TI PPC NOTES

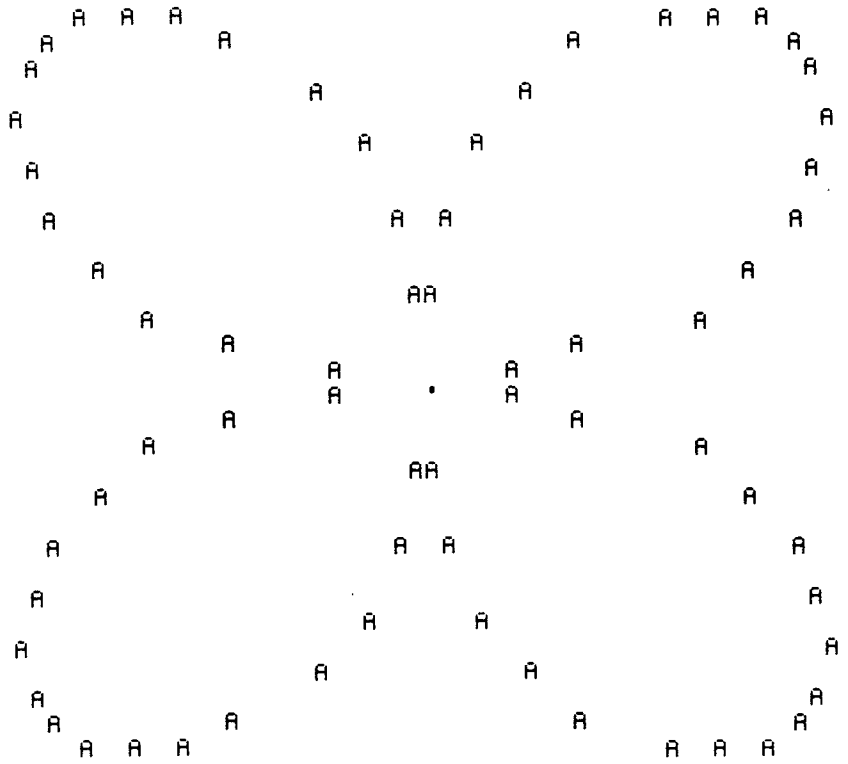
NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB

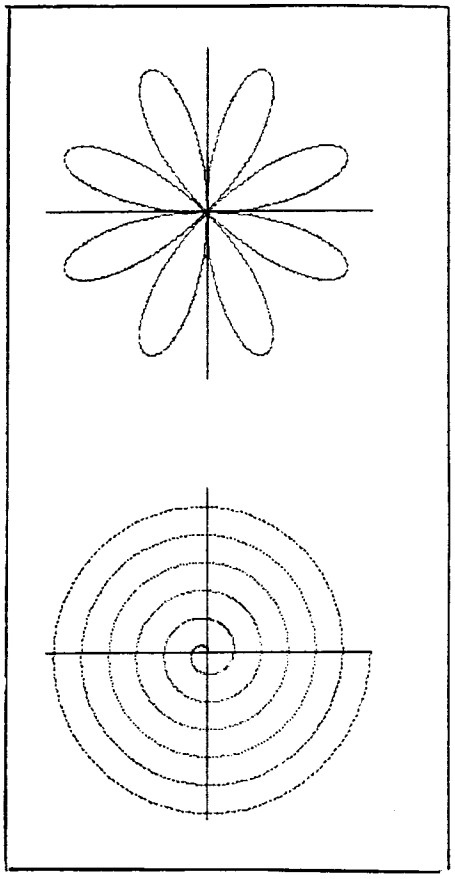
P.O. Box 1421, Largo, FL 34294

Volume 10, Number 2 Second Quarter 1985

Plotting functions from polar coordinates - see pages 12 and 14.

TI59/PC100 CC40/HX1000





2. 2. 2.

ERRATA:

Product Formula for the Geometric Mean - The equation for the geometric mean on page 4-2 of the Applied Statistics manual is incorrect. The opening parenthesis must be moved in front of the product notation. The equation then becomes

$$g = \left(\prod_{i=1}^{n} x_i^{f_i} \right)^{1/N}$$

Sum of Log(x) Formula for the Geometric Mean - The equation for the geometric mean on page 23 of the Statistics cartridge manual for the CC-40 is incorrect. The correct formula is

$$\bar{x}_g = 10^{\left(\sum_{i=1}^{n} f_i \cdot \log(x_i) \right)/N}$$

MAGNETIC CARD AVAILABILITY - Member J. M. Gallego still has magnetic cards and a limited number of solid state software modules available. His inventory in late May was:

93 Boxes of 40 Blank Magnetic Cards with Carrying Case
3 Real Estate and Investment modules

He will sell the boxes of magnetic cards for eight dollars ($8.00) each, and the modules for sixteen dollars ($16.00) each. Shipping is included in those prices. U.S. members should send money orders only to:

Q. Jose M. Gallego
250 Quintard Avenue, Apt. 96
Chula Vista CA 92011/4924

Members from other countries should write to make appropriate arrangements.

A NEW SENIOR MEMBER NEEDED - For several years Ralph Snyder has been the senior member of our club (see V7N4/5P14, for example). Ralph did not renew his membership for 1985, citing his age (now 85) and limited interests. So we need to identify a new senior member. I have already identified a 73 year old member. Anyone older is invited to apply for the position.

IN MEMORIAM - I am sorry to report the deaths of Professor William W. Buechner of Arlington, Maryland and Robert M. Elliott of Norwich, Connecticut. Prof. Buechner contributed several programs to PPX Exchange. Our club knew him through his article on the use of trig functions to simplify relativistic calculations (V7N1/2P10). Robert Elliott wrote several programs for navigation solutions such as the Relative Motion of Ships and Aircraft (V6N3P8). The club has lost two valuable members.

PRODUCT OVERFLOW WITH THE TI-59 STATISTICS MODULE - Palmer Hanson

I was trying to use the Applied Statistics module programs to analyze grouped univariate data. My first entry was an x value of about 50 with a frequency of 96. The calculator returned a flashing one to the display. After some experimenting I found that if I entered the 96 values in three groups of 32 each I would avoid the flashing display, and obtain a result for the entire problem which was in agreement with other calculations of the mean and second moment. How could this be? As I scanned the intermediate data base I found that the content of R11 was 9.9999999 99 flashing, an indication of overflow. Table 3-1a of the Applied Statistics manual showed that the contents of R11 should be πx^f, the product of all the x's, where the notation I have usually seen uses an upper case pi. I recognized that the product is used in the calculation of the geometric mean, and an attempt to print out the geometric mean with Pgm-08-B yielded an unreasonable result with a flashing display. But how did splitting the entry into three groups avoid the flashing display at data entry? Examination of the listing for ST-03 showed that the initialization Pgm-03-2nd-E' stored a one in R11. As each group was entered the value x^f was calculated (steps 166-172) and a PRD 11 command (steps 202-203) was used to update the accumulated product in R11. While examining this problem I noted that the equation for the geometric mean on page 4-2 of the Applied Statistics manual was incorrect. See the correction in the errata on page 2 of this issue.

A similar overflow problem does not occur when using the Means and Moments routines of the Statistics cartridge for the CC-40. That routine accumulates the sums of the logs of the x's, not the product of the x's, so that overflow would be approached much more slowly. Curiously enough, the equation for the geometric mean using this method which appears on page 23 of the manual is also incorrect. See the correction in the errata on page 2 of this issue.

MORE PPX PROGRAM AVAILABILITY - Thomas Wysmuller writes that all programs that he submitted to the old PPX Exchange are available free to TI PPC Notes subscribers. The programs listed in the September 1980 issue of the PPX catalog are:

018003	Monthly/Fiscal Year Linear Trend Projection
018004	Organizational Tech. Level Index - Average Grade
088014	Maximum Allowable Monthly Rent
208008	Polynomial Regression
398110	Pythagorean Analysis with Integer Solutions
908115	Utility Routine: Subroutine and Loop Timer
908142	Bucket Sort (up to 99 items)

where 208008 and 908142 were previously listed as available from other members. Write to Thomas Wysmuller, Phoenix Mutual, 1 American Row, Hartford CT 06115 or call (203) 275 5496.

SHIPPING AND HANDLING CHARGES AT NTIS - Earlier issues of TI PPC Notes have carried abstracts of calculator related documents available from NTIS. Prices were included using the schedules published with Government Reports Announcements. The latest issue of GRA carried the following insert: "Effective on orders received June 1, 1985 the following NTIS shipping and handling charges apply: U.S., Canada, Mexico - add $3 per total order; all other countries - add $4 per total order." Orders requesting NTIS rush handling are excepted.

OTHER PUBLICATIONS FOR NEWCOMERS - offered on a "while they last" basis.

Programming Workbooks - These books were used in seminars to allow students to do exercises in class. They can also be used for self study, but the user is cautioned that Exercise 10-1 shows the correct sequence before editing. Send two dollars (U.S. only) .

Old Single Issues of TI PPC Notes - These are residual inventory which are not part of one year sets:

V6N9/10 (1981) - 32 pages - two dollars

V7N4/5 (1982) - 30 pages - two dollars

V7N6 (1982) - 18 pages - one dollar

Compilations - I assembled these documents in 1982/1983 for the benefit of new members who did not want to purchase the complete 1980 through 1982 sets in order to obtain the background on various subjects:

Hexadecimal Codes - twelve pages which trace the parallel early discoveries by Michael Sperber and Patrick Acosta. Examples include the use of h12 for fast mode entry, and the use of h25 for high resolution graphics. Also discusses other hexadecimal codes and their effects. Three dollars.

TI-58/58C/59 Firmware - twenty pages extracted from TI PPC Notes, 52 Notes, and other sources. Explains how to download the firmware, discusses the equations mechanized, and includes cautions on the use of the firmware. Four dollars.

MAILBAG - Due to the favorable response to V10N1P25 I have included additional comments received with the 1985 subscription forms. The comments continue to show a wide range of opinion as to what should or should not be included in TI PPC Notes:

"If membership is falling, perhaps it is time to poll subscribers and see where we go. You fill a real need on coverage for pocketable devices." R.S.

"Let's have 'Letters to the Editor', author contact (way to reach them), continuous pagination throughout a volume. Should we include Radio Shack? Yes." C.W.

"It would be a big help if every time a program is published a sample problem is also given, as in V9N6P23-24." J.V.

"Keep up the good work! I've been a 'calculator freak' for the last 12 years and I enjoy every issue, even if I don't try out every TI-58/59 program." W.M.

"Thank you for the second reminder - I have been busy trying to learn the operation procedures of an IBM XT Computer." R.S.

"... my interest is in the TI-59 only. More emphasis for this calculator please. In the alternative a newsletter for the 59 only." C.D.

"This year I would like to see additional information for electronics folks, I'm in radio. How about information on converting TI-59 programs for use in IBM PC's or TI home computer." F.S.

"I am interestd in civil engineering programs." L.P.

MORE ON CALCULATORS IN NAVIGATION - P. Hanson. In V9N6P13 I listed ten articles on the use of calculators in navigation which had appeared in NAVIGATION, the Journal of the Institute of Navigation (U.S.A.) In subsequent review of back issues I have found four additional articles on the use of calculators and personal computers in navigation. The same availability applies. Send one dollar for each article to cover the cost of copying and handling.

1. "Computer Sight Reduction Based on Intersection of Equal Altitude Circles", by R. W. Flynn, Vol. 19, No. 1, Spring 1972, pp 7-10.

2. "Navigation Applications of the HP-65 Calculator", by K. E. Newcomer, Vol. 22, No. 2, Summer 1975, pp 152-156. Essentially the same paper was previously presented at the ION National Marine Meeting,12-13 March 1974.

3. "An Analytical Solution of the Two Star Sight Problem of Celestial Navigation", by J. A. Van Allen, Vol. 29, No. 1, Spring 1981, pp 40-43.

4. "Applications of Personal Computers to Marine Navigation and Surveying", by J. R. Stolz and Jack O. Hill, Vol. 31, No. 4, Winter 1984-85, pp 275-288.

FASTER LRN MODE OPERATION ON THE TI-66 - R. Prins - V9N1P5/6 discussed the slow execution speed of the TI-66 and noted that it was very easy to enter program code faster than the calculator could accept it in LRN mode. One remedy is to always be entering code near the end of the current partitioning. As an example, press 0-Op-17 and see the display "511", indicating that there are 512 program steps available, but no data registers available. Press RST, go to LRN, see "St" in the display indicating that entered code will be inserted beginning at location 000, and start entering code at a high rate. You will find that most of the code is not accepted by the TI-66. Now press LRN to return to keyboard control, press GTO-480-LRN and see "480 XXX" in the display, where XXX is the mnemonic for whatever program code is resident at that location, or the equivalent code to whatever data was stored in the equivalent memory register. You will find that it is practically impossible to enter code faster than the TI-66 can accept it.

This dependence of the speed of entry of program steps on the proximity to the end of the current partitioning seems to associated with "insert" method of program entry used with the TI-66. Each newly entered step is inserted immediately after the code which appears in the display; thus, as each new step is entered, each of the subsequent program steps must be pushed down one location in the stack.

A HARDWARE MODIFICATION TO INCREASE EXECUTION SPEED OF THE TI-66 - R. Prins

Editor's Note: I have not yet performed the following modification on my TI-66. Readers are cautioned that hardware modifications are at their own risk, and are likely to void any warranty.

A better way to increase the speed--the method in the previous article is only useful in keying in programs, not for execution time--is to replace the small 180K resistor next to the CPU chip with a smaller one. If you ask me how I know, well I took a calculated gamble, as an article in the German Funkschau (1983 V6 P77) described the same modification, also replacing a resistor, in the Casio FX-601P/602P. That resistor, you won't believe it, was also 180K and was also closest to the CPU chip. I first paralleled it with a 180K one, which increased the speed by about 40 per cent. Then I replaced it with a 47K resistor (in the Funkschau article it was paralleled with 47K, resulting in about 37K) and it still worked OK making the calculator more than twice as fast. The only trouble I had was finding a small enough resistor to put in the same place of the original one.

A SERIOUS EE QUIRK WITH THE TI-66 - R. Prins. With either a TI-59 or a TI-66 in the turn-on mode (Fix 9) press 1 EE 2 = and see "1. 02" in the display. Now try to change the exponent to five. With the TI-59 we can press EE 0 5 = and see "1. 05" in the display, and press INV EE and see the expected "100000." in the display. With the TI-66 the entry of the additional EE 0 5 steps yields "1. 05" in the display, but after pressing =, or almost any other function, the display reverts to "1. 02", and pressing INV EE verifies that the value in the display register really is still 100.

Now press 1 . 0 0 0 3 EE 2 = and see the expected "1.0003 02" in the display. Press EE 0 5 = and see "1.0003 05" in the display indicating that the exponent has been changed. Press INV EE and see "100030" in the display to verify the exponent change. Does this mean that somehow the exponent change does not work with integer mantissa's? No.

Press 2nd FIX 4 and repeat the original sequence of steps with the TI-66. After the 1 EE 2 = sequence the display will contain the expected "1.0000 02" . Try to change the exponent to five by pressing EE 0 5 = and the display will read "1.0000 05", and pressing INV EE to leave the scientific notation mode yields "100000.0000" in the display, verifying that the change of exponent had worked in the FIX 4 mode. The use of the display modes FIX 0 through FIX 3 will not permit changing the exponent in this manner. We conclude that for some reason the display must contain at least four digits to the right of the decimal point if this method of altering the exponent is to work. But there are more complex aspects to this EE quirk.

Enter 159.0457574 in the display and press INV 2nd Log. As expected you will see the "...Error.." indication. What does this have to do with the EE function? Now press 1 EE 1 = EE . . With the TI-59 pressing the decimal point after EE lets you add digits to the mantissa, rather than change the exponent. The same feature is available with the TI-66; see page 4-7 of the manual. But again, an unexpected effect will occur if the mantissa in the display does not include four digits to the right of the decimal point. After the decimal point entered previously, press 9 1 . (This second decimal point is essential to what is to follow with the TI-66, but does not affect the operation of the TI-59). Now press 1 1 1 1 1 1 and look at the display. The TI-59 will show "1.9111111 01"; but the TI-66 will show "1.111111 9" with only a single digit in the exponent. What sort of number is that? Press 2nd log INV EE and see 159.0457574, indicating that the exponent which showed as a single digit 9, was really 159! By changing the nine and the ones in the keyboard sequence above you can create any number between 1 EE 151 and 9.999999 EE 159. There doesn't seem to be much you can do with these numbers however. You can transfer them to the t register and bring them back to the display unchanged. If you store them in the data register, and then bring them back to the display they are altered again. After STO 01 RCL 01 the number created above appears as "1.111111 59" in the display, and 2nd Log INV EE yields "59.04575745" in the display. All of these curious things will not happen if the TI-66 is in FIX 4 mode so that there are four digits to the right of the decimal point in the mantissa; then the response of the TI-66 will be exactly like that of the TI-59.

THE LOGARITHM ALGORITHM FOR THE CC-40 - Louis Krumpleman of Richmond, Kentucky writes that he has disassembled about eighty per cent of the internal code of the CC-40. An example is the Ln algorithm:

1. The input argument is converted to the form $m \times 10^n$ where $0.1 < m < 1.0$. In CC-40 BASIC this may be accomplished by finding n = INT(Log(X)) + 1, and then m = X/(10^N).

2. $m' = \sqrt{10} \times m$, where

$\sqrt{10} = 3.162277660168$

3. $t = (m' - 1)/(m' + 1)$

4. $z' = t \times SA / SB$, where

$SA = ((A1 \cdot t^2 + A2)t^2 + A3)t^2 + A4$

$SB = (((B1 \cdot t^2 + B2)t^2 + B3)t^2 + B4)t^2 + B5$

and the constants are

A1	=	-22.764761571152
A2	=	197.6446297035
A3	=	-429.4834828658
A4	=	265.5224908516
B1	=	1.0
B2	=	-31.416484482822
B3	=	158.6018962727
B4	=	-258.9954899200
B5	=	132.7612454259

5. $Ln(X) = Ln(10) \times (n - 0.5) + z'$, and

$Ln(10) = 2.302585092994$

```
100 OPEN #1,"10.e=0",OUTPUT
105 A1=-22.764761571152
110 A2=197.6446297035
115 A3=-429.4834828658
120 A4=265.5224908516
125 B1=1
130 B2=-31.416484482822
135 B3=158.6018962727
140 B4=-258.99548992
145 B5=132.7612454259
150 S10=3.162277660168
155 E10=2.302585092994
190 X=0
200 X=X+.1
205 N=INT(LOG(X))+1
210 M=X/(10^N)
215 MP=S10*M
220 T=(MP-1)/(MP+1)
225 T2=T*T
230 SA=((A1*T2+A2)*T2+A3)*T2+A4
235 SB=(((B1*T2+B2)*T2+B3)*T2+B4)*T2
+B5
240 ZP=T*SA/SB
245 Y=E10*(N-.5)+ZP
250 REM PRINT USING"#.##############"
;Y:PAUSE
255 Y0=LN(X)
260 REM PRINT USING"#.##############"
;Y0:PAUSE
270 IF Y=Y0 THEN GOTO 200
280 PRINT #1,X,Y-Y0
300 GOTO 200
900 CLOSE #1
999 END
```

Editor's Note: Louis observes that this looks like a Hastings or Abramowitz approximation. To test whether it is truly the internal Ln algorithm I wrote the little CC-40 program at the right above. It calculates Ln(X) using Louis' algorithm in BASIC, compares the result with Ln(X) using the internal algorithm, and prints the result if the two values are not equal. For the range of x from 0.1 through 100 in 0.1 step, only 94 of the 1000 values did not match exactly, and the largest difference was 1.E-12 . If you wish to view the values calculated remove the REM at steps 250 and 260. Then the program will stop with each calculated value in the display, and you press CLEAR to continue.

ANY PC-200'S OUT THERE? - In response to a telephone call in June a TI representative indicated that PC-200's, the printers for use with the BA-55 and TI-66, should start appearing on retail shelves in limited quantities at mid-year. So far I haven't seen any in the Tampa Bay area. Has anyone seen them elsewhere?

ELLIPTIC TRANSFER FUNCTION FOR LOW PASS FILTERS - Robert Leaman

V10N1P2 reported that a program to calculate the elliptic transfer function for low pass filters had appeared in the December 27, 1984 issue of Electronic Design. Use of the program as published raised several deficiencies. The program does not have a print routine, the instructions neglect to say that the order must be odd, the example in Table II of the article shows more than the proper number of zeroes, and the user must enter the number of each pole to be calculated. The revised program permits use of similar instructions to the original program in Electronic Design, but also provides

* Elimination of some curious program steps, see steps 146 through 150 of the original program for example.
* Addition of an automatic printer sensing routine which stops with each output value in the display if a printer is not used, but continues to the next output if a printer is used.
* Automatic indication that the proper number of poles has been calculated.

The revised program appears on page 9. A sample printout appears at the right for the problem used in the article.

User Instructions:

1. Enter the filter order (n) and press A. The order will be displayed, and will be printed if a printer is used.

2. Enter the pass-band cutoff frequency (ω_c) in rad/sec, and press B. The input value will be displayed and printed.

3. Enter the stop-band frequency (Ω_s) in rad/sec, and press C. The calculator will run for about ten seconds and stop with the input value in the display. With a printer the value will be printed.

4.a. Enter the reflection coefficient (ρ) in percent and press D, or

4.b. Enter the the passband ripple (A_{max}) in db and press E. The passband ripple will be printed. The calculator will not stop to permit readout of A_{max} from the display.

With a printer all the zeros are printed and the calculator stops. Without a printer the calculator stops with the first zero in the display after about 30 seconds. Pressing R/S will bring additional zeros to the display. A zero in the display indicates that all the transmission zeros have been displayed.

5. Press 2nd-C' to print and display the stop band attenuation (A_{min}) in db.

6. Press 2nd-D' to calculate the poles. After about 45 seconds the calculator prints the real and imaginary parts of the first pole and proceeds to print the remaining poles at 45 second intervals. Without a printer, the calculator stops after 45 seconds with the real part of the first

```
                5.
                1.
  1.414213562

  .5773502692

  1.465436966
  2.165997414


  51.40644195


 -.3186089933
                0.

 -.2095320012
  .6875524703

 -.0582836518
  0.991367843
```

Elliptic Transfer Function - (cont)

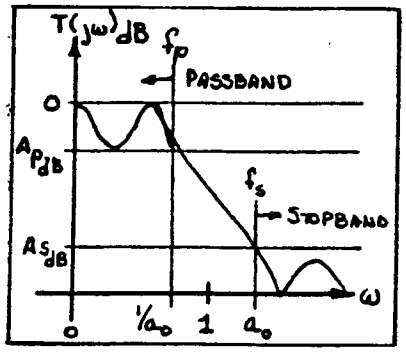
pole in the display. Pressing R/S brings the imaginary part to the display. Pressing R/S again causes the real part of the second pole to come to the display after another 45 seconds, and so on. A series of zeros indicates that all the poles have been displayed.

Program Listing:

000	76	LBL	080	02	2	160	99	PRT	240	71	SBR	320	43	RCL	400	01	01
001	81	RST	081	48	EXC	161	70	RAD	241	04	04	321	55	55	401	43	RCL
002	92	RTN	082	00	00	162	01	1	242	75	75	322	95	=	402	57	57
003	61	GTO	083	42	STO	163	06	6	243	02	2	323	35	1/X	403	42	STO
004	81	RST	084	07	07	164	42	STO	244	44	SUM	324	22	INV	404	02	02
005	05	5	085	55	÷	165	00	00	245	01	01	325	86	STF	405	36	PGM
006	05	5	086	43	RCL	166	00	0	246	97	DSZ	326	00	00	406	05	05
007	42	STO	087	08	08	167	42	STO	247	00	00	327	42	STO	407	15	E
008	00	00	088	95	=	168	01	01	248	02	02	328	05	05	408	43	RCL
009	04	4	089	34	FX	169	04	4	249	34	34	329	33	X²	409	58	58
010	42	STO	090	42	STO	170	42	STO	250	25	CLR	330	85	+	410	42	STO
011	02	02	091	11	11	171	02	02	251	98	ADV	331	01	1	411	03	03
012	02	2	092	43	RCL	172	01	1	252	81	RST	332	54	)	412	43	RCL
013	45	Y^X	093	07	07	173	04	4	253	76	LBL	333	34	FX	413	57	57
014	53	(	094	65	×	174	42	STO	254	18	C'	334	85	+	414	42	STO
015	43	RCL	095	43	RCL	175	03	03	255	71	SBR	335	43	RCL	415	04	04
016	10	10	096	08	08	176	43	RCL	256	00	00	336	05	05	416	36	PGM
017	75	-	097	95	=	177	01	01	257	05	05	337	95	=	417	04	04
018	01	1	098	34	FX	178	65	×	258	43	RCL	338	87	IFF	418	10	E'
019	54	)	099	42	STO	179	89	π	259	07	07	339	00	00	419	36	PGM
020	65	×	100	09	09	180	55	÷	260	33	X²	340	81	RST	420	04	04
021	43	RCL	101	43	RCL	181	02	2	261	65	×	341	45	Y^X	421	17	B'
022	15	15	102	11	11	182	55	÷	262	73	RC*	342	43	RCL	422	73	RC*
023	45	Y^X	103	33	X²	183	43	RCL	263	00	00	343	10	10	423	52	52
024	43	RCL	104	42	STO	184	10	10	264	45	Y^X	344	35	1/X	424	65	×
025	10	10	105	01	01	185	54	)	265	04	4	345	95	=	425	02	2
026	95	=	106	33	X²	186	39	COS	266	54	)	346	42	STO	426	95	=
027	72	ST*	107	75	-	187	55	÷	267	85	+	347	59	59	427	35	1/X
028	00	00	108	01	1	188	43	RCL	268	01	1	348	43	RCL	428	42	STO
029	69	DP	109	95	=	189	15	15	269	54	)	349	10	10	429	03	03
030	20	20	110	34	FX	190	95	=	270	35	1/X	350	85	+	430	00	0
031	85	+	111	85	+	191	85	+	271	28	LOG	351	01	1	431	42	STO
032	35	1/X	112	43	RCL	192	35	1/X	272	65	×	352	54	)	432	04	04
033	54	)	113	01	01	193	54	)	273	01	1	353	55	÷	433	36	PGM
034	55	÷	114	95	=	194	55	÷	274	00	0	354	02	2	434	04	04
035	02	2	115	72	ST*	195	02	2	275	94	+/-	355	95	=	435	13	C
036	95	=	116	00	00	196	55	÷	276	95	=	356	42	STO	436	42	STO
037	34	FX	117	69	DP	197	73	RC*	277	98	ADV	357	05	05	437	58	58
038	97	DSZ	118	20	20	198	03	03	278	99	PRT	358	00	0	438	32	X:T
039	02	02	119	97	DSZ	199	95	=	279	98	ADV	359	42	STO	439	42	STO
040	00	00	120	02	02	200	69	DP	280	81	RST	360	06	06	440	57	57
041	27	27	121	01	01	201	33	33	281	76	LBL	361	43	RCL	441	01	1
042	72	ST*	122	03	03	202	97	DSZ	282	19	D'	362	06	06	442	22	INV
043	00	00	123	43	RCL	203	02	02	283	98	ADV	363	65	×	443	44	SUM
044	92	RTN	124	07	07	204	01	01	284	71	SBR	364	89	π	444	52	52
045	76	LBL	125	99	PRT	205	91	91	285	00	00	365	55	÷	445	97	DSZ
046	12	B	126	81	RST	206	72	ST*	286	05	05	366	43	RCL	446	00	00
047	42	STO	127	76	LBL	207	00	00	287	43	RCL	367	10	10	447	03	03
048	08	08	128	14	D	208	69	DP	288	07	07	368	95	=	448	97	97
049	99	PRT	129	55	÷	209	20	20	289	33	X²	369	42	STO	449	43	RCL
050	81	RST	130	01	1	210	69	DP	290	35	1/X	370	02	02	450	58	58
051	76	LBL	131	00	0	211	21	21	291	85	+	371	00	0	451	71	SBR
052	11	A	132	00	0	212	43	RCL	292	01	1	372	42	STO	452	04	04
053	42	STO	133	54	)	213	01	01	293	54	)	373	01	01	453	71	71
054	10	10	134	33	X²	214	32	X:T	294	34	FX	374	36	PGM	454	43	RCL
055	29	CP	135	94	+/-	215	43	RCL	295	85	+	375	05	05	455	57	57
056	55	÷	136	85	+	216	10	10	296	43	RCL	376	17	B'	456	71	SBR
057	02	2	137	01	1	217	22	INV	297	07	07	377	65	×	457	04	04
058	95	=	138	54	)	218	67	EQ	298	35	1/X	378	43	RCL	458	71	71
059	22	INV	139	28	LOG	219	01	01	299	54	)	379	59	59	459	98	ADV
060	59	INT	140	65	×	220	69	69	300	65	×	380	95	=	460	69	DP
061	22	INV	141	01	1	221	75	-	301	43	RCL	381	42	STO	461	26	26
062	67	EQ	142	00	0	222	01	1	302	58	58	382	58	58	462	97	DSZ
063	00	00	143	94	+/-	223	54	)	303	95	=	383	32	X:T	463	05	05
064	68	68	144	54	)	224	55	÷	304	71	SBR	384	65	×	464	03	03
065	01	1	145	76	LBL	225	02	2	305	03	03	385	43	RCL	465	61	61
066	44	SUM	146	15	E	226	95	=	306	25	25	386	59	59	466	25	CLR
067	10	10	147	55	÷	227	42	STO	307	65	×	387	95	=	467	98	ADV
068	43	RCL	148	01	1	228	00	00	308	43	RCL	388	42	STO	468	98	ADV
069	10	10	149	00	0	229	01	1	309	57	57	389	57	57	469	81	RST
070	99	PRT	150	54	)	230	07	7	310	95	=	390	05	5	470	68	NOP
071	81	RST	151	22	INV	231	42	STO	311	71	SBR	391	42	STO	471	65	×
072	76	LBL	152	28	LOG	232	01	01	312	03	03	392	00	00	472	43	RCL
073	13	C	153	75	-	233	98	ADV	313	25	25	393	01	1	473	11	11
074	42	STO	154	01	1	234	73	RC*	314	65	×	394	05	5	474	95	=
075	00	00	155	95	=	235	01	01	315	43	RCL	395	42	STO	475	99	PRT
076	04	4	156	34	FX	236	65	×	316	56	56	396	52	52	476	69	DP
077	42	STO	157	42	STO	237	43	RCL	317	65	×	397	43	RCL	477	08	08
078	02	02	158	07	07	238	09	09	318	02	2	398	58	58	478	91	R/S
079	01	1	159	98	ADV	239	95	=	319	55	÷	399	42	STO	479	92	RTN

DARLINGTON'S ELLIPTIC FILTER ALGORITHMS - Robert Leaman

This program calculates the normalized transmission function pole and zero locations, and minimum stop-band rejection for odd order elliptic filters. The output data is normalized to the passband cutoff frequency (f_p); however, the algorithm is normalized to the geometric mean of the passband and stopband edge frequencies as shown by the figure at the right.



The program is a HP67 translation based on Prof. Darlington's unpublished algorithm for elliptical filters. The program calculates the necessary outputs "in place" and has no upper limit on the order of the filter. Since there is no upper limit, one must use care so that when very high orders are calculated, it is possible to have a calculator error when calculating $A_{min}(db)$. There will be an overflow. The program listing appears on page 11.

User Instructions:

1.a. Enter passband ripple (A_{pdb}) in db and press A, or

1.b. Enter the reflection coefficient (ρ) as a decimal fraction, not as a percent, and press 2nd-A'. ϵ is printed and displayed.

2. Enter the stopband to passband frequency ratio, (f_s/f_p) or ($a_o/(1/a_o)$, and press B. The input is printed and displayed.

3. Enter the order of the filter (n) and press C. The order is printed and displayed. If an even order is entered it is converted to the next higher order.

4. Calculate the normalized transmission zero frequencies and minimum stopband loss by pressing D. With a printer the zeros are printed followed by A_{min} and the calculator stops. Without a printer the calculator stops with the first zero in the display. Press R/S as required to bring the remaining zeros and A_{min} to the display. Once A_{min} is displayed, additional R/S's will return A_{min} to the display.

5. Press E to find the real and imaginary parts of the normalized transmission function poles. With a printer the real and imaginary parts for each pole are grouped with the real part printed first. Without a printer each real and imaginary part is brought in turn to the display by pressing R/S as required. When the last imaginary part has been displayed, additional R/S's will return a zero to the display.

```
 .3333333333
 1.414213562
           5.

 1.465436967
 2.165997415

 51.40644197

   318.60899-03
   22.040115-12

    209.532-03
   687.55247-03

   58.283652-03
   991.36784-03
```

A sample printout for the same problem illustrated on page 8 appears at the right above. This program yields equivalent answers to the program on pages 8 and 9, but in less time. For example, the poles are calculated in about 25 seconds, as opposed to 45 seconds.

Darlington's Elliptic Filter Algorithms - (cont)

Editor's Note: You will find that the signs of the real parts of the poles are opposite for the two programs. Bob indicates this is a result of the difference between the viewpoints of filter design engineers and control system engineers.

Program Listing:

```
000  76 LBL    080  69 DP     160  75  75    240  08  08    320  02  2     400  00  0
001  11  A      081  28  28    161  02  2     241  01  1     321  44 SUM    401  00  0
002  29 CP      082  92 RTN    162  44 SUM    242  00  0     322  09  09    402  00  0
003  22 INV     083  22 INV    163  09  09    243  44 SUM    323  43 RCL    403  00  0
004  77  GE     084  86 STF    164  43 RCL    244  08  08    324  17  17    404  00  0
005  16 A'      085  00  00    165  17  17    245  43 RCL    325  32 X:T    405  00  0
006  55  ÷      086  85  +     166  32 X:T    246  15  15    326  43 RCL    406  00  0
007  01  1      087  35 1/X    167  43 RCL    247  35 1/X    327  09  09    407  00  0
008  00  0      088  54  )     168  09  09    248  34 fX     328  22 INV    408  00  0
009  54  )      089  55  ÷     169  22 INV    249  71 SBR    329  77  GE    409  00  0
010  22 INV     090  02  2     170  77  GE    250  02  02    330  02  02    410  00  0
011  28 LOG     091  95  =     171  01  01    251  23  23    331  83  83    411  00  0
012  61 GTO     092  87 IFF    172  29  29    252  65  ×     332  22 INV    412  00  0
013  00  00     093  00  00    173  98 ADV    253  73 RC*    333  57 ENG    413  00  0
014  23  23     094  00  00    174  01  1     254  08  08    334  25 CLR    414  00  0
015  76 LBL     095  76  76    175  01  1     255  69 DP     335  98 ADV    415  00  0
016  16 A'      096  92 RTN    176  42 STO    256  38  38    336  98 ADV    416  00  0
017  33 X²      097  76 LBL    177  08  08    257  95  =     337  98 ADV    417  00  0
018  94 +/-     098  14  D     178  43 RCL    258  97 DSZ    338  98 ADV    418  00  0
019  85  +      099  60 DEG    179  04  04    259  07  07    339  61 GTO    419  00  0
020  01  1      100  98 ADV    180  85  +     260  02  02    340  00  00    420  00  0
021  54  )      101  09  9     181  24  CE    261  49  49    341  29  29    421  00  0
022  35 1/X     102  00  0     182  54  )     262  85  +     342  42 STO    422  00  0
023  75  -      103  55  ÷     183  45 Y×    263  24  CE    343  07  07    423  00  0
024  01  1      104  43 RCL    184  43 RCL    264  54  )     344  32 X:T    424  00  0
025  95  =      105  17  17    185  17  17    265  35 1/X    345  37 P/R    425  00  0
026  42 STO     106  95  =     186  55  ÷     266  65  ×     346  32 X:T    426  00  0
027  15  15     107  42 STO    187  02  2     267  73 RC*    347  65  ×     427  00  0
028  99 PRT     108  19  19    188  95  =     268  08  08    348  53  (     428  00  0
029  92 RTN     109  01  1     189  42 STO    269  95  =     349  43 RCL    429  00  0
030  61 GTO     110  42 STO    190  07  07    270  71 SBR    350  07  07    430  00  0
031  00  00     111  08  08    191  04  4     271  02  02    351  35 1/X    431  00  0
032  29  29     112  42 STO    192  48 EXC    272  23  23    352  33 X²     432  00  0
033  76 LBL     113  09  09    193  07  07    273  45 Y×     353  75  -     433  00  0
034  17 B'      114  04  4     194  42 STO    274  43 RCL    354  01  1     434  00  0
035  60 DEG     115  42 STO    195  10  10    275  17  17    355  54  )     435  00  0
036  38 SIN     116  07  07    196  71 SBR    276  35 1/X    356  42 STO    436  00  0
037  35 1/X     117  43 RCL    197  00  00    277  95  =     357  07  07    437  00  0
038  76 LBL     118  16  16    198  84  84    278  42 STO    358  95  =     438  00  0
039  12  B      119  34 fX     199  97 DSZ    279  18  18    359  32 X:T    439  00  0
040  42 STO     120  42 STO    200  07  07    280  00  0     360  65  ×     440  00  0
041  16  16     121  00  00    201  01  01    281  42 STO    361  53  (     441  00  0
042  61 GTO     122  71 SBR    202  96  96    282  09  09    362  43 RCL    442  00  0
043  00  00     123  00  00    203  33 X²     283  04  4     363  07  07    443  00  0
044  28  28     124  69  69    204  33 X²     284  42 STO    364  85  +     444  00  0
045  76 LBL     125  97 DSZ    205  65  ×     285  08  08    365  02  2     445  00  0
046  13  C      126  07  07    206  43 RCL    286  43 RCL    366  95  =     446  00  0
047  42 STO     127  01  01    207  15  15    287  09  09    367  22 INV    447  00  0
048  17  17     128  22  22    208  85  +     288  65  ×     368  37 P/R    448  00  0
049  29 CP      129  03  3     209  01  1     289  43 RCL    369  32 X:T    449  00  0
050  55  ÷      130  42 STO    210  54  )     290  19  19    370  55  ÷     450  00  0
051  02  2      131  08  08    211  28 LOG    291  95  =     371  02  2     451  00  0
052  95  =      132  43 RCL    212  65  ×     292  32 X:T    372  95  =     452  00  0
053  22 INV     133  04  04    213  01  1     293  43 RCL    373  92 RTN    453  00  0
054  59 INT     134  55  ÷     214  00  0     294  18  18    374  00  0     454  00  0
055  22 INV     135  53  (     215  95  =     295  71 SBR    375  00  0     455  00  0
056  67  EQ     136  43 RCL    216  71 SBR    296  03  03    376  00  0     456  00  0
057  00  00     137  09  09    217  04  04    297  42  42    377  00  0     457  00  0
058  61  61     138  65  ×     218  75  75    298  55  ÷     378  00  0     458  00  0
059  01  1      139  43 RCL    219  98 ADV    299  73 RC*    379  00  0     459  00  0
060  85  +      140  19  19    220  61 GTO    300  08  08    380  00  0     460  00  0
061  43 RCL     141  54  )     221  00  00    301  95  =     381  00  0     461  00  0
062  17  17     142  39 COS    222  29  29    302  97 DSZ    382  00  0     462  00  0
063  95  =      143  95  =     223  85  +     303  08  08    383  00  0     463  00  0
064  42 STO     144  71 SBR    224  53  (     304  02  02    384  00  0     464  00  0
065  17  17     145  00  00    225  33 X²     305  95  95    385  00  0     465  00  0
066  61 GTO     146  83  83    226  85  +     306  71 SBR    386  00  0     466  00  0
067  00  00     147  55  ÷     227  01  1     307  03  03    387  00  0     467  00  0
068  28  28     148  73 RC*    228  54  )     308  42  42    388  00  0     468  00  0
069  33 X²      149  08  08    229  34 fX     309  32 X:T    389  00  0     469  00  0
070  85  +      150  95  =     230  95  =     310  37 P/R    390  00  0     470  00  0
071  53  (      151  97 DSZ    231  92 RTN    311  32 X:T    391  00  0     471  00  0
072  33 X²      152  08  08    232  76 LBL    312  71 SBR    392  00  0     472  00  0
073  75  -      153  01  01    233  15  E     313  04  04    393  00  0     473  00  0
074  01  1      154  44  44    234  57 ENG    314  75  75    394  00  0     474  00  0
075  54  )      155  71 SBR    235  60 DEG    315  32 X:T    395  00  0     475  99 PRT
076  34 fX     156  00  00    236  03  3     316  71 SBR    396  00  0     476  69 DP
077  95  =      157  83  83    237  42 STO    317  04  04    397  00  0     477  08  08
078  72 ST*     158  71 SBR    238  07  07    318  75  75    398  00  0     478  91 R/S
079  08  08     159  04  04    239  42 STO    319  98 ADV    399  00  0     479  92 RTN
```

PLOTTING FUNCTIONS DEFINED BY POLAR COORDINATES - This program from the TISOFT library was brought to my attention by Robert Prins. The original author is Jean-Paul Van den Abeele. The program plots functions defined by polar coordinates of the form $r = f(\theta)$ where $-1 \leq f(\theta) \leq +1$. The function is plotted on three strips of printer paper. Each of the three strips is divided into two parts which are 20 characters wide and 20 lines long. The three strips are pasted together to form complete plots such as that on the first page of this issue. The points to be plotted in each of the six segments are determined by scanning the full range of input angles and selecting those which are in the segment to be plotted next. This all takes time--the rose on page 1 requires about one hour.

User Instructions:

1. Load the program and set the partitioning to 9 Op 17 .

2. Define the function $f(\theta)$ to be plotted. Press GTO A LRN and see 229 00 in the display. Key in the function of up to 11 steps ending in RTN. The angle θ will be in the display register at the start of the call of the function subroutine. Press LRN to exit the programming mode.

3. Enter the starting angle, the ending angle, the angular increment, and two other initialization constants:

   a. Enter $\theta_{min} - \theta_{step}$ in the display and press STO 00.

   b. Enter θ_{step} and press STO 01.

   c. Enter θ_{max} and press STO 02.

   d. Enter 2 STO 04 and 3 STO 05.

4. Press RST R/S to begin plotting.

Sample Problem:

The plot of a four leaved rose on the first page was made using the function defined at the right. The plotting parameters were:

$\theta_{min} - \theta_{step}$:	5 +/- STO 00
θ step:	5 STO 01
θ_{max} :	360 STO 02
	2 STO 04
	3 STO 05

227	76	LBL
228	11	A
229	65	×
230	02	2
231	95	=
232	38	SIN
233	92	RTN
234	00	0
235	00	0
236	00	0
237	00	0
238	00	0
239	00	0

Additional sample plots appear on page 13. Note that increasing the number of plotted points by reducing the step size does not always yield a more pleasing plot due to the influence of step size.

The print code for an "A" is 13 which is stored in the appropriate Op 01 through Op 04 locations for plotting. If the plotting routine calls for the same point to be plotted twice, then the print code is doubled to 26, equivalent to a "K" telling the user that that point is an intersection or a writeover.

Happy plotting! Robert Prins has defined a modification which will allow longer functions, and is also working on a fast mode, high resolution graphics version of this program.

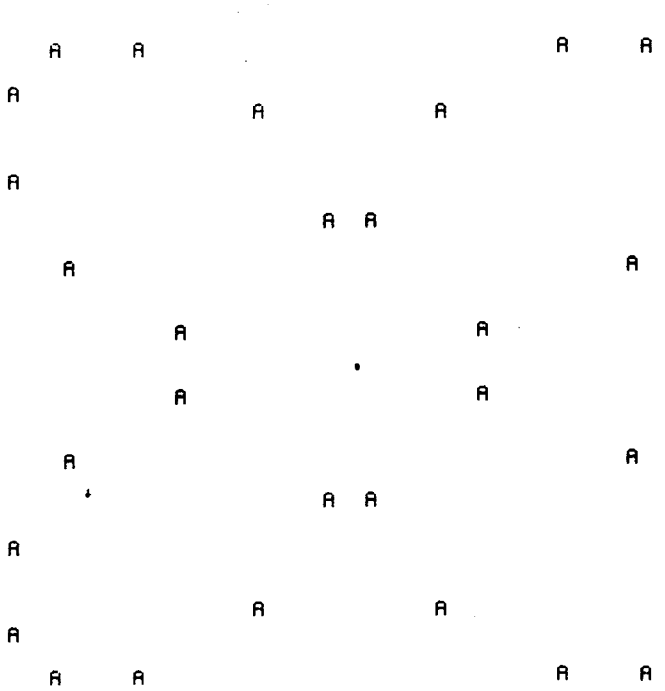
Plotting Functions Defined by Polar Coordinates - (cont)

Program Listing:

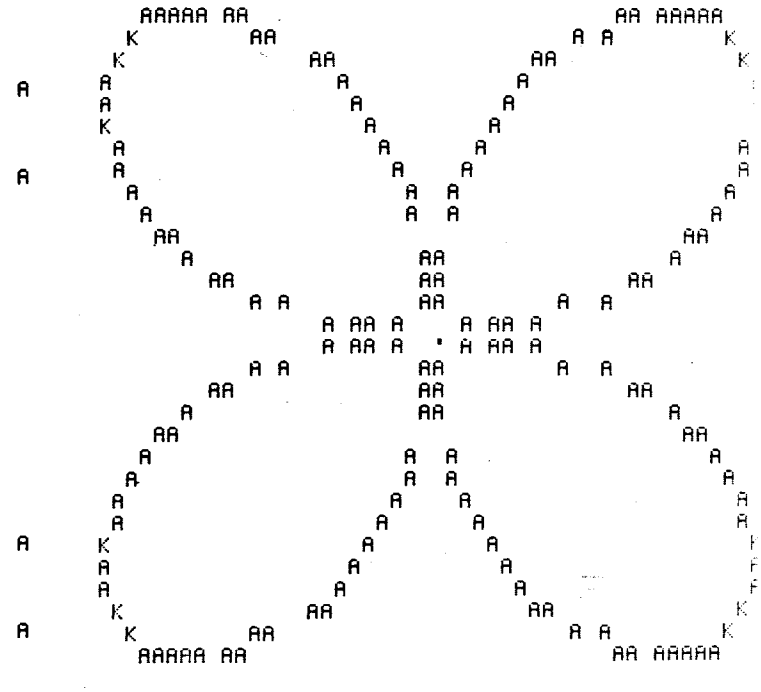
```
000  43 RCL   040  46  46   080  43 RCL   120  02  2    160  42 STO   200  95  =
001  00  00   041  69 DP    081  03  03   121  00  0    161  06  06   201  32 X:T
002  42 STO   042  29  29   082  37 P/R   122  95  =    162  93  .    202  04  4
003  03  03   043  69 DP    083  85  +    123  32 X:T   163  05  5    203  65  ×
004  76 LBL   044  26  26   084  01  1    124  43 RCL   164  95  =    204  43 RCL
005  14  D    045  13  C    085  95  =    125  07  07   165  55  ÷    205  07  07
006  43 RCL   046  69 DP    086  65  ×    126  22 INV   166  05  5    206  75  -
007  01  01   047  05  05   087  02  2    127  77  GE   167  85  +    207  01  1
008  44 SUM   048  01  1    088  09  9    128  14  D    168  01  1    208  05  5
009  03  03   049  42 STO   089  95  =    129  43 RCL   169  95  =    209  01  1
010  43 RCL   050  06  06   090  59 INT   130  05  05   170  59 INT   210  85  +
011  03  03   051  69 DP    091  42 STO   131  75  -    171  42 STO   211  08  8
012  32 X:T   052  29  29   092  06  06   132  01  1    172  08  08   212  00  0
013  43 RCL   053  43 RCL   093  32 X:T   133  95  =    173  65  ×    213  65  ×
014  02  02   054  09  09   094  85  +    134  65  ×    174  05  5    214  43 RCL
015  77  GE   055  32 X:T   095  01  1    135  02  2    175  94 +/-   215  04  04
016  00  00   056  09  9    096  95  =    136  00  0    176  85  +    216  85  +
017  76  76   057  00  0    097  65  ×    137  95  =    177  05  5    217  43 RCL
018  01  1    058  67  EQ   098  01  1    138  42 STO   178  95  =    218  08  08
019  00  0    059  00  00   099  09  9    139  09  09   179  44 SUM   219  95  =
020  42 STO   060  62  62   100  95  =    140  32 X:T   180  06  06   220  42 STO
021  09  09   061  13  C    101  59 INT   141  43 RCL   181  01  1    221  06  06
022  01  1    062  97 DSZ   102  42 STO   142  06  06   182  03  3    222  32 X:T
023  42 STO   063  04  04   103  07  07   143  22 INV   183  65  ×    223  74 SM*
024  06  06   064  34 FX    104  06  6    144  77  GE   184  01  1    224  06  06
025  76 LBL   065  02  2    105  00  0    145  14  D    185  00  0    225  61 GTO
026  13  C    066  42 STO   106  75  -    146  32 X:T   186  45 Y×   226  14  D
027  73 RC*   067  04  04   107  43 RCL   147  85  +    187  53  (    227  76 LBL
028  09  09   068  99 PRT   108  04  04   148  02  2    188  08  8    228  11  A
029  84 DP*   069  97 DSZ   109  65  ×    149  00  0    189  75  -
030  06  06   070  05  05   110  02  2    150  95  =    190  02  2
031  00  0    071  34 FX    111  00  0    151  32 X:T   191  65  ×
032  72 ST*   072  91 R/S   112  95  =    152  43 RCL   192  43 RCL
033  09  09   073  76 LBL   113  32 X:T   153  06  06   193  06  06
034  43 RCL   074  34 FX    114  43 RCL   154  77  GE   194  54  )
035  06  06   075  81 RST   115  07  07   155  14  D    195  95  =
036  32 X:T   076  43 RCL   116  77  GE   156  75  -    196  52 EE
037  04  4    077  03  03   117  14  D    157  43 RCL   197  95  =
038  67  EQ   078  11  A    118  32 X:T   158  09  09   198  22 INV
039  00  00   079  32 X:T   119  75  -    159  85  +    199  52 EE
```

More Sample Plots Using Different Step Sizes:

Step Size 10

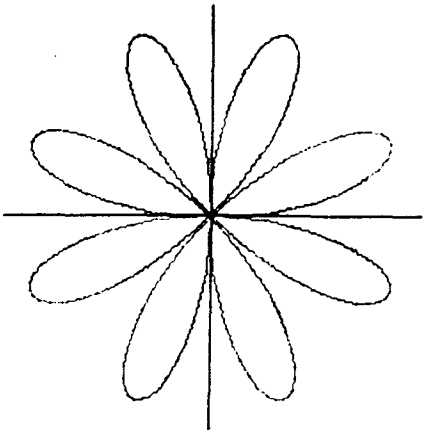


Step Size 2



POLAR COORDINATE PLOTTING WITH THE CC-40/HX-1000 - Palmer Hanson

Pages 12 and 13 illustrate the kind of techniques which must be used for plotting on a unidirectional printer if the coordinates to be plotted along the direction of paper motion do not increase monotonically. Printer-plotters such as the HX-1000 or the Radio Shack CGP-115 permit the paper to be moved both forward and back under computer control. One result of the increased capability is that functions defined by polar coordinates can be plotted directly without intermediate storage to sort the along paper coordinates. The plots at the right on page 1 are examples of the kind of plots which can be obtained with the CC-40/HX-1000 using the graphics mode. An enlarged copy of one of the plots and the program used to obtain it appears at the right.



```
100 OPEN #1,"1@",OUTPUT
110 PRINT #1,CHR$(19)
120 PRINT #1,"L(0,-107):1,(215,-107)
"
130 PRINT #1,"L(107,0),(107,-215)"
140 PRINT #1,"M(107,-107)"
150 PRINT #1,"O"
160 PRINT #1,"C4"
190 X1=0:Y1=0
200 FOR THETA=0 TO 360 STEP 2
210 RHO=100*SIN(4*THETA)
220 X2=INT(RHO*COS(THETA))
230 Y2=INT(RHO*SIN(THETA))
240 P$="L("&STR$(X1)&","&STR$(Y1)&"
),("&STR$(X2)&","&STR$(Y2)&")"
250 PRINT #1,P$
260 X1=X2:Y1=Y2
270 NEXT THETA
900 CLOSE #1
999 END
```

Program Comments:

Line 110 - CHR$(19) sets graphics mode.

Lines 120-130 draw the x and y axes.

Lines 140-150 move the pen to the intersection of the x and y axes and define that point as the origin for further plotting.

Line 160 changes the pen color to red.

Line 190 defines the origin as the first point for the draw command.

Lines 210-230 calculate the second point for the draw command. Since $\rho = \sin(4\theta)$ this will be an eight leaved rose.

Line 240 converts the first and second points for the first segment of the function plot into a string command which can be used by the printer. There are two important considerations when generating the string command: (1) the coordinates of the first and second points must be integers. If these arguments are not integers the computer will return the message "I/O error 80 #1". The integer functions in lines 220 and 230 ensure that requirement is met; and (2) this method of generating a string command is the only way that variable points can be plotted.

Line 250 delivers the assembled string command to the printer. The printer responds by drawing a line segment between the first and second points in the string command.

Line 260 changes the end point of the first line segment into the beginning point for the second line segment.

Line 270 sends the computer back to calculate the next end point. With this technique a four leaved rose can be plotted in about three minutes.

COMMENTS ON THE ML-15 RANDOM NUMBER GENERATOR - Laurance Leeds has been working with random number generators, both with the TI-59 and with the Model 100. He found an anomaly when he tried to mechanize the equations from page 54 of the Master Library manual in a different way. He writes:

> "If ML-15 is used to generate a uniformly distributed sequence on the interval (0,10) with any selected seed, the sequence generated will be different from that obtained from a program which you can readily write using identical equations and number values which are used in ML-15. I find that only the first two or three numbers are the same in the two sequences. Why?"

This is a good example of the effects of finite word length on calculations. The first equation on page 54 of the Master Library manual indicates that the next seed (x_{n+1}) is generated from the present seed (x_n) using the formula

$$x_{n+1} = (ax_n + c) \bmod m$$

In ML-15 the mod m function is mechanized by dividing the quantity in parentheses by m, taking the INV INT to save the fractional part of the quotient, and multiplying the result by m. Due to truncation effects this will not yield an exact mod m value. A procedure which will yield an exact mod m value is to take the quantity in the parentheses, divide it by m, take the INT (not the INV INT) to save the whole number portion, multiply the resulting integer by m, and subtract the resulting value from the original value in the parentheses. Readers who participated in the speedy factor finding competition (e.g., V8N5P19) will recognize that sequence as the method by which we implemented modulo arithmetic and extended the factor finding capability from twelve to thirteen digits. You might think that the small differences due to truncation can't make that much difference. The two short programs below show that indeed it can. The left hand program and printout uses the problem at the top of page 53 of the Master Library manual, and prints out the seed, the fractional part of the seed, and the random number using the ML-15 program as is. The right hand program and printout uses the same problem but mechanizes an exact mod m function (steps 043 through 060), and then returns to ML-15 to complete the random number generation (steps 061 through 070). The generated random numbers are the same for the first two iterations, but the fractional part of the seed is slightly different at the second iteration, 0.3599999 versus 0.36 . This small difference results in different random numbers at the next iteration!

000	76	LBL	0.32
001	11	A	0.32
002	36	PGM	5.87341
003	15	15	
004	10	E'	107766.36
005	93	.	0.3599999
006	03	3	7.34635
007	02	2	
008	36	PGM	140337.2535
009	15	15	0.2534955
010	15	E	3.5911
011	01	1	
012	36	PGM	57298.41926
013	15	15	0.41926172
014	11	A	1.63531
015	01	1	
016	00	0	14050.22008
017	36	PGM	0.22007753
018	15	15	9.05329
019	12	B	
020	76	LBL	178083.4428
021	12	B	0.4428279
022	43	RCL	7.5061
023	09	09	
024	99	PRT	143870.8083
025	22	INV	0.8082912
026	59	INT	7.29874
027	99	PRT	
028	36	PGM	139285.8527
029	15	15	0.852655
030	13	C	9.3007
031	99	PRT	
032	98	ADV	
033	61	GTO	
034	12	B	

0.32	000	76	LBL
0.32	001	11	A
5.87341	002	36	PGM
	003	15	15
107766.36	004	10	E'
0.36	005	93	.
7.34635	006	03	3
	007	02	2
140337.28	008	36	PGM
0.28	009	15	15
3.62026	010	15	E
	011	01	1
57942.44	012	36	PGM
0.44	013	15	15
7.29217	014	11	A
	015	01	1
139140.12	016	00	0
0.12	017	36	PGM
2.16811	018	15	15
	019	12	B
25830.76	020	76	LBL
0.76	021	12	B
2.63611	022	43	RCL
	023	09	09
36179.48	024	99	PRT
0.48	025	22	INV
6.91984	026	59	INT
	027	99	PRT
130907.04	028	02	2
0.04	029	04	4
9.57205	030	02	2
	031	09	9
	032	08	8
	033	65	×
	034	43	RCL
	035	09	09
	036	85	+
	037	09	9
	038	09	9
	039	09	9
	040	09	9
	041	01	1
	042	95	=
	043	75	-
	044	53	(
	045	24	CE
	046	55	÷
	047	01	1
	048	09	9
	049	09	9
	050	00	0
	051	01	1
	052	07	7
	053	42	STO
	054	07	07
	055	54	)
	056	59	INT
	057	65	×
	058	43	RCL
	059	07	07
	060	95	=
	061	36	PGM
	062	15	15
	063	71	SBR
	064	00	00
	065	38	38
	066	36	PGM
	067	15	15
	068	71	SBR
	069	00	00
	070	59	59
	071	99	PRT
	072	98	ADV
	073	61	GTO
	074	12	B

PARANOIA - George Thomson writes "Did you see Karpinski's article in BYTE about a program 'Paranoia'? I sent away as directed and got the IEEE Draft of the FP standards and one of the most horrendous programs you could imagine. The disk is called 'Paranoia' and it is all about finding out more than you'll ever know about internal computer arithmetic and whether there are 'Serious Flaws' or 'Defects'....."

The article George mentions,"Paranoia: A Floating-point Benchmark" by Richard Karpinski, appeared on pages 223 through 235 of the February 1985 issue of BYTE. The author of the program is William Kahan, who also wrote the "Mathematics Written in Sand ..." paper which has been discussed at length in earlier issues of TI PPC Notes (for example, see V9N2P15). Once again, the emphasis is on IEEE arithmetic and the proper use of guard digits. The complete Paranoia program is some 700 lines of BASIC, and the article contains instructions for obtaining a copy. The article also includes a limited version which tests for the use of a guard digit in addition and subtraction, what George Thomson calls "an itsy-bit of Paranoia".

The printout and program at the right are the implementation of the "itsy-bit" on the CC-40, where the only changes were extensive censoring of the comments. The output was much as we would have expected from our previous discussions of the number representation in the CC-40 (see V9N5P6), seven radix 100 digits. The program indicates that the CC-40 does have an add/subtract guard digit.

Testing the other computers that I have available was not quite so straightforward. Consider statement 30 at the right. BASIC implementations such as in the Model 100, Commodore 64, etc., do not permit the use of the ON combination of letters in a variable name. Other BASIC implementations on other computers have other lists of reserved words. Now consider statements 480 and 790 in the listing at the right which define the variables RADIX and RADIXMINUS. Many versions of BASIC allow the user to use several letters in the variable name, but only the first two characters are used by the computer. In such implementations RADIX and RADIXMINUS are seen as the same variable. Not surprisingly, the listing at the right will not operate properly in those computers.

```
Radix =  100
Precision =  7
Fpwidth =  1.E+14
Ulpone =  1.E-14
Add/subtract has a
 guard digit
```

```
10 OPEN #1,"10",OUTPUT
30 ONE=1
40 HALF=.5
50 ZERO=0
60 MINUSONE=-1
290 WIDE=ONE
310 WIDE=WIDE+WIDE
320 X=WIDE+ONE
340 Y=X-WIDE
350 Z=Y-ONE
370 IF (MINUSONE+ABS(Z))<ZERO THEN 3
10
400 Y=ONE
480 RADIX=WIDE+Y
490 Y=Y+Y
500 RADIX=RADIX-WIDE
520 IF RADIX=ZERO THEN 480
540 PRINT #1,"Radix = ";RADIX
590 PRECISION=ZERO
600 FPWIDTH=ONE
620 PRECISION=PRECISION+ONE
630 FPWIDTH=FPWIDTH*RADIX
640 Y=FPWIDTH+ONE
660 IF (Y-FPWIDTH)=ONE THEN 620
680 PRINT #1,"Precision = ";PRECISIO
N
700 PRINT #1,"Fpwidth = ";FPWIDTH
720 ULPONE=ONE/FPWIDTH
740 PRINT #1,"Ulpone = ";ULPONE
760 ONEMINUS=(HALF-ULPONE)+HALF
770 ULPRADIX=RADIX*ULPONE
790 RADIXMINUS=RADIX-ONE
800 RADIXMINUS=(RADIXMINUS-ULPRADIX)
+ONE
820 X=ONE-ULPONE
830 Y=ONE-ONEMINUS
840 Z=ONE-X
860 S=RADIX-ULPRADIX
870 T=RADIX-RADIXMINUS
880 U=RADIX-S
900 IF Y=ULPONE THEN 920
910 GOTO 960
920 IF T=ULPRADIX AND U=ULPRADIX THE
N 940
930 GOTO 960
940 PRINT #1,"Add/subtract has a gua
rd digit"
950 GOTO 980
960 PRINT #1,"Add/subtract lacks gua
rd digit"
980 CLOSE #1
990 END
```

Paranoia - (cont)

It was my unhappy experience with this kind of thing with FORTRAN in the mid-1960's that led to the formulation of Hanson's First Law of Higher Order Languages:

"NO PROGRAM WRITTEN FOR ONE COMPUTER WILL RUN AS IS ON ANY OTHER"

While it is possible to write programs which will travel well, very few such programs are written. The listing below makes the required changes in variable names to permit the "itsy-bit of Paranoia" to run on the Radio Shack Model 100.

```
Radix =  10
Precision =  14
fpwidth =  1E+14
Closest relative separation found is UlpOne =  1E-14
Add/subtract has a guard digit as it should.


30 WUN =       1.0
40 HALF =      0.5
50 ZERO =      0.0
60 NEGWUN = -1.0
290 WIDE = WUN
310 WIDE = WIDE + WIDE
320 X = WIDE + WUN
340 Y = X - WIDE
350 Z = Y - WUN
370 IF (NEGWUN + ABS(Z)) < ZERO THEN 310
460 Y = WUN
480 RADIX = WIDE + Y
490 Y = Y + Y
500 RADIX = RADIX - WIDE
520 IF RADIX = ZERO THEN 480
540 LPRINT "Radix = ";RADIX
590 PRECIS = ZERO
600 FPWIDT = WUN
620 PRECIS = PRECIS + WUN
630 FPWIDT = FPWIDT * RADIX
640 Y = FPWIDT + WUN
660 IF (Y-FPWIDT) = WUN THEN 620
680 LPRINT "Precision = ";PRECIS
700 LPRINT "fpwidth = ";FPWIDT
720 ULPWUN = WUN/FPWIDT
740 LPRINT "Closest relative separation found is UlpOne = ";ULPWUN
760 MINWUN = (HALF - ULPWUN) + HALF
770 URADIX = RADIX * ULPWUN
790 MRADIX = RADIX - WUN
800 MRADIX = (MRADIX - URADIX) + WUN
820 X = WUN - ULPWUN
830 Y = WUN - MINWUN
840 Z = WUN - X
860 S = RADIX - URADIX
870 T = RADIX - MRADIX
880 U = RADIX - S
900 IF Y = ULPWUN THEN 920
910 GOTO 960
920 IF T = URADIX AND U = URADIX THEN 940
930 GOTO 960
940 LPRINT "Add/subtract has a guard digit as it should."
950 GOTO 980
960 LPRINT "Add/subtract lacks guard digit, cancellation obscured."
980 END
```

Again, as we expected, the program finds that the Model 100 uses fourteen radix 10 digits, and has an add/subtract guard digit. The listing above travels well, because I carefully chose the variable names. With appropriate changes in the output commands (You ALWAYS have to change the output commands when going from one computer to the next.) the program has been run on the Radio Shack Color Computer, the Commodore 64, and the Sharp EL-5500II. The Color Computer and the Commodore 64 use 32 radix 2 digits. That probably explains why the Color Computer generates least significant digit "garbage" when one tries to strip off the higher order digits in the manner that we use to view the guard digits on the TI-59 (see V8N5P10). My tests show that the Color computer has a gurad digit but the Commodore does not

Paranoia - (cont)

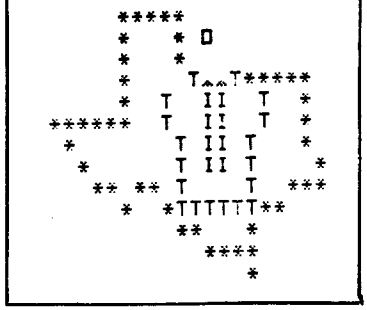
The program indicates that the Sharp uses ten radix 10 digits, and lacks a guard digit. What about the TI-59? As we would expect, an equivalent program shows that it uses thirteen radix 10 digits, and does not have a guard digit--in Paranoia's scheme of things, no notice is taken of TI's claim that the last three digits are guard digits, and only the ten most significant digits can be expected to be accurate. A listing of a program which will perform the "itsy-bit" test appears below.

```
76 LBL    064 77  GE    128 43 RCL    192 02  2    256 75  -    320 03  3
11  A     065 12  B     129 01  01    193 07  7    257 43 RCL   321 07  7
01  1     066 43 RCL    130 95  =     194 03  3    258 12  12   322 00  0
93  .     067 01  01    131 42 STO    195 03  3    259 95  =    323 00  0
00  0     068 42 STO    132 09  09    196 00  0    260 42 STO   324 69 OP
42 STO    069 07  07    133 43 RCL    197 02  2    261 07  07   325 04  04
01  01    070 76 LBL    134 10  10    198 69 OP    262 43 RCL   326 43 RCL
00  0     071 13  C     135 65  ×     199 04  04   263 01  01   327 07  07
93  .     072 43 RCL    136 43 RCL    200 43 RCL   264 75  -    328 32 X:T
05  5     073 05  05    137 08  08    201 11  11   265 43 RCL   329 43 RCL
42 STO    074 85  +     138 95  =     202 69 OP    266 15  15   330 11  11
02  02    075 43 RCL    139 42 STO    203 06  06   267 95  =    331 67  EQ
00  0     076 07  07    140 10  10    204 53  (    268 42 STO   332 15  E
93  .     077 95  =     141 43 RCL    205 43 RCL   269 06  06   333 61 GTO
00  0     078 42 STO    142 10  10    206 02  02   270 43 RCL   334 10 E'
42 STO    079 08  08    143 85  +     207 75  -    271 08  08   335 76 LBL
03  03    080 43 RCL    144 43 RCL    208 43 RCL   272 75  -    336 15  E
01  1     081 07  07    145 01  01    209 11  11   273 43 RCL   337 43 RCL
93  .     082 85  +     146 95  =     210 54  )    274 13  13   338 13  13
00  0     083 43 RCL    147 75  -     211 85  +    275 95  =    339 32 X:T
94 +/-    084 07  07    148 43 RCL    212 43 RCL   276 42 STO   340 43 RCL
42 STO    085 95  =     149 10  10    213 02  02   277 16  16   341 17  17
04  04    086 42 STO    150 95  =     214 95  =    278 43 RCL   342 67  EQ
43 RCL    087 07  07    151 32 X:T    215 42 STO   279 08  08   343 16 A'
01  01    088 43 RCL    152 43 RCL    216 12  12   280 75  -    344 61 GTO
42 STO    089 08  08    153 01  01    217 43 RCL   281 43 RCL   345 10 E'
05  05    090 75  -     154 67  EQ    218 08  08   282 14  14   346 76 LBL
76 LBL    091 43 RCL    155 14  D     219 65  ×    283 95  =    347 16 A'
12  B     092 05  05    156 03  3     220 43 RCL   284 42 STO   348 43 RCL
43 RCL    093 95  =     157 03  3     221 11  11   285 17  17   349 18  18
05  05    094 42 STO    158 03  3     222 95  =    286 43 RCL   350 22 INV
85  +     095 08  08    159 05  5     223 42 STO   287 08  08   351 67  EQ
43 RCL    096 32 X:T    160 01  1     224 13  13   288 75  -    352 10 E'
05  05    097 43 RCL    161 07  7     225 43 RCL   289 43 RCL   353 02  2
95  =     098 03  03    162 01  1     226 08  08   290 16  16   354 03  3
42 STO    099 67  EQ    163 05  5     227 75  -    291 95  =    355 01  1
05  05    100 13  C     164 69 OP    228 43 RCL   292 42 STO   356 03  3
43 RCL    101 03  3     165 04  04    229 01  01   293 18  18   357 03  3
05  05    102 05  5     166 43 RCL    230 95  =    294 69 OP    358 06  6
85  +     103 01  1     167 09  09    231 42 STO   295 00  00   359 00  0
43 RCL    104 03  3     168 69 OP    232 14  14   296 04  4    360 00  0
01  01    105 01  1     169 06  06    233 53  (    297 07  7    361 69 OP
95  =     106 06  6     170 02  2     234 43 RCL   298 06  6    362 02  02
75  -     107 04  4     171 01  1     235 14  14   299 03  3    363 69 OP
43 RCL    108 04  4     172 03  3     236 75  -    300 02  2    364 05  05
05  05    109 69 OP    173 03  3     237 43 RCL   301 00  0    365 91 R/S
95  =     110 04  04    174 04  4     238 13  13   302 00  0    366 76 LBL
75  -     111 43 RCL    175 03  3     239 54  )    303 00  0    367 10 E'
43 RCL    112 08  08    176 69 OP    240 85  +    304 69 OP    368 02  2
01  01    113 69 OP    177 04  04    241 43 RCL   305 01  01   369 07  7
95  =     114 06  06    178 43 RCL    242 01  01   306 02  2    370 01  1
42 STO    115 43 RCL    179 10  10    243 95  =    307 02  2    371 03  3
06  06    116 03  03    180 69 OP    244 42 STO   308 01  1    372 01  1
43 RCL    117 42 STO    181 06  06    245 14  14   309 03  3    373 05  5
03  03    118 09  09    182 43 RCL    246 43 RCL   310 03  3    374 02  2
32 X:T    119 43 RCL    183 01  01    247 01  01   311 05  5    375 06  6
43 RCL    120 01  01    184 55  ÷     248 75  -    312 01  1    376 03  3
04  04    121 42 STO    185 43 RCL    249 43 RCL   313 06  6    377 06  6
85  +     122 10  10    186 10  10    250 11  11   314 69 OP    378 69 OP
43 RCL    123 76 LBL    187 95  =     251 95  =    315 03  03   379 02  02
06  06    124 14  D     188 42 STO    252 42 STO   316 01  1    380 69 OP
50 I×I    125 43 RCL    189 11  11    253 15  15   317 06  6    381 05  05
95  =     126 09  09    190 04  4     254 43 RCL   318 02  2    382 91 R/S
22 INV    127 85  +     191 01  1     255 01  01   319 02  2    383 00  0
```

Finally, apparently after working with the full version of Paranoia, George Thomson states "... any machine worth its salt knows that the square root of zero is zero. But what happens if I write in BASIC X = 0; X = -X: Y = SQR(X)?" Some machines give an error message such as "Illegal Function Call". The CC-40, Model 100, and CoCo do not.

WHAT IS IT? - Robert Prins found a TI-59 Solid State module which has a white X painted where the label usually occurs. He found eleven programs in the module. Pgm 01 includes 480 steps, and seems to be the same as the the calculator diagnostic card which came with each TI-59. Pgm 09 is a printer demonstration program. With the calculator on a PC-100 pressing Pgm 09 B yields the printout at the right. Robert also noted that Pgm 11 contained a lot of alpha code. He was not sure what the remaining programs would do, and traded the module to me for a set of magnetic cards.

```
*****GREETINGS?*****

I AM A TI-59/PC-100A

I CAN BE PROGRAMMED
TO PRINT, GRAPH,
PLOT, AND EVEN DRAW
PICTURES.

FOR EXAMPLE, IF YOU
WILL PRESS THE
LETTER 'A', I WILL
DRAW THE TI SYMBOL
FOR YOU.
```



So far I have concentrated on the possible use of Pgm 11 of this strange module. The entire 950 program steps are filled with short routines such as those illustrated at the right for steps 374 through 403. Those two routines contain the alpha code for the words "EXTRA" and "FINAL". Further analysis of the routines suggests that Pgm 11 was intended as a library of words or acronyms which could be called to a user program and stored in a print register, where the print register is selected by the number in the display at the time the subroutine is called. Thus, to place the alpha code for "EXTRA" in print register 1, the user program would contain the code

374	42	STO		389	42	STO
375	00	00		390	00	00
376	01	1		391	02	2
377	07	7		392	01	1
378	04	4		393	02	2
379	04	4		394	04	4
380	03	3		395	03	3
381	07	7		396	01	1
382	03	3		397	01	1
383	05	5		398	03	3
384	01	1		399	02	2
385	03	3		400	07	7
386	84	DP+		401	84	DP+
387	00	00		402	00	00
388	92	RTN		403	92	RTN

1 Pgm 11 SBR 374

The 67 words or acronymns which are available include

align	analy	assy	autin	batt	bezel	broke
btcse	burn	cap	card	clip	cnd	compo
conn	cosmt	crom	data	date	dfect	diode
dsply	elect	enter	etch	extra	final	flex
hndin	ic	intrn	keybd	logic	loose	mech
misc	miss	mos	other	pack	pcb	plstc
port	prntr	pwrsb	read	rejct	resis	rev
scrap	sldr	swtch	thrpt	tip	today	total
tph	tpcs	vled	warp	work	write	wrong
xfrm	xstr	yield	zener			

You might think that these routines were called from one of the other programs in the module; however, I printed out all the other programs and found no call to Pgm 11. If you have a possible use for this module, or would like to help with the analysis of the remaining programs in the module, please let me know.

---

120 DIGIT SQUARE ROOT - Sven-Arne Wallin

This is one of three multi-precision square root programs which appeared in the latest issue of the Swedish newsletter Programbiten. The program calculates square roots for arguments in the range $1 < X < 100$ with a maximum of eight decimal places. I have not yet been able to push through a translation of the technique (it's slow going with my Swedish-English dictionary). I'll try to have that for the next issue.

The user instructions are concise. Load the program. Press A and see "10." in the display. Press RST 2nd Stflg 2nd IND 7 EE and see a zero in the display. Enter the argument and press R/S. The argument will be printed, followed by lines of the square root of the argument as they are completed. When the solution is complete the square of the solution is printed. The solution for the square root of 3 appears at the right. The square root also is available in R00 through R24, and the square of the answer is available in R25 through R73. The solution time is about 3 hours 45 minutes.

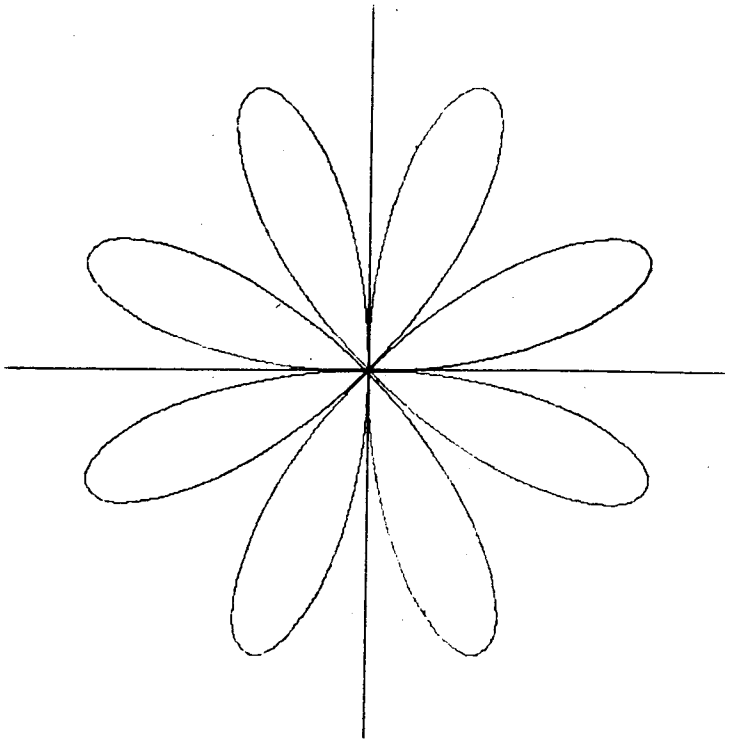
```
     3.


 17320.
 50807.
 56887.
 72935.
 27446.
 34150.
 58723.
 66942.
 80525.
 38103.
 80628.
  5560.
 69794.
 51933.
  1690.
 88000.
 37081.
 14618.
 67572.
 48575.
 67562.
 61414.
 15406.
 70302.
 99699.
```

```
299999999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    99999.
    84378.
    63412.
    94082.
    79733.
    19201.
    93419.
    44248.
    53212.
    24041.
    13943.
    73693.
    92240.
    18743.
    31980.
    84396.
    73280.
    71029.
    53259.
    71473.
    56908.
    70688.
    54140.
    36973.
    77594.
    90601.
```

000	92	RTN	045	01	01	090	02	2	135	05	5	180	54	)	225	02	02
001	91	R/S	046	54	54	091	08	8	136	42	STO	181	74	SM*	226	29	29
002	98	ADV	047	22	INV	092	82	HIR	137	78	78	182	79	79	227	75	-
003	99	PRT	048	77	GE	093	34	34	138	98	ADV	183	01	1	228	01	1
004	98	ADV	049	00	00	094	61	GTO	139	73	RC*	184	94	+/-	229	54	)
005	65	×	050	62	62	095	01	01	140	78	78	185	44	SUM	230	74	SM*
006	08	8	051	03	3	096	54	54	141	99	PRT	186	74	74	231	79	79
007	22	INV	052	94	+/-	097	86	STF	142	01	1	187	44	SUM	232	65	×
008	28	LOG	053	61	GTO	098	01	01	143	44	SUM	188	79	79	233	82	HIR
009	54	)	054	00	00	099	86	STF	144	78	78	189	97	DSZ	234	18	18
010	82	HIR	055	63	63	100	02	02	145	97	DSZ	190	77	77	235	54	)
011	05	05	056	87	IFF	101	61	GTO	146	76	76	191	01	01	236	22	INV
012	08	8	057	01	01	102	01	01	147	01	01	192	73	73	237	74	SM*
013	69	OP	058	01	01	103	54	54	148	39	39	193	82	HIR	238	74	74
014	17	17	059	04	04	104	22	INV	149	98	ADV	194	16	16	239	01	1
015	47	CMS	060	86	STF	105	86	STF	150	98	ADV	195	74	SM*	240	94	+/-
016	02	2	061	01	01	106	01	01	151	98	ADV	196	76	76	241	44	SUM
017	05	5	062	02	2	107	22	INV	152	25	CLR	197	43	RCL	242	74	74
018	42	STO	063	65	×	108	86	STF	153	81	RST	198	76	76	243	44	SUM
019	78	78	064	82	HIR	109	02	02	154	43	RCL	199	85	+	244	79	79
020	05	5	065	17	17	110	01	1	155	76	76	200	24	CE	245	97	DSZ
021	22	INV	066	54	)	111	00	0	156	42	STO	201	85	+	246	77	77
022	28	LOG	067	82	HIR	112	82	HIR	157	74	74	202	01	1	247	02	02
023	82	HIR	068	06	06	113	67	67	158	85	+	203	85	+	248	15	15
024	08	08	069	07	7	114	97	DSZ	159	01	1	204	42	STO	249	82	HIR
025	04	4	070	06	6	115	75	75	160	85	+	205	77	77	250	15	15
026	22	INV	071	82	HIR	116	00	00	161	42	STO	206	02	2	251	32	X:T
027	28	LOG	072	04	04	117	33	33	162	77	77	207	04	4	252	82	HIR
028	82	HIR	073	61	GTO	118	73	RC*	163	85	+	208	85	+	253	14	14
029	07	07	074	01	01	119	76	76	164	02	2	209	42	STO	254	42	STO
030	05	5	075	54	54	120	99	PRT	165	03	3	210	79	79	255	77	77
031	42	STO	076	22	INV	121	01	1	166	54	)	211	01	1	256	43	RCL
032	75	75	077	77	GE	122	44	SUM	167	42	STO	212	54	)	257	25	25
033	05	5	078	00	00	123	76	76	168	79	79	213	42	STO	258	83	GO*
034	65	×	079	56	56	124	97	DSZ	169	82	HIR	214	74	74	259	77	77
035	82	HIR	080	82	HIR	125	78	78	170	16	16	215	73	RC*	260	76	LBL
036	17	17	081	17	17	126	00	00	171	33	X^2	216	74	74	261	11	A
037	54	)	082	94	+/-	127	25	25	172	85	+	217	55	÷	262	25	CLR
038	82	HIR	083	82	HIR	128	02	2	173	82	HIR	218	82	HIR	263	93	.
039	06	06	084	06	06	129	49	PRD	174	16	16	219	18	18	264	01	1
040	04	4	085	22	INV	130	76	76	175	65	×	220	75	-	265	34	FX
041	07	7	086	87	IFF	131	97	DSZ	176	02	2	221	22	INV	266	33	X^2
042	82	HIR	087	02	02	132	76	76	177	65	×	222	59	INT	267	35	1/X
043	04	04	088	00	00	133	51	BST	178	73	RC*	223	29	CP	268	91	R/S
044	61	GTO	089	97	97	134	02	2	179	74	74	224	77	GE	269	00	0

THE RADIO SHACK CGP-115 AND THE HX-3000 RS-232 INTERFACE FOR THE CC-40

You may wonder at my interest in the CGP-115. The full size figure at the right should help you understand. The mode of operation is very similar to that of the HX-1000 Printer/Plotter, but the paper is twice as wide. Maurice Swinnen and the editors of TISOFT have some marvelous programs for the 99 operating with the CGP-115; but so far neither Maurice or I have been able to get the CC-40 and HX-3000 RS-232 Interface to operate satisfactorily with the CGP-115. I tried the delay trick that worked for the Radio Shack Model 7 Printer (V10N1P8), but it didn't help. Before the next issue I hope to have the HX-3000 modified for the parallel interface, and will try that way to connect the CC-40 to the CGP-115.



Any ideas would be appreciated.

PRINTER PAPER IN DETROIT - George Thomson writes that he has three packs of TI-59/PC-100 printer paper (three rolls per pack) which he could make available to someone in the Detroit area. Seven dollars per pack--you pick it up. You can call George at 313-835-0400.

STRUCTURAL ENGINEERING LIBRARY - G. Tohier forwarded a description of this module for the TI-59/PC-100. The description states "Our goal has not been to attain the maximum size or complexity of possible structures, but rather to bring with this module a significant decrease in the volume of uninteresting routine calculations faced by Civil or Structural Engineers in their day-to day activities. Emphasis has been put on comprehensibility of operating procedure, identification of input and output, and general ease of handling. All the analysis programs are based on the classical theory of linear elasticity, with deformations other than those due to bending being ignored." The programs include a diagnostic and

1. Single Span Beam with four options for the support conditions and four options for the load type.
2. Continuous Beam with four options for the load conditions.
3. Properties of sections.
4. Truss Beam
5. Portal Frame with three options for loading conditions, etc.

Has anyone seen a copy of this module?

---

100 RANDOM INTEGERS - The 100 random integer problem was originally proposed in V5N8P1 as "Produce a list of integers between two predetermined limits. 'Randomize' them in such a way that there is no repetition. In other words, each 'randomly' generated integer is unique. A minimum of 100 different integers are required, but any larger series is accepted with gratitude ..."

V6N1P9 described solutions by Richard Snow and Jeff Rosedale. The Rosedale program was relatively slow, requiring about 17 minutes 40 seconds to provide the 100 integers, but the program was capable of expansion to generate as many as 990 integers. The Snow program was reported as producing 100 integers in 2 minutes 35 seconds.

In V7N4/5P13 Sidney Hack reported a program which would provide 100 integers in 2 minutes 14 seconds when run in fast mode. Sidney reported a quirk which under some conditions would print extra zeroes at the end of the list. Since I have a predilection for working on quirks, I tried to analyze Sidney's program. I was unable to even reproduce the quirk. While studying the program I rcognized that his scrambling technique was similar to that described in David Borger's letter "Faster Algorithms" in the February 1982 issue of BYTE. I also found that Borger's technique was the same one used by Richard Snow and attributed to Personal Computing. The idea is to first generate the a list of numbers which meet the requirements, and then scramble them.

Working with the Snow and Hack programs I soon realized that the execution times were not "apples to apples". The 2 minute 35 seconds reported for the Snow routine was the time to produce the list in memory. An additional 30 seconds was required to print the list using a built-in INV-List sequence. Therefore, the time to produce the list as a printer output was actually 3 minutes 5 seconds.

The execution time quoted for the Hack program was also misleading. Step 3 of the initialization sequence included a 22 second wait while a list of 100 sequential integers was generated. After the seed was entered in step 4 it required an additional 2 minutes 14 seconds to "randomize" the sequence. The Hack program printed each integer in the randomized sequence as it was found. Therefore, the time to produce the list of integers as a printer output was 2 minutes 36 seconds.

There were other differences between the two programs. The Snow program allowed a selection of limits, while the Hack program was limited to the 100 sequential integers between 1 and 100. As a result, the Snow program required nearly 70 seconds to generate the list to present to the scrambler, while the Hack program generated the sequential list from 1 to 100 in only 22 seconds. The use of INV-List to print the Snow program's output provided sequence numbers for the list. The printout from the Hack program did not include sequence numbers. Finally, the Snow program had the randomized list stored in memory. The Hack program did not.

I decided to try to combine the best of both programs to see what could be accomplished in printing a scrambled list of the integers 1 through 100. The resulting program prints the integers as they are found, like the Hack program, but also stores them for subsequent use if needed, like the Snow program. The program prints the seed, uses 22 seconds to generate the list from 1 to 100, and uses another 73 seconds to complete the printout, for a total execution time of 1 minute 35 seconds. The randomized list is stored in data registers R00 through R99, but in reverse order from that printed. It turns out that printing the list as it is generated adds very little to the execution time. This illustrates Richard Snow's observation in V5N8P1 that the TI-59 will execute following commands while printing, but only if the PRINT commands (or Op 05 commands) are not too closely spaced.

100 Random Integers - (cont)

3.141592654

87.	28.
20.	100.
42.	47.
66.	67.
68.	92.
35.	36.
95.	84.
24.	15.
99.	74.
69.	96.
29.	21.
43.	44.
23.	83.
85.	56.
7.	51.
16.	98.
72.	1.
9.	91.
60.	58.
90.	63.
33.	79.
37.	25.
65.	3.
8.	27.
82.	64.
61.	59.
86.	57.
46.	77.
4.	39.
11.	30.
40.	34.
38.	5.
54.	26.
73.	19.
80.	41.
10.	88.
48.	50.
52.	53.
76.	17.
89.	45.
62.	13.
6.	78.
71.	12.
94.	81.
49.	55.
97.	22.
31.	18.
70.	14.
75.	32.
93.	2.

User Instructions:

1. Record the program in the power up partitioning 6 Op 17. The zeroes between 160 and 165 are needed as shown so that Lbl A appears at program locations 166/167.

2. From turn-on, enter side 1 of the magnetic card. See a 1 in the display.

3. Press A. See a zero in the display. Re-enter the same magnetic card. The printer will advance and a zero will appear in the display.

4. Enter a seed number and press R/S. The seed will be printed.

5. After about 22 seconds the printer will advance the paper and begin printing the randomized sequence. When the list is complete the calculator will stop with a zero in the display. It is still in fast mode, waiting for a new seed.

6.a. If you want a new list of randomized integers, enter a different seed and press R/S.

6.b. If you want to use the list previously generated, press RST to leave fast mode. You may then List the randomized sequence, but remember that it is in reverse order from the one printed.

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

Program Listing:

000	98	ADV	031	43	RCL	062	67	EQ	093	89	π	124	99	PRT	155	61	GTO
001	01	1	032	99	99	063	00	-00	094	95	=	125	82	HIR	156	00	00
002	00	0	033	82	HIR	064	78	78	095	22	INV	126	15	15	157	00	00
003	69	OP	034	08	08	065	82	HIR	096	59	INT	127	65	×	158	68	NOP
004	17	17	035	09	9	066	18	18	097	65	×	128	89	π	159	68	NOP
005	25	CLR	036	09	9	067	32	X:T	098	09	9	129	95	=	160	00	0
006	91	R/S	037	32	X:T	068	82	HIR	099	09	9	130	22	INV	161	00	0
007	99	PRT	038	82	HIR	069	17	17	100	95	=	131	59	INT	162	00	0
008	82	HIR	039	15	15	070	82	HIR	101	82	HIR	132	65	×	163	00	0
009	05	05	040	65	×	071	08	08	102	05	05	133	43	RCL	164	00	0
010	09	9	041	89	π	072	32	X:T	103	59	INT	134	98	98	165	00	0
011	09	9	042	95	=	073	82	HIR	104	42	STO	135	95	=	166	76	LBL
012	42	STO	043	22	INV	074	07	07	105	99	99	136	82	HIR	167	11	A
013	00	00	044	59	INT	075	61	GTO	106	67	EQ	137	05	05	168	02	2
014	85	+	045	65	×	076	00	00	107	01	01	138	42	STO	169	04	4
015	01	1	046	01	1	077	84	84	108	24	24	139	99	99	170	00	0
016	75	-	047	00	0	078	82	HIR	109	82	HIR	140	97	DSZ	171	42	STO
017	72	ST*	048	00	0	079	18	18	110	17	17	141	98	98	172	00	00
018	00	00	049	95	=	080	63	EX*	111	63	EX*	142	01	01	173	36	PGM
019	97	DSZ	050	82	HIR	081	99	99	112	99	99	143	18	18	174	02	02
020	00	00	051	05	05	082	82	HIR	113	82	HIR	144	43	RCL	175	71	SBR
021	00	00	052	59	INT	083	08	08	114	07	07	145	00	00	176	40	IND
022	15	15	053	42	STO	084	99	PRT	115	61	GTO	146	99	PRT	177	00	00
023	98	ADV	054	99	99	085	09	9	116	01	01	147	82	HIR	178	97	DSZ
024	25	CLR	055	67	EQ	086	08	8	117	24	24	148	17	17	179	99	99
025	69	OP	056	00	00	087	42	STO	118	73	RC*	149	42	STO	180	99	PRT
026	20	20	057	84	84	088	98	98	119	98	98	150	98	98	181	00	0
027	43	RCL	058	32	X:T	089	32	X:T	120	63	EX*	151	82	HIR	182	00	0
028	98	98	059	09	9	090	82	HIR	121	99	99	152	18	18	183	00	0
029	82	HIR	060	08	8	091	15	15	122	72	ST*	153	42	STO	184	00	0
030	07	07	061	22	INV	092	65	×	123	98	98	154	99	99	185	00	0

FROM THE EDITOR - An introductory offer for a computer publication, CompuWare Monthly, has been enclosed with the U.S. distribution. This is offered only as a possible convenience to the members. No endorsement of any kind by our club is either intended or implied.

In June two of our members, George Thomson and Albert Smith, became interested in the Chinese Remainder Theorem. Their interests were completely different. George was working with solutions for linear equations. Albert was working with Mark DiPippo's PPX program "The Remainder Game (PPX 918167)". Both were looking for a copy of Milton Brown's program for the "Chinese Remainder Theorem", PPX 268008, but had noted that it was not included in our informal exchange. However, I found that we did have other programs by Milton Brown. I asked the holders of those programs, Paul Sperry and Myer Boland, to send Milton's address to George so that he could try to make contact. George obtained a copy of the program, and sent a copy to Albert, and to me for inclusion in our exchange. George also forwarded his sincere thanks to all those who helped. Albert wrote "It is somewhat gratifying to receive help from people unknown to me". My response is that that is the difference between a club such as ours and a publication. In the next issue I hope to have some discussion on the Chinese Remainder Theorem. George has already submitted a BASIC program.

A number of members have expressed interest in conversion of TI-59 programs to BASIC for use with their personal computers. I suspect that such conversion may satisfy a real need--there seems to be a dearth of technically oriented programs for the home computers. Larry Leeds and Myer Boland have been particularly active in that area. Myer and I tend toward literal translations, but find that we often run into snags because of the tricks used by so many of the TI-59 programmers; even so, in a coming issue I hope to present some of the guidelines one needs for literal translations. In contrast, Larry tends to use the TI-59 programs as resource material only. To each his own ...

The life of a CC-40 owner continues to be filled with frustration. There is no known way to save programs--the 18K version of the CC-40 relieves the problem somewhat, but even 18K is rapidly filled. At first I thought that my problems with the RS-232 interface reflected my limitations in electronics. However, Maurice Swinnen seems to have fared no better. Has anyone out there had any real success with the RS-232 interface using the serial port?

TABLE OF CONTENTS

Magnetic card service will continue to be available for programs published in TI PPC Notes - one dollar per card plus a stamped and self-addressed envelope.

Single CCL-144 cleaning strips are also available to club members for two dollars each.