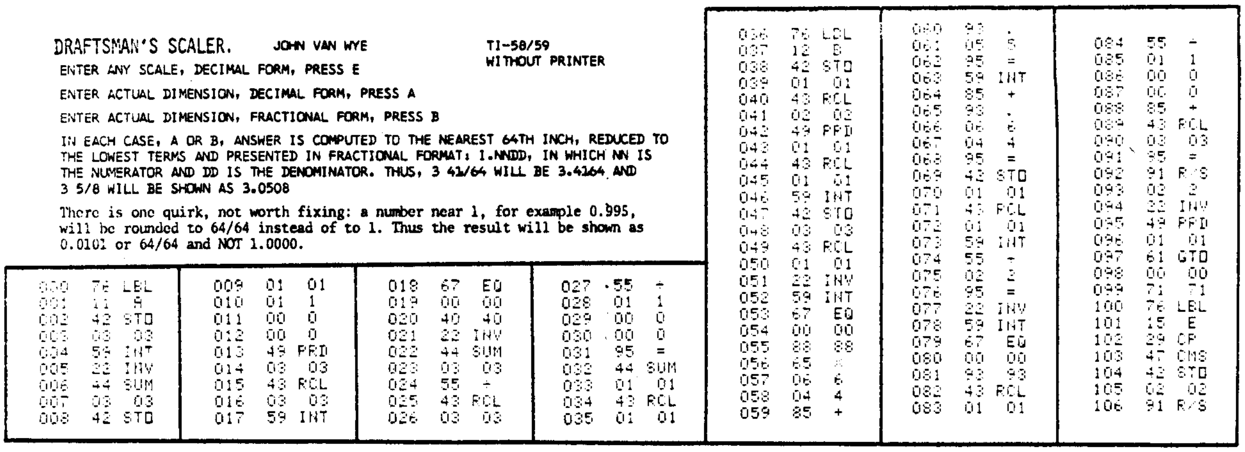
Registers used : R01, R02 and R03, t-reg.

Who sends me an SR-52 and an SR-56 version?

DRAFTMAN'S SCALER.- Lots of professions in the US have switched to the metric system. Pharmacists have given up their ounces, grains and drams and use grams instead. Chemists have since long embraced the new system. Engineers is another matter: most of them think they have switched. But I often encounter descriptions as "... torque is 100 millinewtons·meter .... extending the lever a few inches, however..." No comment.

But professions you would think would benefit most from the switch, such as draftsmen for example, are still measuring by 64ths as their smallest unit. On the other hand, mechanical engineers and technicians most of the time specify dimensions in decimal inches. Thus, draftsmen are constantly faced with converting decimal inches to 64ths. On top of that they have to contend with scale factors, requiring them to multiply 64ths by a scale factor, later to reduce the results to the nearest 64ths again.

John Van Wye, who, as far as I know, is no draftsman, sent me the program below. It is fast and should be very practical.



DRAFTSMAN'S SCALER. JOHN VAN WYE TI-58/59 WITHOUT PRINTER

ENTER ANY SCALE, DECIMAL FORM, PRESS E

ENTER ACTUAL DIMENSION, DECIMAL FORM, PRESS A

ENTER ACTUAL DIMENSION, FRACTIONAL FORM, PRESS B

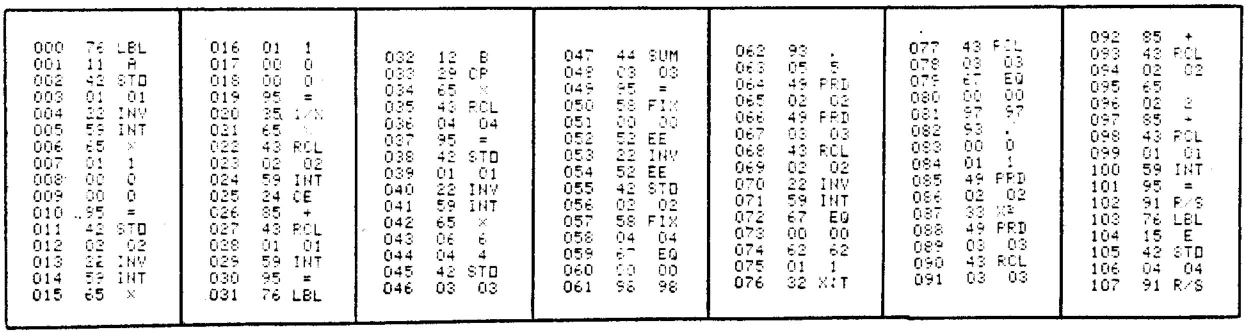
IN EACH CASE, A OR B, ANSWER IS COMPUTED TO THE NEAREST 64TH INCH, REDUCED TO THE LOWEST TERMS AND PRESENTED IN FRACTIONAL FORMAT: I.NNDD, IN WHICH NN IS THE NUMERATOR AND DD IS THE DENOMINATOR. THUS, 3 41/64 WILL BE 3.4164 AND 3 5/8 WILL BE SHOWN AS 3.0508

There is one quirk, not worth fixing: a number near 1, for example 0.995, will be rounded to 64/64 instead of to 1. Thus the result will be shown as 0.0101 or 64/64 and NOT 1.0000.

```
000  76  LBL
001  11   A
002  42  STO
003  03   03
004  59  INT
005  22  INV
006  44  SUM
007  03   03
008  42  STO
009  01   01
010  01   1
011  00   0
012  00   0
013  49  PRD
014  03   03
015  43  RCL
016  03   03
017  59  INT
018  67   EQ
019  00   00
020  40   40
021  22  INV
022  44  SUM
023  03   03
024  55   ÷
025  43  RCL
026  03   03
027  55   ÷
028  01   1
029  00   0
030  00   0
031  95   =
032  44  SUM
033  01   01
034  43  RCL
035  01   01
036  76  LBL
037  12   B
038  42  STO
039  01   01
040  43  RCL
041  02   02
042  49  PRD
043  01   01
044  43  RCL
045  01   01
046  59  INT
047  42  STO
048  03   03
049  43  RCL
050  01   01
051  22  INV
052  59  INT
053  67   EQ
054  00   00
055  88   88
056  65   ×
057  06   6
058  04   4
059  85   +
060  93   .
061  05   5
062  95   =
063  59  INT
064  85   +
065  93   .
066  06   6
067  04   4
068  95   =
069  42  STO
070  01   01
071  43  RCL
072  01   01
073  59  INT
074  55   ÷
075  02   2
076  95   =
077  22  INV
078  59  INT
079  67   EQ
080  00   00
081  93   93
082  43  RCL
083  01   01
084  55   ÷
085  01   1
086  00   0
087  00   0
088  85   +
089  43  RCL
090  03   03
091  95   =
092  91  R/S
093  02   2
094  22  INV
095  49  PRD
096  01   01
097  61  GTO
098  00   00
099  71   71
100  76  LBL
101  15   E
102  29  CP
103  47  CMS
104  42  STO
105  02   02
106  91  R/S
```

As you may observe, John says that his proqram contains an insignificant quirk. What follows is not intended to embarrass John. Far from it, his program is a gem. My only intention is to show how beautiful our "referee" system is.

I sent the program to two reviewers: Palmer O. Hanson and Richard Snow. Neither could resist to get that quirk out. But, in the process of doing it, Palmer found another one: the program gets lost in a loop for an input of, say, 1.005, which should yield 1.0064. So he inserted a test for zero after the INT command at step 063. His program follows:

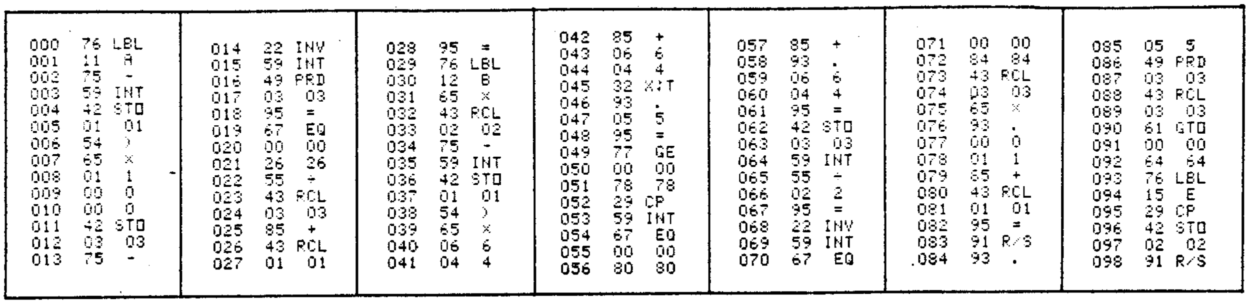


```
000  76  LBL
001  11   A
002  42  STO
003  01   01
004  22  INV
005  59  INT
006  65   ×
007  01   1
008  00   0
009  00   0
010  95   =
011  42  STO
012  02   02
013  22  INV
014  59  INT
015  65   ×
016  01   1
017  00   0
018  00   0
019  95   =
020  35  1/X
021  65   ×
022  43  RCL
023  02   02
024  59  INT
025  24  CE
026  85   +
027  43  RCL
028  01   01
029  59  INT
030  95   =
031  76  LBL
032  12   B
033  29  CP
034  65   ×
035  43  RCL
036  04   04
037  95   =
038  42  STO
039  01   01
040  22  INV
041  59  INT
042  65   ×
043  06   6
044  04   4
045  42  STO
046  03   03
047  44  SUM
048  03   03
049  95   =
050  58  FIX
051  00   00
052  52  EE
053  22  INV
054  52  EE
055  42  STO
056  02   02
057  58  FIX
058  04   04
059  67   EQ
060  00   00
061  98   98
062  93   .
063  05   5
064  49  PRD
065  02   02
066  49  PRD
067  03   03
068  43  RCL
069  02   02
070  22  INV
071  59  INT
072  67   EQ
073  00   00
074  62   62
075  01   1
076  32  X:T
077  43  RCL
078  03   03
079  67   EQ
080  00   00
081  97   97
082  93   .
083  00   0
084  01   1
085  49  PRD
086  02   02
087  33  X²
088  49  PRD
089  03   03
090  43  RCL
091  03   03
092  85   +
093  43  RCL
094  02   02
095  65   ×
096  02   2
097  85   +
098  43  RCL
099  01   01
100  59  INT
101  95   =
102  91  R/S
103  76  LBL
104  15   E
105  42  STO
106  04   04
107  91  R/S
```

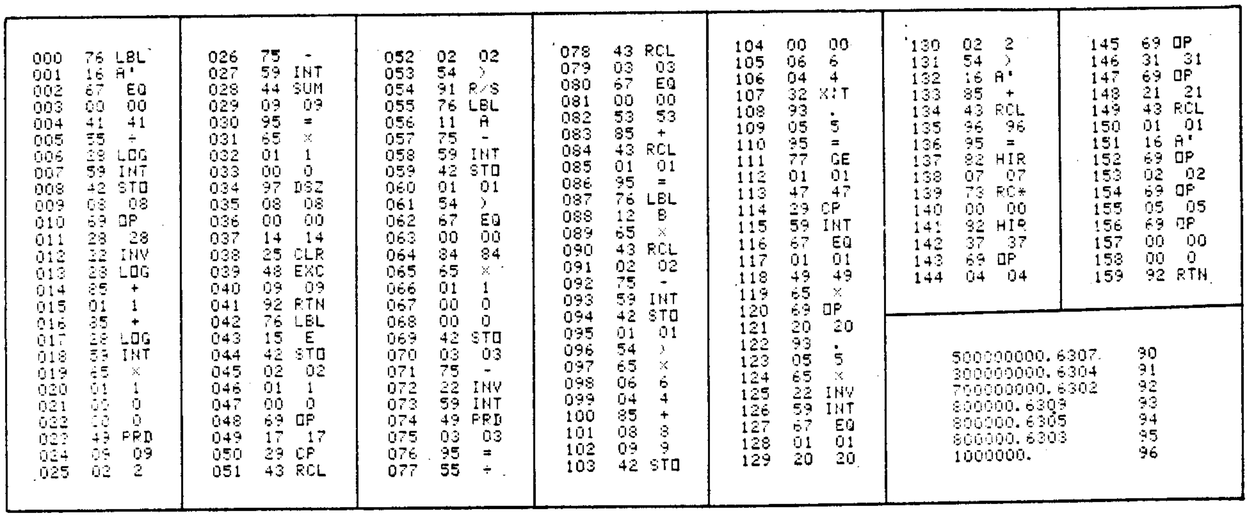
Palmer's program runs a little longer than John's, but it is worth it. Richard Snow, on the other hand, also sent me an enhanced version and a print version. But both had the "endless loop" quirk in them. So, I alarmed Richard about it and in less than a week I had two new versions, both running perfectly. So, here follows Richard's "calculator only" version, followed by the "print only" variety.

(over)

(Draftman's scaler. Cont.)



```
000  76 LBL
001  11  A
002  75  -
003  59 INT
004  42 STO
005  01  01
006  54  )
007  65  ×
008  01  1
009  00  0
010  00  0
011  42 STO
012  03  03
013  75  -
014  22 INV
015  59 INT
016  49 PRD
017  03  03
018  95  =
019  67  EQ
020  00  00
021  26  26
022  55  ÷
023  43 RCL
024  03  03
025  85  +
026  43 RCL
027  01  01
028  95  =
029  76 LBL
030  12  B
031  65  ×
032  43 RCL
033  02  02
034  75  -
035  59 INT
036  42 STO
037  01  01
038  54  )
039  65  ×
040  06  6
041  04  4
042  85  +
043  06  6
044  04  4
045  32 X:T
046  93  .
047  05  5
048  95  =
049  77  GE
050  00  00
051  78  78
052  29 CP
053  59 INT
054  67  EQ
055  00  00
056  80  80
057  85  +
058  93  .
059  06  6
060  04  4
061  95  =
062  42 STO
063  03  03
064  59 INT
065  55  ÷
066  02  2
067  95  =
068  22 INV
069  59 INT
070  67  EQ
071  00  00
072  84  84
073  43 RCL
074  03  03
075  65  ×
076  93  .
077  00  0
078  01  1
079  85  +
080  43 RCL
081  01  01
082  95  =
083  91 R/S
.084  93  .
085  05  5
086  49 PRD
087  03  03
088  43 RCL
089  03  03
090  61 GTO
091  00  00
092  64  64
093  76 LBL
094  15  E
095  29 CP
096  42 STO
097  02  02
098  91 R/S
```



```
000  76 LBL
001  16 A'
002  67  EQ
003  00  00
004  41  41
005  55  ÷
006  28 LOG
007  59 INT
008  42 STO
009  08  08
010  69 OP
011  28  28
012  22 INV
013  28 LOG
014  85  +
015  01  1
016  85  +
017  28 LOG
018  59 INT
019  65  ×
020  01  1
021  00  0
022  00  0
023  49 PRD
024  09  09
.025  02  2
026  75  -
027  59 INT
028  44 SUM
029  09  09
030  95  =
031  65  ×
032  01  1
033  00  0
034  97 DSZ
035  08  08
036  00  00
037  14  14
038  25 CLR
039  48 EXC
040  09  09
041  92 RTN
042  76 LBL
043  15  E
044  42 STO
045  02  02
046  01  1
047  00  0
048  69 OP
049  17  17
050  29 CP
051  43 RCL
052  02  02
053  54  )
054  91 R/S
055  76 LBL
056  11  A
057  75  -
058  59 INT
059  42 STO
060  01  01
061  54  )
062  67  EQ
063  00  00
064  84  84
065  65  ×
066  01  1
067  00  0
068  00  0
069  42 STO
070  03  03
071  75  -
072  22 INV
073  59 INT
074  49 PRD
075  03  03
076  95  =
077  55  ÷
078  43 RCL
079  03  03
080  67  EQ
081  00  00
082  53  53
083  85  +
084  43 RCL
085  01  01
086  95  =
087  76 LBL
088  12  B
089  65  ×
090  43 RCL
091  02  02
092  75  -
093  59 INT
094  42 STO
095  01  01
096  54  )
097  65  ×
098  06  6
099  04  4
100  85  +
101  08  8
102  09  9
103  42 STO
104  00  00
105  06  6
106  04  4
107  32 X:T
108  93  .
109  05  5
110  95  =
111  77  GE
112  01  01
113  47  47
114  29 CP
115  59 INT
116  67  EQ
117  01  01
118  49  49
119  65  ×
120  69 OP
121  20  20
122  93  .
123  05  5
124  65  ×
125  22 INV
126  59 INT
127  67  EQ
128  01  01
129  20  20
130  02  2
131  54  )
132  16 A'
133  85  +
134  43 RCL
135  96  96
136  95  =
137  82 HIR
138  07  07
139  73 RC*
140  00  00
141  82 HIR
142  37  37
143  69 OP
144  04  04
145  69 OP
146  31  31
147  69 OP
148  21  21
149  43 RCL
150  01  01
151  16 A'
152  69 OP
153  02  02
154  69 OP
155  05  05
156  69 OP
157  00  00
158  00  0
159  92 RTN

500000000.6307   90
300000000.6304   91
700000000.6302   92
800000.6309      93
800000.6305      94
800000.6303      95
1000000.         96
```

After which, for good measure, Richard wrote this SR-56 version. Although it was never actually tested on an SR-56, both Richard and I ran it on a TI-59 and it works fine. We are that confident!

The instructions are slightly different, of course:
Enter any scale factor, in decimal form, such as 1.75 or 2.45, etc. Press RST R/S. Enter the actual dimension in fraction form, such as 1.0564 for 1 5/64 and press R/S. See the dimension multiplied by the scale factor, reduced to the simplest terms. As an example, enter scale factor 2 and press RST R/S. Then enter 1.0732 for 1 7/32. Press R/S. Result should be, as expected, 1.0716 for 1 7/16.

```
00:  CP  STO 2 R/S  - INT STO 1 ) X 100  STO 3  - INV INT PRD 3  =  X=T 28 ÷
25:  RCL 3  +  RCL 1  =  X  RCL 2  -  INT  STO 1  =  X=T 80  X  64  +  64  X⇄T
49:  .  5  =  X≥T 78  CP  INT  +  . 64  =  STO 3  INT  ÷  2  =  INV INT  X=T
71:  87  RCL 3  X  .01  +  RCL 1  = R/S  GTO 04  .5  PRD 3  RCL 3  GTO 64
```

Last step used : 95.

MORE HIRMANIA.- Re-v5n4/5p26, the easy key board method by Richard Snow. It seems
-------------- that Clyde Durbin, in Dallas Texas, has been using a similar method for some time. It is simplicity itself! Clyde puts one, single 82 code at the very last step of the current partition. It is easy to go to this step: just press R/S. Then, if flashing occurs, press CLR. Now, in order to store something, say, in HIR 8, enter the number and press SST 08 CLR. To recall the same, press SST 18 CLR. Instead of SST you can also use R/S. All other HIR functions...etc.

PRINT CODE CONVERTERS.- Re- v5n3p15: Palmer O. Hanson observes that in both the original Snow routine and the Skillman sequence R09 and R08 have to be cleared prior to calling either routine if a single zero has to be generated. Bill Skillman notes the same in v5n4/5p 26, Errata. Palmer has a remedy for it in the form of a revised sequence involving the same number of steps. It illustrates the concept of jumping into the middle of a command to obtain a different result. The revised Snow routine will look as follows:

```
000:  LBL PRT  CP  EQ  041  DIV  LOG  INT  STO 08  OP 28  INV LOG  +  1  +  LOG
019:  INT  X  100  PRD  01  2  -  INT SUM 01  =  X 10  DSZ 8 015  CLR  RCL 01 RTN
```

While the Skillman routine is written as:

```
000:  LBL PRT  CP  EQ  052  X:T  INV GE  012  20  STO 01  X:T  ABS  DIV  LOG  INT
020:  STO 08  OP 28  INV LOG +  1  +  LOG INT  X  100  PER 01  2  -  INT  SUM 01
042:  =  X  10  DSZ 8  026  CLR  RCL 01  RTN
```

Karl Gailer, Auckland, New Zealand, has also been delving in print code converters. He says that most of them are rather slow. Don't we all? So, for those special occasions where you don't need the full range of numbers from 0 to 99999, you could speed up things quite a bit by using his limited-range solutions. For digits 0 through 8, this simple one is ideal:

```
LBL PRT  +  1  ) RTN  ( only 6 steps)
```

A range from 0 through 18 could be realized with this 24-step routine:

```
LBL PRT  X:T  9  X:T  INV GE  COS  EQ  TAN  +  188  LBL TAN  +  2  LBL COS  +  1
=  RTN
```

A 32-step routine can extend the range to 0 through 89:

```
LBL PRT  DIV  10  +  1.1  + INT  X 99  +  ( INV  INT  X 10  +  OP 10  X 88  -  88
=  RTN
```

And finally, I had 21 inqueries as to what the original Snow routine looked like! So, for those who never saw v2n10p2 of 52-Notes, here is Robert Snow's original print code converter, useful in the range 0 through 99999:

```
LBL PRT CP  EQ  COS  DIV  LOG INT  STO 01  OP 21  INV LOG  LBL TAN  +  X:T  100
PRD 02  8  GE  COS  2  +  LBL COS  1  -  INT SUM  02  =  X  10 DSZ 1  TAN  CLR
EXC  02  RTN
```

To which Elmer Clausen has this variation that does not use the t-register. It has the same number of steps,though.

```
LBL PRT  +  .5  STO 08  =  DIV LOG  INT  SUM 08  INV LOG  + 1 + LOG  INT  X  100
PRD 09  2  -  INT SUM 09  =  X  10 DSZ 8  015  CLR  EXC 09  RTN
```

This routine should be started at step 000. If you want to relocate it, change the 015 address after DSZ 8.

ERRATA.- Most people caught the typo on v5n4p25. But just for the record: In the bottom routine "LBL A PGM 15..." change to read "LBL A PGM 05..." Sorry about that.

Søren Birkelund Hansen, in Denmark, caught this one: Having been brought up with cm, meters and km, I have a hard time keeping ordinary land miles and nautical miles straight. I know that one is .86897624 times the other, but which one is larger? PGM 24 of the ML library does it correctly for you, but I placed the descriptors in the wrong routine. So in v5n1p8 :
Exchange steps 187 to 192 with steps 199 to 204 and
steps 212 to 217 with steps 224 to 229.

TRAVERSE.- As we promised in one of the former issues, Frank Blachly wrote an SR-52 program, more or less equivalent to the TI-59 program by the same name in v5n4/5p27-28. Here it is:

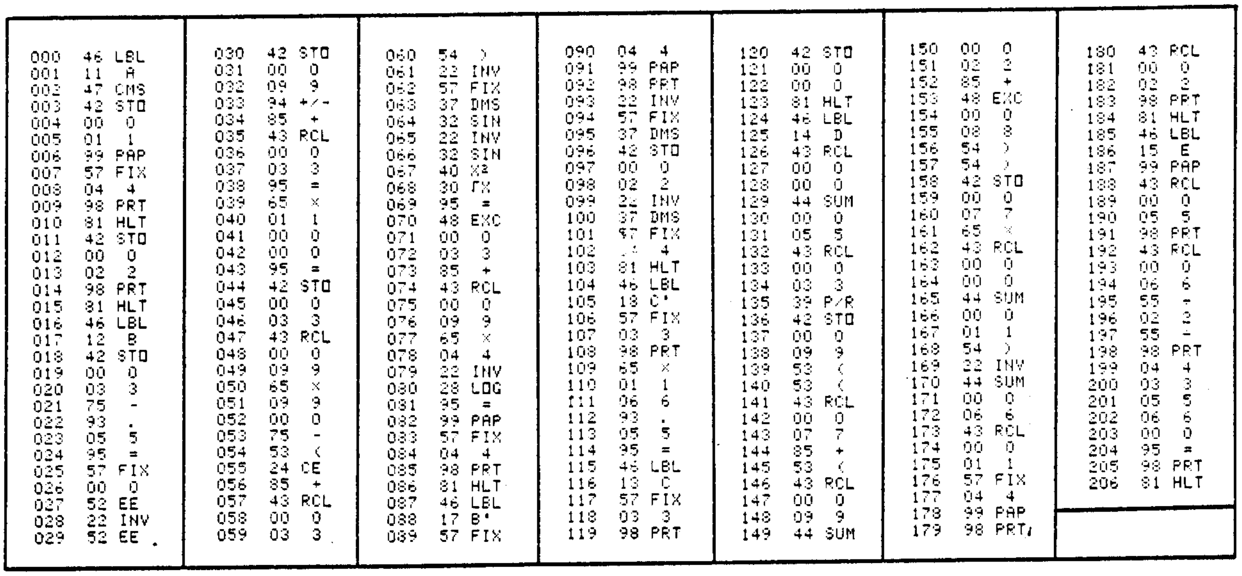
BEARING TRAVERSE for the SR-52.- With reference to the same program for the TI-59 in v5n4/5p27-28, this program does almost everything the TI-59 program does, be it with simpler print-outs. The program works both with and without the printer.

Instructions:

1. Enter Beginning Northing and press A.
2. Enter Beginning Easting and press R/S.

3a. Enter either Quadrant & Bearing as Q.Br and press B, <u>OR</u>
3b. enter Azimuth and press 2nd B'.
4a. Enter either Distance in feet and press C, <u>OR</u>
4b. enter Distance in perches and press 2nd C'.

5. Compute coordinates, press D.
   (accumulates perimeter and area)
6. Print perimeter and area, press E
   (perim. sqft acres)



```
000  46 LBL
001  11  A
002  47 CMS
003  42 STO
004  00  0
005  01  1
006  99 PAP
007  57 FIX
008  04  4
009  98 PRT
010  81 HLT
011  42 STO
012  00  0
013  02  2
014  98 PRT
015  81 HLT
016  46 LBL
017  12  B
018  42 STO
019  00  0
020  03  3
021  75  -
022  93  .
023  05  5
024  95  =
025  57 FIX
026  00  0
027  52 EE
028  22 INV
029  52 EE .

030  42 STO
031  00  0
032  09  9
033  94 +/-
034  85  +
035  43 RCL
036  00  0
037  03  3
038  95  =
039  65  ×
040  01  1
041  00  0
042  00  0
043  95  =
044  42 STO
045  00  0
046  03  3
047  43 RCL
048  00  0
049  09  9
050  65  ×
051  09  9
052  00  0
053  75  -
054  53  (
055  24 CE
056  85  +
057  43 RCL
058  00  0
059  03  3

060  54  )
061  22 INV
062  57 FIX
063  37 DMS
064  32 SIN
065  22 INV
066  32 SIN
067  40 X²
068  30 √X
069  95  =
070  48 EXC
071  00  0
072  03  3
073  85  +
074  43 RCL
075  00  0
076  09  9
077  65  ×
078  04  4
079  22 INV
080  28 LOG
081  95  =
082  99 PAP
083  57 FIX
084  04  4
085  98 PRT
086  81 HLT
087  46 LBL
088  17 B'
089  57 FIX

090  04  4
091  99 PAP
092  98 PRT
093  22 INV
094  57 FIX
095  37 DMS
096  42 STO
097  00  0
098  02  2
099  22 INV
100  37 DMS
101  57 FIX
102  04  4
103  81 HLT
104  46 LBL
105  18 C'
106  57 FIX
107  03  3
108  98 PRT
109  65  ×
110  01  1
111  06  6
112  93  .
113  05  5
114  95  =
115  46 LBL
116  13  C
117  57 FIX
118  03  3
119  98 PRT

120  42 STO
121  00  0
122  00  0
123  81 HLT
124  46 LBL
125  14  D
126  43 RCL
127  00  0
128  00  0
129  44 SUM
130  00  0
131  05  5
132  43 RCL
133  00  0
134  03  3
135  39 P/R
136  42 STO
137  00  0
138  09  9
139  53  (
140  53  (
141  43 RCL
142  00  0
143  07  7
144  85  +
145  53  (
146  43 RCL
147  00  0
148  09  9
149  44 SUM

150  00  0
151  02  2
152  85  +
153  48 EXC
154  00  0
155  08  8
156  54  )
157  54  )
158  42 STO
159  00  0
160  07  7
161  65  ×
162  43 RCL
163  00  0
164  00  0
165  44 SUM
166  00  0
167  01  1
168  54  )
169  22 INV
170  44 SUM
171  00  0
172  06  6
173  43 RCL
174  00  0
175  01  1
176  57 FIX
177  04  4
178  99 PAP
179  98 PRT

180  43 RCL
181  00  0
182  02  2
183  98 PRT
184  81 HLT
185  46 LBL
186  15  E
187  99 PAP
188  43 RCL
189  00  0
190  05  5
191  98 PRT
192  43 RCL
193  00  0
194  06  6
195  55  ÷
196  02  2
197  55  ÷
198  98 PRT
199  04  4
200  03  3
201  05  5
202  06  6
203  00  0
204  95  =
205  98 PRT
206  81 HLT
```

QUADRATIC EQUATIONS- In v5n4/5p13, Stuart Cox presented a program to solve quadratic equations. It was a program especially conceived for the TI-58/59, that "also ran" on the TI-57. But the latter is in a category all by itself. Somebody who is really at home on the 57 might write shorter or faster programs by manipulating that calculator's special functions. Here is a program written by Peter Van Roy, that solves quadratic equations with, for TI-58/59 users, rather unusual key strokes.

Given an equation of the form $ax^2 + bx + c = 0$, in which $a \neq 0$.

Key in the program, enter the first coefficient a and press R/S. Enter b, followed by R/S and finally enter c and press R/S.

See the first root displayed. Press x:t and see the second root displayed.

IF THE DISPLAY IS BLINKING, IT MEANS THAT THE ROOT IS COMPLEX. In that case, see the real part displayed. Press x:t and see the imaginary part displayed.

00: STO0 CT CLR R/S STO1 2 DIV1 CLR R/S STO2 σ^2 +/- X<0? GTO2

14: $\sqrt{X}$ MIN7 - $\bar{X}$ MIN7 = R/S RST LBL2 +/- $\sqrt{X}$ X:T $\bar{X}$ +/- GTO4 RST

Last step used is 29. As may be observed, the use of σ^2 and $\bar{x}$ helped to greatly reduce the number of program steps.

MASTERMIND on the TI-57.- Responding to my call for good TI-57 programs, the editor of TI-SOFT in Kapellen, Belgium, Tom Coppens sent me several written by Peter Van Roy. Peter is considered the undisputed champion of the TI-57 programming crowd, and with reason.
For those who have never seen a TI-57 program: most steps are merged.(STO, RCL, EXC, PROD, SUM, SBR, GTO and others) This way the 50 available steps go a long way. Besides this, some functions are much, much faster than on the TI-58/59. For example, x̄ executes several times faster than on a 58/59. There are no user-defined labels on a 57, so programs start usually with RST R/S.

MASTERMIND,as most of you will remember, requires you to guess a secret number entered by a friend (or maybe an enemy, says Peter) or you might do it yourself, blindfolded. But stay away from the zero key. That one is taboo. Enter any positive number up to 10 digits long.

The result of your guess is returned as b.c, in which b is equal to the number of black pegs, i.e. the number of digits in the right position and c is the number of white pegs, i.e. the number of correct digits regardless of position.

Instructions: Initialize: (once only) 10 STO 0 , 0 STO 6.
Enter secret number STO 2 , RST CLR
Enter guess R/S . Result appears after about 20 sec.
Keep entering your guess and pressing R/S.
For new game, go to line 2. (# STO 2 RST CLR)

```
00: STO4  STO3  RCL2  STO1  SBR8  STO3  EXC6  STO5  (  RCL4  STO1  SBR8  EXC5
13: STO1  RCL6  STO3  SBR8  STO6  )  RCL5  =  R/S  RST  LBL8  CT  X̄   X=0?
27: RET (INVSBR)  (  STO1  FRAC  MIN1  STO7  X  RCL0  )  10^X  SUM6  INV X̄
39: STO3  FRAC  MIN3  X≥T?  X⇄T  SUM5  X≠T?  GTO8  1  +  GTO8
```

EPIDEMICS: In v5n4/5 I showed the German cartoon about the Moonlanding epidemic. When I was in Lubbock last fall, there was a veritable WUMPUS epidemic: The PPX program analysts were wading through oodles of submissions with names such as THE HUNT FOR THE WUMPUS, or KILL THE WUMPUS, and WHO IS AFRAID OF THE BIG, BAD WUMPUS and other such attractive titles. As they have a rule to shy away from obvious duplications, but not wanting to offend all the ...ahem... members who submitted such nice programs, their handwringing was agonizing to behold. Sue Battig, executing one of her last official acts as outgoing editor of the PPX newsletter and chief of analysts, finally cut the Gordian knot and ... ... I cannot tell how. It is considered inside information. But Wumpus programs are definitely out in the PPX catalog. The same thing seems to have happened with MASTERMIND programs in Tom Coppens' TI-SOFT newsletter. He doesn't want any more submissions of it. So, please, don't send me any TI-58/59 Mastermind programs either. Everybody has tried his/her hand on those. I even had one manuscript returned from BYTE. I had submitted it in 1978: Mastermind for the SR-52. They even said I don't have to return the money they paid me to keep it on hold! But they were rather definite in their pronounciation: It wouldn't stand a chance of being published in the near future!

LABELS.- With respect to Jared Weinberger's routine in v5n4/5p8, Karl-Joseph Meusch offers this shorter routine. As opposed to Jared's routine, this one requirers two registers to be preloaded:
R00: 9200.760869 and R01: 9199.760869

```
000:100  STO 02  LBL  CLR  RCL 01  +  RCL 02  =  STO 00  CLR  OP 17  SBR  955
021: 9  OP 17  DSZ 2  CLR  R/S  ( last step used 027 )
```

Execute with RST R/S. All labels will be shown at step 958.

NEWCOMER'S CORNER.- Anatomy of a program. Tom Wysmuller, Essex, Connecticut, sent me a financial program called MONEY MARKET STATUS. There are about 25 Money Market Funds available today, Dryfuss Liquid Asset. E.F. Hutton Cash Reserve Management, etc. etc. People also invest in "T" bills, commercial paper, and like investments to yield fairly high returns; close to triple what savings accounts are permitted to pay. Interest is calculated on a daily basis, but only on whole dollars of the account. Any cents remaining are added to cents remaining as of the next day's computation. And if the cents remaining add up to over a dollar, only that dollar is added to the total used to compute the next day's interest. Standard practice is to compute interest only on the day following your cash deposit. The rate paid is usually revised weekly and sometimes daily!
All this to explain the algorithm:

$$\text{PRESENT AMOUNT} = \frac{\text{PAST AMOUNT (whole \$ only) X (decimal) INTEREST}}{365}$$

So, we could write a simple program in which we enter the amount by means of key A, the interest in decimal form by means of key B and we compute one day interest by pressing key C. As many times as we press C, we obtain the cumulative interest for the next day, provided the interest didn't change or we did not deposit nor withdraw anything from the amount. To allow for the latter, we could use key E, to simply add either a positive or a negative entry to a register and then recall that register into the display. The program would look like this.

```
000  76 LBL   005  65  ×    010  06  6    015  43 RCL   020  42 STO   025  42 STO   030  44 SUM
001  13  C    006  43 RCL   011  05  5    016  01  01   021  01  01   026  02  02   031  01  01
002  43 RCL   007  02  02   012  95  =    017  91 R/S   022  91 R/S   027  91 R/S   032  43 RCL
003  01  01   008  55  ÷    013  44 SUM   018  76 LBL   023  76 LBL   028  76 LBL   033  01  01
004  59 INT   009  03  3    014  01  01   019  11  A    024  12  B    029  15  E    034  91 R/S
```

This program works. For example, enter 2000, press A. Enter 0.125 (for 12.5%), press B. Press C and obtain the new, updated amount of 2000.694444
One glaring "fault" is to show all the decimals. For money it is easier to work with two digits only, the cents. So, press GTO A LRN. Then we open op two spaces by pressing 2nd INS twice. In those spaces we put FIX 2.
Now, do the same instructions again and see how much neater everything looks, rounded to two places. But good programmers can never leave well enough alone. For example, it is still "unfriendly" to the user to force him to enter the interest in decimal form. If you want him to enter, for example, 14.5 %, why not have him enter it in that form and have the program divide this number by 100 ? So, press GTO B and insert five steps: DIV 100 = . If we now press B, after entering , say, 14.5, we see 0.14, in FIX 2 mode. Confusing, to say the least. So, we insert a zero just before the last step of routine B, the R/S command. The display now always returns a zero, irrespective of the number entered. Try again the instructions and check if your program still runs OK.
Now suppose we have placed the amount at a certain interest for , say, 5 days. We could then press C five times, to obtain the cumulative interest for five days. But, the probability of error is large.You might easily press C six times. Therefore, we should have some way by which the programs counts the number of days and stops the execution when the last day has been taken into account. This we can do by storing the number of days over which the present interest is valid, through a separate key. We could use key D. (D for days) Then, after the last step, R/S, we could add a DSZ scheme, such that if the DSZ is NOT at zero, the program returns each time to its own beginning. If the DSZ register IS at zero, the program should stop and, to draw the user's attention, flash a zero. So, we do RST LRN and make space for 4 insertions. There we key LBL D STO 00 . Then after the R/S step we insert 6 steps: DSZ 0 C 0 1/X 0 R/S LBL A.... (The underlined steps are inserted)
Try the instructions now: 2000 A 14.5 B 5 D (5 days) and we see the amount + interest for the first day. Pressing now four times R/S in succession will show the amount + interest after day 2, after day 3..... If we press R/S once too many, we get a flashing zero.

(over)

(Newcomer's corner, continued.)

A further embellishment would be,to be able to enter the number of days by means of key D, after which we press key D as many times as we entered number of days, instead of having to press a different key (R/S). Thus, the D routine should recognize an ENTERED number, but ignore a LEFT-OVER number in the display. The decimal point trick (see v5n3p7) could be used here. An entry of zero days is inconceivable, so the trick is justified and will not contribute to error.
We press GTO D and insert four spaces: . CP EQ C' after which we go out of LRN. Then we go to the first R/S and immediately after it we insert two steps: LBL C'. This means now that, if we enter, say, 4 days and press D, the decimal point will be ignored and the 4 stored in R00. Then the main part of the program, LBL C is executed and the amount + interest is shown. Now pressing D again, with the left-over "amount + interest" in the display, the decimal point will render that number zero. CP makes the t-register contents zero, such that the EQ test will be true, with the program branching to C'.LBL C' will DSZ register 00 and, if NOT zero,return to LBL C. If the DSZ test results in zero, we will get a flashing zero.
LBL E can be used to either add to or subtract from the amount-register R01. For addition enter a positive number; for subtraction, enter a negative one. Routine E will always halt with the present amount in the display.
This is as far as I will normally go with a "calculator-only" program. In order to check if you made all the necessary modifications,here is the completed program.

```
000  76 LBL
001  14  D
002  93  .
003  29 CP
004  67  EQ
005  18 C'
006  42 STO
007  00  00
008  76 LBL
009  13  C
010  43 RCL
011  01  01
012  59 INT
013  65  x
014  43 RCL
015  02  02
016  55  ÷
017  03  3
018  06  6
019  05  5
020  95  =
021  44 SUM
022  01  01
023  43 RCL
024  01  01
025  91 R/S
026  76 LBL
027  18 C'
028  97 DSZ
029  00  00
030  13  C
031  00  0
032  35 1/X
033  00  0
034  91 R/S
035  76 LBL
036  11  A
037  58 FIX
038  02  02
039  42 STO
040  01  01
041  91 R/S
042  76 LBL
043  12  B
044  55  ÷
045  01  1
046  00  0
047  00  0
048  95  =
049  42 STO
050  02  02
051  00  0
052  91 R/S
053  76 LBL
054  15  E
055  44 SUM
056  01  01
057  43 RCL
058  01  01
059  91 R/S
```

So that SR-52 usersmight have some fun too, here follows the translation, complete with decimal point trick. The INT function has been synthesized.

```
000  46 LBL
001  14  D
002  93  .
003  90 IFZ
004  52 EE
005  42 STO
006  00  0
007  00  0
008  46 LBL
009  17 B'
010  43 RCL
011  00  0
012  01  1
013  22 INV
014  37 DMS
015  22 INV
016  37 DMS
017  57 FIX
018  00  0
019  37 DMS
020  57 FIX
021  02  2
022  65  x
023  43 RCL
024  00  0
025  02  2
026  55  ÷
027  03  3
028  06  6
029  05  5
030  95  =
031  44 SUM
032  00  0
033  01  1
034  43 RCL
035  00  0
036  01  1
037  81 HLT
038  46 LBL
039  52 EE
040  58 DSZ
041  17 B'
042  00  0
043  20 1/X
044  00  0
045  81 HLT
046  46 LBL
047  11  A
048  57 FIX
049  02  2
050  42 STO
051  00  0
052  01  1
053  81 HLT
054  46 LBL
055  15  B
056  55  ÷
057  01  1
058  00  0
059  00  0
060  95  =
061  42 STO
062  00  0
063  02  2
064  00  0
065  81 HLT
066  46 LBL
067  15  E
068  44 SUM
069  00  0
070  01  1
071  43 RCL
072  00  0
073  01  1
074  81 HLT
```

And finally here is Tom Wysmuller's program with all the bells and whistles, for TI-59/PC100 use only. Obviously, we don't need the decimal point trick here, as we simply enter the number of days by means of key D and the printer prints out a list of that many days and the corresponding amounts + interest.
Tom used direct addresses to speed up program execution. So don't make changes that will upset the correct program flow.

```
000  76 LBL
001  16 A'
002  43 RCL
003  01  01
004  59 INT
005  65  x
006  43 RCL
007  02  02
008  55  ÷
009  03  3
010  06  6
011  05  5
012  95  =
013  44 SUM
014  01  01
015  69 OP
016  24  24
017  92 RTN
018  76 LBL
019  14  D
020  42 STO
021  03  03
022  00  0
023  42 STO
024  04  04
025  16 A'
026  43 RCL
027  04  04
028  71 SBR
029  00  00
030  78  78
031  43 RCL
032  01  01
033  69 OP
034  06  06
035  97 DSZ
036  03  03
037  00  00
038  25  25
039  98 ADV
040  91 R/S
041  76 LBL
042  11  A
043  58 FIX
044  02  02
045  42 STO
046  01  01
047  01  1
048  03  3
049  03  3
050  00  0
051  03  3
052  07  7
053  69 OP
054  04  04
055  43 RCL
056  01  01
057  69 OP
058  06  06
059  91 R/S
060  76 LBL
061  12  B
062  32 X:T
063  06  6
064  01  1
065  69 OP
066  04  04
067  32 X:T
068  69 OP
069  06  06
070  55  ÷
071  01  1
072  00  0
073  00  0
074  95  =
075  42 STO
076  02  02
077  91 R/S
078  29 CP
079  67  EQ
080  01  01
081  04  04
082  55  ÷
083  28 LOG
084  59 INT
085  42 STO
086  08  08
087  69 OP
088  28  28
089  22 INV
090  28 LOG
091  85  +
092  32 X:T
093  01  1
094  00  0
095  00  0
096  49 PRD
097  09  09
098  08  8
099  77  GE
100  01  01
101  04  04
102  02  2
103  85  +
104  01  1
105  75  -
106  59 INT
107  44 SUM
108  09  09
109  95  =
110  65  x
111  01  1
112  00  0
113  97 DSZ
114  08  08
115  00  00
116  91  91
117  25 CLR
118  48 EXC
119  09  09
120  69 OP
121  04  04
122  92 RTN
123  76 LBL
124  15  E
125  32 X:T
126  01  1
127  03  3
128  01  1
129  06  6
130  01  1
131  06  6
132  69 OP
133  04  04
134  32 X:T
135  44 SUM
136  01  01
137  69 OP
138  06  06
139  43 RCL
140  01  01
141  11  A

001  16 A'
019  14  D
042  11  A
061  12  B
124  15  E
```

Maurice E.T. Swinnen.

• BOWLING BY THE SNOW BROTHERS.•

THIS PROGRAM, RATHER THAN JUST KEEP SCORE, ALLOWS TWO PLAYERS A VERY REALISTIC GAME OF BOWLING. THE PLAYERS ARE ABLE NOT ONLY TO DETERMINE THE DIRECTION OF THE THROW, BUT ALSO THE TYPE OF DELIVERY, SUCH AS STRAIGHTS, BACKUPS, CURVES AND EVEN HOOKS. THE GAME SHOULD BE PLAYED ON THE TI-59 WITH THE PC100 PRINTER ATTACHED. AFTER EACH THROW THE RESULTS ARE PRINTED IN THE FORM OF A DRAWING, CORRESPONDING TO THE PINS KNOCKED DOWN AND THOSE STILL LEFT STANDING. HERE ARE THE INSTRUCTIONS:

I. INITIALIZE: ENTER ANY NUMBER AS A SEED AND PRESS 2ND E'.

II.FIRST BALL:

1. DIRECTION OF BALL, PRESS A THROUGH E, AS SHOWN BELOW. (A IS ON THE LEFT-MOST AND E ON THE RIGHT-MOST SIDE OF THE LANE.)
2. TYPE OF DELIVERY: PRESS A THROUGH D.
   A IS STRAIGHT.)
   B IS BACKUP.) FOR THOSE UNFAMILIAR WITH THESE TERMS,
   C IS CURVE.) SEE DRAWING BELOW.
   D IS HOOK.)

3. RESULTS ARE PRINTED:
   STRIKE X.
   PINS THAT ARE LEFT STANDING.
   GUTTER 0.
   FOUL
   THE NUMBER OF PINS KNOCKED DOWN IS DISPLAYED.

III. SECOND BALL:

1. DIRECTION OF BALL, A THROUGH E.
2. RESULTS ARE PRINTED, SPARE /
   PINS THAT ARE LEFT
   GUTTER 0
   FOUL
   THE NEXT BOWLER'S NAME AND SCORE.

IV. GO TO II FOR SECOND BOWLER.

V. 10TH FRAME:

1.ONE EXTRA BALL IS GIVEN FOR A SPARE.
2.TWO EXTRA BALLS ARE GIVEN FOR A STRIKE.

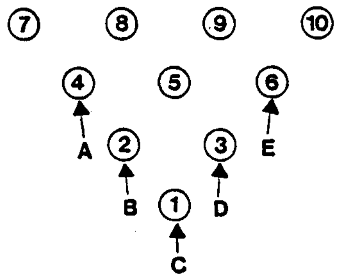
VI. NEW GAME:

1. STARTS AUTOMATICALLY AFTER LAST GAME, OR PRESS 2ND E'.

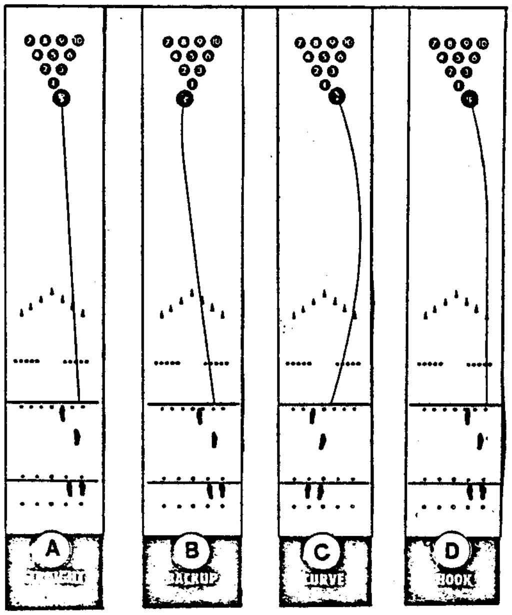
NOTE: THE FRAME NUMBER IS IN THE DISPLAY.

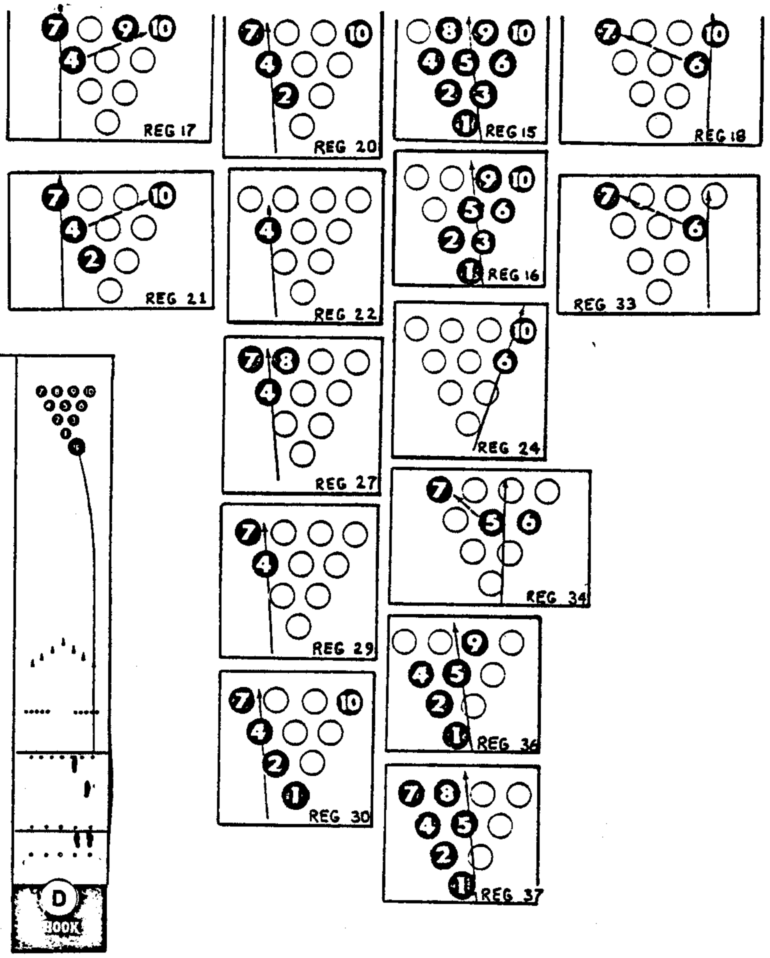
FOR THOSE INTERESTED IN THE PROGRAMMING ITSELF, THE DRAWING ON THE RIGHT SIDE GIVES THE STRATEGIES BUILT INTO THE PROGRAM. THE SEVEN REGISTERS ON THE LEFT CONTAIN THE STRATEGIES FOR GETTING LEFT-SIDE SPARES, WHILE THE SEVEN ON THE RIGHT DETERMINE HOW TO CONVERT RIGHT-HAND SPARES.

BALL DIRECTION



BOWLING DELIVERIES





B O W L I N G   BY ROBERT AND RICHARD SNOW

```
000  83 GO*
001  45  45
002  76 LBL
003  17 B'
004  69 OP
005  03  03
006  69 OP
007  05  05
008  00  0
009  92 RTN
010  76 LBL
011  19 D'
012  97 DSZ
013  10  10
014  03  03
015  89  89
016  73 RC*
017  08  08
018  32 X:T
019  00  0
020  92 RTN
021  76 LBL
022  18 C'
023  04  4
024  42 STO
025  19  19
026  58 FIX
027  04  04
028  16 A'
029  03  3
030  42 STO
031  19  19
032  16 A'
033  02  2
034  42 STO
035  19  19
036  58 FIX
037  04  04
038  03  3
039  16 A'
040  02  2
041  76 LBL
042  16 A'
043  42 STO
044  04  04
045  82 HIR
046  13  13
047  22 INV
048  59 INT
049  65  ×
050  01  1
051  00  0
052  95  =
053  82 HIR
054  03  03
055  59 INT
056  22 INV
057  44 SUM
058  00  00
059  65  ×
060  03  3
061  04  4
062  85  +
063  04  4
064  00  0
065  95  =
```

```
066  84 OP*
067  04  04
068  69 OP
069  34  34
070  97 DSZ
071  19  19
072  00  00
073  45  45
074  69 OP
075  05  05
076  69 OP
077  00  00
078  22 INV
079  58 FIX
080  92 RTN
081  76 LBL
082  88 DMS
083  44 SUM
084  09  09
085  22 INV
086  59 INT
087  65  ×
088  01  1
089  00  0
090  95  =
091  82 HIR
092  44  44
093  04  4
094  32 X:T
095  82 HIR
096  14  14
097  59 INT
098  82 HIR
099  54  54
100  22 INV
101  77  GE
102  01  01
103  05  05
104  02  2
105  44 SUM
106  09  09
107  73 RC*
108  09  09
109  82 HIR
110  03  03
111  42 STO
112  09  09
113  32 X:T
114  00  0
115  67  EQ
116  02  02
117  17  17
118  77  GE
119  03  03
120  20  20
121  09  9
122  09  9
123  77  GE
124  03  03
125  12  12
126  43 RCL
127  38  38
128  61 GTO
129  03  03
130  22  22
131  76 LBL
```

```
132  15  E
133  01  1
134  85  +
135  76 LBL
136  14  D
137  08  8
138  93  .
139  06  6
140  22 INV
141  77  GE
142  88 DMS
143  01  1
144  85  +
145  76 LBL
146  13  C
147  01  1
148  05  5
149  93  .
150  07  7
151  22 INV
152  77  GE
153  88 DMS
154  01  1
155  85  +
156  76 LBL
157  12  B
158  05  5
159  93  .
160  06  6
161  22 INV
162  77  GE
163  88 DMS
164  01  1
165  85  +
166  76 LBL
167  11  A
168  93  .
169  08  8
170  22 INV
171  77  GE
172  88 DMS
173  01  1
174  05  5
175  95  =
176  44 SUM
177  09  09
178  07  7
179  87 IFF
180  01  01
181  00  00
182  91  91
183  09  9
184  09  9
185  32 X:T
186  91 R/S
187  76 LBL
188  10 E'
189  85  +
190  89  π
191  95  =
192  38 SIN
193  50 I×I
194  82 HIR
195  04  04
196  43 RCL
197  14  14
```

```
198  42 STO
199  45  45
200  09  9
201  35 1/X
202  82 HIR
203  03  03
204  18 C'
205  01  1
206  69 OP
207  17  17
208  47 CMS
209  07  7
210  69 OP
211  17  17
212  02  2
213  01  1
214  42 STO
215  10  10
216  81 RST
217  87 IFF
218  02  02
219  02  02
220  98  98
221  87 IFF
222  01  01
223  02  02
224  92  92
225  93  .
226  00  0
227  04  4
228  74 SM*
229  08  08
230  43 RCL
231  26  26
232  69 OP
233  02  02
234  43 RCL
235  31  31
236  17 B'
237  86 STF
238  01  01
239  01  1
240  00  0
241  42 STO
242  00  00
243  73 RC*
244  08  08
245  22 INV
246  59 INT
247  63 EX*
248  08  08
249  32 X:T
250  01  1
251  00  0
252  64 PD*
253  08  08
254  04  4
255  93  .
256  03  3
257  77  GE
258  02  02
259  85  85
260  01  1
261  00  0
262  85  +
263  43 RCL
```

```
264  00  00
265  95  =
266  74 SM*
267  07  07
268  43 RCL
269  00  00
270  87 IFF
271  01  01
272  02  02
273  78  78
274  86 STF
275  01  01
276  29 CP
277  91 R/S
278  87 IFF
279  02  02
280  02  02
281  84  84
282  74 SM*
283  07  07
284  81 RST
285  03  3
286  77  GE
287  03  03
288  05  05
289  61 GTO
290  02  02
291  63  63
292  93  .
293  02  2
294  74 SM*
295  08  08
296  43 RCL
297  52  52
298  69 OP
299  02  02
300  43 RCL
301  58  58
302  61 GTO
303  02  02
304  36  36
305  01  1
306  77  GE
307  02  02
308  68  68
309  61 GTO
310  02  02
311  63  63
312  01  1
313  00  0
314  42 STO
315  00  00
316  18 C'
317  61 GTO
318  02  02
319  43  43
320  43 RCL
321  63  63
322  69 OP
323  02  02
324  00  0
325  17 B'
326  42 STO
327  09  09
328  61 GTO
329  02  02
```

```
330  43  43
331  43 RCL
332  62  62
333  42 STO
334  45  45
335  02  2
336  42 STO
337  07  07
338  06  6
339  42 STO
340  08  08
341  43 RCL
342  32  32
343  69 OP
344  04  04
345  73 RC*
346  07  07
347  69 OP
348  06  06
349  43 RCL
350  39  39
351  19 D'
352  67  EQ
353  10 E'
354  61 GTO
355  03  03
356  79  79
357  69 OP
358  23  23
359  01  1
360  42 STO
361  07  07
362  05  5
363  42 STO
364  08  08
365  43 RCL
366  25  25
367  69 OP
368  04  04
369  73 RC*
370  07  07
371  69 OP
372  06  06
373  43 RCL
374  62  62
375  19 D'
376  67  EQ
377  03  03
378  31  31
379  86 STF
380  02  02
381  01  1
382  22 INV
383  77  GE
384  02  02
385  68  68
386  03  3
387  05  5
388  09  9
389  42 STO
390  45  45
391  00  0
392  42 STO
393  09  09
394  42 STO
395  00  00
396  29 CP
397  43 RCL
398  03  03
399  91 R/S
```

1. KEY IN PROGRAM AND LOAD REGISTERS WITH 7 OP 17 PARTITION.

2. RECORD 4 BANKS WITH 6 OP 17.

ALPHA & CONSTANTS REGISTERS

```
 47.1110111111     15
 47.1100110111     16
 36.1101001        17
 29.10011          18
 37.100100101      20
 36.100100101      21
 43.0000001        22
 23.10001          24
-143214.           25
       BOB
 36373524.         26
      STRI
 43.0011001        27
 43.0001001        29
 37.1001001011     30
 2617004400.       31
      KE X
-16241526.         32
      DICK
 30.00011          33
 30.000111         34
 47.0100011011     36
 47.0011011011     37
-2241373735.       38
      GUTTR
 357.              39
 0.10001           40
 0.10001           41
 .00001            42
 .0001             44
 335.              45
 .00001            46
 .00011            47
 .00011            48
 .0001             49
 .0001             51
 3633133517.       52
      SPARE
 0.1000001         53
 0.1               55
 0.1001001         56
 .0001             57
 6300371731.       58
      / TEN
 .0000001          59
 .0000001          61
 335.              62
-21324127.         63
      FOUL
 .000000001        64
 .0000001          65
 .000010011        67
 .0000000012       68
 .000000011        69
```