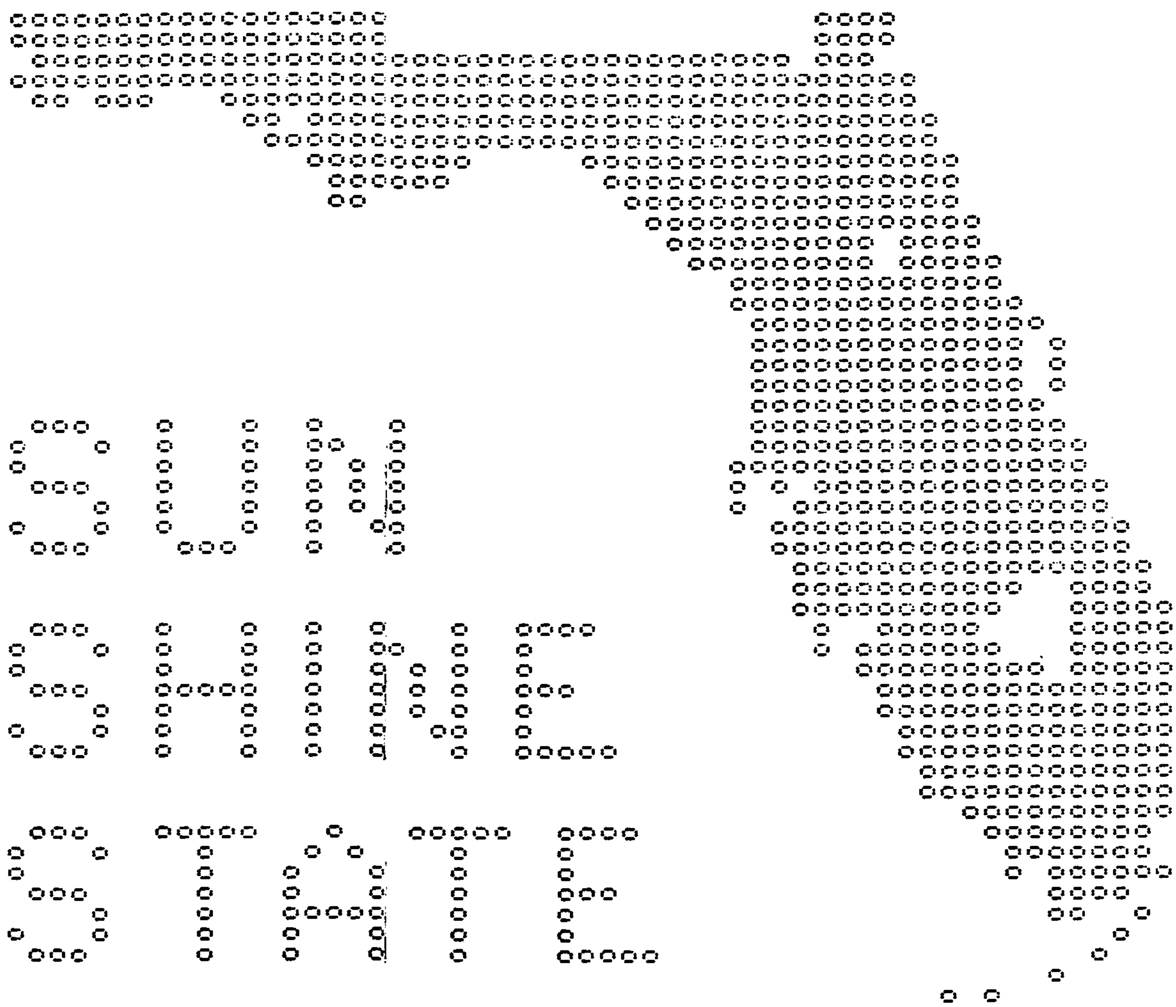
TI PPC NOTES

NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB

P.O. Box 1421, Largo, FL 34294

Volume 9, Number 3 May/June 1984

Fast Mode Scribbler - see page 18.



ERRATA:

One Minute 23 Second Calendar - V9N2P7 - The table of constants on page 27 did not print well and some members had problems entering the correct values. The table is reprinted below:

0.	00	10.00000161715	18	2.010011000012	36	2.010307000310	54
0.	01	1.000036000030	19	2.010012000201	37	2.010310000311	55
0.	02	1.014300003700	20	2.010201000202	38	2.010311000312	56
0.	03	1.000021000036	21	2.010202000203	39	2.010312000401	57
0.	04	98.99000000000	22	2.010203000204	40	2.010401000402	58
0.	05	2.010000000000	23	2.010204000205	41	1.000000000002	59
0.	06	2.010000000000	24	2.010205000206	42	6.010311000000	60
10.00000251331	07	2.010000000000	25	2.010206000207	43	6.010312000000	61
8.000000211714	08	2.010000000000	26	2.010207000210	44	6.010401000000	62
10.00000301335	09	2.010000000000	27	2.010210000211	45	6.010402000000	63
9.000000133335	10	2.010000000002	28	2.010211000212	46	7.000000211714	64
10.00000301345	11	2.010002000003	29	2.010212000301	47	0.	65
9.000025413117	12	2.010003000004	30	2.010301000302	48	0.	66
10.00025412745	13	2.010004000005	31	2.010302000303	49	0.	67
10.00000134122	14	2.010005000006	32	2.010303000304	50	0.	68
9.000036173337	15	2.010006000007	33	2.010304000305	51	0.	69
10.00000321537	16	2.010007000010	34	2.010305000306	52		
9.000000313242	17	2.010010000011	35	2.010306000307	53		

Richard Snow's 13 Digit Register List - V5N9/10P17

10.00000251331	07
8.000000211714	08
10.00000301335	09
9.00000013333	10
10.00000301345	11
9.00002541311	12
10.00025412745	13
10.00000134122	14
9.00003617333	15
10.00000321537	16
9.00000031324	17
10.00000161715	18
1.00003600003	19
1.0143000037	20
1.000021000036	21
98.99	22
2.01	23

The table of constants for Patrick Acosta's one minute 23 second calendar on V9N2P7 and in the errata above were printed with Clyde Durbin's 13 digit register list program from V5N9/10P17. Hewlett Ladd managed to enter the marginally legible constants properly. Then he printed them using Richard Snow's 13 digit register list program (also from V5N9/10P17). He found that when the most significant digit was a nine, then the thirteenth digit would not print. Compare his listing with the Snow routine at the right with the listing from the Durbin routine above at data registers 10, 12, 15 and 17. The Snow routine does have some advantages--for example, it truncates trailing zeroes, while the Durbin routine does not, as illustrated by the printouts of the contents of memory registers 19, 20, 22 and 23. But the Snow routine also has problems with exponents. Elsewhere in this issue there is a Newcomer's Column dedicated to a comparison of several of the 13 digit register list programs which are available.

Non-commutative Multiply on the TI-59 - V9N2P15. On V9N2P1 I stated that I had searched old issues of 52 Notes, TI PPC Notes, PPX Exchange, and TISOFT without finding any mention of this quirk. Robert Prins forwarded three pages from an issue of TISOFT that is earlier than those in my files which gives a detailed analysis of the non-commutative multiply quirk. Dejan Ristanovic wrote that he recalled reading about the non-commutative multiply, but could not give a reference. Neither Prins nor Ristanovic could recall previous discussion of the double divide technique. If you would like a copy of the TISOFT discussion send a stamped and self addressed envelope.

ERRATA (cont)

The 1.000001 Squared 27 Times Test - V9N2P11 - The "exact" value listed for Method B was incorrectly shown as 674630.4707... . It should be 674530.4707... . That was the best "exact" value available at the time. Laurance Leeds has provided an answer correct to thirty digits as

674530.470741 084559 382689 178015 +

This shows that the Method C solution of the Model 100 and the TI-66 is good to ten significant figures. Members have sent in solutions for a wide variety of calculators and computers. So far I have not seen a calculator solution which even approaches the errors obtained from the Apple and Commodore 64.

Ohlsson's Solution for Linear Equations - V8N6P14 - Lars Hedlund writes that others were involved in the development of this program. He participated in optimization, and Bjorn Gustavsson provided the conversion to fast mode.

George Thomson has found that the Ohlsson program, and its derivatives by Prins and Ristanovic, cannot solve some configurations of linear equations; for example,

```
9 0 0  18     is successfully solved with roots 2, 1, 3
0 6 0   6
0 0 4  12

9 0 0  18     yields a flashing display after the 4 is entered
0 0 4  12     in the second row. If you continue to enter the
0 6 0   6     remainder of the problem while ignoring the
              flashing you get the incorrect 2, 0, 1 solution.

0 0 4  12     yields a flashing display after the second zero
0 6 0   6     in the first row is entered. If you continue on
9 0 0  18     anyway, you will get the incorrect solution of
              0, 1, 1 .
```

These mini-problems suggest that some prearrangement may be necessary to avoid zeroes in the diagonal terms. This is probably caused by the reduction of the rows as the program proceeds, a feature that is the key to obtaining the solution for up to 16 simultaneous equations.

Polynomial Regression with Variance - V9N2P20 - Gene Friel, who did the programming for this extension of Thomas Wysmuller's program from PPX Exchange offers the following comments and corrections on the documentation in V9N2P20, pages 20-24:

(1) In paragraph 9 on V9N2P22 the second sentence should read "Record banks 2 and 3 if the number of data pairs is from 16 through 30."

(2) In paragraph 14 it is important that the user understand that the

ERRATA (cont)

Polynomial Regression with Variance (cont)

statistical label D should only be pressed before pressing E. After E has been pressed the sums are no longer stored in the registers accessed in subroutine D and can't be properly transferred to registers 1 through 6 for use by the statistics routines.

(3) The capability of generating a series of estimated Y' values for equally spaced X' values which was in the Wysmuller program is still available. After performing step 7 on V9N2P21 to establish the first X',Y' pair, then

7.a. Enter an increment for x (positive or negative), press B' once, and then R/S as many times as required to obtain the desired number of estimates, or

7.b. If a printer is being used, enter the increment and press C'. This prints out Y' values with increments of X'. Stop by pressing RST.

(4) In program B on V9N2P24 steps 125 through 130 should be

Print X^2 Sum 15 Op 20 .

QUIRKS - Palmer Hanson. With your TI-58 or TI-59 in degree mode and with zeroes in the t register and in the display, press INV-2nd-P/R. The number 45 will be returned to the display. This occurs because zero divided by zero yields a flashing 1, and arctan(1) = 45 degrees. The flashing is removed by the CE at location 282 of the firmware. INV-P/R with the TI-66 with zeroes in x and t yields a zero returned to the display; zero divided by zero on the TI-66 yields a zero with an error indication.

The "Zero to Zero Point One" brainteaser from TISOFT was presented on V8N1P12. The problem is to start with zero in the display and obtain 0.1 in the display without using any number keys. Myer Boland and Laurance Leeds submitted four and five step solutions based on the sequence cos - DIV - INV - log . During my investigations leading to the double-divide techniques as a workaround for minimizing errors from the non-commutative multiply quirk I stumbled on a curious six step sequence which relies on error effects to solve the zero to zero point one problem:

1/x X 1/x = 1/x CE

JUMP ADDRESSES FOR STFLG-IND AT END OF PARTITION - Carl Rabe

In V6N8P3 Patrick Acosta defined the constant which must be generated if the STFLG IND at the end of the partition method of fast mode entry is to cause a jump to a program step other than 001 . The jump address is 8*(WXY) + Z + 1 where W, X, Y and Z are the ninth through twelfth digits. Carl Rabe tired of calculating the required values. He wrote a short program to print a table of program steps versus digits ten through 13 for jump addresses from 1 through 480. His table appears on page 5. The integer part is the address to which the program will jump at fast mode entry if digits ten through thirteen contain the values indicated in the corresponding decimal portion. A handy table.

Jump Addresses for Stflg-Ind at End of Partition - (cont)

	80.0972	160.1972	240.2972	320.3972	400.4972
1.0002	81.1002	161.2002	241.3002	321.4002	401.5002
2.0012	82.1012	162.2012	242.3012	322.4012	402.5012
3.0022	83.1022	163.2022	243.3022	323.4022	403.5022
4.0032	84.1032	164.2032	244.3032	324.4032	404.5032
5.0042	85.1042	165.2042	245.3042	325.4042	405.5042
6.0052	86.1052	166.2052	246.3052	326.4052	406.5052
7.0062	87.1062	167.2062	247.3062	327.4062	407.5062
8.0072	88.1072	168.2072	248.3072	328.4072	408.5072
9.0102	89.1102	169.2102	249.3102	329.4102	409.5102
10.0112	90.1112	170.2112	250.3112	330.4112	410.5112
11.0122	91.1122	171.2122	251.3122	331.4122	411.5122
12.0132	92.1132	172.2132	252.3132	332.4132	412.5132
13.0142	93.1142	173.2142	253.3142	333.4142	413.5142
14.0152	94.1152	174.2152	254.3152	334.4152	414.5152
15.0162	95.1162	175.2162	255.3162	335.4162	415.5162
16.0172	96.1172	176.2172	256.3172	336.4172	416.5172
17.0202	97.1202	177.2202	257.3202	337.4202	417.5202
18.0212	98.1212	178.2212	258.3212	338.4212	418.5212
19.0222	99.1222	179.2222	259.3222	339.4222	419.5222
20.0232	100.1232	180.2232	260.3232	340.4232	420.5232
21.0242	101.1242	181.2242	261.3242	341.4242	421.5242
22.0252	102.1252	182.2252	262.3252	342.4252	422.5252
23.0262	103.1262	183.2262	263.3262	343.4262	423.5262
24.0272	104.1272	184.2272	264.3272	344.4272	424.5272
25.0302	105.1302	185.2302	265.3302	345.4302	425.5302
26.0312	106.1312	186.2312	266.3312	346.4312	426.5312
27.0322	107.1322	187.2322	267.3322	347.4322	427.5322
28.0332	108.1332	188.2332	268.3332	348.4332	428.5332
29.0342	109.1342	189.2342	269.3342	349.4342	429.5342
30.0352	110.1352	190.2352	270.3352	350.4352	430.5352
31.0362	111.1362	191.2362	271.3362	351.4362	431.5362
32.0372	112.1372	192.2372	272.3372	352.4372	432.5372
33.0402	113.1402	193.2402	273.3402	353.4402	433.5402
34.0412	114.1412	194.2412	274.3412	354.4412	434.5412
35.0422	115.1422	195.2422	275.3422	355.4422	435.5422
36.0432	116.1432	196.2432	276.3432	356.4432	436.5432
37.0442	117.1442	197.2442	277.3442	357.4442	437.5442
38.0452	118.1452	198.2452	278.3452	358.4452	438.5452
39.0462	119.1462	199.2462	279.3462	359.4462	439.5462
40.0472	120.1472	200.2472	280.3472	360.4472	440.5472
41.0502	121.1502	201.2502	281.3502	361.4502	441.5502
42.0512	122.1512	202.2512	282.3512	362.4512	442.5512
43.0522	123.1522	203.2522	283.3522	363.4522	443.5522
44.0532	124.1532	204.2532	284.3532	364.4532	444.5532
45.0542	125.1542	205.2542	285.3542	365.4542	445.5542
46.0552	126.1552	206.2552	286.3552	366.4552	446.5552
47.0562	127.1562	207.2562	287.3562	367.4562	447.5562
48.0572	128.1572	208.2572	288.3572	368.4572	448.5572
49.0602	129.1602	209.2602	289.3602	369.4602	449.5602
50.0612	130.1612	210.2612	290.3612	370.4612	450.5612
51.0622	131.1622	211.2622	291.3622	371.4622	451.5622
52.0632	132.1632	212.2632	292.3632	372.4632	452.5632
53.0642	133.1642	213.2642	293.3642	373.4642	453.5642
54.0652	134.1652	214.2652	294.3652	374.4652	454.5652
55.0662	135.1662	215.2662	295.3662	375.4662	455.5662
56.0672	136.1672	216.2672	296.3672	376.4672	456.5672
57.0702	137.1702	217.2702	297.3702	377.4702	457.5702
58.0712	138.1712	218.2712	298.3712	378.4712	458.5712
59.0722	139.1722	219.2722	299.3722	379.4722	459.5722
60.0732	140.1732	220.2732	300.3732	380.4732	460.5732
61.0742	141.1742	221.2742	301.3742	381.4742	461.5742
62.0752	142.1752	222.2752	302.3752	382.4752	462.5752
63.0762	143.1762	223.2762	303.3762	383.4762	463.5762
64.0772	144.1772	224.2772	304.3772	384.4772	464.5772
65.0802	145.1802	225.2802	305.3802	385.4802	465.5802
66.0812	146.1812	226.2812	306.3812	386.4812	466.5812
67.0822	147.1822	227.2822	307.3822	387.4822	467.5822
68.0832	148.1832	228.2832	308.3832	388.4832	468.5832
69.0842	149.1842	229.2842	309.3842	389.4842	469.5842
70.0852	150.1852	230.2852	310.3852	390.4852	470.5852
71.0862	151.1862	231.2862	311.3862	391.4862	471.5862
72.0872	152.1872	232.2872	312.3872	392.4872	472.5872
73.0902	153.1902	233.2902	313.3902	393.4902	473.5902
74.0912	154.1912	234.2912	314.3912	394.4912	474.5912
75.0922	155.1922	235.2922	315.3922	395.4922	475.5922
76.0932	156.1932	236.2932	316.3932	396.4932	476.5932
77.0942	157.1942	237.2942	317.3942	397.4942	477.5942
78.0952	158.1952	238.2952	318.3952	398.4952	478.5952
79.0962	159.1962	239.2962	319.3962	399.4962	479.5962

TREASURE ISLAND - Dejan Ristanovic. You have come to an island looking for treasure. You have a map with a 10 x 10 grid locating the treasure and certain physical features of the island, but you don't know your starting point. You can move one square on the grid at a time in one of the cardinal directions. The calculator tells you of any physical feature of the square you are on. Your problem is to use the map and the information describing the squares you have been on to to deduce your position and move to the square with the treasure. One point is added to your score for each square you have occupied. The game ends when you reach the treasure. There are sharks in the water surrounding the island. If you try to go off the island a shark attack will occur. If you try to leave the island a second time you will be eaten, the game ends, and penalty points will be added to your score.

The Master Library module and a printer are required. The program to be loaded in banks 1 and 2 in the turn-on partitioning is listed on page 8. Constants must also be loaded in data registers 40 through 59 in bank 3 as listed below. More than ten digits are required in some cases.

43.44332321636	40	3241350021.	50
30.43032413137	41	2417271600.	51
37.13735171 7	42	9.99362313852	52
15.21513421 7	43	45324100.	53
35.23532152 6	44	1335170016.	54
75.93633131517	45	1713167300.	55
3637133537.	46	1624160024.	56
2020202020.	47	6200.062	57
43133600.	48	3773000000.	58
32310045.	49	7575757575.	59

```
+------------+
: ▲▲▲▲▲▲▲▲▲▲ :
: ▲▲▲▲▲▲▲▲▲W :
: ▲▲W▲▲▲▲▲▲C :
: ▲T▲▲▲▲▲▲▲▲ :
: ▲▲▲▲M▲▲▲▲▲ :
: ▲▲M▲X▲▲▲▲▲ :
: ▲▲▲▲▲▲▲▲▲W :
: M▲▲▲▲▲▲▲▲▲ :
: ▲▲MC▲▲▲R▲▲ :
: ▲▲▲▲RW▲▲▲▲ :
+------------+

SPACE ON YOUR FIELD
WOODS ON YOUR FIELD
-----SHARK-----
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
ROCK  ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD

 YOU DID IT?
    START WAS AT
        6.8
        13.     SCR.

SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
TREE  ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
MOUNT ON YOUR FIELD
SPACE ON YOUR FIELD

 YOU DID IT?
    START WAS AT
        1.8
        15.     SCR.

SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
SPACE ON YOUR FIELD
-----SHARK-----
SPACE ON YOUR FIELD
-----SHARK-----

 YOU ARE DEAD?
    START WAS AT
        1.2
        24.     SCR.
```

Instructions

1. Enter banks 1 through 3 in the start-up partitioning.

2. Enter a seed number in the display and press E. After about two minutes a 10 x 10 grid map will be printed. A sample printout for a starting seed of 16463 appears at the right. The code for the features on the map is:

△ means a space
W means woods
M means mount
T means tree
C means cave
R means rock
X means treasure

Treasure Island (cont)

3. The feature describing the square on which you start will be printed. You command a move to an adjacent square using the following code:

Press A to move one square west
Press B to move one square east
Press C to move one square north
Press D to move one square south

After a few seconds the feature describing your new position on the grid will be printed. If you have tried to go into the water a shark attack is indicated and you do not move. Continue to use keys A, B, C, and D to move one square at a time until you reach the treasure or suffer a second shark attack.

4. You can play again with the same map but a different starting position by pressing 2nd-C'. To play with a different map return to step 2 and use a different seed number.

5. For the three games with the map on page 6 the sequence of keys was, after entering the seed of 16463 in the display:

First Game: E, B, B, D, D, A, A, A, A, A, C, C, C

Second Game: 2nd-C', A, A, A, A, A, A, A, D, D, D, D, B, B, B

Third Game: 2nd-C', A, A, A, C, C

Editor's Comment: This program is an example of efficient use of memory. Consider the value in data register 40 of 44.4453232136 . The integer portion 44 is used to print the feature description on the map, in this case a W (program steps 184-188). The first digit to the right of the decimal point defines how many times the "woods" feature will appear on the map (program steps 189-199). The remaining ten digits are used with an HIR 05 command to print the WOODS portion of the description of the present position on the grid (program steps 391-408). You can change the number of occurrences of the various features by changing the first digit to the right of the decimal point at data registers 40 through 44. As you increase the number of features the time to generate the map will increase.

Now consider the value 6200.062 in data register 57 and program steps 253 through 258. After recall of the value to the display the HIR 08 command uses the four least significant digits to print the space and exchange symbol (code 62) at the right hand border of the map. The Op 01 command uses the four most significant digits to print the exchange symbol and the space at the left hand border. The zero at the ones digit is used for both print operations.

These techniques for packing print commands were first described by Clyde Durbin in the February 1978 issue of 52 Notes. Additional discussion appears in V5N3P14 of TI PPC Notes and in the May/June 1982 issue of PPX Exchange. The use of the exchange symbol for vertical borders, the dash (code 20) for horizontal borders, and the plus (code 47) for corners was suggested by Richard Snow in V4N2P6 of 52 Notes.

Treasure Island - (cont)

Program Listing:

000	98	ADV	080	32	32	160	00	00	240	04	4	320	25	25	400	50	50
001	43	RCL	081	61	GTO	161	43	RCL	241	00	0	321	29	CP	401	69	OP
002	53	53	082	95	=	162	59	59	242	32	X:T	322	77	GE	402	03	03
003	69	OP	083	76	LBL	163	72	ST*	243	43	RCL	323	03	03	403	43	RCL
004	01	01	084	14	D	164	00	00	244	00	00	324	55	55	404	51	51
005	00	0	085	69	OP	165	69	OP	245	77	GE	325	69	OP	405	69	OP
006	69	OP	086	22	22	166	20	20	246	01	01	326	00	00	406	04	04
007	04	04	087	61	GTO	167	43	RCL	247	84	84	327	43	RCL	407	69	OP
008	69	OP	088	95	=	168	00	00	248	02	2	328	47	47	408	05	05
009	05	05	089	76	LBL	169	32	X:T	249	00	0	329	69	OP	409	43	RCL
010	69	OP	090	19	D'	170	03	3	250	42	STO	330	02	02	410	01	01
011	00	00	091	43	RCL	171	09	9	251	00	00	331	69	OP	411	42	STO
012	43	RCL	092	19	19	172	77	GE	252	16	A'	332	04	04	412	10	10
013	46	46	093	65	×	173	01	01	253	43	RCL	333	43	RCL	413	43	RCL
014	69	OP	094	02	2	174	61	61	254	57	57	334	52	52	414	02	02
015	02	02	095	85	+	175	03	3	255	82	HIR	335	82	HIR	415	42	STO
016	43	RCL	096	02	2	176	01	1	256	08	08	336	07	07	416	11	11
017	48	48	097	00	0	177	22	INV	257	69	OP	337	69	OP	417	32	X:T
018	69	OP	098	95	=	178	44	SUM	258	01	01	338	05	05	418	91	R/S
019	03	03	099	42	STO	179	30	30	259	68	NOP	339	43	RCL	419	43	RCL
020	01	1	100	08	08	180	04	4	260	68	NOP	340	10	10	420	54	54
021	03	3	101	43	RCL	181	04	4	261	73	RC*	341	42	STO	421	69	OP
022	03	3	102	18	18	182	42	STO	262	00	00	342	01	01	422	02	02
023	07	7	103	32	X:T	183	00	00	263	69	OP	343	43	RCL	423	43	RCL
024	52	EE	104	04	4	184	73	RC*	264	02	02	344	11	11	424	55	55
025	06	6	105	77	GE	185	00	00	265	69	OP	345	42	STO	425	69	OP
026	22	INV	106	01	01	186	59	INT	266	20	20	346	02	02	426	03	03
027	52	EE	107	14	14	187	42	STO	267	73	RC*	347	87	IFF	427	02	2
028	69	OP	108	69	OP	188	04	04	268	00	00	348	01	01	428	49	PRD
029	04	04	109	28	28	189	73	RC*	269	69	OP	349	04	04	429	06	06
030	69	OP	110	05	5	190	00	00	270	03	03	350	19	19	430	01	1
031	05	05	111	22	INV	191	22	INV	271	69	OP	351	86	STF	431	02	2
032	43	RCL	112	44	SUM	192	59	INT	272	05	05	352	01	01	432	44	SUM
033	03	03	113	18	18	193	65	×	273	69	OP	353	25	CLR	433	06	06
034	99	PRT	114	73	RC*	194	01	1	274	20	20	354	91	R/S	434	81	RST
035	03	3	115	08	08	195	00	0	275	43	RCL	355	43	RCL	435	76	LBL
036	06	6	116	55	÷	196	95	=	276	00	00	356	01	01	436	16	A'
037	01	1	117	53	(	197	59	INT	277	32	X:T	357	22	INV	437	69	OP
038	05	5	118	01	1	198	42	STO	278	03	3	358	77	GE	438	00	00
039	03	3	119	00	0	199	03	03	279	09	9	359	03	03	439	43	RCL
040	05	5	120	75	-	200	17	B'	280	77	GE	360	25	25	440	47	47
041	04	4	121	02	2	201	42	STO	281	02	02	361	43	RCL	441	69	OP
042	00	0	122	65	×	202	18	18	282	61	61	362	01	01	442	02	02
043	69	OP	123	43	RCL	203	17	B'	283	16	A'	363	42	STO	443	69	OP
044	04	04	124	18	18	204	42	STO	284	98	ADV	364	18	18	444	03	03
045	43	RCL	125	54	)	205	19	19	285	76	LBL	365	43	RCL	445	04	4
046	06	06	126	22	INV	206	19	D'	286	18	C'	366	02	02	446	07	7
047	69	OP	127	28	LOG	207	32	X:T	287	98	ADV	367	42	STO	447	02	2
048	06	06	128	52	EE	208	07	7	288	17	B'	368	19	19	448	00	0
049	91	R/S	129	82	HIR	209	05	5	289	42	STO	369	19	D'	449	69	OP
050	76	LBL	130	04	04	210	22	INV	290	01	01	370	32	X:T	450	01	01
051	17	B'	131	95	=	211	67	EQ	291	17	B'	371	04	4	451	02	2
052	36	PGM	132	42	STO	212	02	02	292	42	STO	372	04	4	452	00	0
053	15	15	133	17	17	213	00	00	293	02	02	373	67	EQ	453	04	4
054	71	SBR	134	22	INV	214	43	RCL	294	85	+	374	04	04	454	07	7
055	88	DMS	135	59	INT	215	04	04	295	43	RCL	375	64	64	455	52	EE
056	65	×	136	22	INV	216	44	SUM	296	01	01	376	04	4	456	06	6
057	01	1	137	44	SUM	217	16	16	297	55	÷	377	05	5	457	22	INV
058	00	0	138	17	17	218	43	RCL	298	01	1	378	42	STO	458	52	EE
059	95	=	139	65	×	219	16	16	299	00	0	379	05	05	459	69	OP
060	59	INT	140	01	1	220	55	÷	300	95	=	380	73	RC*	460	04	04
061	92	RTN	141	00	0	221	01	1	301	42	STO	381	05	05	461	69	OP
062	76	LBL	142	00	0	222	00	0	302	03	03	382	59	INT	462	05	05
063	12	B	143	95	=	223	00	0	303	25	CLR	383	67	EQ	463	92	RTN
064	69	OP	144	42	STO	224	85	+	304	42	STO	384	03	03	464	43	RCL
065	21	21	145	16	16	225	43	RCL	305	06	06	385	91	91	465	56	56
066	76	LBL	146	59	INT	226	17	17	306	69	OP	386	69	OP	466	69	OP
067	95	=	147	22	INV	227	95	=	307	26	26	387	35	35	467	02	02
068	61	GTO	148	44	SUM	228	65	×	308	01	1	388	61	GTO	468	43	RCL
069	03	03	149	16	16	229	82	HIR	309	00	0	389	03	03	469	58	58
070	06	06	150	22	INV	230	14	14	310	32	X:T	390	80	80	470	69	OP
071	76	LBL	151	52	EE	231	95	=	311	43	RCL	391	73	RC*	471	03	03
072	11	A	152	92	RTN	232	72	ST*	312	01	01	392	05	05	472	81	RST
073	69	OP	153	76	LBL	233	08	08	313	77	GE	393	82	HIR	473	00	0
074	31	31	154	15	E	234	97	DSZ	314	03	03	394	05	05	474	00	0
075	61	GTO	155	42	STO	235	03	03	315	25	25	395	43	RCL	475	00	0
076	95	=	156	09	09	236	02	02	316	43	RCL	396	49	49	476	00	0
077	76	LBL	157	02	2	237	00	00	317	02	02	397	69	OP	477	00	0
078	13	C	158	00	0	238	69	OP	318	77	GE	398	02	02	478	00	0
079	69	OP	159	42	STO	239	30	30	319	03	03	399	43	RCL	479	00	0

DETERMINANTS FOR THE TI-57 - Reginald van Genechten

This program which will solve second and third order determinants on the TI-57 is one more demonstration of the computing power of that little device.

Given a matrix A of the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

then with the program at the right in the calculator and for the third order determinant:

1. Enter a 3, the order of the matrix, in the display and press RST R/S.
2. Enter the value of a_{11} and press R/S;
   Enter the value of a_{12} and press R/S;
   Enter the value of a_{13} and press R/S;
   Enter the value of a_{21} and press R/S;
   and so on, continuing to enter the elements of the matrix by rows. When the a_{33} element has been entered, the value of the determinant of the matrix will be returned to the display.
3. To solve another third order determinant you need only enter the order and press R/S. The RST is provided by program step 41.

For a second order determinant with the calculator in the start-up condition with zeroes in all of the data registers:

1. Enter a 2, the order of the matrix, into the display and press RST R/S.
2. Enter the value of a_{11} and press R/S.
   Enter the value of a_{12} and press R/S.
   Enter the value of a_{21} and press R/S.
   Enter the value of a_{22} and press R/S. The value of the determinant will be returned to the display.
3. To solve a second order determinant in the shadow of a previous second or third order solution it is necessary to clear the data registers, say with a keyboard sequence INV 2nd C.t , before entering the order and pressing R/S.

```
00   32 7       STO 7
01   02           2
02   66         X=T
03   51 1       GTO 1
04   81         R/S
05   32 0       STO 0
06   32 1       STO 1
07   81         R/S
08   32 2       STO 2
09   32 3       STO 3
10   81         R/S
11   32 4       STO 4
12   32 5       STO 5
13   81         R/S
14   39 5       PRD 5
15   39 2       PRD 2
16   81         R/S
17   39 0       PRD 0
18   39 4       PRD 4
19   81         R/S
20   39 3       PRD 3
21   39 1       PRD 1
22   33 4       RCL 4
23  -34 3  INV  SUM 3
24   81         R/S
25   39 3       PRD 3
26   33 5       RCL 5
27  -34 1  INV  SUM 1
28   86 2       LBL 2
29   81         R/S
30   39 1       PRD 1
31   33 1       RCL 1
32  -34 3  INV  SUM 3
33   33 2       RCL 2
34  -34 0  INV  SUM 0
35   81         R/S
36   39 0       PRD 0
37   33 0       RCL 0
38   34 3       SUM 3
39   33 3       RCL 3
40   81         R/S
41   71         RST
42   86 1       LBL 1
43   81         R/S
44   32 0       STO 0
45   81         R/S
46   32 1       STO 1
47   51 2       GTO 2
```

Editor's Note: The requirement to clear the data registers with a keyboard sequence before proceeding with a second order solution in the shadow of a previous solution is imposed by program steps 32, 33, 38, and 39 which perform arithmetic with the contents of data registers 2 and 3 without first clearing them. The need for keyboard clearing can be eliminated by inserting an INV 2nd C.t command at step 43, immediately after LBL 1. The INV 2nd C.t command appears as -19 in LRN mode.

THE USE OF INS IN A PROGRAM - Palmer Hanson. Peter Stromgren of Denmark asked "What does a programmed INS do?" The only known use for an INS (code 46) in a program is as a dummy operator, say to move a variable inside parentheses in the same manner as a CE (see page V-15 of Personal Programming). Ralph Snyder asks why programmers use INS for this purpose when it requires eight keystrokes (for example, RCL-4-6-Bst-Bst-2nd-Del-SST) to synthesize an INS versus only one keystroke for a CE. Examples of the use of INS appear at step 098 in David Mah's Decimal/Octal conversion program (V9N1P12) and at steps 003 and 007 in Robert Prins' solution to the alpha code challenge (V9N1P18).

The idea comes from early issues of TI PPC Notes. V5N6P3 reported that Philip Rowley of the British TI newsletter had found that IND (code 40) can be used as a dummy operator instead of CE, and IND does not clear an existing error condition. In V5N8P2 I reported that INS (code 46) can also be used as a dummy operator to avoid clearing error indications, and has the advantages that (1) it is not used in a program for any other function, and (2) "Insert" is an accurate description of what is being done. In most cases users can safely replace an INS with a CE.

The TI-66 does not require the use of a dummy operator to bring a value inside parentheses.

BOOK REVIEW - The Calculator Puzzle Book. Claude Birtwistle. 1978 Bell Publishing Co. Hardbound, 125 pages.

The Educalc Mail Store catalog issue #21 has a special sale on this book at $2.50 plus a $1.00 shipping and handling charge. That is a real bargain for a nicely written little book which contains 94 puzzles which can be solved on a standard four-function calculator. The puzzles introduce subjects ranging from Fibonacci series, prime numbers, and palindromic numbers to square roots by successive approximation. The book would be an appropriate gift for a child who is just beginning to show an interest in calculators and mathematics. Order from Educalc Mail Store, 27953 Cabot Road, Laguna Niguel, CA 92677.

PRINTER PAPER - In V9N1P2 I announced that I had found a source of old style printer paper and volunteered to deliver it to members for a nominal price. Hewlett Ladd wrote to report that he has been receiving excellent quality printer paper from Elek-Tek. I ordered some and found that it prints very much "black-on-white" as compared with the "black-on-tan" of the old style paper. Some copiers seem to like the "black-on-tan" better. Current Elek-Tek prices for TI-59 related items are:

TP-30250	3-roll pack printer paper	$ 6.50
BP-1A	Battery Pack	8.00
BC-59	40 Mag cards and case	11.00

where the magnetic card price is clearly not a bargain as long as J M. Gallego has the same item for sale for eight dollars. Elek-Tek has a four dollar shipping and handling charge for the first item ordered, with an additional dollar for each additional item, but only fifty cents for additional battery packs or magnetic cards. The address is Elek-tek, Inc., 6557 North Lincoln Ave., Chicago, IL 60645. You can place a credit card order by calling 800-621-1269 toll free.

SIN(X) AND COS(X) TO THIRTY-SIX DIGITS - Peter Messer

This fast mode program is based on the following algorithms by Laurance Leeds:

$$\sin(x) = x(1 - \frac{x^2}{2\cdot 3}(1 - \frac{x^2}{4\cdot 5}(1 - \frac{x^2}{6\cdot 7}(\cdots \quad \frac{x^2}{38\cdot 39}$$

$$\cos(x) = 1 - \frac{x^2}{1\cdot 2}(1 - \frac{x^2}{3\cdot 4}(1 - \frac{x^2}{5\cdot 6}(\cdots \quad \frac{x^2}{37\cdot 38}$$

where the unique feature of the algorithms is that, given the entry x, the only power of x that needs to be found is the square of x. The program finds the sine or cosine correct to 36 digits over the range $0 \le x \le \pi/2$.

Instructions:

1. Enter banks 1 and 2 of the program listed on page 13. The listing as shown is for the calculator only mode.

2. Press A to initialize. See a zero in the display.

3. Enter the value of x in radians. x is entered in six blocks of digits where the first block may hold six or seven digits depending upon whether $x \geq 1$, and the remaining five blocks hold six digits. A decimal point is not entered, but there is an implied decimal point between the first and second digits of the first block. As each block is entered, press R/S, see the entered value reappear in the display, and proceed to enter the next block. Trailing zeroes must be entered in a given block, but blocks of trailing zeroes do not need to be entered. Thus to enter one radian, the user enters the value 1000000 and presses R/S. No further entries need be made before proceeding to step 4.

4. To find sin(x) press B; or to find cos(x) press C. The calculator will stop with a flashing 1 in the display. Do not clear the display, but press 7 and then EE. The calculator will run for about seven minutes and stop with the first block of six digits of the solution in the display. Press R/S again and again to see the remaining blocks. When R/S is pressed after the sixth block has been displayed, a flashing zero will appear in the display. The output format is the same as the input format.

5. To display the solution a second time, press CE to clear the flashing display, and then press SBR-4-4-4.

6. To start a new problem press CE to clear the flashing display and return to step 2.

7. With a printer attached you may obtain automatic printout of all six blocks of the solution by changing the R/S at program step 457 to Nop. With a printer the input blocks of x are also printed. To suppress printing of the input change the Print at program step 050 to a Nop.

Accuracy:

The program is expected to be accurate to one digit in the last place. Sample solutions appear on the next page. We hope to publish a program to convert multi-precision degrees to multi-precision radians in a subsequent issue.

Sin(x) and cos(x) to Thirty-six Digits - (cont)

Sample Results:

A sample printout of the solution for the sin(π/3) is at the right. The first thirty digits agree with the longest table I have readily available. The cosine of π/6 yields an identical result. The sine and cosine of π/4 differ by one in the least significant digit. The cosine of π/3 and the sine of π/6 are returned as exactly 0.5 . The sine of π/2 is returned as all nines; but changing the least significant digit of π/2 from a 2 to a 3, which is not correct in a rounded sense, will then return the sine as a one followed by 36 zeroes. All these results are consistent with an accuracy to within one in the thirty-sixth decimal place.

```
1047197.
 551196.
 597746.
 154214.
 461093.
 167628.

 866025.
 403784.
 438646.
 763723.
 170752.
 936184.
```

Editor's Notes:

This program contains several examples of efficient programming:

(1) The 45-tan-x² sequence at program steps 475 through 478 obtain the fast mode initialization constant 1 + 2E-12 in a minimum number of steps. The "1" sets flag 4 and resets flag 9 at fast mode entry so thre is no problem with dropping into TRACE at fast mode exit.

(2) The fast mode entry constant causes a jump to program step 001 at fast mode entry. A GTO 193 sequence at program locations 001 through 003 is used to transfer to the desired program location for execution in fast mode. This technique is about five steps shorter than the use of an entry constant which permits direct transfer to the fast mode starting point. It may not be as elegant as the direct transfer, but it is an efficient use of memory.

(3) The R/S at program step 000 is not used.

(4) The program uses a RTN to leave fast mode as described in V9N1P15 and V9N2P3. The SBR call prior to fast mode entry is at program steps 014 through 016. The RTN at program step 462 causes the program to leave fast mode and return to step 017.

(5) The subroutine call E at program step 018 provides a single step capability to stop the program and give an error indication, since there is no LBL E in the program.

THE ROBERT PRINS BRAINTEASER REVISITED - V9N2P13 defined the problem as starting from the turnon condition you are to create a flashing one in the display. The last key you press from the keyboard should be CLR, but you are not allowed to use GTO or SBR immediately before it. To make things a little harder, the only functions or keys you may use are 2nd, INV, CLR, LRN, RCL, STO, SUM, PRD, GTO and R/S, but you may use R/S only once. Robert pointed out that my so-called solution was not acceptable since I had used Lbl (2nd-SBR) and that was not in the allowable list of functions. Other members had submitted similar solutions which avoided the prohibition on SBR or GTO immediately before CLR by the use of the 2nd-CLR (code 20). Several members correctly used the INV-PRD sequence to obtain a divide by zero to obtain a one. As promised Robert sent his solution. It uses only the functions listed. He says the problem was inspired by V3N4P6 of 52 Notes. That hint was not much help to me. An end-of-partition quirk is involved.

Sin(x) and cos(x) to Thirty-six Digits - (cont)

Program Listing:

000	91	R/S
001	61	GTO
002	01	01
003	93	93
004	76	LBL
005	12	B
006	69	OP
007	28	28
008	69	OP
009	29	29
010	69	OP
011	36	36
012	76	LBL
013	13	C
014	71	SBR
015	04	04
016	75	75
017	25	CLR
018	15	E
019	76	LBL
020	11	A
021	47	CMS
022	01	1
023	52	EE
024	06	6
025	42	STO
026	59	59
027	04	4
028	09	9
029	42	STO
030	01	01
031	01	1
032	09	9
033	42	STO
034	00	00
035	42	STO
036	07	07
037	09	9
038	42	STO
039	03	03
040	03	3
041	07	7
042	42	STO
043	08	08
044	03	3
045	08	8
046	42	STO
047	09	09
048	25	CLR
049	91	R/S
050	99	PRT
051	69	OP
052	21	21
053	69	OP
054	20	20
055	69	OP
056	23	23
057	72	ST*
058	01	01
059	72	ST*
060	00	00
061	72	ST*
062	03	03
063	61	GTO
064	00	00
065	49	49
066	02	2
067	00	0
068	42	STO
069	02	02
070	02	2
071	01	1
072	42	STO
073	03	03
074	04	4
075	00	0
076	42	STO
077	04	04
078	06	6
079	42	STO
080	05	05
081	43	RCL
082	08	08
083	65	×
084	43	RCL
085	09	09
086	95	=
087	42	STO
088	00	00
089	73	RC*
090	02	02
091	55	÷
092	43	RCL
093	00	00
094	95	=
095	59	INT
096	72	ST*
097	04	04
098	65	×
099	43	RCL
100	00	00
101	75	-
102	73	RC*
103	02	02
104	95	=
105	94	+/-
106	65	×
107	43	RCL
108	59	59
109	95	=
110	74	SM*
111	03	03
112	69	OP
113	22	22
114	69	OP
115	23	23
116	69	OP
117	24	24
118	97	DSZ
119	05	05
120	00	00
121	89	89
122	02	2
123	22	INV
124	44	SUM
125	08	08
126	22	INV
127	44	SUM
128	09	09
129	02	2
130	05	5
131	42	STO
132	02	02
133	04	4
134	05	5
135	42	STO
136	03	03
137	06	6
138	42	STO
139	05	05
140	25	CLR
141	75	-
142	73	RC*
143	03	03
144	95	=
145	72	ST*
146	02	02
147	22	INV
148	77	GE
149	04	04
150	63	63
151	69	OP
152	33	33
153	69	OP
154	32	32
155	97	DSZ
156	05	05
157	01	01
158	40	40
159	97	DSZ
160	07	07
161	01	01
162	93	93
163	43	RCL
164	06	06
165	77	GE
166	04	04
167	44	44
168	06	6
169	42	STO
170	05	05
171	09	9
172	42	STO
173	00	00
174	04	4
175	09	9
176	42	STO
177	01	01
178	22	INV
179	86	STF
180	04	04
181	69	OP
182	20	20
183	69	OP
184	21	21
185	73	RC*
186	01	01
187	72	ST*
188	00	00
189	97	DSZ
190	05	05
191	01	01
192	81	81
193	43	RCL
194	10	10
195	65	×
196	43	RCL
197	20	20
198	95	=
199	42	STO
200	30	30
201	43	RCL
202	11	11
203	65	×
204	43	RCL
205	20	20
206	85	+
207	43	RCL
208	10	10
209	65	×
210	43	RCL
211	21	21
212	95	=
213	42	STO
214	31	31
215	43	RCL
216	12	12
217	65	×
218	43	RCL
219	20	20
220	85	+
221	43	RCL
222	11	11
223	65	×
224	43	RCL
225	21	21
226	85	+
227	43	RCL
228	10	10
229	65	×
230	43	RCL
231	22	22
232	95	=
233	42	STO
234	32	32
235	43	RCL
236	13	13
237	65	×
238	43	RCL
239	20	20
240	85	+
241	43	RCL
242	12	12
243	65	×
244	43	RCL
245	21	21
246	85	+
247	43	RCL
248	11	11
249	65	×
250	43	RCL
251	22	22
252	85	+
253	43	RCL
254	10	10
255	65	×
256	43	RCL
257	23	23
258	95	=
259	42	STO
260	33	33
261	43	RCL
262	14	14
263	65	×
264	43	RCL
265	20	20
266	85	+
267	43	RCL
268	13	13
269	65	×
270	43	RCL
271	21	21
272	85	+
273	43	RCL
274	12	12
275	65	×
276	43	RCL
277	22	22
278	85	+
279	43	RCL
280	11	11
281	65	×
282	43	RCL
283	23	23
284	85	+
285	43	RCL
286	10	10
287	65	×
288	43	RCL
289	24	24
290	95	=
291	42	STO
292	34	34
293	43	RCL
294	15	15
295	65	×
296	43	RCL
297	20	20
298	85	+
299	43	RCL
300	14	14
301	65	×
302	43	RCL
303	21	21
304	85	+
305	43	RCL
306	13	13
307	65	×
308	43	RCL
309	22	22
310	85	+
311	43	RCL
312	12	12
313	65	×
314	43	RCL
315	23	23
316	85	+
317	43	RCL
318	11	11
319	65	×
320	43	RCL
321	24	24
322	85	+
323	43	RCL
324	10	10
325	65	×
326	43	RCL
327	25	25
328	95	=
329	42	STO
330	35	35
331	43	RCL
332	15	15
333	65	×
334	43	RCL
335	21	21
336	85	+
337	43	RCL
338	14	14
339	65	×
340	43	RCL
341	22	22
342	85	+
343	43	RCL
344	13	13
345	65	×
346	43	RCL
347	23	23
348	85	+
349	43	RCL
350	12	12
351	65	×
352	43	RCL
353	24	24
354	85	+
355	43	RCL
356	11	11
357	65	×
358	43	RCL
359	25	25
360	95	=
361	42	STO
362	36	36
363	00	0
364	42	STO
365	29	29
366	03	3
367	06	6
368	42	STO
369	04	04
370	03	3
371	05	5
372	42	STO
373	03	03
374	02	2
375	07	7
376	42	STO
377	02	02
378	07	7
379	42	STO
380	05	05
381	73	RC*
382	04	04
383	55	÷
384	43	RCL
385	59	59
386	95	=
387	75	-
388	59	INT
389	74	SM*
390	03	03
391	95	=
392	65	×
393	43	RCL
394	59	59
395	95	=
396	72	ST*
397	02	02
398	69	OP
399	32	32
400	69	OP
401	33	33
402	69	OP
403	34	34
404	97	DSZ
405	05	05
406	03	03
407	81	81
408	43	RCL
409	29	29
410	42	STO
411	20	20
412	87	IFF
413	01	01
414	04	04
415	40	40
416	42	STO
417	10	10
418	43	RCL
419	21	21
420	42	STO
421	11	11
422	43	RCL
423	22	22
424	42	STO
425	12	12
426	43	RCL
427	23	23
428	42	STO
429	13	13
430	43	RCL
431	24	24
432	42	STO
433	14	14
434	43	RCL
435	25	25
436	42	STO
437	15	15
438	86	STF
439	01	01
440	87	IFF
441	04	04
442	00	00
443	66	66
444	01	1
445	09	9
446	42	STO
447	00	00
448	06	6
449	42	STO
450	05	05
451	98	ADV
452	69	OP
453	20	20
454	73	RC*
455	00	00
456	99	PRT
457	91	R/S
458	97	DSZ
459	05	05
460	04	04
461	52	52
462	92	RTN
463	43	RCL
464	59	59
465	74	SM*
466	02	02
467	69	OP
468	33	33
469	01	1
470	74	SM*
471	03	03
472	61	GTO
473	01	01
474	53	53
475	04	4
476	05	5
477	30	TAN
478	33	X^2
479	86	STF

NEWCOMER'S CORNER - 13 DIGIT REGISTER LIST PROGRAMS - P. Hanson

(Note: This review was prompted by Hewlett Ladd's discovery that Richard Snow's 13 digit register list program will not print the thirteenth digit when the most significant digit is a nine.)

The sequence INV-List from the keyboard or in a program will print the contents of all the data registers beginning with the number in the display when the sequence is called. The number of the data register is printed to the right of the contents (Personal Programming page VI-4). The printout is limited to the ten most significant digits, or to the eight most significant digits and the exponent if the EE format is required. As users became familiar with the use of the guard digits it became convenient to be able to list the complete contents of the data registers, including all thirteen digits of the mantissa, the exponent if in the EE format, and the data register number, all on one line. Examples include the listing of the constants used for Patrick Acosta's calendar printing program (page 2) and for Dejan Ristanovic's Treasure Island game (page 6 of this issue).

V5N9/10 published three thirteen digit register list programs:

1. Richard Snow's program (V5N9/10P17) which can list registers 01 through 89 at a rate of 29 seconds per register. The program can also list registers 90 through 99, but only if the user manually re-entered the data after loading the program. Register 00 cannot be listed. The program lists from a user controlled beginning register to the end of the partition. Data registers which contain zero are not listed, a feature I find to be a nuisance. The program has more deficiencies than the one found by Hewlett Ladd. There is no good reason to use the program as it presently stands.

2. Clyde Durbin's program (V5N9/10P17) which can list registers 00 through 85 at a rate of 26 seconds per register. The user can control both the beginning and ending registers. If mantissa/exponent notation is required only the eight most significant digits of the mantissa are printed. There is also one inconvenience--trailing zeroes are printed.

3. The Worthington/Regelman fast mode program (V5N9/10P15) which can list registers 00 through 99 at a rate of 23 seconds per register. The contents must be stored on magnetic cards which are then re-entered under program control. The beginning and ending registers can be controlled. A left justification option is available, but you must change program step 025 from CLR to Pause if the program is to operate in accordance with the instructions. Some unusual, but not incorrect, printouts can occur.

The May/June 1982 issue of PPX Exchange contains a program by M. S. Markusson which accomplishes a similar register list function. The data register number is at the left margin, and all printouts are in a mantissa/exponent notation. The program will print registers 00 through 89 at a rate of about 17 seconds per register, and it is not a fast mode program. Only the beginning register is controlled. If you want to use this program you must include the addition which is on page 9 of the July/August 1982 issue of PPX Exchange.

Printouts of data registers 00 through 29 for the various printing routines appear on the next page. The data register contents were selected to exercise various idiosyncrasies of the programs.

INV-List		Durbin (V5N9/10P17)		Snow (V5N9/10P17)		Worthington/Regelman (V5N9/10P15) Justified		Worthington/Regelman (V5N9/10P15) Normal		Markusson (PPX Exchange)		
						0.29		0.29				
3.141592654	00	3.141592653590	00			3.14159265359	00	3.14159265359	00	00	3.141592653590	00
1.111111111	01	1.111111111111	01	1.11111111111	01	1.11111111111	01	1.11111111111	01	01	1.11111111111	00
2.222222222	02	2.222222222222	02	2.22222222222	02	2.22222222222	02	2.22222222222	02	02	2.22222222222	00
3.333333333	03	3.333333333333	03	3.33333333333	03	3.33333333333	03	3.33333333333	03	03	3.33333333333	00
4.444444444	04	4.444444444444	04	4.44444444444	04	4.44444444444	04	4.44444444444	04	04	4.44444444444	00
5.555555556	05	5.555555555555	05	5.55555555555	05	5.55555555555	05	5.55555555555	05	05	5.55555555555	00
6.666666667	06	6.666666666666	06	6.66666666666	06	6.66666666666	06	6.66666666666	06	06	6.66666666666	00
7.777777778	07	7.777777777777	07	7.77777777777	07	7.77777777777	07	7.77777777777	07	07	7.77777777777	00
8.888888889	08	8.888888888888	08	8.88888888888	08	8.88888888888	08	8.88888888888	08	08	8.88888888888	00
10.	09	9.999999999999	09	9.9999999999	09	9.99999999999	09	9.99999999999	09	09	9.99999999999	00
1.	10	.999999999999	10	0.99999999999	10	.999999999999-00	10	.999999999999-00	10	10	9.99999999999	-01
9.869604401	11	9.869604401090	11	9.86960440109	11	9.86960440109	11	9.86960440109	11	11	9.869604401090	00
-3.1415927-55	12	-3.1415927-55	12	-0.	12	-3.1415927-55	12	-3.1415927-55	12	12	-3.141592700000	-55
0.	13	0.	13			0.	13	0.	13	13	0.000000000000	00
0.	14	0.	14			0.	14	0.	14	14	0.000000000000	00
3.141592654	15	3.141592653590	15	3.14159265359	15	3.14159265359	15	3.14159265359	15	15	3.141592653590	00
1.	16	1.000000000000	16	1.	16	1.	16	1.	16	16	1.000000000000	00
2.718281828	17	2.718281828459	17	2.718281828459	17	2.718281828459	17	2.718281828459	17	17	2.718281828459	00
4.3170165 63	18	4.3170165 63	18	4317016463003O	18	4.317016463003 63	18	4.317016463003 63	18	18	4.317016463003	63
.1111111111	19	.1111111111111	19	0.11111111111	19	.1111111111111	19	.1111111111111	19	19	1.111111111111	-01
.7777777778	20	.7777777777777	20	0.77777777777	20	.7777777777777	20	.7777777777777	20	20	7.77777777777	-01
-.5555555556	21	-.5555555555555	21	-0.55555555555	21	-.5555555555555	21	-.5555555555555	21	21	-5.55555555555	-01
123456789.	22	123456789.0000	22	123456789.	22	123456789.	22	123456789.	22	22	1.234567890000	08
.0000000001	23	1.23-10	23	0.00000000123	23	.00000000123	23	.00000000123	23	23	1.230000000000	-10
1.23 14	24	1.23 14	24	1230000000000	2.	123000000000000.	24	1.23 14	24	24	1.230000000000	14
.4971498727	25	.4971498726941	25	0.497149872694	25	.4971498726941	25	.4971498726941	25	25	4.971498726941	-01
1.	26	.9999999999950	26	0.99999999995	26	.999999999995	26	.999999999995	26	26	9.999999999950	-01
1.23-14	27	1.23-14	27	0.	27	.0000000000000123	27	1.23-14	27	27	1.230000000000	-14
0.	28	0.	28			0.	28	0.	28	28	0.000000000000	00
0.	29	0.	29			0.	29	0.	29	29	0.000000000000	00

<u>Comments on printouts:</u>

Durbin - R18 shows only eight digits when EE mode applies. Trailing zeroes are seen in R00, R16, R22, R24, R26 and R27.

Snow - R09 shows missing 13th digit when first digit is a nine. R10 and R25 show missing 13th digit when the printout is of form 0.XXX... . R13 and 14 show spillover of the decimal point into the register number for some exponentials. R18 shows 14 digits, but no indication that number is much larger. No printouts for R19, R20, R21, R28 and R29 which contain zeroes.

Worthington/Regelman Justified - R10 shows exponent of -00. R13 through R15 show more than thirteen digits; magnitudes are correct.

Worthington/Regelman Normal - R10 shows exponent of -00.

Markusson - No anomalies.

<u>THIRTEEN DIGIT VIEWER</u> - Laurance Leeds. Previous issues have published hidden digit viewers and guard digit viewers, but V9N2P2 indicated that none seemed satisfactory for all situations. The following program seems to handle all of the recognized problem cases, and provides a multi-stage display which ensures that there can be no confusion with leading zeroes:

```
000  76 LBL    024  22 INV    048  01  1     072  65  ×
001  11  A     025  28 LOG    049  52 EE     073  01  1
002  42 STO    026  52 EE     050  01  1     074  00  0
003  00  00    027  22 INV    051  00  0     075  00  0
004  29 CP     028  52 EE     052  95  =     076  00  0
005  50 I×I    029  95  =     053  42 STO    077  95  =
006  28 LOG    030  42 STO    054  03  03    078  42 STO
007  77  GE    031  02  02    055  22 INV    079  03  03
008  00  00    032  59 INT    056  52 EE     080  59 INT
009  16  16    033  67  EQ    057  59 INT    081  91 R/S
010  59 INT    034  00  00    058  91 R/S    082  43 RCL
011  75  -     035  45  45    059  43 RCL    083  03  03
012  01  1     036  43 RCL    060  03  03    084  22 INV
013  95  =     037  02  02    061  22 INV    085  59 INT
014  42 STO    038  55  ÷     062  59 INT    086  65  ×
015  01  01    039  01  1     063  65  ×     087  01  1
016  59 INT    040  00  0     064  01  1     088  52 EE
017  42 STO    041  95  =     065  00  0     089  01  1
018  01  01    042  61 GTO    066  00  0     090  00  0
019  43 RCL    043  00  00    067  00  0     091  95  =
020  00  00    044  30  30    068  95  =     092  22 INV
021  55  ÷     045  43 RCL    069  91 R/S    093  52 EE
022  43 RCL    046  02  02    070  43 RCL    094  91 R/S
023  01  01    047  65  ×     071  02  02    095  00  0
```

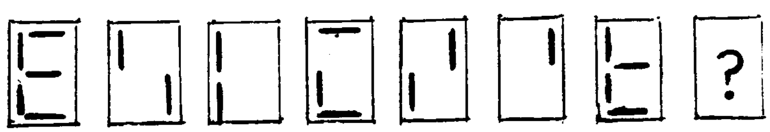
To run the program and view all thirteen digits of the value in the display, press A. The calculator will stop with the first ten digits in the display. Press R/S and see the last three digits. Press R/S again and see the first three digits. Press R/S a final time and see the last ten digits.

A modification of the program provides an even more error free readut:

```
000  76 LBL    017  42 STO    034  00  00    051  00  0
001  11  A     018  01  01    035  45  45    052  95  =
002  42 STO    019  43 RCL    036  43 RCL    053  22 INV
003  00  00    020  00  00    037  02  02    054  52 EE
004  29 CP     021  55  ÷     038  55  ÷     055  75  -
005  50 I×I    022  43 RCL    039  01  1     056  59 INT
006  28 LOG    023  01  01    040  00  0     057  91 R/S
007  77  GE    024  22 INV    041  95  =     058  95  =
008  00  00    025  28 LOG    042  61 GTO    059  65  ×
009  16  16    026  52 EE     043  00  00    060  01  1
010  59 INT    027  22 INV    044  30  30    061  00  0
011  75  -     028  52 EE     045  43 RCL    062  95  =
012  01  1     029  95  =     046  02  02    063  61 GTO
013  95  =     030  42 STO    047  65  ×     064  00  00
014  42 STO    031  02  02    048  01  1     065  55  55
015  01  01    032  59 INT    049  52 EE     066  00  0
016  59 INT    033  67  EQ    050  01  1     067  00  0
```

To run the program and view all thirteen digits of the value in the display, press A. The calculator will stop with the first ten digits in the display. Press R/S and see the eleventh digit. Press R/S and see the twelfth digit. Press R/S a third time and see the thirteenth digit. Thus you obtain all the digits, including leading and trailing zeroes. There is no need to count digits.

<u>BRAINTEASER</u> - Kelley Stanage of HP (George Thomson's niece) asks: What goes in the last space? Why?



IT'S THE TI-MBA THAT'S WRONG - Lem Matteson. V9N1P4 described the lack of A.O.S. in the BA-55, and noted that the earlier business programmable, the TI-MBA did use A.O.S. The BA-55 uses what is known as "Adding Machine Logic" where each previous operation is completed because the mechanical adding machines had to operate that way. Business formulas and books are written for this format, and adding machine logic is assumed. So, for business use, it is the TI-MBA which is wrong.

THE ProStar ELECTRONIC FLIGHT COMPUTER - Dave Leising forwarded the brochure for this $225 device available from Jeppesen Sanderson, 55 Inverness Drive East, Englewood, CO 80112-5499. The unit solves time, speed and distance problems; altitude, airspeed and wind problems; latitude and longitude, great circle bearing and distance; Mach number, true altitude, crosswind component, and weight and balance. It has 34 direct conversions and two user available memories. At first glance Dave and I thought that with all that computing power the device might be a resurrected TI-88. But a closer look shows that it calculates to eleven digits and displays eight, has smaller mode prompters under the numerical display, has OFF and ON/C keys, and a tilted keyboard--all features of the TI-55II, TI-57LCD and BA-55 class of calculators. Even the dimensions are identical with those devices. There is no indication of a programmable capability. Interested readers might want to call Jeppesen Sanderson at (303)-799-9090.

MEMORY PROTECTION ON THE CC-40 - V9N1P19 discussed memory protection on the CC-40 during replacement of the batteries. Maurice Swinnen had reported successful changes without losing memory when the time to replace was less than a minute. In late May I purchased an AC Adapter for my CC-40 from Educalc (Stock No. AC-9201, $14.95 plus shipping and handling). Just in time! In mid June the battery low indicator appeared on my CC-40. I connected the AC adapter, replaced the batteries at a leisurely pace, and found no loss of memory. Further experiments showed that the CC-40 will work satisfactorily with either the batteries or the AC adapter, whichever is available. If the batteries are installed, and you connect the AC adapter cable, but do not plug into AC power, the CC-40 still runs from battery power.

That feature is not available with some other portables. The Radio Shack Model 100 mechanization disconnects the batteries when the AC power adapter is connected. The instructions are very explicit--first, you connect the adapter to an AC outlet, then you connect the adapter cable to the computer. If the adapter cable is connected to the computer without a connection to AC power the computer will not operate. Memory is held up by the NiCad memory retention battery. This would seem to permit a condition in which inadvertently leaving the adapter connected to the computer and not connected to AC power could eventually cause a loss of memory as the NiCad battery runs down. I have written to Radio Shack for information. I have also written to TI for approval of the use of the AC Adapter during battery replacement.

DMS, INV DMS CHALLENGE - V9N1P18 contained Robert Prins' suggestion that optimized DMS and INV DMS routines might be useful in fast mode programs. Ralph Snyder reminds me that the Letters to the Editor column in the July/August 1981 issue of PPX Exchange carried an extended discussion of such optimization. Ralph was the author.

FAST MODE SCRIBBLER - Dejan Ristanovic. This program will allow you to produce any drawing (scribble) with your TI-59/PC-100. You may combine as many tapes as you wish. The dimensions of the drawing are not limited--the more mag cards you are willing to use for intermediate storage, the larger the picture you can produce. The idea for this program came from Frederic De Mees' Mickey Mouse program (V6N6/7P14). The printing portion of the program uses Michael Sperber's graphics mode to reduce the space between lines and produce a denser picture, and Patrick Acosta's h12 fast mode entry technique to reduce execution time. The program also provides space for program modifications so that you can add text, change the graphics symbols, etc.

The Fast Mode Scribbler consists of two separate programs which are combined to fit on two magnetic cards. Additional magnetic cards are required for intermediate data storage. The first program is a data assembler which converts the user input to a format which can be read by the second program, a printer control program. The data assembler listing appears on page 20. The printer program including the contents of data registers 30 through 69 appears on pages 20 and 21. There must be a zero in data register 30. Both programs should be recorded in the start-up partitioning, even though you will want to go to 7-Op-17 to enter the contents of the data registers.

To produce a drawing you first must superimpose a grid on the drawing. Graph paper which is ruled 10 x 10 lines to the inch is handy--it will provide an almost exact size duplicate from the printer. The selected grid should consist of an array of m x n bits. m and n are practically unlimited, but m must be divisible by 20, since the m dimension is divided into tapes which are twenty characters wide. Once the grid is overlaid you need to mark off the tapes (20 bits wide) and four segments in each tape (5 bits each) which will correspond to the Op 01 through Op 04 positions of the printer. In the remaining discussion each group of twenty characters will be called a line. You will enter the individual bits of each line in groups of five. If you enter a one, the printer program will print a small zero at that position. If you enter a zero, the printer program will print a space at that position.

At various points in the operation of the program the display will become blank except for a steady but faint C at the left hand edge. This means that the data accumulation registers (00 to 29) are full and need to be recorded on a magnetic card to make room for more data. (The calculator is stopped at the Write at step 229 of the program waiting for you to insert a magnetic card.) When the card has been written, the program returns to the next step of the data accumulation process. It may happen that a card must be recorded when you are part way through a line. That will not cause a problem. After the magnetic card is written the program will automatically call for the next five bit segment. The printer program takes all of this into account and provides proper bridging between the data from two magnetic cards.

Instructions:

1. Enter the data assembler program (bank 1 only) in the start-up partitioning. Press 2nd-E' to initialize. See a one in the display.

2. The one means that the calculator is ready to accept the first five bit segment of the first tape, which will eventually control the contents of Op 01 during printing. For each bit of the grid

Fast Mode Scribbler - (cont)

where you want to print a little zero, you enter a 1, and for each bit where you want a blank space, you enter a 0. If you enter more than five bits of information, only the last five will be considered. When you have entered the five bits of information, you press A. The calculator will run for a while, and return a 2 to the display. That means it is ready to accept the second five bit segment for the line. Again, you enter the desired five digits and press A. The calculator runs for a while, and returns a 3 to the display. You enter the desired five digits and press A. The calculator runs a while and returns a 4 to the display. You enter the last five digits of the line and press B which means that you have made the last entry for the line. Actually, the program will accept an A equally as well if you have entered the set of five digits in response to the prompt "4". The calculator will run a while and return a 1 to the display, indicating it is ready for the first five digits of the next line. If a five bit segment is all blank, you do not need to type in a zero, but may simply press A or B as appropriate. Even though there is a number in the display, the program will detect that it was not entered by the user (This is the "decimal point trick" described in V5N3P7). If you have just finished entering a five digit segment in response to a prompt of "1", "2", or "3" and there are no more non-blank symbols on that line, then press B rather than A. The program will recognize that, fill out the line of zeroes, and return with a prompt of "1" for the succeeding line. If a line is all zeroes, simply respond to a prompt of "1" with B. You conserve space on the magnetic cards by using B when the remainder of a given line is blank.

2. After entering n lines you are at the end of a tape. You must have ended the last line with either an A or a B, and the prompt "1" should be in the display. Press C to end the tape. The program will return a "1" to indicate it will accept the first five bit segment in the first line of the next tape.

3. After ending an entire drawing, including ending the last line in the last tape, press E. The calculator will respond with the steady faint C at the left hand edge indicating it is ready to record the last magnetic card. The program will return a "1" to the display when the card has been written. Remember to mark the magnetic cards used for intermediate storage so you will know the order in which to insert them for the print program.

4. Turn the calculator off, and then on again and enter the three card sides for the print program. Press E and the printer will type the keyboard sequence for setting up the hexadecimal codes for fast mode and high resolution graphics as illustrated at the right. You may ignore any flashing display generated when following the procedure. When you press A as the last step in the sequence the calculator will stop with the faint C at the left hand edge of the display. The program is at the INV-Write sequence at program steps 010/011, ready to accept the first intermediate data card. Enter the first card and the printing of the "scribble" will start. When the data from one card has been printed, the calculator will stop again, waiting for the next card to be entered. If you wish, you can insert the following card as soon as one has been read, and

	PRESS	
365	14	D
366	61	GTO
367	00	00
368	24	24
369	36	PGM
370	12	12
371	71	SBR
372	04	04
373	44	44
374	91	R/S
375	37	P/R
376	31	LRN
377	46	INS
378	31	LRN
379	81	RST
380	22	INV
381	57	ENG
382	61	GTO
383	01	01
384	52	52
385	36	PGM
386	12	12
387	71	SBR
388	04	04
389	44	44
390	91	R/S
391	37	P/R
392	31	LRN
393	46	INS
394	31	LRN
395	81	RST
396	22	INV
397	57	ENG
398	11	A
399	98	ADV

Fast Mode Scribbler - (cont)

it will automatically be read when it is needed.

5. When the printout is finished cut each tape and paste them together to produce the picture. A dash is printed to show the end of each tape.

Program Listing - Data Assembler

000	76	LBL	040	00	00	080	00	00	120	01	01	160	14	D	200	17	17
001	12	B	041	25	CLR	081	49	49	121	37	37	161	73	RC*	201	01	1
002	93	.	042	29	CP	082	03	3	122	01	1	162	32	32	202	42	STO
003	71	SBR	043	42	STO	083	00	0	123	52	EE	163	32	X:T	203	31	31
004	01	01	044	38	38	084	32	X:T	124	09	9	164	25	CLR	204	94	+/-
005	80	80	045	03	3	085	43	RCL	125	32	X:T	165	67	EQ	205	42	STO
006	22	INV	046	00	0	086	37	37	126	73	RC*	166	02	02	206	32	32
007	86	STF	047	42	STO	087	65	×	127	32	32	167	22	22	207	01	1
008	00	00	048	37	37	088	22	INV	128	77	GE	168	01	1	208	44	SUM
009	42	STO	049	01	1	089	87	IFF	129	02	02	169	52	EE	209	32	32
010	39	39	050	00	0	090	00	00	130	07	07	170	09	9	210	25	CLR
011	05	5	051	22	INV	091	00	00	131	01	1	171	22	INV	211	72	ST*
012	44	SUM	052	49	PRD	092	96	96	132	00	0	172	77	GE	212	32	32
013	31	31	053	39	39	093	67	EQ	133	00	0	173	02	02	213	02	2
014	61	GTO	054	43	RCL	094	01	01	134	00	0	174	22	22	214	09	9
015	00	00	055	39	39	095	05	05	135	64	PD*	175	25	CLR	215	32	X:T
016	41	41	056	22	INV	096	01	1	136	32	32	176	14	D	216	43	RCL
017	76	LBL	057	59	INT	097	00	0	137	25	CLR	177	61	GTO	217	32	32
018	11	A	058	22	INV	098	85	+	138	43	RCL	178	01	01	218	22	INV
019	93	.	059	44	SUM	099	43	RCL	139	31	31	179	61	61	219	67	EQ
020	71	SBR	060	39	39	100	31	31	140	92	RTN	180	55	÷	220	01	01
021	01	01	061	65	×	101	95	=	141	76	LBL	181	01	1	221	37	37
022	80	80	062	01	1	102	74	SM*	142	13	C	182	52	EE	222	09	9
023	86	STF	063	00	0	103	32	32	143	25	CLR	183	05	5	223	52	EE
024	00	00	064	65	×	104	22	INV	144	01	1	184	95	=	224	09	9
025	42	STO	065	02	2	105	86	STF	145	03	3	185	22	INV	225	42	STO
026	39	39	066	45	Y^X	106	01	01	146	01	1	186	59	INT	226	29	29
027	04	4	067	43	RCL	107	25	CLR	147	76	LBL	187	65	×	227	25	CLR
028	32	X:T	068	38	38	108	01	1	148	14	D	188	01	1	228	04	4
029	43	RCL	069	95	=	109	44	SUM	149	74	SM*	189	52	EE	229	96	WRT
030	31	31	070	52	EE	110	31	31	150	32	32	190	05	5	230	01	1
031	22	INV	071	44	SUM	111	87	IFF	151	61	GTO	191	95	=	231	61	GTO
032	67	EQ	072	37	37	112	00	00	152	01	01	192	22	INV	232	02	02
033	00	00	073	01	1	113	01	01	153	18	18	193	52	EE	233	04	04
034	41	41	074	44	SUM	114	18	18	154	76	LBL	194	92	RTN	234	00	0
035	09	9	075	38	38	115	01	1	155	15	E	195	76	LBL	235	00	0
036	42	STO	076	43	RCL	116	42	STO	156	25	CLR	196	10	E'	236	00	0
037	31	31	077	39	39	117	31	31	157	01	1	197	25	CLR	237	00	0
038	22	INV	078	22	INV	118	87	IFF	158	01	1	198	07	7	238	00	0
039	86	STF	079	67	EQ	119	01	01	159	04	4	199	69	OP	239	00	0

Program Listing - Bank 1 of Printer Program Before Hex Code Generation

000	83	GO*	030	00	0	060	03	3	090	95	=	120	43	43	150	40	IND
001	69	69	031	00	0	061	00	0	091	65	×	121	01	1	151	12	12
002	76	LBL	032	25	CLR	062	00	0	092	05	5	122	44	SUM	152	37	P/R
003	11	A	033	42	STO	063	32	X:T	093	95	=	123	63	63	153	52	EE
004	25	CLR	034	63	63	064	03	3	094	42	STO	124	04	4	154	30	TAN
005	07	7	035	68	NOP	065	22	INV	095	65	65	125	42	STO	155	00	0
006	69	OP	036	68	NOP	066	28	LOG	096	73	RC*	126	62	62	156	00	0
007	17	17	037	04	4	067	95	=	097	64	64	127	61	GTO	157	00	0
008	98	ADV	038	03	3	068	59	INT	098	84	OP*	128	00	00	158	00	0
009	25	CLR	039	61	GTO	069	77	GE	099	65	65	129	43	43	159	01	1
010	22	INV	040	01	01	070	00	00	100	29	CP	130	02	2	160	01	1
011	96	WRT	041	42	42	071	77	77	101	82	HIR	131	00	0	161	06	6
012	04	4	042	43	RCL	072	42	STO	102	14	14	132	69	OP	162	42	STO
013	42	STO	043	62	62	073	69	69	103	67	EQ	133	04	04	163	69	69
014	62	62	044	22	INV	074	25	CLR	104	01	01	134	69	OP	164	16	A'
015	61	GTO	045	28	LOG	075	81	RST	105	18	18	135	05	05	165	69	OP
016	00	00	046	52	EE	076	55	÷	106	01	1	136	69	OP	166	00	00
017	33	33	047	22	INV	077	01	1	107	06	6	137	00	00	167	16	A'
018	76	LBL	048	57	ENG	078	00	0	108	01	1	138	61	GTO	168	01	1
019	16	A'	049	65	×	079	95	=	109	42	STO	139	01	01	169	01	1
020	25	CLR	050	33	X^2	080	42	STO	110	69	69	140	18	18	170	08	8
021	69	OP	051	95	=	081	64	64	111	25	CLR	141	42	STO	171	61	GTO
022	05	05	052	35	1/X	082	22	INV	112	81	RST	142	69	69	172	01	01
023	68	NOP	053	65	×	083	59	INT	113	98	ADV	143	25	CLR	173	42	42
024	74	SM*	054	73	RC*	084	65	×	114	25	CLR	144	58	FIX	174	00	0
025	80	80	055	63	63	085	02	2	115	92	RTN	145	00	00	175	00	0
026	00	0	056	95	=	086	75	-	116	16	A'	146	43	RCL	176	00	0
027	00	0	057	22	INV	087	59	INT	117	97	DSZ	147	68	68	177	00	0
028	00	0	058	59	INT	088	82	HIR	118	62	62	148	60	DEG	178	00	0
029	00	0	059	65	×	089	04	04	119	00	00	149	86	STF	179	00	0

Fast Mode Scribbler - (cont)

Program Listing - Bank 1 of Printer Program After Hex Code Generation

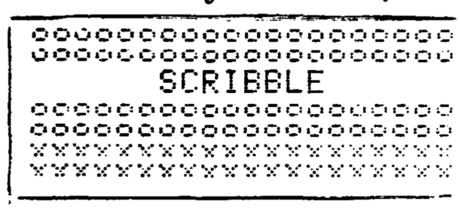
000	83	GO*	030	00	0	060	65	×	090	04	04	120	00	00	150	86	STF
001	69	69	031	29	CP	061	03	3	091	95	=	121	43	43	151	40	IND
002	76	LBL	032	83	GO*	062	00	0	092	65	×	122	01	1	152	12	12
003	11	A	033	25	25	063	00	0	093	05	5	123	44	SUM	153	68	NOP
004	25	CLR	034	42	STO	064	32	X:T	094	95	=	124	63	63	154	83	GO*
005	07	7	035	63	63	065	03	3	095	42	STO	125	04	4	155	69	69
006	69	OP	036	68	NOP	066	22	INV	096	65	65	126	42	STO	156	39	COS
007	17	17	037	68	NOP	067	28	LOG	097	73	RC*	127	62	62	157	21	2ND
008	98	ADV	038	04	4	068	95	=	098	64	64	128	61	GTO	158	42	STO
009	25	CLR	039	03	3	069	59	INT	099	84	OP*	129	00	00	159	46	46
010	22	INV	040	61	GTO	070	77	GE	100	65	65	130	43	43	160	35	1/X
011	96	WRT	041	01	01	071	00	00	101	29	CP	131	02	2	161	01	1
012	04	4	042	42	42	072	77	77	102	82	HIR	132	00	0	162	01	1
013	42	STO	043	43	RCL	073	42	STO	103	14	14	133	69	OP	163	06	6
014	62	62	044	62	62	074	69	69	104	67	EQ	134	04	04	164	42	STO
015	61	GTO	045	22	INV	075	25	CLR	105	01	01	135	69	OP	165	69	69
016	00	00	046	28	LOG	076	81	RST	106	18	18	136	05	05	166	16	A'
017	33	33	047	52	EE	077	55	÷	107	01	1	137	69	OP	167	69	OP
018	76	LBL	048	22	INV	078	01	1	108	06	6	138	00	00	168	00	00
019	16	A'	049	57	ENG	079	00	0	109	01	1	139	61	GTO	169	16	A'
020	25	CLR	050	65	×	080	95	=	110	42	STO	140	01	01	170	01	1
021	69	OP	051	33	X²	081	42	STO	111	69	69	141	18	18	171	01	1
022	05	05	052	95	=	082	64	64	112	25	CLR	142	42	STO	172	08	8
023	68	NOP	053	35	1/X	083	22	INV	113	81	RST	143	69	69	173	61	GTO
		0	054	65	×	084	59	INT	114	98	ADV	144	25	CLR	174	01	01
025	92	RTN	055	73	RC*	085	65	×	115	25	CLR	145	58	FIX	175	42	42
026	40	IND	056	63	63	086	02	2	116	92	RTN	146	00	00	176	00	0
027	49	PRD	057	95	=	087	75	-	117	16	A'	147	43	RCL	177	00	0
028	99	99	058	22	INV	088	59	INT	118	97	DSZ	148	68	68	178	00	0
029	00	0	059	59	INT	089	82	HIR	119	62	62	149	60	DEG	179	00	0

Program Listing - Banks 2 and 3 of the Printer Program

334	76	LBL	356	06	6	378	31	LRN
335	14	D	357	03	3	379	81	RST
336	25	CLR	358	06	6	380	22	INV
337	09	9	359	69	OP	381	57	ENG
338	69	OP	360	03	03	382	61	GTO
339	17	17	361	69	OP	383	01	01
340	91	R/S	362	05	05	384	52	52
341	76	LBL	363	25	CLR	385	36	PGM
342	15	E	364	90	LST	386	12	12
343	25	CLR	365	14	D	387	71	SBR
344	07	7	366	61	GTO	388	04	04
345	69	OP	367	00	00	389	44	44
346	17	17	368	24	24	390	91	R/S
347	69	OP	369	36	PGM	391	37	P/R
348	00	00	370	12	12	392	31	LRN
349	03	3	371	71	SBR	393	46	INS
350	03	3	372	04	04	394	31	LRN
351	03	3	373	44	44	395	81	RST
352	05	5	374	91	R/S	396	22	INV
353	01	1	375	37	P/R	397	57	ENG
354	07	7	376	31	LRN	398	11	A
355	03	3	377	46	INS	399	98	ADV

0.	30	1100110000.	50
11.	31	1100110011.	51
1100.	32	1100111100.	52
1111.	33	1100111111.	53
110000.	34	1111000000.	54
110011.	35	1111000011.	55
111100.	36	1111001100.	56
111111.	37	1111001111.	57
11000000.	38	1111110000.	58
11000011.	39	1111110011.	59
11001100.	40	1111111100.	60
11001111.	41	1111111111.	61
11110000.	42	0.	62
11110011.	43	0.	63
11111100.	44	0.	64
11111111.	45	0.	65
1100000000.	46	0.	66
1100000011.	47	0.	67
1100001100.	48	4000000000002.	68
1100001111.	49	0.	69

Editor's Note: The drawing on page 1 of this issue was made with the fast mode scribbler. The program also provides a capability to add titles, change the graphics symbols, etc., by writing your own routines in the program space 176 through 298 (step references are after the hex code initialization). During the data assembly process you simply enter the program step of the start of your routine and press D. You can insert as many of these jumps out of the scribbler process as you chose, limited only by the available program space. The short printer output at the right illustrates one possibility. You change the graphics symbols by changing the values in data registers 30 through 61. Your routine should use neither labels nor data registers, and should end with GTO 118. It is risky to run a subroutine in the middle of a line. Dejan asks that you send any good drawings obtained with this program to him at Gvozdiceva 34, 11000 Belgrade, Yuguslavia.

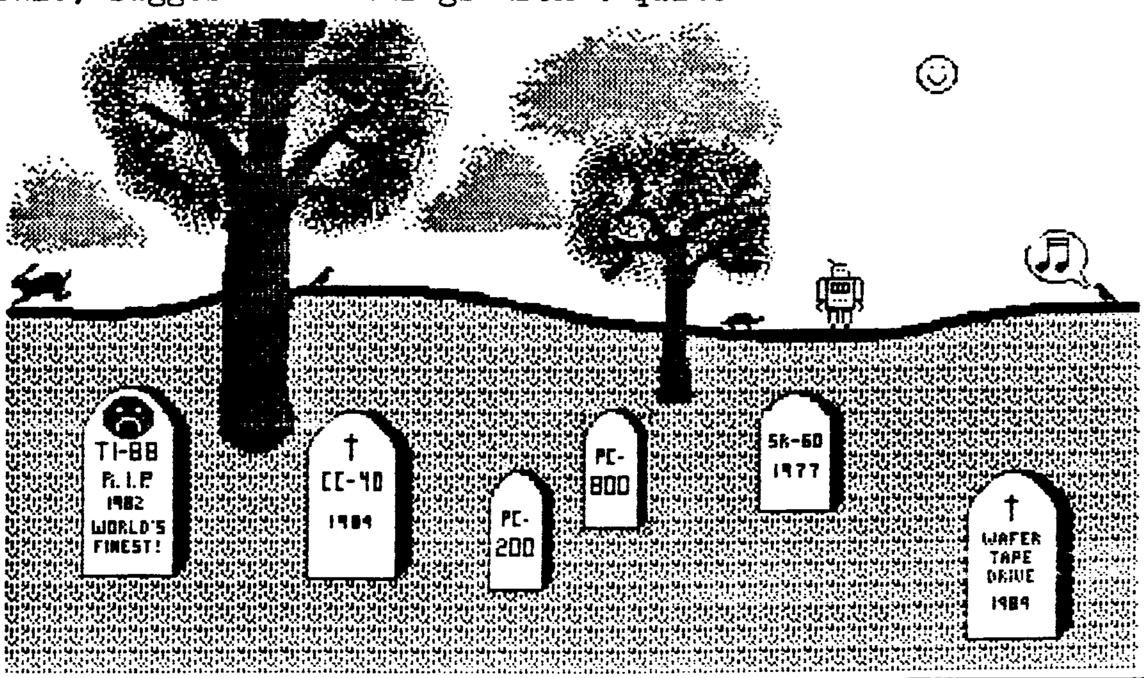


PERIPHERALS FOR THE CC-40 AND TI-66 - Palmer Hanson. In August 1983 I ordered a BA-55 and a PC-200 from Elek-Tek. The BA-55 was delivered shortly, but the PC-200 was back-ordered. In May of this year I received a letter from Elek-Tek stating "... We have recently been informed by Texas Instruments that this item will not be available. Therefore, we have taken the liberty of canceling your order ... ." I called TI's calculator information number and was told that the PC-200 was expected to be available later this year. I also wrote to TI Consumer Relations. The reply stated that the PC-200 is expected to become available in the last quarter of 1984. The latest Elek-Tek catalog (Volume 8) has dropped the PC-200. Meanwhile, Robert Prins has submitted an optimized version of Dave Leising's calendar program for the TI-66/PC-200 (V9N2P9). But neither Robert or I have a PC-200, so I have forwarded the program to Dave for review. This all sounds reminiscent of 1981/1982 when Patrick Acosta was writing successful programs for the TI-59/PC-100 when he had only a TI-58C to work with.

George Thomson had written to TI about peripherals for the CC-40. He received a reply from J. A. Hollis of Technical Communications which stated:

> "Texas Instruments will be releasing an 8K cartridge. This will provide a storage option for the CC-40.
>
> There is presently the printer/plotter and the RS-232 interface on the market for printing. The printer/plotter would allow a small compact printing capability. The RS-232 interface would hook up to a large 80 column printer for full size printouts."

Elek-Tek had also dropped the CC-40 product line from the latest catalog. A telephone call in early July confirmed that they did not have the printer/plotter available. The latest Educalc catalog (issue 21) did have the CC-40 and peripherals listed. A telephone call in early July confirmed that they had the HX-1000 Plotter/Printer and the HX-3000 RS-232 Interface without the Parallel Interface Option in stock. I have ordered a printer/plotter. I will hope to have some sample printouts and plots for the next issue. The schedule slips, the on-again, off-again discussions of the wafer-tape drive in the public press, etc., all combined to induce one member to provide the following "graveyard" illustration. TI's assurances that the PC-200 is forthcoming, coupled with the availability of CC-40 peripherals from Educalc, suggest that things aren't quite that bad!



SIMULTANEOUS EQUATIONS WITH THE CC-40 MATHEMATICS MODULE - P. Hanson

V8N5P14-16 discussed the matrix operations programs in the CC-40 mathematics module. V8N6P19 reported excellent results using those techniques to solve the 7x7 sub-Hilbert, but presented the results only to enough digits to establish the relative error. James Walters proposed another method of error evaluation, that is, multiplying the solution vector by the original matrix and comparing the result with the input unity vector. To use that method it was important to use all of the digits of the solution. No difficulties were found in doing that with any of the TI-59 solutions, or for any of the solutions on personal computers using the program on V8N6P20; but the Mathematics module solution from the CC-40 would only yield the 9 to 11 digits shown at the right. After a lot of agonizing over items such as whether my application of "PRINT USING" was proper, and the like, I finally found that the truncated result from the CC-40 Mathematics module is a direct result of the method of solution.

56.00003229
-1512.000787
12600.00591
-46200.0192
83160.0311
-72072.0246
24024.007436

I had assumed that the method of solution from the CC-40 Mathematics module and from the TI-59 Master Library (ML-02) was the same. The discussions under "Method Used" on page 13 of the Master Library manual and on pages 53-54 of the CC-40 Mathematics module manual are identical. Experiments show that the methods for solution of linear equations must be quite different. The ML-02 solution on the TI-59 does not seem to make direct use of the inverse of the matrix. The CC-40 solution seems to obtain the inverse, and simply multiply the inverse by the vector to get the solution. Where the vector is the unity vector as in our 7x7 sub-Hilbert test problem, the solution may be obtained by simply adding up the rows of the inverse matrix. For the 7x7 problem, the seventh (or bottom row) of the inverse matrix is listed at the right. Now, if you sum the terms in the row from the top, as would be reasonable for a loop in the computer, then you will obtain exactly the solution for the seventh element in the table at the top of the page. Similar results can be obtained for the other elements of the solution by reading out the elements of the inverse matrix. You must remember to always truncate each intermediate sum to the fourteen digit limit of the computer. The truncated output arises because the last summation is between two numbers of about 41 million but of opposite sign, yielding an answer of about 24 thousand. The same sort of result can be obtained with the ML-02 programs by not solving simultaneous equations with ML-02 Program E, but rather obtaining the inverse matrix with ML-02 Program B', and summing the rows. The printout on the left below is the ML-02 solution using the standard method. The printout at the right was obtained using the inverse matrix method. Again, the truncation effect is evident. Until TI choses to release the program details of the CC-40 Solid State Modules we can only continue to try to understand through experimentation.

C(7,1)	=	168,168.045 045 95
C(7,2)	=	-4,036,033.121 075
C(7,3)	=	30,270,248.620 461
C(7,4)	=	-100,900,829.259 5
C(7,5)	=	166,486,368.937 2
C(7,6)	=	-133,189,095.560 0
C(7,7)	=	41,225,196.345 304

56.0081897448	56.0081896
-1512.189567429	-1512.189528
12601.38627848	12601.38624
-46204.53435755	-46204.5337
83167.37180486	83167.3708
-72077.82742612	-72077.8273
24025.7860121	24025.78576

<u>MINEFIELD</u> <u>III</u> - Lem Matteson. This fast mode version of Dejan Ristanovic's Minefield II program from V8N5P7 reduces the time to set up the game from ten minutes to six and one-half minutes. The original program included several levels of subroutines; but subroutines are not allowed in fast mode. Pseudo-subroutines were mechanized by using absolute address transfers to enter the routine, and ending the routine with an indirect GTO (code 83) instruction where the return address is loaded into the indirect register before the pseudo-subroutine is called.

An example pseudo-subroutine is used as a random number generator to replace the Pgm-15-SBR-DMS which is in the original program. That random number generator required 55 program steps--more than were available in the conversion to fast mode. A much shorter pseudo-random number generator was used instead (program steps 302 to 323). One call of the pseudo-subroutine appears at program steps 213 to 215. The return address for the program step following the call (step 216) was loaded in the return register (data memory 21) by steps 208 through 212. There is a second call to the same pseudo-subroutine in the steps immediately following. This time a new return address (226) is loaded by steps 218 through 222.

The instructions are the same as those for Minefield II except:

1. Enter the three banks of the program, enter a seed, and press E'. Do not clear the flashing 1, but press 7 and then EE. After about 6½ minutes the setup is complete.
2. Use A' rather than SBR SBR to set the shield.
3. C' is available to replay the same setup.

A sample printout for a seed of pi appears at the right.

Newcomers who do not have V8N5 can obtain a copy of the complete instructions by sending a SASE to the editor.

```
POS.   00    0 MINES
POS.   01    0 MINES
POS.   02    0 MINES
POS.   03    0 MINES
POS.   04       ROCK
POS.   13    0 MINES
POS.   23    1 MINES
POS.   33    1 MINES
POS.   43    1 MINES
POS.   53    0 MINES
POS.   54    0 MINES
POS.   55    1 MINES
POS.   65    1 MINES
POS.   75    1 MINES
POS.   76    0 MINES
POS.   77    0 MINES
POS.   87    1 MINES
POS.   88    2 MINES
POS.   89      SHLD.
        20.     TRYS
```

Program Listing - steps 000 through 159

```
000  61  GTO
001  03   03
002  65   65
003  76  LBL
004  11   A
005  69  OP
006  34   34
007  61  GTO
008  03   03
009  78   78
010  76  LBL
011  12   B
012  69  OP
013  24   24
014  61  GTO
015  03   03
016  78   78
017  76  LBL
018  13   C
019  69  OP
020  33   33
021  61  GTO
022  03   03
023  78   78
024  76  LBL
025  14   D
026  69  OP
027  23   23
028  61  GTO
029  03   03
030  78   78
031  76  LBL
032  15   E
033  86  STF
034  01   01
035  92  RTN
036  76  LBL
037  10  E'
038  82  HIR
039  05   05
040  04   4
041  69  OP
042  17   17
043  61  GTO
044  06   06
045  19   19
046  76  LBL
047  16  A'
048  01   1
049  82  HIR
050  34   34
051  22  INV
052  86  STF
053  01   01
054  92  RTN
055  76  LBL
056  18  C'
057  98  ADV
058  61  GTO
059  03   03
060  65   65
061  47  CMS
062  82  HIR
063  15   15
064  42  STO
065  09   09
066  01   1
067  00   0
068  42  STO
069  06   06
070  02   2
071  04   4
072  42  STO
073  08   08
074  09   9
075  42  STO
076  00   00
077  08   8
078  05   5
079  42  STO
080  20   20
081  08   8
082  61  GTO
083  02   02
084  06   06
085  97  DSZ
086  00   00
087  00   00
088  77   77
089  09   9
090  07   7
091  42  STO
092  20   20
093  09   9
094  61  GTO
095  02   02
096  06   06
097  69  OP
098  33   33
099  02   2
100  44  SUM
101  04   04
102  03   3
103  42  STO
104  05   05
105  03   3
106  94  +/-
107  44  SUM
108  04   04
109  03   3
110  42  STO
111  00   00
112  22  INV
113  86  STF
114  00   00
115  01   1
116  02   2
117  05   5
118  42  STO
119  20   20
120  43  RCL
121  03   03
122  61  GTO
123  02   02
124  86   86
125  01   1
126  03   3
127  05   5
128  42  STO
129  20   20
130  43  RCL
131  04   04
132  61  GTO
133  02   02
134  86   86
135  87  IFF
136  00   00
137  01   01
138  88   88
139  08   8
140  32  X:T
141  01   1
142  04   4
143  09   9
144  42  STO
145  21   21
146  61  GTO
147  03   03
148  24   24
149  77  GE
150  01   01
151  88   88
152  85   +
153  01   1
154  95   =
155  69  OP
156  24   24
157  85   +
158  82  HIR
159  12   12
```

Minefield III - (cont)

Program Listing - steps 160 through 639

160	95	=	240	06	6	320	95	=	400	02	02	480	05	5	560	06	6
161	55	÷	241	42	STO	321	59	INT	401	86	86	481	07	7	561	69	OP
162	01	1	242	21	21	322	83	GO*	402	43	RCL	482	32	X:T	562	04	04
163	00	0	243	61	GTO	323	21	21	403	04	04	483	71	SBR	563	43	RCL
164	44	SUM	244	03	03	324	01	1	404	71	SBR	484	05	05	564	08	08
165	03	03	245	24	24	325	00	0	405	02	02	485	89	89	565	69	OP
166	85	+	246	77	GE	326	44	SUM	406	86	86	486	65	×	566	06	06
167	82	HIR	247	02	02	327	03	03	407	87	IFF	487	01	1	567	91	R/S
168	13	13	248	08	08	328	69	OP	408	00	00	488	00	0	568	03	3
169	95	=	249	82	HIR	329	24	24	409	05	05	489	00	0	569	06	6
170	65	×	250	18	18	330	43	RCL	410	35	35	490	95	=	570	02	2
171	43	RCL	251	69	OP	331	04	04	411	03	3	491	69	OP	571	03	3
172	04	04	252	24	24	332	94	+/-	412	03	3	492	03	03	572	02	2
173	22	INV	253	85	+	333	22	INV	413	03	3	493	03	3	573	07	7
174	28	LOG	254	82	HIR	334	28	LOG	414	02	2	494	00	0	574	01	1
175	52	EE	255	12	12	335	52	EE	415	03	3	495	02	2	575	06	6
176	95	=	256	95	=	336	65	×	416	06	6	496	04	4	576	04	4
177	72	ST*	257	55	÷	337	73	RC*	417	04	4	497	03	3	577	00	0
178	03	03	258	01	1	338	03	03	418	00	0	498	01	1	578	69	OP
179	01	1	259	00	0	339	95	=	419	69	OP	499	01	1	579	04	04
180	00	0	260	44	SUM	340	82	HIR	420	01	01	500	07	7	580	69	OP
181	22	INV	261	03	03	341	03	03	421	43	RCL	501	03	3	581	05	05
182	44	SUM	262	85	+	342	22	INV	422	03	03	502	06	6	582	01	1
183	03	03	263	82	HIR	343	59	INT	423	71	SBR	503	69	OP	583	82	HIR
184	22	INV	264	13	13	344	82	HIR	424	05	05	504	04	04	584	34	34
185	52	EE	265	95	=	345	53	53	425	89	89	505	82	HIR	585	22	INV
186	69	OP	266	65	×	346	65	×	426	82	HIR	506	12	12	586	86	STF
187	34	34	267	43	RCL	347	01	1	427	06	06	507	29	CP	587	01	01
188	69	OP	268	04	04	348	00	0	428	43	RCL	508	69	OP	588	92	RTN
189	24	24	269	22	INV	349	22	INV	429	04	04	509	05	05	589	85	+
190	97	DSZ	270	28	LOG	350	44	SUM	430	71	SBR	510	61	GTO	590	32	X:T
191	00	00	271	52	EE	351	03	03	431	05	05	511	00	00	591	08	8
192	01	01	272	95	=	352	95	=	432	89	89	512	51	51	592	77	GE
193	12	12	273	72	ST*	353	82	HIR	433	85	+	513	43	RCL	593	05	05
194	69	OP	274	03	03	354	02	02	434	82	HIR	514	01	01	594	97	97
195	23	23	275	01	1	355	59	INT	435	16	16	515	42	STO	595	03	3
196	97	DSZ	276	00	0	356	82	HIR	436	65	×	516	03	03	596	76	LBL
197	05	05	277	22	INV	357	52	52	437	01	1	517	43	RCL	597	01	1
198	01	01	278	44	SUM	358	22	INV	438	00	0	518	02	02	598	95	=
199	05	05	279	03	03	359	52	EE	439	00	0	519	42	STO	599	92	RTN
200	97	DSZ	280	22	INV	360	69	OP	440	95	=	520	04	04	600	00	0
201	06	06	281	52	EE	361	34	34	441	69	OP	521	03	3	601	00	0
202	00	00	282	69	OP	362	92	RTN	442	02	02	522	05	5	602	00	0
203	89	89	283	34	34	363	83	GO*	443	01	1	523	03	3	603	00	0
204	20	CLR	284	83	GO*	364	21	21	444	32	X:T	524	02	2	604	00	0
205	81	RST	285	20	20	365	25	CLR	445	82	HIR	525	01	1	605	00	0
206	82	HIR	286	65	×	366	42	STO	446	14	14	526	05	5	606	00	0
207	08	08	287	53	(	367	03	03	447	67	EQ	527	02	2	607	00	0
208	02	2	288	40	IND	368	42	STO	448	05	05	528	06	6	608	00	0
209	01	1	289	75	-	369	04	04	449	68	68	529	69	OP	609	00	0
210	06	6	290	09	9	370	42	STO	450	71	SBR	530	04	04	610	00	0
211	42	STO	291	95	=	371	08	08	451	03	03	531	69	OP	611	00	0
212	21	21	292	32	X:T	372	82	HIR	452	24	24	532	05	05	612	00	0
213	61	GTO	293	00	0	373	04	04	453	82	HIR	533	09	9	613	00	0
214	03	03	294	77	GE	374	03	3	454	02	02	534	92	RTN	614	00	0
215	02	02	295	02	02	375	05	5	455	32	X:T	535	01	1	615	00	0
216	42	STO	296	99	99	376	42	STO	456	09	9	536	04	4	616	00	0
217	03	03	297	86	STF	377	07	07	457	67	EQ	537	03	3	617	00	0
218	02	2	298	00	00	378	69	OP	458	05	05	538	02	2	618	00	0
219	02	2	299	92	RTN	379	28	28	459	35	35	539	03	3	619	00	0
220	06	6	300	83	GO*	380	69	OP	460	08	8	540	02	2	620	93	.
221	42	STO	301	20	20	381	00	00	461	67	EQ	541	03	3	621	01	1
222	21	21	302	43	RCL	382	01	1	462	05	05	542	00	0	622	82	HIR
223	61	GTO	303	09	09	383	08	8	463	13	13	543	07	7	623	35	35
224	03	03	304	23	LNX	384	32	X:T	464	43	RCL	544	03	3	624	07	7
225	02	02	305	50	I×I	385	43	RCL	465	03	03	545	69	OP	625	04	4
226	42	STO	306	22	INV	386	03	03	466	42	STO	546	04	04	626	02	2
227	04	04	307	59	INT	387	85	+	467	01	01	547	69	OP	627	52	EE
228	85	+	308	42	STO	388	43	RCL	468	43	RCL	548	05	05	628	01	1
229	43	RCL	309	09	09	389	04	04	469	04	04	549	25	CLR	629	02	2
230	03	03	310	65	×	390	95	=	470	42	STO	550	68	NOP	630	94	+/-
231	95	=	311	04	4	391	67	EQ	471	02	02	551	61	GTO	631	85	+
232	29	CP	312	22	INV	392	05	05	472	25	CLR	552	61	GTO	632	01	1
233	67	EQ	313	23	LNX	393	53	53	473	87	IFF	553	03	3	633	95	=
234	02	02	314	95	=	394	22	INV	474	01	01	554	07	7	634	22	INV
235	08	08	315	22	INV	395	86	STF	475	05	05	555	03	3	635	52	EE
236	08	8	316	59	INT	396	00	00	476	07	07	556	05	5	636	58	FIX
237	32	X:T	317	65	×	397	43	RCL	477	22	INV	557	04	4	637	00	00
238	02	2	318	01	1	398	03	03	478	97	DSZ	558	05	5	638	60	DEG
239	04	4	319	00	0	399	71	SBR	479	07	07	559	03	3	639	86	STF

DISTANCE TO THE NEAREST INTEGER - Charlie Williamson proposed this brainteaser in V9N1P17: Find the distance from x to its nearest integer. Two classes of solution are desired; those which use the t register, and those which do not. All the solutions here are without the t register.

Stanley Becker submitted a 26 step solution using Op 10:

LBL A |x| - (CE - INT STO 01 - . 5)Op 10 STO 02 1 SM* 02
RCL 01 = |x| R/S

Myer Boland submitted a 19 step solution:

LBL A |x| STO 01 + . 5) - INV INT) - RCL 01) |x| R/S

Dave Leising submitted a 16 step solution:

LBL A |x| INV INT - . 5 = |x| +/- + . 5 = R/S

Don Graham and Stanley Becker both submitted 14 step sooutions:

LBL A |x| - (CE + . 5) INT = |x| R/S

and, Don Graham submitted a 13 step solution:

LBL A - FIX 0 EE INV EE = INV FIX |x| R/S

I was happy with these solutions until Larry Leeds wrote that Charlie had found that many solutions, including his own, would give incorrect answers when the input was 1 DIV 9 x 9 = .999 999 999 999 9 . Dave Leising's routine gives the correct answer of 1e-13. All the other routines above give an incorrect answer of 1e-12 .

DISPLAY 24 - This brainteaser from V8N6P3 stated: Using a single digit integer three and only three times, use the functions of the calculator to arrive at 24."

Myer Boland submitted (3 + 3 / 3) !

Myer, Larry Leeds, and Gregory Hoen submitted $3\ y^x - 3 =$

and Larry Leeds submitted $3\ x\ x^2 - 3 =$.

W. J. Widmer recognized that the problem could be solved for any integer used three times. He wrote: Although the problem may seem trivial, it is a fair exercise on the use of calculator functions and pending-operation status. The latter requires either a CLR or =, except on the TI-57LCD which uses a double ON/C or OFF ON/C to clear; but a single ON/C can be used with confidence if no accumulation functions + or - follow. I've worked up several routines for digits 0 through 9 and also for "no digit" (not inputted thrice?) on different TI PPC's. e^x and 10^x , for example are not "digits" and so may be used; but in any event, I don't guarantee that my routines are the shortest possible for the indicated machines. 45 routines appear on page 27. In the starred routines 1 through 5, 14, and 21 one or more digit inputs are "dummy" and are used simply to satisfy the "thrice input" requirement; for these, alternate bonafide sequences are also given. The least step routine I've found for each machine is underlined. I note that the TI-59, 58C and 57 have no x! key (I purposely did not use the Master Library call for factorial on the TI-58/59, considering that a cop-out); the TI-55 has no INT, and the TI-57LCD has no CE (although STO on that machine can serve as a CE carry-over function in some operations). Finally, I assume that STO N and RCL N satisfy the requirement for digit N input.

Routines for DISPLAY 24 (W.J. Widmer)

Routine	Digit N	Routine Key-in	# of Steps	TI59	TI 58C	TI57 LCD	TI57	TI55
1*	0	CLR 000 COS + CE) x^2 x!	10					24
2*	0	= 000 COS + STO) x^2 x!	10			24		
3*	0	CLR 00 ÷ 0 + CE) x^2 x!	10					24
4*	0	= 00 COS + 0 COS = x^2 x!	10			24		24
5*	0	= 000 INV LOG LN INT x^2 x!	10			24		
6	0	0 COS 10^x 10^x + STO 0) FIX 0 LN	11					24
7	0	0 COS INV LOG INV LOG + STO 0) FIX 0 LN	13	24	24	24	24	
8	0	CLR COS + 0 COS + 0 COS + 0 COS) x!	13					24
9	0	ON ON COS + 0 COS + 0 COS + 0 COS) x!	14			24		
10	1	CLR 1 1 + 1 + CE =	8	24	24		24	24
11	1	ON 1 1 + 1 + STO =	8			24		
12	1	1 1 1 LOG INT x^2 x!	7			24		
13	1	1 1 1 INV LN √x LOG INT	8	24	24	24	24	
14*	2	2 ÷ 2 X 2 x^2 x!	7			24		24
15	2	2 2 2 LOG INT x^2 x!	7			24		
16	2	2 2 2 INV LN √x √x LOG INT	9	24	24	24	24	
17	2	= 2 x^2 x^2 + 2 X 2 x^2 =	10	24	24	24	24	24
18	3	= 3 y^x 3 - 3 =	7	24	24	24	24	24
19	3	= 3 3 - 3 x^2 =	7	24	24	24	24	24
20	3	3 3 3 LOG INT x^2 x!	7			24		
21*	4	4 ÷ 4 X 4 x!	6			24		24
22	4	= 4 x^2 + 4 + 4 =	8	24	24	24	24	24
23	5	5 5 5 LOG INT x^2 x!	7			24		
24	5	= 5 x^2 - 5 ÷ 5 =	8	24	24	24	24	24
25	6	6 6 6 LOG INT x^2 x!	7			24		
26	6	= 6 x^2 - 6 - 6 =	8	24	24	24	24	24
27	7	7 7 7 LOG INT x^2 x!	7			24		
28	7	7 7 . 7 TAN INT x!	7			24		
29	7	7 7 7 INV TAN INT x!	7			24		
30	7	= 7 + 7) ÷ 7 = x^2 x!	10			24		24
31	7	CMS Σ + 7 + STO 7 + RCL 7 =	11	24	24			24
32	7	CMS Σ Σ Σ + 7 + 7 + 7 =	11	24	24			24
33	7	CLR COS + 7 + STO 7 + RCL 7 =	11	24	24		24	24
34	8	= 8 + 8 + 8 =	7	24	24	24	24	24
35	9	9 9 9 LOG INT x^2 x!	7			24		
36	9	= 9 √x X 9 - 9 √x =	9	24	24	24	24	24
37	9	= 9 √x x! + 9 + 9 =	9			24		24
38	-	CA Σ Σ x^2 x!	5					24
39	-	CA Σ Σ Σ Σ x!	6					24
40	-	CLR COS + CE) x^2 x!	7					24
41	-	ON COS INV LN INT x^2 x!	7			24		
42	-	ON ON COS + STO) x^2 x!	8			24		
43	-	CLR COS + (CE + 10^x) + CE =	11					24
44	-	CLR COS + (CE + INV LOG) + CE =	12	24	24		24	
45	-	CLR COS + CE INV LOG + LOG INT + CE =	12	24	24		24	

FROM THE EDITOR:

Again, I must apologize for a late issue. And again, I have an excuse. In late May my recurrent back problems were diagnosed as a ruptured disc, and I had surgery in early June. For a minimum of two months I am under severe restrictions on activity. That means that at least the next issue will be late as well.

The first page was devoted to another illustration using high resolution graphics techniques by Dejan Ristanovic. V8N2 had published some high resolution, 3D graphics techniques, but had elicited very little feed-back; but there is renewed interest on the part of newcomers such as Hewlett Ladd and Lelio Mortola, and there are some very nice high resolution graphs in publications such as Programbiten and from Peter Poloczek's TI-58/59 Club. What seems to be missing is a good source of help for newcomers. In V6N6/7P12 Maurice Swinnen gave a cursory review of some features of high resolution graphics and indicated that he hoped to publish a summary in the next issue. That never materialized and newcomers are left to largely "re-invent the wheel". I too hope to publish tips for the use of high resolution graphics in coming issues. I have been able to demonstrate the effect at many more, but not all, of the 0 mod 8 possible locations than the 008, 016, 024 and 184 mentioned by Dave Leising. So expect more in this area. As my friend Bob Upton has aptly observed, "Palmer is the kind of person, that when asked for the time of day, tells you how to construct a clock."

Robert Prins has continued to investigate interaction of the flag register and the generation of hexadecimal codes. He has found that under some circumstances h54 can be used to recall the contents of the flag register to the display, for example. He has compiled an eighteen page, closely spaced discussion of his results. While I will publish excerpts in coming issues, there is just too much material to be covered completely. Robert will air mail a copy to readers for $6.00 US funds. Write to Robert A. H. Prins, Alfred Nobellaan 112, 3731 DX De Bilt, Netherlands. Send a postal money order or cash. This is a must for serious investigators of hexadecimal codes.

We have managed to get our club listed in a number of places such as BYTE, EDN, Computer Shopper, and the like. One of the advantages (?) of such listings is that the club mailbox is continually filled with bulk mailings for a wide range of products--computers, software, discs, computer supplies, etc. I have little use for most of it. But some time ago one of our members had written for information on a subject, and I just happened to get some incoming mail on the same subject, and was able to forward it to him. If you are looking for something in that line ranging from discount prices, catalogs, etc., send a SASE together with your desires and I will forward anything that seems to be pertinent.

TABLE OF CONTENTS

Magnetic card service and card reader cleaning cards continue to be available at the same rates formerly quoted.