PPX Exchange

Vol. 4 Number 3 Copyright 1980 May/June 1980

Attention: Programming Buffs

PPX member Jeff Jones has submitted a challenge for PPX members to write a program that will generate the print code table using a PC-100 A/C (as seen on page VI-7 of the Personal Programming manual). The program should be designed to meet the following specifications:

A. Begin in the power-up partition (479.59).
B. Leave the calculator in the power-up partition when program execution is finished.
C. Make use of a minimum number of program steps and pre-stored data registers.
D. Fit on one magnetic card side.
E. Generate the table as fast as possible.

We will feature the best program received in our September/October issue of the Exchange. To allow us time to compare programs, please mail all print code programs by August 1, 1980. Just to give you an idea of the magnitude of this project, the program that Mr. Jones submitted with his challenge contains only 140 program steps, requires 10 pre-stored data registers, and has a run time of 38 seconds. With this in mind, gentlemen, start your pencils.

BENEATH THE PLANET OF THE FLAGS

or, Flags Revisited

By Robert Wyer

As with any field of endeavor, there are factions or schools of thought which will polarize themselves around any given subject. This seems to be especially true of programming. Some programmers avoid the use of flags at any cost and others find them to be a very useful tool. It is to the latter of these that this letter is submitted.

When using flags to indicate various conditions, a time may arise when 10 flags are not enough. Well, fear not, for there are methods, or as programmers call them, 'techniques' by which the number of flags available to the user may be increased to a virtually limitless, plethoric number.

• The first technique is to use a combination of flags to indicate various conditions. By this method n number of flags may signal 2^n different conditions. The example below uses three flags to show 2^3 or 8 different conditions (an optimality discussion on the example will not be engaged for fear of creating another faction). The example simply allows the user to initiate one of 8 conditions by the labels

continued on page 2

Hierarchy Strikes Again

Editor's note: Hierarchy (HIR, code 82), unlike lightning is striking the same place twice. Basic information on this code was given in the May 1978 issue of the PPX Exchange. The following article reviews the information in that article and introduces some of the limitations of using the HIR code.

The TI-59 contains eight internal memory registers which were not mentioned in the Personal Programming Manual. These registers are the hierarchy registers. They were designed to temporarily store operands during execution of arithmetic and for internal use by certain programmed functions (Σ +, $\bar{x}$, OP 11-15, P→R and D.MS). Each time a pending operation is called upon, whether the result of the use of parentheses or of the Algebraic Operating System entry method (AOS), one more HIR register is used.

AOS is the method that your TI-59 was equipped with so that there is only one answer to a given computation which uses mixed operations. For example, 3 + 10 – 2 x 14 ÷ 7 = ?, has 9 as it one and only answer. Below is the same example in stepwise form with a comment about what is happening internally in the calculator:

ENTER/PRESS	DISPLAY	COMMENTS
3 +	3	3 is stored in HIR 1
10	10	10 in display register
–	13	13 is computed and stored in HIR 1
2 x	2	2 is stored in HIR 2
14 ÷	28	Result of 2 x 14 is stored in HIR 2.
7	7	7 in display register
=	9	All computations completed. Result is in display register.

Note that a number is not placed into a HIR register until an operator is pressed (e.g., 3 by itself is in the display register, 3 followed by + forces the 3 into a hierarchy register).

The hierarchy registers can be accessed and manipulated through program memory by the use of the HIR command, code 82. This command cannot be directly keyed in, but may be written by pressing STO 82, backstepping and deleting the STO. There is a two-digit XY number which follows the 82 command. X stands for one

continued on page 3

FLAGS

A, B, C, D, E, A', B', and C'. SBR x^2 interprets which routine was chosen and responds numerically by displaying 0-7 corresponding respectively to the labels above.

Simple or complex routines may be substituted for the numeric output. By this method, up to 1024 "flags" may generated when using all ten flags.

000	76	LBL	026	86	STF	052	02	02	078	01	01
001	11	A	027	01	01	053	52	EE	079	35	1/X
002	22	INV	028	61	GTO	054	00	0	080	87	IFF
003	76	LBL	029	12	B	055	11	A	081	02	02
004	15	E	030	76	LBL	056	92	RTN	082	44	SUM
005	86	STF	031	18	C'	057	76	LBL	083	04	4
006	00	00	032	86	STF	058	52	EE	084	11	A
007	22	INV	033	00	00	059	01	1	085	92	RTN
008	76	LBL	034	61	GTO	060	11	A	086	76	LBL
009	13	C	035	14	D	061	92	RTN	087	44	SUM
010	86	STF	036	76	LBL	062	76	LBL	088	05	5
011	01	01	037	16	A'	063	42	STO	089	11	A
012	22	INV	038	86	STF	064	87	IFF	090	92	RTN
013	76	LBL	039	00	00	065	02	02	091	76	LBL
014	12	B	040	61	GTO	066	45	Y^x	092	35	1/X
015	86	STF	041	12	B	067	02	2	093	87	IFF
016	02	02	042	00	0	068	11	A	094	02	02
017	92	RTN	043	76	LBL	069	92	RTN	095	43	RCL
018	76	LBL	044	33	X²	070	76	LBL	096	06	6
019	17	B'	045	87	IFF	071	45	Y^x	097	11	A
020	86	STF	046	00	00	072	03	3	098	92	RTN
021	00	00	047	34	FX	073	11	A	099	76	LBL
022	61	GTO	048	87	IFF	074	92	RTN	100	43	RCL
023	13	C	049	01	01	075	76	LBL	101	07	7
024	76	LBL	050	42	STO	076	34	FX	102	11	A
025	14	D	051	87	IFF	077	87	IFF	103	92	RTN

• The other commonly used technique is to use data registers as flags. Each of the 13 digits (including guard digits) may represent a flag 0-9, thus giving 130 "flags." Or, by using the technique above—well, the actual number is left as an exercise. For the sake of convenience, the example will consider the need of only one extra "flag." The "flag" will be set if a non-zero value (in this case 1) exists in register 00. Pressing A resets the flag and B sets the flag. As before SBR x^2 interprets the flag arrangement and displays 0 if the flag was not set or 1 if the flag was set. Again, routines may replace the numeric output in the interpretation routine.

000	76	LBL	007	12	B	014	29	CP	021	76	LBL
001	11	A	008	01	1	015	43	RCL	022	16	A'
002	00	0	009	42	STO	016	00	00	023	01	1
003	42	STO	010	00	00	017	67	EQ	024	92	RTN
004	00	00	011	92	RTN	018	16	A'			
005	92	RTN	012	76	LBL	019	00	0			
006	76	LBL	013	33	X²	020	92	RTN			

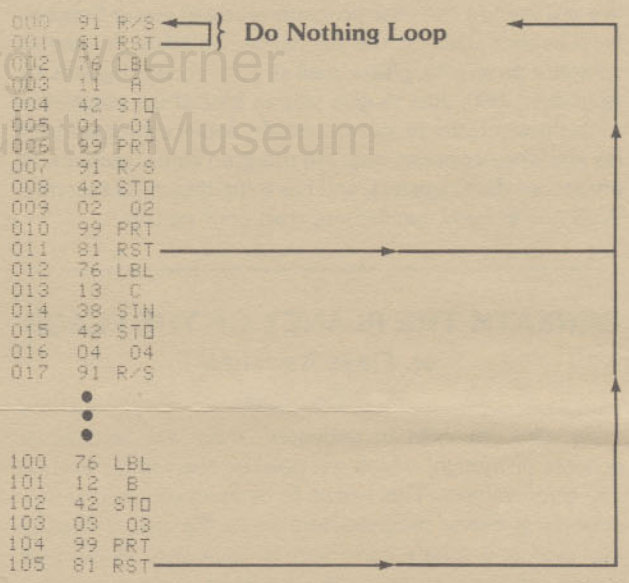
Note in the second example that CP is used in the interpretation routine. This insures that a zero is located in the t-register. This particular point reinforces a good programming technique of "knowing what the conditions are before testing." This is not to allude to the point that if the conditions are known, why test? The fact is that the calculator has the "audacity" to allow "bugs" to creep into programs. In the first example, if all the flags were not reset before the toggling sequence (labels A-C') a faulty reaction upon interpretation may occur. Thus, to prevent programmer embarrassment by a non-responsive, intelligent calculator, insuring that conditions are in a "preset" mode before setting up a "toggle" sequence is a good practice.

Hopefully, these methods will be of some use to that school which prefers or generally engages in the use of flags. For that other school of thought, this may be filed under the heading of useless dribble. But, you never know.

Using "Do-Nothing" Loops

Aubrey B. Hutchison, of Pompano Beach, Florida, sent PPX the following idea of a "do nothing loop" which will reduce operator error for user programs that use R/S or user defined keys: In those cases where termination of a user definable routine is made by the use of RTN or R/S, an incorrect extra operation by the user of R/S key will advance the calculator to the next sequence of instructions. In most cases, this causes irrecoverable errors to be made.

A "do nothing loop" can be used to prevent this type of error. It consists of only two steps: R/S, RST. For example: consider the simple program given below:



For the program above, suppose a given list of variables had to be entered in the following manner: Enter time, press A; enter velocity, press R/S; enter distance, press B; enter acceleration and execute program, press C. If the RST at location 011 was a R/S and if R/S was pressed after entering distance instead of label B, the sine of the velocity would be incorrectly computed and stored into 04.

For those members who have printers, an alpha numeric message could be inserted at location 000 so that a warning could be printed.

HIERARCHY

of the following HIR register operations:

X	OPERATION
0	STO
1	RCL
3	SUM
4	Prd
5	INV SUM
6-9	INV Prd

Y stands for the hierarchy register to be accessed (1-8). XY may be entered in the same manner as code 82 if XY by itself is an invalid keyboard entry.

The following sequence will be used to demonstrate the HIR command. Key in LRN mode (location 000): LBL A 4 x (3 – 2 x (6 – 1 x (8 – 7 x (5 – 9 INV SBR.

Execution of this sequence will cause all eight hierarchy registers to be filled. The 4 will be placed in the first hierarchy register, 3 in the second, 2 in the third, ..., 5 in the eighth, and 9 in the display register. At location 024, key in: LBL B A HIR 13 = R/S. Press B, the following will be executed: 4 x (3 – 2 x (6 – 1 x (8 – 7 x (5 – 9 = –140). HIR 13 recalls the contents (2) of hierarchy register 3 and places it in the display register (i.e., replacing the 9 in the above sequence, LBL A).

Although hierarchy can be useful, it must be used with some caution. This is because, as mentioned before, some of the hard-wired functions of the TI-59 use the hierarchy registers:

	HIR #7	HIR #8	HIRs FOR PEND. OPS	TOTAL HIRs USED
$\Sigma+$	✓	✓	0	2
INV $\Sigma+$	✓	✓	0	2
$\bar{X}$			1	1
INV $\bar{X}$		✓	2	3
OP 11			2	2
OP 15		✓	3	4
OP 14		✓	3	4
OP 13			4	4
OP 12		✓	3	4
P→R	✓	✓	1	3
INV P→R	✓	✓	2	4
D.MS		✓	2	3
INV D.MS		✓	2	3

The functions that specifically require hierarchy registers 7 and 8 have checkmarks in the first two columns of the table. The third column shows the number of HIR registers needed to store the pending operations. The HIRs used start with the lowest available hierarchy register and follow sequentially as needed. The lowest available depends upon the number of pending operations at the time the function is called. For example, note the key sequence: 5 + 6.2 D.MS. The number 5 is stored as a pending operation in HIR register 1 after the + is pressed. This causes the D.MS function to use HIR registers 2 and 3. D.MS also uses HIR register 8 as shown in the table.

NOTES:

1) OP 01 through 04 use HIR registers 5 through 8.
2) CMS, CLR, and CE will not clear the HIRs, but OP 00 will clear HIRs 5 through 8.
3) Negative exponent numbers are handled incorrectly unless the number in the display register is in the EE (or Eng) format.
4) If any operation (STO, SUM, Prd, INV SUM, INV Prd) is performed into a pending operations register that is holding a subtraction, the operation will be changed to addition. A similar fate happens if the register is holding a division and an operation is performed on the register: the ÷ is changed to a x.

potpourri

1. **New Format.** The black ink in which our last issue was printed received such an enthusiastic response that we decided to make the change permanent. Beginning with this issue, we are expanding to an 8-page format to allow the use of more feature articles and our new column, "Précis," which will present abstracts of programs which have been added to the PPX program library since issuance of the last addendum. These programs are now available for purchase. Our regular features, "From the Analyst's Desk," "Potpourri," and "Programming Corner," will appear under distinctive heads in each issue so they can be easily located. We hope these changes will enhance your enjoyment of the newsletter, and welcome any comments you may have.

2. Help us serve you—by making sure that all your correspondence/program orders reach PPX. Besides writing the correct address on your envelope (P.O. Box 53, Lubbock, Texas 79408), be sure to address your letters to Texas Instruments PPX. This is necessary because PPX shares P.O. Box 53 with other TI departments. Since the mail has to be separated, there is a chance that your letter will get side-tracked if it is not addressed to PPX.

Also be sure to make your checks payable to PPX. This will insure that if your order does get side-tracked, you will still get proper credit for your payment.

3. **Going Up:** Unfortunately, PPX is not immune to the inflation bug. We've been bitten, and the only cure is to raise the price on our $35 libraries to $40 (still a great deal, as the average library would cost about $57 for documentation alone if the programs were bought separately). This increase will be effective on orders received after August 1.

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. Please limit your submissions to four double-spaced typed pages, and forward them to:

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

REGISTER OPERATIONS

by

George Vogel

When it comes to shifting or otherwise processing large blocks of stored data, happiness is a program that will do it for you quickly and reliably. The one below will do everything from shifting a block of data to dividing one block of registers by another. In addition to these features, this program also allows the user to process data by using a custom subroutine. This added feature greatly enhances the usefulness of the program, which is limited only by your imagination.

USER INSTRUCTIONS:

1. Partition to 239. 89 by pressing 9 Op 17.
2. Enter program. When keying program in, note that there are three keystrokes which are entered by using special techniques:
   a. HIR xx—enter STO 82 STO xx (where xx is the two digit number following the HIR keystroke), BST and Del the two STO's. Then, SST past the two locations containing the HIR xx instruction.
   b. Dsz 89 115 (Locations 147 through 150)—press STO 89 GTO 115, backstep to the STO keystroke (code 42) and press Dsz. Then delete the GTO instruction (code 61). SST to location 151.
   c. Lbl SBR LRN—enter Lbl SBR STO 31 and delete the STO. SST to location 201.
3. Press SBR CLR to initalize program. (SBR CLR can also be used to cancel a wrong move. See instruction number 5 below.)
4. Press the user-defined key for the desired operation. Given below are the various operations and the user-defined key that will initiate them. Note that a...a' refers to block of registers a through a'; b...b' refers to the same number of registers from register b through b'; c is a numerical constant.

KEY	OPERATION	ENTER & PRESS R/S AFTER EACH:
A	Shift block a...a' to b...b'. This operation shifts either direction equally well (the two blocks may overlap or coincide).	a, a', b
B	Exchange data in registers a...a' with registers b...b'.	a, a', b
C	Invert order of data within block a...a'.	a, a'
D	Add constant c to a...a'. (subtract if c is followed by +/-.)	a, a', c
E	Multiply a...a' by constant c. (divides if c is followed by 1/x.)	a, a', c
A'	Add a...a' to b...b'.	a, a', b
B'	Subtract a...a' from b...b'.	a, a', b
C	Multiply b...b' by a...a'.	a, a', b
D'	Divide b...b' by a...a'.	a, a', b
E'	Activate custom subroutine to convert data a; from a...a' into $f(a_i)$.	

Operating notes:

- For operations A' through D', a...a' operates on b...b' and the result is stored in b...b'. Obviously, enough room must be allowed for register block b...b'.
- Registers 0 to 84 are available for data storage; 85 to 89 are needed by the program. CMs clears all registers; to clear only from a to a': Press E and then enter a, a' and 0 by pressing R/S after each entry.
- The program always handles block a...a' from the left to right except, of necessity, when shifting to an overlapping block b...b' to the right (i.e., when $a < b \leq a'$); however, the latter will have visible consequences only for output (e.g., Prt) via the custom subroutine, which will then be from a' to a. It is of no consequence for the data in registers.

5. Error recovery: As long as execution has not started, SBR CLR and start again. When the program is already running, it is usually best to let it run its course since most operations can be reversed.

Custom Subroutines:

As noted above, the ability to input your own subroutine is a principle feature of this program. It allows data a_i to be converted to processed data $f(a_i)$. The processed data can be displayed, shifted or added to another block, subtracted from or exchanged with another block, etc. To exploit the infinite number of possibilities, one only has to remember the following: Upon arriving at the custom subroutine, the display contains just-recalled a_i. This is available again, if required, by RC* 88, and the number of the a register, by RCL 88; the corresponding b_i (if appropriate), by RC* 86, and the number of the b register, by RCL 86. The subroutine must be activated (E') each time before an operation is selected, otherwise it is bypassed (in operation C, it is by-

passed in any case). Remember that the subroutine will cause $f(a_i)$ instead of a_i to operate on its counterpart in the b block; where no b block is involved (D, E), the subroutine naturally has no effect on the stored data, offering further flexibility. For example $f(a_i)$ may be output via operation E, and depending on whether c = 1 or 0, the data in a...a' will be retained or cleared. By use of operation A, by contrast, each a_i in a...a' can be changed into $f(a_i)$.

The following examples will give you some idea of the power of this program:

- Enter external data sequentially into a...a' by pressing SBR SBR R/S RTN (INV SBR) LRN. To use, press E', A, then enter a, a', a and the data by pressing R/S after each entry.

Example: Enter the following data sequentially in registers 20 through 25: 6, 15, 18, 3, 12, 50.

ENTER	PRESS	COMMENTS
	SRR CLR	
	SBR SBR	Enter Subroutine
	R/S INV SBR	
	LRN, E' A	
20	R/S	
25	R/S	
20	R/S	
6	R/S	
15	R/S	
18	R/S	
3	R/S	
12	R/S	
50	R/S	

- Sequential positive integers m to n (n_{max} = 84) can be entered into a...a' by using RCL 88 RTN as the subroutine. To use: press E', A, and then enter m, n, and a by pressing R/S after each entry.
- Print lots of the contents of a...a': log Prt RTN. To use: Press E; D and enter A, A', O (R/S after each entry). Data in a...a' is left unchanged.
- Multiply data in b...b' by squares of the data in a...a': x^2 RTN. To use: Press E', C' and enter a, a', b (R/S after each entry).

000	91	R/S	052	01	1	104	89	89	156	49	PRD
001	76	LBL	053	85	+	105	44	SUM	157	89	89
002	10	E'	054	76	LBL	106	86	86	158	61	GTO
003	86	STF	055	17	B'	107	02	2	159	01	01
004	03	03	056	01	1	108	82	HIR	160	15	15
005	91	R/S	057	08	8	109	58	58	161	72	ST*
006	76	LBL	058	06	6	110	82	HIR	162	86	86
007	12	B	059	95	=	111	57	57	163	92	RTN
008	03	3	060	42	STO	112	01	1	164	63	EX*
009	85	+	061	85	85	113	44	SUM	165	86	86
010	76	LBL	062	91	R/S	114	89	89	166	68	NOP
011	11	A	063	42	STO	115	73	RC*	167	72	ST*
012	01	1	064	88	88	116	88	88	168	88	88
013	06	6	065	75	-	117	68	NOP	169	92	RTN
014	01	1	066	01	1	118	22	INV	170	63	EX*
015	61	GTO	067	82	HIR	119	87	IFF	171	87	87
016	00	00	068	08	08	120	03	03	172	68	NOP
017	59	59	069	82	HIR	121	01	01	173	72	ST*
018	76	LBL	070	07	07	122	30	30	174	88	88
019	13	C	071	91	R/S	123	87	IFF	175	92	RTN
020	86	STF	072	42	STO	124	01	01	176	43	RCL
021	01	01	073	87	87	125	01	01	177	86	86
022	01	1	074	95	=	126	30	30	178	74	SM*
023	07	7	075	94	+/-	127	71	SBR	179	88	88
024	00	0	076	42	STO	128	02	02	180	92	RTN
025	61	GTO	077	89	89	129	01	01	181	43	RCL
026	00	00	078	87	IFF	130	71	SBR	182	86	86
027	59	59	079	01	01	131	40	IND	183	64	PD*
028	76	LBL	080	01	01	132	85	85	184	88	88
029	15	E	081	52	52	133	82	HIR	185	92	RTN
030	05	5	082	91	R/S	134	18	18	186	22	INV
031	85	+	083	42	STO	135	44	SUM	187	74	SM*
032	76	LBL	084	86	86	136	88	88	188	86	86
033	14	D	085	87	IFF	137	87	IFF	189	92	RTN
034	01	1	086	02	02	138	02	02	190	22	INV
035	07	7	087	01	01	139	01	01	191	64	PD*
036	06	6	088	12	12	140	47	47	192	86	86
037	86	STF	089	32	X:T	141	82	HIR	193	92	RTN
038	02	02	090	43	RCL	142	17	17	194	76	LBL
039	61	GTO	091	88	88	143	44	SUM	195	25	CLR
040	00	00	092	76	LBL	144	86	86	196	25	CLR
041	59	59	093	01	01	145	44	SUM	197	81	RST
042	76	LBL	094	12	12	146	87	87	198	76	LBL
043	18	C'	095	43	RCL	147	97	DSZ	199	71	SBR
044	01	1	096	88	88	148	89	89	200	31	LRN
045	85	+	097	22	INV	149	01	01	201	68	NOP
046	76	LBL	098	77	GE	150	15	15	202	68	NOP
047	19	D'	099	01	01	151	81	RST	203	68	NOP
048	03	3	100	12	12	152	02	2	204	92	RTN
049	85	+	101	42	STO	153	82	HIR			
050	76	LBL	102	88	88	154	57	57			
051	16	A'	103	43	RCL	155	22	INV			

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 soon expire.

A renewal card and reminder will be sent to each member in ample time to renew. Return the card to PPX with your check or money order for $18. Be sure to include your membership number on both your card and your check.

Membership Number	Renewal Due:
907199-907688	July 15
907689-908732	August 15
908733-909706	September 15
909707-910537	October 15
918259-918854	July 15
918855-919572	August 15
919573-920478	September 15
920479-921076	October 15

Précis

Editor's Note: This column presents the new PPX programs which are now available. The abstracts here are from programs that the Analysts thought would be of special interest to members. If you have a need for a specific program, send a note to PPX. There is a chance that the program may have already been written. If it has, we will put the abstract in the next issue of the Exchange.

038010F Length of Rolled Material
The length of material rolled on a cylindrical core often must be known for size calculations and inventory estimates. Paper, cloth, rugs and belts are examples of materials produced in this manner. The program uses an equation for length derived for the algebraic and trigonometric relationships using roll diameter and material thickness (caliper). Both Imperial and S. I. unit versions of the program exist.
G. J. Garner, Quebec, Canada
191 steps

128013F Yearly Amortization Schedule - II
Provides continuous authomatic printout of amortization schedule for entire life of loan; or, without printer, displays same data by repeating R/S. 20-year loan printout in about 1 minute. Finds all variables.
Ralph W. Snyder, Indianapolis, IN
365 steps

148011F Investment Versus Inflation
Calculates growth of investment and compares value of investment buying power with growth. Allows use of varying inflation rates for length of investment.
Ken Wehrkamp, Dayton, OH
290 steps, PC-100A

218040F Unlimited Two Way ANOVA With Interaction
Computes all the necessary sums of squares to compile a detailed ANOVA Table. The total number of columns should not exceed 20. However, unlike PPX #218016 the number of rows and replicates are unlimited. Mean squares and F statistics are also calculated.
M. Sabet, Tillsonburg, Ontario, Canada
536 steps, PC-100A

398181F Improved Solution of Quartics
The standard procedures for solution of a quartic equation have been programmed with added subroutines to prevent damage due to round-off errors below the 10th place. Access to the program is also given to correct damage when round-off errors are greater. When the discriminant is zero, this program avoids the usual transfer mistakes. Much faster (8-14 sec. runtime) than other programs. Exact answers are presented.
Theodore M. Bones, Princeton, WV
343 steps

468006F Unit Hydrograph Simulation (Rainfall Runoff)
This program develops a unit hydrograph for a basin given the main channel length and slope, and the basin drainage area. The output gives the incremental period of rise, time in minutes, percent flow, cumulative flow, and CFS flow rate for each time increment. In addition, after all parameters of the unit hydrograph are calculated, the hydrograph is plotted as flow rate versus time.
Douglas Darrow, Tulsa, OK
479 steps, PC-100A

568003F Two-Compartment Pharmacokinetics
This program consists of a series of routines that estimate the parameters of drug distribution in a two-compartment model. This provides a relatively accurate model of drug distribution, equilibration, and excretion of metabolism in a system consisting of a central and peripheral compartment.
Barry S. Tepperman, Ontario, Canada
809 steps, PC-100A

618049F Pipeline Pressure Drop—Any Liquid or Gas
Calculates the pressure drop in a pipeline for a compressible fluid or liquid for all ranges or Reynolds number turbulent or laminar flow. For compressible flow, examines elements of line length chosen by user and iteratively calculates the pressure loss for the total length of line. For liquid flow, the program solves the line loss calculation by one single step. Caution: program does not identify pressure drop exceeding the critical pressure ratio for fluid considered. At higher pressure, the gas constant must be corrected for non-ideal gas.
Frederick G. Young, Ypsilanti, MI
470 steps

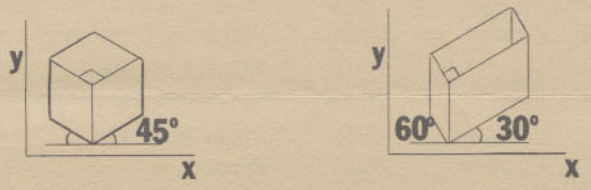
758006F Optimum Disc Blocking
This program will compute the high optimum or low optimum disk blocking factor. Program tries to find the blocking factor that will give 100 percent utilization of disk space or the closest to 100 percent. For oddball record sizes, will provide a much better utilization of disk and is faster than trying to manually find a good blocking factor. Three parameters are required: record size, disk segment size, and maximum buffer size available.
Raymond H. Martinez, San Antonio, TX
234 steps

908149F Business Bar Graph Plotter
Plots values stored in registers 11-51 in bar graph format with a maximum of 20 bar print positions. Program also produces appropriate scales and user defined descriptors for both axes. Data can be stored on magnetic cards and updated graphs can be easily produced.
Laurence V. Moore, Arvada, CO
440 steps, PC-100A

918185F Baseball III
Pits two players against each other in a field of constantly varying odds. Players take turns pitching & batting with the calculator as umpire. Variables include 10 pitches, 4 grades of batters, 9 batters per team, 2 choices at bat, and the "pitcher fatigue factor". A subroutine will plot the current odds at any point in the game.
L. A. Hoetzlein, Colorado Springs, CO
1447 steps, PC-100A, Mod 01

from the Analyst's Desk

• HELP!—PPX member, Timothy M. Rice, has a problem with the Axonometric Projection program (PPX #698006B. It's also in the 3-D Graphics Pakette.) He needs to know what angle of rotation values (A, B, C) are needed to get the "standard" Axonometric Projections shown below:



If anyone has the solution to this problem, please send it to PPX, and we'll forward the answers to Mr. Rice.

• Alphanumeric Inputs—Hal Schneider of Woodland Hills, California, sent PPX the following note concerning the use of registers to input alphanumeric code:

In many programs shown in the newsletter, the numeral codes for alphanumerics are part of the program. Since each digit takes one step, this can be "wasteful" under certain circumstances. An alternate method is to store the value (i.e. digits) in a memory and then recall it prior to the "OP" program steps. Normally, the 10 digit capacity of the memory just meets the maximum five letter capacity of a single print position.

To provide even more capacity, one should recognize that a memory register can actually store 13 digits (remember the guard digits)! If one can live with 3-letter code, you can store two (2) distinct codes per memory as follows:

1. To store the codes "ABC" and "DEF" in memory, register 01: Enter 131415 and press STO 01. Then enter .161721 and press sum 01. "131415.161721" is now in memory register 01. If you RCL 01, the "21" digits will not be displayed, but they are there! To see them, press RCL 01 INV INT.)

2. To select either in the program:

a. For "ABC"; simply RCL 01 prior to OP OX (where X is a value from 1 through 4).

b. For "DEF": RCL 01, INV INT, EE, 6, =, INV EE prior to OP OX. For repeated usage, make these steps a subroutine.

• When submitting a program that is closely related or an improvement to another program, please include a note with your submission stating the differences and advantages of your program over other programs. This procedure will help PPX Analysts not to mistake your program as a duplication.

• When attaching the listing of a program to PPX-59 submission forms, please use an adhesive that will hold the entire listing (not only the corners) in place. Glue Stic by Dennison Manufacturing is excellent for this purpose. Program listings that are not affixed at all edges are vunerable to tearing. Also remember, tape over printing causes fading.

• Mr. Paul D. Sperry of Boulder, Colorado, sent PPX a couple of printer techniques for recording recurrent labeled results:

1. If you want to print data in fixed notation, you normally must program the calculator to execute INV FIX prior to OP 04 and then FIX 0 through 9 prior to Op 06. One way to get around wasting these four steps is to use a single letter for your label and then be sure you use only even fix notations including FIX 0, FIX 2, FIX 4, or FIX 6. Your data will be the alphanumeric symbol changing printing position depending upon the FIX mode selected.

2. A method for using any fix notation without fear of losing or changing your label is to fill the Op 04 register with five letters even though only four can be printed. Thus to print TIME while in FIX 7 mode, use any space holding the first two codes (e.g. 99 plus 37243017). If a single letter such as Y is desired, 9900000045 would place it in the last column while 9945000000 would place the Y in the first column.

• PPX has had several queries on the Truss Design Program (PPX #628091), which involved lines that had been omitted. Below are the missing program steps:

Card 1

216	04	4
272	00	0
379	43	RCL
495	03	3

Card 2

054	53	(
055	43	RCL
111	43	RCL
208	51	51
219	39	COS
275	75	-
382	38	SIN
383	95	=
439	75	-
440	43	RCL

• After giving it more thought, PPX member William H. Beebe has improved his flag testing routine which appeared in the Jan./Feb. issue of the Exchange. This routine has the following four advantages: 1) It will not disturb pending operations, 2) It will not change the contents of the t-register, 3) It makes temporary use of only one data register, returning the original contents of the register when finished, and 4) It will work in any fixed mode.

000	76	LBL	008	00	00
001	11	A	009	12	12
002	48	EXC	010	69	OP
003	01	01	011	99	99
004	22	INV	012	48	EXC
005	87	IFF	013	01	01
006	40	IND	014	91	R/S
007	01	01			

To use this routine, simply enter the number of the flag to be tested and press A. If the flag is set, the calculator display will flash the flag number. If the flag is not set, the display will remain constant.

Of course, there is always the direct keyboard approach for testing flags which consists of the keystrokes 2nd, if flag n, 9, 9, 9 where n is the number of the flag to be tested. Again, if the flag is set, the display will flash.

PROGRAMMING CORNER

"If at first you don't suceed, try try again." The Programming Corner guidelines presented in the March/April Exchange have caused some confusion. A concise version of these guidelines follows:

1. Are there programs that you would like to see available through PPX? Send us a description of your program needs, along with your name, address and member number on a postcard or letter (no phone requests, please). As many of these requests as space permits will be featured in this column.
2. Programs submitted to PPX to fill a Programming Corner request should be accompanied by a note so stating.
3. Programs should be postmarked by the deadline set in each Programming Corner article.
4. All programs received for the same request will be reviewed by PPX Analysts, and the author of the program which we consider the best will receive an order form entitling him to a complimentary Solid State Software™ module of his choice.
5. Other members who submitted acceptable programs (according to standard PPX criteria, Member's Guide Pg. 3) to fill the request will receive an order form entitling them to a complimentary Specialty Pakette of their choice.

Programs to fill the following requests will be accepted until August 31, 1980:

- A program for the sizing of pressure relief values.
- A golf handicapping program that will handle at least fifty names and uses the Callaway handicapping method.
- A program for separating means using the least significant difference test and Duncan's multiple range test.

TI-59 Programming Seminar

A Texas Instruments Programming Seminar may be coming to your area. These seminars will provide beginning and intermediate programming training on the TI-59. Classes consist of two 8-hour days of hands-on training that begin at 8:30 A.M. and last until 4:30 P.M. A luncheon will be served daily. You must provide your own TI-59 and it is highly recommended that you also bring your PC-100 A/C Printer. As a service to our members, we are including the schedule below for dates and locations.

SEMINAR DATES		LOCATION
August	7/8	Dallas, TX
August	28/29	Westbury, Long Island
Sept.	17/18	Philadelphia, PA
Sept.	22/23	Pittsburgh, PA
October	14/15	Costa Mesa, CA
October	20/21	Phoenix, AZ
Nov.	13/14	Detroit, MI
Nov.	17/18	Cleveland, OH
Dec.	4/5	Orlando, FL
Dec.	11/12	Burlingame, CA

To attend, send your name, mailing address, and telephone number to: TI-59 Seminar; P.O. Box 10508, M/S 5873; Lubbock, Texas, 79408. Also include your check or money order for the tuition fee of $150 per person (purchase orders are not accepted). Those members that want to attend need to respond as quickly as possible as there is a limit of 50 participants per class.

ADDRESS CHANGES

In order to ensure uninterrupted service, please submit address changes to PPX at lease six weeks prior to the effective date of the change. Send your name, membership number, old and new addresses to:

PPX
P.O. Box 53
Lubbock, TX 79408

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