* TI PPC NOTES * V6N3, 1981

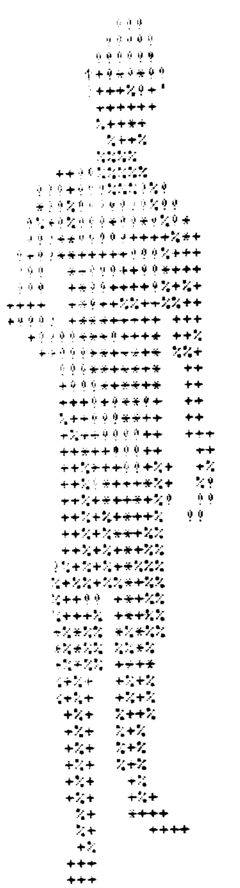
NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB.

9213 Lanham Severn Road, Lanham MD, 20801, USA.

Since a couple of months I have been experimenting with an exciting discovery by Michael Sperber from Fuerth, West Germany: a high-resolution plotting method on the PC100A,C,B,D. It is rather slow, but gives about five times better resolution than the OP 07 method. I am waiting now for permission to publish and will probably do so in the next issue. So, be on the look out for the new FINE PLOT.

I have been busy lately teaching two-day seminars for TI. That requires carrying two TI-59/PC100A's on airplanes, in taxis, etc. I have been sporting them each one in its own attractive carrying case. I must admit, they were sent to me as samples. One is specifically manufactured for the TI-58/59/PC100A and offers ideal protection and portability. It is sewn of soft leather and leather-like vinyl and has both a hand grip and a shoulder strap.Sponge-foam is used to protect the unit from possible damage. It also has a key lock. The second one is an American Tourister that has been costum fitted with sponge foam to hold the calculator-printer. Although, in my humble opinion, it is not as attractive as the soft-leather case, it has more inside room, so that about 20 card holder booklets fit in it too. The second case is hard formed vinyl and has accordian type filing pockets inside the lid. It also has a combination digital lock. The manufacturers, System-7 Inc., 2315 50th Street, Suite F, Lubbock, TX, 79412, normally sell only to large quantity buyers. However, they have been persuaded to make an exception for the members of our club. So, if you want one of these carrying cases, write to the above address. The soft one costs $ 55.00 and the hard one is priced at $ 80.00. I checked at the local department store and found the same hard case, by American Tourister, non-costumized, at $ 85.00!

BEAMING ABOARD



SPOCK

John Worthington and Emil Regelman have written another super program in the series: Zeros of Functions. This bisection method is slower than their fast Secant method in v5n6p6&7. But it generally will find all the roots within the search limit. It is presented on pages 11 and 12.

The newcomer's corner is back again, to pacify those who threaten to quit the club if we don't reinstitute it. I'll do my utmost best to write one each time in the future.

I especially draw your attention to the new quirk discovered by Frédéric De Mees. You can read all about it on page 6. Does anybody have an explanation for it ?

The nice picture on the left was just an idle doodle. No program exists for it, so if you care, write one and I'll publish it.

The HP PPC Journal Eastern conference on March 28 in Rockville MD was exciting. Frank Blachly and yours truly defended successfully the colors of our club, in a round-table discussion about "What should future hand-helds look like, beyond the HP-41C and the TI-88 !!!!"Yes, they assured me that TI will come out with a calculator by that name. No confirmation from TI, though.

Maurice E.T. Swinnen

IN THIS ISSUE:

BUBBLE MEMORY.- Bubble memory techniques have not yet been used for programmable calculators. But I would not be surprised if one day we will see this technique employed to give us mass memory capability in a very small space.

Bubble memory technology is really coming into its own, and more and more bubble memory devices will be available in the near future. Here is a quick look at how a bubble memory works:

Thin films of certain magnetic materials, i.e. layers of magnetic garnet artificially grown on a non-magnetic garnet substrate, contain randomly shaped domains. When a magnetic bias field is applied by two permanent magnets placed on either side of the thin film, these randomly shaped domains shrink into "bubbles", actually cylindrical magnetic domains of fixed volume, the polarization of which is opposite to that of the thin film.

If the polarization of the film is "south", then the bubbles would be like floating islands of "north" in a sea of "south." These bubbles can be seen under great magnification in polarized light as contrasts within the film.

Magnetic bubbles are stable over a wide range of conditions, and they can be moved from point to point at very high speeds. When the bubbles "move", it is not matter that moves, but rather the very rapid transfer of magnetic properties of the crystaline elements of the garnet.

To guide and control the movement of the bubbles, a permalloy pattern of chevrons is applied to the surface of the film to form "paths". When, in addition to the bias magnetic field, a rotating magnetic field is applied by two coils that are part of the magnetic bubble device, the bubbles can move at extremely high speed along these paths.

The permalloy paths are formed as loops. The presence of a bubble at a certain position on the loop corresponds to a "ONE bit", the absence of a bubble to a "ZERO bit." A "block" consists of bubbles in the same relative position in each loop.

The output circuitry on the bubble memory device includes a detector that exactly transforms the bubble to be read into an electronic pulse.The "read" operation preserves the bubbles as they are on the device, making non-destructive readout a prominent feature of the bubble memories.A different kind of output can be used to transfer or erase the bubbles.

Data is entered by generators at the other end of the chip. Bit information is transferred at gates, thus forming bubbles at known positions in the loop.

CALCULATOR CLOUT, by Maurice D. Weir, Naval Postgraduate School, Monterey CA, 93940. 256 pp., illus., 7x9¼ ", (April 1981) $ 17.95, A Spectrum Book, available from Prentice-Hall Inc., Englewood Cliffs, N.J. 07632, USA. Order the book under # 11041-1.

TI-59 programs to solve more complex mathematical problems than ever before. Maurice is a TI PPC Club member and a mathematician and excellent programmer. The book contains easy-to-follow applied examples in precalculus and business math, elementary numerical methods of basic calculus.

I highly recommend it to beginner as well as to more advanced programmers.

CALCULATORS QUICKLY FIND TIER I REVENUE, VOLUME, WPT, by Frank W. Lewis and Dipak K. Sinha, Oil & Gas Journal, March 16, 1981, pp80-84. The Crude Oil Windfall Profit Tax Act of 1980 is a complex piece of legislation causing independent producers and royalty owners difficulty in calculating Tier I production revenues, volume and windfall profit taxes. This program for a TI-59 will do all of that easily and fast. Well documented with a completely worked out sample problem. Frank is a TI PPC Club member and is with the Resource Analysis and Management Group in Oklahoma City, Oklahoma. Mr. Sinha is with the Oklahoma Department of Energy, Oklahoma City, Oklahoma.

DAY OF THE WEEK.- Robert H. Baker, from Humboldt, Iowa, thinks that Bill Beebe's program in v6n2p2 could run a little faster if you substitute for his present steps 004, 005 and 006 the following sequence:

PGM 20 SBR 174

One-Card-Side Auto Record Book

Bob Patton

The idea is to keep the program and and all basic costs and usage figures for one automobile on one side of a magnetic card.

The following items are stored for the current month and cumulative for the year: Miles, Gallons Gas$, Oil$, Other$. Three basic calculations are made via user keys: Average miles/gallon, Average ¢/gallon, Total dollars.

You enter raw data using keys A - E. After each entry you see the current total for that item. Pressing any key A to E without data entry also gives the current total (using the decimal point trick, V5N3P7).

An example best shows how to use the program.

Given the following data:

Date	Miles	Gallons	Gas$	Oil$	Other$
12-30	32835	(last fill-up of previous month)			
1- 7					3.83
1-13	33125	12.3	11.04		
1-23	33450	10.7	11.00		

You would handle it like this:

```
Load card.
Initialize:      RTN R/S R/S --> 159.99
Start new year: 2nd CMs
32835  A   -->  32835
33450  A   -->    615    Jan. miles
 12.3  B   -->   12.3
 10.7  B   -->   23.0    Jan. gallons
11.04  C   -->  11.04
   11  C   -->  22.04    Jan. gas$
 3.83  E   -->   3.83    Jan. other$
  2nd  A   -->   26.7    Jan. mpg
  2nd  B   -->   95.8    Jan. ¢/gallon
  2nd  C   -->  25.87    Jan. total$
  2nd  E   -->   feed in card to save January
                 data in cumulative registers.
```

After the end of February you might have further data:

Date	Miles	Gallons	Gas$	Oil$	Other$
2- 1	33603	5.2	5.60	.62	
2-15	33911	10.8	11.60		
2-25	34214	10.0	10.90	.62	
2-28	34512	7.9	8.62		

```
Load card.
Initialize:  RTN R/S R/S  -->  159.99
33450  A  -->  33450
34512  A  -->   1062  Feb. miles
  5.2  B  -->    5.2
 10.8  B  -->   16.0
   10  B  -->   26.0
  7.9  B  -->   33.9  Feb. gallons
  5.6  C  -->   5.60
 11.6  C  -->  17.20
 10.9  C  -->  28.10
 8.62  C  -->  36.72  Feb. gas$
  .62  D  -->    .62
  .62  D  -->   1.24  Feb. oil$
  2nd  A  -->   31.3  Feb. mpg
  2nd  B  -->  108.3  Feb. ¢/gallon
  2nd  D                     add Feb. to Jan.
  2nd  A  -->   31.3  combined mpg
  2nd  B  -->  103.3  average ¢/gallon
       A  -->   1677  total miles
     x:t              save them
  2nd  C  -->  63.88  total $
div x:t X 100 = 3.81  operating cost, ¢/mile
                      for year to date
  2nd  E              feed in card and save
                      year-to-data data.
```

To enter program:
Partition to 159.99 by 10 Op 17.
Key in program.
Partition to 479.59 by 6 Op 17.
Record Program.

Suggested card labelling:

```
|(-Ford     AUTO RECORD BOOK       Chev->|
| Init = RST R/S R/S   New year = *CMs  |
-----------------------------------------
|  mpg  | ¢/gal |total$ |recall | store |
|       |       |       | cum   | data  |
-----------------------------------------
| MILES | GAL.s | GAS$  | OIL$  |OTHER$ |
```

```
000 91 R/S     033 02  02     066 43 RCL     099 76 LBL
001 01  1      034 44 SUM     067 91  91     100 19 D'
002 00  0      035 92  92     068 95  =      101 87 IFF
003 69 OP      036 43 RCL     069 91 R/S     102 00  00
004 17  17     037 92  92     070 76 LBL     103 00  00
005 91 R/S     038 91 R/S     071 17 B'      104 00  00
006 76 LBL     039 76 LBL     072 58 FIX     105 86 STF
007 11  A      040 14  D      073 01  01     106 00  00
008 93  .      041 93  .      074 43 RCL     107 09  9
009 58 FIX     042 58 FIX     075 92  92     108 05  5
010 00  00     043 02  02     076 65  x      109 42 STO
011 75  -      044 44 SUM     077 01  1      110 01  01
012 43 RCL     045 93  93     078 00  0      111 09  9
013 90  90     046 43 RCL     079 00  0      112 00  0
014 95  =      047 93  93     080 55  ÷      113 76 LBL
015 50 I×I     048 91 R/S     081 43 RCL     114 42 STO
016 42 STO     049 76 LBL     082 91  91     115 42 STO
017 90  90     050 15  E      083 95  =      116 02  02
018 91 R/S     051 93  .      084 91 R/S     117 05  5
019 76 LBL     052 58 FIX     085 76 LBL     118 42 STO
020 12  B      053 02  02     086 18 C'      119 00  00
021 93  .      054 44 SUM     087 58 FIX     120 76 LBL
022 58 FIX     055 94  94     088 02  02     121 48 EXC
023 01  01     056 43 RCL     089 43 RCL     122 25 CLR
024 44 SUM     057 94  94     090 92  92     123 63 EX*
025 91  91     058 91 R/S     091 85  +      124 01  01
026 43 RCL     059 76 LBL     092 43 RCL     125 74 SM*
027 91  91     060 16 A'      093 93  93     126 02  02
028 91 R/S     061 58 FIX     094 85  +      127 69 OP
029 76 LBL     062 01  01     095 43 RCL     128 21  21
030 13  C      063 43 RCL     096 94  94     129 69 OP
031 93  .      064 90  90     097 95  =      130 22  22
032 58 FIX     065 55  ÷      098 91 R/S     131 97 DSZ
                                             132 00  00
                                             133 48 EXC
                                             134 25 CLR
                                             135 92 RTN
                                             136 76 LBL
                                             137 10 E'
                                             138 09  9
                                             139 00  0
                                             140 42 STO
                                             141 01  01
                                             142 09  9
                                             143 05  5
                                             144 71 SBR
                                             145 42 STO
                                             146 22 INV
                                             147 58 FIX
                                             148 06  6
                                             149 69 OP
                                             150 17  17
                                             151 01  1
                                             152 96 WRT
                                             153 81 RST
```

MAGIC SQUARES.- Dejan Ristanović, Belgrade, Yugoslavia wrote this program. The idea behind magic squares is that you write numbers in a square consisting of an even number of rows and columns, such that the sum in any column, any row, any diagonal is always the same.

Vol. 2, no. 5, pp 5 & 6 of the PPC EXChange, SEPT 1978 has a program for each the TI-59 and the SR-52 for a 4 by 4 magic square. Bill Skillman made it more attractive by actually printing a 4 X 4 square, rather than simply outputting a series of numbers, as the PPX programs do. Bill's program follows Dejan's.

Dejan's unique contribution to the problem is, that he produced a fast program that allows any N up to 19. Of course, it does not print squares, as Bill's does, but it gives you all the numbers to be put in the magic square of your choice.

Instructions: Remove the ML-module and put in the M/U module.
Key in or read in the program.
Enter N and press A. Numbers will be printed.
Write the numbers according to the rows and columns in a square.

17.	000 76 LBL	026 82 HIR	051 32 X:T	076 22 INV	101 55 55	126 61 GTO
24.	001 11 A	027 04 04	052 82 HIR	077 77 GE	102 29 CP	127 00 00
1.	002 32 X:T	028 22 INV	053 15 15	078 01 01	103 82 HIR	128 49 49
8.	003 01 1	029 59 INT	054 75 -	079 29 29	104 15 15	129 43 RCL
15.	004 00 0	030 29 CP	055 01 1	080 43 RCL	105 67 EQ	130 99 99
23.	005 69 OP	031 22 INV	056 95 =	081 99 99	106 01 01	131 49 PRD
5.	006 17 17	032 67 EQ	057 65 ×	082 32 X:T	107 22 22	132 99 99
7.	007 01 1	033 25 CLR	058 43 RCL	083 82 HIR	108 82 HIR	133 01 1
14.	008 09 9	034 04 4	059 99 99	084 17 17	109 14 14	134 82 HIR
16.	009 77 GE	035 93 .	060 85 +	085 22 INV	110 32 X:T	135 07 07
4.	010 00 00	036 03 3	061 82 HIR	086 77 GE	111 43 RCL	136 82 HIR
6.	011 17 17	037 03 3	062 14 14	087 00 00	112 99 99	137 17 17
13.	012 76 LBL	038 03 3	063 95 =	088 95 95	113 77 GE	138 36 PGM
20.	013 25 CLR	039 03 3	064 36 PGM	089 01 1	114 00 00	139 08 08
22.	014 25 CLR	040 42 STO	065 08 08	090 82 HIR	115 49 49	140 13 C
10.	015 35 1/X	041 01 01	066 12 B	091 35 35	116 01 1	141 99 PRT
12.	016 91 R/S	042 01 1	067 01 1	092 61 GTO	117 82 HIR	142 01 1
19.	017 32 X:T	043 82 HIR	068 82 HIR	093 00 00	118 04 04	143 82 HIR
21.	018 42 STO	044 05 05	069 36 36	094 47 47	119 61 GTO	144 37 37
3.	019 99 99	045 82 HIR	070 82 HIR	095 01 1	120 00 00	145 97 DSZ
11.	020 85 +	046 06 06	071 16 16	096 82 HIR	121 49 49	146 99 99
18.	021 01 1	047 82 HIR	072 32 X:T	097 37 37	122 43 RCL	147 01 01
25.	022 95 =	048 07 07	073 43 RCL	098 82 HIR	123 99 99	148 36 36
2.	023 55 ÷	049 82 HIR	074 99 99	099 34 34	124 82 HIR	149 61 GTO
9.	024 02 2	050 16 16	075 33 X²	100 82 HIR	125 05 05	150 25 CLR
	025 95 =					

4 X 4 MAGIC SQUARE.- Bill Skillman, Linthicum Heights (Baltimore) Maryland. As opposed to Dejan's program, above, this program allows you to enter X, Y, Z and W. The relation of these entries to the final output is given in the table below:

34 X+Y+Z-2W	33 X-Y-Z	32 X-Y-Z+W	31 X+Y+Z+W
38 X-Y+Z+W	37 X+Y-Z+W	36 X+Y-Z	35 X-Y+Z-2W
42 X+Y-Z+2W	41 X-Y+Z	40 X-Y+Z-W	39 X+Y-Z-W
46 X-Y-Z-W	45 X+Y+Z-W	44 X+Y+Z	43 X-Y-Z+2W

```
3125.      X
2589.      Y
2698.      Z
3999.      W

 414-2162 1837 ****
7233 7015 3016-4764
**** 3234 -765 -983
-6161 4413 8412 5836
```

In this magic square, the sum of any row, of any column, of any diagonal is the same. Also the sum of the four center cells or of the four corner cells is the same. The two inner rows and the tow outer rows may be interchanged and so may be the two inner columns and the two outer columns, with the sum still staying the same.

The entries or variables may be positive or negative, but should best be integers, as the program prints only the integer part of the results. The program does not protect against fractions, which may work or may give as result "****". Neither does the program protect against too large entries.

In the table, above, the numbers 31 through 46 are the registers in which that particular sum is stored.

Instructions are: Enter X, press A. Enter Y, press B, enter Z, press C, enter W, press D. See example in left top corner. Printer will print the matrix. A too large sum will result in "****".

SEE PROGRAM NEXT PAGE.

4 X 4 MAGIC SQUARES - Bill Skillman. Listing.

```
000 76 LBL
001 11 A
002 42 STO
003 01 01
004 32 X:T
005 04 4
006 04 4
007 76 LBL
008 10 E'
009 69 OP
010 04 04
011 32 X:T
012 69 OP
013 06 06
014 92 RTN
015 76 LBL
016 12 B
017 42 STO
018 02 02
019 32 X:T
020 04 4
021 05 5
022 61 GTO
023 10 E'
024 76 LBL
025 17 B'
026 73 RC*
027 06 06
028 85 +
029 43 RCL
030 01 01
031 95 =
032 69 OP
033 26 26
034 92 RTN
035 76 LBL
036 13 C
037 32 X:T
038 04 4
039 06 6
040 10 E'
041 85 +
042 43 RCL
043 02 02
044 95 =
045 42 STO
046 31 31
047 42 STO
048 34 34
049 42 STO
050 44 44
051 42 STO
052 45 45
053 94 +/-
054 42 STO
055 32 32
056 42 STO
057 33 33
058 42 STO
059 43 43
060 42 STO
061 46 46
062 85 +
063 02 2
064 65 ×
065 43 RCL
066 02 02
067 95 =
068 42 STO
069 36 36
070 42 STO
071 37 37
072 42 STO
073 39 39
074 42 STO
075 42 42
076 94 +/-
077 42 STO
078 35 35
079 42 STO
080 38 38
081 42 STO
082 40 40
083 42 STO
084 41 41
085 92 RTN
086 76 LBL
087 14 D
088 32 X:T
089 04 4
090 03 3
091 10 E'
092 44 SUM
093 32 32
094 44 SUM
095 37 37
096 44 SUM
097 38 38
098 44 SUM
099 31 31
100 44 SUM
101 43 43
102 44 SUM
103 43 43
104 44 SUM
105 42 42
106 44 SUM
107 42 42
108 94 +/-
109 44 SUM
110 39 39
111 44 SUM
112 35 35
113 44 SUM
114 35 35
115 44 SUM
116 40 40
117 44 SUM
118 45 45
119 44 SUM
120 46 46
121 44 SUM
122 34 34
123 44 SUM
124 34 34
125 98 ADV
126 04 4
127 42 STO
128 08 08
129 03 3
130 01 1
131 42 STO
132 06 06
133 04 4
134 42 STO
135 05 05
136 17 B'
137 71 SBR
138 01 01
139 54 54
140 84 OP*
141 05 05
142 97 DSZ
143 05 05
144 01 01
145 36 36
146 69 OP
147 05 05
148 97 DSZ
149 08 08
150 01 01
151 33 33
152 98 ADV
153 92 RTN
154 32 X:T
155 00 0
156 67 EQ
157 01 01
158 99 99
159 22 INV
160 77 GE
161 01 01
162 64 64
163 02 2
164 00 0
165 42 STO
166 09 09
167 32 X:T
168 50 I×I
169 55 ÷
170 28 LOG
171 59 INT
172 42 STO
173 00 00
174 32 X:T
175 03 3
176 22 INV
177 77 GE
178 02 02
179 16 16
180 43 RCL
181 00 00
182 69 OP
183 20 20
184 22 INV
185 28 LOG
186 85 +
187 32 X:T
188 01 1
189 00 0
190 00 0
191 49 PRD
192 09 09
193 08 8
194 77 GE
195 01 01
196 99 99
197 02 2
198 85 +
199 01 1
200 75 -
201 59 INT
202 44 SUM
203 09 09
204 95 =
205 65 ×
206 01 1
207 00 0
208 97 DSZ
209 00 00
210 01 01
211 86 86
212 25 CLR
213 48 EXC
214 09 09
215 92 RTN
216 25 CLR
217 05 5
218 01 1
219 05 5
220 01 1
221 05 5
222 01 1
223 05 5
224 01 1
225 92 RTN
```

TALLEY SHEET.- Dick Blayney wrote the original program and it was later a little enhanced by yours truly. I made it a little more user-friendly by adding the automatic record feature, the automatic list provision and the printing of a dotted line. The extra steps required seemed to me worthwhile.

The program allows to talley two categories, 49 classes. For example, you may have a maximum of 49 costumers. In that case the register number will be your costumer number and you may store for each costumer the number of checks entered and the total amount entered. Other uses are many and I leave it to the inventive user to discover a few more.

Instructions: Key in the program in normal turn-on partition and record one card side only in that same partition.

To use the program for the first time, press CLR and insert the one card side only.
ALWAYS INITIALIZE WITH 2nd A'. A dotted line will be printed, as a confirmation.

1. Enter costumer number and press A.
2. Enter amount of the present check from this costumer and press R/S.
3. Enter new costumer number and press A. Enter amount and press R/S.
   The entry may be in any order, as long as it is always followed by an amount.
4. When all required amounts are entered, press E.
   The first thing you will see is a listing of the number of entries per costumer, with the costumer number on the far right.
   Then you will see a listing of the total amount for each costumer, with the costumer number on the far right, from which you should subtract exactly 50, to correspond to the same costumer number in the first listing.

Next the printer will print a "1", asking for card side 1. Put that side in the slot. Next the printer asks for side 2, by printing a 2, etc. up to side 4.
That was the automatic WRITE, or recording of all your data on three card sides.
To confirm again that everything is done, the printer will print again a dotted line.

When you read in your four card sides, next day, read them in the usual manner, 1, enter a side, 2, enter a side, etc. up to 4.

Note: Costumer numbers may range from 1 through 49 only.

```
000 76 LBL
001 11 A
002 42 STO
003 00 00
004 01 1
005 74 SM*
006 00 00
007 05 5
008 00 0
009 48 EXC
010 00 00
011 44 SUM
012 00 00
013 95 =
014 91 R/S
015 74 SM*
016 00 00
017 91 R/S
018 76 LBL
019 15 E
020 05 5
021 69 OP
022 17 17
023 01 1
024 22 INV
025 90 LST
026 98 ADV
027 01 1
028 00 0
029 69 OP
030 17 17
031 05 5
032 01 1
033 22 INV
034 90 LST
035 06 6
036 69 OP
037 17 17
038 98 ADV
039 01 1
040 99 PRT
041 96 WRT
042 02 2
043 99 PRT
044 96 WRT
045 03 3
046 99 PRT
047 96 WRT
048 04 4
049 99 PRT
050 96 WRT
051 76 LBL
052 16 A'
053 01 1
054 00 0
055 69 OP
056 17 17
057 02 2
058 00 0
059 02 2
060 00 0
061 02 2
062 00 0
063 02 2
064 00 0
065 02 2
066 00 0
067 69 OP
068 01 01
069 69 OP
070 02 02
071 69 OP
072 03 03
073 69 OP
074 04 04
075 69 OP
076 05 05
077 69 OP
078 00 00
079 25 CLR
080 91 R/S
```

A NEW QUIRK.- Frédéric De Mees, Jumet, Belgium, has discovered this interesting bug: From turn-on, key in this sequence:

LRN 1 LRN PGM 17 SBR 1 PGM LRN A CLR

You are now in a curious "Fix 1" state. Enter pi^2 and see -10 in the display. But where in the display ? Press PRT and see what is printed!

At 000 in user memory, key in this short sequence to store something in HIR 08:

LBL A HIR 08 R/S

Then in calculator mode, enter 999121314 STO 00 .1516 sum 00 RCL 00 A

We now normally would have in HIR 8 the print code for the characters "9ABCD", with the three 9's added for proper left-justification. We can print this by simply pressing OP 05, and effectively get 9ABCD. But press OP 06 and everything goes haywire. Where are at least 4 characters ?

RST has no effect on this state, but any FIX N will restore normality.

FAST MODE.- Dave Leising has some thoughts on this subject:
" I believe what happens in the fast mode is that code in user memory is executed with the processor protocol invoked for execution of code out of the TMC0571 on board CROM (see TI-59 schematic) ("revealed microcode"; P-R, DMs, $\bar{X}$, etc.) The reason for the speed is that the time-outs for the keyboard scan are masked out. One should explore the behavior of the HIR 2X series in this mode."

REMAINDER ROUTINE.- Michael Sperber, Fuerth, West Germany, offers this version, which is four steps less than the one in v5n7p15:

LBL E' (X:T - (X:T DIV X:T) 1/X INT X X:T) RTN

Instructions are the same.

FRACTURED DIGITS.- Fractured digits may be produced with the Statistics Library module in place, according to Michael Sperber. Just call 10 OP 17 PGM 14 SBR 024 P-R LRN SST DEL DEL DEL DEL.....
In general, call a sequence such as PGM xx in a module, with PGM xx non-existent. PGM IND NN has the same effect when you store xx in register NN.
On a TI-59, 10 OP 17 causes the second sign to be a quote (") , whereas 9 OP 17 and 8 OP 17 cause it to be a zero. With 8 OP 17, use INV DMs instead of P-R.
On a TI-58, 5 OP 17 and 4 OP 17 cause a quote sign, and 3 OP 17 and 2 OP 17 produce a zero as second sign.
I think the second sign is a sort internal flag used to indicate whether a bank is defined as user memory, totally or partially. This information is needed when protected cards are read.

LOAN SCHEDULE.- Lambert Programming Service, 434 N.Crescent Heights Blvd., Los Angeles, CA, 90048 (a TI PPC Club member) has a Loan Schedule program in its catalog. It will print in about 3 min/year for the TI-59 or 1 min/year for the HP-41C a complete schedule at any interest rate, breaking down each payment by principal and interest amounts and showing remaining balance. Handles also balloon payments. The program is intended for printer output. Program contained on mag cards and extensive documentation. For the TI-59/PC100C the price is $ 60.00. For the HP-41C the price is $ 80.00. California residents, add tax and all others add $ 1.50 handling charges. Tel.: (213) 658-MATH.

RADIOLOGY.- The journal STRAHLENTHERAPIE, ZEITSCHRIFT FUER RADIOLOGIE UND ONKOLOGIE, 156 (1980), 272-274 (Nr 4), has an article called EIN MAGNETKARTEN-PROGRAMMIERBARER TASCHENRECHNER ALS BESTRAHLUNGSZEITTABELLE.(A Pocket Calculator programmed by Magnetic Cards used as Irradiation Time Table. Author is Dr. F. Krispel, University Clinic, Graz, Austria. Program is for the TI-59.

PRINT CODE CONVERTERS.- Since we are seriously thinking about producing our own utility module, it would be a good idea to publish a few more utility routines. One of the most useful ones is a good print code converter. I received a few more of those. Each one has its particular merit. So, until we decide which ones are ultimately going into the module, here are some more ideas:
Clyde Durbin, Dallas Texas, has this short and fast one for digits 0 through 9:

LBL PRT + 1 + LOG INT X 2 = RTN

Clyde also wrote one, a little longer, for the numbers 0 through 18, positive only.

LBL PRT - 9 + OP 10 X 95 + HIR 12 X^2 X 93 + 12 = RTN

And finally Clyde presents this one which takes 7 sec execution time for positive numbers between 0 and 99999. Note that in all Clyde's routines no t-register is used and only the fast DSZ transfer. So no EQ or GE comparisons.

000: LBL PRT + .2 = DIV LOG INT STO 01 OP 21 INV LOG + 100 PRD 02 1 +

023: LOG INT X 2 - INT SUM 02 = X 10 DSZ 1 015 CLR EXC 02 RTN

Michael Sperber (no not the one from PPX) from Fürth, West Germany, also wrote one that takes 7 sec execution time. It converts positive numbers between 0 and 99999.

000: LBL PRT DIV LOG INT COS INV COS STO 08 OP 28 INV LOG + 1 + LOG INT

019: X 100 PRD 09 2 - INT SUM 09 = X 1 DSZ 8 012 CLR EXC 09 RTN

And not wanting to be outdone by his compatriot, Stephan Böhm wrote this one. Note the almost carbon copy writing. The only deviation seems to be at step 033 and following. In fact, they both look a lot like Richard Snow's original converter. Their special attraction is the absence of GE and EQ comparisons and that they have only a fast DSZ.

000: LBL PRT DIV LOG INT COS INV COS STO 08 OP 28 INV LOG + 1 + LOG INT

019: X 100 PRD 09 2 - INT SUM 09 = X 10 DSZ 8 014 CLR EXC 09 RTN

Note to the newcomers: If you want to try out these routines, write this short short routine at the end of one of the above:

LBL A SBR PRT OP 02 OP 05 R/S

Enter the number you want to convert and press A. See your number printed in the OP 02 print sector. Any other print sector may be substituted, of course.

ROUNDING ROUTINE.- In v5n9/19p24, Milton Cragg presented a rounding routine. There where also rounding routines used in the Draftman's Scaler programs in v6n6p8. John Van Wye and Richard Snow both used the same".5 technique"used by Milton. There is, however, another technique, the"EE INV EE technique,"which Palmer Hanson, Jr used in his program. Here is Palmer's equivalent of Milton's rounding routine. It is three steps shorter, but at the cost of needing a scratch pad location for the indirect FIX. It is also slightly faster.

LBL A PRT X:T R/S STO 00 FIX IND 00 X:T EE INV EE INV FIX PRT R/S

Instructions are identical to the ones used in Milton's program:
Enter the number and press A. Enter the number of decimal places desired and press R/S.

13-DIGIT VIEWER.- I once said I didn't want to see any more hidden digits viewers. Well, that was not really meant litterally. I just tried to stem the tide. Here is the shortest one I have seen so far, written by Bill Beebe. You will notice that it is in reality Dick Blayney's routine, shortened by 5 steps:

LBL A DIV (EE ABS EE DIV EE 0 0 DIV 1 EE 3 INV EE = INV INT R/S

KEYED LABEL PRINTER.- Dave Leising contributed this example of an application of
------------------ self-altering code and special syntax.

Load sides 1 and 2, press GTO 037 R/S. See a flashing zero. DO NOT PRESS CLR. Press any directly keyable label, including 2nd functions. See opcode and mnemonic printed, followed by halt for another entry. The PRT and ADV on the PRINTER even call up their labels!

Note how the SBR at the last step in memory "multiplies" the amount of user-defined keys available!

```
000 95 =
001 24 CE
002 59 INT
003 65 x
004 93 .
005 00 0
006 00 0
007 01 1
008 85 +
009 09 9
010 93 .
011 02 2
012 00 0
013 00 0
014 07 7
015 06 6
016 00 0
017 08 8
018 06 6
019 09 9
020 95 =
021 32 X:T
022 07 7
023 69 OP
024 17 17
025 32 X:T
026 42 STO
027 60 60
028 06 6
029 69 OP
030 17 17
031 25 CLR
032 98 ADV
033 71 SBR
034 04 04
035 75 75
036 98 ADV
037 07 7
038 69 OP
039 17 17
040 07 7
041 93 .
042 01 1
043 42 STO
044 60 60
045 06 6
046 69 OP
047 17 17
048 25 CLR
049 61 GTO
050 04 04
051 79 79
052 76 LBL
053 11 A
054 01 1
055 01 1
056 93 .
057 76 LBL
058 12 B
059 01 1
060 02 2
061 93 .
062 76 LBL
063 13 C
064 01 1
065 03 3
066 93 .
067 76 LBL
068 14 D
069 01 1
070 04 4
071 93 .
072 76 LBL
073 15 E
074 01 1
075 05 5
076 93 .
077 76 LBL
078 16 A'
079 01 1
080 06 6
081 93 .
082 76 LBL
083 17 B'
084 01 1
085 07 7
086 93 .
087 76 LBL
088 18 C'
089 01 1
090 08 8
091 93 .
092 76 LBL
093 19 D'
094 01 1
095 09 9
096 93 .
097 76 LBL
098 10 E'
099 01 1
100 00 0
101 93 .
102 76 LBL
103 22 INV
104 02 2
105 02 2
106 93 .
107 76 LBL
108 23 LNX
109 02 2
110 03 3
111 93 .
112 76 LBL
113 24 CE
114 02 2
115 04 4
116 93 .
117 76 LBL
118 25 CLR
119 02 2
120 05 5
121 93 .
122 76 LBL
123 27 INV
124 02 2
125 07 7
126 93 .
127 76 LBL
128 28 LOG
129 02 2
130 08 8
131 93 .
132 76 LBL
133 29 CP
134 02 2
135 09 9
136 93 .
137 76 LBL
138 20 CLR
139 02 2
140 00 0
141 93 .
142 76 LBL
143 32 X:T
144 03 3
145 02 2
146 93 .
147 76 LBL
148 33 X²
149 03 3
150 03 3
151 93 .
152 76 LBL
153 34 √X
154 03 3
155 04 4
156 93 .
157 76 LBL
158 35 1/X
159 03 3
160 05 5
161 93 .
162 76 LBL
163 36 PGM
164 03 3
165 06 6
166 93 .
167 76 LBL
168 37 P/R
169 03 3
170 07 7
171 93 .
172 76 LBL
173 38 SIN
174 03 3
175 08 8
176 93 .
177 76 LBL
178 39 COS
179 03 3
180 09 9
181 93 .
182 76 LBL
183 30 TAN
184 03 3
185 00 0
186 93 .
187 76 LBL
188 42 STO
189 04 4
190 02 2
191 93 .
192 76 LBL
193 43 RCL
194 04 4
195 03 3
196 93 .
197 76 LBL
198 44 SUM
199 04 4
200 04 4
201 93 .
202 76 LBL
203 45 YX
204 04 4
205 05 5
206 93 .
207 76 LBL
208 47 CMS
209 04 4
210 07 7
211 93 .
212 76 LBL
213 48 EXC
214 04 4
215 08 8
216 93 .
217 76 LBL
218 49 PRD
219 04 4
220 09 9
221 93 .
222 76 LBL
223 52 EE
224 05 5
225 02 2
226 93 .
227 76 LBL
228 53 (
229 05 5
230 03 3
231 93 .
232 76 LBL
233 54 )
234 05 5
235 04 4
236 93 .
237 76 LBL
238 55 ÷
239 05 5
240 05 5
241 93 .
242 76 LBL
243 57 ENG
244 05 5
245 07 7
246 93 .
247 76 LBL
248 58 FIX
249 05 5
250 08 8
251 93 .
252 76 LBL
253 59 INT
254 05 5
255 09 9
256 93 .
257 76 LBL
258 50 I×I
259 05 5
260 00 0
261 93 .
262 76 LBL
263 61 GTO
264 06 6
265 01 1
266 93 .
267 76 LBL
268 65 ×
269 06 6
270 05 5
271 93 .
272 76 LBL
273 66 PAU
274 06 6
275 06 6
276 93 .
277 76 LBL
278 67 EQ
279 06 6
280 07 7
281 93 .
282 76 LBL
283 68 NOP
284 06 6
285 08 8
286 93 .
287 76 LBL
288 69 OP
289 06 6
290 09 9
291 93 .
292 76 LBL
293 60 DEG
294 06 6
295 00 0
296 93 .
297 76 LBL
298 71 SBR
299 07 7
300 01 1
301 93 .
302 76 LBL
303 75 -
304 07 7
305 05 5
306 93 .
307 76 LBL
308 76 LBL
309 07 7
310 06 6
311 93 .
312 76 LBL
313 77 GE
314 07 7
315 07 7
316 93 .
317 76 LBL
318 78 Σ+
319 07 7
320 08 8
321 93 .
322 76 LBL
323 79 x̄
324 07 7
325 09 9
326 93 .
327 76 LBL
328 70 RAD
329 07 7
330 00 0
331 93 .
332 76 LBL
333 81 RST
334 08 8
335 01 1
336 93 .
337 76 LBL
338 85 +
339 08 8
340 05 5
341 93 .
342 76 LBL
343 86 STF
344 08 8
345 06 6
346 93 .
347 76 LBL
348 87 IFF
349 08 8
350 07 7
351 93 .
352 76 LBL
353 88 DMS
354 08 8
355 08 8
356 93 .
357 76 LBL
358 89 π
359 08 8
360 09 9
361 93 .
362 76 LBL
363 80 GRD
364 08 8
365 00 0
366 93 .
367 76 LBL
368 91 R/S
369 09 9
370 01 1
371 93 .
372 76 LBL
373 93 .
374 09 9
375 03 3
376 93 .
377 76 LBL
378 94 +/-
379 09 9
380 04 4
381 93 .
382 76 LBL
383 95 =
384 09 9
385 05 5
386 93 .
387 76 LBL
388 96 WRT
389 09 9
390 06 6
391 93 .
392 76 LBL
393 97 DSZ
394 09 9
395 07 7
396 93 .
397 76 LBL
398 98 ADV
399 09 9
400 08 8
401 93 .
402 76 LBL
403 99 PRT
404 09 9
405 09 9
406 93 .
407 76 LBL
408 90 LST
409 09 9
410 00 0
411 93 .
412 71 SBR
```

MIRROR FUNCTION.- Dave Leising tells me that he just discovered that the "mirror
--------------- function" executed on fractional numbers in the x-register when used as HIR arithmetic is also executed with INV LIST.

Examples: Input :	See in INV LIST:
0.1	10
0.11	11
0.111	11.1
0.001	1000
0.09	900

The INV LIST mirror only works if the peripheral is sensed. (KP and DO diode, see TI-59 schematic)

To try this, put the machine in 0 OP 17 to prevent nuisance listings.

RELATIVE MOTION.- (of ships or aircraft) Robert M. Elliot, 29 Ox Hill Road, Norwich,
-------------- CT, 06360 (a TI PPC Club member) sent me a super program, consisting of three sets of cards and three sets of instructions to calculate the relative motion of ships or aircraft. I cannot vouch for the accuracy of the results, but I see good programming here, very friendly to the user. It is a pitty the whole program system is so long, otherwise I would publish it in its entirety. Bob is willing to send any member all three sets of mag cards and instructions for $ 3.00 US. A bargain.

NUMERICAL PRINT CODE CONVERTER.- I found this program in Funkschau (West Germany) 1980. Heft 18, pp 77-78. The author is Harald Lindner.

It is intended as a SBR and will convert any number up to five digits into the required print code and store it in one of the print sector registers. It will do this for all four print registers and subsequently print all of them on one line.

The original program required SBR calls for each of the print sectors. So, I changed those calls to LBL A', B', C' and D'. Then I added LBL E' to it, to enable also a one-key printing call. The original program furthermore used R00. It was changed to R09. Most programs use the first registers as working registers. If you need R09 in your program you might even change to R59 for this SBR. OP 29 becomes then 1 SUM 59 and all the direct addresses above step 030 have to be increased by 1.

A typical call in your main program would look like this:
RCL 25 A' RCL 26 B' RCL 27 C' RCL 28 D' E' Or it might also look like this:
123 A' 45678 B' 39 +/- C' 22 D' E'

It is rather fast, faster than Snow's print code converter. In defense of the latter I could say that this SBR has a lot less to accomplish. It converts only numeric data, not alphanumeric characters.

This routine will convert only integers, positive or negative. It is not necessary to fill all of the print sectors. Just skip the ones you don't need. So, it is perfectly alright to call, for example: 125 A' 25 B' E'

If you plan to use the routine directly from the keyboard, rather than as a SBR in your main program, by all means make it easy on yourself and change all the secondary user-defined keys to the primary ones. Thus, A' becomes A, B' becomes B, etc. That reduces the amount of key punching considerably.

```
000 29 CP      017 32 X:T     034 77 GE      051 01 1       069 00 00      087 03 03      105 09 09
001 22 INV     018 50 IxI     035 00 00      052 00 0       070 69 OP      088 92 RTN     106 82 HIR
002 67 EQ      019 55 ÷       036 39 39      053 97 DSZ     071 01 01      089 76 LBL     107 04 04
003 00 00      020 28 LOG     037 02 2       054 09 09      072 92 RTN     090 19 D'      108 00 0
004 07 07      021 59 INT     038 85 +       055 00 00      073 76 LBL     091 71 SBR     109 82 HIR
005 01 1       022 48 EXC     039 01 1       056 32 32      074 17 E'      092 00 00      110 03 03
006 92 RTN     023 09 09      040 00 0       057 25 CLR     075 71 SBR     093 00 00      111 09 9
007 32 X:T     024 82 HIR     041 00 0       058 82 HIR     076 00 00      094 69 OP      112 32 X:T
008 22 INV     025 04 04      042 82 HIR     059 14 14      077 00 00      095 04 04      113 61 GTO
009 77 GE      026 43 RCL     043 43 43      060 42 STO     078 69 OP      096 92 RTN     114 00 00
010 00 00      027 09 09      044 01 1       061 00 00      079 02 02      097 55 ÷       115 32 32
011 14 14      028 69 OP      045 75 -       062 82 HIR     080 92 RTN     098 05 5       116 76 LBL
012 02 2       029 29 29      046 59 INT     063 13 13      081 76 LBL     099 22 INV     117 10 E'
013 00 0       030 22 INV     047 82 HIR     064 92 RTN     082 18 C'      100 28 LOG     118 69 OP
014 82 HIR     031 28 LOG     048 33 33      065 76 LBL     083 71 SBR     101 95 =       119 05 05
015 03 03      032 85 +       049 95 =       066 16 A'      084 00 00      102 32 X:T     120 69 OP
016 09 9       033 22 INV     050 65 ×       067 71 SBR     085 00 00      103 05 5       121 00 00
                                             068 00 00      086 69 OP      104 48 EXC     122 92 RTN
```

CALCULATOR STATUS ROUTINES.- Bill Beebe gave me over the phone these two routines. The first one, when executed, returns a "0" if the calculator is in the standard mode, a "1" if the calculator is in the EE mode and a "2" if the calculator happens to be in the ENG mode.

LBL A EE 0 0 INV ENG √X √X √X INT RTN

The second routine is a FIX mode indicator. It works for the normal mode for FIX 0 through FIX 9 and, something the TI routine in M/U-20 doesn't do, it works for EE and ENG modes from o through 6. It does not use the t-reg as the TI routine does. Both routines are shorter by several steps, compared to the TI routine.

These two might be good candidates for our own module, unless somebody finds a way to improve upon them.

LBL A (.3 1/X - EE INV EE) LOG INT IXI RTN

SOLAR ENERGY .- The Ohio Solar Energy Association, 13125 Dortyh Drive, Chesterland, OH 44026, USA, runs a TI-59 program exchange. Five programs are in their catalog so far: 1. Residential heating. 2. F-chart for liquid space heating and DHW systems. 3. F-chart for air space heating and DHW systems. 4. F-chart for DHW systems. 5. Lue and Jordon solar radiation on titled surface. ($ 10.00 each)
Many other subjects are waiting to be programmed. If you would like to volunteer for a program assignment, contact Joe Barbish at 216 729-9350, after 6 PM.

LOG ANALYSIS BY MICROMETER by Dr. George B. Asquith. Publishers: PennWell Books, P.O. Box 1260, Tulsa, Oklahoma, 74101. Price: $ 23.00.

This book on petroleum geology borehole logging analysis begins with a simple but effective introduction to microcomputers and general instructions for their operation. Chapter two reviews basic log analysis techniques and contains excellent references for more detailed discussion. Chapters three to six explain in a well-organized manner the different logging analysis formulas, their use and programs in BASIC. The programs are well thought out, documented with care and the examples are easy to follow.

Even though the book is written for microcomputers utilizing the BASIC language, the programs are easily adaptable to AOS or RPN programmable calculator language. The book is an excellent guide to novice or expert users of microcomputers who have a knowledge of hydrocarbon exploration and logging techniques.

(Reviewed by Barry Frantz)

ANGLE CONVERTERS.- Re- v5n3p4 and v5n4/5p25. After careful analysis it has been shown that indeed Ralph Donnely's routine has fewer steps and that it can be used for angles in both D.dd and D.MS formats. It cannot be used however for angles less than zero, which are encountered rather often, since the P/R function returns angles between -90 and +270 degrees.

In order to incorporate this ability to handle negative numbers, more restrictions and more program steps are needed in Frank Blachly's routine.

CARD READER CARE.- Beware of the use of graphite (lead) pencils to mark your magnetic cards. The graphite wears off and gets into the reader mechanism. It will ruin the mechanism in less than a few months.

Smoke from cigarettes will deposit a fine layer of tar on the magnetic cards and this will have the same disastrous effect on the reader mechanism.

By the way, the flat fee TI repair centers are charging for the exchange of a calculator has been increased to $ 63.00.

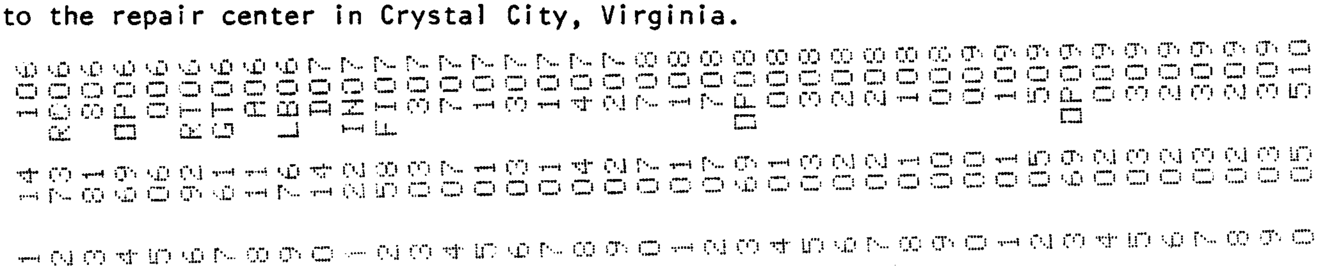
A TRULY DEDICATED TI-59.- Bob Patton in Arlington, Texas, ran across one. Here is his account of the event:

As I stepped up to the desk of a university student center, a strange TI-59/PC100 caught my eye. The 59 was gutted. Most of the keys had been removed and the keyholes were covered with black tape. A short set of mimeographed instructions lay beside this curious assembly.

The 59 was a gatekeeper for the photo club darkrooms. It verified the combination of club member number, student ID, and key wanted. Then it printed a receipt to be signed by the student and stamped by the student center desk clerk as he issued the key.

The only buttons left on this now single-purpose machine were A - E, CLR, RST, and the digits 0 to 9 .

PC-100A ON THE FRITS.- One day Frank Blachly's printer produced this nice listing of one of his programs. Needless to say, it was sent "sito presto" to the repair center in Crystal City, Virginia.



ZEROS OF FUNCTIONS (Bisection Method)

By John Worthington and Emil Regelman

The program will search between upper and lower limits designated by the user for the value of any of up to ten variables which will make a given function equal to zero. The values of the known variables, detected roots and appropriate descriptors are automatically printed.

The bisection method, though significantly slower than the secant method, does overcome two of the latter's limitations. The bisection method will generally find all roots within the search limits, and can be restricted from discontinous portions of the function. This program has been designed to operate very similiarly to the "ZEROS OF FUNCTIONS (Secant Method)" (TI PPC V5N6p6&7).

PROGRAM OPERATION

1. Initialize: [SBR] [RST]

2. Enter the Function to Be Searched: Rearrange the function, so that it is set equal zero. Assign one of the User Defined Keys for each variable value. Press [GTO] [SBR] [LRN] and enter the function as a subroutine using AOS so that it will equal zero with the appropriate variable values. The entered subroutine may be listed by pressing [SBR] [List]

   For example: to evaluate the function $P = I^2 \times R$, rearrange the expression to $I^2 \times R - P = 0$, transform the expression to $B\ X^2 \times C - A =$ RTN, press [GTO] [SBR] [LRN] and enter the keystrokes.

3. Store the Values of the Known Variables:
   a) Enter the value of each variable and press the assigned User Defined Key.
   b) If descriptors (up to 3 characters) are desired, store the corresponding print codes in "t" before entering the variable value. The default descriptors are A through E'.

4. Store the Lower Limit of the Search:
   a) Enter the lower limit and press the User Defined Key assigned to the unknown variable.
   b) If a descriptor is desired, see 3b above.

5. Store the Upper Limit of the Search and the Search Interval:
   a) Enter the upper limit, press [EE] and [A].
   b) Enter the search interval, press [EE] and [B]

6. Initiate the First Search: [SBR] [R/S] will designate the last variable in which a value was stored as the unknown, assume this value to be the lower limit of the search, and...
   a) print the lower and upper limits of the search;
   b) print the search interval (if one was entered);
   c) print the values of the known variables and their descriptors;
   d) search for the roots;
   e) print the roots with "=" (as they are found) along with its descriptor.

   When all roots have been found in the specified interval, the display with show the last root found (fixed but not rounded). The corresponding value (rounded off at the fixed number of decimal places) will have been stored in the corresponding variable memory. If no roots are found, the program will stop with a flashing over-flow in the display.

7. Perform Additional Searches for the Same Variable: (These operations may be performed in any order.)
   a) Change the known variables (as desired, see #3 above).
   b) Enter the new lower limit (see #4 above).
   c) Enter a new upper limit and search interval (as desired, see #5 above).
   d) Press [R/S]. ([R/S] will perform the same operations as [SBR] [R/S] described in #6 except that the program will search for the previously designated unknown.)

8. Perform a Search for a Different Unknown
   a) Change the known variables (as desired, see #3 above).
   b) Enter the new lower limit (see #4 above).
   c) Enter a new upper limit and search interval (as desired, see #5 above).
   d) Press [SBR] [R/S]. ([SBR][R/S] will perform the same operations as [R/S] described in #7 except that the program will designate the last variable in which a value was stored as the unknown).

NOTES

A. The bisection method evaluates a function at specific points: the lower limit, the lower limit + (the interval), the lower limit + 2x(the interval),....the lower limit + nx(the interval), up to the upper limit. When a change of sign between two sample points is found, the interval is successively bisected until the requested accuracy is achieved (see note B). If two roots occur within a designated interval, neither root will be detected. If no interval is specified, the upper and lower search limits are assumed to be the search interval.

B. [SBR] [RST] repartitions the calculator to 640 program steps, clears all stored values and fixes three decimal places. The fixed mode is used to limit the accuracy of the calculated root, and provides a fixed format for the print-out. If a different level of accuracy is desired, fix the appropriate number of decimal places. Up to 160 program steps are available to write the function. The program uses memories 7 through 39. Memories 0 through 6 are available to the user.

C. [SBR] [RCL] will print the search parameters, the values of the known variables (with their respective descriptors), and the variable specified as the unknown (that is, the variable which would be searched for if [R/S] were pressed), with its descriptor and an asterisk.

D. The stored variable values may be recalled individually by setting flag 0 and pressing the appropriate User Defined Key. After this operation [RST] [R/S] must be pressed to return to normal operation.

E. To delete a descriptor and return to the default descriptor, enter a value into the "t" register which is smaller than "1", but not "0". Then enter the appropriate value for that variable into the display and press the assigned User Defined Key. [SBR] [RST] does not clear the stored descriptors, which remain unchanged unless written-over by the storing procedure, or are cleared by [CMS].

F. If the program is interupted during operation, [RST] [R/S] [CLR] must be pressed before any subsequent operations are attempted.

G. The entered function and descriptors may be stored on a magnetic card by recording bank 3. The recorded function may also be used with the Secant Method Program by forcing it into bank 2. However, the descriptors will have to be stored again, since different memories are used by the Secant Program.

H. If a function is divisible by another function an even number of times, the program will not detect the corresponding root unless it is entered as the lower limit of the search, since, effectively, there are an even number of roots at the same point. If such a situation is suspected, the first derivative of the original function should be evaluated, to determine any maxima or minima, and these points checked by entering them (one at a time) as the lower limit of a search. If they are roots, they will be printed.

I. The program may also be used to evaluate the function with specific variable values. Store the desired values (as described in #3 above), set flag 0, and press [SBR] [SBR]. After this operation [RST] [R/S] must be pressed to return to normal operation.

J. The use of [SBR] [R/S] and [R/S] and the comments in Notes C, D, E, F, G, and I also generally apply to the secant program.

ZEROS OF FUNCTIONS
(BISECTION METHOD)

```
000  29 CP      060  69 OP      120  73 RC*     180  08 8       240  12 12      300  02 02      360  15 15      420  07 7
001  92 RTN     061  27 27      121  11 11      181  82 HIR     241  32 X:T     301  16 16      361  54 )       421  87 IFF
002  61 GTO     062  76 LBL     122  83 GO*     182  58 58      242  43 RCL     302  87 IFF     362  55 ÷       422  02 02
003  01 01      063  18 C'      123  07 07      183  43 RCL     243  14 14      303  02 02      363  02 2       423  04 04
004  30 30      064  69 OP      124  76 LBL     184  09 09      244  75 -       304  04 04      364  95 =       424  26 26
005  76 LBL     065  27 27      125  91 R/S     185  42 STO     245  67 EQ      305  10 10      365  72 ST*     425  08 8
006  12 B       066  76 LBL     126  43 RCL     186  18 18      246  02 02      306  35 1/X     366  12 12      426  52 EE
007  69 OP      067  17 B'      127  11 11      187  29 CP      247  54 54      307  42 STO     367  25 CLR     427  03 3
008  27 27      068  69 OP      128  42 STO     188  22 INV     248  29 CP      308  16 16      368  43 RCL     428  33 X²
009  76 LBL     069  27 27      129  12 12      189  67 EQ      249  73 RC*     309  86 STF     369  15 15      429  95 =
010  11 A       070  76 LBL     130  22 INV     190  01 01      250  14 14      310  20 20      370  52 EE      430  82 HIR
011  87 IFF     071  16 A'      131  76 LBL     191  98 98      251  67 EQ      311  43 RCL     371  32 X:T     431  08 08
012  00 00      072  69 OP      132  43 RCL     192  73 RC*     252  02 02      312  08 08      372  25 CLR     432  25 CLR
013  00 00      073  27 27      133  86 STF     193  12 12      253  93 93      313  32 X:T     373  43 RCL     433  73 RC*
014  88 88      074  76 LBL     134  02 02      194  95 =       254  07 7       314  43 RCL     374  14 14      434  12 12
015  53 (       075  15 E       135  98 ADV     195  42 STO     255  87 IFF     315  19 19      375  52 EE      435  42 STO
016  24 CE      076  69 OP      136  25 CLR     196  18 18      256  01 01      316  77 GE      376  67 EQ      436  16 16
017  85 +       077  27 27      137  01 1       197  76 LBL     257  02 02      317  04 04      377  04 04      437  69 OP
018  09 9       078  76 LBL     138  02 2       198  69 OP      258  64 64      318  45 45      378  10 10      438  06 06
019  22 INV     079  14 D       139  22 INV     199  06 6       259  75 -       319  42 STO     379  25 CLR     439  22 INV
020  28 LOG     080  69 OP      140  28 LOG     200  25 CLR     260  04 4       320  14 14      380  73 RC*     440  87 IFF
021  52 EE      081  27 27      141  85 +       201  69 OP      261  93 .       321  85 +       381  12 12      441  02 02
022  94 +/-     082  76 LBL     142  02 2       202  00 00      262  03 3       322  43 RCL     382  71 SBR     442  03 03
023  32 X:T     083  13 C       143  07 7       203  69 OP      263  05 5       323  18 18      383  04 04      443  10 10
024  65 ×       084  69 OP      144  03 3       204  05 05      264  95 =       324  95 =       384  80 80      444  81 RST
025  00 0       085  27 27      145  02 2       205  42 STO     265  65 ×       325  42 STO     385  65 ×       445  25 CLR
026  77 GE      086  69 OP      146  02 2       206  13 13      266  01 1       326  19 19      386  43 RCL     446  98 ADV
027  00 00      087  27 27      147  07 7       207  01 1       267  52 EE      327  42 STO     387  17 17      447  98 ADV
028  31 31      088  48 EXC     148  03 3       208  09 9       268  04 4       328  15 15      388  95 =       448  98 ADV
029  86 STF     089  07 07      149  09 9       209  42 STO     269  95 =       329  72 ST*     389  29 CP      449  43 RCL
030  06 06      090  22 INV     150  95 =       210  14 14      270  82 HIR     330  12 12      390  67 EQ      450  16 16
031  01 1       091  52 EE      151  82 HIR     211  02 2       271  38 38      331  71 SBR     391  04 04      451  52 EE
032  65 ×       092  42 STO     152  08 08      212  09 9       272  43 RCL     332  04 04      392  10 10      452  72 ST*
033  32 X:T     093  11 11      153  73 RC*     213  42 STO     273  14 14      333  80 80      393  77 GE      453  12 12
034  00 0       094  42 STO     154  12 12      214  15 15      274  32 X:T     334  42 STO     394  04 04      454  25 CLR
035  54 )       095  10 10      155  42 STO     215  22 INV     275  43 RCL     335  17 17      395  03 03      455  48 EXC
036  22 INV     096  02 2       156  19 19      216  22 INV     276  12 12      336  43 RCL     396  73 RC*     456  16 16
037  87 IFF     097  00 0       157  69 OP      217  86 STF     277  22 INV     337  14 14      397  12 12      457  81 RST
038  06 06      098  44 SUM     158  06 06      218  01 01      278  67 EQ      338  72 ST*     398  42 STO     458  76 LBL
039  00 00      099  11 11      159  01 1       219  05 5       279  02 02      339  12 12      399  15 15      459  81 RST
040  88 88      100  00 0       160  04 4       220  42 STO     280  88 88      340  71 SBR     400  61 GTO     460  22 INV
041  32 X:T     101  48 EXC     161  00 0       221  17 17      281  82 HIR     341  04 04      401  03 03      461  58 FIX
042  08 8       102  07 07      162  01 1       222  01 1       282  18 18      342  80 80      402  55 55      462  03 3
043  44 SUM     103  87 IFF     163  52 EE      223  44 SUM     283  42 STO     343  29 CP      403  73 RC*     463  69 OP
044  07 07      104  00 00      164  04 4       224  14 14      284  13 13      344  52 EE      404  12 12      464  17 17
045  32 X:T     105  01 01      165  82 HIR     225  44 SUM     285  61 GTO     345  67 EQ      405  42 STO     465  47 CMS
046  72 ST*     106  20 20      166  38 38      226  15 15      286  02 02      346  04 04      406  14 14      466  04 4
047  07 07      107  72 ST*     167  25 CLR     227  52 EE      287  93 93      347  10 10      407  61 GTO     467  69 OP
048  42 STO     108  11 11      168  43 RCL     228  01 1       288  25 CLR     348  65 ×       408  03 03      468  17 17
049  07 07      109  03 3       169  08 08      229  02 2       289  73 RC*     349  48 EXC     409  53 53      469  25 CLR
050  00 0       110  00 0       170  75 -       230  82 HIR     290  14 14      350  17 17      410  43 RCL     470  58 FIX
051  48 EXC     111  44 SUM     171  69 OP      231  08 08      291  69 OP      351  95 =       411  13 13      471  03 03
052  07 07      112  10 10      172  06 06      232  01 1       292  06 06      352  77 GE      412  29 CP      472  86 STF
053  81 RST     113  00 0       173  01 1       233  32 X:T     293  25 CLR     353  03 03      413  22 INV     473  09 09
054  76 LBL     114  67 EQ      174  07 7       234  73 RC*     294  97 DSZ     354  10 10      414  67 EQ      474  81 RST
055  10 E'      115  01 01      175  00 0       235  15 15      295  17 17      355  53 (       415  04 04      475  76 LBL
056  69 OP      116  20 20      176  01 1       236  77 GE      296  02 02      356  43 RCL     416  19 19      476  90 LST
057  27 27      117  32 X:T     177  08 8       237  02 02      297  22 22      357  14 14      417  98 ADV     477  90 LST
058  76 LBL     118  72 ST*     178  09 9       238  70 70      298  87 IFF     358  85 +       418  81 RST     478  76 LBL
059  19 D'      119  10 10      179  08 8       239  43 RCL     299  01 01      359  43 RCL     419  85 +       479  71 SBR
```

LABELS

```
006  12 B
010  11 A
053  10 E'
059  19 D'
063  18 C'
067  17 B'
071  16 A'
075  15 E
079  14 D
083  13 C
125  91 R/S
132  43 RCL
459  81 RST
476  90 LST
479  71 SBR
```

SR-56 LISTING ON A TI-59/PC100.- Here is Bill Skillman's program as promised in v6n2. It will list any SR-56 program by simply entering the code of each particular step. The program counter will be incremented automatically.

Instructions:

1. Read in banks 1, 2, 3 and 4.
2. Enter the starting step, from 0 to 99. Press A. The default value (if no step # is entered) is 0.
   The feature that you can start at any step # is very handy, almost a "must." Suppose you are running along just fine and you have listed your program up to step 87. Then you make a mistake. Normally you would have to start all over, requiring you to swear to all devils in hell. Now, you just enter 87 and press A and you are back on the track. Just glue the two listings together.
3. Enter key code and press R/S.Program prints the listing for this step and displays the key code. If the key code is illegal, it will flash that illegal code and print nothing. If the illegal code follows 26, F(n), it will flash 26. (0 to 5 are legal) It is not necessary to clear the flashing. Program does it automatically.
   Repeat step 3 as needed.

000	76	LBL	041	32	HIR	082	03	03
001	10	E'	042	14	14	083	19	D'
002	22	INV	043	91	R/S	084	06	6
003	59	INT	044	82	HIR	085	32	X:T
004	52	EE	045	04	04	086	76	LBL
005	04	4	046	32	X:T	087	69	OP
006	22	INV	047	25	CLR	088	68	NOP
007	52	EE	048	02	2	089	82	HIR
008	50	I×I	049	06	6	090	14	14
009	92	RTN	050	32	X:T	091	91	R/S
010	76	LBL	051	67	EQ	092	77	GE
011	19	D'	052	00	00	093	00	00
012	82	HIR	053	74	74	094	37	87
013	13	13	054	42	STO	095	24	CE
014	42	STO	055	26	26	096	85	+
015	26	26	056	73	RC*	097	82	HIR
016	73	RC*	057	26	26	098	04	04
017	26	26	058	29	CP	099	01	1
018	10	E'	059	22	INV	100	00	0
019	69	OP	060	77	GE	101	01	1
020	01	01	061	00	00	102	95	=
021	69	OP	062	40	40	103	69	OP
022	05	05	063	69	OP	104	02	02
023	01	1	064	03	03	105	65	×
024	82	HIR	065	73	RC*	106	05	5
025	33	33	066	26	26	107	75	-
026	92	RTN	067	10	E'	108	04	4
027	76	LBL	068	69	OP	109	04	4
028	11	A	069	02	02	110	05	5
029	93	.	070	19	D'	111	95	=
030	82	HIR	071	61	GTO	112	42	STO
031	04	04	072	00	00	113	26	26
032	82	HIR	073	41	41	114	73	RC*
033	03	03	074	03	3	115	26	26
034	69	OP	075	00	0	116	69	OP
035	00	00	076	07	7	117	03	03
036	01	1	077	69	OP	118	61	GTO
037	00	0	078	02	02	119	00	00
038	69	OP	079	43	RCL	120	70	70
039	17	17	080	11	11			
040	34	ΓX	081	69	OP			

REGISTERS 11, 30, 57, 60, 65 AND 99 HAVE TO BE LOADED BY THE STO AND SUM METHOD.
FOR EXAMPLE: REG 65 IS LOADED AS FOLLOWS:
30171331 +/- STO 65 .0706 +/- SUM 65
REG 11 IS LOADED AS FOLLOWS:
21553156 STO 11 .0202 SUM 11

1.0101	00	-0.0601	50
2.0102	01	1515.0602	51
3.0103	02	55.0603	52
4.0104	03	56.0604	53
5.0105	04	72.0605	54
6.0106	05	-0.0606	55
7.0107	06	1533.0607	56
8.0108	07	36411435.0608	57
9.0109	08	353731.0609	58
12.0112	09	331341.0612	59
152735.0201	10	-36161742.0701	60
21553156.0202	11	-0.0702	61
243142.0203	12	-0.0703	62
273144.0204	13	-0.0704	63
5466.0205	14	50.0705	64
152735.0206	15	-30171331.0706	65
-0.0207	16	-0.0707	66
243142.0208	17	-0.0708	67
273222.0209	18	-0.0709	68
30166.0212	19	53.0712	69
26344.0301	20	-336335.0801	70
-0.0302	21	-0.0802	71
323732.0303	22	-0.0803	72
362431.0304	23	-0.0804	73
153236.0305	24	20.0805	74
371331.0306	25	-356333.0806	75
0.	26	-0.0807	76
163646.0308	27	-0.0808	77
244424.0309	28	-0.0809	78
243137.0312	29	351316.0812	79
33353216.0401	30	-7747.0901	80
-0.0402	31	-0.0902	81
446337.0403	32	-0.0903	82
363722.0404	33	-0.0904	83
351527.0405	34	47.0905	84
364130.0406	35	-7720.0906	85
-0.0407	36	-0.0907	86
446437.0408	37	-0.0908	87
153036.0409	38	-0.0909	88
174415.0412	39	47.0912	89
565245.0501	40	-0.1201	90
356336.0502	41	-0.1202	91
353637.0503	42	40.1203	92
4470.0504	43	476320.1204	93
1717.0505	44	64.1205	94
4566.0506	45	-0.1206	95
313233.0507	46	-0.1207	96
2217.0508	47	333537.1208	97
5244.0509	48	331333.1209	98
212444.0512	49	27243637.1212	99

CORRECTION- In Orthographic Projection by Fred Fuller, v6n1p13, a few typos sneeked in: line 10 should read:

x2 = cosA2 cosA3 x3 - cosA2 sinA3 y3 + sinA2 z3

line 11 is as follows:

y2 = (sinA1 sinA3 - cosA1 sinA2 cos A3) x3 + (sinA1 cosA3 + cosA1 sinA2 sinA3) y3 + (cosA1 cosA2) z3

line 12 should read:

These formulas rotate the object about the z-axis first, then the y-axis and finally about the x-axis.

NEWCOMER'S CORNER.- *In a recent program by John Worthington I saw the neatest trick in years. The program was intended to test your knowledge of the TI-59. The subject was required to give one of five choice answers to one hundred questions. Answering was done by pressing either one of keys A through E. John made this program especially for the teachers in the TI-PPP program, of which yours truly is one of the four teachers. John's program tabulated your score and printed it out after all one hundred questions are answered. The TI-PPP teachers up to now used a similar list of one hundred questions,supplied by TI. However, everything up to now was done on paper, which required the teacher to correct in a hurry about twelve questionnairs. Now, the calculator will take care of that.*

As you can see, the correct answer to each question had to be stored in the program, called to the t-register and compared to the answer given by the student.

The normal way to do this would be to write in user-memory a number between 1 and 5 followed by RTN each time. Suppose you start doing this at step 100. A simple SBR IND 00 will get your first digit, provided register 00 had the number 100 in it. You now increase register 00 by 2, such that at the next question it contains the number 102. The SBR IND 00 will now call the digit starting at step 102, finds a RTN at step 103 and returns. The next digit is written at step 104, with another RTN at step 105, etc.

To store 100 digits this way, you will need 200 program steps.

Of course, you might even consider to store each digit separately in a data register. But that is a wasteful way of doing things. And even if you partition to 10 OP 17, you wont be able to store all 100 digits, because you will need at least one register for a RCL IND. Moreover, it will cost you 100 X 8 = the equivalent of 800 program steps.

John managed to do it in 101 steps. To illustrate how it is done, enter the following short sequence in user memory starting at step 000:

LBL A SBR IND 00 DIV 1 EE 9 INV EE = INT OP 20 R/S

Go out of learn again and press GTO 100. Then at 100, in user memory, write all the required digits sequentially, one digit per step. The very last step should be a RTN.

Now go out of LRN again, and press 100 STO 00.

If you now press A, the very first digit, the one from step 100, will appear in the display. If you press A again, the digit from step 101 will be displayed, etc. etc. How does this work?

Your pointer register contains initially the number 100. So, a SBR IND 00 will call the subroutine starting at step 100. The program will read digits until the display is is full. That means it will read exactly 10 digits. The program pointer will now keep running until it finds the RTN command at the very end. All the digits it encounters on its way will simply be ignored. The sequence in LBL A will divide the 10-digit number by 10^9 or 1,000,000,000 and will leave the number with a single decimal point thus: 1.234512345 after which the integer part is taken and the 1 displayed. If you want to check on it, put the calculator-printer in TRACE. Register 00 is now increased by 1, by the special OP : OP 20, which is the same as saying: 1 SUM 00, but is one step shorter.

The next time you press A, the sequence will call (indirectly) SBR 101 and digits 2345123451 will be read, divided by 10^9 to give 2.345123451. After the INT command only the 2 will be left over, etc. etc.

This same scheme may be used with a multitude of variations: you may call any number of digits, provided you increase register 00 by the same amount. You might even call print code this way. It is not the fastest way known, but it is economical in program steps, as you use only one step per digit. Numbers stored in data registers do not use the data register to the fullest if they are shorter than 8 digits long. Above eight digits it becomes more economical to use the data-register-storing scheme.

See you next time, Maurice