PPX Exchange

Vol. 6 Number 3 Copyright 1982 May/June 1982

There's Gold In Those Guard Digits

By Palmer O. Hanson, Jr.

Page V-5 of *Personal Programming* introduces the TI-58/59 user to the guard digits with the following statement:

"Even though a maximum of 10 digits can be entered or displayed, the internal display register always retains results to 13 digits. The results are then rounded for display only. These extra digits guard the displayed value to insure its accuracy and are not intended to be extended precision. Using these guard digits should be done with extreme care ..."

Page C-5 of *Personal Programming* illustrates the kind of problems which can occur due to the guard digits with the sequence:

45 2nd sin - 45 2nd cos =

where the calculator returns the answer 7 X 10 —13, not the zero which might be expected, or at least hoped for. The illustration goes on to caution that care must be used when testing a calculated result against a test value, say with a t-register test. Use of the sequence EE-INV-EE, which truncates the guard digits of a result leaving only the rounded display for further use, is suggested as a method for guard digit control.

To the contrary of the cautions in *Personal Programming*, experimenters soon learned that thirteen digit values could be entered through the display register, for example, the keyboard sequence:

815323116 + .4437 =

will place the thirteen digit value 916323117.4437 in the display register. The thirteen digit number may be stored for future use. One of the earliest uses for thirteen digit numbers entered in this manner was for efficient print code storage. Clyde Durbin described the technique in the February 1978 issue of *52 Notes*. His approach took advantage of the difference between the ways in which the Op 04 and the HIR 08 commands transfer code to the print registers. Assuming that the thirteen digit number which was generated in the example above had been stored in data memory register R_{01}, then the program sequence:

RCL 01 OP 04 OP 06

continued on page 3

TI-59 Test

This is a test to determine how well you *really* know your TI-59. This test was developed by Maurice Swinnen, one of our Texas Instruments Professional Productivity Program instructors, for use with program seminar students.

Our programming seminars are designed to give the student the basic knowledge about the TI-59 necessary for him to write his own programs. The course has been designed to lead the student into becoming familiar with the TI-59, its functional grouping of keys, its various features and to learn how to solve basic problems in programming.

The course is built for individuals who have little or no programming knowledge and utilizes video tape, 35mm slides and a programming workbook as tools. It is presented in two sessions each consisting of three and one-half hours of 'hands on' experience with the TI-59. The first session, Beginning Programming, concentrates on the basic functional operations and simple programming. The second session, Intermediate Programming, is an introduction to features that can be used to optimize and increase the flexibility of the participants' own programs.

continued on page 5

The Computer Age Comes To PPX

After struggling along for 5 years with our membership rolls kept on floppy disks and our programs in file drawers, PPX has finally acquired a complete computer system to streamline PPX operations and to enhance the service to the membership. The system being installed is a TI-990/12 with 911 terminals for the program analysts. This will allow us to be more timely and accurate in updating address changes, as well as allowing a myriad of interesting statistical computations to be accumulated about the membership. The length of telephone calls will be shortened as program documentation will be more readily available, thereby saving the membership time and money. The results of last summer's survey indicated that there are a number of avid computer fans out there, and they may be interested in the possibility of telecommunications with PPX in the future.

Currently, only the program abstracts are updated on a database management system, and our access to the data is

continued on page 2

Computer Age *(cont'd from page 1)*

marginal at best. The programs themselves are kept in conventional files in our main office. Photocopies are filed in a storeroom. When a member calls for assistance with a program he has purchased, it is necessary to flip through drawers of folders to find the correct program. With the proposed system we will have the capability to examine the program at a video terminal and make necessary modifications quickly and efficiently.

Our mailing lists are updated using a card index file and a dinosaur (not a TI computer!) which uses floppy disks for permanent storage. This has been inefficient and time-consuming. With the new system, the mailing lists will be updated using sophisticated sorting and crosstabbing software utilities. Occasionally we receive a letter which has been damaged in the mail in which a member is asking for a list of items or help with a program he has purchased from PPX, and the only decipherable part of the address may be the last three digits of his zip code, or four letters in the middle of his last name. Previously, these requests were impossible to handle. But no more! We can now track him down with seemingly insignificant information and have his answer in the mail that same day.

When PPX needs to know statistical information concerning the membership, it is necessary to either compile the information by hand or contract a second party to do so. Any and all information of this kind will be available with a minimum effort using the software utilities provided with the system. Surveys will be conducted, and the wishes of the membership can be more easily understood.

Order processing will be sped up considerably as the computer will check the inventory and status of all programs ordered, and a high-speed printer will print a packing list.

The newsletter will increase in quality as time required to produce a final copy will be reduced due to the editing capabilities of the new equipment. It will be possible to send out copies of a revised program to all members who have purchased an unrevised copy. Recognition can be given to the most prolific authors, and specialized lists can be compiled on request.

Finally, if the benefits to the membership are proved and the funding made available, PPX members will be able to send in orders and programs using their home computers.

In conclusion, the installation of the 990/12 will make a positive contribution to the membership of PPX by saving them time and providing better and more varied services.

PROGRAMMING CORNER

(This column serves a dual purpose. It informs members of what non-PPX software is currently available and also lists descriptions of programs our members would like to see.)

- Dave Leising of Jet Electronics and Technology Inc. has sent us a brochure on an interesting application of the TI-59. The engineers at Jet have incorporated the TI-59 into the DAC-7000 3-D navigation system. Any TI-59 can be used with a special umbilical cord and DAC-7000 programming card. This allows route information to be

continued on page 4

13 Digit Register Lister Results

The challenge submitted to the membership was: "create a program that will list all thirteen digits of the contents of data registers 00 through 89 in the format shown below".

```
00  3.14159265359 00
01 -2.71828182845 9 16
02  0.0000000000 00
03  2.71828182845 9-23
```

In that article, it was noted that Jay Claborn had a program that turned in a 'sluggish' time of 3 minutes and 41 seconds for ten registers. Well, judging by the poor response to the contest, this was not as sluggish as he thought. For whatever reason, less than 20 entries were submitted. Execution time ranged from 2 minutes and 35 seconds to 3 minutes and 49 seconds.

The winner was Mr. Markus Sveinn Markusson from Iceland. His program crunches out 10 registers containing -3.141592700000-55 in 2 minutes, 35.8 seconds using our TI-59 and PC-100C. This was more than one minute less than the time Jay turned in with the challenge. Mr. Markusson's program flew through 10 registers in 2 minutes and 57 seconds when those registers contained -9.999999999999-99. The programs were judged using the registers 40 through 49 to eliminate any advantage that some programs may have had in handling lower registers. Our congratulations to Mr. Markusson for his remarkable effort!

000	76	LBL	048	53	53	096	52	EE	144	90	90
001	16	A'	049	93	.	097	55	÷	145	16	A'
002	55	÷	050	01	1	098	28	LOG	146	52	EE
003	01	1	051	95	=	099	77	GE	147	06	6
004	00	0	052	92	RTN	100	01	01	148	82	HIR
005	75	-	053	93	.	101	04	04	149	35	35
006	22	INV	054	01	1	102	01	1	150	01	1
007	59	INT	055	85	+	103	94	+/-	151	44	SUM
008	65	×	056	93	.	104	59	INT	152	90	90
009	77	GE	057	00	0	105	82	HIR	153	82	HIR
010	00	00	058	02	2	106	34	34	154	33	33
011	16	16	059	95	=	107	22	INV	155	69	OP
012	93	.	060	92	RTN	108	28	LOG	156	17	17
013	09	9	061	76	LBL	109	52	EE	157	71	SBR
014	95	=	062	11	A	110	75	-	158	00	00
015	92	RTN	063	32	X:T	111	22	INV	159	24	24
016	93	.	064	01	1	112	59	INT	160	82	HIR
017	09	9	065	00	0	113	82	HIR	161	06	06
018	85	+	066	69	OP	114	08	08	162	71	SBR
019	93	.	067	17	17	115	85	+	163	00	00
020	00	0	068	32	X:T	116	32	X:T	164	24	24
021	02	2	069	82	HIR	117	08	8	165	82	HIR
022	95	=	070	03	03	118	77	GE	166	07	07
023	92	RTN	071	42	STO	119	01	01	167	25	CLR
024	01	1	072	90	90	120	23	23	168	82	HIR
025	52	EE	073	32	X:T	121	02	2	169	18	18
026	05	5	074	69	OP	122	85	+	170	52	EE
027	82	HIR	075	17	17	123	73	RC*	171	02	2
028	48	48	076	73	RC*	124	90	90	172	71	SBR
029	82	HIR	077	90	90	125	29	CP	173	00	00
030	18	18	078	50	I×I	126	77	GE	174	41	41
031	59	INT	079	82	HIR	127	01	01	175	82	HIR
032	82	HIR	080	04	04	128	33	33	176	08	08
033	58	58	081	29	CP	129	02	2	177	82	HIR
034	77	GE	082	67	EQ	130	52	EE	178	14	14
035	00	00	083	01	01	131	03	3	179	50	I×I
036	38	38	084	13	13	132	85	+	180	71	SBR
037	92	RTN	085	55	÷	133	43	RCL	181	00	00
038	16	A'	086	28	LOG	134	91	91	182	41	41
039	16	A'	087	77	GE	135	95	=	183	75	-
040	16	A'	088	00	00	136	82	HIR	184	82	HIR
041	16	A'	089	91	91	137	05	05	185	14	14
042	65	×	090	52	EE	138	93	.	186	29	CP
043	93	.	091	59	INT	139	08	8	187	77	GE
044	01	1	092	82	HIR	140	09	9	188	01	01
045	65	×	093	04	04	141	32	X:T	189	94	94
046	77	GE	094	22	INV	142	25	CLR	190	01	1
047	00	00	095	28	LOG	143	43	RCL	191	09	9

192	94	+/-	200	94	+/-	208	36	36	216	13	13
193	76	LBL	201	82	HIR	209	82	HIR	217	32	X:T
194	01	1	202	38	38	210	37	37	218	08	8
195	95	=	203	25	CLR	211	82	HIR	219	09	9
196	22	INV	204	09	9	212	38	38	220	77	GE
197	52	EE	205	09	9	213	69	OP	221	00	00
198	52	EE	206	35	1/X	214	05	05	222	74	74
199	06	6	207	82	HIR	215	82	HIR	223	91	R/S

To run the program enter the starting register number and press [A].

There's Gold ... *(cont'd from page 1)*

will yield the tag DONE (print code 16 32 31 17). Similarly, the program sequence:

RCL 01 HIR 08 OP 06

will yield the tag NEXT (print code 31 17 44 37). The leading nine in the stored number is a dummy filler. It is needed to fill the thirteen digit print register and force the right justification such that the HIR 08 OP 06 sequence will print in response to the eight least significant digits. The nine is ignored by the OP 04 OP 06 sequence which prints in response to the eight least significant digits to the left of the decimal point. The dummy filler may not be needed for certain print code combinations. In a recent program I needed to print the letters EXT at the OP 02 location in one part of the program, and a ▲ LAT tag in another part of the program. I stored the thirteen digit number 174437.5271337 at data memory location R_{38}. Then the sequence:

RCL 38 OP 02 OP 05

yielded EXT at the OP 02 location, using the six digits to the left of the decimal point. The sequence:

RCL 38 HIR 08 OP 06

yielded the ▲LAT tag, where the seven from the 37 code for the T in EXT provided the seven in the code 75 needed for the ▲ in ▲LAT.

A similar method of synthesizing thirteen digit numbers in the display register can be used to extend the capability of program which might have appeared to be limited to ten digit inputs. The documentation for Daniel Drucker's "Prime Factors of an Integer" program (PPX 398075) included the example of a twelve digit input integer

1,111,111,111 x 100 + 111,111,111,111

for which the prime factors are 3, 7, 11, 13, 37, 101, and 9901. The same technique can be used to extend the range of George Vogel's prime factor finder in the article "It pays to Analyze Your Problem" in the January/February issue of *PPX Exchange*, or with the Prime Factors program in the Math/Utilities module. Those programs operate properly with twelve digit inputs, but do not yield correct results with most thirteen digit inputs. The limitation is due to the method used to test for prime factors. The three programs, and most other prime factor programs written for the TI-58/59, use some variation of the test sequence:

RCL 01 ÷ RCL 02 = INV INT

where the integer to be factored is in R_{01} and the integer to be tested as a factor is in r_{02}. The result is tested against a zero in the t-register. If the result is not zero, the test integer is not a factor, and the program generates a new test integer. Clearly, there must be at least one digit available to the right of the decimal point if the test is to work. This sets the limit on the input integer at twelve digits for that algorithm.

The truncation characteristics of the display register can be used to devise an algorithm which will work for thirteen digit input integers. While the ten digit display is obtained by rounding from the three guard digits, the thirteen digits in the display register seem to be obtained in a truncated, not a rounded format. If we fill the register with a thirteen digit integer and divide by a small integer such that the quotient is still a thirteen digit number to the left of the decimal point then a "built-in" integer function will have taken place. We may use this characteristic to do modulo arithmetic to find prime factors with the program sequence:

RCL 01 - (CE ÷ RCL 01) x RCL 02 =

The result is tested against a zero in the t-register, with the same decision process as for the earlier algorithm. To cover the cases where the division yields a quotient of less than thirteen digits to the left of the decimal point we add an integer function to the sequence:

RCL 02 - (CE ÷ RCL 02) INT x RCL 02 =

The double integer function which occurs when the quotient has thirteen digits to the left of the decimal point is an idiosyncrasy of the technique which does no harm.

Once the tests of 2, 3, 5, and 7 as factors have been completed using the thirteen digit algorithm the tests of subsequent factors must necessarily yield a quotient with twelve or less digits to the left of the decimal point. It might seem that this would permit switching to the shorter, and faster, algorithm for the subsequent testing. Unfortunately, that algorithm will sometimes fail for large input integers. An example will serve to illustrate the problem. Consider the thirteen digit integer 9,999,999,999,991. If after testing the factors 2, 3, 5, and 7 with the longer algorithm, then the next prime, 11, is tested using the shorter algorithm the division yields

909090909090.09090909...

which the display register *truncates* to

909090909090.0

The INV INT function then yields a zero indicating that 11 is a factor, when it is not. A little reflection will show that the quotient after division by a test integer must allow a fractional part in the display register with as many decimal places to the right of the decimal point as there are digits in the test integer. This constraint will be satisfied if the input integer is limited to twelve digits. Thus, the longer algorithm must be used if thirteen digit input integers are allowed, at least until a factor is found which reduces the input integer to twelve or less digits. PPX 398278 is a thirteen digit factor finder which uses the longer algorihm throughout. After somewhat over one hundred twenty hours that program declared the number 9,999,999,999,971 to be prime. If nothing else that exercise is a demonstration of the reliability of my TI-59. The program

continued on page 10

Programming Corner *(con't from page 2)*

entered at home or office. More information can be obtained from:

JET ELECTRONICS AND TECHNOLOGY, INC
5353 52nd Street
Grand Rapids, Michigan 49508
Telephone (616) 949-6600
Telex 22-6453 JETELECTEC GDR

PROGRAMS WANTED

The program requests for this issue are listed below. All submissions to fill these requests should be postmarked no later than June 30, 1982.

- A program utilizing "Applied Optimal Estimation" written by the technical staff of the Analytical Sciences Corporation to take a one variable monotonically increasing set of values and make future predictions to create a recursive discrete Kalman filter that will use the optimal predictor formulas to generate value estimates, to have a subroutine that will generate upper and lower limits for both the predicted value and smoothed value within certain confidence intervals and to have a subroutine that will utilize the optimal linear smoothing formulas to generate smoothed predictions.
- Post-tension concrete slab on grade design. Design in accordance with post-tensioning institute principles and the BRAB report. Design should include slab thickness, spacing of the tendons for given size tendon, and prestressing force. Slab design is for highly plastic soils with potential vertical movement.
- Reinforced concrete beam design (U.S.D.) with cantilevered end. Design should accomodate numerous loading types and given a trial section provide top and bottom steel areas, bar cut off points; stirrup size and spacing.
- A program to determine moon/sun conjunction for each month for years 1981 to 2600 and beyond to the nearest minute.
- A program to fit a curve to data for a function of two independent variables, i.e., Z = F(x,y). Given two coordinates and a function value, the program should return coefficients as in a least-mean-square fit for a function of one independent variable. If this is not possible, the program should perform an interpolation between cardinal points; given the (x,y) value it should return a value for Z = f(x,y).

Letters to the Editor

Do you have comments or questions on the Exchange or other aspects of PPX that might benefit other members? We have always welcomed letters from our membership, and, therefore, we provide this space in the newsletter for you to share your views with your fellow members. Approximately 2-4 letters dealing with issues of general interest will be featured as space permits.

Dear Editor,

The article entitled "Root Finding: A Natural Application" by Blake DeBerry and Jay Claborn that appears in the January/February 1982 issue of the *PPX Exchange* was of considerable interest to me and was well done. The "Regula Falsi" or false position method is of particular value since it will find all the real roots of an equation lying within a specified range if the sampling interval is chosen properly.

As I experimented with the Regula Falsi program listed in the article, I noticed that it will not identify roots which happen to lie at boundary values of the range and the sampling intervals. In other words, using the sample equation: $f(x) = X^3 + 5X^2 - 64X - 140$, choosing -20 as the left bound and $+10$, is located at a sampling interval boundary. Similarly, if a sampling interval of 1 is chosen, no roots will be located since all fall at sampling interval boundaries.

Judicious selection of the left and right bounds and the sampling interval may not always be practical if the Regula Falsi routine is used in a more comprehensive program where variables may constitute oefficients, exponents and/or constants in a basic equation. Also, sampling intervals may have to be made relatively small to insure that all roots are isolated.

One possible solution, using the Regula Falsi program listed in the above mentioned article, is to choose irregular boundary values and sampling intervals. For example, with bounds of -20.1 and $+10.1$, and a sampling interval of 5 or 1 or 1.1, all three roots of the sample equation are found. This technique will not always eliminate the general hazard of missing a root which happens to fall at a sampling interval boundary.

Another solution, which I have undertaken, is to modify the Regula Falsi program so that it will identify roots that fall at any boundary value withing and including the lower and upper limits. The program is merely an add-on to the program listed in the article, and does not involve any major restructuring of the program flow. The change does seem to overcome the problem described previously, and a copy of the modified program is attached for your review.

In closing, I have found recent *PPX Exchange* newsletter articles that deal with specific problems and application, and provide illustrative program listings, to be very helpful. Please keep them coming.

Your truly,
John A. Lawlor
Sharon, MA

```
000  76 LBL   026  03  03   052  61  61   078  65  ×
001  11  A    027  91 R/S   053  87 IFF   079  43 RCL
002  42 STO   028  43 RCL   054  01  01   080  10  10
003  01  01   029  07  07   055  01  01   081  95  =
004  22 INV   030  91 R/S   056  61  61   082  29 CP
005  86 STF   031  76 LBL   057  86 STF   083  77  GE
006  01  01   032  14  D    058  01  01   084  14  D
007  93  .    033  43 RCL   059  42 STO   085  43 RCL
008  00  0    034  01  01   060  06  06   086  09  09
009  01  1    035  42 STO   061  43 RCL   087  55  ÷
010  42 STO   036  05  05   062  06  06   088  53  (
011  04  04   037  85  +    063  16 A'    089  24 CE
012  91 R/S   038  43 RCL   064  87 IFF   090  75  -
013  76 LBL   039  03  03   065  01  01   091  43 RCL
014  12  B    040  95  =    066  17 B'    092  10  10
015  42 STO   041  42 STO   067  42 STO   093  54  )
016  02  02   042  01  01   068  10  10   094  65  ×
017  91 R/S   043  42 STO   069  43 RCL   095  53  (
018  76 LBL   044  06  06   070  05  05   096  43 RCL
019  15  E    045  42 STO   071  16 A'    097  06  06
020  42 STO   046  07  07   072  29 CP    098  75  -
021  04  04   047  32 X:T   073  67  EQ   099  43 RCL
022  91 R/S   048  43 RCL   074  01  01   100  05  05
023  76 LBL   049  02  02   075  64  64   101  54  )
024  13  C    050  77  GE   076  42 STO   102  85  +
025  42 STO   051  00  00   077  09  09   103  43 RCL
```

```
104  05  05   | 130  11  11   | 156  42  STO  | 182  61  61
105  75  -    | 131  65  ×    | 157  09  09   | 183  76  LBL
106  48  EXC  | 132  43  RCL  | 158  61  GTO  | 184  16  A'
107  07  07   | 133  09  09   | 159  00  00   | 185  42  STO
108  95  =    | 134  95  =    | 160  85  85   | 186  12  12
109  33  X²   | 135  29  CP   | 161  25  CLR  | 187  65  ×
110  42  STO  | 136  77  GE   | 162  35  1/X  | 188  53  (
111  08  08   | 137  01  01   | 163  91  R/S  | 189  24  CE
112  43  RCL  | 138  50  50   | 164  43  RCL  | 190  33  X²
113  07  07   | 139  43  RCL  | 165  05  05   | 191  85  +
114  16  A'   | 140  07  07   | 166  91  R/S  | 192  05  5
115  42  STO  | 141  42  STO  | 167  14  D    | 193  65  ×
116  11  11   | 142  06  06   | 168  76  LBL  | 194  43  RCL
117  33  X²   | 143  43  RCL  | 169  17  B'   | 195  12  12
118  85  +    | 144  11  11   | 170  29  CP   | 196  75  -
119  43  RCL  | 145  42  STO  | 171  67  EQ   | 197  06  6
120  08  08   | 146  10  10   | 172  01  01   | 198  04  4
121  95  =    | 147  61  GTO  | 173  77  77   | 199  54  )
122  34  √X   | 148  00  00   | 174  61  GTO  | 200  75  -
123  32  X:T  | 149  85  85   | 175  00  00   | 201  01  1
124  43  RCL  | 150  43  RCL  | 176  67  67   | 202  04  4
125  04  04   | 151  07  07   | 177  43  RCL  | 203  00  0
126  77  GE   | 152  42  STO  | 178  06  06   | 204  95  =
127  00  00   | 153  05  05   | 179  91  R/S  | 205  92  RTN
128  28  28   | 154  43  RCL  | 180  61  GTO  | 206  00  0
129  43  RCL  | 155  11  11   | 181  01  01   | 207  00  0
```

Dear Exchange Editor:

After 2½ years, my model 59 looked "grimy". The keyboard needed cleaning. But Texas Instruments has not provided instructions for cleaning the calculator!

I am also involved in photography (Nikon equipment). Lens cleaning fluid did the trick. I moistened small pads of tissue with the fluid (Kodak Lens Cleaner) and swabbed back and forth and up and down between the keys.

Now my older calculator looks as good as my new model 59. I bought a second calculator and printer for my desk at home. The older one is "at work". Incidentally, I enjoy having two calculators. Carrying one to and from work is a bother. Why not promote this for everyone?

Sincerely,
John Schmidt
Whitefish Bay, WI

TI-59 Test *(cont'd from page 1)*

In 1977 and 1978, Texas Instruments, Inc. equipped ten thousand employees with the TI-59 and trained them in the use of the calculator with the express purpose of increasing productivity. The employees equipped with the TI-59 perceived that by using the TI-59 to perform their repetitive tasks for them the average employee saved over two hours per week leaving them free for their more critical job responsibilities.

Texas Instruments found that by using the TI-59, productivity increased and we feel that both individuals and corporations can invest in productivity by attending programming seminars and utilizing the TI-59.

Please do not try to decipher the program as you key it in. After keying in the program and recording it on magnetic cards, if desired, press [CLR] and begin by pressing the user defined key corresponding to your answer to question 1. Read each question carefully and determine the appropriate response. Press the user-defined key corresponding to the question when prompted with the question number in the display. It is important that the question number be in the display when you select your answer. After tabulation, the number of the next question will appear in the display indicating that the calculator is ready for your next answer. If you make a mistake, or desire to answer a question out of sequence, enter the question number and press the appropriate user defined key. The display will prompt you with the next question number.

When using the program in conjunction with the PC-100A(C), you may obtain a list of your answers by pressing [SBR] [List]. When you have completed the test, press [SBR] [=] to obtain your score and a list of incorrect answers (if any). The incorrect question numbers will be flashed in the display for those of you not using a printer. Some of the members of our staff were quite surprised at the results we received. The answers and program listing can be found on page 11 of this issue.

1. The startup partition, as displayed by OP code 16 is ...
   a. 159.99   b. 479.59   c. 959.00   d. 249.29   e. none of these

2. Which of the following keys, when depressed in the calculate mode, will automatically branch to program memory location 000?
   a. [SBR]   b. [GTO]   c. [R/S]   d. [RST]   e. none of these

3. Which of the following keys clears only the display?
   a. [CE]   b. [CLR]   c. [CMS]   d. [CP]   e. none of these

4. According to the AOS™ algebraic heirarchy of the calculator, which of these operations will be performed first?
   a. [+]   b. [Y^x]   c. [$\sqrt{x}$]   d. [X^2]   e. none of these

5. If you press [LRN] to enter the learn mode, which sequence do you press to exit the learn mode?
   a. [CLR]   b. [INV] [SBR]   c. [LRN]   d. [CLR] [LRN]   e. none of these

6. When the calculator is first powered up, which of the following modes is it in?
   a. Radian   b. Learn   c. Trace   d. Grad   e. Degree

7. Which is an example of a user defined label?
   a. [LBL] [STO]   b. [LBL] [A]   c. [GTO] [C]   d. [SBR] [SIN]   e. [GTO] [IND]

8. Which of these is not a subroutine call?
   a. [C]   b. [SBR] [SIN]   c. [x=t] [A]   d. [SBR] [A]   e. none of these

9. Which of the following keys, when depressed in the calculate mode, will multiply a value stored in a memory by a number in the display?
   a. [x≷t]   b. [PRD]   c. [SUM]   d. [IND]   e. none of these

10. Which of the following [OP] codes will download a module program into the user program memory?
    a. [OP] 01   b. [OP] 09   c. [OP] 10   d. [OP] 13   e. [OP] 18

11. If one finds that, while keying in a program in the learn mode, an instruction has been left out, which key will allow easy entry of that instruction?
   a. [GTO] d. [INS]
   b. [RST] e. none of these
   c. [DEL]

12. Four of these instructions are unconditional branching instructions. Which one is a conditional branching instruction?
   a. [DSZ] 0 [SIN] d. [SBR] 859
   b. [GTO] 248 e. [GTO] [B']
   c. [RST]

13. Most of the keys on a TI-59 keyboard can be used as common labels. Which of these cannot be used?
   a. [NOP] d. all of these
   b. [INV] e. none of these
   c. [IND]

14. Which of the following is not a flag use?
   a. Keeping track of program execution history
   b. Controlling program options manually before program execution
   c. Testing program conditions during program execution
   d. All of the above
   e. None of the above

15. How many keys must be pressed in the calculate mode in order to find 8^{30}?
   a. 4 d. 7
   b. 5 e. 8
   c. 6

16. Looking at the program steps:

[LBL]	[LBL]
[A]	[B]
58	[STO] [IND]
[STO]	45
45	[R/S]
[RTN]	

   What values will be stored in which registers after the sequence: Press A; Enter 60; Press B?
   a. 58 in R_{45}, 60 in R_{45} d. 60 in R_{45}, 45 in R_{58}
   b. 58 in R_{45}, 60 in R_{58} e. None of the above
   c. 60 in R_{45}, 45 in R_{58}

17. [OP] 08 performs the following operations:
   a. Downloads a program from the module into user program space
   b. Prints an asterisk in a predetermined location on the PC100A(C)
   c. Partitions memory
   d. Lists labels
   e. None of these

18. Which of the following math equations cannot be keyed in as written?
   a. (1+2) (3+4) = 21
   b. (((2×(2×(2×(2×(2×(2×Y^x(2+.2))(2+2)))))) /2)/2) = 36.75834736
   c. 1500 × (1+.06)Y^{x}5 = 2007.338366
   d. (6/3) × 10 + (46) = 18
   e. All of these may be keyed in as written

19. How many significant figures may be stored in a data register?
   a. 12 d. 8
   b. 13 e. 1
   c. 10

20. What is the maximum number of data memories available in the TI-59?
   a. 60 d. 959
   b. 99 e. 960
   c. 100

21. Which of the following is not a decision-making function?
   a. The "t" register instructions d. The [EXC] instruction
   b. The [DSZ] instructions e. [OP] 19
   c. Flags

22. Approximately how many memories are available in a library module?
   a. 0 d. 1000
   b. 100 e. 5000
   c. 625

23. When you read a protected program into the calculator, which of the following operations are made inoperative?
   a. List d. Repartitioning memories
   b. Trace e. All of these
   c. Single-Step

24. When in the learn mode, which of the following keys does not enter an instruction?
   a. Print key on the keyboard d. [ADV] key on the PC100A(C)
   b. Print key on the PC100A(C) e. All of these
   c. Trace key on the PC100A(C)

25. If a program in user memory is written to call a subroutine in a library module, what should be taken into consideration to assure successful operation?
   a. Whether or not the library module program has been downloaded
   b. The user instructions for the specific module
   c. Appendix A in the module library book
   d. The number of user program steps available
   e. All of these

26. If you want to add a displayed value to the contents of data memory 08, which of the following key sequences should you use?
   a. [RCL] 08 [+] [=] [STO] 08 d. [SUM] 08
   b. [STO] 08 e. [(] [+] [RCL] 08 [)]
   c. [+] [RCL] 08 [STO] 08 [+]

27. Having just completed a compound interest problem using Master Library program 18, you attempt to read in a 220 step program from a magnetic card with the standard power-up partition, and you get a flashing number in the display. What should you do to correct the problem?
   a. Change the partition to 239 program steps
   b. Clear the data memories
   c. Press [INV] [FIX]
   d. Press [CP] to clear the program
   e. None of these

28. Which is an example of an unconditional loop?
   a. [LRN] [+] 1 [=] [PAU] [GTO] 000
   b. [OP] 20 [DSZ] 9 000 [R/S]
   c. [LBL] [A] [+] 1 [=] [GTO] 020
   d. 5 [STO] 01 [GT*] 01 [RTN]
   e. 4 [+] 1 [=] [IFF] 1 000 [RTN]

29. The 2nd key was pressed by mistake. In order to recover from this error, press ...
   a. [CE] d. [INV] [2nd]
   b. [2nd] e. [CLR]
   c. [CLR] [2nd]

30. When you reduce your data memories by 20, how many program locations do you add?
   a. 20 d. 160
   b. 100 e. 240
   c. 120

31. Which would be expected to have the fastest program execution?
   a. A program using only absolute addressing
   b. A program using only label addressing
   c. A program using only indirect addressing
   d. A program using both indirect and label addressing
   e. A program using both indirect and absolute addressing

32. The value 2.42 05 is displayed. What is displayed if [ENG] is press-

Announcing the TI Programmable 88!

Texas Instruments, inventor of the integrated circuit and one of the largest manufacturers of programmable calculators in the world, brings a new perspective to hand-held computing with the introduction of the TI Programmable 88. Whether you're an engineer, scientist, educator, or businessman, you can solve problems easier than ever before using the new TI-88 with Texas Instruments' unique alphanumeric display capability and *Solid State Software*™.

The TI-88 is more than a programmable calculator; it is a new world of portable programming convenience. A variety of special features and accessory peripheral devices offers problem-solving power unmatched by any other hand-held programmable calculator.

The TI-88's two module ports provide the user with the unprecedented capability of installing up to **30,000 steps** of preprogrammed *Solid State Software*, or, by installing TI's new *Constant Memory*™ expansion modules, over **2,000 program steps** or nearly **300 data registers** can be added for use in program development.

The **new user-response keys** [YES], [NO], [UNK], [ENT], and [CONT] offer a convenience afforded previously only by sophisticated large-scale computers. By selecting the system prompting option, the TI-88 can greet you with the message "MAY I HELP YOU?" Responding by pressing the [YES] key leads to a series of alphanumeric prompts designed to assist you in running *Solid State Software* programs and in setting the calculator's time, date, and alarm options.

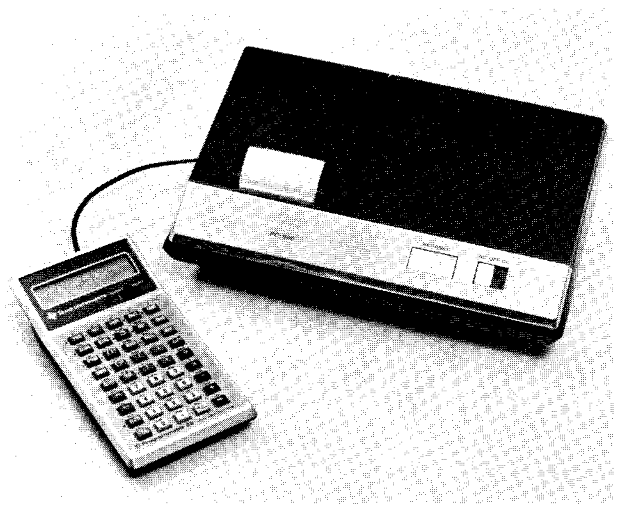
By adding the PC-800 Printer Accessory, a printed copy can provide an instant record of all calculations. With the optional CA-800 Cassette Interface Accessory and a cassette recorder connected to the calculator, you can permanently record programs and data on standard audio cassette tapes for later retrieval, providing virtually unlimited data storage.

The TI-88 can communicate directly with other devices using the built-in input/output port on the calculator and a **rechargable 150-hour battery** ensures total portability.

Timekeeping circuitry can act as an alarm and update the year, date, day, hour, and second continuously even when the calculator is off. The time can be accessed through program control for added convenience in record-keeping and calculations involving time.

Several tone instructions allow the options of beep on keypress, prompt, or error. Programs can be fully 'traced' in the display without a printer. The key buffering feature allows up to 15 key presses to be made while the calculator is busy. The calculator processes each key press as it finishes the last task.

These features coupled with a reputation that's second to none in the electronics industry, makes the TI-88 a leader among programmable calculators.



ALPHANUMERIC DISPLAY

The ultimate has finally arrived for alphanumeric hand-held calculators. The TI-88 has a 16 character Liquid Crystal Display. Each of the **128 available characters** is represented by a 5 X 7 dot matrix. The LCD dot matrix is better defined than the segmented display format used by other programmable calculators, and the TI-88's new styling includes a tilted display for ease of viewing.

The ability to display messages provides unprecedented built-in software conveniences. Upper and lower case letters, punctuation, superscripts, common Greek letters, and other special characters greatly increase the flexibility and applicability of the already high performance computing device. A special Alpha Mode is provided to aid in the entry of characters into the display or for use in programs.

The advantages of this new capability are endless. Prompting messages guide the user through applications programs. System error messages are displayed using plain English. Program execution can be traced in the display. Current calculator status, special functions and their definitions are available with only a few keystrokes.

Program development takes place in the dramatically improved Learn Mode where every instruction is represented by English abbreviations. These meaningful abbreviations can be interpreted more easily than previous numeric code systems. The instructions are scrolled from right to left as they are keyed in. Several instructions are displayed at one time making it easy to examine your own programs for errors.

ENHANCED AOS™ SOLVES PROBLEMS EASIER

The TI-88 has all of the features which has made the TI-59 famous for ease of use, plus much more. As you key in a problem, each keystroke is echoed in the display. Since the last keystroke is usually visible, fewer missed key sequences occur, and the possibility of multiple entries is lessened.

The TI-88 is equipped with an enhanced version of Texas Instruments' Algebraic Operating System (*AOS™*). Most problems are solved by entering them into the calculator in the order written. Answers are obtained simply and directly—no awkward transpositions, intermediate calculations, or misleading key sequences are required. Now, **implied multiplication** is recognized by the *AOS* and the square root, logarithmic, and trigonometric functions can be followed by their arguments as when working with pencil and paper.

EQUATION ENTRY SYSTEM

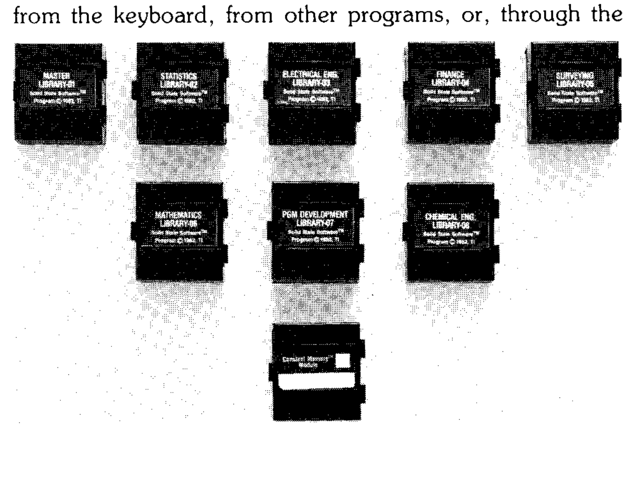
To further aid in solving equations, the TI-88 incorporates a special Equation Entry System. This amazing feature allows entry of an equation directly into memory for later evaluation. As the equation is entered, the function is scrolled across the display for easy verification. When the equation has been keyed in, evaluation is accomplished repeatedly by simply pressing the [EVAL] key.

By placing easy-to-use instructions in front of the equation, the calculator can prompt the user to enter the value of each variable defined in the equation. A total of 88 instructions can be entered into the Equation Entry System's special dedicated memory. Combined with the *Constant Memory* feature which retains the equation even when turned off, the power of the calculator for solving problems becomes truly remarkable.

CONSTANT MEMORY™ EXPANSION MODULES

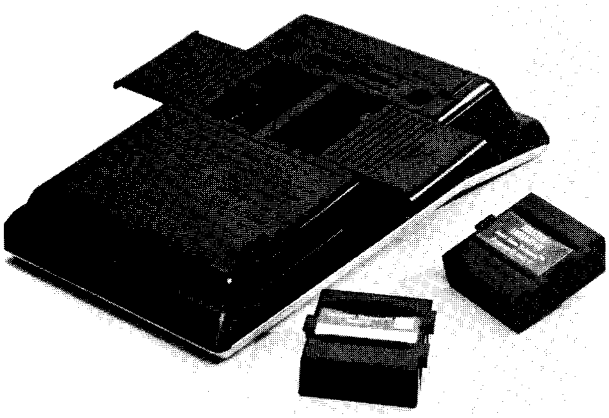
The memory capacity and problem-solving capability of the calculator can be greatly enhanced by installing optional *Constant Memory* expansion modules in one or both of the ports in the TI-88. Installing a *Constant Memory* module expands memory capacity to 268 data memories or 2,144 program steps. Installing a second *Constant Memory* module boosts maximum memory capacity to **416 data memories** or **3,328 program steps.**

You can also create your own customized program modules by assigning an "identification number" to any *Constant Memory* module. These modules can be **protected** from user examination of program content. After being "numbered," a *Constant Memory* module has 1,160 program steps and can store as many as **ten programs.** Since each module contains its own battery, it can be removed from the calculator without clearing or changing its contents. When you wish to access the programs stored in the module, simply reinstall the module in the calculator. Programs stored in *Constant Memory* modules may be executed in the same manner as *Solid State Software* programs from the keyboard, from other programs, or, through the



Main Prompting Sequence.

The battery contained in a *Constant Memory* module maintains all information stored in the module for an estimated **lifetime of five years.** Since modules are solid state devices, they do not wear or degrade with use and are **fully compatible** from one calculator to another.



SOLID STATE SOFTWARE™

Most people who need software don't have the time or desire to design all of the programs they require. To take the worry out of programming, Texas Instruments offers software for the TI-59 in the form of inexpensive, pre-programmed, 5,000 step modules. These modules are used to great advantage by doctors, lawyers, engineers, and businessmen within their professional discipline, and students around the world have found that learning is more exciting when repetitive work is eliminated with the use of a hand-held device. Continuing this tradition, the TI-88 is supported by quality software in 15,000 step modules which will help provide professional solutions to professional problems.

The use of the programs within *Solid State Software* modules is greatly enhanced by the calculator's response keys. The [YES], [NO], [UNK], [ENT], and [CONT] keys provide a truly **user-friendly** environment. No training or knowledge of programming is needed to answer the prompted messages which guide you to the solution of your particular problem. Some TI-88 applications modules even offer prompting in other languages such as German and French.

Master Library

Included with each TI-88 is the Master Library Module which contains 12 professionally written, easy-to-use programs chosen to serve a broad range of interests and to provide an overview of the calculator's capabilities. The 12 programs require no programming knowledge or experience to use, and allow you to begin immediately taking advantage of the programming power of your calculator.

The **Table of Contents** program is provided as a quick alphanumeric reference to the contents of the library.

A **Diagnostic** program tests the TI-88's functions, memory (including installed memory expansion modules), and display.

The **Finance** program calculates compound interest, computes ordinary annuities and annuities due, and prepares amortization schedules (with periodic totals and subtotals selectable by the user) for ordinary annuity/present value situations.

Computing moving averages is quick and easy with the **Moving Averages** program.

Find all real roots of a function using the **Root Finder** program.

Numeric integration is handled using the Romberg method in the **Integration** Program.

The **Matrices** program can handle up to a 9 by 9 matrix (15 by 15 with a TI memory expansion module installed). Linear algebraic operations such as addition, multiplication, inversion, determinants, and solution of simultaneous equations are executed quickly and accurately.

The **Linear Regression** program can estimate the coefficients of a model with as many as 9 predictors (13 with a TI memory expansion module installed). A correlation matrix and additional statistical information can also be compiled.

The **Random Number Generator** program produces sequences of uniformly or normally distributed random numbers.

Guess a four-digit number while playing **Codebreaker**, an all-time favorite calculator game.

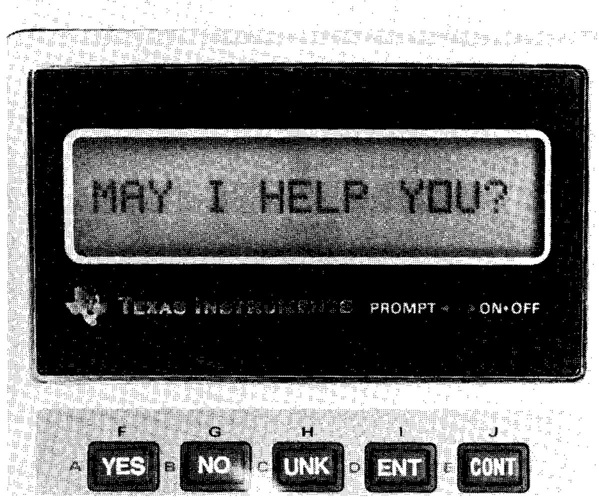
The **Sorting** program uses the efficient partition-exchange algorithm to order numeric lists.

The **Function Evaluator** program computes the value of a function at selected points and prints simple plots when used with the optional PC-800 printer.

Applications Modules

Applications modules will be available for the TI-88 in a variety of fields including Math, Statistics, Engineering, Games, and Business.

A Program Development Module will be available' to translate a TI-59 program and leave it resident in program memory ready to run on the TI-88. This will allow an easy transition for TI-59 owners wanting to upgrade to a more powerful calculator.



REAL PROGRAMMING POWER

In addition to improving TI's *Algebraic Operating System*, the TI engineers designed the most efficient and easy-to use language that has ever been implemented on a programmable calculator. Combined with increased memory size, the TI-88 has programming power to reduce the most formidable of programming tasks previously handled only by large-scale computers to a hand-held solution.

New Register Addressing

The first 26 data registers in the calculator can be addressed numerically or by a new short-form addressing technique in which the register contents are accessed by an alphabetic address from A to Z. Combined with implied multiplication and the enhanced *AOS*, this addressing mode makes possible the recall of variables by simply mentioning their names. Expressions such as $\text{ASinB} - \text{CD}^2$ are entered into the Learn Mode exactly as written.

Versatile Branching Instructions

In addition to transferring program execution directly to specific program addresses, branching can take place to any of 26 alpha labels and 100 numeric labels. Replacing keytop labels used by the TI-59 with numeric labels allows the TI-88 to expand its indirect addressing capabilities to include indirect transfers to label addresses. Relative addressing, another new branching feature, can be used in creating programs which are easily relocated in program memory without fear of affecting branching locations.

Powerful Decision-Making Instructions

The new decision-making instructions allow the conditional execution of any TI-88 instruction; not just branching instructions. All six mathematical comparisons are available and can now be made against any data register instead of requiring a dedicated test register. The new user-response instructions allow interactive decision-making by the use of the [YES], [NO], [UNK], [ENT], and [CONT] keys. The TI-88 also has 24 user-defined flags and four system flags for use in decision-making. Decision-making instructions can be concatenated to produce highly sophisticated decision-making structures.

New Advanced Instructions

Access to the 63 hierarchy registers and the presence of advanced instructions allow the user more intimate control over the inner workings of the calculator than ever before. Each of the 16 digits in a hierarchy register can be accessed with recall and store digit commands and each bit within each digit can be set, reset, flipped, or tested.

The program counter can be directly accessed, as well as the subroutine stack, the display, the *AOS* stack and other hierarchy registers. A special "unformatted" display mode allows examination and entry of numbers and program instructions in internal format. These features allow the programmer to use his own creative resources in ways never before possible.

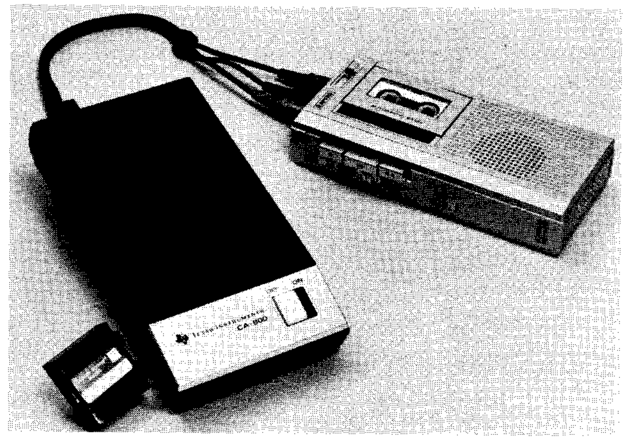
Numerous Special Functions

Over eighty special operation codes allow execution of a multitude of powerful functions ranging from statistics and conversions to peripheral control. Also provided is an "OP" code to display (in English) the definitions of all of the other "OP" codes. These operations provide the professional with the extra edge he needs to quickly and accurately determine solutions to important problems in today's fast-paced world.

PRINTER CONTROL

An important feature of the calculator is its capability to control an optional printer accessory. With a printer connected to the calculator, you can perform the following operations:

- Print the contents of the display at any time.
- Print alphanumeric prompts, operating instructions, and messages.
- Print program input and output.
- Print the contents of program or data memory.
- Print all labels used in a program.
- Print OP code and flag definitions, calculator settings, and alpha entry positions.
- Print tracings of keyboard calculations and program execution showing each function executed with its result.



CASSETTE CONTROL

With the optional CA-800 Cassette Interface accessory and a cassette recorder connected to the calculator, you can record programs and data on standard audio cassette tapes for later retrieval and use. Four types of file recordings can be made:

- You can record all of calculator memory on tape. This type of file allows the contents of program and data memory to be saved together.
- You can record programs on tape. Program files may range in size from a minimum of eight program steps to all of program memory. When needed again, a program file can be read into any part of program memory.
- You can record data on tape. Data files are useful primarily for storing large quantities of information, but may be as short as one data register. When needed again, a data file can be read into any part of data memory.
- You can record the contents of a numbered *Constant Memory* module on tape, freeing the module for other use. When needed again, a module file can be read into a numbered or unnumbered *Constant Memory* module.

SHARE PROGRAMS THROUGH PPX

There may be times when you need a complex specialty program, and you would like the convenience of having a ready-made program that is not a bother to obtain. This is where TI's Professional Program Exchange (PPX) can be of help.

Your yearly PPX membership will open the door to discovery of the many interesting programs written by others in your profession. As an active member, you are a part of a network designed to exchange TI programmable calculator programs within all professions. Using PPX as a vehicle to contribute and obtain programs, you can broaden your professional base while you increase your productivity.

ed?
a. 242000
b. 2.42 05
c. 242 03
d. .0000242
e. 2.42 04

33. Which [OP] code is used to change memory partition?
a. [OP] 16
b. [OP] 17
c. [OP] 18
d. [OP] 19
e. [OP] 11

34. How many flags can be set in a TI-59 programmable calculator?
a. 6
b. 10
c. 11
d. 20
e. Depends upon the partition

35. How many subroutine levels are available in a TI-59?
a. 0
b. 4
c. 7
d. 9
e. None of these

36. After storing 50 in memory 50, you would like to list the values stored in all the data memories. What do you press?
a. [CMS] [LIST]
b. [CLR] [LIST]
c. [CLR] [INV] [LIST]
d. [INV] [LIST]
e. [RCL] [LIST]

37. Which OP codes are used to increment data registers?
a. 10-20
b. 16-17
c. 18-19
d. 20-29
e. 30-39

38. Program memory location 480 is in which bank?
a. 1
b. 2
c. 3
d. 4
e. Depends upon the partition

39. What happens when a [DSZ] (decrement and skip on zero) encounters a zero?
a. The program skips to a predetermined address
b. The next two program steps are skipped
c. The call to go to a predetermined address is skipped
d. The program skips to program location 000
e. The program skips to the next [R/S] or [RTN]

40. In the trace mode "ISM*" represents . . .
a. Inverse sum
b. Subtract the display from a memory using indirect addressing
c. Indirect to a Solid State Software™ module
d. Increment special memory
e. Inverse seconds/minutes

41. How many digits does each alpha print code have?
a. 1
b. 2
c. 3
d. 10
e. 13

43. If flag 1 is set, you can reset it with [INV] [STF] 1. What do you press to reset all the flags at the same time?
a. [CLR]
b. [RST]
c. [CLR] [STF]
d. [INV] [IFF]
e. Cannot be done

44. What is the maximum number of memories with which the [DSZ] instruction can be directly used?
a. 1
b. 9
c. 10
d. 99
e. None of these

45. What instruction momentarily halts execution of a program for display purposes?
a. [PRT]
b. [RST]
c. [INV] [R/S]
d. [PAUSE]
e. Cannot be done

46. Which instruction automatically clears the subroutine return register?
a. [RST]
b. [RTN]
c. [INV] [SBR]
d. [2nd] [CLR]
e. None of these

47. What bank is data memory 28 in?
a. 1
b. 2
c. 3
d. 4
e. Depends upon the partition

48. In order to compare two numbers, which memory or register should be used?
a. "t" register
b. Memory 00
c. Subroutine return register
d. "Y" register
e. Indirect register

49. There is no key labeled $\sqrt[x]{y}$. Keystrokes serving this function are:
a. [2nd] [y^x]
b. [Ind] [y^x]
c. [INV] [y^x]
d. [y^x] [1/x]
e. [2nd] [x^2]

50. If [LBL] [A] is in a program and [A] is pressed ...
a. The program branches directly to location A and begins execution
b. The program branches to location 000, searches for [LBL] [A] and executes the first instruction past [LBL] [A]
c. The program tests the subroutine return register for [LBL] [A] address, branches to that address and executes
d. The program does not execute because we did not press [GTO] [A]
e. None of the above

51. Which of the following is a merged key code?
a. [IxI]
b. [INV] [SBR]
c. [Pause]
d. [CLR]
e. [INV] [GTO]

52. How many data memories can you read on one side of a magnetic card?
a. 30
b. 60
c. 80
d. 100
e. 240

53. Why should you be careful about using [=] in a subroutine?
a. clears the subroutine return register
b. Resets all flags
c. Completes all pending operations
d. Clears the "t" register
e. None of the above

54. Which key stroke sequence discards the fractional part of the number in the display?
a. [INV] [INT]
b. [EE] [INV] [EE]
c. [IxI]
d. [FIX] 0
e. None of the above

55. Which keystroke sequence makes the display register positive?
a. [INV]
b. [EE] [INV] [EE]
c. [IxI]
d. [FIX] 9
e. None of the above

56. Given the program below, what is the correct result after the following execution: Enter 100; press [A]; Enter 50; Press [B]?

[LBL] [A] [X ≷] [R/S] [LBL] [X ≷] [COS] [R/S]
[LBL] [COS] [STO] 05 [R/S]

a. t = 50, R_{05} = 100
b. t = 50, display = 100
c. t = 50, R_{05} = 100
d. t = 100. R_{05} = 50
e. None of the above

57. If you want to clear the program memory, press ...
a. [CLR]
b. [CE]
c. [CP]
d. [CMS]
e. None of the above

58. Which key sequence in a running program acts like [R/S] when not used in a subroutine?
a. [INV] [SBR]
b. [SBR]
c. [INV] [GTO]
d. [Pause]
e. None of the above

59. If you only want to print the displayed value, which key sequence would you use?
a. [OP] 06
b. [PRT]
c. [OP] 07
d. [Trace]
e. None of the above

60. Given the program listing ... [LBL] [A] [STO] 20 [SBR] [IND] 20 [R/S] 1 [RTN] [R/S] ... and the program execution ... Enter 8; Press [A]; What is the correct result after execution?
a. R_{20} = 1, program stopped at 008, 1 displayed
b. R_{20} = 8, program stopped at 007, 1 displayed
c. R_{20} = 8, program stopped at 007, 8 displayed
d. R_{20} = 1, program stopped at 008, 8 displayed
e. None of the above

61. You are in the learn mode, with 005 43 displayed. After you press one or more keys, the display is 005 00. What was pressed?
a. 0
b. [LRN]
c. [DEL]
d. [INS]
e. None of the above

62. A program runs successfully 3 times, but fails on the 4th run. Pressing [RST] permits the program to be run 3 more times, but it again fails on the 4th run. What is wrong?
a. Flags were left set during the previous runs
b. The "t" register is full
c. The program exceeded the subroutine return register's capacity
d. A [GT*] (indirect go to) instruction exceede available program locations
e. A, C, and D only

63. When [+/−] is pressed after [EE]. what happens?
a. Changes the sign of the exponent
b. Changes the sign of the displayed number
c. Changes the sign of both the exponent and the displayed number
d. Changes addition to subtraction or subtraction to addition
e. Depends upon the displayed value

64. Which of the following immediately replaces the displayed value with its functional value?
a. [Y^x]
b. [FIX]
c. [X^2]
d. [÷]
e. [INV]

65. Which of the following keys when pressed is interpreted as a program instruction?
a. [DEL]
b. [LRN]
c. [SST]
d. [GTO]
e. None of the above

66. You have data in memories 00 through 25. You wish to write on a magnetic card. Which is the correct procedure?
a. Enter 1, Press [Write]. Insert card
b. Enter 2, Press [Write]. Insert card
c. Enter 3, Press [Write]. Insert card
d. Enter 4, Press [Write]. Insert card
e. Enter 1, insert card

67. Which pair of the following are *not* inverse functions?

FUNCTION	**INVERSE FUNCTION**
a. [X = t]	[X ≠ t]
b. [DSZ]	skip on nonzero
c. [Eng]	[INV] [EE]
d. [Y^x]	[INV] [Y^x]
e. [Iff]	[INV] [Iff]

68. If you enter a 10 in the display and press [OP] 17, the display reads ...
a. 10
b. 17
c. 100
d. 159.99
e. 160.10

69. Which of the following is *not* true about the [STO] XX function?
a. Stores the display register value into memory register XX
b. Previously stored data is temporarily saved
c. XX is not displayed
d. Displayed register does not change
e. None of the above

70. Which of the following does not involve the "t" register?
a. Data entry with coordinate conversions
b. Repartitioning during program execution
c. Calculates the Y-intercept in a Linear Regression problem
d. Data entry for statistical problems
e. [CP]

71. How could the program instructions [DSZ] 10 [SIN] be keyed-in?
a. [DSZ] [E'] [SIN]
b. [DSZ] 1 0 [SIN]
c. [DSZ] [STO] 10 [BST] [DEL] [SIN]
d. None of the above
e. All of the above

72. Which of the following key sequences assimilates variable pairs [x,y] into data register 01 - 06 during a statistical problem?
a. [IxI]
b. [2nd] [+]
c. [x≷t]
d. [SUM]
e. [Σ +]

73. Which of the following sequences uses a program or common label?
a. [GTO] [A]
b. [GTO] 01 38
c. [RCL] 01 [SIN]
d. [Iff] 1 [COS]
e. None of the above

74. A program which runs properly fails after being modifies to include printing ([OP] 01 through [OP] 05). What is wrong?
a. Failed to protect intermediate values when entering print codes
b. Loading print registers wrote over pending calculations
c. The [OP] 05 function truncated the displayed value
d. All of the above
e. Only A and C above

75. Which of the [OP] codes only print the characters loaded by [OP] 04 and the display?
a. 03
b. 05
c. 06
d. 08
e. 19

76. For which of the following purposes are the data memories *not* used?
a. A "scratch pad" for calculations
b. Storage of flag status
c. Storage of variable data
d. Storage of constant data
e. All of the above

77. Looking at the keyboard of the TI-59 Programmable Calculator, which key has key code 57?
a. [RCL]
b. [ENG]
c. [(]
d. [EE]
e. [−]

78. For later prompting, alpha character codes can be stored in ...
a. Data registers
b. Program steps
c. Solid State Software™
d. The "t" register
e. Only a,b, and d above

79. In a running program, what is the difference between [A] and [GTO] [A]
a. There is no difference
b. [A] is a label address; [GTO] [A] is an absolute address

c. [A] acts as a subroutine call; [GTO] [A] does not
d. [A] is an absolute address
e. [A] branches the program and continues operation; [GTO] [A] branches the program to [A], but then stops the program from running

80. The memory partition has been adjusted to 70 data memories. How many program instructions can you put in the program memory?
a. 279 d. 400
b. 280 e. Variable
c. 399

81. There is data located at register 94 and 95. In order to write that data on a magnetic card, what bank number must be displayed?
a. 0 d. 3
b. 1 e. 4
c. 2

82. You are about to remove and replace the Solid State Software™ module. What should you always do first?
a. Check the diagnostic
b. Read User Instructions for Program 01
c. Turn the calculator off, then on
d. Open the module compartment
e. Discharge any static electricity by grounding yourself

83. For no apparent reason the reader/writer motor in the calculator turns on. What is a possible cause of the problem?
a. Pressed the [R/S] Key
b. Bad Solid State Software module
c. You got question #82 wrong
d. Pressed 2nd, then [Write]
e. This cannot happen

84. The partition is correct for a program you are trying to read from magnetic card. You enter a 2 in the display and pass the card through. The display is flashing 3. What is the problem?
a. Entered a "2" but read in side 3
b. Forgot to press [Write]
c. Tried to read a protected program
d. Bad magnetic card
e. Forgot to press [INV] [Write]

85. Which of the following is a function of the PC-100A(C)?
a. Charges batteries
b. Lists Programs and data
c. Secures handheld calculators
d. Prints up to 20 characters on one line
e. All of the above

86. Which of these keys are conversion keys?
a. [cos] d. [P/R]
b. [D.MS] e. All of these
c. [INV] [sin]

87. I take a black card from my card case and read it, an error is evident. What is my problem?
a. Read wrong side of card d. Still in fixed decimal mode
b. The card is plastic e. None of the above
c. Partition is incorrect

88. What is the highest data memory in which you can store a variable in a TI-59?
a. 30 d. 99
b. 59 e. 479
c. 90

89. What would the following Program be used for?

000 [LBL]	006 [LBL]	013 [+]
001 [E]	007 [A]	014 [1]
002 [1]	008 [ST*]	015 [=]
003 [STO]	009 [00]	016 [STO]
004 [00]	010 [RCL]	017 [00]
005 [R/S]	012 [00]	018 [R/S]

a. Stores a "1" in consecutive data locations
b. Stores several variables using one User Defined Key
c. Does not work because we did not SUM into 00
d. Adds one to everything we store in data memories
e. Decreases stored numbers by one each time

90. How often should I clean the PC-100A(C) printhead?
a. Every morning or evening
b. Whenever it does not print clearly
c. After each roll of paper
d. Cleaning the printhead will void the guarantee
e. After each use

91. After you perform the following operations, what value remains in the display?
[3] [1/X] [x] [1/X] [x] [3] [=] [Int]
a. 0 d. 2.
b. 0.333333333 e. 3.
c. 1.

92. How many digits are in the standard TI-59 display?
a. 10 d. 13
b. 11 e. 20
c. 12

93. Which is the correct procedure for clearing all memories, flags, and special registers?
a. Remove the batteries
b. [RST], [CLR]
c. [RST], [CLR], [CE]
d. Turn the calculator off.
e. [CP], [CLR], [CMs], [RST], [CE]

94. You wish to print "STOP." After keying in and storing the correct print codes (OP 01), the OP 05 call prints "2/!HM." What is wrong?
a. The print register is full
b. Printed the display value
c. Calculator is in an oddnumber fixed mode
d. Pending calculations have interfered with the operation
e. None of the above

95. If there is 347.2 displayed and Op Code 10 is pressed, what is displayed?
a. 347 d. 0
b. 347.2 e. 347
c. 1

96. Which of these are an exception to the AOS™ (Algebraic Operational System)?
a. 30 y^x 5 + 24300000. =48600000.
b. 5 x 5 - 4 = 5
c. 30 sin = 0.5
d. 8 [X^2] = 64
e. They are all correct as written

97. The key sequence [Pgm] [0] [1] [SBR] [=] gives you a "1" in the display. What can you conclude from that?
a. Nothing
b. The Master Library Module is installed and working
c. The Math/Utility Library Module is installed and working
d. Magnetic Card side 1 should be read
e. Error 1 should be referred to

98. What is the effect of [INV] preceeding [x≷t]?
a. Both calls are ignored
b. x is stored in "t", contents of "t" register appears in display
c. [x≷t] becomes inoperative
d. The display flashes
e. Depends on current partition

99. If a [Nop] is encountered in a program, the following happens:
   a. Program stops
   b. Sets flag 7
   c. Resets all flags
   d. Clears all pending operations
   e. Disregards [Nop] and proceeds

100. A-E' are ...
   a. Alpha prompting keys
   b. Common labels
   c. Program labels
   d. DSZ keys
   e. User defined keys

continued on page 11

There's Gold *(cont'd from page 3)*

also illustrates the use of a newly discovered, more versatile method for fast mode entry.

Appendix C of *Personal Programming* lists several cautions on the accuracy of the trigonometric functions, and of the powers and roots functions. Except for limited areas of concern the guard digits seem to have been efficiently used by the built-in algorithms of the TI-58/59. For example, the square root solutions for the first twenty-five prime numbers are correct for all thirteen digits in a truncated sense. Similarly, the natural logarithm (1n) of the first twenty-five prime numbers are correct for all thirteen digits, but in a rounded sense. The natural logarithm function does lose accuracy for entries very near to one. In the June 1977 issue of *52 Notes* Joel Pitcairn and Barbara Osofsky reported that 1n(1.0000007) which is calculated as 0.0000006999999 is correct to only six significant figures. The solution is correct for twelve digits to the right of the decimal point. Joel Pitcairin also found that sin(0.00007) was correct to only seven significant figures; again, the solution is correct to twelve digits to the right of the decimal point. Although those results were reported for the SR-52, the same accuracies hold for the TI-59. Even earlier, Larry Mayhew reported difficulties for the sine function in the neighborhood of ninety degree inputs; for example, sin(89.9999999954) = 1.000000000004. Again, this error occurs with both the Sr-52 and the TI-59. The cautions in Appendices B and C of *Personal Programming* clearly apply, particularly if your program can be expected to find logarithms of numbers near one, or to find trigonometric functions at inputs near multiples of ninety degrees. So, as noted in *Personal Programming*, when in doubt, round using the EE-INV-EE sequence.

Appendix C of *Personal Programming* also states that for the trigonometric functions "All displayed digits in standard display format are accurate to +/-1 in the 10th digit for a+36,000 degree range, +200 pi radians, and a+40,000 grads ...". For special cases, the ranges for accurate results may be much greater. A recent programming challenger in *TI PPC Notes* asked for efficient solutions for powers of minus one. That is, given an integer in the display, finds minus one to that power. Defining the problem more carefully, the output should be *exactly* plus one for even integer inputs, and *exactly* minus one for odd integer inputs. One class of solutions used a special case of deMoivre's theorem,

$$(\cos 0)^n = \cos(n0)$$

When applying the limitation defined in *Personal Programming*, a solution of the form

LBL A DEG x 180) COS RTN

might be expected to fail for input integers as small as 200. Actually, for input integers smaller than 2,000,005 not only will the display indicate the desired value, but the display register contents will be correct. At inputs of 2,000,005 and above the display will be correct, but the display register will not. The range of allowable integers can be extended by using an OP 10 command to "round" the results after the cosine function, i.e.,

LBL A DEG x 180) COS OP 10 RTN

The solution will then yield exactly plus one or minus one for integers over the full ten digit range of the display, and for thirteen digit entries as large as 2,000,000,000,000. But for the input integer 2,000,000,000,001 which can be synthesized in the display register with the sequence

2,000,000,000 x 1,000 + 1 =

and all larger odd integer inputs the function will yield a plus one rather than the desired minus one.

In an attempt to extend the range of input integers even further, I decided to try using indirect flag tests. The idea comes from the article "Flag Sorting" in the January 1979 issue of *PPX Exchange*. The author, William Bowman, stated "When performing the IF FLAG INDIRECT XX instruction, the TI-59 only recognizes the first digit to the left of the decimal point of the number stored in the indirect register XX. For example, if 96124.03129 is stored in indirect register XX, *IFF*IND XX tests for flag 4 ...". Of course, use of this technique will effectively limit use of the routine to calculator only, that is without the printer unless one is willing to accept the automatic entry into TRACE mode whenever flag 9 is used. The routine I devised was

STF IND 00 9 STO OO OP 30 IFF IND OO 019 DSZ 0 006
0 - 1 = RTN LBL A x STO 00 RST

To my surprise this routine gave erratic results for input integers greater than 999,999,999,999. The output didn't seem to have any rational relationship to whether the input integer was even or odd, positive or negative. Eventually, I stumbled on an unpublished quirk of the TI-59. The STF IND XX command sequence sets a flag according to the ones digit if the value in register XX is an integer of twelve digits or less. If the input integer is a thirteen digit value then the STF IND XX function sets a flag on the basis of the tens digit! There may be some use for this quirk. In this case the quirk defeated my attempt to extend the range of input integers for a powers of minus routine. The experience also reinforces the cautions in *Personal Programming* concerning the use of the guard digits.

For those who may be interested, a routine which will extend the range of a powers of minus one routine to the full range of thirteen digit integers is

LBL A IxI - (CE ÷ 2) INT x 2 = x 2 - 1 = + / RTN

where the reader will recognize another use of the truncation characteristics of the display register.

ANSWERS TO TI-59 TEST

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

TI-59 Test Listing

000	98	ADV
001	98	ADV
002	98	ADV
003	36	PGM
004	01	01
005	71	SBR
006	25	CLR
007	48	EXC
008	00	00
009	92	RTN
010	81	RST
011	76	LBL
012	19	D'
013	22	INV
014	58	FIX
015	73	RC*
016	02	02
017	34	ΓX
018	38	SIN
019	52	EE
020	22	INV
021	52	EE
022	92	RTN
023	76	LBL
024	15	E
025	69	OP
026	20	20
027	76	LBL
028	14	D
029	69	OP
030	20	20
031	76	LBL
032	13	C
033	69	OP
034	20	20
035	76	LBL
036	12	B
037	69	OP
038	20	20
039	76	LBL
040	11	A
041	69	OP
042	20	20
043	22	INV
044	52	EE
045	50	I×I
046	48	EXC
047	00	00
048	32	X:T
049	97	DSZ
050	00	00
051	51	BST
052	43	RCL
053	00	00
054	55	÷
055	05	5
056	00	0
057	42	STO
058	02	02
059	01	1
060	00	0
061	75	-
062	59	INT
063	44	SUM
064	02	02
065	95	=
066	65	×
067	01	1
068	00	0
069	85	+
070	01	1
071	95	=
072	59	INT
073	22	INV
074	28	LOG
075	88	DMS
076	42	STO
077	03	03
078	65	×
079	19	D'
080	95	=
081	75	-
082	22	INV
083	59	INT
084	42	STO
085	04	04
086	95	=
087	55	÷
088	01	1
089	00	0
090	95	=
091	59	INT
092	65	×
093	01	1
094	00	0
095	85	+
096	32	X:T
097	85	+
098	43	RCL
099	04	04
100	95	=
101	55	÷
102	43	RCL
103	03	03
104	95	=
105	22	INV
106	38	SIN
107	33	X²
108	72	ST*
109	02	02
110	02	2
111	44	SUM
112	00	00
113	61	GTO
114	00	00
115	03	03
116	76	LBL
117	95	=
118	18	C'
119	71	SBR
120	40	IND
121	01	01
122	32	X:T
123	25	CLR
124	19	D'
125	67	EQ
126	02	02
127	35	35
128	87	IFF
129	01	01
130	01	01
131	80	80
132	86	STF
133	01	01
134	02	2
135	04	4
136	03	3
137	01	1
138	01	1
139	05	5
140	69	OP
141	01	01
142	03	3
143	02	2
144	03	3
145	05	5
146	03	3
147	05	5
148	01	1
149	07	7
150	01	1
151	05	5
152	69	OP
153	02	02
154	03	3
155	07	7
156	08	8
157	00	0
158	01	1
159	03	3
160	03	3
161	01	1
162	03	3
163	06	6
164	69	OP
165	03	03
166	04	4
167	03	3
168	01	1
169	07	7
170	03	3
171	05	5
172	03	3
173	06	6
174	08	8
175	00	0
176	69	OP
177	04	04
178	69	OP
179	05	05
180	32	X:T
181	42	STO
182	03	03
183	19	D'
184	42	STO
185	06	06
186	01	1
187	00	0
188	49	PRD
189	03	03
190	49	PRD
191	06	06
192	29	CP
193	43	RCL
194	03	03
195	67	EQ
196	02	02
197	41	41
198	59	INT
199	22	INV
200	44	SUM
201	03	03
202	32	X:T
203	43	RCL
204	06	06
205	59	INT
206	22	INV
207	44	SUM
208	06	06
209	67	EQ
210	02	02
211	28	28
212	43	RCL
213	04	04
214	85	+
215	32	X:T
216	82	HIR
217	18	18
218	95	=
219	32	X:T
220	22	INV
221	67	EQ
222	02	02
223	26	26
224	66	PAU
225	66	PAU
226	99	PRT
227	76	LBL
228	69	OP
229	20	CLR
230	69	OP
231	24	24
232	61	GTO
233	01	01
234	86	86
235	01	1
236	00	0
237	44	SUM
238	00	00
239	44	SUM
240	04	04
241	01	1
242	00	0
243	44	SUM
244	01	01
245	69	OP
246	22	22
247	97	DSZ
248	05	05
249	01	01
250	19	19
251	98	ADV
252	69	OP
253	00	00
254	04	4
255	05	5
256	03	3
257	02	2
258	04	4
259	01	1
260	03	3
261	05	5
262	08	8
263	00	0
264	69	OP
265	02	02
266	03	3
267	06	6
268	01	1
269	05	5
270	03	3
271	02	2
272	03	3
273	05	5
274	01	1
275	07	7
276	69	OP
277	03	03
278	69	OP
279	05	05
280	58	FIX
281	06	06
282	06	6
283	01	1
284	69	OP
285	04	04
286	22	INV
287	58	FIX
288	43	RCL
289	00	00
290	69	OP
291	06	06
292	81	RST
293	76	LBL
294	18	C'
295	98	ADV
296	01	1
297	42	STO
298	04	04
299	00	0
300	42	STO
301	00	00
302	03	3
303	07	7
304	01	1
305	42	STO
306	01	01
307	05	5
308	00	0
309	42	STO
310	02	02
311	01	1
312	00	0
313	42	STO
314	05	05
315	92	RTN
316	76	LBL
317	90	LST
318	18	C'
319	01	1
320	00	0
321	42	STO
322	04	04
323	19	D'
324	42	STO
325	06	06
326	01	1
327	00	0
328	49	PRD
329	06	06
330	43	RCL
331	06	06
332	69	OP
333	20	20
334	59	INT
335	22	INV
336	44	SUM
337	06	06
338	67	EQ
339	03	03
340	44	44
341	75	-
342	08	8
343	85	+
344	02	2
345	00	0
346	95	=
347	58	FIX
348	06	06
349	69	OP
350	04	04
351	22	INV
352	58	FIX
353	43	RCL
354	00	00
355	69	OP
356	06	06
357	97	DSZ
358	04	04
359	03	03
360	26	26
361	69	OP
362	22	22
363	97	DSZ
364	05	05
365	03	03
366	19	19
367	00	0
368	42	STO
369	00	00
370	81	RST
371	02	2
372	04	4
373	01	1
374	04	4
375	03	3
376	05	5
377	02	2
378	03	3
379	02	2
380	02	2
381	04	4
382	01	1
383	03	3
384	05	5
385	02	2
386	02	2
387	04	4
388	01	1
389	02	2
390	03	3
391	04	4
392	01	1
393	05	5
394	03	3
395	03	3
396	04	4
397	05	5
398	01	1
399	02	2
400	04	4
401	01	1
402	03	3
403	02	2
404	02	2
405	05	5
406	03	3
407	04	4
408	03	3
409	03	3
410	01	1
411	02	2
412	02	2
413	02	2
414	04	4
415	04	4
416	01	1
417	04	4
418	01	1
419	03	3
420	02	2
421	02	2
422	01	1
423	03	3
424	05	5
425	03	3
426	05	5
427	03	3
428	01	1
429	02	2
430	05	5
431	04	4
432	03	3
433	05	5
434	03	3
435	04	4
436	04	4
437	03	3
438	04	4
439	02	2
440	02	2
441	01	1
442	05	5
443	04	4
444	04	4
445	03	3
446	02	2
447	02	2
448	05	5
449	03	3
450	04	4
451	02	2
452	05	5
453	03	3
454	01	1
455	05	5
456	05	5
457	02	2
458	04	4
459	02	2
460	03	3
461	04	4
462	01	1
463	04	4
464	03	3
465	03	3
466	02	2
467	02	2
468	02	2
469	05	5
470	05	5
471	55	÷
472	01	1
473	52	EE
474	01	1
475	00	0
476	95	=
477	92	RTN

TI-59 Programming Seminars

There may be a seminar coming to your area. These seminars are open to anyone with a TI-59 regardless of programming background. The seminars provide both beginning and intermediate programming training on the TI-59 in a "hands on" fashion. Tuition for the two day class is $150.00 per person. This includes the instruction, workbook, and luncheon for the two days. You should supply your own TI-59.

To register send your check for $150.00 payable to Texas Instruments to:

TI-59 Seminar
Texas Instruments
P.O. Box 10508 M/S 5873
Lubbock, TX 79408

If you have any further questions regarding the seminars or if you would like information on setting up a company seminar, please contact Mary Ann Barley at 806—741-2202. The schedule for upcoming seminars is listed below.

SEMINAR DATES	LOCATION
May 13-14	Columbia
May 18-19	Hartford
May 27-28	Charlotte
June 8-9	Dallas
June 14-15	Oklahoma
June 17-18	Albuquerque
June 24-25	Lubbock
June 28-29	San Antonio
July 13-14	Houston
July 22-23	Phoenix
July 26-27	El Paso
July 29-30	Memphis
August 5-6	Miami
August 12-13	Omaha
August 19-20	Houston
August 23-24	Pittsburg
August 29-30	Pittsburg
September 2-3	Cincinnati

The PPX Exchange is published bimonthly and is the only newsletter published by Texas Instruments for TI-59 owners. Members are invited to contribute articles and items of general interest to other TI-59 users. Authors of accepted feature articles for the newsletter will receive their choice of either a one year complimentary PPX membership or a Solid State Software™ module. Please double-space and type all submissions, and forward them to:

Texas Instruments, PPX
P.O. Box 53
Lubbock, Texas 79408
Attn: PPX Exchange Editor

Membership Renewals

Is your membership about to expire? To ensure that you will miss no newsletters, catalogs, or ordering privileges, check the renewal table to find out if your membership will expire soon. (If your number is not included in the range of the table, it is not time for you to renew). The next issues of the Exchange will list additional renewal dates.

A renewal card and reminder will be sent to each member before the time to renew. Return the card promptly to PPX with your check or money order for $20.00. Please do not procrastinate in returning your renewal material as our membership coordinator must remove delinquent members from our computer listing. Be sure to include your membership number on both your card and your check and mail to: Texas Instruments PPX Department, P.O. Box 109, Lubbock, TX 79408.

Membership Number	Renewal Date:
907526-908257	July 31
918474-919157	July 31
926499-927039	July 31
931520-931650	July 31
908258-909339	August 31
919158-920093	August 31
927040-927572	August 31
931798-932167	August 31

TEXAS INSTRUMENTS
INCORPORATED
PPX • P.O. Box 53 • Lubbock, Texas 79408
U.S. CALCULATOR PRODUCTS DIVISION

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. POSTAGE
PAID
Permit No. 1
Lubbock, Texas