PPX Exchange

Vol. 5 Number 5 Copyright 1981 September/October 1981

Contest Winners

Congratulations to the lucky PPX members whose names were drawn from among the 6500 members who responded to our newsletter survey.

Ernest Bently, Hobart Davis, E.S. Evans, Leon Jackson, and Douglas Kalcik will each receive a Texas Instruments 800 Series digital watch.

We would like to thank all of the members who sent in survey responses. The information gained will help us serve you better.

Short Program Storage Addendum

By R.J. Jensen

The article "Short Programs and The Efficient Use of Magnetic Cards" by Sven E. Johannsen, PPX May/June 81, really hit the spot with me. I immediately began consolidating many small programs on a single magnetic card and feel the method is excellent. However, I would like to recommend a change in the Label E indexing routine. As presented, the Label E indexing routine increases in length with each program added and allows the use of only two labels (A and B) in the stored programs. Since the Label E routine changes in length, it must be stored at the end of the file because direct addressing is used for the stored programs. This restriction requires new programs be inserted prior to Label E.

The following describes a modified version of the Label E indexing routine which is a fixed length (56 steps). It allows the use of four labels (A, B, C and D) in the stored programs and uses one data register to store the packed addresses for each stored program. The fixed length allows the Label E routine to be located at the beginning of the stored file which permits additional programs to be added at the end of the file rather than inserted prior to Label E.

Included in the suggested Label E routine is a repartition to protect the address memories, CMs to clear used memories, and a repartition to re-expose the required address registers. This permits the clearing of used memories between runs, thereby eliminating the need to clear used memories in each stored program.

Registers 00-09 are usable by the stored programs. Registers 10-13 store the addresses of the four labels. Registers 14-29 store the addresses in packed form for up to

continued on page 4

The G Update

The "G" Addendum to the Software Catalog is being sent out along with this newsletter. This "Update" represents the abstracts of approximately 500 new programs. This addition brings the total PPX program offering to about 3000 programs. For instructions on the use of your "G Update" please see the frontal matter of your Update.

Note: It has come to our attention that the designation "PC-100A" which appears on the bottom line of some of the abstracts in the catalog is causing confusion. This designation indicates that either the PC-100A or PC-100C Print Cradle is required to run the program.

IT PAYS TO ANALYZE YOUR PROBLEM ≈ REVISITED ≈

By George Vogel

(Editor's Note: Continuing in the vein of his previous article which appeared in the January/February issue of the Exchange, Mr. Vogel shares the benefits of "Analyzing Your Problem.")

It is fairly safe to say that virtually any program involves obtaining values of some quantity y corresponding to some other quantity x.

In principle, two situations may arise: (1) A mathematical relationship is known for y as a function of x, i.e., $y = f(x)$, and we therefore can **calculate** y from x; (2) such a relationship is not known to us, and we can, in essence, only **look up** each y corresponding to a given x (or range of values of x) in a table.

When a mathematical relationship is not immediately obvious, we tend to think of the "look up" method only. But often a "calculate" approach is discovered on closer analysis of the problem; and it will very likely prove far more interesting, and produce a shorter and often faster program than the "look up" method.

As an example, let us write a program (usable as a subroutine) which will convert a % score into the corresponding letter grade and print the two together. The scale will be as follows: below 60, F; from 60 to less than 70, D range; from 70 to less than 80, C range; from 80 to less than 90, B range (each range subdivided into three equal subranges with −, nothing, or + attached, as is customary).

continued on page 10

Programming Techniques: Multiple Card Usage

By Jay Claborn

(Editor's Note: As I was preparing this article, PPX member Lem Matteson submitted an article on the same subject. Althóugh I was not able to print Mr. Matteson's article, I appreciate the added perspective that was gained from his submission.)

The "Personal Programming" manual briefly introduces the subject of "Reading a Card from a Program." This article will explore the usefulness and implementation of this technique.

In order to use the process of reading cards under program control, it is necessary that one understand how the memory storage area is allocated on the TI-59. Page V-42 of "Personal Programming" addresses this topic.

GENERAL TECHNIQUE

When the sequence "N INV Write" (where N is an integer between -4 and 4, inclusive) is encountered in a program, the TI-59 is instructed to read one bank of memory from a magnetic card. The number "N" designates which bank of memory is to receive the information read from the card. When "N" is 1, 2, 3, or 4, the calculator will read the information into banks 1, 2, 3, or 4, respectively, if two conditions are met. First, the number of the bank on the recorded card must be "N", and, second, the recorded partitioning must match the current machine partitioning. If either one of these requirements is not met, the drive roller motor will continue to run until the card is removed from the slot. The program will also halt with the display flashing the number of the recorded bank. If "N" is zero, the recorded bank will be read into the bank that corresponds to the recorded bank number if the recorded and machine partitioning are the same. The consequences of different partitions are the same as described above. If "N" is -1, -2, -3, or -4, the recorded bank will be "forced" into banks 1, 2, 3, or 4, respectively, even if the partitions do not match. If the card misreads for some reason other than mis-matched banks or partitioning, the program will halt with a flashing display of zero.

Once a program is running, the card side that is to read by the program should be placed in the read/write slot so that 1⅛ to 1¼ inches of the card remains visible. This is done so that the card can be read as soon as the "N INV Write" sequence is encountered in the program. If the card is not inserted in the slot before the program execution gets to the "N INV Write", the program will wait for a card to be put in the slot. The only indication that the program is waiting for a magnetic card is the status of the faint "[" on the far lefthand side of the display. When the program is executing, this "[" flickers slightly. When the program is waiting for a card, this "[" becomes solid and does not flicker.

LOADING DATA

One use of reading a card while under program control is to allow easy processing of large data sets. When used in this manner, the data cards are usually prepared by a data input program and are subsequently read under program control from the data processing program. Consider, for instance, the possibility of processing the results from a twenty question survey with numerical responses on the TI-59. The data input program in this case would perform several functions. As one person's numerical response to each question was entered, it would be stored in a data register corresponding to the question number (registers 1-20). Once a person's responses were entered, a data check and correction routine could be performed, and the data input program could write (4 Write) the data from each survey from bank four onto one magnetic card side. In order to read the data, the processing program (or programs) would contain the sequence "4 INV Write" each time a new survey was to be processed. Data registers above 29 would be available for use by the processing program in summing the results of the survey. The advantages of this type of processing are many.

1) The data set is permanently stored.
2) If one piece of data is found to be wrong, only the data card for that piece is redone. It is not necessary to reenter all the data.
3) More data pieces can be added on when desired.
4) Multiple processing of the same data set is possible.

There are also two disadvantages to this type of processing.

1) The processing program must be "baby-sat" (that is, one must insert data dards in the read slot while the program is running).
2) Many magnetic cards are required.

LOADING PROGRAM CODE

As one might suspect, the technique of reading cards under program control is not limited to reading data; program code can also be read. By reading program steps while a program is executing, it is possible to reload banks of memory (blocks of 240 program steps) with different program codes several times without ever stopping the program. This process allows one to implement quite lengthy programs effectively on the TI-59.

There are two distinct methods of loading program code from a program. With the first method, a bank of code is loaded by an "N INV Write" command which is located in a different bank. As an example, consider a program which requires the use of 60 data registers and 900 program steps. Without segmentation such a program would not be feasible on the TI-59. By utilizing banks one and two twice, there is the capacity for 960 program steps and 60 data registers. If the program under consideration could be divided into two roughly equal, semi-independent parts, it could be run using the "loading a card from a program" technique. ("Semi-independent" means the two parts must be able to run alone in that one cannot branch to the other or call a sub-routine from the other; however, one part may depend on the status of the "t" register, data registers and the flags as set by the other part.) The code for the first part of the program could be recorded in banks one and two on a card - call it Card A, and the second part of the program could be recorded in banks one and two of Card B. Steps 475 through 479 of Card A could contain the sequence "1 INV Write GTO 000". This sequence would read in bank one of Card B and transfer execution to step 000 of this new bank of code. Somewhere in bank one of Card B the sequence "2 INV Write" would be used to read in bank two of Card B, and,

thusly, replace the remainder of the code from Card A with code from Card B in the program memory area.

The location of the "N INV Write" statements should be selected with care. Placing the command to reload bank one at the end of bank two is the safest place since all transfers back to the original bank one would have already taken place; however, it may be placed earlier in bank two as long as the original bank one is not branched to in the code following the reload command. Lem Matteson suggests that the reload command might appear in the middle of a nonrelated sequence such as "RCL 01 + INV Write RCL 02 =". The advantage in constructing the code in this manner is that it allows bank one to be loaded without the "1" that remains in the display register after the load getting incorporated in the program.

The second method of loading program code from a program involves the reloading of a bank by a reload command located within the bank to be reloaded. Many programs require large amounts of data to be retained between program segments so that the only memory bank available for reloading is bank one. In such a case, it is necessary for bank one to load the new code over itself. An understanding of the program code processing buffer is required to effectively load a bank over itself. As explained on page V-42 of "Personal Programming" there are 120 registers available for storage in the memory storage area. Eight program steps can be stored in each register of memory. Each bank of memory contains 30 registers which is 240 program steps. When a program is executing, one register (8 program steps) at a time is taken from the memory storage area and placed in the processing buffer. Once these eight steps are performed, another set of eight steps is brought into the processing buffer and performed. If the code in the memory storage area is reloaded by a "1 INV Write", the program code remaining in the processing buffer after the "1 INV Write" will execute, and then the next register of code, which will contain freshly loaded code, will be placed in the processing buffer and executed. It is important, therefore, that one structure the program so that the "1 INV Write" is located in an appropriate position. The thirtieth register in bank one contains program steps 231 through 239. If steps 234 through 239 are "1 INV Write GTO 000", the new bank one will be loaded and the execution will be transferred to step 000 of this new code. This is, of course, not the only acceptable location for the reload statement, however it is probably the safest location.

FOOLPROOFING

When multiple card programs are intended to be used by someone other than the author, it is desirable for the program to check that the correct card has been read. By designating the bank to be loaded in the reload command (i.e. use 1, 2, 3, or 4 INV Write instead of 0, -1, -2, -3, or -4 INV Write) the program will only load a card containing the correct bank number; however, if the bank is to be reloaded several times, one could still put an out-of-sequence card in the read/write slot and have it read into the machine. One method of assuring that the cards are loaded in the correct sequence is to record the different segments of the program at different partition settings. Whenever a bank is to be reloaded, the program can change the partition to the partitioning of the expected card, load the card, and change back to the operating partition. If a card with the incorrect bank or partitioning was placed in the read/write slot, the TI-59 would stop and the display would flash as described earlier.

Another method of foolproofing is to place a card sequence number in the "t" register before the reload command and have the newly loaded bank check to see that it is in sequence. If, for example, a program were to utilize bank one a total of four times, the reload command on the first card side in the sequence might be "2 x→t 1 INV Write GTO 000." The first steps of the second card side could be "2 x=t A CLR 1/x Lb1 A." The other card sides would start in a similar manner. If a card were entered out of sequence, the display would flash nines.

When loading data cards for which the order of entry is important, each data card can have a data register set aside to contain the sequence number. A test similar to the one described above can be performed to see that the data cards were entered in the correct order.

TAKING IT TO THE LIMIT

In general, the maximum number of data registers that can be retained between program segments that reload into bank one is 90. For those few cases when it is necessary to retain more than 90 data registers between reloads, PPX member Gregory Stark has devised a method to save the contents of data registers 90 through 98 using the eight hierarchy registers and "t" register. This method consumes 50 steps of each bank one card in the series except for the first and last card side. For the first card side only 26 steps are required, and only 24 steps are required on the last card side. The number "98" should be stored in data register 99 and written on each card side along with the appropriate program codes. The routines to load the hierarchy registers and "t" register with the contents of registers 90 through 98 (upload) and to load registers 90 through 98 from the hierarchy registers and "t" register (download) are shown below.

DOWNLOAD ROUTINE

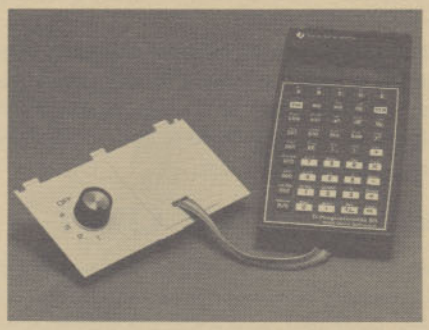
000	32	X:T
001	10	E'
002	10	E'
003	10	E'
004	10	E'
005	10	E'
006	10	E'
007	10	E'
008	10	E'
009	10	E'
010	61	GTO
011	00	00
012	24	24
013	76	LBL
014	10	E'
015	72	ST*
016	99	99
017	01	1
018	22	INV
019	44	SUM
020	99	99
021	00	0
022	54	)
023	92	RTN

UPLOAD ROUTINE

134	08	8
135	09	9
136	42	STO
137	99	99
138	53	(
139	01	1
140	44	SUM
141	99	99
142	73	RC*
143	99	99
144	85	+
145	69	OP
146	19	19
147	22	INV
148	87	IFF
149	07	07
150	01	01
151	38	38
152	32	X:T
153	24	CE
154	01	1
155	22	INV
156	96	WRT
157	61	GTO
158	00	00
159	00	00

New 59/58 Peripherals

As our recent survey indicates, there is quite a bit of interest in peripherals for the TI-59/58. It has come to our attention that American Micro Products, Inc. of Richardson, Texas is manufacturing two 59/58 peripheral devices called module selectors. These products interface through the library module port of the calculator and enable the user to either manually or automatically select one of four Solid State Software™ modules without having to turn the calculator off.



In the Manual Selector, modules are housed in numbered locations at the bottom of the selector. Once the modules are in place, it is an easy matter to dial the desired module per the rotary switch at the top of the selector. Although the Manual Selector will operate on a stand-alone basis, it has been designed to fit into the well of the PC-100A/C printer with the battery removed.

The Auto Selector is significantly more powerful than the Manual Selector because it allows the user to automatically select a module and then execute a specific routine in that module through program or user control. The modules to be accessed are denoted by numbers 0-3 and are accessed in the following manner. The module number is input (0-3) followed by the code "77 2nd OP 04 2nd OP 05". Thus, "277 2nd OP 04 2nd OP 05" would access module number 2. All standard module commands now prevail. For example, "2nd PGM 11" would now call program 11 of module 2. This unit must be operated with the PC-100A/C. The module number and a "SIGMA" character are printed when the access code is executed.

Both selectors are CMOS devices and, therefore, require no external power. Further inquiries should be addressed to American Micro Products by writing to:

American Micro Products, Inc.
705 N. Bowser Suite 112
Richardson, Texas 75080

or calling: (214) 238-1815.

Short Program Storage

(Continued from page 1)

16 programs.

To setup this indexing routine, a program is added to the end of the file (address 056 for the first program). As noted in the original article, the Label A, B, C or D steps are omitted from the stored programs. The address for the beginning of the A, B, C and D portions of the programs are noted. It is not necessary to use all four labels in each stored program. Each stored program is assigned a number, and the packed absolute addresses for that program are stored in the appropriate register in the AAA.BBBCCCDDD format. For example, the first program is assigned the number one, and its addresses are stored in register 14. Subsequent programs are assigned numbers 2 through 16, and the addresses are stored in registers 15-29, respectively. Since use of the gaurd digits is required to store all twelve digits of the four absolute addresses for each program, the AAA.BBBCCCDDD should be entered as AAA + .BBBCCCDDD =. If a label is not used in a particular program, it is advisable to enter "999" as its absolute address. Doing this will cause the display to flash if one erroneously presses the unused label.

To use any program, enter the assigned program number and press E. The Label E routine clears the usable memories (registers 00-09), finds the stored addresses, unpacks them, and stores them in registers 10-13. The stored program can then be used as any regular program by pressing user defined keys A, B, C or D as applicable.

As repartitioned by Label E, this program allows 720 program steps and up to 16 stored programs. These parameters can be modified by the user if one so desires. To expand the usable registers, the packed address registers can be moved into higher number registers if appropriate partitioning and Label E changes are made.

As noted above, Label E repartitions the calculator; so if a program is modified or added, repartition to 6 OP 17 before rewriting the magnetic card. Be sure to record bank four which contains the program addresses.

Included in the listing (shown on next page) is a sample program which converts the calculator/printer to a printing adding machine. The addresses of Labels A and B (C and D are not used) should be stored in register 14 so that it contains 56.060999999. Key A enters the numbers to be added. Key B prints the total and clears the memory for reuse.

000	76	LBL	024	83	GO*	048	19	D'
001	19	D'	025	13	13	049	42	STO
002	22	INV	026	76	LBL	050	12	12
003	59	INT	027	15	E	051	19	D'
004	65	×	028	85	+	052	42	STO
005	03	3	029	01	1	053	13	13
006	22	INV	030	03	3	054	25	CLR
007	28	LOG	031	95	=	055	91	R/S
008	95	=	032	42	STO	056	44	SUM
009	92	RTN	033	10	10	057	00	00
010	76	LBL	034	01	1	058	99	PRT
011	11	A	035	69	OP	059	91	R/S
012	83	GO*	036	17	17	060	98	ADV
013	10	10	037	47	CMS	061	43	RCL
014	76	LBL	038	03	3	062	00	00
015	12	B	039	69	OP	063	99	PRT
016	83	GO*	040	17	17	064	25	CLR
017	11	11	041	73	RC*	065	42	STO
018	76	LBL	042	10	10	066	00	00
019	13	C	043	42	STO	067	98	ADV
020	83	GO*	044	10	10	068	98	ADV
021	12	12	045	19	D'	069	98	ADV
022	76	LBL	046	42	STO	070	91	R/S
023	14	D	047	11	11			

The Challenge of a Game

By David S. Lane

(Editor's Note: In the past year Mr. Lane has distinguished himself as a TI-59 games specialist by his submission of twelve excellent game programs. See your "G" Update for program titles and numbers. In the following article, Mr. Lane explains what his criterion for a good TI-59 game are. For a sample of Mr. Lane's work see "Misadventure" on page 8 of this issue.)

Conceiving and writing a game that others will enjoy playing is a fascinating and rewarding challenge. Here are some thoughts concerning gaming.

The unpredictable should be predictable. In other words, the user should know exactly where and how the game is random. This is done so that players do not become frustrated by not knowing what aspects of the game are random and what aspects are deterministic. In most cases randomness should be exaggerated. Real life situations and normal probability are often too dull for games. For example, in "Yot Race" (PPX 918281G), the wind can change radically in a few turns - in reality it is pretty constant for hours.

Sophisticated games should be sophisticated. When trying to stimulate a "real life" situation in a game, as many of the "real life" factors as possible should be included. Doing this may require some research on the part of the author into how each factor affects the situation. For example, in "Stock Market" (PPX 918235G) long, intermediate, and short market cycles are used to drive the game with each stock having its own beta multiplier. In "Yot Race", the dynamics of the wind speed and direction with respect to sailboat direction are used to determine the movements of the boats.

Simple games can take advantage of human foibles. Even though a game is simple in concept, it should not be easy to master. In the game "Depth Charge" (PPX 918243G), the fact that most people find it difficult to compensate for the added dimension of time makes a rather elementary game a challenge to master. Another human weakness that can be exploited is the tendency to be overly optimistic about the outcome of events, even those for which probability theory predicts a low chance of occurrence.

In situation games, the players should be able to visualize the intended environment. The use of descriptive words on the PC-100A/C printer can greatly enhance this aspect of a game. Complete documentation of the physical layout in games where the player is searching for a hidden object is also helpful.

The game should be easy to operate on the calculator. One should be able to explain the rules in a maximum of two minutes. Players should be able to easily memorize the proper keys to use so that the rules do not have to be read each time before inputing commands to the calculator. Making use of the user defined keys will help make the input simple to remember.

Some other points which are fairly self evident are:

- Games should not be boring. For most games the playing length should be less than 15 minutes.
- Calculator response should be prompt. Players will tolerate a long initialization, but they want fast responses in the actual playing phase of the game.
- In competition games with several players, each player must have the same probability of winning.
- The rules should explain everything happening. Of course, there are exceptions to this statement, the program "Misadventure" being a good example.

The final requirement for a game is that it be played. There is no substitute for the "hands on" testing of a game. When your daughter's college friends choose to solve the mysteries of "Misadventure" rather than go to the beach, you know you have created a good game. But when your wife asks to play "Treasure Hunt" during the television show "Dallas", you know you have conquered the challenge of a game.

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. 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 MS 5820
Lubbock, Texas 79408

If you have further questions regarding the seminar call Sherry Schroeder at 806-741-3277. The schedule for the remainder of 1981 is:

SEMINAR DATES		LOCATION
October	8-9	Washington, DC
October	22-23	Detroit, MI
November	5-6	Cincinnati, OH

Checksum-Number Routine Assures Correct Program Entry

By Colin Gyles

(Editor's Note: Reprinted with permission from "Electronic Design", Vol. 29, No. 11; copyright Hayden Publishing Co., Inc., 1981.)

A 234-step routine for a TI-59/PC-100 programmable calculator checks the correctness of a program entered from a published list, if the author of the published list has provided a checksum number for it. When a lengthy program is keyed into a calculator manually, the chance of an entry error is quite high. Rarely do the examples accompanying the program check the program thoroughly; however, the Checksum-Number routine can.

Run the published program against the Checksum-Number Generator routine and compare the resulting number with the number the author supplied (which is also derived with the Checksum-Number Generator routine). If the two numbers are identical, the newly entered program corresponds to the author's original (to a very high probability level).

The Checksum-Number Generator calculates a checksum number for one program-record card (or bank of 240 steps), when the instructions (Table 1), are followed, starting with pressing label A. The number of the first card to be checked is entered when the question "BANK?" is printed out on the PC-100 printer. Then label B is pressed. When the instruction "INSERT CARD" is printed, insert the card to be checksummed, even while the program is running.

Table 1. Instructions for Checksum-Number Generator

Procedure	Press	Printout	Comments
1. Start*	A	BANK?	
2. Enter card number	B (or R/S)	INSERT CARD	
3. Insert card to be checksummed			While program is running
4.		Prints step number of any offending steps†	Run time about 55 s
5.		LRN, INS (4X) LRN, C	
6.	LRN, INS INS, INS INS, LRN, C		Shifts bank 2 by 4 locations
7.		Prints step number of any offending steps†	Run time about 51 s
8.		Prints checksum	

* During execution, program automatically partitions calculator to 239.89, but restores it to 479.59 by the end.

† Offending steps stored in registers numbered 10 and higher, for use when printer is not employed.

The data on the inserted card are forced within the calculator into its bank-2 memory region, which is then converted to registers (numbered 60 to 89) by the partitioning code 9 OP 17. The TI-59 partitioning is automatically restored to its turn-on condition (479.59, or 480 program steps and 60 registers) at the end of the checksum routine.

Within each of the 30 bank-2 registers (each of which is equivalent to eight program steps), the inserted card's key codes are entered as 16 digits, represented by letters A through P (Table 2). In the process of calculating the checksum, 13 digits (the digits A through M) are converted to the mantissa in the calculator's scientific notation (EE).

Table 2. Data format in bank-2 registers

	Location	Key-code digits	
REG 60	479	AB	16 digits
	478	CD	
	477	EF	
	476	GH	
	475	IJ	
	474	KL	
	473	MN	
	472	OP	
REG 61	471	AB	
	•	•	
	•	•	
	•	•	

Register data in scientific notation

displayed → ← internal

Mantissa	A.BCDEFGHIJKLM
Exponent	None

	Bit no.	3	2	1	0
P is 4-bit code:	Function	Error bit	Exp. sign	Mant. sign	Spare

In scientific notation, no more than eight digits (A through H) are displayed, but all 13 enter the internal summing operation for calculating the checksum. However, to assure adequate weighting for digits I through P, in accordance with steps 5 and 6 of the instructions, the inserted program in bank two is shifted four steps, and the checksum routine is repeated to arrive at a final checksum, which is displayed as a 10-digit number (Fig. 1).

The application of the checksum routine is straightforward except for the P digit. The program to be checksummed may contain key codes with the number eight or nine in the P-digit position, which occurs when the P digit falls on an address that is zero or a multiple of four. If this condition does occur, that particular register is not included in the checksum, and the location to the "offending" key code is printed out (Fig. 2). Then, the offending locations can be manually checked. The check should include eight locations above each offending step.

This deviation from a straightforward approach can be avoided by inserting "dummy zeros" into the original program to be checked, to displace the location of any P digits with the value of eight or nine. Note that the NOP code should not be used as a dummy zero - its key code ends in an eight. In fact, NOP is used for offending codes in Fig. 2.

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

Checksum Listing

000	76	LBL	078	65	×	156	01	1
001	13	C	079	08	8	157	06	6
002	00	0	080	95	=	158	52	EE
003	00	0	081	99	PRT	159	08	8
004	00	0	082	72	ST*	160	22	INV
005	00	0	083	03	03	161	52	EE
006	04	4	084	60	DEG	162	69	OP
007	94	+/-	085	60	DEG	163	03	03
008	44	SUM	086	69	OP	164	60	DEG
009	02	02	087	23	23	165	69	OP
010	17	B'	088	61	GTO	166	05	05
011	99	PRT	089	19	D'	167	02	2
012	91	R/S	090	00	0	168	94	+/-
013	00	0	091	00	0	169	22	INV
014	76	LBL	092	76	LBL	170	96	WRT
015	17	B'	093	11	A	171	17	B'
016	25	CLR	094	47	CMS	172	60	DEG
017	09	9	095	69	OP	173	69	OP
018	69	OP	096	00	00	174	00	00
019	17	17	097	01	1	175	03	3
020	06	6	098	04	4	176	03	3
021	00	0	099	01	1	177	03	3
022	42	STO	100	03	3	178	05	5
023	00	00	101	03	3	179	01	1
024	76	LBL	102	01	1	180	07	7
025	18	C'	103	02	2	181	03	3
026	25	CLR	104	06	6	182	06	6
027	32	X:T	105	07	7	183	03	3
028	73	RC*	106	01	1	184	06	6
029	00	00	107	69	OP	185	69	OP
030	69	OP	108	01	01	186	01	01
031	19	19	109	69	OP	187	02	2
032	60	DEG	110	05	05	188	07	7
033	87	IFF	111	91	R/S	189	03	3
034	07	07	112	76	LBL	190	05	5
035	10	E'	113	12	B	191	03	3
036	67	EQ	114	99	PRT	192	01	1
037	19	D'	115	65	×	193	05	5
038	50	I×I	116	02	2	194	07	7
039	28	LOG	117	04	4	195	69	OP
040	22	INV	118	00	0	196	02	02
041	59	INT	119	85	+	197	02	2
042	22	INV	120	04	4	198	04	4
043	28	LOG	121	07	7	199	03	3
044	44	SUM	122	02	2	200	01	1
045	01	01	123	95	=	201	03	3
046	76	LBL	124	42	STO	202	06	6
047	19	D'	125	02	02	203	05	5
048	60	DEG	126	01	1	204	05	5
049	69	OP	127	00	0	205	00	0
050	20	20	128	42	STO	206	05	5
051	09	9	129	03	03	207	69	OP
052	00	0	130	69	OP	208	03	03
053	32	X:T	131	00	00	209	05	5
054	43	RCL	132	02	2	210	00	0
055	00	00	133	04	4	211	05	5
056	22	INV	134	03	3	212	06	6
057	67	EQ	135	01	1	213	52	EE
058	18	C'	136	03	3	214	06	6
059	06	6	137	06	6	215	22	INV
060	60	DEG	138	01	1	216	52	EE
061	69	OP	139	07	7	217	69	OP
062	17	17	140	03	3	218	04	04
063	43	RCL	141	05	5	219	69	OP
064	01	01	142	69	OP	220	05	05
065	92	RTN	143	01	01	221	25	CLR
066	00	0	144	03	3	222	69	OP
067	76	LBL	145	07	7	223	01	01
068	10	E'	146	00	0	224	60	DEG
069	25	CLR	147	00	0	225	69	OP
070	22	INV	148	01	1	226	04	04
071	86	STF	149	05	5	227	01	1
072	07	07	150	01	1	228	05	5
073	43	RCL	151	03	3	229	69	OP
074	02	02	152	03	3	230	03	03
075	75	-	153	05	5	231	69	OP
076	43	RCL	154	69	OP	232	05	05
077	00	00	155	02	02	233	91	R/S

```
BANK?
          1.
INSERT CARD
PRESS LRN; INS(4×)
      LRN;     C
 183.0606268 ←— 10-digit checksum
                number
```

1. The Checksum-Number Generator program, when run through its own checksum routine, produces the ten-digit checksum number 183.0606268. Try it to see whether you entered the program correctly.

```
BANK?
          1.
INSERT CARD
          0.  ←—┐
PRESS LRN; INS(4×)  —"Offending" step numbers
      LRN;     C          in
          4.  ←—┘   program being
 172.2024023          checksummed
```

2.When an offending code, such as NOP, is inserted in steps 0 and 4 of the generator program to be checksummed (nót the executing program), registers 10 and 11 receive the offending step numbers.

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. (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 to PPX with your check or money order for $20.00. 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 MONTH
910895-911973	November
921595-922334	November
928271-928718	November
900001-901982	December
911974-912576	December
922335-922787	December
928719-929148	December

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

Misadventure

By David S. Lane
(Note: Requires the PC-100A/C Print Cradle.)

There's no dice rolling or other randomness in this game. It's strictly deterministic - solved by your brilliant, logical mind. You start at the mouth of a tunnel with your flashlight in hand. Some one says there is gold in there; there is, at the end of the tunnel. But before you get there, there are many impediments with their associated mollifications (which, of course, you must find within the tunnel and its appendages). The only calculator keys you use are the user defined keys and R/S - but you'll use all of them, many times. If you find the gold, the calculator will flash the price in dragoons.

RULES OF THE GAME

1. You start at the mouth of the tunnel facing northward into the tunnel. You must get to the end of the tunnel to find the gold.
2. You can move north, south, east, west, up or down.
3. At various points in the tunnel you will run into an obstacle such as a door. To get past the obstacle you must usually find a matching object such as a key. The object is generally off the main path of the tunnel in an appendage.
4. At certain places you must use magic to find objects; however, indiscriminate use of magic can be disastrous.
5. At any time you can obtain a list of the items you are carrying.
6. Although the game is strictly deterministic, there are several surprises awaiting you. The whole object of the game is to get to the gold, therefore, detailed rules are not provided. A list of hints is provided below.
   - Treasure hunters never stray too far from the main path, since you can only advance when you go forward.
   - Neither too far to the east or west need you stray.
   - Some rooms are not just right, but are bigger than first realized.
   - Caves are very deep and seem almost bottomless, but holes are endless.
   - You may have lost more than you realize.
   - To relight your life, step back, cogitate, take a new look, and give it the old college try again.
   - An abyss is too big for material things; you've got to do your own thing.
   - Too much fooling around will get you the shaft.
   - The key to success is double.
   - It really is magic, but the door also unlocks the key.
   - Candy is stuck up by gum.
   - The cache is not a catch.
   - Gnomes are afraid of pixies, but really, pixies are sweet things who love magic.
   - Be sure all loose ends are tied up for the cliff.
   - Straight shooters can kill snakes.
   - Beekeepers cover-up to avoid stings.
   - Timing is as important as finding.

RECORDING INSTRUCTIONS

1. Store the prestored data shown below in their respective data registers.

DATA REGISTER	REGISTER CONTENTS
20	2724222337.
21	261745.
22	35323317.
23	15132217.
24	30133626.
25	224130.
26	224131.
27	1513311645.
28	33244445.
29	261745.
30	3623132137.
31	16323235.
32	1527242121.
33	14243516.
34	14171736.
35	1314453636.
36	3631132617.
37	33244445.
38	2231323017.
39	22133717.
40	3032413723.
41	3124152317.
42	35323230.
43	23322717.
44	15134217.
45	332437.
46	1513152317.
47	31323226.
48	43132727.
49	22322716.

2. Repartition by pressing "5 2nd OP 17".
3. In the learn mode, key in the program steps.
4. Repartition by pressing "6 2nd OP 17".
5. Record banks 1 and 2 on one magnetic card, and record banks 3 and 4 on a second magnetic card.
6. The printout from the "Checksum" program (page 6) is shown below to help you verify your program code.

```
BANK?
          1.
INSERT CARD
        168.
         72.
PRESS LRN, INS (4×)
      LRN,     C
        228.
        212.
        140.
        124.
         92.
         84.
         76.
 179.4618758
```

```
BANK?
          2.
INSERT CARD
        440.
        400.
        392.
        240.
PRESS LRN, INS (4×)
      LRN,     C
        428.
        420.
        404.
        396.
        284.
        252.
        244.
 140.4600352
```

```
BANK?                    BANK?
        3.                       4.
INSERT CARD              INSERT CARD
        536.             PRESS LRN, INS(4x)
        496.                   LRN,     C
        488.              47.87301481
PRESS LRN, INS(4x)
      LRN,     C
        532.
 124.3739546
```

USER INSTRUCTIONS

1. Press E to start.
2. To move in a direction press the corresponding user defined key as shown in the table below.

Direction	Key
North	A
South	A'
East	B
West	B'
Up	C
Down	C'

3. Press D to display current location. Three coordinates will be printed corresponding to distance north, east, and up, respectively.
4. Press D' for a list of the objects you have.
5. Press E' to try magic.
6. Press E to restart the game.

(Editor's Note: According to Mr. Lane there are only four people that he knows of who have completely solved the game. After about five hours of play, I have yet to get past that pixie critter, so don't get discouraged. I recommend that as you explore the tunnel you make a map of where everything is.)

PROGRAM LISTING

000	76	LBL	034	76	LBL	068	44	SUM	102	10	E'	136	43	RCL	225	76	LBL	314	32	X:T	403	61	GTO
001	19	D'	035	90	LST	069	22	INV	103	92	RTN	137	07	07	226	13	C	315	03	3	404	29	CP
002	98	ADV	036	42	STO	070	86	STF	104	76	LBL	138	85	+	227	43	RCL	316	94	+/-	405	76	LBL
003	01	1	037	00	00	071	06	06	105	11	A	139	02	2	228	09	09	317	67	EQ	406	85	+
004	00	0	038	75	-	072	69	OP	106	87	IFF	140	09	9	229	32	X:T	318	23	LNX	407	86	STF
005	42	STO	039	01	1	073	27	27	107	06	06	141	95	=	230	01	1	319	61	GTO	408	06	06
006	02	02	040	00	0	074	76	LBL	108	44	SUM	142	42	STO	231	67	EQ	320	65	×	409	61	GTO
007	09	9	041	95	=	075	65	×	109	71	SBR	143	00	00	232	10	E'	321	76	LBL	410	65	×
008	42	STO	042	48	EXC	076	98	ADV	110	58	FIX	144	73	RC*	233	69	OP	322	79	ß	411	76	LBL
009	01	01	043	00	00	077	00	0	111	43	RCL	145	00	00	234	29	29	323	01	1	412	14	D
010	29	CP	044	72	ST*	078	91	R/S	112	07	07	146	69	OP	235	43	RCL	324	85	+	413	43	RCL
011	76	LBL	045	00	00	079	15	E	113	32	X:T	147	03	03	236	07	07	325	76	LBL	414	07	07
012	70	RAD	046	76	LBL	080	76	LBL	114	00	0	148	76	LBL	237	32	X:T	326	37	P/R	415	99	PRT
013	69	OP	047	99	PRT	081	89	¶	115	77	GE	149	69	OP	238	03	3	327	01	1	416	43	RCL
014	21	21	048	42	STO	082	00	0	116	44	SUM	150	69	OP	239	67	EQ	328	85	+	417	08	08
015	73	RC*	049	00	00	083	42	STO	117	06	6	151	05	05	240	38	SIN	329	76	LBL	418	99	PRT
016	01	01	050	32	X:T	084	19	19	118	67	EQ	152	43	RCL	241	06	6	330	57	ENG	419	43	RCL
017	67	EQ	051	73	RC*	085	05	5	119	75	-	153	07	07	242	67	EQ	331	01	1	420	09	09
018	60	DEG	052	00	00	086	00	0	120	01	1	154	32	X:T	243	85	+	332	85	+	421	99	PRT
019	42	STO	053	69	OP	087	32	X:T	121	01	1	155	05	5	244	08	8	333	76	LBL	422	61	GTO
020	00	00	054	03	03	088	61	GTO	122	67	EQ	156	67	EQ	245	67	EQ	334	39	COS	423	65	×
021	73	RC*	055	69	OP	089	44	SUM	123	95	=	157	69	OP	246	96	WRT	335	01	1	424	76	LBL
022	00	00	056	05	05	090	76	LBL	124	09	9	158	01	1	247	61	GTO	336	85	+	425	95	=
023	69	OP	057	00	0	091	58	FIX	125	85	+	159	67	EQ	248	65	×	337	76	LBL	426	98	ADV
024	03	03	058	91	R/S	092	29	CP	126	32	X:T	160	34	fX	249	76	LBL	338	23	LNX	427	43	RCL
025	69	OP	059	15	E	093	43	RCL	127	95	=	161	00	0	250	18	C'	339	01	1	428	49	49
026	05	05	060	76	LBL	094	08	08	128	42	STO	162	91	R/S	251	43	RCL	340	85	+	429	69	OP
027	76	LBL	061	55	÷	095	22	INV	129	00	00	163	15	E	252	09	09	341	76	LBL	430	03	03
028	60	DEG	062	32	X:T	096	67	EQ	130	29	CP	164	76	LBL	253	32	X:T	342	67	EQ	431	69	OP
029	97	DSZ	063	02	2	097	10	E'	131	73	RC*	165	16	A'	254	03	3	343	01	1	432	05	05
030	02	02	064	01	1	098	43	RCL	132	00	00	166	71	SBR	255	94	+/-	344	85	+	433	98	ADV
031	70	RAD	065	67	EQ	099	09	09	133	22	INV	167	58	FIX	256	67	EQ	345	76	LBL	434	61	GTO
032	61	GTO	066	89	¶	100	22	INV	134	67	EQ	168	69	OP	257	34	fX	346	59	INT	435	78	Σ+
033	65	×	067	76	LBL	101	67	EQ	135	55	÷	169	37	37	258	69	OP	347	01	1	436	76	LBL
												170	29	CP	259	39	39	348	85	+	437	29	CP
												171	43	RCL	260	43	RCL	349	76	LBL	438	69	OP
												172	07	07	261	07	07	350	87	IFF	439	03	03
												173	77	GE	262	32	X:T	351	01	1	440	69	OP
												174	65	×	263	05	5	352	85	+	441	05	05
												175	22	INV	264	67	EQ	353	76	LBL	442	22	INV
												176	86	STF	265	42	STO	354	86	STF	443	86	STF
												177	00	00	266	04	4	355	02	2	444	02	02
												178	86	STF	267	67	EQ	356	01	1	445	01	1
												179	01	01	268	77	GE	357	95	=	446	85	+
												180	61	GTO	269	61	GTO	358	61	GTO	447	76	LBL
												181	65	×	270	65	×	359	90	LST	448	15	E
												182	76	LBL	271	76	LBL	360	76	LBL	449	01	1
												183	12	B	272	43	RCL	361	77	GE	450	95	=
												184	43	RCL	273	43	RCL	362	00	0	451	69	OP
												185	08	08	274	08	08	363	42	STO	452	17	17
												186	32	X:T	275	32	X:T	364	12	12	453	47	CMS
												187	02	2	276	01	1	365	42	STO	454	05	5
												188	67	EQ	277	67	EQ	366	19	19	455	69	OP
												189	15	E	278	49	PRD	367	04	4	456	17	17
												190	69	OP	279	02	2	368	94	+/-	457	22	INV
												191	28	28	280	67	EQ	369	85	+	458	87	IFF
												192	43	RCL	281	87	IFF	370	76	LBL	459	02	02
												193	07	07	282	15	E	371	96	WRT	460	80	GRD
												194	32	X:T	283	76	LBL	372	01	1	461	43	RCL
												195	02	2	284	38	SIN	373	85	+	462	30	30
												196	67	EQ	285	43	RCL	374	76	LBL	463	69	OP
												197	97	DSZ	286	08	08	375	88	DMS	464	03	03
												198	03	3	287	32	X:T	376	02	2	465	69	OP
												199	67	EQ	288	01	1	377	85	+	466	05	05
												200	43	RCL	289	67	EQ	378	76	LBL	467	76	LBL
												201	61	GTO	290	59	INT	379	28	LOG	468	80	GRD
												202	65	×	291	02	2	380	02	2	469	98	ADV
												203	76	LBL	292	67	EQ	381	85	+	470	86	STF
												204	17	B'	293	39	COS	382	76	LBL	471	02	02
												205	43	RCL	294	15	E	383	49	PRD	472	04	4
												206	08	08	295	76	LBL	384	01	1	473	00	0
												207	32	X:T	296	42	STO	385	85	+	474	87	IFF
												208	02	2	297	43	RCL	386	76	LBL	475	00	00
												209	94	+/-	298	09	09	387	97	DSZ	476	99	PRT
												210	67	EQ	299	32	X:T	388	04	4	477	86	STF
												211	15	E	300	01	1	389	01	1	478	00	00
												212	69	OP	301	94	+/-	390	95	=	479	02	2
												213	38	38	302	67	EQ	391	61	GTO	480	00	0
												214	43	RCL	303	28	LOG	392	99	PRT	481	42	STO
												215	07	07	304	02	2	393	76	LBL	482	10	10
												216	32	X:T	305	94	+/-	394	75	-	483	00	0
												217	05	5	306	67	EQ	395	98	ADV	484	00	0
												218	67	EQ	307	67	EQ	396	98	ADV	485	22	INV
												219	47	CMS	308	61	GTO	397	98	ADV	486	87	IFF
												220	07	7	309	65	×	398	98	ADV	487	01	01
												221	67	EQ	310	76	LBL	399	98	ADV	488	99	PRT
												222	88	DMS	311	47	CMS	400	98	ADV	489	22	INV
												223	61	GTO	312	43	RCL	401	43	RCL	490	86	STF
												224	65	×	313	09	09	402	35	35	491	01	01

Listing Continued Above

Listing Continued on Page 10

```
492  43 RCL   505  00   0   518  57 ENG   531  61 GTO
493  40  40   506  67  EQ   519  02   2   532  99 PRT
494  69 DP    507  79   X   520  00   0   533  76 LBL
495  03  03   508  04   4   521  67  EQ   534  34 fX
496  69 DP    509  01   1   522  79   X   535  98 ADV
497  05  05   510  67  EQ   523  22 INV   536  98 ADV
498  02   2   511  86 STF   524  77  GE   537  22 INV
499  00   0   512  22 INV   525  15   E   538  86 STF
500  61 GTO   513  77  GE   526  08   8   539  00  00
501  99 PRT   514  34 fX    527  67  EQ   540  43 RCL
502  76 LBL   515  02   2   528  37 P/R   541  45  45
503  10 E'    516  05   5   529  04   4   542  61 GTO
504  05   5   517  67  EQ   530  08   8   543  29 CP
```

It Pays To Analyze

(continued from page 1)

Finally, 90 to 93.33 . . ., A-; and above that, A.

Now it is not hard to write a program that will accomplish this. It would simply determine (via 11 tests) which of the 12 ranges the given % score falls into, and direct processing to the proper one of 12 subroutines to produce the desired print code. This is something one would write for immediate practical use. It would require at least 170 steps. It would work. It would also be deadly boring.

But we can do much better than that, and perhaps learn something new in the process. Let us set up a table based on the above, and multiply the score at the bottom of each range by 0.3 (ignore the 4th column for now):

	%	x 0.3 =	÷3 = q	r
A	93.33 . . .	28	9	1
A-	90	27	9	0
B+	86.66 . . .	26	8	2
B	83.33 . . .	25	8	1
B-	80	24	8	0
C+	76.66 . . .	23	7	2
C	73.33 . . .	22	7	1
C-	70	21	7	0
D+	66.66 . . .	20	6	2
D	63.33 . . .	19	6	1
D-	60	18	6	0
F	<60	<18		

We see that, for example, a score of 82.0 would give 24.6, and that the integer part would be the same for **all** other scores deserving a B-. Thus the integer part of the product with 0.3 gives the letter grade. This suggests the first possible improvement. We could convert the integer part into an absolute address for the appropriate print-code-generating subroutine, and use indirect addressing to get there. This would avoid all but two of the tests (grades A and F obviously need a special treatment). But we can do even better than this!

The periodic recurrence of -, no sign, and + suggests modulo 3 arithmetic, and so we divide the numbers in the third column by 3 into quotient (q) and remainder (r), and immediately see that the quotient correlates with the letter, and the remainder with the sign. We might now be tempted to write a program with five subroutines for the letter and two or three for the sign (a saving over the original twelve subroutines), but we can do much better again!

It so happens that the print codes for A, B, C, and D can be obtained by simply subtracting the quotients in the fourth column from 22 (e.g., for C, 22 - 7 = 15). This is much more straight-forward than testing and branching. As for the + and - signs, two tests would be enough to decide whether the print codes 20, 0, or 47 should be produced for a remainder of 0, 1, and 2, respectively. But — now that we have not going — let's better even this! The six values represent three ordered pairs, three points on a parabola, whose equation is easily found to be $y = 33.5x^2 - 53.5x + 20$. Although the appropriate values for letter and sign could be combined to the complete print code and input via op 04, a more efficient way is to assemble 1000022 - q = $0.01(33.5r^2 - 53.5r + 20)$ in the hierarchy register HIR 08, which will provide printing in the rightmost segment. When using this method of entry, the first three digits and the decimal point are ignored in printing, and the result is the same as when entered in the conventional manner, except that it is **immune to any fix mode one might be in.**

The exceptional grades A and F require special treatment. If the result of the multiplication by 0.3 is greater than 28, it is changed to 28 so as to produce an A. If the result is less than 18, it is replaced by 4, which will eventually lead to an F. These then are the only two tests remaining of the original eleven. But how much did we gain? For an answer, look at the program below (to use: key in % score, press A — naturally a printer is required).*

The program has 77 steps (vs. at least 170 for the "utility" version). It has only 2 "internal" labels—or absolute addresses—vs. at least 12. Its printing is immune to Fix. But above all, it was more fun to write, and we may have learned a few things while doing it.

*The purpose of EE at location 004 is twofold: (1) Without it one might, for example, input 59.95 under Fix 1 mode and the answer would be 60.0 F. No student would like that! With the EE, the answer will be 59.95 F under Fix 2, but 60.0 D- under Fix 1; (2) it will cause the program to run in the scientific mode until reset by CLR at loc. 071; in normal mode certain HIR operations give incorrect results with a number having an absolute value of less than 1.

```
000  76 LBL   026  39 COS   052  32 X:T
001  11  A    027  76 LBL   053  93   .
002  42 STO   028  38 SIN   054  03   3
003  01  01   029  32 X:T   055  03   3
004  52 EE    030  01   1   056  05   5
005  65   x   031  08   8   057  75   -
006  02   2   032  32 X:T   058  93   .
007  08   8   033  77  GE   059  05   5
008  32 X:T   034  39 COS   060  03   3
009  01   1   035  04   4   061  05   5
010  00   0   036  76 LBL   062  54   )
011  00   0   037  39 COS   063  65   x
012  00   0   038  59 INT   064  32 X:T
013  00   0   039  75   -   065  85   +
014  02   2   040  53   (   066  93   .
015  02   2   041  24 CE    067  02   2
016  82 HIR   042  55   ÷   068  95   =
017  08  08   043  03   3   069  82 HIR
018  93   .   044  54   )   070  38  38
019  03   3   045  59 INT   071  25 CLR
020  95   =   046  82 HIR   072  43 RCL
021  22 INV   047  58  58   073  01  01
022  77  GE   048  65   x   074  69 DP
023  38 SIN   049  03   3   075  06  06
024  32 X:T   050  95   =   076  92 RTN
025  61 GTO   051  65   x
```

PROGRAMMING CORNER

This column serves as a link between program users and program writers. We invite any PPX member with a specific program request to send it to us. Since PPX is not staffed to produce custom software, we publish program requests so that these needs can be made known to other PPX members. PPX provides incentives for those authors whose programs are accepted to fill "Programming Corner" requests.

We have recently become aware of a couple of programs that may fill past requests. Although these programs are not available through PPX, we believe they may be of interest to some PPX members.

Material Balance Program

Available from C&Co, P.O. Box 353, Pine Brook, N.J. 07058, this program was created to handle material balance calculation based on "streams". Each stream consists of its identification, composition, and the sum of its components. Addition, substraction, multiplication, and division of two streams can be performed by this program. In addition to material balance calculation, this program can be used in a wide variety of engineering and scientific applications. For example, conversion of a composition from mass basis to mol basis and vice versa, balancing chemical reaction equations, and heat or energy balance.

F-Chart Sizing Programs

Available from Sunshine Power Company, 1018 Lancer Drive, San Jose, CA 95129 are programs for both air and water system sizing. The "f-Chart Air System Sizing" program prints the monthly percent solar contribution from an air collector system with rock storage and domestic water preheat. The "f-Chart H_2O System Sizing" program prints the monthly percent solar contribution from a water collecter system with water storage and domestic water preheat. Both programs allow all system parameters to be varied and printed.

PPX member John Dorsey is interested in locating someone who can translate punch key cards for a Monroe 1665 into TI-59 program code. If you are interested please contact Mr. Dorsey at Room 202 Courthouse, Canyon, TX. 79015.

The program requests for this issue are listed below. All submissions to fill these requests should be submitted by December 31, 1981.

- A program to play a miniature version of the oriental game of "Go".
- A program that computes the frictional values of poly vinyl chloride (PVC) pipe carrying water with multiple outlets over varying distances and adds the total losses in the pipe.

CUSTOM MODULE APPLICATIONS: REC 20

Mr. Jerry Parks of Dallas, Texas has recently developed the REC 20 pre-programmed real estate computer consisting of a modified TI-58C with a custom software module. The REC 20 module contains these twenty programs: New Loan Payments, Assumption Payments, Payment Comparison, Add-On Lien Payments, Balloon Note Payment, Balloon (Balance Due), Income Conversion, Seller's Net Equity, Buyer's Closing Cost, Pro-Rate Interest, Pro-Rate Insurance, Pro-Rate Taxes, Amortization Table, Interest Only Table, How Much House Buyer Can Afford, Floor Area, FHA Down Payment, FHA Parameters, and Days Between Dates.

Mr. Parks, the author of the programs, is the owner of six Dallas area brokerage offices and founder of Jerry Parks Builders, Inc. He has also authored six books on real estate sales and office management, and numerous real estate courses accredited by the Texas Real Estate Commission. His diversified interests also include banking, a national real estate franchise organization, and part ownership of one of the largest proprietary real estate schools in the country.

Since the REC 20 module was designed by a real estate expert it provides for fast and simple computation of the most common real estate transactions. Of course, the REC 20 comes complete with easy to use program guides and worksheets which are contained in a black briefcase which also provides a space for storage of the calculator.

Other benefits derived from the REC 20 system include:

- Salespeople can compute instantly how much house a client can afford with his income, thus avoiding time wasted showing the prospect homes he can't buy.
- Clients don't cool off while the salesman goes through a dozen different computations to figure a loan payment on every type and amount of loan at any interest rate or term, or how much income a prospect needs to qualify for a loan.
- Compares difference in actual cost to a purchaser of a higher priced home with a lower interest rate with cost of lower priced home at a higher interest rate.
- Computes square footage of any area for quick dollar per square foot value comparison between homes.
- Accurately computes closing costs for both buyers and sellers; pre-programming assures no costs are accidently left out by the agent.

For further information on the REC 20 Real Estate Calculator please write:

REC, Inc.
2725 Valley View, Suite 102
Dallas, Texas 75234

Précis

This column presents some of the new PPX Programs which have been recently accepted. The abstracts here are from programs that the analysts thought would be of special interest to members. You can purchase these programs at a cost of $4.00 each. Send your order to: Texas Instruments: C/O PPX Department; P.O. Box 109, Lubbock, TX 79408. Include an additional $2.00 to cover postage and handling.

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. Requests for programs not yet written will be placed in the "Programming Corner" column.

018014H Financial Statement Preparation
This program takes a worksheet utilized by accountants during the preparation of financial statements and can be utilized for manufacturing and non-manufacturing concerns. It provides zero proof totals for all columns and prints the cost of whether it is a net income or loss figure. This program can be used with any size worksheet, no matter how many pages or accounts are utilized by the user.
Thomas K. Lehman, Magna, UT
587 Steps, PC-100A

658159H Inverse Laplace Transform
Uses the Heaviside Expansion Theorem for unrepeated real and complex poles (denominator roots) to find the time response of a linear network to a particular input. For example, the step responses of low pass, band pass, and high pass filters may easily be calculated. The case of repeated poles is seldom of interest in engineering applications; however, should repeated poles occur, the convolution program of the Electrical Engineering module (EE-11) can extend the use of this program with repeated poles. To use the program it is only necessary to enter the Laplace function.
Lee Payne, Tucson, AZ
850 Steps, Mod 1

668168H Restrictive Orifice Sizing
Sizes a restrictive orifice to limit gas flow. This program sizes the orifice so that it will pass peak load with the regulator working and it calculates the volume through the orifice if the regulator were to fail open. This program is used in comparing the restricted flow to the relief valve capacity in order not to overpressure the downstream (lower pressure) piping.
James N. Phillips, Dallas, TX
606 Steps, PC-100A

778035H Lambert, State (Coordinate Transformation)
Transforms coordinates from geographic to grid and from grid to geographic for those state systems employing a Lambert projection. Additionally, computes meridian convergence and scale factor for points given their geographic coordinates or grid coordinates.
Thomas W. Dickson, Vivian, LA
409 Steps, PC-100A

868022H Imputed BTU (Heat Value) of Natural Gas
Program calculates BTU of natural gas from molecular percentages of gas components. Input variables are molecular percentages of gas components from a gas analysis test.
Dave Enarson, Sugarland, TX
399 Steps

908217H Block Edit Program Relocator
This program permits the relocation of programs resident in memory banks 2, 3, and part of 4. Inherent characteristics of the structure of numbers used by the TI-59 limits the valid transfer of program memory to about 65%, so that relocated code requires manual editing.
Clive McCarthy, Santa Clara, CA
224 Steps, PC-100A

918294H Chess Descriptive Notation
Program prints all moves of a standard chess game in descriptive notation. Up to 99 moves may be entered, more than adequate for most games. Special moves, such as castling or en passant, can be entered with ease. Includes all standard symbols.
Ronald W. Rushing, Albany, GA
389 Steps, PC-100A, Mod 10

919301H Liars Dice
A challenging game of bluffing for 2 to n people using 1 to n calculators. A player "rolls" five dice to make up a poker hand (which he alone sees). He then announces his hand -either real or a bluff. The next player must challenge or accept. If he accepts, he looks at the dice and can roll all, some or none of the dice. He must then announce a hand higher than the one he accepted. The play continues until a challenge occurs. At this time, the round ends with the player who made the last announcement winning or losing dependent on whether his roll was real or a bluff.
David S. Lane, Clearwater, FL
66 Steps

928052H Print Long Division
Given a dividend and a divisor, this program prints all intermediate steps and the final quotient and the remainder in the long division format that is taught in grade school.
David Kantrowitz, Brookline, MA
480 Steps, PC-100A

958020H Exposure Compensation Factor
Calculates the printing exposure factor resulting from changes in focal length or aperture of lenses or distance separating negatives and images. Values are accepted in millimeters or focal lengths, inches or millimeters for spacing and numerics for aperatures; these may be entered in any order, and are retained until changed. The exposure factor is calculated as a ratio and in photographic stops. Changes in times or apertures are readily determined at the keyboard. Lateral magnifications are intrinsically available.
Thurman E. Smithey, Chula Vista, CA
273 Steps