TI PPC NOTES

NEWSLETTER OF THE TI PROGRAMMABLE CALCULATOR CLUB

P.O. Box 1421, Largo, FL 34294

Volume 10, Number 3 Third Quarter 1985

From the Editor:

Several members have asked about a membership list. As part of the subscription for for 1984 we asked if members would like to have their names published; nearly half said no. If you would like to have your name and address published in a membership list send a card or letter to that effect and we will distribute a list with the next issue.

As you can see by the distribution of this "Third Quarter" newsletter in early November I have been unable to catch up. Therefore, the final issue for 1985 will be distributed in early 1986.

In the last issue I noted that Ralph Snyder had retired and that we would like to identify a new senior member. From the responses so far it appears that at eighty Larry Leeds has the honors.

The last issue also noted that members have expressed interest in the conversion of TI-59 programs to BASIC. A number of members have responded with suggestions that we publish a list of conversions. If you have either conversions or equivalents and are willing to share them with other members send the information to me and I will try to publish a listing similar to that providing access to PPX programs. In the meantime if you have need of a particular program let me know. I will try to put you in touch with someone who has the program. If we can't find a program that satisfies your needs I will request help in the next issue.

Since the next issue certainly won't be out until 1986 - HAPPY HOLIDAYS!

TABLE OF CONTENTS

Magnetic card service is available for the programs in this issue, and for 1983 and 1984 issues, for the price of one dollar per card plus a stamped and self-addressed envelope.

CCL-144 Cleaning Strips are two dollars each. No checks, please.

ERRATA

Roots of a Cubic on the TI-57 - The equation in the example at the bottom of V1ON1P23 is incorrect. The equation should have been $x^3 + 6x^2 + 11x + 6 = 0$.

MAILBAG

"I am the bewildered owner of a CC40, HX-3000 RS232 Interface, HX-3100 MODEM, and Memo Processor Software Package. What are my options for storage devices? ..." T.H.

"I would greatly appreciate it if you could inform me when someone produces a (recording) interface from the RS-232 output." R. B.

"Although I see an increased emphasis on the CC-40 in 'TI PPC Notes' I hope you'll continue on the 'good ole' TI-58 and -59. I assume that the many expert users will continue to contribute the excellent articles on these machines as they have in the past." W.M.

"Page numbers - let's start with 1 and continue on up throughout a volume; issue 2 might start with page 35, etc. The next volume starts with page 1 again." C.W.

"I like the PPC evolving perspective as we all grow older and need to focus more on repair and maintenance. I am writing a bibliography of applications in architecture and planning which I will send along for your continuing series. I liked the navigation bibliography ..." R.E.

"Please send me another CCL-144 cleaning strip ... ... They sure work like magic. Too bad TI didn't come out with them at the same time they put out the TI-59! Did I ever tell you that I bought a brand-new TI-59 in 1977 and fought with it for over aa year trying to get it to read cards and finally exchanged it for a beat up looking 'reconditioned' model that I now have. I believe all my first one needed was a good cleaning with a CCL-144." C.R.

BOOK REVIEW - The Calculator Afloat: A Mariner's Guide to the Electronic Calculator. H. Shufeldt and K. Newcomer. U. S. Naval Institute, Annapolis, Maryland. 220 pages. I purchased this book when it was offered at a sale price to U. S. Naval Institute members. The book is essentially a second edition of a former book, Slide Rule for the Mariner, which was published in 1972. The foreword states:

> "... The Calculator Afloat contains the same formulae, in some cases restated for use with the calculator, plus many others. It is intended for users of all types of scientific calculators, from the simplest to the most advanced programmable types. ... ... We have confined ourselves for the most part to stating the formulae designed to solve the various problems and left the keying procedure to the readers, on the assumption that they are familiar with the operation of their own calculators. ..."

If you are looking for solutions in equation form this book may help. If you are looking for program listings you will be disappointed. If you would like to borrow the book send two dollars to cover postage.

EVEN MORE ON CALCULATORS IN NAVIGATION - P. Hanson. In V9N6P13 and V10N2P5 I listed fourteen articles on the use of calculators and personal computers in navigation which had appeared in NAVIGATION, the Journal of the Institute of Navigation (U.S.A.). I have found four more articles. The same availability applies. Send one dollar for each article to cover the costs of copying and handling.

1. "Longitude by Lunar Observations and the Pocket Calculator", by D. W. Kerst, Vol. 25, No. 4, Winter 1978-1979, pp 430-433.

2. "Hand Held Calculators - An Evaluation of Their Use for Celestial Navigation", by A. Bralove, Volume 25, No. 4, Winter 1978-1979, pp 434-446.

3. "Sight Reduction with Matrices", by R. G. Huenemann, Vol. 25, No. 4, Winter 1978-1979, pp 447-448.

4. "New Power for Calculator-using Navigators", by Kenneth E. Newcomer, Thirty-eighth Annual Meeting, June 1982, pp 25-27.

CALCULATOR PROGRAMS FOR CHEMICAL ENGINEERS - Page 48 of issue #28 of the Educalc catalog lists two books by the staff of Chemical Engineering. The abstract in the catalog states:

> "A whole library of useful programs - general purpose ones like curve-fitting and statistics, and quite specific ones for design or everyday operations like physical properties estimation, equipment sizing, and boiler efficiency. Volume II contains recently published programs useful in process design, maintenance, lab and pilot plant research. It also has a bibliography of chemical engineering programs from other sources. All programs were written in two versions - one is for the HP-67, HP-97, or HP-41 calculators. The other version runs on the TI-59 and in many cases will also fit into the TI-58."

Volume I (softbound - stock #E-323) with 328 pages is $29.95. Volume II (hardbound - stock #E324) with 258 pages is $37.50. Add a one dollar shipping/handling charge to order from Educalc Mail Store, 27953 Cabot Road, Laguna Niguel, CA 92677.

PASCAL TO C TRANSLATOR - Milton Brown, the author of the Chinese Remainder Theorem program discussed elsewhere in this issue offers a translator program which will enable you to translate the natural language PASCAL programs to the powerful and abstract C language. He states

> "By this translation you can obtain the speed, power, and transportability of the C programming language from you existing PASCAL programs. Once your programs are translated, the precision can be easily extended by the natural C concepts of 'long' for integers and 'double' for floating point numbers. This enables extension for integers past the 32,767 range limit for PC PASCAL compilers. The translated programs also allow the utilization of the higher execution speed of C compilers, even on PCs. It also allows the ability to use these C programs on mini-computers and main-frame computers. The interactive translation of the TRANSLATOR also allows easy learning of the C programming language concepts, and the development of good programming techniques."

Send $129.95 to Milton Brown Software, 935 Chester Avenue, San Marino, CA 91108.

CODEWORKS - A copy of the first issue of this magazine arrived in August. The magazine will be devoted to programming, with emphasis on the use of BASIC. The idea of using Data statements for data entry in the sorting program which appears elsewhere in this issue came from the magazine. The subscription price is $24.95 per year for six issues. United States only. The second page of the magazine makes the following offer:

"Sample copies: if you have a friend who would like to see a copy of CodeWorks just send the name and address and we will send a sample (at no cost)."

Write to CodeWorks, 3838 South Warner Street, Tacoma, WA 98409. Please mention that you saw it in TI PPC Notes.

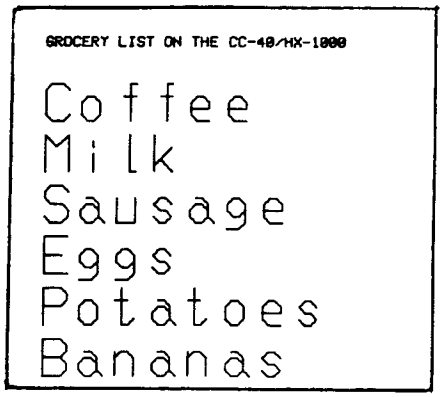
FINDING PI - The method of approximating the value of pi with the ratio 355/113 is well known. The answer 3.141592920353 is correct to seven digits. Ronald Wagner tells me of another method which he obtained at a computer trade show many years ago - divide 2143 by 22 and take the square root two times. On my TI-59 this yields 3.141592652582 which is correct to nine digits. Ron says that this was described as the Hicks method, or was it the "hick's" method. Can anyone provide a source?

MY HOME COMPUTER - My home computer is a tool
That causes friends to stare and drool.
I now have power to explore
Those worlds I never knew before.

It costs a lot, but that's OK
"I'm worth it", I've been heard to say
Who wouldn't spend a couple of grand
To have such power at his hand?

I stare in awe at this machine
They say can do most anything
But somehow all that awe gets missed
When I type in my grocery list.

E. E. Bard

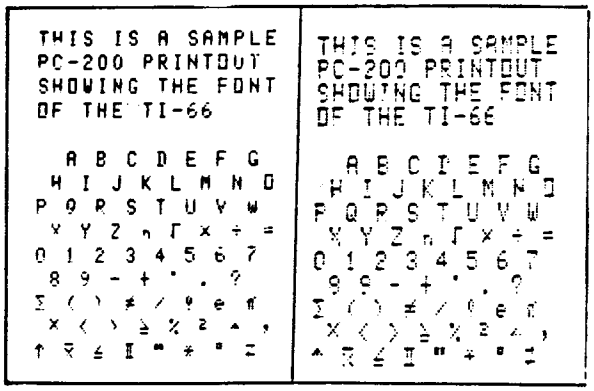


SPEEDY FACTOR FINDERS - P. Brassine, R. Fruit and L. Leeds. V9N6P4 presented one version of a modulo 210 factor finder in BASIC. Execution times were compared for the TI-59, the CC-40, and the Model 100. Phillip Brassine converted the BASIC program for use on his IBM PCJr defining the variables D, N, R, and S as double precision. His run times for various problems are very similar to those reported for the Model 100; for example, his program declares 9999999967 to be prime in 18 minutes 45 seconds versus the 18 minutes 8 seconds for the model 100.

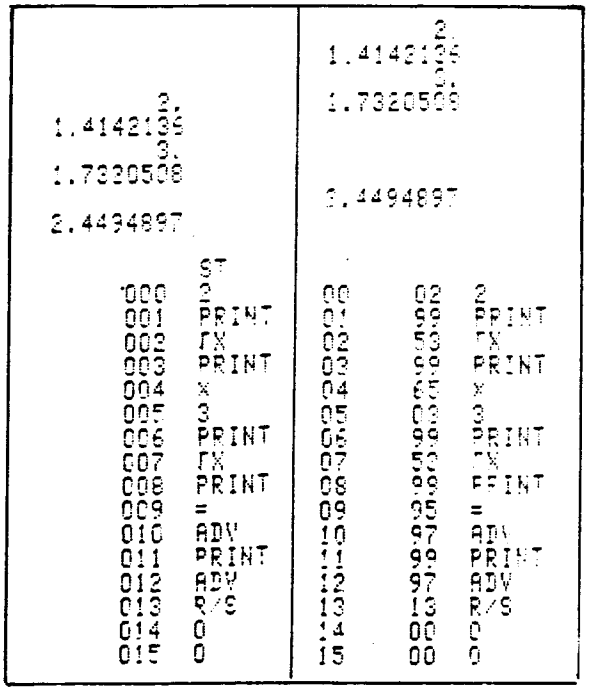
Robert Fruit converted the modulo 210 speedy factor finder program for use with Turbo Pascal 3.0. Since this is a compiled language we would expect the solution to be much faster, and it is. Bob tried his program in three configurations. The times to declare 9999999967 to be prime were:

Radio Shack Model 16 (Z80 at 4 MHz)	2 minutes 52 seconds
IBM PC	1 minute 27 seconds
IBM PC using 8087 chip	36 seconds

THE PC-200 ARRIVES - P. Hanson. In V9N1P7 and V9N2P9 Dave Leising presented printouts from a prototype PC-200 printer he had obtained from TI. V9N3P22 reported that TI did not expect the PC-200 to become available to the public until the last quarter of 1984. I have been watching the advertisements and catalogs for information ever since. In early September the latest issue of the Educalc catalog indicated the printer was now available and I ordered one.



The first surprise was that TI had changed the print font from that used with Dave Leising's prototype! The printouts at the right compare the two fonts for the sample printout that Dave had used in V9N1P7. The left-hand printout is from Dave's prototype. The right-hand printout is from the unit I received from Educalc. The number of lines per inch has remained the same, but the characters are taller, and the space between lines has been reduced. The net effect is reduced legibility. This only confirms the old bookkeeping rule that small figures adequately spaced are more legible than larger figures closely spaced.



The second surprise was that while an Advance with the TI-66 moved the paper forward one line as expected, an Advance with the BA-55 moved the paper forward three lines. The printouts at the right illustrate the effect. The left-hand printout is with a TI-66. The right-hand printout is with a BA-55. In each case the program was run, and a List was commanded immediately afterward. Interesting!

In V9N1P4 I had observed that the BA-55 did not use the A.O.S. logic. In V9N3P17 Lem Matteson noted that the BA-55 used "Adding Machine Logic" which was consistent with business practice. Is there also an "Adding Machine Logic" rule that an Advance should cause the paper to move forward three spaces, or do I have a defective BA-55?

One adverse comment--there is no advance key on the PC-200. Thus there is no way to move the paper forward except to perform repeated 2nd-Adv sequences from the keyboard. That is downright inconvenient.

The PC-200 is listed as Stock # PC-200 on page 42 of issue #28 of the Educalc catalog. The price is $69.95 plus a shipping/handling charge of $1.00 for a total of $70.95. Write to:

Educalc Mail Mail Store
27953 Cabot Road
Laguna Niguel, CA 92677

Mention that you saw it in TI PPC Notes please.

---

400 DIGIT SQUARE ROOT - Sven-Arne Wallin and Robert Prins. V10N2P20 presented a 120 digit square root program by Sven-Arne Wallin which had been published in the Swedish newsletter Programbiten. Robert Prins writes:

> "I sent a copy of the article "Personal Calculator Algorithms I: Square Roots" which appeared in HP Journal of May 1977. Shortly after I sent him that article I received his program with the question if I could optimize it as it was rather slow. His program needed about half an hour to generate the first ten digits. I succeeded in optimizing it considerably so that my version needs only about one minute 45 seconds to do the same ... ... (This program made in cooperation between Sven and me, two persons who know each other only from articles in Programbiten, once again shows the difference between a club and a magazine. By the way, why don't we have a membership list?)"

User Instructions:

1. Enter the program listed on the next page.

2. Enter the argument and press A. The argument must be greater than 1 and less than 100.

3. Enter the number of groups of ten digits and press R/S. The number must not exceed 40.

4. When a flashing "1." appears press 7 and then EE to start the calculations in fast mode.

5. If a printer is being used then each group of ten digits is printed as it is completed, and the calculator stops with a zero in the display when calculations are complete. A sample printout of 400 digits of the square root of 3 appears at the right.

6. To generate additional groups of ten digits after the calculator has stopped you can enter the number of additional groups and press C (for continue). You must make sure that the total number of groups entered using modes A and C does not exceed 40.

7. With the continue mode the calculator will stop with a flashing "1. 10" in the display. Again, you press 7 and then EE to start calculations in fast mode. Additional groups of ten digits will be printed as they are completed.

8. You may repeat the continue mode as many times as you like providing that the total number of groups of ten digits does not exceed 40.

9. For operation without a printer change the command at program step 088 from Nop to R/S. The program will stop to permit readout of each group of ten digits as it is completed. Press R/S to proceed to the next group.

```
                3.

1732050807.
5688772935.
2744634150.
5872366942.
8052538103.
8062805580.
6979451933.
 169088000.
3708114618.
6757248575.
6756261414.
1540670302.
9969945094.
9989524788.
1165551209.
4373648528.
 932319023.
 558206797.
4820101084.
6749232650.
1531234326.
6903322886.
6506722546.
6892183797.
1227047131.
6603678615.
8801904998.
6537379859.
3894676503.
4750657605.
 756618348.
1296061009.
4760218719.
 325083145.
8295239598.
3299778982.
4508288714.
4638329173.
4722416398.
4587855397.
```

```
               11.

3316624790.
3553998491.
1493273667.
 686683927.
 885455893.
5359705868.
2146116484.
6426090438.
4670884339.
9128290650.
9070125578.
4952745659.
```

```
               97.

9848857801.
7961047217.
4621141491.
7624481696.
1362874427.
6417172315.
4529836440.
5837076786.
3009320078.
4115425762.
4381011988.
7174919971.
```

400 Digit Square Root (cont)

Sven's program from V10N2P20 would find 120 digits of the square root of 3 in about 3 hours 45 minutes. This program will find 120 digits of the square root of three in 1 hour 12 minutes, and will find 400 digits in 12 hours 14 minutes. Each group of ten digits takes somewhat longer than the preceeding group. It requires about 30 minutesa to get the fortieth group.

Program Listing:

000	92	RTN	040	22	INV	080	24	CE	120	01	01	160	97	DSZ	200	00	0
001	53	(	041	74	SM*	081	01	1	121	14	14	161	07	07	201	76	LBL
002	05	5	042	06	06	082	00	0	122	02	2	162	01	01	202	11	A
003	00	0	043	73	RC*	083	48	EXC	123	42	STO	163	25	25	203	32	X:T
004	75	-	044	06	06	084	00	00	124	07	07	164	09	9	204	69	DP
005	43	RCL	045	77	GE	085	48	EXC	125	43	RCL	165	94	+/-	205	00	00
006	08	08	046	00	00	086	09	09	126	08	08	166	61	GTO	206	69	DP
007	42	STO	047	60	60	087	99	PRT	127	48	EXC	167	00	00	207	05	05
008	07	07	048	43	RCL	088	68	NOP	128	03	03	168	71	71	208	09	9
009	85	+	049	01	01	089	69	DP	129	29	CP	169	76	LBL	209	69	DP
010	42	STO	050	74	SM*	090	28	28	130	22	INV	170	13	C	210	17	17
011	03	03	051	06	06	091	97	DSZ	131	74	SM*	171	42	STO	211	47	CMS
012	42	STO	052	69	DP	092	02	02	132	06	06	172	02	02	212	32	X:T
013	05	05	053	36	36	093	00	00	133	69	DP	173	53	(	213	42	STO
014	04	4	054	01	1	094	97	97	134	36	36	174	43	RCL	214	89	89
015	00	0	055	22	INV	095	00	0	135	32	X:T	175	01	01	215	99	PRT
016	54	)	056	74	SM*	096	81	RST	136	74	SM*	176	85	+	216	92	RTN
017	42	STO	057	06	06	097	43	RCL	137	06	06	177	01	1	217	42	STO
018	04	04	058	69	DP	098	07	07	138	53	(	178	02	2	218	02	02
019	42	STO	059	26	26	099	42	STO	139	73	RC*	179	93	.	219	01	1
020	06	06	060	69	DP	100	03	03	140	06	06	180	00	0	220	00	0
021	73	RC*	061	25	25	101	01	1	141	55	÷	181	02	2	221	42	STO
022	03	03	062	69	DP	102	00	0	142	43	RCL	182	54	)	222	00	00
023	69	DP	063	26	26	103	49	PRD	143	01	01	183	61	GTO	223	22	INV
024	23	23	064	97	DSZ	104	09	09	144	54	)	184	02	02	224	28	LOG
025	32	X:T	065	07	07	105	64	PD*	145	53	(	185	39	39	225	42	STO
026	73	RC*	066	00	00	106	05	05	146	59	INT	186	00	0	226	01	01
027	04	04	067	38	38	107	69	DP	147	65	×	187	00	0	227	01	1
028	69	DP	068	69	DP	108	25	25	148	32	X:T	188	00	0	228	42	STO
029	24	24	069	29	29	109	97	DSZ	149	43	RCL	189	00	0	229	08	08
030	67	EQ	070	02	2	110	03	03	150	01	01	190	00	0	230	42	STO
031	00	00	071	44	SUM	111	01	01	151	54	)	191	00	0	231	49	49
032	21	21	072	49	49	112	05	05	152	97	DSZ	192	00	0	232	69	DP
033	22	INV	073	61	GTO	113	33	X^2	153	03	03	193	00	0	233	05	05
034	77	GE	074	00	00	114	64	PD*	154	01	01	194	00	0	234	60	DEG
035	00	00	075	01	01	115	06	06	155	30	30	195	00	0	235	04	4
036	76	76	076	97	DSZ	116	69	DP	156	05	5	196	00	0	236	05	5
037	29	CP	077	00	00	117	26	26	157	00	0	197	00	0	237	30	TAN
038	73	RC*	078	00	00	118	97	DSZ	158	42	STO	198	00	0	238	33	X^2
039	05	05	079	97	97	119	07	07	159	06	06	199	00	0	239	86	STF

Editor's Note: For operation without a printer the user will find that the replacement of the Nop with a R/S as defined in step 9 of the user instructions yields inconvenient operation. However, an INV-List after the program stops will show that the desired solution is not stored in memory. Examination of the listing will reveal that the solution multiplied by two appears in memory with the first group in data register (50 - N) where N is the number of groups, and the last group is in data register 49. The short program at the right uses the value N from data register 07, and the constant 1e14 from data register 01 to divide the stored solution by 2 and reprint the correct result. If the Print at program step 033 is changed to an R/S then the program will stop with the first group of ten digits of the desired solution in the display. Pressing R/S will retrieve each remaining group in sequence. Flashing nines indicate the end.

000	76	LBL	023	09	09
001	14	D	024	65	×
002	05	5	025	43	RCL
003	00	0	026	01	01
004	75	-	027	75	-
005	43	RCL	028	22	INV
006	07	07	029	59	INT
007	42	STO	030	42	STO
008	00	00	031	09	09
009	95	=	032	95	=
010	42	STO	033	99	PRT
011	02	02	034	69	DP
012	00	0	035	22	22
013	42	STO	036	97	DSZ
014	09	09	037	00	00
015	76	LBL	038	15	E
016	15	E	039	76	LBL
017	73	RC*	040	16	A'
018	02	02	041	25	CLR
019	55	÷	042	35	1/X
020	02	2	043	91	R/S
021	85	+	044	61	GTO
022	43	RCL	045	16	A'

NOTES ON SORTING AND FAST MODE - P. Hanson. The November 11, 1981 issue of EDN included an article "Adapt Computer-sort Algorithms to Programmable Calculators" by Gary Morella and Jerry Puckett of TI (pages 265 through 270). The article concludes that you can adapt some computer algorithms to work more efficiently than others on programmable calculators. The article presents TI-59 routines for a Shellsort and a Quicksort routine, and presents representative run times. If you try the routines on your calculator you will find that you cannot even approach the stated execution times. Let's examine the Shellsort routine first. The program listed in EDN is:

000	76	LBL	022	82	HIR	044	04	4	066	32	X:T	088	82	HIR	110	61	GTO
001	15	E	023	05	05	045	22	INV	067	82	HIR	089	54	54	111	00	00
002	00	0	024	55	÷	046	77	GE	068	17	17	090	01	1	112	60	60
003	42	STO	025	02	2	047	00	00	069	44	SUM	091	32	X:T	113	00	0
004	00	00	026	54	)	048	53	53	070	00	00	092	82	HIR	114	92	RTN
005	92	RTN	027	59	INT	049	01	1	071	73	RC*	093	14	14	115	76	LBL
006	76	LBL	028	61	GTO	050	67	EQ	072	00	00	094	77	GE	116	13	C
007	11	A	029	00	00	051	01	01	073	77	GE	095	00	00	117	01	1
008	69	OP	030	53	53	052	13	13	074	00	00	096	62	62	118	22	INV
009	20	20	031	82	HIR	053	82	HIR	075	97	97	097	01	1	119	90	LST
010	72	ST*	032	18	18	054	07	07	076	32	X:T	098	82	HIR	120	42	STO
011	00	00	033	82	HIR	055	82	HIR	077	72	ST*	099	36	36	121	00	00
012	92	RTN	034	05	05	056	55	55	078	00	00	100	82	HIR	122	76	LBL
013	61	GTO	035	82	HIR	057	01	1	079	82	HIR	101	16	16	123	14	D
014	11	A	036	17	17	058	82	HIR	080	14	14	102	32	X:T	124	73	RC*
015	76	LBL	037	32	X:T	059	06	06	081	42	STO	103	82	HIR	125	00	00
016	12	B	038	01	1	060	82	HIR	082	00	00	104	15	15	126	69	OP
017	43	RCL	039	03	3	061	04	04	083	32	X:T	105	22	INV	127	20	20
018	00	00	040	22	INV	062	42	STO	084	72	ST*	106	77	GE	128	92	RTN
019	53	(	041	77	GE	063	00	00	085	00	00	107	00	00	129	61	GTO
020	82	HIR	042	00	00	064	73	RC*	086	82	HIR	108	31	31	130	01	01
021	08	08	043	53	53	065	00	00	087	17	17	109	32	X:T	131	24	24

HIR commands are used to preserve the maximum number of data registers for the list to be sorted. A two step HIR 18 sequence such as that at program steps 031-032 can be synthesized by pressing GTO-8-2-BST-BST-2nd-Del-SST-2nd-C.

The article notes that the Shellsort routine is included in the TI-59 Math/Utilities library. If you download that routine you will find a listing exactly like that above. The user instructions are:

1. Initialize by pressing E. See a zero in the display.

2. Enter the first number to be sorted and press A. The input value is returned to the display.

3. Repeat step 2 as required. For the second and subsequent entries you may press either A or R/S.

4. To sort press B. At the completion of sorting the calculator stops with a zero in the display.

5. To list the sorted numbers press C. The sorted numbers are printed together with sequence numbers.

6. In step 5 the calculator will stop with the first sorted value in the display. To display additional sorted values in sequence press R/S or D.

Notes on Sorting and Fast Mode (cont)

If you run the program from the Math/Utilities module then you achieve the run times reported in the article. The same speed could be obtained by running the downloaded program in fast mode. I modified the program to add fast mode and to make it more user friendly. The changes relative to the program above were:

* Addition of fast mode using the Stflg at the end of partitioning method (steps 136-141 and 150-159). The 162 at steps 150-152 transfer the program counter to location 015 at fast mode entry. This permits the sort portion of the program to be used as is.

* The initialization routine (Lbl E) was moved from the beginning of the program to steps 142-149. The routine was changed to partition to 10-Op-17 as required by fast mode. In the initialization routine for the original program the 0-STO-00 sequence only clears R00 and any residual data in the other registers remains. A Cms is used in the revised program to clear all the data registers.

* The data input routine (Lbl A) was revised to print each element as it was entered and to return the number of elements entered to the display. To suppress the printing change program step 006 from Print to Nop.

* In the original program the routine for reading out the sorted values to the display (Lbl C and D) lists all the data registers in sequence from R01 to the end of the partitioning even though the list to be sorted was shorter. If you did not count the list to be sorted, or did not read out the length from R00 before sorting, then you did not know how many of the displayed values were valid. A new output routine (Lbl C) was written which displays a flashing "1. 12" to indicate that the last sorted value has been displayed (program steps 118 through 135). The number of elements which was sorted was needed for this routine. The value in R00 at the end of the sorting is not always correct. Steps 113-116 were inserted to store the correct value in R00 at the end of sorting. This also provides a display of the number of elements sorted. The flashing "1. 12" indication of the end of the list was obtained by letting the program run through the fast mode initialization sequence when the last element had been displayed.

* The 1-INV-List sequence was used in mode C of the original program to print out the sorted values together with sequence numbers. This feature was deleted from the revised program to provide the largest possible sorting capability. The feature is available from the keyboard by pressing 1-INV-2nd-List. The printing of values beyond the end of the sorted list remains. The Cms during initialization helps resolve the uncertainty by placing zeroes in all the unused data registers.

Notes on Sorting and Fast Mode (cont)

The revised program listing is:

```
000  76 LBL    027  59 INT    054  07  07    081  42 STO    108  31  31    135  91 R/S
001  11  A      028  61 GTO    055  82 HIR    082  00  00    109  32 X:T    136  76 LBL
002  69 DP     029  00  00    056  55  55    083  32 X:T    110  61 GTO    137  17 B'
003  20  20     030  53  53    057  01  1     084  72 ST*    111  00  00    138  71 SBR
004  72 ST*    031  82 HIR    058  82 HIR    085  00  00    112  60  60    139  01  01
005  00  00     032  18  18    059  06  06    086  82 HIR    113  82 HIR    140  50  50
006  99 PRT    033  82 HIR    060  82 HIR    087  17  17    114  16  16    141  91 R/S
007  43 RCL    034  05  05    061  04  04    088  82 HIR    115  42 STO    142  76 LBL
008  00  00     035  82 HIR    062  42 STO    089  54  54    116  00  00    143  15  E
009  91 R/S    036  17  17    063  00  00    090  01  1     117  92 RTN    144  01  1
010  61 GTO    037  32 X:T    064  73 RC*    091  32 X:T    118  76 LBL    145  00  0
011  11  A      038  01  1     065  00  00    092  82 HIR    119  13  C      146  69 DP
012  68 NOP    039  03  3     066  32 X:T    093  14  14    120  00  0      147  17  17
013  76 LBL    040  22 INV    067  82 HIR    094  77  GE    121  48 EXC    148  47 CMS
014  12  B      041  77  GE    068  17  17    095  00  00    122  00  00    149  91 R/S
015  98 ADV    042  00  00    069  44 SUM    096  62  62    123  32 X:T    150  01  1
016  25 CLR    043  53  53    070  00  00    097  01  1     124  69 DP     151  06  6
017  43 RCL    044  04  4     071  73 RC*    098  82 HIR    125  20  20     152  02  2
018  00  00     045  22 INV    072  00  00    099  36  36    126  73 RC*    153  85  +
019  53  (      046  77  GE    073  77  GE    100  82 HIR    127  00  00    154  01  1
020  82 HIR    047  00  00    074  00  00    101  16  16    128  99 PRT    155  52 EE
021  08  08     048  53  53    075  97  97    102  32 X:T    129  43 RCL    156  01  1
022  82 HIR    049  01  1     076  32 X:T    103  82 HIR    130  00  00    157  02  2
023  05  05     050  67  EQ    077  72 ST*    104  15  15    131  22 INV    158  95  =
024  55  ÷      051  01  01    078  00  00    105  22 INV    132  67  EQ    159  86 STF
025  02  2      052  13  13    079  82 HIR    106  77  GE    133  01  01
026  54  )      053  82 HIR    080  14  14    107  00  00    134  24  24
```

The user instructions for the revised Shellsort program are:

1. Initialize by pressing E. See "159.99" in the display.

2. Enter the first number to be sorted and press A. A "1" is returned to the display.

3. Repeat step 2 as required. For the second and subsequent entries you may press either A or R/S. The number of elements entered is returned to the display.

4.a. To sort in normal mode press B. At the completion of sorting the calculator stops with the number of elements sorted in the display.

4.b. To sort in fast mode press 2nd-B' and see a flashing "1. 12" in the display. Do not clear the flashing, but press 7 and then EE to enter fast mode. Fast mode sorting will provide a reduction in sorting time by about a factor of two. As with normal mode sorting the number of elements sorted will be displayed when sorting is complete.

5.a. To print the sorted numbers press C. The sorted numbers are printed and the calculator stops with "1. 12" in the display.

5.b. To display the sorted numbers when a printer is not available change the Print at location 128 to a R/S. Press C and display the first sorted element. Press R/S as many times as needed to display additional elements in sequence. A flashing "1. 12" indicates that all the elements have been displayed.

6. To print the sorted list with accompanying sequence numbers press 1-INV-2nd-List from the keyboard.

Notes on Sorting and Fast Mode (cont)

The article in EDN also discussed a Quicksort routine in detail. As with the Shellsort routine the run times cannot be approached without adding a fast mode capability. The article notes that data registers 00 through 029 are needed for the housekeeping required by the Quicksort method with the result that only 70 elements can be sorted. However, the published program includes 209 program steps which would require a partitiioning of 9-Op-17. This provides a total of 90 data registers. The use of 30 data registers for housekeeping leaves only 60 registers for the input list. How did the TI authors arrive at a potential for sorting 70 elements? Either the Quicksort program was on a program module or it was run on the device called an "emulator" which provides the equivalent of operating from a program module. That would also explain the reported run speed. Does anyone know of this routine being used in a program module? The limit of 60 elements to be sorted versus 99 for Shellsort combined with marginal improvements in run time relative to Shellsort and the need for another routine to enter the unsorted list or display the sorted list makes the Quicksort program an unattractive alternative. I have provided a listing with a fast mode entry routine for those who might wish to experiment. Note that program step 224 is 2nd-CLR (code 20) not CLR (code 25).

000	76	LBL	040	63	EX*	080	01	1	120	42	42	160	01	1	200	42	STO
001	25	CLR	041	23	23	081	54	)	121	43	RCL	161	44	SUM	201	22	22
002	53	(	042	72	ST*	082	77	GE	122	21	21	162	27	27	202	73	RC*
003	42	STO	043	24	24	083	01	01	123	72	ST*	163	44	SUM	203	27	27
004	28	28	044	01	1	084	70	70	124	27	27	164	29	29	204	42	STO
005	85	+	045	94	+/-	085	53	(	125	53	(	165	43	RCL	205	21	21
006	03	3	046	49	PRD	086	43	RCL	126	53	(	166	22	22	206	61	GTO
007	00	0	047	25	25	087	23	23	127	43	RCL	167	61	GTO	207	00	00
008	42	STO	048	43	RCL	088	75	-	128	23	23	168	01	01	208	23	23
009	21	21	049	25	25	089	43	RCL	129	85	+	169	77	77	209	92	RTN
010	75	-	050	29	CP	090	21	21	130	01	1	170	53	(			
011	01	1	051	77	GE	091	54	)	131	54	)	171	24	CE			
012	42	STO	052	00	00	092	32	X:T	132	42	STO	172	75	-			
013	27	27	053	61	61	093	01	1	133	21	21	173	02	2			
014	54	)	054	01	1	094	22	INV	134	75	-	174	54	)			
015	42	STO	055	94	+/-	095	77	GE	135	02	2	175	42	STO			
016	22	22	056	44	SUM	096	01	01	136	54	)	176	22	22			
017	01	1	057	24	24	097	11	11	137	72	ST*	177	32	X:T			
018	01	1	058	61	GTO	098	53	(	138	29	29	178	43	RCL	218	25	CLR
019	42	STO	059	00	00	099	43	RCL	139	61	GTO	179	21	21	219	91	R/S
020	29	29	060	64	64	100	23	23	140	01	01	180	22	INV	220	61	GTO
021	43	RCL	061	01	1	101	85	+	141	60	60	181	77	GE	221	00	00
022	21	21	062	44	SUM	102	01	1	142	43	RCL	182	00	00	222	02	02
023	42	STO	063	23	23	103	54	)	143	22	22	183	23	23	223	76	LBL
024	23	23	064	43	RCL	104	42	STO	144	72	ST*	184	01	1	224	20	CLR
025	01	1	065	24	24	105	21	21	145	29	29	185	94	+/-	225	71	SBR
026	42	STO	066	32	X:T	106	43	RCL	146	53	(	186	44	SUM	226	02	02
027	25	25	067	43	RCL	107	22	22	147	53	(	187	27	27	227	29	29
028	43	RCL	068	23	23	108	61	GTO	148	43	RCL	188	44	SUM	228	91	R/S
029	22	22	069	22	INV	109	01	01	149	23	23	189	29	29	229	02	2
030	42	STO	070	77	GE	110	77	77	150	85	+	190	01	1	230	07	7
031	24	24	071	00	00	111	53	(	151	01	1	191	32	X:T	231	01	1
032	73	RC*	072	32	32	112	43	RCL	152	54	)	192	43	RCL	232	02	2
033	23	23	073	32	X:T	113	22	22	153	72	ST*	193	27	27	233	85	+
034	32	X:T	074	43	RCL	114	75	-	154	27	27	194	22	INV	234	01	1
035	73	RC*	075	22	22	115	43	RCL	155	75	-	195	77	GE	235	52	EE
036	24	24	076	32	X:T	116	24	24	156	02	2	196	02	02	236	01	1
037	77	GE	077	53	(	117	54	)	157	54	)	197	09	09	237	02	2
038	00	00	078	24	CE	118	77	GE	158	42	STO	198	73	RC*	238	95	=
039	48	48	079	85	+	119	01	01	159	22	22	199	29	29	239	86	STF

For operation in normal mode enter the unsorted list starting at data register R30. Enter the number of elements in the list and press SBR CLR. Print out the sorted list with the keyboard sequence 30-INV-2nd-List.

Notes on Sorting and Fast Mode (cont)

For operation in fast mode enter the unsorted list starting at data register R30. Press GTO-2nd-CLR-R/S and see a flashing "1. 12" in the display. Press 7 and then EE and see a zero in the display. Enter the number of elements to be sorted and press R/S. Read out the sorted list with the 30-INV-2nd-List sequence.

SORTING ON THE TI-66/PC-200 - The HIR commands are not available on the TI-66. The Shellsort routine was modified for use on the TI-66 by using data registers R39 through R43 to replace the hierarchy registers H04 through H08. The result is that a TI-66 Shellsort program can accomodate only 38 elements. The program listing for the TI-66 is:

```
      ST
000   LBL     027   00      054   SUM     081   00      108   29      135   DP
001   A       028   51      055   40      082   X≥T     109   X≥T     136   20
002   DP      029   RCL     056   1       083   STO*    110   GTO     137   RCL*
003   20      030   43      057   STO     084   00      111   00      138   00
004   STO*    031   STO     058   41      085   RCL     112   59      139   R/S
005   00      032   40      059   STO     086   42      113   RCL     140   RCL
006   PRINT   033   RCL     060   39      087   INV     114   43      141   00
007   RCL     034   42      061   STO     088   SUM     115   STO     142   INV
008   00      035   X≥T     062   00      089   39      116   00      143   X=T
009   R/S     036   1       063   RCL*    090   1       117   R/S     144   01
010   GTO     037   3       064   00      091   X≥T     118   LBL     145   35
011   A       038   INV     065   X≥T     092   RCL     119   C       146   ADV
012   LBL     039   X≥T     066   RCL     093   39      120   DP      147   0
013   B       040   00      067   42      094   X≥T     121   20      148   1/X
014   ADV     041   51      068   SUM     095   00      122   PART    149   GTO
015   RCL     042   4       069   00      096   61      123   IND     150   01
016   00      043   INV     070   RCL*    097   1       124   00      151   39
017   (       044   X≥T     071   00      098   SUM     125   DP      152   LBL
018   STO     045   00      072   X≥T     099   41      126   30      153   E
019   43      046   51      073   00      100   RCL     127   1       154   CLR
020   STO     047   1       074   97      101   41      128   R/S     155   PART
021   40      048   X=T     075   X≥T     102   X≥T     129   LBL     156   44
022   ÷       049   01      076   STO*    103   RCL     130   D       157   CMS
023   2       050   13      077   00      104   40      131   0       158   R/S
024   )       051   STO     078   RCL     105   INV     132   EXC     159   0
025   INTG    052   42      079   39      106   X≥T     133   00
026   GTO     053   INV     080   STO     107   00      134   X≥T
```

The Part-2nd-Ind-XX command on the TI-66 provides partitioning under program control data register by data register rather than in groups of ten as with the TI-58/59. That feature was used at program steps 122-124 to control operation of the INV-2nd-List function so that only the data registers which contain sorted elements will be printed with accompanying sequence numbers. Unfortunately the TI-66/PC-200 will not perform the INV-List command under program control so that function must be accomplished from the keyboard after the proper partitioning has been selected by the program. The user instructions are:

1. Initialize by pressing E. See "159.43" in the display.

2. Enter the first number to be sorted and press A. A "1" is returned to the display.

3. Repeat step 2 as required. For the second and subsequent entries you may press A or R/S. The number of elements entered is returned to the display.

Sorting on the TI-66/PC-200 (cont)

4. Press B to sort. The calculator stops with the number of elements sorted in the display.

5. To select the partitioning for printing the sorted list with sequence numbers press C. The calculator stops with a "1" in the display. Press INV-2nd-List to print the list.

6. To bring the sorted numbers to the display press D. The calculator stops with the first value in the display. Press R/S to bring the subsequent values to the display. An "Error" in the display indicates that all of the sorted values have been displayed.

SORTING ON THE CC-40 - V8N6P21 discussed the Shell sort subprogram in the Statistics cartridge for the CC-40. A sample program provided prompting for data entry, callup of the subprogram, and display of the results. The program would sort 60 random numbers in 31 seconds. The TI-59 using the Shell sort from the Math/Utilities module takes 295 seconds.

There are other sorting routines than Shell sort. An examination of some of them was triggered by Albert Nijenhuis' article "A Confusion of Sorts" in the June 1985 issue of Creative Computing. For comparison I wrote a program which has five options for sorting:

1. An old "bubblesort" routine which I wrote many years ago while learning BASIC programming.

2. An "insertion sort" routine adapted from an earlier Nijenhuis article in the August 1980 issue of Creative Computing.

3. The Shell sort which is included with the Statistics cartridge for the CC-40.

4. A "heapsort" routine adapted from the program in a Nijenhuis' article in the September 1980 issue of Creative Computing.

5. An "address sort" routine which doesn't use comparison techniques, but rather uses array processing to order a set of input integers.

As expected the bubblesort program was by far the slowest--three times slower than the insertion sort and five times slower than the heapsort. Bubblesort and insertion sort run about a factor of two slower for the "worst case" set of an inverted list. Shell sort actually runs substantially faster with the inverted list than with random numbers. Heapsort execution time is essentially the same for random numbers or an inverted list. The address sort is by far the fastest, but the user must live with some limitations:

* Only integers can be sorted.

* Two one dimensional arrays are needed. Thus for a given memory size the address sort can only accept about half as many elements as the remaining methods.

Sorting on the CC-40 - (cont)

Program Comments:

Line 110 - The call to the UP subprogram with the arguments listed will display the message in quotations for about three seconds and then display the prompt "Use Printer?". A response of "N" will set the variable PN to zero, and proceed to the next line of the program. A response of "Y" will set the variable PN to one and display the prompt "Enter Filename:". If you enter a "10" for the HX-1000 the subprogram will open file #1 for output, set the printer to the 32 character per line option, print the message in the first part of the argument of the UP subroutine, and proceed to the next program line.

Lines 120 through 170 provide selection from three options for data entry.

Lines 200 through 230 provide prompts and controls for data entry from the keyboard. The use of the IF NOT NUMERIC function in line 222 provides an easy way to respond if the input string is not a valid number.

Lines 235 through 280 provide prompts and controls for data input from DATA statements. The data statements must have been previously entered. The ON WARNING and CALL ERROR statements provide automatic sensing of the end of data. As written the program will read all the DATA statements in the program. The idea for the use of DATA statements for data entry was obtained from Codeworks (see page 4 of this issue).

Lines 285 either prints or displays the number of input data points depending on the value of PN set in line 110.

Lines 290 through 298 provide an option of printing (or displaying without a printer) the input data points.

Lines 300 through 340 provide options for the method of sorting.

```
100 DIM X(300),Y(300)
110 CALL UP("Sorting Demonstration",
PN)
120 A$="Data Options: "
130 PRINT A$&"1 = Keyboard":PAUSE 1
140 PRINT A$&"2 = Data File":PAUSE 1
150 PRINT A$&"3 = Test Numbers":PAUS
E 1
160 INPUT "Which Input Option (1-3)?
 ";OP
170 ON OP GOTO 200,235,1000
200 REM Input from Keyboard
205 PRINT "Press <C> to End Input":P
AUSE 2
210 N=0
215 INPUT "X("&STR$(N+1)&") = ";X$
218 IF X$="C"OR X$="c"THEN 285
222 IF NOT NUMERIC(X$)THEN PRINT "In
valid Entry: ";:GOTO 215
225 X(N+1)=VAL(X$)
230 N=N+1:GOTO 215
235 REM Input from Data File
240 ON WARNING ERROR:ON ERROR 265
245 N=0
250 READ X(N+1)
255 N=N+1
260 GOTO 250
265 ON ERROR 990
270 CALL ERR(E,T,F,F1)
275 IF E<>43 THEN 990
280 RETURN 285
285 PRINT #PN,"Number of Data Points
 = ";N:PAUSE 2
290 INPUT "Print the Input Data (Y/N
)? ";Q$
295 IF Q$="N"OR Q$="n"THEN 300
298 GOSUB 1100
300 A$="Options:  "
305 PRINT A$&"1 = Bubble Sort":PAUSE
 1
310 PRINT A$&"2 = Insertion Sort":PA
USE 1
315 PRINT A$&"3 = Shell Sort":PAUSE
1
320 PRINT A$&"4 = Heap Sort":PAUSE 1
325 PRINT A$&"5 = Address Sort":PAUS
E 1
330 INPUT "Which Sorting Option (1-5
)? ";OP
335 PRINT "Sorting"
340 ON OP GOSUB 400,500,600,700,800
345 DISPLAY BEEP
350 INPUT "Print the Sorted Data (Y/
N)? ";Q$
355 IF Q$="N"OR Q$="n"THEN 900
360 GOSUB 1100
370 GOTO 900
400 REM Bubble Sort
410 FOR I=1 TO N-1
420 IF X(I+1)>=X(I)THEN 480
430 A=X(I+1)
440 X(I+1)=X(I)
450 X(I)=A
460 I=I-2
470 IF I<0 THEN I=0
480 NEXT I
490 RETURN
500 REM Insertion Sort
510 FOR J=1 TO N-1
520 B=X(J+1)
530 FOR I=J TO 1 STEP -1
540 IF B>=X(I)THEN 580
550 X(I+1)=X(I)
560 NEXT I
570 I=0
580 X(I+1)=B
590 NEXT J
595 RETURN
```

Sorting with the CC-40 - (cont)

Lines 345 through 370 indicate that the sorting is complete with a "beep" and provide an option of printing (or displaying without a printer) the sorted data points.

Lines 400 through 490 are an elementary bubble sort subroutine that I wrote in an old programming class.

Lines 500 through 595 are an insertion sort subroutine adapted from the program on page 37 of the August 1980 issue of Creative Computing.

Lines 600 through 620 call the Shell sort subroutine of the Statistics cartridge.

Lines 700 through 796 are a heapsort subroutine adapted from the program on page 137 of the September 1980 issue of Creative Computing.

Lines 800 through 895 are a subroutine for sorting using indirect address techniques.

Lines 900 through 999 close file #1 if it was opened at line 110, and provide program ending.

Lines 1000 through 1040 provide two options for generating test numbers. The random option provides 100 random integers in the range from 1 to 100. The inverse option provides the integers from 1 to 100 in reverse order.

Lines 1100 through 1175 provide a subroutine for display or printout of the input and output data. Use of PRINT #PN statements with PN set by the UP subroutine in line 110 in line provides automatic control of whether the output is to the display or the printer.

Lines 5000 through 5030 contain the DATA statements for the 26 test integers used to test the heapsort in the September 1980 issue of Creative Computing.

```
600 REM Shell Sort with Statistics M
odule
610 CALL SORT(X(),N)
620 RETURN
700 REM Heap Sort
705 M=N
710 FOR L=INT(N/2)TO 1 STEP -1
715 B=X(L)
720 GOSUB 760
725 NEXT L
730 L=1
735 FOR M=N-1 TO 1 STEP -1
740 B=X(M+1):X(M+1)=X(1)
745 GOSUB 760
750 NEXT M
755 RETURN
760 I=L
765 J=I+I
770 IF J>M THEN 794
775 IF J=M THEN 785
780 IF X(J+1)>X(J)THEN J=J+1
785 IF B>=X(J)THEN 794
790 X(I)=X(J):I=J
792 GOTO 765
794 X(I)=B
796 RETURN
800 REM Address Sort
805 M=N:IM=0
810 FOR I=1 TO M
815 Y(X(I))=Y(X(I))+1
820 IF X(I)>IM THEN IM=X(I)
830 NEXT I
840 M=N:L=1
850 FOR I=1 TO IM
860 FOR J=1 TO Y(I)
870 IF Y(I)=0 THEN 890
875 X(L)=I
880 L=L+1
885 NEXT J
890 NEXT I
895 RETURN
900 IF PN=1 THEN CLOSE #1
990 STOP
999 END
1000 PRINT "Options for 100 Test Num
bers:":PAUSE 2
1005 PRINT "Press 1 for Random Integ
ers, or":PAUSE 2
1010 INPUT "Press 2 for an Inverted
List:";Q
1015 FOR I=1 TO 100
1020 IF Q=1 THEN X(I)=INTRND(100)
1025 IF Q=2 THEN X(I)=101-I
1030 NEXT I
1035 N=100
1040 GOTO 285
1100 REM Printing Subroutine
1105 PRINT #PN:IF PN=0 THEN 1160
1110 I=1
1115 FOR J=0 TO 4
1120 PRINT #PN,X(I+J);
1125 IF I+J=N THEN 1150
1130 NEXT J
1135 IF PN=0 THEN PAUSE
1140 PRINT #PN
1145 I=I+5:GOTO 1115
1150 IF PN=0 THEN PAUSE
1155 PRINT #PN:RETURN
1160 FOR I=1 TO N
1165 PRINT X(I):PAUSE
1170 NEXT I
1175 RETURN
5000 DATA 19,8,18,15,14,25
5010 DATA 16,10,12,13,3,26
5020 DATA 24,17,20,9,4,7
5030 DATA 11,6,5,2,1,21,23,22
```

Sorting on the CC-40 - (cont)

I did not include the commentary in the program through the use of REM statements to conserve memory in the CC-40. The program does include fairly extensive prompting to the display. A user should be able to run the program with very little reference to other instructions. Simply enter RUN in the display, press <ENTER> and follow the prompts. A sample printout which results from selection of the second input option, from DATA statements, appears at the right.

```
Number of Data Points =  26

 19  8  18  15  14
 25  16  10  12  13
 3  26  24  17  20
 9  4  7  11  6
 5  2  1  21  23
 22

 1  2  3  4  5
 6  7  8  9  10
 11  12  13  14  15
 16  17  18  19  20
 21  22  23  24  25
 26
```

The execution times in seconds for the five sorting techniques for different numbers of random integers are:

	Number of Integers				
Method	10	30	100	300	1000
Bubble	2	24	252	2435	26685
Insertion	1	9	86	735	
Shell	3	13	73	390	
Heap	2	11	51	190	780
Address	1	3	10	30	

The table shows that execution times for the bubble sort grow as the square of N, the execution times for the heap sort grow as N LogN, and the execution times for the address sort grow linearly with N. Clearly, one should use some version of the addres sort if execution time is important and the problem permits.

For those times the random number input option (Lines 1000 through 1040) was modified such that the range of the positive random integers was the same as the number of data items to be sorted. That gives an advantage to the address sort. To check on how much an advantage that was I reran the 100 number case, but with the range of the integers raised to 300. The execution times for the five sorting methods were then 269, 89, 73, 55 and 15 seconds respectively.

The execution times for an input of 100 integers in reverse order were 490, 160, 39, 50, and 10 seconds respectively.

A NEW PORTABLE FROM TI? - CHHU is a new HP users group headed by Richard Nelson, the long-time editor of the PPC Calculator Journal. PPC is now apparently under other leadership. CHHU has a telephone bulletin board which will provide a recorded message (call 714-754-4557). Charlie Williamson reports that a recent bulletin stated "... rumor has it that TI will reenter the handheld marlet with new products, ala the Sharp 5100 and the TI-59. Will the TI -88 be reborn? ..."

SCRAMBLING IN BASIC - Larry Leeds writes "Your new scramble program (V10N2P23) in fast mode is an excellent example of absolutely ingenious programming which was necessary to circumvent the limitations of the TI-59. It might be of interest to the members to note that when the same algorithm is used in a program in BASIC, the programming is very elementary."

Larry's program for the Radio Shack Model 100 scrambles the integers from 1 to 100 in just six seconds. The conversion for the CC-40 will scramble the integers from 1 to 100 in twelve seconds. A sample printout appears at the right. The CC-40 program appears below.

```
Scramble Program

Seed = 234

 15  76  50  20  34
 17   1  10  78  72
 36  82  99  22  91
 71  93  88  84  24
 73  42   7   0  23
 11  94  83  14  12
 33  62  63  28   2
 86  39  59  47  69
 37  30  97   8  10
 13  46  58  43  56
 35   5  45  55  31
 80  25  32  49  67
 54  90  61  38  27
 95  52  87  41  81
  4  92  57  70  64
 51   9  85  60  89
 68  98  29  74  20
 79  53  48  66 100
 44  21  77  40  19
 65  75  18  96   3
```

Program Comments:

Line 20 generates the value pi/2 which is used in the scrambler.

Lines 30 uses the UP routine from the cartridge to set up the control of printing or displaying the results. In answer to the prompt "Enter Filename:" respond with a "10". See page 14 for a more complete description of the UP subroutine.

Lines 50 enters the integers 1 through 100 in ascending order into the array.

Lines 60 through 99 scramble the integers. Larry uses his own random number generator rather than the one in the machine.

Lines 100 through 160 print or display the scrambled integers in groups of five.

```
10 DIM A(100)
20 Q=ATN(1.E+14)
30 CALL UP("Scramble Program",PN)
40 INPUT "Seed ? ";S
45 PRINT #PN,"Seed = ";S:PRINT #PN
50 FOR H=1 TO 100:A(H)=H:NEXT H
60 FOR H=100 TO 1 STEP -1
70 X=S*Q
80 S=100*(X-INT(X)):Z=INT(S)
85 IF Z=0 THEN 70
90 T=A(A(Z)):A(A(Z))=A(H):A(H)=T
95 NEXT H
99 DISPLAY BEEP
100 FOR I=0 TO 95 STEP 5
110 FOR J=1 TO 5
120 PRINT #PN,A(I+J);
130 NEXT J
140 IF PN=0 THEN PAUSE
150 PRINT #PN
160 NEXT I
170 PRINT #PN
180 STOP
190 END
```

UP SUBROUTINE AVAILABILITY - In both the sorting and scrambling programs in this issue I used the UP routine from the cartridge to establish printer/display control. Examination of the manuals shows, and tests verify, that the UP routine is in the three cartridges I own, Mathematics, Statistics and Finance. Is the UP routine available in all the cartridges?

RTN: AN UNEXPECTED RESULT - Charlie Williamson writes: "A running program named EE encounters SBR X≥T (for example): it saves a return address, and branches to execute X≥T. Within X≥T, our program hits R/S and waits for 'operator action' as they say. One then pushes either A or B, both of which contain a RTN. The program runs and encounters that RTN, whereupon it doesn't branch back to EE, but stops.

What's wrong? Obviously an address was saved, and an authentic RTN was present. SBR A (rather than A alone) doesn't help. I don't understand what the 59 does with an A-push in this situation.

One functional solution is to push GTO A R/S. The program then does branch back to EE. This subject is the basis of a George Vogel programming puzzle (V5N7P5). Presumedly his answer is similar.

NEWCOMER'S CORNER: THE ORIGINS OF FAST MODE - My earliest exposure to Martin Neef's technique for obtaining fast mode came in a letter from Maurice Swinnen dated May 11, 1980. Maurice and I exchanged several letters on the subject. The following discussion of the technique appeared in V5N6P4 of TI PPC Notes in a discussion entitled "QUIRKS":

> "Then Martin Neef of the ZEPRA club (Germany) found another exciting use for the sequence: a FAST mode. Programs placed in user memory execute at about twice normal speed! Starting at step 005: Pgm 02 SBR 239 9; execute this sequence, but stay in LRN. Then BST to get to step 000. Now place in user memory:
>
> Cms CP 200 x t Op 20 RCL 00 INV EQ 006 EE INV EE R/S GTO 000
>
> Still in LRN, BST a couple of times to get to the R/S step. Then go out of LRN. Now press R/S. The Dsz-ing of register 00 should take about 35 seconds. Now cancel the special state by pressing RST. Press R/S again and this time execution will take 70 seconds! It seems than in order to make the program halt, the R/S step is not sufficient. You always need EE INV EE in from of it. In the FAST mode you can't use user-defined keys, nor subroutine calls. Neither can uou call subroutines or programs from the module. But you can start a subroutine from the keyboard by pressing SBR xxx.
>
> Palmer Hanson thinks Fast mode puts the calculator in the same mode as when it is accessing RAM memory. Take, for example, the INV SUM+ function. If you program 24 iterations of the function it will take 22 seconds to complete. If you use the RAM program which mechanizes that same function (locations 213 to 248 of the downloaded RAM) and iterate that one 24 times it will take 38 seconds. A second clue that it is not the module but RAM is that RST will NOT interrupt FAST mode program execution."

The deficiency of the technique as defined at that time was the need to key in the program after setting fast mode. In mid-July 1980 I discovered that the desired program could be entered under program control after the fast mode initialization in a sort of "load-and-go" technique. I used that idea to generate a fast mode Print Code Table Printer program in response to the challenge on the front page of the May/June 1980 issue of PPX Exchange. My submission was not mentioned in the discussion of the responses to the challenge which appeared in the September/October 1980 issue of PPX Exchange. First, my program was relatively slow due to the printing of dividers and headings at the top and side. Second, TI was not ready to publish fast mode techniques at that time. A description of how to use fast mode did not appear in PPX Exchange until my article "TI-59 Fast MOde" in the July/August 1981 issue, nearly one year later.

Program Listing:

100020003.	90
4000500.	91
600070008.	92
10002.	93
3000400.	94
500060007.	95
1.0010001	96
1.000010001	97
2020202020.	98
4131243736.	99

000	00	0	030	69	OP	060	69	OP	090	08	8	120	69	OP
001	00	0	031	03	03	061	03	03	091	42	STO	121	05	05
002	00	0	032	43	RCL	062	43	RCL	092	00	00	122	43	RCL
003	00	0	033	90	90	063	93	93	093	09	9	123	96	96
004	00	0	034	69	OP	064	69	OP	094	75	-	124	82	HIR
005	36	PGM	035	02	02	065	02	02	095	43	RCL	125	36	36
006	02	02	036	69	OP	066	03	3	096	00	00	126	82	HIR
007	71	SBR	037	05	05	067	06	6	097	95	=	127	38	38
008	02	02	038	43	RCL	068	52	EE	098	52	EE	128	43	RCL
009	39	39	039	98	98	069	08	8	099	04	4	129	97	97
010	09	9	040	69	OP	070	42	STO	100	85	+	130	82	HIR
011	00	0	041	02	02	071	03	03	101	06	6	131	37	37
012	69	OP	042	69	OP	072	03	3	102	02	2	132	97	DSZ
013	00	00	043	03	03	073	01	1	103	00	0	133	00	00
014	01	1	044	69	OP	074	52	EE	104	00	0	134	01	01
015	00	0	045	04	04	075	08	8	105	95	=	135	16	16
016	69	OP	046	04	4	076	42	STO	106	74	SM*	136	06	6
017	17	17	047	07	7	077	04	04	107	00	00	137	69	OP
018	43	RCL	048	02	2	078	01	1	108	97	DSZ	138	17	17
019	99	99	049	00	0	079	07	7	109	00	00	139	25	CLR
020	69	OP	050	69	OP	080	52	EE	110	00	00	140	91	R/S
021	03	03	051	01	01	081	08	8	111	93	93	141	98	ADV
022	69	OP	052	69	OP	082	42	STO	112	25	CLR	142	47	CMS
023	05	05	053	05	05	083	05	05	113	08	8	143	61	GTO
024	43	RCL	054	43	RCL	084	03	3	114	42	STO	144	00	00
025	92	92	055	95	95	085	07	7	115	00	00	145	12	12
026	69	OP	056	69	OP	086	52	EE	116	73	RC*			
027	04	04	057	04	04	087	08	8	117	00	00			
028	43	RCL	058	43	RCL	088	42	STO	118	69	OP			
029	91	91	059	94	94	089	06	06	119	01	01			

The Origins of Fast Mode - (cont)

To use the program record the program and the constants on a magnetic card. Note that the constant in R96 contains digits which do not appear in the printout from an INV-List. The contents of R96 can be synthesized with the keyboard sequence 1000100010 x 1 EE 12 +/- + 1 = . Enter the program into the calculator from the card. Press RST and R/S and see a "0" in the display. Re-enter the same side of the card and the program will run in fast mode and print the table. About thirteen seconds will be required to deliver the table at the right. The use of the exchange symbol for vertical lines, the dash for horizontal lines, and the plus for corners was suggested by Richard Snow in V4N2P6 of 52 Notes.

		UNITS							
		0	1	2	3	4	5	6	7
	0:		0	1	2	3	4	5	6
	1:	7	8	9	A	B	C	D	E
T	2:	-	F	G	H	I	J	K	L
E	3:	M	N	O	P	Q	R	S	T
N	4:	.	U	V	W	X	Y	Z	+
S	5:	×	÷	[	π	e	(	)	,
	6:	↑	%	;	⁄	=	'	×	x̄
	7:	²	?	÷	⁹	Π	Δ	Π	Σ

No other response to the challenge, either in PPX Exchange or in V5N9/10 of TI PPC Notes, printed the headings and separation lines. The sample printout for the second program on V5N9/10P20 did have the separation lines drawn in by hand after the fact.

From a historical standpoint the program is a landmark. It was the first program to use the "load-and-go" idea with the Neef technique for fast mode entry. I completed the fast mode calendar program (See "The Great Calendar Race, V9N2P6) two weeks later. The program has not been published previously in either PPX Exchange or TI PPC Notes, but was published in the Oct/Nov/Dec 1982 issue of TISOFT, the Belgian newsletter. It was also included in the Fast Mode compilation offered to members in 1983, but the listing failed to include the constants in R90 through R99. The "load-and-go" process for entering a program after fast mode initialization placed a premium on a good magnetic card reader. One bad read and the user had to start over from the beginning. In late 1981 Patrick Acosta's article on fast mode entry using the Stflg h12 or Stflg at the end of partition methods provided welcome relief.

A USEFUL INPUT DATA EFFECT WITH SOME COMPUTERS - Larry Leeds. V9N5P7 reported that if numbers which exceeded the length of the mantissa of the storage word were entered into CC-40, either from the keyboard or from a program, then the computer would examine the extra digits to see if the lowest digit of the mantissa to be stored should be rounded. Appropriate adjustment of the exponent would also occur. Similar effects were found with other computers.

Larry reports that his Radio Shack Model 100 will also accept blanks in the sequence of figures in the display, or in program statements. This permits the use of the blanks to make the entry of numbers with many digits more readable. For example, the statement:

```
X = 111 222 333 + 5
```

will yield x = 111222338, and the statement

```
y = 0.000 000 000 000 123 456
```

will yield x = 1.23456E-13. Tests show that the Radio Shack Color Computer and the Commocore 128 respond similarly. An attempt to introduce blanks with the CC-40 leads to the "Illegal Syntax" diagnostic message.

MORE PPX PROGRAM AVAILABILITY - V9N5P18 set up an informal program exchange to provide access to programs which were formerly available from PPX Exchange. Over six hundred programs were listed. Twenty-eight more programs were listed in V9N6P3, and 104 additional programs were listed in V10N1P19. Five more programs were listed in V10N2P3. 205 additional programs have now been made available. The following list shows the newly available programs and includes the five programs listed in V10N2P3:

H 018003 - Monthly/Fiscal Year Linear Trend Projection
H 018004 - Organizational Tech. Level Index - Average Grade
H 088014 - Maximum Allowable Monthly Rent
1AJ 148007 - PPC Checkbook
1DH 208008 - Polynomial Regression to 7th Power
K 298059 - Generate Joseph's Permutation
K 298075 - Expanded Combinations, Permutations and Factorials
1L 368008 - Chinese Remainder Theorem
KL 368009 - HCF and LCM Calculations
L 368015 - Continued Fractions
49K 368027 - Extended Range Roman Numeral Math II
K 368041 - Integer as Sum of Four Squares
L 398045 - Precise Multiplication
K 398092 - Magic Squares
L 398107 - Monte Carlo Primality Test
H 398110 - Pythagorean Analysis with Integer Solutions
L 398118 - Constant Arithmetic
K 398132 - Pascal's Triangle Term Generator
K 398179 - Magic Squares, 3x3 through 6x6
K 398197 - Happy Numbers
K 398199 - Location of Primes of Specific INtervals
3L 398242 - Square Root - Double Precision
L 398250 - Natural Logarithms - Double Precision
L 398270 - e^x (multiple precision)
1M 628003 - Structural Section Properties
M 628010 - Simple Beam with two Cantilever Ends
M 628032 - Circular Concrete Column
M 628035 - Rectangular Concrete Column Analysis
M 628036 - Masonry Wall Design
M 628052 - Elastic Analysis of Hingeless Curved Structure
M 628054 - Elastic Analysis of One Hinged Curved Structure
M 628065 - Mohr's Circle
M 628066 - Elastic Properties of Nonprismatic Beams
EM 628067 - Steel Beam/Column Design
EM 628069 - Continuous Beam by 3 Moment Equation
M 628080 - Abutments: Stability Design of Bridge Abutments
M 628081 - Abutments: Structural Design of Bridge Abutments
EM 628091 - Truss Design: Open Web Joint
M 628102 - Masonry Column Design
M 628107 - Truss Design: 8 Bay Fink Truss
M 628117 - Wall: Design of Bridge Wingwall's
EM 628123 - Frame Analysis: Moment Distribution
EM 628133 - Rectangular Biaxial Column
M 628145 - Bishop Modified Slope Stability Analysis
M 628171 - Wall: Retaining Wall Design
M 628178 - Simple Beam: Slope, Def., M, V
M 628180 - Roark's Flat Plate Functions

More PPX Program Availability - (cont)

EM 628195 - Column Web Stiffeners
M 628198 - Wood Beam/Joist Design
EM 628201 - Column Base Plate Design with Moment and/or Axial Load
M 628202 - Footing: Underreams-Approximate Analysis
EM 628207 - Footing: Octagonal Foundation Design (USD)
EM 628209 - Footing: Ringwall Foundation Design (USD)
M 628210 - Rectangular & T Beam Design (USD)
EM 628216 - Footing: Square and Rect. Foundation Design (USD)
M 628217 - Footing: Wall Footing Design
EM 628218 - Footing: Battered Pile Foundation
EM 628220 - Footing: Friction Pile Design
EM 628235 - Anchored Sheetpiling: Free-earth Method
M 628236 - Cantilever Sheetpiling in Sandy Soil
M 628237 - Fixed Base Single Frame
M 628238 - Pinned Base Single Frame
M 628289 - Simple Beam with Moving Loads
6K 638025 - HP-65/HP-67 Mini-compiler
9K 788015 - Moon Phase for Any Date
9K 908017 - Calendar Print Generator
K 908042 - Add and Subtract Time
K 908045 - Large Alphanumerics
K 908073 - Cryptographic Encoder-Decoder
H 908115 - Utility Routine: Subroutine and Loop Timer
K 908125 - Message Encoder/Decoder
1H 908142 - Bucket Sort (up to 99 items)
K 908160 - Even More Alphanumeric Register Listers
1J 908172 - Character Set Printout
1K 908192 - High Speed Calendar Printer
K 908199 - Headline Printer - Vertical and Horizontal
K 908219 - Mini-banner Program
K 908226 - List 13 Digit Registers
K 908229 - Complete Date Print
1J 908232 - Typewriter
1JK 908241 - Correcting Typewriter
K 918010 - Phantom Ship
K 918014 - Football
K 918018 - Golf
K 918019 - Princeps Puzzle
19J 918020 - Skydiving
K 918026 - One Arm Bandit
K 918027 - Card Dealer -- Bingo Caller
K 918028 - Football (Solitaire)
K 918031 - Super Code Breaker
5K 918032 - Game of Gale
K 918035 - Scrambled Eggs
K 918037 - Golf
K 918038 - Horse Race
4K 918043 - The One Armed Bandit
4K 918048 - Seven Card Low Poker
K 918050 - Keno
5K 918056 - Automatic Blackjack with Options
K 918060 - Hamurabi
29K 918061 - Pinball

More PPX Program Availability - (cont)

K 918062 - Calculator Ping-pong
K 918064 - Roulette Challenge
K 918065 - Game of Kayles
K 918066 - Basketball
K 918068 - Evens Game
K 918070 - Arithmetic Game
9K 918071 - Memory Flashcard
K 918072 - Intelligence Tester
4K 918073 - Automatic Draw Poker
K 918074 - Game of Qubic
K 918075 - Racetrack
K 918076 - Game of Life
K 918079 - Deluxe Baseball
K 918080 - Ant Invasion
K 918081 - Game of Blitz
K 918082 - Double Play Game
K 918083 - Battleship
9K 918084 - Super Mindbreaker
K 918086 - Simam, 2 Digits
K 918087 - Simam, 3 Digits
K 918088 - Aircraft Carrier Landing
4K 918090 - Seven Card Stud Poker
K 918095 - Casino Roulette
9K 918097 - The Black Box
K 918100 - Blackjack for 13 Players
1JK 918101 - TI-59/PC-100 Demonstration
K 918102 - Hi-9
K 918103 - Interchange
K 918104 - Number Guesser (You Pick It, I'll Find It)
2K 918106 - The Knight's Tour
4K 918107 - Automatic Crap Game
K 918109 - Pattern
K 918110 - Peg Jump
K 918111 - Multiple-player Nim
K 918113 - One Dimensional Life
K 918115 - Hangman
4K 918117 - Action Craps
K 918118 - Anvanced Baseball
K 918119 - Chuck-a-luck Dice Game
K 918120 - Pattern PLanner Game
K 918121 - Dungeons and Dragons
K 918122 - Hanoi Game Simulator
K 918124 - Wipeout!
K 918126 - Duel
K 918127 - Wythoff's Nim
K 918129 - Bullseye Game
K 918132 - Road Race
2K 918133 - Stars - Number Guessing Game
K 918134 - Eight Calculating Queens
K 918136 - Blackjack
K 918140 - Tic Tac Toe
2K 918142 - Son of Jive Turkey
K 918143 - Magic Square
K 918145 - 999 Game

More PPX Program Availability - (cont)

K 918146 - Battleship
K 918148 - Roulette-59
K 918149 - Jumbled Letters
K 918150 - Bango
K 918151 - Hangman
K 918154 - Solitaire Checkers
K 918158 - Greedy
K 918161 - Space Search
K 918162 - Cross Country Auto Race
1JK 918163 - Hunt the Wumpus
K 918164 - Multi-player Race Car
K 918166 - Rock, Scissors, Paper
1 918167 - The Remainder Game
2 918168 - Nim with Strategy
K 918172 - Ski Race
9K 918185 - Baseball III
4K 918186 - Improved Tic Tac Toe
K 918187 - Checkmate!
K 918188 - Roll Five Dice
K 919189 - Two Dice Simulator
K 918190 - N5ch60ach4s' P4zz3e
K 918191 - Trap Game
K 918192 - High Stakes
4K 918193 - Poems with Six Letter Words
K 918199 - Bingo Card Printer
K 918200 - Chomp
K 918203 - Poems with Seven Letter Words
K 918204 - Phantom Banner in Own Background
K 918207 - Pascal Triangle Art
K 918209 - Russian Roulette
K 918211 - Baccarat
K 918212 - Thaipan
K 918213 - Pick a Number
K 918215 - The Mansion
9K 918218 - 4-D Lunar Lander
K 918219 - The Joker's Wild
K 918221 - Chase
K 918224 - Hexapawn
K 918225 - Cribbage Showing
K 918226 - Anagrams
K 918228 - Pinball II
K 918229 - Othello
K 918230 - The Dragon
K 918231 - Oasis
K 918232 - What Now
K 918233 - Treasure Hunt
K 918237 - So Sorry
K 918238 - Asteroid Attack
K 918239 - Anagrams
K 918241 - Mangman Illustrated
K 918242 - Presidential Campaign
9K 918245 - Misadventure
K 918246 - Ballistics Game

More PPX Program Availability - (cont)

K 918247 - Chemin de Fer
K 918248 - Le Cave
K 918249 - San Francisco Liar's Dice
K 918251 - Awari
K 918252 - Odd Ball
K 918255 - Mouse in a Maze
K 918256 - Buried Treasure
K 918257 - 3x3 through 6x6 Magic Square Generator
K 918259 - Hunt the Mugwump
K 918260 - Reverse
K 918261 - Adventurer
K 918264 - Barbudi (Bar Buddy)
K 918265 - Revised H-Lo
K 918266 - Othello II
7K 918267 - Las Vegas Blackjack #2
9K 918268 - Bingo
4K 918269 - Tic Tac Toe for Two
K 918270 - Instant Insanity
K 918271 - Rescue Mission: Planet Nari
K 918272 - Buck Rogers (with AM Radio Tiki Sounds)
K 918273 - Alien Invasion (with AM Radio Sound)
K 918275 - Baccarat
K 918277 - Simon Says
K 918278 - Drag Race
K 918279 - Monopoly (R) Game (for printer)
K 918280 - TI-59 Demonstration (Deluxe Version)
K 918281 - Yot Race
K 918282 - Jive Turkey Revisited
K 918283 - Darts
1JK 918286 - Rule Away
K 918287 - Cover Up
K 918288 - Elongated Banner
1JK 918290 - Poem Machine II
K 918291 - Santa Claus and His Reindeer
K 918295 - Snakes and Ladders
K 918296 - Two Deck Shuffle (Mixed)
K 918299 - Ice Hockey Game
K 918300 - Pictorial Craps
K 918301 - Liars Dice
K 918305 - Computer Perfection (R)
K 918306 - Towers of Hanoi Game with Display of Towers
K 918307 - Richelieu Valley Golf
K 918310 - Farkler
K 918314 - Recursive Tower of Hanoi
K 918318 - Cokes
K 928008 - Interactive Arithmetic Teacher
K 928022 - Spell-a-number
K 928033 - Sum of Digits Game
K 928041 - Sum of Three Digits
K 938015 - Distance between Two Geographic Points
K 938017 - Distance by Rudge's Formula
K 998042 - Julian Calendar Adjustment for ML-20
K 998043 - Date in the Future
4K 998094 - Julian Day Number Calendar

More PPX Program Availability - (cont)

Code 1 means the programs are available on a loan basis fro TI PPC Notes. Send one dollar (two dollars overseas) for each program that you wish to borrow. It is understood that the programs will be returned promptly to be available for other members.

Codes 2 through G were defined in earlier issues.

Code H means the programs are available free to members from Thomas Wysmuller, Phoenix Mutual, 1 American Row, Hartford CT 06115.

Code J (code I was skipped to avoid possible confusion with code 1) means the programs are available from Larry Parsons, 815 Atlantic St. NE, Warren OH 44483. In the U.S. send two dollars per program. From overseas write for terms.

Code K means the programs are available from Albert Smith, 14280 Sandhurst St., Brooksville FL 33573. Send a stamped and self-addressed envelope for costs for individual programs.

Code L means the programs are available from Laurance Leeds, 10232 El Dorado Drive, Sun City AZ 85351. Send a stamped and self-addressed envelope for details.

Code M means the programs are available from Thomas E. Ceteraki, 10010 Alderson St., Schofield WI 54476. Send a stamped and self-addressed envelope for details.

A SHORT PAUSE ON THE TI-58C? - In early 1983 V8N2P18 reported that Pierre Flener had discovered that a Pause on the TI-58C was significantly shorter than a Pause on the TI-58/59. Tests showed that a Pause on the editor's TI-58C lasted for only 0.16 seconds, as opposed to 0.45 seconds for typical TI-59's. Was this in the TI-58C design or was there some sort of ageing effect? A recent check shows a pause on my TI-58C still requires 0.16 seconds; no apparent degradation with time.

THE TI-59: STILL A VALUABLE DESIGN TOOL - Page 57 of the October 1985 issue of Aerospace Engineering is an advertisment by Abex Corporation introducing a new product called the Smartpump (R). The illustration shows one of the devices surrounded by design tools. There is a calculator in the picture. Is it an HP-41 or an HP-15? No. Is it a TI-66? No. Is it our beloved TI-59? YES!!!

1.0000001 TO THE 27TH POWER - V9N2P11 discussed Fred Gruenberger's test of computer precision from the April 1984 issue of Scientific American. Solutions were presented for three algorithms for a variety of machines. The column "Fixes and Updates" in the August 1985 issue of BYTE mentions this test and invites comments on its usefulness. The example in BYTE does not examine the Method C algorithm which is a good test of the accuracy of the logarithm function in the machine.

CHINESE REMAINDER THEOREM - Milton Brown, Larry Leeds, and others. V10N2P24 noted that George Thomson and Albert Smith had become interested in the Chinese Remainder Theorem (C.R.T.), and described the process by which we obtained a copy of Milton Brown's program (PPX 268008). George's application was in high accuracy matrix inversion, a subject we may cover in a future issue. Albert's application was to provide an aid in the solution of problems defined by Mark DiPippo's program "The Remainder Game" (PPX 918167). The abstract with the remainder game program states:

> "The calculator generates a hidden number. The user's job is to find this number. This is accomplished by entering numbers which will be used as divisors into the hidden number. Only the remainder of the division will be outputted. The user tries to find the hidden number in as few divisions as possible. A more advanced version of the game will only tell the user whether the remainder is odd or even."

The player enters the number of digits for the hidden number, the maximum number of digits for the divisor, and a seed used by the random number generator to generate a hidden integer N. In response to a test integer D the calculator returns the remainder of N/D. Albert proposed a specific problem where the hidden number is a four digit number and the test numbers are limited to two digits (As George Thomson points out, if you let the test numbers be sufficiently large relative to the hidden number, then the problem is trivial.):

Remainder of N/99 = 46

Remainder of N/98 = 58

Remainder of N/10 = 4

The appropriate values were entered into Milton Brown's program. The calculator ran slightly over seven minutes and stopped with a flashing 34300 in the display. The problem was that the divisors 99, 98 and 10 did not satisfy the restriction that the divisors must be relatively prime in pairs for the C.R.T. to work. A revised problem with the third divisor set at 5 yielded the correct solution 1234 in just under six minutes.

The long execution times suggested the use of fast mode. The subroutine calls were replaced with indirect addressing and the label addressing was converted to the absolute addressing required by fast mode. Fast mode was implemented with the Stflg at the end of partition method. We also found room to print the input and output with descriptors. The fast mode version of the program would solve Albert's problem in slightly over three minutes.

The program instructions indicated that up to 26 simultaneous congruences could be solved, where the limitation of 26 was related to the number of data registers available. As Larry Leeds continued to work with the program he discovered that there were other limitations related to the largest integers that can be stored in the memory registers of the TI-59. For example, the product of all the moduli (divisors) is one of the values used in the C.R.T. calculations. The product is accumulated in R05 during data input. The product of the

Chinese Remainder Theorem - (cont)

first 12 primes (2, 3, 5, 7 ... 31,37) is the thirteen digit integer 7,420,738,134,810. The product of the first thirteen primes would exceed the integer storage capability of the TI-59. Thus the maximum number of congruences that can be solved in 12, not 26. When dealing with moduli greater than 37 it is essential to have the program abort if the product is too large, even if the number of equations is less than 12.

Larry found other potential overflow conditions: for example, in the sum accumulated in R08 which is a function of the number of equations and the relative values of the moduli. Overflows beyond the integer storage capability of the TI-59 will result in erroneous answers which will not be detected without checking each of the equations. Consider the following problem:

Remainder of N/9973 = 3878

Remainder of N/9967 = 757

Remainder of N/9949 = 9623

The original program will display the solution as 1,899,999,000. But for that value the respective remainders for the divisions are 2878, 9724, and 8623. Printout of the data register contents will show that the values in R08 and R09 overflowed beyond the integer storage capability (exponents of 15). The correct solution to the problem is 1,900,000,000 but that result cannot be obtained with the TI-59 mechanization due to the overflows. Larry provided revisions to the original program to detect overflows and prevent the display of incorrect answers. See the program listing on page 28.

User Instructions:

The user instructions are similar to those in the original program. To solve sets of congruences of the form

$$a_i X = b_i \bmod m_i$$

1. Initialize by pressing E. This sets the partitioning to 9 Op 17, clears the data registers and stores some initial values. A "11" is returned to the display. That is the location where the first a value is to be stored.

2. Enter an a value and press R/S. The value is returned to the display and printed.

3. Enter a corresponding b value and press R/S. The value is returned to the display and printed.

4. Enter a corresponding m value and press R/S. The value is printed. The number of congruences entered is returned to the display.

5. Repeat steps 2 through 4 until all the equations have been entered.

Chinese Remainder Theorem - (cont)

6. Press D to solve. The calculator will stop with a flashing "1. 12" in the display. Do not clear the flashing condition, but press 7 and then EE. When the solution is complete the value of X be printed with annotation and the calculator will stop with X in the display. Press R/S and the value of M will be printed and displayed. M is the product of the input moduli (divisors). All of the solutions to the input problem are defined by X + M*j , where the j's are integers starting at zero.

A sample printout for Albert Smith's problem appears at the right. The execution time for that problem is slightly over 3 minutes.

```
    1.
   46.
   99.

    1.
   58.
   98.

    1.
    4.
    5.

 1234.          X
48510.          M
```

Error Indications:

1. If the product of the input moduli overflows the calculator will display a flashing "9.9999999 99" at step 4.

2. If the input moduli are not mutually prime the calculator will stop with a flashing "0" in the display.

3. If an overflow occurs in R08 during the solution the calculator will stop with a flashing "9.9999999 99" in the display.

Program Listing: You may record in the start-up partitioning since the initialization changes the partitioning needed by the program.

```
000 00  0     040 01  01    080 43 RCL   120 75  -     160 91 R/S   200 30  30
001 42 STO    041 00  00    081 03  03    121 53  (     161 72 ST*   201 32 X:T
002 04  04    042 95  95    082 54  )     122 43 RCL   162 07  07    202 04  4
003 53  (     043 61 GTO    083 42 STO    123 08  08    163 99 PRT   203 04  4
004 53  (     044 00  00    084 01  01    124 55  ÷     164 69 DP    204 69 DP
005 43 RCL    045 74  74    085 49 PRD    125 43 RCL   165 27  27    205 04  04
006 01  01    046 25 CLR    086 09  09    126 05  05    166 91 R/S   206 32 X:T
007 65  ×     047 42 STO    087 01  1     127 54  )     167 72 ST*   207 69 DP
008 43 RCL    048 08  08    088 42 STO    128 59 INT   168 07  07    208 06  06
009 04  04    049 01  1     089 02  02    129 65  ×     169 99 PRT   209 91 R/S
010 75  -     050 01  1     090 86 STF    130 43 RCL   170 69 DP    210 76 LBL
011 43 RCL    051 42 STO    091 01  01    131 05  05    171 27  27    211 13  C
012 02  02    052 07  07    092 61 GTO    132 54  )     172 91 R/S   212 03  3
013 54  )     053 73 RC*    093 00  00    133 92 RTN   173 72 ST*   213 08  8
014 55  ÷     054 07  07    094 00  00    134 76 LBL   174 07  07    214 69 DP
015 43 RCL    055 42 STO    095 49 PRD    135 15  E     175 99 PRT   215 04  04
016 03  03    056 01  01    096 09  09    136 22 INV   176 49 PRD   216 43 RCL
017 54  )     057 69 DP     097 22 INV   137 57 ENG   177 05  05    217 05  05
018 22 INV    058 27  27    098 86 STF   138 69 DP    178 43 RCL   218 69 DP
019 59 INT    059 73 RC*    099 01  01    139 00  00    179 00  00    219 06  06
020 67  EQ    060 07  07    100 43 RCL   140 09  9     180 32 X:T   220 98 ADV
021 00  00    061 42 STO    101 09  09    141 69 DP    181 43 RCL   221 98 ADV
022 37  37    062 02  02    102 44 SUM   142 17  17    182 05  05    222 91 R/S
023 69 DP     063 69 DP     103 08  08    143 01  1     183 77  GE    223 25 CLR
024 24  24    064 27  27    104 43 RCL   144 42 STO   184 02  02    224 69 DP
025 43 RCL    065 73 RC*    105 00  00    145 05  05    185 27  27    225 99  99
026 03  03    066 07  07    106 32 X:T   146 52 EE    186 29 CP    226 92 RTN
027 32 X:T    067 42 STO    107 43 RCL   147 01  1     187 69 DP    227 00  0
028 43 RCL    068 03  03    108 08  08    148 03  3     188 27  27    228 35 1/X
029 04  04    069 69 DP     109 77  GE    149 42 STO   189 69 DP    229 92 RTN
030 77  GE    070 27  27    110 02  02    150 00  00    190 26  26    230 05  5
031 02  02    071 61 GTO    111 27  27    151 25 CLR   191 43 RCL   231 05  5
032 23  23    072 00  00    112 29 CP     152 42 STO   192 06  06    232 02  2
033 29 CP     073 00  00    113 97 DSZ   153 06  06    193 98 ADV   233 85  +
034 61 GTO    074 42 STO    114 06  06    154 01  1     194 61 GTO   234 01  1
035 00  00    075 09  09    115 00  00    155 01  1     195 10 E'    235 52 EE
036 03  03    076 53  (     116 53  53    156 42 STO   196 76 LBL   236 01  1
037 43 RCL    077 43 RCL    117 53  (     157 07  07    197 14  D     237 02  2
038 04  04    078 05  05    118 43 RCL   158 76 LBL   198 71 SBR   238 95  =
039 87 IFF    079 55  ÷     119 08  08    159 10 E'    199 02  02    239 86 STF
```