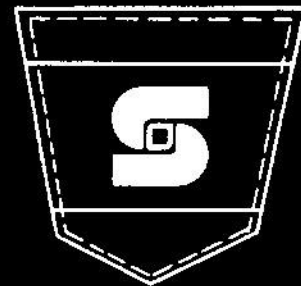


POCKET COMPUTER

NEWSLETTER



© Copyright 1982 — Issue 11

January

SHARP PREPARING NEW POCKET COMPUTER

Reliable sources report that Sharp Electronics, Inc., producer of the popular PC-1211 and Radio Shack TRS-80 versions of pocket computers, is just about ready to release an exciting, dramatically enhanced, new pocket computer. The upgraded unit will be designated the Sharp PC-1500.

Preliminary reports indicate that the new device will rival the capabilities of the recently announced RL-H1000 Hand Held Computer marketed by Panasonic.

For starters, the PC-1500 will have increased memory capacity. It is reported to contain 16 kilobytes of ROM memory containing an improved BASIC language and operating system plus approximately 3.5 kilobytes of user-programmable memory. Additionally, expansion units will allow more memory to be coupled to the system.

A specially-designed 8-bit CPU using CMOS technology is at the heart of the new unit.

The machine will be housed in a "pocketable" case that is slightly larger than the current PC-1211. Measurement figures of approximately 7 inches by 3.5 inches by 1 inch have been reported. A QWERTY keyboard arrangement that is somewhat more comfortable to use than the present PC-1211 version is also claimed, along with six special "programmable" keys.

BASIC language enhancements are said to include the ability to manipulate string variables and a built-in real-time clock.

The really big news, however, is that the unit has been specifically designed to interface with a variety of peripheral devices that Sharp will be introducing during 1982.

The first such add-on to be made available is reported to be a small printer that will be capable of producing print-outs in four different colors! This interface will also enable data to be stored and recovered using an audio tape recorder. Reports indicate that this interface will be referred to by the model designation CE-150.

Another peripheral will enable the PC-1500 to display information on a television set or video display.

Still other add-ons will provide additional memory and the ability to communicate with other devices.

For instance, it has been reported that it will be possible to simultaneously control two different tape units so that data can be read from one unit while it is stored to a second. This will permit relatively powerful data-handling procedures to be implemented. It has also been ascertained that the tape units might eventually be replaced by floppy disks or similar fast-access mass storage devices.

The PC-1500 will reportedly be available in the United States in late February or early March, 1982. A retail price of approximately \$300 has been indicated. As confirmation of this information, it has been reliably reported that the unit will be available in Japan at the start of 1982 at an equivalent U.S. price of \$277.00.

The CE-150 combination 4-color printer and tape cassette interface will probably have a U.S. price of approximately \$250.00.

No word has yet been obtained by PCN as to whether or not Radio Shack will market a version of the PC-1500. However, it is reported that Sharp Electronics, Inc., is forming a new U.S. marketing division to boost the sales of the PC-1500 and similar products.

ROUTINE PERMITS INDIRECT SPECIFICATION OF FILE NAMES

The accompanying routine enables an unskilled user to specify a file name when saving data. Subsequently, the user can call for that same data file to be recovered. This type of *indirect* file name specification eliminates the need for a user to be familiar with the actual I/O commands used by the Radio Shack TRS-80 and Sharp PC-1211 pocket computers.

The routine uses a technique submitted by *Kendal B. Stonebrook*, 811 Princeton Avenue, Lansing, MI 48915. The file name specified by the user is first stored in the highest unused array element available in memory. This element is then recorded as a separate data file having the name INDEX. Immediately following this file, another data file is recorded using the name FILE. This is the file that contains the user's actual data.

When a user-named file is desired, another routine reads each file named INDEX and examines the stored array element for the name specified by the user. When a match is found, the following file is loaded to retrieve the desired data!

Notes

Line 930 may be used to obtain the highest array element available in the PC as this will vary depending on the size of the installed program. (You can convert this line from the general case once the size of a program is determined.) Line 935 then places the user-defined file name in the array element A\$(C) and activates the tape unit to record this element in a data file named INDEX. Line 940 proceeds to record another file named FILE that contains the actual data elements that are being stored. A GOTO or RETURN statement may be appended to this line as appropriate for a specific application.

Recovery of a user-named file begins at line 900.

The routines illustrated use variables A, B, C, A\$(27) and A\$(NNN) where NNN is the highest array element available.

Program Indirect File Names

```
900: "F"INPUT "FI      +1: IF B>12
      LE NAME? ";A      PRINT "N/F":
      $:A$(27)="":      END
      B=0               925:GOTO 905
905: INPUT # "INDE     930: "D"INPUT "#
      X";A$(27)         MEMORIES LEF
910: IF A$(27)=A$      T? ":C=C+2
      BEEP 4:PAUSE      6
      "LOADING "A      935: INPUT "FILE
      $(27)"; FILE     NAME? ";A$(C)
      ":INPUT # "FI    ):PRINT # "IN
      LE":PRINT "F     DEX";A$(C):
      ILE LOADED":     BEEP 4
      GOTO "A"         940:PRINT # "FILE
920: PAUSE "SKIPP      ":PRINT A$(C)
      ING "A$(27)      ": " FILE OUT
      "; FILE":B=B
```

INSTRUCTION EXECUTION TIMES FOR THE RADIO SHACK TRS-80 POCKET COMPUTER

Are you trying to minimize program execution time? Speed can be significantly affected by the proper choice of variables, operations, and method of transfer. This is especially true when a portion of a program needs to be repeated a large number of times.

Norlin Rober has compiled the timing information presented here. Remember, the actual measurements shown may vary slightly on individual PCs. However, the relationships and relative speeds of the various types of directives will remain essentially the same for your unit.

Watch Those Program Transfers!

Perhaps the greatest time-saving technique is to shorten the search time when using GOTO and GOSUB statements. A transfer to a *line number* is relatively fast when the location sought is near the GOTO or GOSUB statement, *regardless of direction!* A transfer to a *label* is fastest when the label is located near the beginning of program memory.

Searches for labels or line numbers can be quite slow if the search must take place over a large number of program bytes. In some cases a single transfer can take as long as 2.7 seconds. If such a transfer is to be performed a large number of times (as, say, part of a loop), some relocation of program lines can significantly reduce execution time.

The exponentiation operation is also relatively slow, taking up to one second. Using AA in place of A^2 , AAA in place of A^3 , etc., does more than simply speed up operation. It also permits the use of negative values of A and often increases accuracy!

A Note about Timing Measurements

Execution times were obtained using the following method: The routine

```

1  FOR X=0 TO 199
2  NEXT X
3  BEEP 1

```

takes 36.1 seconds to execute (on the test TRS-80 PC). The BEEP is used to assist in obtaining the actual timing. If line 2 is altered to:

2 W=A:W=A:W=A:W=A:W=A:NEXT X
execution takes 95.5 seconds. That is 59.4 additional seconds for the 1000 iterations of W=A. Thus, for purposes of the study, W=A takes 0.0594 seconds per execution.

Storage and Recall

Storage in variables A to V takes .017 seconds longer than storage in W to Z. However, recall of A to V takes .016 seconds less than recall of W to Z! The measurement for W=A was .059, for W=W it was .075, A=A took .076 and A=W was .092 seconds. Storage (into W) of a constant (entered by a program statement) takes from .055 to .150 seconds, depending on the number of bytes making up the constant. W= π takes .055 seconds.

If the number being stored is 0, then an additional .003 seconds is required.

Variables specified by subscript require an additional .030 seconds if the subscript has one digit. It takes .038 seconds if the subscript has two digits and .046 for a three-digit subscript.

Addition and Subtraction

Execution time depends largely on the number of shifts required to align the decimal points and on whether the first or second number has the larger exponent.

$W=A+B$ or $W=A-B$ takes from .089 to .300 seconds if the exponent of B is the larger, and from .089 to .367 seconds if the exponent of A is the larger. An additional .001 to .011 seconds is used if shifting is needed to normalize the calculated result.

A negative result requires an additional .002 seconds. A zero result takes an extra .003 seconds.

If variables having longer store or recall times are used, the execution times are modified accordingly.

Multiplication

Execution time depends primarily on the sum of the digits of the multiplier. $W=AB$ takes from .091 to .153 seconds, with .005 seconds

additional if one of the factors is zero. $W=A*B$ takes .009 seconds longer than $W=AB$. If constants or variables having longer store or recall times are used, then execution times are modified accordingly.

Division

Execution time depends considerably on the sum of the digits in the twelve-digit internally calculated quotient. $W=A/B$ requires from .120 to .193 seconds. If constants or variables having longer store or recall times are used, then execution times are modified accordingly.

Logical Calculations

The logical calculations $A>B$, $A\geq B$, $A=B$ and $A<>B$ are evaluated by calculation of $A-B$. The time required is .002 to .003 seconds longer than the subtraction time.

The calculations $A<B$ and $A\leq B$ use the subtraction $B-A$. These take .003 to .004 seconds longer than the subtraction time.

Exponentiation

The calculation of A^B is performed using $10^{B*\text{LOG } A}$. Execution time is the combined time of the calculation of $\text{LOG } A$, multiplication by B and calculation of the required power of 10.

Thus, calculation time for $W=A^B$ is typically .094 seconds when $A=0$. It varies from .279 seconds to approximately 1.0 seconds when A is greater than zero.

Remarks

A REMark statement is bypassed in .012 to .146 seconds, depending on its length.

The GOTO Statement

The statement *GOTO line number* results in a search forward if the line referred to is greater than the current line number. The search is in the reverse direction if the line number is less than the current line number. When searching in the forward direction, execution of this GOTO directive takes .067 seconds plus .185 seconds per 100 bytes traveled. In the reverse direction the traversing time is .147 seconds per 100 bytes.

The statement *GOTO label* results in a search for the referenced line beginning at the start of program memory. This procedure takes .042 seconds plus .182 seconds for each 100 bytes searched. Additionally, each label preceeding the one sought consumes .004 seconds of time.

The GOSUB Statement

The GOSUB directive requires .024 seconds longer than the corresponding GOTO statement. The extra time is required to store the return address and execute the RETURN statement. Note that the RETURN transfer does not have to search memory for an address.

The IF Statement

The statement IF A takes .051 seconds if A is positive and .064 to .200 seconds if A is not positive, depending on the length of the remainder of the line that contains the IF statement.

The statement IF (expression) takes .005 seconds less than execution of the statement $W=(\text{expression})$ if the expression is positive. Otherwise it takes .005 to .141 seconds longer.

Loops

The FOR statement using W , X , Y or Z (not requiring calculation to determine loop parameters) takes from .112 to .161 seconds, with .040 to .066 seconds additional when a STEP is specified. If one of the variables A to V is used as the control variable, the time required is .017 seconds longer.

The NEXT statement, when the mating FOR statement ends a line and when the control variable is W , X , Y or Z , takes .178 seconds for each cycle of the loop. If the FOR statement does not end a line, .014 seconds is added. When A through V serve as the control variable, .050 seconds is added.

Functions

The times noted for functions do not include the time required to store or recall the variable. Thus, the time shown for $\text{ABS } A$ is the difference in time between executing $W=\text{ABS } A$ and $W=A$.

$\text{ABS } A$.011 to .013 seconds.

$-A$.012 to .014 seconds.

$\text{SGN } A$.014 to .016 seconds.

$\text{INT } A$.016 to .019 seconds for A greater than zero.
.020 for $A = 0$.

.017 to .020 for negative integers.

.023 to .036 for negative non-integers. The additional time is used for subtracting 1.

\sqrt{A} .056 to .159 seconds. The time depends mainly on the sum of the digits in the resulting value.

$\text{LOG } A$ The time varies and depends primarily on the number of factors of the form 2, 1.1, 1.01, 1.001, etc., whose product equals the mantissa of A .

For $A>1$, the time is from .108 to about .390 seconds.

For $A<1$, from .035 to .108 additional seconds is required for the division performed in calculating $\text{LOG } A$ in the form $-\text{LOG}(1/A)$.

$\text{LN } A$ The time required is from .035 to .108 seconds longer than for $\text{LOG } A$. This is because of the division needed to calculate $\text{LN } A$ as $(\text{LOG } A)/.434294481903$.

10^A Although not directly accessible to the user, a routine that performs this calculation is used by the computer in calculating $\text{EXP } A$ and A^B .

The time to perform the function varies depending on the number of terms of the form $\text{LOG } 2$, $\text{LOG } 1.1$, $\text{LOG } 1.01$, $\text{LOG } 1.001$, etc., whose sum is the fractional part of A .

For negative values of A , $1/10^{-A}$ is calculated. This division operation adds some .035 to .108 seconds.

$\text{EXP } A$ This is calculated as $10^{A*.434294481903}$. When A is greater than zero, the time ranges from .129 to approximately .360 seconds. For $A=0$ the time is .107 seconds. For A less than zero, an additional .035 to .108 seconds is used.

Summary

With the timing information provided here, the sophisticated programmer can do a great deal to increase the operating speed of many routines.

A MESSAGE FOR AUTHORS

PCN invites the submission of quality programs for pocket computers as well as practical application and tutorial articles of specific interest to users of pocket computers.

PCN purchases all rights to accepted materials at rates ranging from \$50 to \$150 per finished page as it appears in PCN. The payment rate is highly dependent upon the editorial and technical quality, timeliness and general suitability of the material for our readers.

Please double-space all manuscript material. Include a clean printer listing and a tape cassette copy of substantial listings. Be sure and include a self-addressed, stamped envelope if you desire the return of material that has been submitted for our consideration.

We keep all materials, including cassette tapes, and provide an Author's Agreement when a submission meets our requirements.

Since all submitted material must be critically evaluated and reviewed, it may take three to six weeks before any decision regarding a submission is made.

PCN also welcomes letters to the editor that express opinions, provide operating tips, programming tricks or news and general information of interest to fellow users of pocket computers. Selected material from such letters may be published, without monetary remuneration, as a service to subscribers.

Please direct all submissions for editorial review to:

The Editor
POCKET COMPUTER NEWSLETTER
35 Old State Road
Oxford, CT 06483

SUPERMARKET COMPANION

Here are two programs for the harried supermarket shopper. Both are for the Radio Shack TRS-80 or Sharp PC-1211 pocket computers. The first version is for people who do not own or do not want to carry the printer unit into the store. The second version is for those who would like printed output.

Operating Instructions

After loading the program, place the computer in the DEF mode. If you are using the printer version, be sure you have the printer initialized by pressing the ON button twice. Press the SHIFT key and SPC to begin.

Enter the sales tax for your state. For instance, if it is 6%, then just enter 6. If it is 3.75%, then enter 3.75.

Enter the budget allowed for the current shopping trip. Once this has been done, the printer version will output a copy of the data inputted as you shop.

The computer will now display: ENTER AMOUNT. There are several options that may be elected in answering this query:

1. *Non-taxable Item.* Enter the amount of the item you are going to purchase. Example: A loaf of bread is \$1.29. You would enter 1.29.

2. *Taxable Item.* For taxable items, enter the amount as 1.29T. The computer then automatically figures the tax. How does it do that? No real mystery. During the initialization process the variable T was given the value of your tax. If you inputted 6% for your tax, then the variable T now holds the value 1.06. Remember, the computer pre-solves arithmetic before acting upon it in the program. You could answer

Program Supermarket Companion

```
05: REM "<1981> K. SLAUGHTER"
10: " "
(CLEAR & INITIALIZE BLOCK)

20: CLEAR
22: USING"#####.##"           Output Format
                                [IMAGE=5+2]

25: B=10^-20                    B=Exit loop command for
                                input block. (See line 120)

30: PAUSE "SUPERMARKET: COMP 1.0"
40: PAUSE "INITIALIZE. . ."
50: INPUT "SALES TAX (AS 6.5%)? ";X      X=Input variable
60: IF X<0 PAUSE "WHOLE NUMBERS!";GOTO 50
70: IF X=0 PAUSE "WARNING. . .":
    PAUSE "SALES TAX. . .":
    PAUSE "NOT INITIALIZED!";
    GOTO 100
80: T=1+(.01X)                    Tax=1 + decimal value of
                                state sales tax. Example:
                                If state tax X=6.75% then
                                (.01X) = .0675. Add 1 to
                                it gives 1.0675.

90: INPUT "BUDGET LIMIT? ";B          B=Budget Limit
95: IF B<0 BEEP 1;GOTO 90

[INPUT BLOCK]

100: X=0:INPUT"ENTER AMOUNT: ";X      X=Input variable
110: IF X=0 GOSUB 200:GOTO 100        Display running totals
120: IF X=E GOTO 300                  Exit flag
125: M$="":INPUT"ITEM: ";M$           M$=Message variable
130: Z=Z+X                            Z=Total accumulator
140: R=B-Z                            R=Remaining of budget

150: IF R<0 BEEP 1:                  If over-budget display
    PAUSE"WARNING. . .":              warning.
    PAUSE"OVER BUDGET!";
    GOSUB 200

190: GOTO 100                        Return to start of input
                                block for more input.

[SUBROUTINE: DISPLAY ACCUMULATED RESULTS]

200: PRINT"T=";Z;" R=";R:             Display total accumulated (Z)
    RETURN                            and amount of budget remaining (R)

300: FOR I=1 TO 3                    Message flashed on display 3x
310: PAUSE "END. . ."
320: NEXT I

330: GOSUB 200                        Display totals one more time
                                before terminating Program.

340: END
```

Program Supermarket Companion - Printer Version

```
05: REM "<1981> K. SLAUGHTER"
10: " "
(CLEAR & INITIALIZE BLOCK)

20: CLEAR
22: USING"#####.##"           Output format
                                [IMAGE=5+2]

25: B=10^-20                    B=Exit loop command for
                                input block. (See line 120)

30: PAUSE "SUPERMARKET: COMP/PTR 1.1"
40: PAUSE "INITIALIZE. . ."
50: INPUT "SALES TAX (AS 6.5%)? ";X      X=Input variable
60: IF X<0 PAUSE "WHOLE NUMBERS!";GOTO 50
70: IF X=0 PAUSE "WARNING. . .":
    PAUSE "SALES TAX. . .":
    PAUSE "NOT INITIALIZED!";
    GOTO 100
80: T=1+(.01X)                    Tax=1 + decimal value of
                                state sales tax. Example:
                                If state tax X=6.75% then
                                (.01X) = .0675. Add 1 to
                                it gives 1.0675.

90: INPUT "BUDGET LIMIT? ";B          B=Budget Limit
95: IF B<0 BEEP 1;GOTO 90

99: GOSUB 500                       Print initialization data
                                at printer

[INPUT BLOCK]

100: X=0:INPUT"ENTER AMOUNT: ";X      X=Input variable
110: IF X=0 GOSUB 200:GOTO 100        Display running totals
120: IF X=E GOTO 300                  Exit flag
125: M$="":INPUT"ITEM: ";M$           M$=Message variable
130: Z=Z+X                            Z=Total accumulator
140: R=B-Z                            R=Remaining of budget

150: IF R<0 BEEP 1:                  If over-budget display
    PAUSE"WARNING. . .":              warning.
    PAUSE"OVER BUDGET!";
    GOSUB 200

180: PRINT M$                        Print item ID on paper
185: PRINT X                          Print amount on paper

190: GOTO 100                        Return to start of input
                                block for more input.
                                Note: Block exit is to 120

[SUBROUTINE: DISPLAY ACCUMULATED RESULTS]

200: PAUSE"TOTAL=";Z                  Display accumulated total (Z)
210: PAUSE"REMAINING=";R              Display amount of budget
                                remaining (R)
220: RETURN

[EXIT]

300: FOR I=1 TO 3                    Message flashed on display 3x
310: PAUSE "END. . ."
320: NEXT I

330: GOSUB 200                        Display totals one more time
                                before terminating Program.

340: PRINT"*****"                    Print break line at printer
                                (16 characters)

350: PRINT"TOTAL=";Z                  Print total

360: IF R<0 PRINT "OVER BUDGET!";     If over budget . . .
    R=-R:                             Change negative $ to positive
    PRINT R:                           Print amount over
    GOTO 380                           Jump to end

370: PRINT "UNDER BUDGET!";           If under budget . . .
    PRINT R:                           Print amount under

380: PRINT " "                        Indicate end on paper tape
390: PRINT"---END"

395: FOR I=1 TO 3                    Advance paper to bear point
396: PRINT " "
397: NEXT I

400: END

[PRINTER INITIALIZATION]

500: PRINT"PRINTER READY. . ."
510: PRINT " "
520: PRINT " * SUPERMARKET * "
530: PRINT " "
540: PRINT"TAKE: ";X;"$"
550: PRINT " "
560: PRINT"BUDGET= $";B
570: PRINT " "
580: RETURN
```

any numeric input question with a formula such as: 69×1.29 , if you desired. Likewise, the computer recognizes $1.29T$ as $1.29 \times T$ or, in this example, 1.29×1.06 .

3. Press ENTER to Display Totals. If you do not enter a value, the computer will display the accumulated total and the amount of your budget that remains. You may press ENTER as many times as you like.

4. Enter 'E' to End. To exit the data input loop and display final totals, just enter the letter E (followed by ENTER, of course). This operation works the same way that using the letter T does when inputting a price. The variable E was initialized to a value that is not likely to be entered as an item price. When the program finds this unlikely value it transfers control to the termination sequence.

5. Enter a Credit. If you want to reverse a charge, simply enter the value again as a negative number. If the item was taxable, don't forget to add the T suffix. Example: $-1.25T$.

If you use the printer version of the program, a second question termed ITEM is asked. You can thus enter a brief description (maximum of seven characters) of each purchase or credit.

If you go over budget, the computer warns you, but it will continue program execution.

Imagine the surprise on your checker's face when you hand over your itemized purchase slip!

Thanks for this nicely documented package go to: Ken Slaughter, 2916 Bangor Avenue, Highland, CA 92346.

TRIANGLES

When three parts of a triangle are given, including at least one side, this program determines the missing sides and angles, as well as the area of the triangle.

Operating Instructions

Make sure the computer is set to the DEGREE mode. Note that side A denotes the side opposite angle A, etc., in a triangle ABC. Execute the RUN command to start the program. Enter inputs as prompted. If the side or angle requested is not known, key ENTER alone.

An Example

Given side A=5, side C=8 and angle A=35 degrees:

```
SIDE A? 5
SIDE B? -
SIDE C? 8
ANGLE A? 35
ANGLE B? -
ANGLE C? -
```

The calculated results:

FIRST OF TWO SOLUTIONS:

```
SIDE A = 5.
SIDE B = 8.539329288
SIDE C = 8.
ANGLE A = 35.
ANGLE B = 78.40466339
ANGLE C = 66.59533661
AREA = 19.59183225
SECOND SOLUTION:
SIDE A = 5.
SIDE B = 4.567103421
SIDE C = 8.
ANGLE A = 35.
ANGLE B = 31.59533661
ANGLE C = 113.4046634
AREA = 10.47833162
```

Program Notes

Variables I and J are used to keep track of which sides and/or angles were given. If one side and two angles are given, the Law of Sines is used in line 20 - 22. If two sides and the angle opposite one of them are given, the Law of Sines is used in lines 30 - 35. If two sides and the included angle are given, lines 40 - 43 are used. Although the Law

of Cosines could be used here, the method used by the program gives improved accuracy in certain cases. When all three sides are given, the angles are found using the Law of Cosines in lines 50 - 51.

This program submitted by: Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.

Program Triangles

```
10: CLEAR INPUT
   "SIDE A? "IA
   I=1
11: INPUT "SIDE
   B? "B: I=I+2
12: INPUT "SIDE
   C? "C: I=I+3-1
13: IF I=0 GOTO 5
   0
14: INPUT "ANGLE
   A? "A: J=1
15: INPUT "ANGLE
   B? "B: J=J+2
16: INPUT "ANGLE
   C? "C: J=J+3-1
   J
17: IF A(I)=0
   GOTO 30
20: L=90-D-E-F+9
   0: IF L<=0
   GOTO 70
21: A(J+3)=L: FOR
   K=1 TO 3: IF K
   <> I LET A(K)=
   SIN A(K+3)*A
   (I)/SIN A(I+3)
22: NEXT K: GOTO
   60
30: IF J=1 GOTO 4
   0
31: G=A(J): H=A(6
   -I-J): L=A(J+3): IF H*SIN
   L>G GOTO 70
32: M=ASN (H*SIN
   L/G): A(9-I-J)
   =M: IF L+M=1
   80 GOTO 70
33: A(I+3)=180-L
   -M: A(I)=G*
   SIN (L+M)/
   SIN L: IF G>H
   *SIN L IF H>G
   PRINT "FIRST
   OF TWO SOLU
   TIONS: "N=1
34: GOTO 60
35: A(9-I-J)=180
   -M: A(I+3)=M-
   L: A(I)=G*SIN
   (M-L)/SIN L:
   N=0: PRINT "S
   ECOND SOLUTI
   ON: "GOTO 60
40: L=A(I+3): FOR
   K=4 TO 6
41: IF K<>I+3 LET
   G=A(K-3): H=A
   (9-I-K): A(K)
   =90: IF H<>G:
   COS L LET A(K
   )=ATN (G*SIN
   L/(H-G)*COS L
42: IF -A(K) LET
   A(K)=180+A(K
   )
43: NEXT K: A(I)=
   J COS *SIN L*
   SIN L+(G COS
   L-H)*(G COS
   L-H): GOTO 60
50: IF (A+B-C)*(
   A+C-B)*(B+C-
   A)<=0 GOTO 70
51: D=ACS ((BB+C
   C-AA)/2BC): E=
   ACS ((AA+CC-
   BB)/2AC): F=
   ACS ((AA+BB-
   CC)/2AB)
60: K=.5AB*SIN F
   : PRINT "SIDE
   A= "IA:
   PRINT "SIDE
   B= "B: PRINT
   "SIDE C= "C
61: PRINT "ANGLE
   A= "A:
   PRINT "ANGLE
   B= "B:
   PRINT "ANGLE
   C= "C: IF
   PRINT "AREA=
   "K: IF N
   GOTO 35
62: END
70: PRINT "NO SO
   LUTION"
```

CORRECTION TO TIME ADDITION PROGRAM

Alan B. Owens was the first alert reader to point out that line 150 in the Time Addition program printed on page 6 of PCN Issue 10 should read:

```
150 B=10000*DMS B
```

Apparently a zero was dropped in the program listing originally sent to PCN.

A PROGRAM FOR AVIATORS

The accompanying program is designed to take care of the tedious calculations that aviators perform when plotting a flight plan. The program is specifically designed for a Radio Shack TRS-80 or Sharp PC-1211 and should be operated in the DEF mode. Follow these steps:

1. Press SHIFT/A. The computer will ask for the variance from magnetic north. Input the variance in degrees with the appropriate sign. Use a minus sign for east variations. West variations should be entered as a positive value. Press ENTER after keying the data.
2. Input the wind direction as requested.
3. Input the wind velocity in statute miles-per-hour. Since most flight services report wind velocity as knots, a separate conversion routine is provided. Use SHIFT/B to convert knots-per-hour to statute miles-per-hour. (This should be done prior to starting this program.)
4. Enter the true air speed at which you plan to fly your craft.
5. Enter the fuel burn rate for your ship in gallons-per-hour.
6. Enter the true course plotted.
7. Enter the distance of the flight (or flight leg) in statute miles.
8. The computer will output: the wind correction angle (WCA), the true heading at which to fly your ship to account for the wind, the magnetic heading, the projected ground speed in miles-per-hour, the estimated enroute time and the predicted fuel usage.
9. Respond appropriately to the query from the computer concerning calculations for a new flight leg.
10. If you use a printer, all the output will be printed at one time. If you use the PC alone, you must press ENTER after each computed value has been displayed.

This program was submitted by: Gene Robertson, 5757 Southwestern Boulevard, Hamburg, NY 14075.

Program Aviation

```

10:"A"
20:PAUSE "FLIGHT DATA"
30:INPUT "VARIANCE (E-W)";A
40:INPUT "WIND DIRECTION";B
   "WIND VELOCITY";C
50:INPUT "T.A.S";D;"BURN";E
60:INPUT "TRUE COURSE";F
70:INPUT "DISTANCE";G
80:H=ASN((C/D)*SIN(B-F))
90:INPUT "W.C.A";H
100:I=H+F
110:PRINT "TRUE HEADING";I
120:J=I+A
130:PRINT "MAG. HEADING";J
140:K=D*COS(I-F)-C*COS(B-F)
150:PRINT "GROUND SPEED";K
160:L=G/K
170:M=L*60
180:PRINT "TIME ENROUTE";M
190:N=E*L
200:PRINT "FUEL USED";N
210:INPUT "NEW LEG? (YES/NO)";A$
220:IF A$="YES" GOTO 30
230:PRINT "HAVE A GOOD FLIGHT"
240:END
250:"B"
260:PAUSE "KNOTS TO MPH"
270:INPUT "ENTER KNOTS";Q
280:P=Q*1.15
290:PRINT P;" M. P.H."
300:END

```

LINEAR REGRESSION ANALYSIS

When N pairs (X,Y) of numbers are entered, the computer displays the slope and Y-intercept of the *regression line* (also known as the *least-squares* or *trend line*). Other relevant statistical information is also provided.

An Example of Program Operation

Given the pairs (15,37), (18,40), (17,40), (22,46) and (20,48). Start the program by giving the RUN directive. Respond to program prompts as follows:

```

NO OF PAIRS? 5
X? 15
Y? 37
X? 18
Y? 40
X? 17
Y? 40
X? 22
Y? 46
X? 20
Y? 48

```

Information is then provided by the program as follows:

```

REGRESSION LINE:
SLOPE= 1.52739726
Y-INT = 14.09589042

```

Therefore, the equation of the regression line is approximately:

$$Y = 1.527X + 14.096$$

The program continues to yield:

```

VAR.= 5.559360731
STD DEV= 2.357829665

```

These are the variance and standard deviation of the errors, using the sample values. The standard deviation is also known as the *standard error of estimate*. Note: the program calculates these as the *unbiased* variance and standard deviation.

The program yields still more:
CORR.= 8.962837176 E-01

Thus, the coefficient of correlation in the example is .896. The program will now show:

```

SIG LEV FOR LINEARITY,
1.97333026 E-02

```

This may be interpreted as meaning that if there is no linear relationship between X and Y, the probability of a sample such as the one used here for illustration, would be only about .0197. Customarily, a probability less than .05 is considered as sufficient evidence that a linear relationship exists. At this point, if the variable T were displayed, it would be 3.500507443. This is the *Student's T*. With N-2 degrees of freedom, which is 3 in this case, the probability of a value of T this large is .097.

```

The program offers still more:
95% C.I. FOR SLOPE:
MIN, 0.13878151
MAX, 2.91601301

```

The preceding shows a 95 percent confidence interval for the slope of the *population* regression line.

```

There is still more:
95% CI FOR CORR COEFF:
MIN, 6.69917843 E-02
MAX, 9.931811736 E-01

```

The *population* coefficient of correlation is thus most likely between .067 and .993 for the example data.

Now, predictions of Y for a given X may be calculated. Press the ENTER key one more time:

```

GIVEN X? 16 (This number chosen merely as an example.)
Y= 38.53424658

```

Thus, with X=16, the regression-line value of Y is about 38.534. You can get more information about this prediction. Press ENTER again:

```

95% CI FOR MEAN Y:
MIN, 33.80479176
MAX, 43.2637014
95% CI FOR PREDICTED Y:
MIN, 29.66448046
MAX, 47.4040127

```

```

10: INPUT "NO OF PAIRS?" N: A
   =0: B=0: C=0: D
   =0: E=0: FOR Z
   =1 TO N
11: INPUT "X? ";
   X: "Y? "; Y: A=
   A+X: B=B+Y: C=
   C+X*X: D=D+X*Y:
   E=E+Y*Y: NEXT
   Z
12: F=(N-A): G=DN
   -AB: H=EN-BB:
   K=N-2: R=G/F
   H: V=(FH-GG)/
   FKN: S=H/V: M=G
   /F: L=(B-AM)/
   N
13: PRINT "REGRE-
   SSION LINE:"
   : PRINT "SLOP
   E=" : M: PRINT
   "Y-INT=" : L
   : PRINT "VAR.
   =" : V: PRINT
   "STD DEV=" :
   S
14: PRINT "CORR.
   =" : R
20: T=ABS M/(F/
   N): X=K/(K+T)
   : P=K): Y=P
21: IF K>2 FOR Y=
   K-3 TO 2 STEP
   -2: P=1+PXY/(
   Y+1: NEXT Y
22: IF Y=1 LET P=
   .5PT/(K+1):
   GOTO 24
23: P=PTX/(K+R+
   ATM (T/(K+1)
   80
24: P=.5-P: BEEP
   1: PRINT "SIG
   LEV FOR LIN
   EARITY:" :
   PRINT P
25: IF K<6 GOTO 3
   2+K
26: T=((K+.17/K+.
   578)/K+.7359
   )/(K+1.58953)
   /K+2.55595
27: T=((T/K+2.82
   24986)/K+2.3
   7227123)/K+1
   .959963985:
   GOTO 40
28: T=TAN 85.5:

```

```

GOTO 40
34: T=38/778:
   GOTO 40
35: T=3.18244630
   5: GOTO 40
36: T=2.77644510
   5: GOTO 40
37: T=2.57058183
   6: GOTO 40
38: T=2.44691185
   1: GOTO 40
39: T=2.36462425
   2
40: Y=T*(CNV/F): X
   =N-Y: Y=M+Y:
   PRINT "95% C
   .I. FOR SLOP
   E:" : PRINT "M
   IN:" : X:
   PRINT "MAX:
   " : Y
41: IF N=3 GOTO 5
   0
42: Y=LN ((FH+G
   )/(FH-G): Z=3
   .91927969/T
   (N-3): X=(EXP
   (Y-Z)-1)/(1+
   EXP (Y-Z)
43: Y=(EXP (Y+Z)
   -1)/(1+EXP (
   Y+Z): PRINT "9
   5% CI FOR CO
   EFF COEFF:" :
   PRINT "MIN:
   " : X: PRINT "M
   AX:" : Y
50: INPUT "GIVEN
   Y?" : X: Y=MX+
   L: PRINT "Y=
   " : Y: U=X-A/N:
   V=U+U/F+1/N
51: Z=T+JUV: I=Y-
   Z: J=Y+Z:
   PRINT "95% C
   I FOR MEAN Y
   " : PRINT "MI
   N:" : I: PRINT
   "MAX:" : J
52: Z=T/(U+1): V:
   I=Y-Z: J=Y+Z:
   PRINT "95% C
   I FOR PREDIC
   TED Y:" :
   PRINT "MIN:
   " : I: PRINT "M
   AX:" : J: GOTO
   50

```

This information may be interpreted as follows:

When X=16, the "best guess" for Y is 38.534. When X=16 the mean value of Y should be between 33.80 and 43.26. In an individual case, Y may be expected to fall between 29.66 and 47.40.

The value of Student's T being used in determining these confidence

intervals is 3.182446305, which is $t_{.025}$ for 3 degrees of freedom, correct to about 10 digits. This value may be observed by displaying the variable T. (It is calculated in line 30 through 39 of the program.)

This program submitted by: Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.

ANY BASE CONVERSION PROGRAM

Emerich Auerbacher, 41 King Street - Apt. 2, Belleville, NJ 07109 provided this program that converts any number (fractional parts, too) in base A to its equivalent in base B. The bases A and B can range from 2 to 100. A special feature of this program is that the number to be converted is inputted in its entirety instead of digit by digit. This makes the program easy to use. (The secret to this capability is a "parser" in the program that separates the digits and examines them one at a time!)

Note: When the base is in the range 11 to 100, two positions are allocated to each "digit" (symbol). Make sure you remember this when interpreting results.

[Thanks for this version of a program that a number of readers had requested. Your parsing procedure provides a special touch of elegance to its practical application. - N.W.]

Program Any Base Conversion

```

2: PAUSE " BASE
   CONVERSION
   PROG.": USING
4: CLEAR : INPUT
   "KNOWN NUMBE
   R:" : X: "BASE
   OF NUMBER:" :
   B: "BASE DESI
   RED:" : P
6: IF P=10 LET W
   =X: GOSUB 20:
   B=P: GOTO 12
8: IF B=10 LET S
   =X: V=S: B=P:
   GOSUB 180: Y=
   N: GOTO 12
10: W=X: GOSUB 20
   : S=Y: V=S: B=P
   : GOSUB 180: Y
   =N
12: BEEP 3: PAUSE
   "ANSWER >>—
   >"
14: PRINT "A=" : Y
   : " BASE " : B
16: GOTO 4
20: N=0: F=10: IF
   B>10 LET F=E2
25: IF X<1 GOTO 9
   0
30: T=X/(10^N):
   IF T<.1 GOTO
   50
40: N=N+1: GOTO 3
   0
50: IF N=0 GOTO 9
   0
55: S=INT (X/(F^
   (N-2)))
58: IF N<2 GOTO
   90
60: FOR Z=1 TO N-
   2
65: D=INT (X/(F^
   (N-2-Z))) : F*
   INT (X/(F^(N
   -1-Z)))
70: S=(S*B)+D
80: NEXT Z
90: Y=0: IF X=INT
   X: GOTO 150
95: X=X-INT X
100: N=1
105: V=X/F^N: IF V
   -INT V=0 GOTO
   125
110: N=N+1: GOTO 1
   05
125: FOR Z=1 TO N
126: IF Z=1 LET D=
   INT (V/(F^(N
   -1))) : GOTO 1
   35
130: D=INT (V/(F^
   (N-Z))) : F*
   INT (V/(F^(N
   -Z+1)))
135: Y=Y+D*B^(N-Z)
140: NEXT Z
150: Y=Y+S: RETURN
160: IF 10=B LET
   R=10
165: IF B>10 LET R
   =100
190: N=0: T=0
195: IF S<=0 GOTO
   220
200: W=LN S/LN B
205: W=INT (W)
210: N=N+R^W: IF T
   =0 GOTO 220
215: S=S-B^W: T=N:
   GOTO 195
220: RETURN

```

AN ELECTRONIC CARD DECK

Emerich Auersbacher, 41 King Street - Apt. 2, Belleville, NJ 07109, sent in this combination card shuffling and dealing program. To "draw" a card at random from the "electronic deck," just press the ENTER button. (After, of course, you have started the program with the RUN command.) Continue pressing the ENTER key to get as many cards dealt as needed. Once a card has been displayed, it will not show up again unless the deck is reshuffled. The program may be used as is to play a number of card games. Alternately, you might want to use the program as a subroutine in a game of your own design. Note that the program displays a card by both rank and suit such as:

CARD: J OF CLUBS

Program Cards

```

5: USING !BEEP
1: PAUSE " E
LECTRONIC CA
RD DECK": S=5
3
10: A$="DIAMOND"
, B$="HEART",
C$="CLUB", D$
="SPADE"
20: INPUT "PICK
ANY #": G=
ABS (439147+
E6*(G+G)): H=E8
+1
30: IF S>=42 BEEP
1: PAUSE "SHU
FFLE DECK": S
=0: GOTO 100
40: J=23: G=1-
INT (1/H)*H,
R=INT (52*G/
H)+1
45: IF A$(20+R)=
"*" GOTO 40
50: S=S+1, T=R/4,
T=INT (T-
INT T)*4)+1
55: A$(5)=A$(20+
R): A$(20+R)=
"*"
60: BEEP 1: PRINT
"CARD: "A$
(5)" OF "A$
(CT): "S"
70: GOTO 30
100: FOR J=21 TO 2
4: A$(J)="A":
NEXT J: FOR J
=25 TO 28: A$(
J)="2": NEXT
J: FOR J=29 TO
32: A$(J)="3"
: NEXT J
101: FOR J=33 TO 3
6: A$(J)="4":
NEXT J: PAUSE
"SHUFFLING":
FOR J=37 TO 4
0: A$(J)="5":
NEXT J
102: FOR J=41 TO 4
4: A$(J)="6":
NEXT J: FOR J
=45 TO 48: A$(
J)="7": NEXT
J: FOR J=49 TO
52: A$(J)="8"
: NEXT J
103: FOR J=53 TO 5
6: A$(J)="9":
NEXT J: PAUSE
"SHUFFLING":
FOR J=57 TO 6
0: A$(J)="10"
: NEXT J
104: FOR J=61 TO 6
4: A$(J)="J":
NEXT J: FOR J
=65 TO 68: A$(
J)="Q": NEXT
J: FOR J=69 TO
72: A$(J)="K"
: NEXT J
105: RETURN

```

OPERATING TIP SPEEDS INSERTION OF NEW STATEMENTS

You can store a string of dashes using a reservable key, for instance:
SHIFT SPC: -----

When you want to insert something new in an existing line, key in a SHIFT INSert at the appropriate point, then hit SHIFT SPC to drop in the string of dashes. Key in the new characters right over the dashes and eliminate unused dashes with spaces. Submitted by *George Fergus*.



35 Old State Rd, Oxford, CT 06483

FROM THE WATCH POCKET

Sinclair Research Limited reports selling over a quarter of a million of their ZX81 personal computers in less than a year. Sales of it and its predecessor, the ZX80, total more than 400,000 units. While the unit doesn't quite fit in a pocket, it is pretty small and weighs in at just 12 ounces. Apparently a lot of PCN readers are interested in the unit. Of course, a lot of the routines presented herein are readily adaptable to the ZX80 or ZX81. For more information on the ZX81 write directly to Sinclair at 50 Staniford Street, Boston, MA 02114. The phone number is (617) 742-4826. The unit is only sold by mail order at the present time in the United States. They are doing just fine, though, moving some 4,000 units per week!

It didn't take long for my prediction of last month to come true. The Sharp PC-1500 represents a new generation of PC. The price range of \$300.00 is sure to please many. Yes indeed, 1982 is going to be an exciting year for pocket computer enthusiasts!

By the way, if you want to be among the first to get your hands on a PC-1500 in the U.S., better get a call in to *Mort Rosenstein* at Atlantic N.E. Marketing. The phone number is (617) 639-0285. He indicates they are scheduled to get some of the first units released. However, all he can do right now is reserve a unit in your name.

I have received several firm reports that Hewlett-Packard is circulating some pre-production versions of a HHC. This unit sounds intriguing with reports that it will be able to handle 32K of memory. HP has also developed a system that will enable its HHCs and even some of its calculators to interface directly with its larger computers. The portable units can thus acquire data in the field, dump it into a larger machine for processing, and retrieve that information to maintain a current, condensed data base.

Programs for pocket computers fared much better in the popular trade press the past month. 80 Microcomputing has a whole slew of PC programs in the December, 1981 issue. They include a Metric/English Conversion program, a Depreciation Analysis routine, a program that calculates Loan Payment/Interest, another that performs Investment Calculations and a program you can use on visits to the supermarket called Sharp Marketing. While most of these types of applications have been covered in previous issues of PCN, you might want to take a look at another programmer's approach.

Interface Age has several PC articles in its January, 1982 issue. *Bob McElwain* has a well documented version of a game called Bagles Plus One. *David D. Busch* presents a program that keeps track of gasoline usage, miles-per-gallon performance and other statistics when on an automobile trip.

Cass R. Lewart has a PC program in the December 15, 1981 issue of *Electronics*, page 159, that determines suitable scales for computer-generated plots.

Missouri Indexing, Inc., 7 Watch Hill Road, St. Louis, MO 63124, has published a reference called *The Index*. There are some 30,000 entries in it covering some 6 years of articles, editorials and columns from over 800 issues of 45 computer-related publications. The reference was compiled by *William H. Wallace*, an attorney. It utilizes a method of indexing called *Key Word in Context*. Using it you can rapidly locate information for particular types of computers. For pricing and additional information write to the publisher at P.O. Box 301, St. Ann, MO 63074. The phone number is (314) 997-6470.

It is interesting to note that Panasonic is advertising their HHC as *The Link* in magazines such as *Fortune*. As yet, I have not seen them advertising it in the personal computer journals.

In response to requests from readers, we are planning on having some hardware-related articles in the near future. How would you like to use your PC as a simple controller? We have located a clever electronic designer who has done amazing things with a PC in this regards. I hope to be able to provide you with the details of his exciting work in the near future.

Since this is the start of the second year of publishing PCN, I would like to welcome all the new subscribers who are coming aboard with this issue as well as thank the high percentage of Charter Subscribers who are staying with us.

Best wishes for a successful 1982!

— *Nat Wadsworth, Editor*

POCKET COMPUTER

NEWSLETTER



© Copyright 1982 — Issue 12

February

SHARP ELECTRONICS ANNOUNCES THE PC-1500

Sharp Electronics has announced that in February it will begin shipments of its second generation pocket computer. The model has been designated the Sharp PC-1500.

The new unit is housed in a 7-11/16 by 1 by 3-3/8 inch case and has a weight of slightly more than 13 ounces. It will operate for 50 hours of four internal batteries. Power is automatically conserved when the computer is not in use through a shut off feature that powers down the display and CPU while still conserving the contents of memory.

The unit contains a 16 kilobyte ROM that holds the operating system and an advanced version of the BASIC language. A total of 3.5K of RAM is included in the unit. Of this, 900 bytes are reserved for use by the operating system, 1850 bytes serve as the user's program/data area, 624 bytes are used for fixed variables A — Z (or A\$ — Z\$) and 188 bytes are available for "reserve" routines. An optional accessory, the CE-151 Memory Module, provides an additional 4K of user memory. This module is plugged directly into the back of the PC-1500 and does not increase its physical size.

A custom designed CMOS 8-bit processor driven by a 1.3 megahertz crystal clock provides fast operation. The crystal clock circuitry also provides programmable timing functions that can indicate time to the month, day, hour, minute and second.

The PC-1500 is capable of producing audio tones under program control.

The BASIC language interpreter provides the following commands: RUN, NEW, LIST, CONT, TR ON, TR OFF, LOCK, UNLOCK, STATUS and MEM. It supports the statements: INPUT, PRINT, GPRINT, CURSOR, GCURSOR, PAUSE, USING, WAIT, CLS, IF... THEN, STOP, GOTO, ON... GOTO, GOSUB, ON... GOSUB, RETURN, ON ERROR GOTO, FOR... TO... STEP, NEXT, END, DIM, LET, REM, DATA, READ, RESTORE, BEEP, AREAD, ARUN, CLEAR, RANDOM, DEGREE, RADIAN, GRAD, BEEP ON, BEEP

OFF. The following functions are provided: SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, DEG, DMS, RND, SQR, SGN, ABS, INT, PI, LEFT\$, RIGHT\$, MID\$, ASC, VAL, LEN, CHR\$, STR\$, POINT. The interpreter can process two-character variables as well as two-dimensional numeric and string arrays. It also provides the special directives: INKEY\$ and TIME.

The unit sports a QWERTY keyboard with numeric pad and six special programmable keys that may be programmed to provide up to 18 different functions or operations.

A 7 by 156 dot display can be programmed in a graphics mode or in an alphanumeric text mode, producing both upper and lower case.

Sharp supplies the unit complete with a soft carrying case, a set of four batteries, 2 keyboard templates, an applications manual and an instruction manual.

New Printer/Plotter with Dual Cassette Interface Also Announced

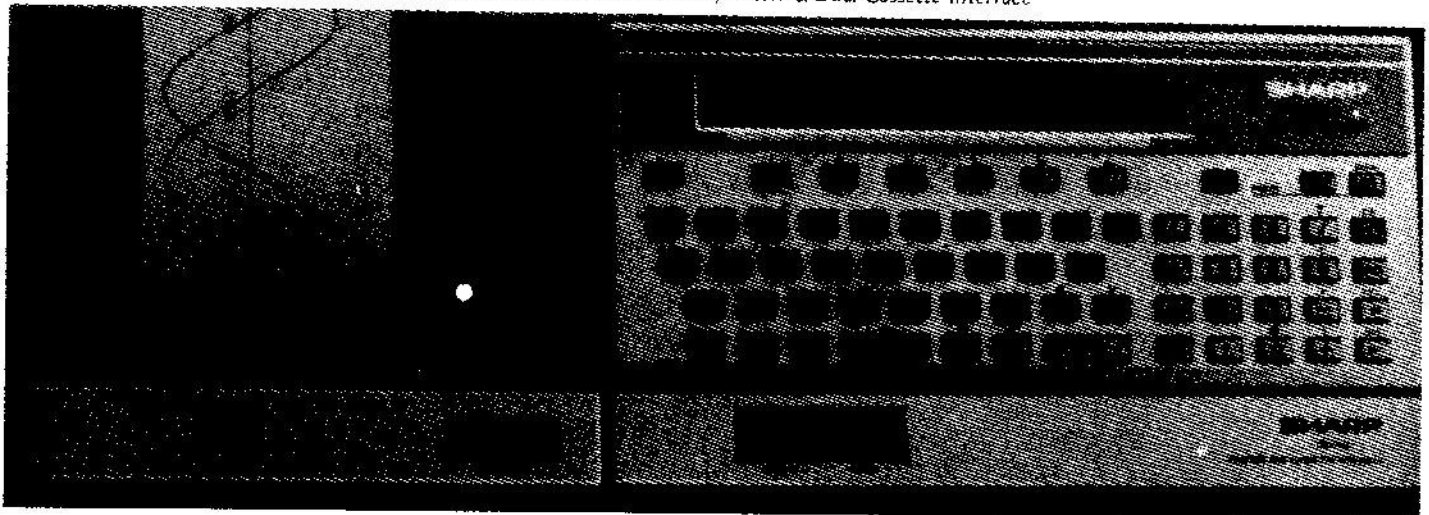
Along with the PC-1500, Sharp has announced a companion unit that is able to print and/or plot in four colors as well as control two audio cassette units!

The CE-150 is being called a Color Graphic Printer by Sharp Electronics. It is capable of operating in two modes. In the text mode, it can output 4, 5, 6, 7, 9, 12, 18 or 36 characters to a line at a rate of up to 11 characters-per-second. Characters can be printed on a character-by-character basis in the colors red, blue, green or black. In the graphics mode, the unit is operated as an X-Y axis plotter system with a resolution of 1/64 inch.

The CE-150 is controlled by the BASIC commands LLIST, TEST, CSAVE, CLOAD, CLOAD? and MERGE, as well as the statements LPRINT, TAB, LF, ROTATE, COLOR, USING, GLCURSOR, SORGN, LINE, RLIN, CSIZE, TEXT, GRAPH, INPUT#, PRINT#, CHAIN, RMT ON and RMT OFF.

The CE-150 weighs just two pounds, runs on rechargeable batteries, and measures 13 by 2 by 4-1/2 inches.

Photo The Sharp PC-1500 Pocket Computer Mounted in the CE-150 Printer/Plotter & Dual Cassette Interface



RADIO SHACK ANNOUNCES NEW POCKET COMPUTER

Radio Shack has obviously teamed up with Sharp Electronics again to provide what it is calling the TRS-80 Pocket Computer Model PC-2.

The new PC measures 1-1/16 by 7-11/16 by 3-3/8 inches. It can operate for up to 50 hours on four internal AA batteries or be operated using an optional alternating current power supply.

The new PC sports a much more powerful BASIC interpreter than its smaller predecessor. The new interpreter and operating software reside in 16 kilobytes of ROM within the PC-2. The interpreter executes 42 statements, 34 functions and 6 commands. Greatly enhanced string handling capability provides for 2-dimensional string arrays having up to 80 characters per element. String operations include: LEFT\$, MID\$, RIGHT\$, LEN, VAL, CHR\$ and STR\$. While being much more powerful, this extended version of BASIC is still compatible with the earlier TRS-80 PC BASIC. Thus programs prepared for the earlier unit can be immediately loaded and executed on the PC-2! The language provided can be extended even further by mating the PC-2 with "intelligent" peripherals.

The PC-2 is supplied with user memory that allows up to 1850 bytes of storage for a BASIC program or data, 600 bytes of fixed data memory and 190 bytes of reserve memory. However, the unit will also accept a plug in memory module that will permit user storage to be expanded by as much as 16K. A variety of such modules have been promised to allow a user to mix both RAM and ROM elements. Initially, a 4K memory module will be available providing the PC-2 with approximately 6.6K of "user" memory.

The new model carries a QWERTY typewriter-format keyboard containing 65 alphanumeric keys. Six of these keys may be programmed to provide up to 18 functions, 18 may be operated as "soft-keys," and 10 may serve as pre-programmed command keys.

The PC-2 also features a liquid-crystal display that will please many users. In standard use, it displays full 5 by 7 dot alphanumeric characters in both upper and lower case. The display can also be addressed as a 7 by 156 dot matrix graphics unit! Thus, special character sets and graphics effects can be generated.

The new unit also has substantial processing speed. A specially designed 8-bit CMOS central processor unit with a 1.3 Megahertz clock provides an operating speed that is roughly ten times greater than the earlier TRS-80 Pocket Computer. The design of the system also permits direct memory access plus the ability to perform maskable and timer interrupts.

Included in the basic unit is a real time quartz clock system. The clock can be accessed from a BASIC program or the keyboard. It can

provide month, day, hour, minute and second timing operations and readouts.

To facilitate its application for use as a process controller, data logger or instrument monitor, the PC-2 is equipped with a 60-pin input/output connector that provides access to address, data, interrupt, timing and CPU control signals.

Radio Shack Computer Centers, stores and participating dealers are scheduled to begin carrying the PC-2 in the second quarter of 1982. The price of the new Radio Shack TRS-80 Pocket Computer Model PC-2 has been announced as \$279.95 in the U.S.

RADIO SHACK ANNOUNCES FOUR-COLOR PRINTER/PLOTTER COMBINED WITH DUAL CASSETTE INTERFACE FOR THE PC-2

The first peripheral announced for the new PC-2 is a combination unit that again provides significant capability over its predecessor.

The printer/plotter portion of the unit uses a drum plotter type of mechanism. Four miniature ball point pens are mounted on a disk. The disk can be rotated to select a particular pen color. The unit is equipped with software intelligence that adds twenty-five BASIC statements and commands to the PC-2 specifically for printing, plotting and graphing operations. Using these special instructions, a user can direct the unit to print text in nine different character sizes ranging from 4 to 36 characters per line. Or, the unit may be directed to draw full X/Y/Z axis graphics using any of the available pen colors.

The printer/plotter uses ordinary 2-1/4 inch wide cash-register or calculator type paper which is inexpensive and widely available!

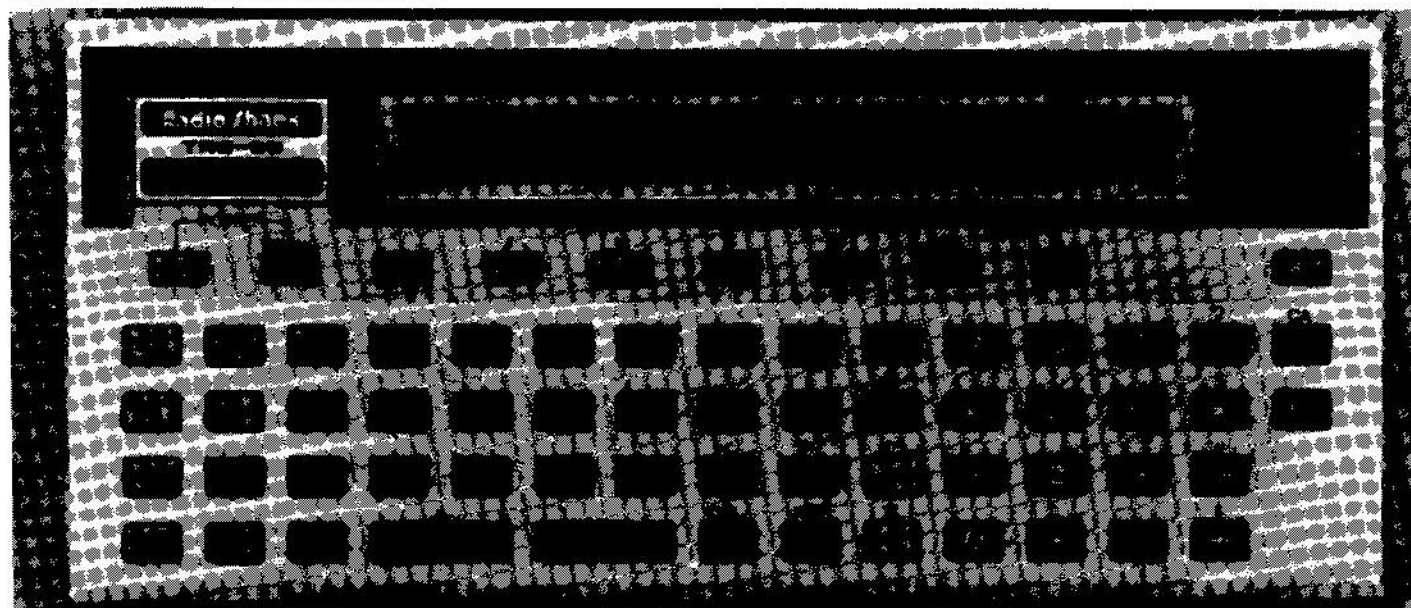
Resolution of the printer/plotter unit is specified at an astounding 0.2 millimeters on a grid of 216 by 512 positions, making it possible to draw amazingly accurate pictorials and artwork.

Along with the printer/plotter is a dual cassette control unit. This permits a user to simultaneously control two separate ordinary audio cassette recorders, thus providing automatic program overlay, data storage, chaining or acquisition of data from one unit while storing updated information on another. Furthermore, this interface can be combined with a future RS-232C interface and modem so that a complete data communications and logging system can be assembled.

Finally, the unit extends the 60-pin input/output connector so that all of the operating signals from the PC-2 can still be accessed when the interface is in use.

The unit is scheduled for U.S. delivery in mid-1982 and has an announced U.S. price of \$239.95.

Photo The Radio Shack PC-2 Pocket Computer



AN INTERPOLATION PACKAGE

The accompanying program provides three methods of interpolation: linear, Lagrangian and first or second order least squares. The least squares routine also provides a curve fit through the given data points. The program is designed to run with the Radio Shack TRS-80 or Sharp PC-1211 PC in the DEF mode.

Linear Interpolation

Use SHIFT/A to perform linear interpolation. Data is inputted as X,Y pairs. The user is then prompted for a value of X and the program determines Y(X). The routine is suitable for extrapolations.

Lagrangian Interpolation

Begin execution of this routine by using SHIFT/B. Input the number of data points (maximum is about 70) and the X and Y values for each point. Input the value of X at which to find Y(X) when prompted. Execution time is a function of the number of points inputted. This type of interpolation fits a polynomial of order N-1 through the N

Program Interpolation Package

```

10 "A":PAUSE "LINEAR INTERPOLATION":
   INPUT "X1=";A,"Y1=";B
20 INPUT "X2=";C,"Y2=";D:G=(D-B)/(C-A)
30 INPUT "X=";X:Y=(X-A)*G+B:PRINT "Y=";Y:
   GOTO 30
40 "B":PAUSE "LAGRANGIAN INTERPOLATION":
   INPUT "# X,Y PAIRS=";E
50 FOR I=1 TO E:B=11+(I-1)*2:D=B+1
60 INPUT "X(I)=";A(I),"Y(I)=";A(D):NEXT I:
   INPUT "X=";A
70 G=0:FOR I=1 TO E:F=1:B=11+(I-1)*2:D=B+1:
   FOR J=1 TO E
80 C=11+(J-1)*2:IF B=C GOTO 100
90 F=F*(A-A(C))/(A(B)-A(C))
100 NEXT J:G=G+F*A(D):NEXT I
110 BEEP 1:PRINT "Y=";G:INPUT "X=";A:GOTO 70
120 "C":CLEAR:PAUSE "LEAST SQUARES":
   INPUT "# X,Y PAIRS=";A
130 FOR I=1 TO A:INPUT "X(I)=";X,"Y(I)=";Y:
   B=B+X:C=C+X*X
140 D=D+X*X*X:E=E+X*X*X*X:F=F+Y:G=G+X*Y
150 H=H+X*X*Y:NEXT I:INPUT "1ST OR 2ND
   ORDER?(1,2)";J
160 IF J=2 GOTO 180
170 K=A*C-B*B:L=F*C-G*B:M=A*G-B*F:N=0:
   GOTO 220
180 K=A*(C*E-D*D)+B*(D*C-B*E)+C*(B*D-C*C)
190 L=F*(C*E-D*D)+G*(D*C-B*E)+H*(B*D-C*C)
200 M=A*(G*E-D*H)+B*(C*H-F*E)+C*(F*D-G*C)
210 N=A*(C*H-D*G)+B*(F*D-B*H)+C*(B*G-C*F)
220 X=L/K:Y=M/K:Z=N/K:BEEP 2:PRINT "A0=";X
230 PRINT "A1=";Y:PRINT "A2=";Z
240 INPUT "X=";A:B=X+Y*A+Z*A*A:PRINT "Y=";
   B:GOTO 240

```

data points specified. Since higher order polynomials may behave strangely, it is best to use as few data points as possible. Extrapolation using this routine can be risky.

Least Squares Interpolation

Use SHIFT/C to perform least squares interpolation and curve fitting. Data is inputted as X,Y values, but the number of points allowed is unlimited! You may then select first (linear) or second (quadratic) order curve fitting. The coefficients of the polynomials are calculated by the program and displayed. Coefficients are assigned as follows:

$$Y(X) = (A0) + (A1)X + (A2)X^2$$

Coefficient A2 will always be equal to zero for a first order fit. After the coefficients have been presented, the user is prompted for a value of X at which the program will find Y(X).

As a rule, a least squares fit does not pass exactly through the data points. Rather, it provides a smooth curve such that the sum of the squares of the distances from the curve to each data point is minimized. Thus, this routine is best when experimental data is likely to have random errors.

Having all three types of interpolation routines available in the PC at one time (with room to spare) is an advantage.

These routines were provided by: Paul T. Meredith, P.O. Box 784, Issaquah, WA 98027.

SUPER-WUMPUS REVISITED

Gary Heidbrink, author of the program, has replied to the suggestions made by Joseph P. Jones, which were published in Issue 10 of PCN:

Do not alter line 5. It is used for other purposes that will limit the scope of the game if modified. Do not change line 105. IF K is the same as IF K=1, but takes less memory. Line 95 does need the colon before GOTO 40 as stated by Mr. Jones. Change the word ARROWS in line 170 to ARROW. Now change line 200 to read:

200 W=M:GOSUB 10:GOSUB 5:T=INT (27-H/5):M=A(T)
This corrects the problem discovered by Mr. Jones while keeping an even spread as to which room the Wumpus will move.

A MESSAGE FOR AUTHORS

PCN invites the submission of quality programs for pocket computers as well as practical application and tutorial articles of specific interest to users of pocket computers.

PCN purchases all rights to accepted materials at rates ranging from \$50 to \$150 per finished page as it appears in PCN. The payment rate is highly dependent upon the editorial and technical quality, timeliness and general suitability of the material for our readers.

Please double-space all manuscript material. Include a clean printer listing and a tape cassette copy of substantial listings. Be sure and include a self-addressed, stamped envelope if you desire the return of material that has been submitted for our consideration.

We keep all materials, including cassette tapes, and provide an Author's Agreement when a submission meets our requirements.

Since all submitted material must be critically evaluated and reviewed, it may take three to six weeks before any decision regarding a submission is made.

PCN also welcomes letters to the editor that express opinions, provide operating tips, programming tricks or news and general information of interest to fellow users of pocket computers. Selected material from such letters may be published, without monetary remuneration, as a service to subscribers.

Please direct all submissions for editorial review to:

The Editor
POCKET COMPUTER NEWSLETTER
35 Old State Road
Oxford, CT 06483

CALENDAR PROGRAM

Input the month and year for any time between March, 1700 and February, 2200 and this program will output a calendar!

```
S M T W T F S
  1  2  3  4  5  6
  7  8  9 10 11 12 13
 14 15 16 17 18 19 20
 21 22 23 24 25 26 27
 28
```

Program Calendar (Printer Version)

```
15 CLEAR:F$=" 1":G$=" 2":H$=" 3":I$=" 4":J$="
   " 5":K$=" 6":L$=" 7":M$=" 8"
20 N$=" 9":O$="10":P$="11":Q$="12":R$="13":
   S$="14":T$="15":U$="16"
25 V$="17":W$="18":X$="19":Y$="20":Z$="21":
   A$(27)="22":A$(28)="23"
30 A$(29)="24":A$(30)="25":A$(31)="26":A$(32)=
   "27":A$(33)="28":A$(34)="29"
35 A$(35)="30":A$(36)="31"
40 INPUT "MONTH:";A,"YEAR:";B:E=621048:
   C=365.25
45 IF A <= 2 LET D=INT(30.6*(A+13))+INT(C*
   (B-1))-E:GOTO 55
50 D=INT(30.6*(A+1))+INT BC-E
55 IF D > 146097 LET D=D-1
60 IF D <= 73047 LET D=D+1:IF D <= 36522 LET
   D=D+1
70 D=D/7:D=INT(7*(D-INT D))+1.49
75 IF (A=4)+(A=6)+(A=9)+(A=11) LET C=30
80 IF (A=1)+(A=3)+(A=5)+(A=7)+(A=8)+(A=10)+
   (A=12) LET C=31
90 IF A=2 LET C=28:IF (B/4=INT(B/4))+(B/E2 < >
   INT(B/E2))=2 LET C=29
95 IF A=2 IF B=2E3 LET C=29
105 FOR E=1 TO D-1:A$(E+36)=" ":NEXT E
110 FOR E=1 TO C:A$(D+E+35)=A$(E+5):NEXT E
120 BEEP 3:PRINT " S M T W T F S"
125 PRINT A$(37);A$(38);A$(39);A$(40);A$(41);
   A$(42);A$(43)
130 PRINT A$(44);A$(45);A$(46);A$(47);A$(48);
   A$(49);A$(50)
135 PRINT A$(51);A$(52);A$(53);A$(54);A$(55);
   A$(56);A$(57)
140 PRINT A$(58);A$(59);A$(60);A$(61);A$(62);
   A$(63);A$(64)
145 PRINT A$(65);A$(66);A$(67);A$(68);A$(69);
   A$(70);A$(71):PRINT A$(72);A$(73):END
```

If you have a printer for your Radio Shack TRS-80 or Sharp PC1211 PC, then use the main listing whose output statements are formatted to "squeeze" each week into a 16-character line. If you want to use the program on your PC without a printer, substitute the statement lines shown in the auxiliary listing.

Thanks for this highly useful program go to: *Emerich Auersbacher, 41 King Street - Apt. 2, Belleville, Nj 07109.*

Program Calendar (Alterations For Use Without Printer)

```
15 CLEAR:F$=" 1":G$=" 2":H$=" 3":I$="
   " 4":J$=" 5":K$=" 6":L$=" 7":M$=" 8"
20 N$=" 9":O$="10":P$="11":Q$="12":R$="
   " 13":S$="14":T$="15":U$="16"
25 V$="17":W$="18":X$="19":Y$="20":Z$="
   " 21":A$(27)=" 22":A$(28)=" 23"
30 A$(29)=" 24":A$(30)=" 25":A$(31)=" 26":
   A$(32)=" 27":A$(33)=" 28":A$(34)=" 29"
35 A$(35)=" 30":A$(36)=" 31"
105 FOR E=1 TO D-1:A$(E+36)=" " :NEXT E
120 BEEP 3:PRINT " S M T W T H F S"
```

PROGRAM CALCULATES RELATIVE HUMIDITY & DEW POINT

This program determines the relative humidity (in percent of saturation) and the dew point. The raw inputs to the program are the temperatures of a combination wet bulb/dry bulb thermometer as well as the atmospheric pressure. This program is derived from work initially conducted by Dr. Scott Williams of Annapolis, Maryland.

You can construct a simple dry-bulb and wet-bulb psychrometer at home. Separate two thermometers by an inch or more using a plastic or wood spacer. Cover the bulb of one of the thermometers with a closely-fitted "sock." Wet the sock thoroughly with distilled water. (You can get distilled water as run-off from your air conditioner.) Do not use water straight from your drinking tap. Doing so can result in large errors.

Attach the thermometers to a short piece of string and swing them rapidly through the air for about 30 seconds in order to take a set of readings.

Remember that the dew point (the temperature of a surface at which condensation will begin to occur) is a better index than the relative humidity, since it usually does not vary as the surroundings warm or cool.

```
10 INPUT "DRY BULB TEMP, C";A
20 INPUT "WET BULB, C";B
30 INPUT "PRESSURE, MB";C
40 H=B:GOSUB 500
50 E=J*(0.00066*(1+0.00115*B)*C*(A-B)
60 H=A:GOSUB 500
70 D=E*100/J
80 PRINT " R.H. ",D
90 F=LOG (E/6.1078)
100 G=237.3*F/(7.5-F)
110 PRINT "DEW PT",G
120 END
500 J=6.1078*10^((7.5*H)/(237.3+H))
510 RETURN
```

To use the program, simply type RUN and input the data requested. Both relative humidity and the dew point will be displayed.

This program provided by: *James K. Sparkman, Jr., 3917 Birds-ville Road, Davidsonville, MD 21035.*

LOW PASS FILTER DESIGN

The program shown here will aid in the choosing of component values for a second order low pass filter. It will also calculate the theoretical frequency response of the filter so that you can gauge its expected performance.

The program was submitted by: *Hank Librach, 52 Bulkley Drive, Fairfield, CT 06430.*

The program will prompt the user to input the desired filter pass-band gain, cutoff frequency in Hertz and the desired damping. You also select a value for capacitor C_2 (as shown in the accompanying circuit diagram). The remaining components are then determined by the program.

The filter transfer function and other relationships assumed by the design program are:

$$\frac{E_o}{E_{in}} = \frac{1/(R_1 R_3 C_1 C_2)}{s^2 + (1/R_1 + 1/R_2 + 1/R_3)(1/C_1)s + 1/(R_2 R_3 C_1 C_2)}$$

$$K(s) = \frac{-H_0 \omega_0^2}{s^2 + \alpha \omega_0 s + \omega_0^2}$$

$$\alpha = 2\zeta \quad \zeta = \text{damping factor}$$

Formula parameters are assigned to program variables as follows:

H_0 — H
 F_0 — F
 ζ — Z
 C_1 — A
 C_2 — B
 R_1 — C
 R_2 — D
 R_3 — E

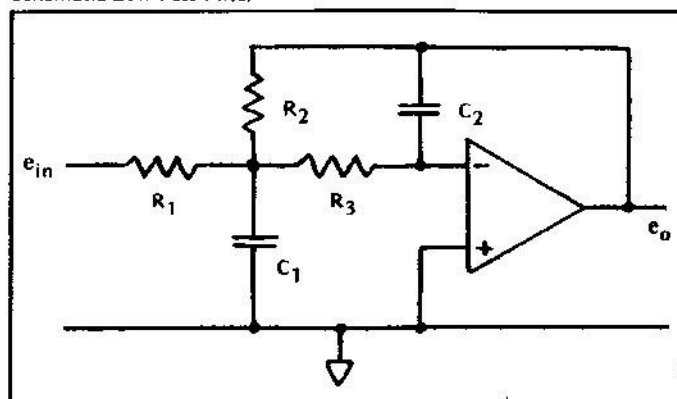
Program Low Pass Filter Design

```

10:PAUSE "LOW P      "#.#^": "C1=
   ASS FILTER"      "#A
20:INPUT "PASSB     105:PRINT "C2=";
   AND GAIN=?";      B
   H                110:PRINT "R1=";
                   C
30:INPUT "CUTOFF    115:PRINT "R2=";
   F FREQ. HZ.=     D
   ? "IF           117:PRINT "R3=";
                   E
40:INPUT "DESIR     120:PAUSE "FREQ.
   ED DAMPING=?     RESPONSE:"
   "#Z             130:INPUT "INPUT
                   F="#IF
50:INPUT "CHOOS     135:J=4+I+I+F+F;
   E VALUE FOR      M=ABS (N-J)
   C2=?"FB          140:K=-G/(J*(M+M+
                   J*I+I))
60:A=(1+H)*B/(Z     145:L=20*LOG (
   *2)              ABS K)
70:D=Z/(2+I*F*B    150:PRINT USING
   ):C=D/H;E=D/     "#.#^": "K="
   (H+1)            IK
80:G=1/(C*E*A*B    155:PRINT "K(CB=
   ):I=(1/C+1/D     )"#L
   +1/E)/A;N=1/     160:GOTO 130
   (D*E*A*B)
90:PAUSE "COMPO
   NENT VALUES
   ARE:"
100:PRINT USING

```

Schematic Low Pass Filter



POLYNOMIAL ARITHMETIC

You can use this program to multiply or divide polynomials. It will also compute integer powers of polynomials.

Instructions

Begin as follows with the PC in the DEF mode:

- SHIFT M to multiply polynomials.
- SHIFT D to divide.
- SHIFT N to find Nth power. (N must be an integer > 1.)

Next, input the data for each polynomial degree as prompted. The coefficients of the polynomial should be entered in order of decreasing powers. If you lose your place, just press the ENTER key without any data. This will give you a prompt of the degree of the term you should be entering.

As an example, consider the division of the polynomial: $6X^6+7X^5+11X^4+30X^3+10X^2+3X-7$ by the polynomial: $3X^2+5X+2$. First press SHIFT D, then enter data in response to the prompts as shown:

```

NUM DEG? 6
? 6
? 7
? 11
? 30
? 10
? 3
? -7
DEN DEG? 2
? 3
? 5
? 2

```

The results are then displayed as:

```

QUO:
DEG 4, 2
DEG 3, -1
DEG 2, 4
DEG 1, 4
DEG 0, -6
REM:
DEG 1, 25
DEG 0, 5

```

This shows that the quotient is $2X^4-1X^3+4X^2+4X-6$, with a remainder of $25X+5$.

As another example, consider finding: $(2X^2-3X+5)^5$. In this case, start by pressing SHIFT N, then continue as follows:

```

DEGREE? 2
? 2
? -3
? 5
POWER? 5

```

It will take about a minute and a half on a Sharp PC-1211 for the result to be obtained. The answer in this case is: $32X^{10}-240X^9+1120X^8$

$-3480X^7 + 8210X^6 - 14643X^5 + 20525X^4 - 21750X^3 + 17500X^2 - 9375X + 3125$.

Restrictions

Calculated polynomials must be of degree less than 100.

Also, when multiplying polynomials, two times the degree of the first polynomial plus the degree of the second, must not exceed 117.

And, for division, the sum of the degrees of the numerator and the denominator must not exceed 117.

Plus, when calculating Nth powers, the product of (N+1) times the degree of the given polynomial must not exceed 118.

An error message results if these restrictions are not followed.

This program submitted by: Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.

Program Polynomial Arithmetic

```

10:"N" CLEAR :
  INPUT "DEGREE
  E? " : D=A+5:
  GOSUB 30:
  INPUT "POWER
  ? " : C=FOR A=
  5 TO D: A(A+D+
  1)=A(A): NEXT
  A: FOR A=2 TO
  C
11: FOR B=D+D+4
  TO D+6 STEP -
  1: FOR C=B+1
  TO C+D: A(C)=
  A(C)+A(B)A(C
  -B+4): NEXT C
  : A(B)=0: NEXT
  B
12: FOR B=B TO A+
  D+7: A(B)=A(
  B+1): NEXT B:
  NEXT A: A=B+6
  : B=B-1: BEEP
  1
20: FOR D=A TO B:
  C=B-D: PRINT
  "DEG" USING
  "###": "I", "
  USING A(C)
  : NEXT D: END
30: B=A+D: FOR A=
  A TO B
31: INPUT A(A):
  NEXT A:
  RETURN
32: C=B-A: PAUSE
  "DEGREE " : C:
  GOTO 31
40: "M" CLEAR :
  INPUT "DEG O
  F 1ST? " : E:A
  =6: D=E: GOSUB
  30: INPUT "
  DEG OF 2ND?
  " : D: A=E+7:
  GOSUB 30: FOR
  A=A TO E+7
  STEP -1
  41: FOR B=A+1 TO
  B+E: A(B)=A(B
  )+A(A)A(B-A+
  5): NEXT B: A(
  A)=0: NEXT A:
  A=E+8: B=B+D:
  BEEP 1: GOTO
  20
50: "D" CLEAR :
  INPUT "NUM D
  EG? " : D: A=6:
  E=D+7: GOSUB
  30: INPUT "DE
  G N DEG? " : D: A
  =E: GOSUB 30:
  PRINT "QUQ:
  51: IF D+7>E
  PRINT O: A=6:
  GOTO 54
52: FOR A=6 TO E-
  D-1: C=A(A)/A
  (E): IF D FOR
  B=A+1 TO A+D:
  A(B)=A(B)-CA
  (B+E-A): NEXT
  B
53: B=E-D-A-1:
  PRINT "DEG":
  USING "###":
  B: "USING
  54: NEXT A: A=
  A+95N D
  54: IF D PRINT "R
  EM: " : B=E-1:
  GOTO 20

```

HAVE YOU HEARD...

About the General who asked his PC the all important question, "Will we win the next war or will we lose it?"

The little PC churned away for a few minutes, then finally displayed its answer: "YES!"

"YES what?" demanded the furious General.

"YES SIR!" responded the PC.

ACCOUNTS RECEIVABLE

This program was provided by: Mr. H. Nath, 101/H Block 'F', New Alipore, Calcutta - 700 053, India. He supplements the program with the following information.

Purpose

This program is designed to assist me during visits to business customers. It will display up to six bills for each of three companies. It can also display sub-totals by customer. Since it may be used with a printer, I sometimes use it to present a customer with an on-the-spot bill! The information held in the PC may also be saved on a tape cassette and recalled for later use or updating.

Operation

The program should be executed with the Sharp PC-1211 or equivalent PC in the DEF mode. Press SHIFT/Space to display a menu of program options.

SHIFT/A is used to enter data, either from the keyboard or from a tape, as well as to save data on a tape.

The initial query by the computer enables the operator to select the desired option: to load data from a tape, save data to a tape or input data from the keyboard. Respond to the query:

LD/SVE DATA TAPE? L/S/N

with the letter N followed by the ENTER key if you simply want to input data from the keyboard.

Respond to the query for a company name using a maximum of 7 characters. Press ENTER.

The display will flash:

NEW FILE

OLD FILES ARE

and then display the number 1. This signifies that it is ready for the first entry for this company. (It would also show the number of the first bill if it had been assigned previously.)

Press ENTER. The computer will query:

ALTER? Y/N

If you input Y at this point, the program will clear out all of the remaining memories for this company. This can take as long as 13 seconds, depending on which bill you are examining.

Next, the program will ask you for the bill numbers, dates and dollar amounts. Enter these in sequence as prompted. Up to six sets of entries may be made under a company name. If you do not want to enter as many as six bills, then press SHIFT/X to exit the routine.

When six sets of entries have been made, the program will automatically flash:

NO ROOM

and exit to the menu portion of the program.

By the way, after asking for each item of data, the computer will display:

ERROR? SHFT Z

Press the ENTER key if you have not made an error in the entry. Press SHFT/Z if you want to re-enter the data. [I found this feature quite irritating and would suggest that line 2 be changed to a PAUSE statement or eliminated. The correction capability would still be available, but the tediousness of having to press the ENTER key an extra time after every single entry is then eliminated. - N.W.]

Similar entries can be made for two more companies.

Once data has been loaded, you use the SHIFT/B option to output the information. When this option is selected, you once again have the opportunity to load or save the data using a tape cassette machine.

Next, the program asks for the name of the company you wish to review. Enter the name of a company whose bills are in the PC. The program will check to make sure the name exists in its files. If not, the query repeats. Once a valid company name has been inputted, the program asks:

PRINTOUT Y/N?

Answer Y if you are using a printer with your PC. (Insert the name of your own business in line 50 of the program!) The program will now display the bill numbers, dates and amounts for the company you

Program Accounts Receivable

```

1  "X" END
2  PRINT "ERROR? SHFT Z":RETURN
3  G=G+1:RETURN
4  INPUT "ALTER? Y/N ";F$:RETURN
5  INPUT "CO. NAME? ";E$:RETURN
6  PAUSE "NO ROOM":GOTO 38
7  IF F$="Y" PRINT " "
8  RETURN
9  IF G=2 LET G=9
10 IF G=3 LET G=27
11 IF G=4 LET G=45
12 RETURN
13 FOR A=1 TO 18:A(G)=0:GOSUB 3:NEXT A:
   G=G-18:RETURN
14 INPUT "LD/SVE DATA TAPE? L/S/N ";H$:
   IF H$="L" INPUT # "RCD":RETURN
15 IF H$="S" PRINT # "RCD"
16 RETURN
18 "Z" GOTO H
20 "S" PRINT "SUB-TOTAL=$ ";A:GOTO 57
22 IF G=27 GOTO E
23 IF G=45 GOTO E
24 IF G=63 GOTO E
25 RETURN
30 FOR G=2 TO 4:IF E$=A$(G) GOTO A
32 NEXT G:RETURN
34 USING :PRINT H;" ";A$(G):GOSUB 4:IF F$=
   "Y" GOTO A
36 RETURN
38 " " PRINT "A/CS RECEIVABLE"
39 PRINT "SHFT A: INPUT"
40 PRINT "SHFT B: OUTPUT"
41 PRINT "SHFT S: SUB-TOTAL"
42 PRINT "SHFT X: EXIT":GOTO 38

44 "B" GOSUB 14
46 GOSUB 5:A=50:GOSUB 30
48 PAUSE "NO SUCH FILE":GOTO 46
50 INPUT "PRINTOUT? Y/N ";F$:IF F$="Y"
   PRINT "XYZ INC.":GOSUB 7:PRINT "A/C.":
   A$(G)
52 GOSUB 9:A=0:E=61:USING "#####. ##"
54 GOSUB 7:PRINT "BILL NO.":A$(G):GOSUB 3
55 PRINT "DATE ";A$(G):GOSUB 3
56 A=A+A(G):PRINT "$ ";A(G)
57 IF G=62 IF A(G) GOTO 61
58 GOSUB 3:H=G+2:IF A(H)=0 GOTO 61
60 GOSUB 22:GOTO 54
61 IF F$="Y" GOSUB 7:PRINT "TOTAL=":
   PRINT "$ ";A:END
62 PRINT "TOTAL=$ ";A:END
64 "A" GOSUB 14
66 GOSUB 5:A=78:GOSUB 30
68 PAUSE "NEW FILE":PAUSE "OLD FILES ARE"
69 H=1:G=2:A=86
70 GOSUB 34
74 H=H+1:GOSUB 3:IF G=5 GOTO 6
76 GOTO 70
78 PAUSE "OLD FILE":PAUSE "BILL NO.S ARE":
   GOSUB 9:H=1:A=88
80 GOSUB 34
84 G=G+3:H=H+1:E=6:GOSUB 22:GOTO 80
86 A$(G)=E$:GOSUB 9:GOSUB 13
88 H=88:INPUT "BILL NO.? (7 CHAR) ";A$(G):
   GOSUB 2:GOSUB 3
90 H=90:INPUT "DATE? (MMDD-YY) ";A$(G):
   GOSUB 2:GOSUB 3
92 H=92:INPUT "$=";A(G):GOSUB 2:GOSUB 3
94 E=6:GOSUB 22:GOTO 88

```

	1ST COMPANY						2ND COMPANY						3RD COMPANY					
STORAGE SEQUENCE	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
A\$(...) BILL NUMBER	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
A\$(...) DATE	10	13	16	19	22	25	28	31	34	37	40	43	46	49	52	55	58	61
A(...) AMOUNT	11	14	17	20	23	26	29	32	35	38	41	44	47	50	53	56	59	62

requested. When all the bills for a firm have been displayed, the program will give a total of all the bills.

If you are not using a printer, press the ENTER key after each item has been displayed.

You can obtain interim totals. When the program is displaying the amount of an invoice, press SHIFT/S. If the program was displaying the amount of the third bill for a company, pressing SHIFT/S would result in a display of the sub-total of the first three bills for that firm. Press

the ENTER key to continue outputting data after using the SHIFT/S option. (The SHIFT/S option should only be used when the amount of a bill is being displayed.)

This program, as shown in the accompanying listing, leaves 317 steps of memory. At least 288 steps are needed to hold the data for 18 bills. Variables A through H are used by the operating portion of the program. Data elements are stored in array elements as shown in the accompanying table.

SHIFTY WORDS

David Motto, 3639 Roosevelt, Jackson, MI 49203, submitted this short entertainment program.

What do the words PIT, ATE and HAL have in common? They can all be traced back to IBM. No, that large computer company didn't coin the use of those words. They are derived by shifting the letters in 'IBM' through the alphabet. For example, if I becomes J, B becomes C and M becomes N, then IBM is transformed to JCN. Continuing this process yields some interesting words.

For instance, JOHN becomes PUNT. RUN transforms to FIB. OPEN becomes STIR. Want to take a look at the situation? Just key this program into your 'CAB' and plug in some words!

Operation

Place the PC in the DEF mode and press SHIFT/Space. Key in a word that you want to try shifting. A word must not exceed 10 characters. Also, note that you key in one letter at a time, pressing ENTER after each character. Pressing ENTER alone terminates the word. After the original word has been inputted, the program automatically shifts all the letters and displays the new arrangement. If the display is too fast for your tastes, change the PAUSE statement in line 6 to PRINT. You will then have each arrangement displayed until you press the ENTER button.

Program Shifty Words

```

1: "CLEAR :           : RETURN
   PAUSE "INPUT
   A WORD"           : RETURN
2: K=K+1: INPUT       20: "J"A$(K)="K"
   "LETTER? "IA       : RETURN
   $(K): IF A$(K)     21: "K"A$(K)="L"
   )=":" THEN 4        : RETURN
3: IF K<11 THEN       22: "L"A$(K)="M"
   2                  : RETURN
4: A$(K)=O:L=K-1      23: "M"A$(K)="N"
5: FOR M=1 TO 26      : RETURN
   : FOR K=1 TO L      24: "N"A$(K)="O"
   : GOSUB A$(K)        : RETURN
   : NEXT K            25: "O"A$(K)="P"
6: PAUSE A$(B$)       : RETURN
   C$(D$)E$(F$)       26: "P"A$(K)="Q"
   G$(H$)I$(J$)       : RETURN
   NEXT M             27: "Q"A$(K)="R"
7: GOTO 1             : RETURN
11: "A"A$(K)="B"      28: "R"A$(K)="S"
   : RETURN           : RETURN
12: "B"A$(K)="C"      29: "S"A$(K)="T"
   : RETURN           : RETURN
13: "C"A$(K)="D"      30: "T"A$(K)="U"
   : RETURN           : RETURN
14: "D"A$(K)="E"      31: "U"A$(K)="V"
   : RETURN           : RETURN
15: "E"A$(K)="F"      32: "V"A$(K)="W"
   : RETURN           : RETURN
16: "F"A$(K)="G"      33: "W"A$(K)="X"
   : RETURN           : RETURN
17: "G"A$(K)="H"      34: "X"A$(K)="Y"
   : RETURN           : RETURN
18: "H"A$(K)="I"      35: "Y"A$(K)="Z"
   : RETURN           : RETURN
19: "I"A$(K)="J"      36: "Z"A$(K)="A"
   : RETURN           : RETURN

```

FROM THE WATCH POCKET

How marvelously exciting it is to see the specifications on the Sharp PC-1500/Radio Shack PC-2. Note that while the two units are similar in functional capability, there are some cosmetic differences. I observed particularly that the Radio Shack PC-2 has the ENTER key just to the right of the SPACE key. I think they may regret that positioning. Touch typists are quite likely to strike the ENTER key accidentally from force of habit, forgetting that it is not the space bar on a typewriter. Time will tell.

Functionally the units represent dreams coming true for many users. The BASIC interpreter has the additions necessary for a serious unit. Enough memory (16K) will eventually be installable in the pocket unit so that it can be highly functional, say, on a business trip, while holding a significant amount of data and perhaps two or three programs that can operate on that data. This is truly a pocketable unit with enough power to serve in a completely stand-alone situation.

Perhaps the most significant aspect is that the units have been designed with fully accessible input/output pathways. The direct memory access (DMA) capability is important, too. This means that the unit will be capable of operating as the central portion of a truly flexible, portable, individualized, personal computing system.

Individuals will be able to plug their own software-customized units into a variety of commonly shared applications/systems.

For instance, Radio Shack has already promised capability for the PC-2 to operate as a communications terminal. That means you will be able to plug into Source, Compuserve or any of a host of other networks, from anywhere that there is a telephone. With 16K of memory in your pocket, you will be able to download a goodly amount of data along with, say, a sophisticated analysis program. You might do this at the start of an airplane trip. Use the program and data as you traveled. Then offload the results at the end of your journey. What a way to increase your efficiency.

DMA capability means it is only a matter of time before there are intelligent disk systems for you to plug into. Remember, the PC-1500 is designed with software "hooks" so that an external "intelligent" device can expand the vocabulary of the BASIC language that is in the PC. Now, the essence of this capability is that you can considerably lower the individual cost of computing power by sharing the peripherals amongst a large group of users. You can put a few disk systems in an office. Perhaps one for each type of commonly needed task. Individual users then plug their own PCs into the appropriate disk system as they need the data/programs stored therein.

Make no mistake about it, there will be myriads of devices coming for you to plug your PC into. For instance, sources indicate that Sharp is working on devices such as a television interface. But even if Sharp or Radio Shack didn't provide such attachments, with those input/output (I/O) connectors accessible, lots of firms will soon be hard at work designing all kinds of useful gadgets for use with this new type of powerful pocket computer.

This and That

Radio Shack has reduced the regular price of its "old" TRS-80 PC to \$169.95. I have seen discount houses selling the Sharp PC-1211 for less than \$150.00. Also, look for deals where the CE-121 cassette interface comes with the PC at no charge.

I know of a number of PC users, yours truly included, who have several PCs around at all times. It is easier to keep several around loaded with all the programs one uses frequently, rather than to have to load in a new program three or four times a day. As the prices of PCs fall, this practice will undoubtedly increase.

80 Microcomputing continues to be one of the few trade magazines that regularly publishes PC articles. The January issue had three articles including one by *Guerrri F. Stevens* describing an exercise log program. Another program that helps keep track of a stock market portfolio and a program that determines the day of the week for any date since September 14, 1752, were also in the issue.

The March issue of *Popular Computing* has a nice multi-page article introducing and describing the original Radio Shack TRS-80 Pocket Computer. Better late than never, eh?

— Nat Wadsworth, Editor

POCKET COMPUTER

NEWSLETTER



© Copyright 1982 — Issue 13

March

THE SHARP PC-1500

An actual Sharp PC-1500 Pocket Computer arrived at PCN headquarters just a few hours before this issue was scheduled to go to press. This report, thus, represents a quickly compiled summary of our initial findings and reactions.

It Is Nice

This is definitely a second-generation PC! One of the first features one notices is the improved ease and speed with which program editing can be accomplished. A lot of the improvements are subtle but definitely noteworthy.

For instance, there are just two basic machine modes: RUN and PROgram. You only have to hit the mode key once to switch from the one to the other, instead of cycling through four modes as in the PC-1211. (There is a third mode, the RESERVE mode. However, since this is generally used far less often, this mode is achieved by pressing the SHIFT/MODE keys.)

Editing execution is virtually instantaneous. If you want, say, line 40 listed in a 40 line program, that line comes up in the blink of an eye. There is no three or four second delay to access the line. The cursor and line scroll operations are also much speedier. You can zip from beginning to end of a 40 line program in just a few seconds using the line scroll. Character scrolling across a line is also faster than on a PC-1211. This snappier editing operation is certain to be appreciated by programmers.

Perhaps the nicest operating/editing feature initially noticed is the implementation of "softkeys." There are six keys directly under the display that are user-programmable. They can act in a manner similar to the "defined" keys on a PC-1211, but with several significant advantages. These keys are "programmed" by placing the PC in the RESERVE mode. However, each key can actually be programmed for up to three different options. The desired option is selected by a Reserve Select key which is used to cycle to the desired option. The status of the Reserve Select key is denoted by indicators on the display which show the Roman numerals I, II or III. The following characteristics make these software keys particularly powerful.

First of all, they only require the pressing of a single key, instead of first having to press the SHIFT button.

Now, these keys can be used in the editing mode, for instance, to insert often used statements in a program under development. Or, they can be used to place a command on the display. However, through the use of a special terminating character (@), these keys can be set up to actually execute a command. For instance, setting up a reserve key to contain the directive:

RUN 100 @

results in your having the capability to execute a program starting at line 100 simply by pressing one key. (This same operation would have taken three keystrokes on a PC-1211: pressing the SHIFT key, the defined key and then the ENTER key.)

But wait, that is not all! With six softkeys, each of which can carry three functions or operations, it could be rather difficult to remember what had been assigned to all the keys. No problem. Whenever you assign capabilities to softkeys you can also create labels that appear

directly above the keys on the display! To access those labels you just press the key marked RCL (recall)! It really brings a whole new level of operating comfort to the PC.

Here Comes Speedy

By PC standards, the Sharp PC-1500 is fast. Three benchmark tests conducted against the PC-1211 showed program execution to be 7 to 8 times faster!

A True Graphics Display

People are going to have a lot of fun with this display. Even in the text mode it has capabilities and advantages over the PC-1211. For instance, using the PAUSE statement (in a repetitive loop) simply causes the display to be updated. There is no more annoying blanking (unless you program that to occur). You can easily locate text anywhere along the display using the CURSOR directive.

Then, of course, there is the ability to address all of the individual points (7 by 156 dots) in the display matrix. Virtually any pattern desired can be obtained through the use of the G_CURSOR and the G_PRINT statements. The status of a dot can be obtained by using the POINT directive. A test routine using a loop to draw a line all the way across the 156 dots of the display executed in a little under 5 seconds. That amounts to about 30 pixels a second when the matrix is accessed on a point basis. Graphics can actually be drawn on a column basis, so in this mode some 200+ pixels-per-second can be manipulated. The impression of speed, in for instance, animations, can be enhanced by skipping columns or rows as an object is moved.

The display can be switched to lower case letters using the SML (small) key. And, with the graphics capability, you can actually design your own character set for special applications, if desired.

The Clock

The PC-1500 has a special function named TIME. You can set the time to the month, day, hour, minute and second. Once set, this built in clock is automatically maintained by the PC. You can then access the time as a function under program or keyboard control. This has obvious applications in programs such as appointment organizers and reminders. You can periodically check the time and then issue an audio signal when appropriate.

Music To Your Ears

The BEEP function is greatly expanded over that of the PC-1211. Not only can the number of beeps be programmed, but also the tone (256 frequencies) and the duration (from just a "pip" to more than a minute). Though the volume is discrete, it is easy to program special annunciator effects and those with the interest can make this little PC play music.

There Is Lots More

It is all described in two manuals that come with the unit. A 160+ page Instruction Manual with numerous examples, plus a 200+ page Applications Manual that contains about 50 complete programs.

The Initial Summary

It is nice.

DETERMINE TAB SETTINGS AUTOMATICALLY

If you have ever had to type charts or data tables, you know how frustrating it can be to arrange columns on a page neatly. Where should you set the tabs? How wide should the margins be? Usually it takes several trial runs before things fit properly. But, with your pocket computer and this program, you can literally take the guesswork out of designing tables.

This program allows columns to be the same or different sizes. The program also compensates for either elite type (102 characters across an 8-1/2 inch wide sheet of paper) or pica type (85 characters on a standard page). You can even vary the size of the paper if you want. Simple modifications will also enable the program to be used with font sizes other than those mentioned.

What You Need to Know

To use the program, you need to do just a little preliminary thinking. The program will want to know:

1. The number of columns across the sheet.
2. How many characters in each of the columns — so that a total in all of the columns can be obtained.
3. The horizontal size of the paper. (The program defaults to 8-1/2 inches if you do not specify a size.)

Here Is an Example

For this illustration there will be 6 columns across the paper. For the sake of simplicity all the columns will have 6 character positions. The paper width will be the standard 8-1/2 inches.

Place the PC in the DEF mode. Press SHIFT/C. This entry point initializes all variables in the program. Respond as illustrated here:

Computer Shows

What You Do

RETURN TO INITIALIZE PGM	Press ENTER.
PICA OR ELITE (P/E)?	Enter E and press ENTER.
TABULATION DESIGNER	Program is now in the regular mode. It is only necessary to enter the initialize mode when you change type sizes. Press the ENTER key.
USING ELITE TYPE.	Press ENTER.
HORIZONTAL INCHES?	Press ENTER since computer will default to the 8-1/2 inch size paper being used for this example.
USING 8.5 INCHES.	Press ENTER.
PREF. LEFT MARGIN.	How wide (number of characters) do you want the margins to be? The program will warn you if you make the margins too wide. Input 10 and press ENTER here.
HOW MANY COLUMNS?	This example has six. Input 6 and press the ENTER key.
ALL THE SAME SIZE?	Press Y and the ENTER button for this case.
COLUMN WIDTH (SPACES)?	Enter the number 6 and press the ENTER key.
MAX. SPCS./COL.=9.2	The program indicates that there will be about 9 spaces between each of the six columns.

If the Spacing Is Not an Integer Value

Few typewriters can perform fractional spacing. If spacing between

Program Automatic Tab Locator

```

1  "C" PRINT "RETURN TO INITIALIZE PGM":CLEAR
2  Q$="X":INPUT "PICA OR ELITE (P/E)? ":Q$
3  IF Q$="E" LET S=12:Q$="ELITE"
4  IF Q$="P" LET S=10:Q$="PICA"
5  IF Q$="X" PRINT "ERROR!":GOTO 2
6  " " PRINT "TABULATION COMPUTER"
7  PRINT "USING ";Q$;" TYPE."
8  H=8.5:INPUT "HORIZONTAL INCHES? ":H
9  PRINT "USING ";H;" INCHES."
10 H=S*H
11 INPUT "PREF. LEFT MARGIN? ":M
12 INPUT "HOW MANY COLUMNS? ":T
13 INPUT "ALL SAME SIZE (Y/N)? ":R$
14 IF R$="N" GOTO 18
15 INPUT "COLUMN WIDTH (SPACES)? ":W
16 FOR I=1 TO T:A(I+26)=W:NEXT I
17 GOTO 23
18 FOR I=1 TO T:A(I)=I+26
19 PAUSE "COLUMN #":I
20 INPUT A(A):IF A(A)>=1 GOTO 22
21 GOTO 19
22 NEXT I
23 X=0
24 FOR I=27 TO (T+26)
25 X=X+A(I)
26 NEXT I
27 W=X+(T-1)
28 IF W<H GOTO 30
29 PRINT "MIN. SPACES REQD. ":W:GOTO 6
30 D=H-(2*M)
31 IF D>X+(T-1) GOTO 38
32 PRINT "MARGIN TOO LARGE..."
33 V=(D-(X+(T-1)))/2:PRINT V;"=MAX. AVAIL. MARGIN"
34 N=0:INPUT "PLEASE RE-ENTER MARGIN: ":N
35 IF N=0 GOTO 34
36 M=N
37 GOTO 30
38 U=(D-X)/(T-1)
39 PRINT "MAX SPCS/COL=":U
40 R$="X":INPUT "RE-DEFINE MARGIN (Y/N)? ":R$
41 IF R$="X" THEN GOTO 40
42 IF R$="N" THEN GOTO 46
43 PRINT "OLD MARGIN=":M
44 N=0:INPUT "NEW MARGIN? ":N
45 M=N:GOTO 27
46 PRINT "LEFT MARGIN=":M
47 L=M
48 FOR I=2 TO T
49 L=L+U+A(25+I)
50 USING "###":PRINT "COLUMN #":I;"=":L
51 NEXT I
52 PRINT "---END RUN":GOTO 6

```


columns is not an integer value, as in this example, use the rounded value as the best choice. Or, you can adjust the between column spacing by re-defining the margins. For instance, the example problem could be continued by pressing the ENTER key again:

Computer Shows	What You Do
RE-DEFINE MARGIN (Y/N)?	Input Y and press ENTER.
OLD MARGIN=10.	Press ENTER.
NEW MARGIN?	Input 8 and press ENTER.
MAX. SPCS./COL.=10	Good! Have the even spacing that was desired. Press ENTER again.
RE-DEFINE MARGIN (Y/N)?	Input N so can exit with solutions. Press ENTER.
LEFT MARGIN=8	Program reminds you where to start. Press ENTER.
COLUMN # 2 = 24	Press ENTER.
COLUMN # 3 = 40	Press ENTER.
COLUMN # 4 = 56	Press ENTER.
COLUMN # 5 = 72	Press ENTER.
COLUMN # 6 = 88	Press ENTER.
---END RUN	Press ENTER to return to the start of the program, if desired.

It Can Be Handy

If you are involved with a lot of statistical typing, accounting work or if you design report formats for computerized jobs, this utility program can save you a lot of time and trouble.

Submitted by: Ken Slaughter, 2916 Bangor Avenue, Highland, CA 92456.

AMORTIZATION PROGRAM

This program, submitted by *Emerich Auersbacher*, produces a complete loan amortization schedule. All you do is input the amount of the loan, the number of monthly payments and the annual interest rate. The pocket computer will then give you the monthly payment needed to repay the loan. It then gives the amount of interest due each month, the amount by which the principal is reduced each month, and the principal balance at the end of each period.

Planning on buying an appliance, car or house in the near future? Take your PC with this program in it along with you!

Program Amortization

```

5  BEEP 1:PAUSE " AMORTIZATION SCHEDULE"
10 CLEAR:INPUT "LOAN AMOUNT $:";V,
    "PERIODS:";N,"INTEREST RATE (%):";Y
15 Y=Y/100,Y=Y/12
20 P=V*(Y/(1-(1+Y)^(-N)))
25 BEEP 1
30 PRINT "PERIODIC PMT= $";USING "#####.##";P
35 L=V
40 FOR Z=1 TO N STEP 1:USING
45 K=(1+Y)^(-Z),B=1/K*(P*(K-1)/Y+V):IF B < E-3 LET B=0
48 BEEP 1:PAUSE "PERIOD ";Z
50 J=B-L+P,L=B,R=P-I
60 USING "#####.##"
65 PRINT "I=";I;" P=";R:USING "#####.##"
68 PRINT "BAL=$";B
70 NEXT Z
80 END

```

FAST-FOURIER-TRANSFORM

The Fourier transform is becoming increasingly popular as a mathematical tool as programmable calculators, and now pocket computers, have gained in capability. The Fourier transform becomes more useful as the number of data points increases. However, the amount of computation increases as the square of the data. Using the "fast" implementation reduces the number of computations required and greatly speeds the problem-solving process.

The compact fast-Fourier-transform program provided here leaves 20 steps of memory free even when 64 complex points are specified as data. (Thus, there is room to modify the program slightly if you desire a different input or output format, etc.)

Only 11 registers are used to store intermediate values, serve as loop counters and other program parameters. Execution speed is quite fast: 4.5 minutes for a 32 complex-point transform, 10.5 minutes for a full 64 complex-point transform.

Operation

Enter the RUN command. Respond to the prompt for "N" with the number of data points to be entered (maximum of 64). Respond to the prompt for "D" by inputting 1 for the forward transform (time to frequency) or -1 for the inverse (frequency to time).

The program is designed to use the printer unit. The data classification (FREQ or TIME) is printed as appropriate and the input of data requested. All points, including zero, must be entered. Data is printed for verification.

When all data points have been entered a bit-reversal routine is executed. Then the transform is performed. The RADIAN statement in line 4 of the program separates the bit reversal routine from the transform instructions.

Finally, the results are outputted to the printer.

Remember that the number of data points specified for a transform should always be a power of two, thus you should specify and enter either 2, 4, 6, 8, 16, 32 or 64 points. The more points used, the more accurate and meaningful the results.

For comparison purposes, you might want to review the FFT program designed for a TI-59 (Texas Instrument's) calculator that appeared on page 233 of the May 10, 1980 issue of *Electronic Design*. The TI-59 version required 13 minutes to solve a 32-point transform. The program presented here performs the same transform in one-third the time.

This program submitted by: A. D. Chamier, 6 Cissbury Hill, Crawley, Sussex, England RH11 8TJ.

Program Fast-Fourier-Transform

```

1  INPUT "N=";A,"D=";D:B=1:C=1:GOSUB 7:I=1:FOR K=1
    TO A-2:J=A
2  IF I > K LET E=11+K:F=11+I:G=A(E):A(E)=A(F):A(F)=G:
    H=A(E+A):A(E+A)=A(F+A):A(F+A)=H
3  J=J/2:IF J < I LET I=J:GOTO 3
4  I=I+J:NEXT K:RADIAN:FOR K=1 TO A/2:E=A-2K:FOR
    J=12 TO 11+K:B=(J-12)D*PI/K:C=COS B:B=SIN B
5  FOR I=J TO J+E STEP 2K:F=I+K:G=CA(F)-BA(F+A):H=
    CA(F+A)+BA(F):A(F)=A(I)-G
6  A(F+A)=A(I+A)-H:A(I)=A(I)+G:A(I+A)=A(I+A)+H:NEXT I
    :NEXT J:K=2K-1:NEXT K:D=-D:B=1:IF -D LET B=A
7  IF D PRINT "TIME
8  IF -D PRINT "FREQ
9  PRINT "N    REAL    IMAG":FOR K=12 TO 11+A:J=
    K+A:IF C INPUT "REAL=";A(K),"IMAG=";A(J)
10 G=A(K)/B:H=A(J)/B:PRINT USING "###.###";K-11:USING
    "###.###";G:H:NEXT K:IF C RETURN

```

USE OF MEMORY IN THE RADIO SHACK TRS-80 PC

The major portion of the random access memory in the pocket computer resides on three chips, each of which contains 512 bytes (capable of storing 1024 hexadecimal digits). The first 48 bytes are used for the RESERVE mode memory. The next 1424 are the regular memory used for storage of a program or data. The next 32 bytes store the variables W, X, Y and Z. The last 32 bytes are used to store information and addresses for the four levels of subroutines and nested FOR loops that the PC is capable of handling. (PCN Issue 06, page 4, contains the hexadecimal addresses of these memory locations.)

The variables A through V, the input buffer, and display buffer are stored on other smaller chips.

Data is stored in binary-coded-decimal (BCD) format. Sixteen hexadecimal digits are used to represent variable values. Two of these digits are used to indicate the signs of the exponent and mantissa. Two are used to store the value of the exponent. Ten are used to store the mantissa value. Two are left over.

String variables are stored in the same locations as numerical variables. One pair of digits identifies the variable as a string. The next seven pairs represent the seven characters making up the string. (PCN Issue 07, page 4, provides a list of the character codes used in the PC.)

Information in the RESERVE memory is stored in the following format: One byte marks the beginning of a segment. The rest of the segment is stored as character codes.

There are 256 different binary patterns that can be represented in an 8-bit byte. The PC uses about 110 of the possible combinations as tokens for the various program words and characters. Several other codes are used to mark the beginning and end of program lines.

The remaining possible codes, over 100, do not seem to be of much practical use. Many are displayed as duplicates of other characters. Some are undisplayable and appear to cause "crashes." An interesting exception is made by the codes AE and AF. These are displayed as STEPS and MEMORIES respectively. The unused codes print as question marks when fed to the printer, except for the codes AE and AF. These are spelled out as PI and SQR.

When the unused codes are placed in program memory and the program is executed, many of them simply produce an error. A few appear to act identically to some of the functions. For instance, the code 87 calculates arcsine. Others perform only part of a function routine. The code 80 seems to act like COS but does not always provide the correct sign. The codes 6B and 6F are equivalent to specifying A(27) and A(31). They might be of some conceivable use. However, the task of placing these codes into a program line is so complex and time-consuming, that it simply isn't worth the effort as a practical matter.

This information provided by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

EXPLORING THE RADIO SHACK AND SHARP PCs

An easy way to begin exploring what takes place inside these PCs is to use the fact that when program lines are deleted, the boundary between program storage and data is relocated, but the previously stored information is not erased! Thus, program lines may be deleted and the previously stored hexadecimal digits recalled as data or strings.

For instance, after clearing out memory with the directive NEW, try entering the program line:

25 N=LMQ

Then, delete line 25. (Just key in 25 and ENTER with the computer in the PRO mode.)

The program line just deleted will still be stored as the hexadecimal pairs:

E0 25 5E 34 5C 5D 61 00

(PCN Issue 05, page 3, describes how pairs of hexadecimal digits are stored as bytes.)

Next, with the PC in the RUN mode, type A(204) and press the ENTER key. This results in the unit displaying:

¥5.¥543¥552

This display may be explained as follows:

A variable storage location holds 16 hexadecimal digits. These are read from memory in the direction *opposite* to that in which program bytes are read. The first three digits specify the sign and magnitude of the exponent. The next eleven represent the sign and digits in the mantissa. The final two digits, serving as the 11th and 12th digits of the mantissa, are not displayed. Note that when displayed in this manner, the hexadecimal digits 0 through 9 are represented directly, however the hexadecimal symbols A through F are displayed as the characters . (period), E (exponent indicator), %, ¥, \$, and π .

When the hexadecimal digits are transcribed to appear in reverse order, the result is:

00 16 D5 C5 43 E5 52 0E

This can be interpreted in the following manner: The first zero indicates a positive exponent value. The next two digits, 0 and 1, provide an exponent value of 1. This explains the decimal point location in the display of A(204) above. The fourth digit, 6, indicates the mantissa is positive in value. The next ten digits represent the mantissa value.

Those who try some experimenting will find that if the first digit is non-zero, the exponent is negative in sign. Negative values are represented in complementary form. If the fourth digit is 8 or greater, then the mantissa is negative.

As an example of storage of a string, enter the program line:

581 ADF+QO

Note that the last character is the letter of the alphabet, not the digit zero. Now delete this line. Displaying A\$(204) will show: U%CS#N. When decoded to hexadecimal pairs (see PCN Issue 07, page 4 for a conversion table) this yields:

E5 81 51 54 35 61 56 5F

Reversing this sequence provides:

F5 65 16 53 45 15 18 5E

The first byte here, starting with the hexadecimal symbol F, indicates that the variable is storing a string. The remaining seven bytes represent the characters in the string.

If you want to explore even further, try using the computer in a mode in which program memory overlaps data memory and the loop/subroutine registers. (Additional information regarding this technique is in PCN Issue 04, page 2, with a correction to the method discussed in Issue 07, page 6.)

[*Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158, developed this method of exploring the PC. His pioneering work in this area is certainly appreciated by PC enthusiasts. — N.W.*]

A MESSAGE FOR AUTHORS

PCN invites the submission of quality programs for pocket computers as well as practical application and tutorial articles of specific interest to users of pocket computers.

PCN purchases all rights to accepted materials at rates ranging from \$50 to \$150 per finished page as it appears in PCN. The payment rate is highly dependent upon the editorial and technical quality, timeliness and general suitability of the material for our readers.

Please double-space all manuscript material. Include a clean printer listing and a tape cassette copy of substantial listings. Be sure and include a self-addressed, stamped envelope if you desire the return of material that has been submitted for our consideration.

We keep all materials, including cassette tapes, and provide an Author's Agreement when a submission meets our requirements.

Please direct all submissions for editorial review to:

The Editor
POCKET COMPUTER NEWSLETTER
35 Old State Road
Oxford, CT 06483

POCKET COMPUTER INTERFACE SIGNALS

Ron Miller, 11918 S.E. Halgate Boulevard, Portland, OR 97266, has been doing some detective work on the signals used to interface with the Radio Shack TRS-80 PC. Here is what he has discovered to date. (The pin numbering convention is the same as that used previously in PCN. It is reviewed in an accompanying diagram.)

Pin 1: Serial data output to the printer. Data is transmitted at a rate of 500 baud. Each character is presented as two 4-bit "nibbles." A nibble is preceded by a start bit. There are 4 stop bits between nibbles. Characters are separated by 5 stop bits. The first nibble in a character contains the four most significant bits. The second nibble contains the four least significant bits. Within a nibble, the least significant bit is sent first, the most significant is last. The character codes are the same as those previously listed by Norlin Rober. (See PCN issue 5, page 3.) However, line numbers are not preceded by the codes E0 - E9. Instead, the code 8E heads a string when a PRINT command is executed or 8D when a LIST command is in progress. Data strings are terminated by the code 00.

Pin 2: Indicates when the printer data buffer is full. It is sent from the printer and tells the PC to stop transmitting. It is normally low. It is brought high when the PC is to inhibit transmission. It appears as though the printer has an input buffer of about 80 characters.

Pin 3: When high, print statements are routed to the printer. This signal is apparently only sampled by the PC when the POWER ON key is pressed twice.

Pin 4: This signal notifies the printer that the PC has data to send. It is normally high. It goes low about 3.35 milliseconds before the first data appears on pin 1. It is kept low until about 8 milliseconds after the last character in a line has been transmitted.

Pin 5: This is the Vcc positive supply pin.

Pin 6: Relay control signal for controlling a tape recorder. Normally in a high state, the signal goes low whenever a tape read or write operation is executed. (See PCN issue 4, page 1.)

Pin 7: Audio input from a tape cassette to the PC. Format is the same as described for pin 8.

Pin 8: Output from the PC to a tape cassette interface. The signal is a square wave with a frequency of approximately 4000 Hertz representing a logic high and a frequency of 2000 Hertz for a logic low. The baud rate is 500.

Pin 9: This is the common (0 volt) reference.

Note: Apparently the printer recognizes statement tokens in the same manner as the PC display. Only the token code is transmitted and tokenized words are never broken up when printed.

For a practical application of the information described here, turn to the next page!

Figure Pin Numbering Convention

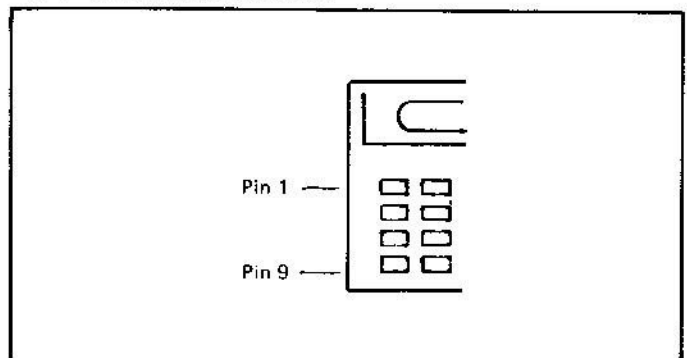


Figure Printer "Handshaking" Logic Timing Diagram

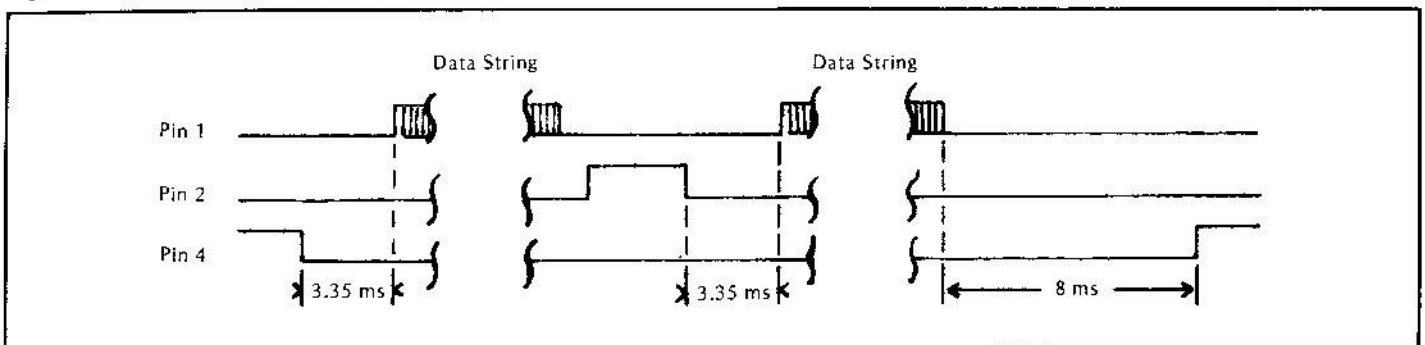


Figure Format Used to Transmit Data to the Printer

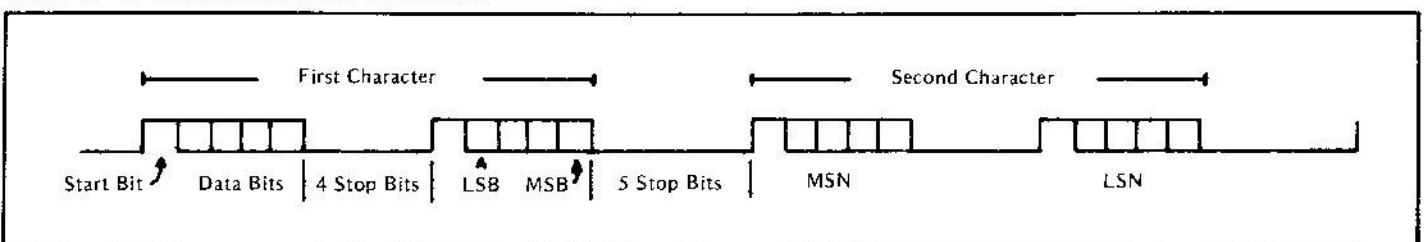
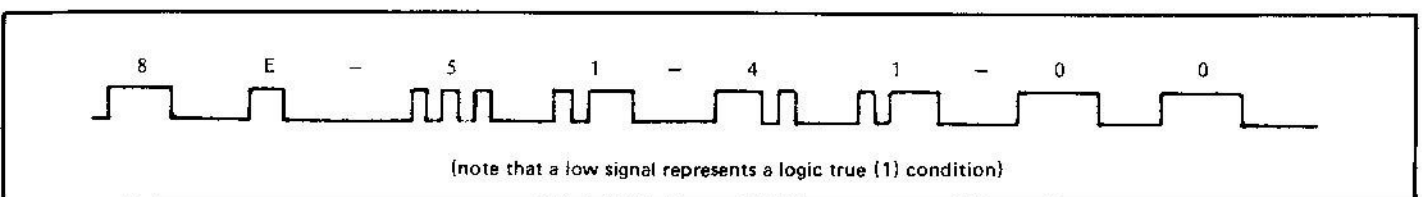


Figure Waveform Transmitted to the Printer by Execution of the Statement: PRINT "A1"



X-Y PLOTTER INTERFACE FOR THE SHARP PC-1211

If you have access to an analog X-Y plotter, you may be interested in building this interface. It allows you to output digital data from a Sharp PC-1211 or Radio Shack TRS-80 PC to the plotter.

The PC-1211 outputs data to the CE-122 printer in BCD (binary coded decimal) format using its own hexadecimal character set. The circuit shown in the accompanying schematic converts this digital data to an analog signal.

The circuit illustrated uses series 7400 low power TTL (transistor-transistor-logic) and two digital-to-analog converters. A separate low frequency transistor is used to isolate the pocket computer and thus avoid any possible damage to the unit.

The circuitry consumes about 330 milliamperes from its own 5 volt power supply. Much of the power is consumed by the 74148 integrated circuits. These devices could be eliminated if the data is converted to octal notation using software before it is transmitted from the PC.

To minimize the parts count, the X and Y data values are limited to the integer range 0 through 99. Because of the manner in which data is outputted, the data must be outputted in the format: 1XXYY. This is accomplished by multiplying the X value by 100, adding this to the Y value, and then adding 10000. The latter step prevents a leading zero in the X value from being blanked. These operations can be combined into a single statement:

```
PRINT IE4+IE2X+Y
```

This effectively formats the value to be outputted in the right hand side of the display complete with a trailing decimal point. This value is then outputted on the printer data pin when the PC is fooled into thinking it is connected to a printer.

When the PC is set to output data to a printer, it proceeds as follows: Each byte of data is transmitted as two 4-bit "nibbles." The transmission is preceded by the value "8E" that signifies that the communication is for the printer. Then the 24 positions of the LCD are sent (even though the display appears blank) proceeding from left to right. The transmission is terminated with the hexadecimal byte 00.

Each 4-bit nibble is preceded by a "start" bit. Thus, the method is that of a simple asynchronous transmitter. The accompanying logic timing diagram clarifies the method.

The X-Y plotter interface uses a simply asynchronous receiver to clock data into shift registers. Since the first nibble of a decimal digit is always 4, these nibbles are blanked out to save hardware. At the conclusion of each data stream (which takes 0.85 seconds) the relevant nibbles representing the X and Y data are latched into the 74LS-273 ICs. This data, now in parallel format, is then decoded to binary format and fed to the DACs (digital-to-analog converters) ready for outputting to the plotter.

While the data receiver may appear complex because it uses three

dual monostables, this configuration actually provided the lowest IC count. It may be necessary to adjust the timing resistor values slightly to match the tolerances of the capacitors used as well as to compensate for variations between PC units, since the PC-1211 does not utilize a crystal clock. Only the monostable oscillators marked D and E in the schematic are really critical. The functions of the monostables are outlined here:

The "word" monostable (W) remains triggered throughout the data stream. When the stream ends, the new data is latched.

The "initial" monostable (I) blanks the initial nibble of each byte so that only the relevant digit-indicating nibbles are captured.

Monostable (A) blanks the first nibble of each byte.

Monostable (B) resets the bit counter and times out the "start" bit that precedes each nibble.

Monostables (D) and (E) serve as an oscillator to clock in the four bits of each nibble to the shift registers.

Counter (C) stops nibble data entry after 4 bits are received.

In practice, small slew movements will be drawn as straight lines between the two data points. Large slews may result in curved lines. The capacitor filter provided on the output of the DAC delays the onset of this effect. The capacitor value shown provides an RC time constant of 220 milliseconds for the DAC used and gave good results. (Remember, it takes at least 0.85 seconds between new data values due to the manner in which the PC serially transmits data.)

The BASIC program that provides the data to be plotted must scale the data points to be integer values in the range 0 to 99. The print statement presented previously can then be used to output the scaled data. The use of exponent notation and implied multiplication in the PRINT statement saves four bytes of program space.

Direct PRINT statements can also be used to draw figures or characters. For instance, graph axes might be displayed using the series:

```
PRINT IE4:PRINT 19900:PRINT 19999:PRINT 10099:PRINT IE4
```

Pen lifting and lowering must be performed manually. However, this can be prompted in the PC's display using a dummy input statement such as:

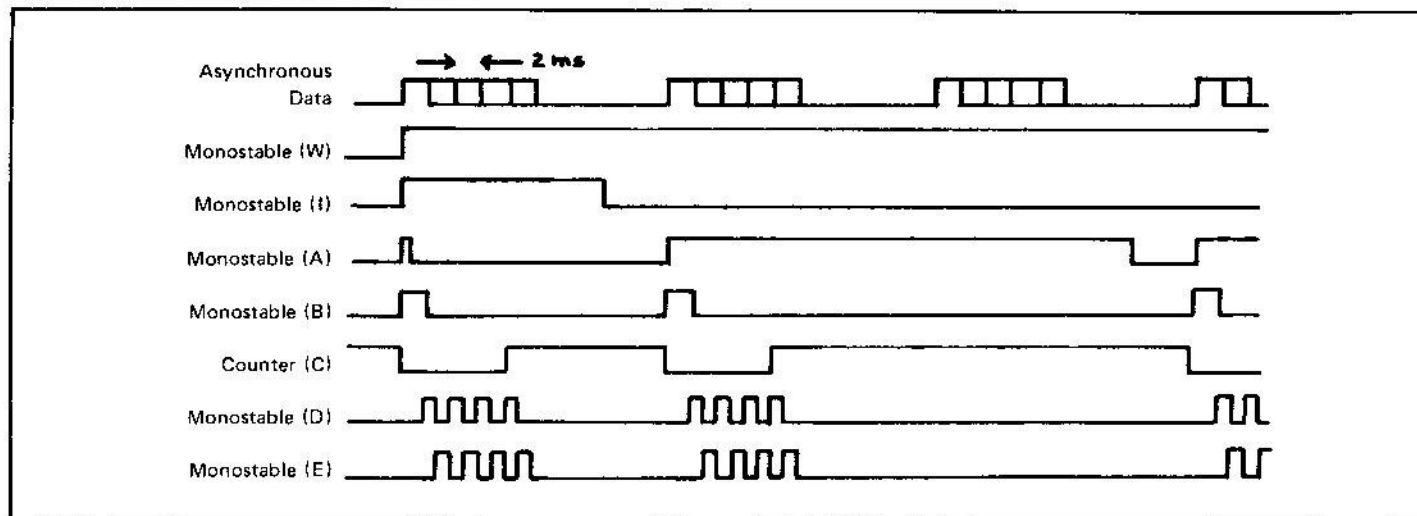
```
INPUT "LOWER PEN";K
```

A possible option for those that want to dabble some more with the hardware, would be to use the decimal point position as a pen status flag. The USING format could be used to allow any digit to be placed as the last character (in place of the decimal point). The value of this digit could be used to raise or lower the pen.

Other options include simply eliminating the binary converters and DACs and using the remaining circuitry as a simple parallel output device to control "whatever."

Thanks for this design go to: A. D. Chamier, 6 Cissbury Hill, Crawley, Sussex, England RH11 8TJ.

Figure X-Y Plotter Interface Logic Timing Diagram



PROGRAMMING TIPS AND TRICKS

Have you ever had a problem with overflow on your pocket computer? If you deal extensively in the fields of science, engineering or mathematics, where large numbers are frequently used, chances are good that you have left problems unsolved because your computer overflowed. Although final answers are generally likely to be within the range of a PC (less than $1E100$), intermediate results in many cases can cause the computer to overflow.

Want to eliminate those overflow aborts? Sure you do. Here is a way to get rid of most such situations: The trick is to use logarithms in all of your calculations where overflow is likely to occur. Then use the output routine presented here to display the results.

Suppose you were dealing with factorials. A conventional routine to solve for factorial values might appear as follows:

```
10 F=1:INPUT "N?";N
20 FOR Z=1 TO N
30 F=F*Z
40 NEXT Z
50 PRINT "N!=";F
60 END
```

Unfortunately, this standard technique overflows when N exceeds 69. If you want to compute 350!, you are out of luck.

However, if the problem is set up using logarithms, a routine for processing factorials could appear as shown here:

```
10 F=LOG 1:INPUT "N?";N
20 FOR Z=1 TO N
30 F=F+LOG Z
40 NEXT Z
50 E=INT F:M=10^(F-INT F)
60 USING:PRINT "N!=";M;"E";E
70 END
```

Now the factorial products become sums. Lines 50 and 60 convert and format the results for output. (The USING statement is recommended, when lines 50 and 60 are used as a subroutine, to cancel any previous format.) Check it out for yourself. Inputting 350! to the above routine yields 1.235873158 $1E740$.

One small note of caution: The technique may result in the loss of one or more significant digits in the mantissa, depending on the size of the exponent. Generally, of course, this result is still much better than not being able to get any answer at all!

Speaking of Factorials

The following asymptotic series approximation can be used for the Gamma function:

$$\Gamma(N) = e^{-N} N^N \sqrt{2\pi/N} [1 + 1/12N + 1/288N^2 - 139/51840N^3 - \dots + \dots]$$

For positive integers:

$$N! = \Gamma(N+1)$$

Satisfactory results for all N greater than or equal to 1 can be obtained using the first 3 terms of the above series. Now a factorial program can be written in the form:

```
10 INPUT "N?";N
20 N=N+1,F=INT(EXP-N*N^N*sqrt((2*pi/N)*(1+1/12N+1/288NN)))
30 PRINT "N!=";F
40 END
```

What this method does is eliminate the loop used in the previous routines. Now all values of N take approximately the same amount of time to calculate. By the way, statement 20 can be used as a general factorial subroutine. Or, you can put it into REStore memory, assign it to the N key and have your own N! button.

Of course, if you want to determine factorial values for N greater than 69, the routine can be re-written to use the log method described earlier:

```
10 INPUT "N?";N
20 N=N+1:F=-N*LOG EXP 1+N*LOG N+LOG(2*pi/N)/2+
LOG(1+1/12N+1/288NN)
```



35 Old State Rd, Oxford, CT 06483

```
30 E=INT F:M=10^(F-INT F)
40 USING:PRINT "N!=";M;"E";E
50 END
```

This routine can save quite a bit of time, yielding answers in seconds instead of minutes. Furthermore, the limit on FOR-NEXT loops is 999, while this routine has no such limit.

Engineering Notation

Many people find it convenient to be able to have answers expressed in engineering (not scientific) notation. In this type of notation, the mantissa is normalized so that it is in the range of 1 to 1000 and the exponent is always displayed as multiples of 3. This way, the units of measure are more readily apparent. Thus $E-6$ can be related directly to micro (as in microfarads) or $E-9$ to nano(seconds), $E03$ to kilo, $1E06$ to mega(cycles) and so forth.

A routine to format a PC display to engineering notation is shown here:

```
900 Z=LOG ABS X:USING
910 E=INT Z:M=10^(Z-INT Z)
920 IF E/3 < > INT(E/3) LET E=E-1:M=10*M:GOTO 920
930 M=M*SGN X:PRINT M;"E";E
```

To check this out to your own satisfaction, add a statement to input values (X) to the start of the above routine, add an END statement to conclude the routine and try a few values. You should see that:

3.25 $1E7$ becomes 32.5 $1E6$

1.25 $1E-7$ becomes 125 $1E-9$

Thanks for these choice tidbits go to: *Emerich Auersbacher.*

FROM THE WATCH POCKET

As you can see from the front page of this issue, the Sharp PC-1500 is now actually available in the United States. I have also ascertained that it is now also available in Canada. The word is that the CE-150 combination Printer/Cassette Interface will be available here, as will the 4K memory module, within just a few days.

With just a few hours to look the unit over, I am initially impressed. It looks as though Sharp has done their homework on this model and put in a lot of the features that PC-1211 users have indicated they wanted in the next generation of pocket computers.

One little word of caution to beginners who get this unit. It looks as though the manuals that come with the PC-1500 may have been a little rushed. Some of the example routines/programs contain the types of errors that indicate they were not thoroughly tested. So, don't be surprised if a few examples don't work quite as stated. There is a lot of material in these manuals, however, and most of it is undoubtedly correct.

On To Other Subjects

H. Walker wants PC users to know that Radio Shack has advised him not to try using the CTR 80 with the Radio Shack TR5-80 Pocket Computer when the recorder is powered by an alternating current source. Could be that hum or line noise interferes with the proper loading of programs. You might want to keep this advice in mind if you are having program recording or loading problems with any tape recorder that is operated by house current. Try using a battery operated tape recorder. If this gives you improved performance, stick with it!

After praising 80 *Microcomputing* for being one of the few magazines to carry articles and programs on PCs, I was unable to find a single such article in the February issue!

But, *Bub McElwain* is still plugging away in the February issue of *Interface Age*. The program presented there can, among other things, calculate how much interest you paid on a mortgage over a specified time span.

When is Radio Shack going to have their PC-2 (equivalent of the Sharp PC-1500) available? Scuttlebutt I have heard puts it at sometime around May. If that turns out to be the case, Sharp could do a nice volume before then.

— *Nat Wadsworth, Editor*

POCKET COMPUTER NEWSLETTER



© Copyright 1982 — Issue 14

April

PROGRAM CODES AND DECODES MESSAGES FOR PRIVACY

This program converts the Radio Shack TRS-80 or Sharp PC-1211 PC into a sophisticated pocket coder! Messages can be encoded and decoded using a personalized keyword (secret code).

Only letters of the alphabet may be encoded. Characters in a message are input one character at a time. A period (decimal point) is used to signify the end of a message. Do not code spaces. Leave them out of messages. They are generally quite easy to place as a message is decoded by the receiver.

The ability to "lock" and "unlock" a message is controlled by a keyword. The key used by this program can be up to 30 characters in length. This allows more than 2.5 to the thirty-eighth power possible key variations. With such a large number of keys, the message becomes quite difficult to break.

The key is specified at the start of operations to either encode or decode a message. To prevent unauthorized use, the key is erased from memory at the conclusion of either type of operation.

A message to be coded or decoded can be of any length. The coder/decoder is quite rapid in operation, taking only a second or so to cipher or decipher each character.

Some people may want to save keystrokes by replacing the PRINT statements in lines 85 and 120 with PAUSE statements. If this is done, you have to pay close attention to the display and write down the output as soon as it appears. However, you won't have to hit the ENTER key after each display. Instead, the program will be ready for the inputting of the next character.

A Few Tests

Here are a few examples of the program's operation that you can use to make sure that the program is loaded properly.

```
KEYWORD: SECRET.
MESSAGE: POCKETCOMPUTERNEWSLETTER.
CIPHER: RCOLSWECYQIWGFZFKVNSFUSU
```

Now, reverse the process to see that you get back the right message:

```
KEYWORD: SECRET.
CIPHER: RCOLSWECYQIWGFZFKVNSFUSU.
MESSAGE: POCKETCOMPUTERNEWSLETTER
```

Here is the same message encoded using a different, longer keyword:

```
KEYWORD ABNFLWNBDVWLYCLTXELWWLY.
MESSAGE POCKETCOMPUTERNEWSLETTER.
CIPHER ZZZZZZZZZZZZZZZZZZZZZZZZZ
```

Can you imagine someone trying to decode a string of Zs without having the key? To check your unit, put the keyword back in, key in the string of Zs, and see that you get the proper message:

```
KEYWORD ABNFLWNBDVWLYCLTXELWWLY.
CIPHER ZZZZZZZZZZZZZZZZZZZZZZZZZ.
MESSAGE POCKETCOMPUTERNEWSLETTER
```

When you are sure that you have keyed in the program properly by performing the above tests, you are ready to put it to use. Just make sure the recipient of your messages has a PC, a copy of the program and the correct keyword. Otherwise, they are going to have a difficult time figuring out what on earth you are talking about!

Note that the essence of this method is in the *keyword*. A person can

have a PC and the program, but still be essentially unable to decode a message unless the keyword is known. So, be sure you appropriately safeguard the keyword at all times!

Thanks for this rather novel application of a PC go to: *Emerich Auersbacher, 41 King Street — Apt. 2, Belleville, NJ 07109.*

Program Message Coder/Decoder

```
10 PAUSE "INITIALIZING":FOR Z=1 TO 26:A(Z)=
  Z:NEXT Z
15 INPUT "CODE/DECODE? (C/D):";A$(53)
20 A$(27)="A":A$(28)="B":A$(29)="C":A$(30)=
  "D":A$(31)="E":A$(32)="F":A$(33)="G"
25 A$(34)="H":A$(35)="I":A$(36)="J":A$(37)="K"
  :A$(38)="L":A$(39)="M":A$(40)="N"
30 A$(41)="O":A$(42)="P":A$(43)="Q":A$(44)=
  "R":A$(45)="S":A$(46)="T":A$(47)="U"
35 A$(48)="V":A$(49)="W":A$(50)="X":A$(51)=
  "Y":A$(52)="Z"
40 BEEP 1:PAUSE "KEYWORD -->":A(55)=58
45 IF A(55) > 87 LET A(57)=A(55):GOTO 60
50 INPUT "LETTER:":A(54):IF A(54)=0 LET A(57)=
  A(55):GOTO 60
55 A(A(55))=A(54)-1:A(55)=A(55)+1:GOTO 45
60 IF A$(53)="D" THEN 90
65 BEEP 1:PAUSE "CODE -->":A(55)=57
70 INPUT "LETTER:":A(54):IF A(54)=0 PAUSE
  "END":CLEAR:END
75 A(55)=A(55)+1:IF A(55) >= A(57) LET A(55)=58
80 A(56)=A(A(55))+A(54)+9:A$(56)=A$(A(56)-26*
  INT(A(56)/26)+27)
85 PRINT "CODED LETTER: ";A$(56):GOTO 70
90 BEEP 1:PAUSE "DECODE -->":A(55)=57
95 INPUT "LETTER:":A(54):IF A(54)=0 PAUSE
  "END":CLEAR:END
100 A(55)=A(55)+1:IF A(55) >= A(57) LET A(55)=58
105 A(56)=A(54)-A(A(55))-11
110 IF A(56) < 0 LET A(56)=A(56)+26:GOTO 110
115 A$(56)=A$(A(56)+27)
120 PRINT "DECODED LETTER: ";A$(56):GOTO 95
```

PRODUCT REVIEW THE PANASONIC HHC

Produced by: *Panasonic Company.*

List Price: \$600.00 (4K memory version).

Availability: *Panasonic distributors.*

Reviewer: *David G. Motta, 3639 Roosevelt, Jackson, MI 49203.*

I received my Panasonic HHC in a very large box directly from the Panasonic Company. I was surprised at the weight of the box. I had only ordered the hand-held unit (4K memory version), the battery charger and the Microsoft BASIC cartridge. When I opened up the boxes I determined why they weighed so much. These units are *heavy!*

Maybe that is one reason that I feel that they are well made. This HHC feels like a good piece of machinery in the hand, somewhat like the feel of a good programmable calculator such as the Hewlett-Packard model 67 that I purchased some time ago.

The HHC is bigger than other "pocket computers." By no stretch of the imagination can it be considered pocket-sized. It has a keyboard with 65 keys arranged similar to a typewriter layout with the number keys on the top row. The keys repeat automatically if held down for more than about one-half a second. There are two shift keys. One for upper case and one for other functions.

The display is an 8 by 159 dot liquid-crystal display matrix. There are also seven annunciators that display the status of certain operations, such as if the SHIFT key has been pressed. An eighth annunciator may be activated by applications programs. The display can present up to 26 characters at a time.

The HHC comes equipped with an operating system installed that provides four initial operating modes. Each mode is selected from a main menu. One mode directs the unit to perform as a four-function calculator. Another turns it into a programmable alarm clock. A third provides a file system. The fourth permits the running of applications that reside in plug-in cartridges. Up to three cartridges will fit into the back of the unit at one time. The left side of the unit holds a special port that enables the unit to be connected to accessories.

The Panasonic HHC utilizes a 6502 microprocessor (the same type of CPU as that used in the Apple II, the OSI Challenger, the Commodore PET and other personal desktop computers) running at a clock rate of 1.3 Mhz. It is powered by rechargeable batteries.

Using the HHC

Pressing the digit "1" when the HHC is in the menu selection cycle brings up the calculator mode. In this mode, ten-digit values may be added, subtracted, multiplied and divided. There is a four-key memory and a percent key too.

Menu item number 2 provides a clock/controller package. A sub-menu allows a choice of clock functions. You can (1) set alarms, (2) review alarms, (3) acknowledge alarms, (4) display time or (5) set time. Time is displayed as: day (given as a three-letter abbreviation for the day of the week), hours, minutes, seconds, an AM or PM marker, and then the date. The date is shown as a three-letter mnemonic for the month, two digits for the day of the month and four digits for the year. You can use these capabilities in various ways. Need to find the day of the week for a future appointment? Just key in an alarm for that date.

Item number 3 on the main menu brings up a built-in file system. Many business people will like this system. It allows you to create a file of information, add to it, update it and then search it for a specific item. The sub-menu for this system starts with just two entries: New File and Copy File. The first option allows the creation of a new file name. Once entered, this name appears as the third item in the sub-menu! The Copy File option allows the duplication of an entire file.

Once a file has been named, entries of up to 80 characters per record may be made. A warning sounds when you come close to filling up the available memory space. When records are in memory, the search key may be used to find records containing a group of characters, such as a person's last name. You can, of course, list the records as desired.

The fourth menu option is called "Run SNAP programs." SNAP is an assembler-like language similar in construction to FORTH. This option allows such application programs to be executed. That is about all I can tell you about the option at this time.

The Menu Expands!

When I plugged the Microsoft BASIC cartridge into the back of the unit (which has thoughtfully been designed so that it will not go in the wrong way!), a fifth item suddenly appeared as an option on the main menu! Yes, it was: Microsoft BASIC. Now, when item 5 is selected, a sub-menu offers (1) New File and (2) ... (N) — whatever file names you assign to BASIC programs! It is certainly nice to have the capability of storing several separate files with different programs, each using a favorite group of line numbers and designated by a descriptive name.

The fact that the BASIC provided was written by Microsoft means that it is closer to the BASIC used on many desktop computers than that provided on most other hand-held computers. That means there are more pre-written programs for this HHC available at this time than for other pocket computers.

There are some drawbacks to this language, though. Scientists and those in the technical disciplines may be disappointed to learn that there are no trigonometric functions provided, though the manuals list programs that can be used to calculate sine, cosine, tangent and similar functions.

The other features of this BASIC more than make up for this lack, in my opinion. The string handling capabilities are excellent and the ability to handle multi-dimensional arrays makes matrix handling comparatively easy. For instance, strings can be up to 255 characters in length and the string functions ASC, CHR\$, LEN, LEFT\$, MID\$, RIGHT\$, STR\$ and VAL are supported.

Debugging of programs is facilitated by the TRACE mode and by meaningful two-letter error codes.

When the 4K machine is configured with one file for programs, no alarms set and a one-character program file name, it has 3,076 memory locations available for the storage of programs and data. That is enough room to store some fairly complex programs.

Execution is fast. A simple benchmark program in the form:

```
1 FOR I=1 TO 1000:NEXT I
```

appears to run more than 10 times as fast as the new Sharp PC-1500 pocket computer. *[Really? — N.W.]*

The Microsoft BASIC also includes the PEEK, POKE and CALL functions. Though explained in the manual, few concrete examples of their usage is provided. I suspect these functions may be the key to accessing each dot on the display or making music on the beeper. However, neither of these capabilities is expressly provided as part of the BASIC language. Other display possibilities, such as flashing or reversing displayed characters, are explained.

Accessories

A very necessary accessory, that must be purchased separately, is an adaptor/charger. It weighs two pounds complete with long cords that allow use of the HHC far from an electrical outlet.

The standard HHC comes with a brown padded case, a carrying strap and a plastic "whatzis." The latter item slips over the front or back of the unit, but its use is not explained. I gather it guards against coffee spills or unauthorized use.

The manual that accompanies the unit is friendly and easy to read. It is spiral-bound, has 60 pages and seems to also serve as a sneaky sales pitch by providing a set of instructions for the various accessories. Perhaps they considered it convenient to bind them all together, but I consider it devilishly clever.

The accessories for the unit include a video adaptor allowing the HHC to connect to a TV or video screen, an RS-232C adaptor for communication purposes, a 15-column thermal printer, an I/O expansion unit, an acoustic modem and memory modules. An 8K module is currently available and a number of these modules may be combined in a system.

The Bottom Line

This machine *feels good!* It can be equipped with a well-written BASIC language that executes quickly and has good string functions. Other languages will be available soon. The built-in features such as the file system are easy to use. The unit is expandable and many of the accessories are available *now!*

PRODUCT REVIEW THE SHARP PC-1500

Produced by: Sharp Electronics Corporation.

List Price: < \$300.00

Availability: Sharp distributors.

Reviewer: David G. Motto, 3639 Roosevelt, Jackson, MI 49203.

The new Sharp PC-1500 is not as big as the Panasonic HHC, but it is too large to fit into a shirt pocket.

The keyboard is typewriter-style with staggered keys. A numeric keypad is at the right of the alphabetical keys. The only keys that automatically repeat when held down (in the executive mode) are the cursor controls. These serve the same functions as those on the earlier PC-1211 unit.

The display consists of a 7 by 156 dot liquid-crystal matrix. In the executive or edit mode the display scrolls up to 80 characters across its 26-character width. In the "graphics" mode each dot on the display may be individually activated through a column-addressing technique. The display also has a number of annunciators that indicate the status of the machine.

The PC-1500 has several operating modes. The RUN mode is used to execute programs or perform calculations. The PRO (program) mode is used to create and edit programs. A RESERVE mode may be used to assign operations and functions to a set of six user-definable keys. Each such key may be assigned three different operations. Labels that appear on the display may be created to identify the function of each user-definable key. A key labeled RCL (recall) is used to bring up the key notations when desired. Another key is used to cycle between user-definable-key modes referred to as I, II and III that give those keys their versatility.

There is a 60-pin connector on the left side of the computer. A compartment into which additional memory may be plugged is provided in the back of the unit. Another compartment accessed from the back houses four size AA batteries that power the unit in portable operation. An a.c. adaptor (optional, not provided with the unit) may be plugged into the right side of the unit.

A special TIME function is provided by the computer. It holds the date and time in the numeric format: MMDDHH.mmss where MM represents the month, DD the day of the month, HH the hour, mm the minute and ss the second. The internal clock can be set by the user and then automatically maintains the correct time.

A programmable audio generator permits the playing of notes that may be varied in pitch and duration.

The BASIC language that is provided on this unit is a considerable upgrade from their earlier unit yet it is also upwards compatible. Programs written for the PC-1211 will run on the new PC-1500 as long as no tricks (such as implied multiplication) are attempted. However, there are some differences in the manner in which variables are handled. The PC-1500 has a separate memory area reserved for the single-character variables A through Z and the string variables A\$ through Z\$. You can now use both the variable A and A\$ at the same time. The single character string variables A\$ through Z\$ are limited to sixteen characters maximum. (Other two-character string variables, such as AA\$, can be dimensioned to hold up to 80 characters.) If you want to use the locations set aside for variables A — Z and A\$ — Z\$ as part of an array using subscripting, then the special symbol "@" is used along with the subscript. Thus, @(1) would be stored in the location assigned to variable A and @\$(26) would correspond to Z\$.

Two-character variables are stored in the user's memory area. One- or two-dimensional arrays (numeric or string) may be utilized.

This version of BASIC supports the string functions ASC, CHR\$, LEFT\$, MID\$, RIGHT\$, STR\$, VAL and LEN. Strings can be compared on a character-by-character basis for equality or the greater than or less than (using the ASCII character code) condition.

New BASIC functions include AND, OR and NOT.

Program errors may be located by pressing the line scroll up key immediately after receipt of an error message. There is a TRACE mode that allows the operation of a program to be closely examined. For instance, the values of variables may be examined after the execution of each statement if desired. Error messages are coded as numbers. The

nice feature of being able to go backwards in a program listing has been retained.

An INKEY\$ statement allows keyboard entries (under program control) to be made without having to use the ENTER key.

Programmers will also like the built-in hexadecimal (up to 4 digits) conversion routine that translates to decimal values.

The PC-1500 is supplied with a black vinyl, padded carrying case and two manuals. The Instruction Manual is better written than previous ones, but it sometimes tends to be over-cute. The programs supplied in the Applications Manual are sometimes re-hashes of earlier ones. Many of the programs also require the use of the accessory printer/plotter and cassette interface and/or the 4K optional memory module.

An accompanying program presents the time of day each time the PC-1500 is turned on. It demonstrates some of the new string handling statements available on this PC.

A second short program, for those that may already have a PC-1500, is based on graphics produced by Michael Lamping. It demonstrates simple animation and sound effects.

```
10:ARUN :WAIT 50:
  A=TIME :A=A-10
  0*INT (A/100)
20:D$="AM":B$=
  STR$ (INT A-12
  *(INT A>12)+12
  *(INT A=0)):IF
  A>=12LET D$="P
  M"
30:C$=LEFT$ (MID$
  (STR$ (A-INT A
  ),3,2)+"00",2)
40:CURSOR 9:PRINT
  B$+" ":C$+" ":+
  D$:CLEAR
```

Time Program

This routine, when placed as the first program in memory, will automatically display the current time for a brief period whenever the PC-1500 is turned on. Notice the ARUN statement in line 10 which is the key to this automatic power-on operation.

Animated Runner Program

As an example of the type of animated graphics that can be executed on the PC-1500, this routine presents a surprisingly smooth animated display of a man running across the screen, complete with sound effects. If you want to study how the artist achieved the fluidity, slow the routine down by changing the WAIT statement in line 10 to read WAIT 100.

```
10:WAIT 03:CLS
20:FOR A=0TO 140
  STEP 6
40:GOCURSOR A:
  GPRINT "000048
  241E0F2E1C02"
60:GOCURSOR A+1:
  GPRINT "002020
  2C1E0F0E0C32"
80:GOCURSOR A+2:
  GPRINT "000010
  241E0F0E2404"
100:GOCURSOR A+3:
  GPRINT "000020
  203E0F4E3400":
  BEEP 1,5,5
120:GOCURSOR A+4:
  GPRINT "000000
  002E7F3E0000"
140:GOCURSOR A+5:
  GPRINT "000000
  4C3E0F2E3400"
160:NEXT A:GPRINT
  "00000000000000
  00000"
180:GOTO 20
```

PRODUCT REVIEW

SHARP CE-150 PRINTER/PLOTTER & CASSETTE INTERFACE

Produced by: *Sharp Electronics Corporation.*

List Price: < \$250.00

Availability: *Sharp distributors.*

Reviewer: *Nat Wadsworth, Editor.*

Clever! That is the first impression I had when starting to examine and exercise the Sharp CE-150.

The printing mechanism on this machine is unique to say the least. A plastic cylinder is notched along its axis to hold four miniature ball-point pens at intervals of 90 degrees around its circumference. These pens are about one inch in length, slightly less than 1/4 inch thick in the barrel portion, and are supplied in four colors. When installed on the plastic cylinder the pens look similar to bullets mounted in a revolver.

The tip of each pen fits through a delicate metal membrane that serves to hold the point firmly while also acting as a return spring. When resting, the pens are positioned a fraction of an inch away from the writing surface of the paper. However, when a pen is positioned at the top of the plastic cylinder (or drum, if you prefer), the back of the pen is contacted by a small lever or hammer. This lever presses the pen forward whenever printing or drawing takes place. When the lever is not activated, the thin metal membrane at the tip of the pen pushes the pen back to its resting (non-writing) position.

The paper-feeding portion of this unit is also somewhat unique. As the paper passes around the main printing platten it is gripped tightly on each side by sprockets. The points of these sprockets actually leave tiny pin-prick marks along the sides of the paper about 1/16 of an inch from the edges. This gripping method enables the paper to be advanced or *backed up* a considerable distance (to a maximum of 10 centimeters). Thus, the paper is manipulated as it would be in a drum plotter.

The pen(s) move horizontally across the width of paper and the paper can be moved forwards or backwards. When you combine the two directions of motion you have the operation of the new CE-150 printer/plotter. It is quite a sight to see in operation, especially when printing text. This is because every character is actually drawn as a series of line segments. Drawing a character such as the letter "X" requires simultaneous movement of the paper and pen. The movement of both items is quite rapid and it is further synchronized with appropriate "pen down" and "pen up" commands. The net effect is a jittering motion of both the paper and pen and a rapid (but soft) pitter-patter sound as the pen strikes the paper. But, behold, the message does appear!

I was, frankly, surprised at the precision with which characters are drawn. An accompanying example shows the print sizes that can be produced by the unit when in the text mode. Character sizes are simply chosen using the CSIZE command. The directive CSIZE 1 yields tiny writing that I find useful for archive listings. CSIZE 2 is the size (second line in the example) normally used for producing program listings. CSIZE 9 produces characters that are close to 1/2 inch high.

Basic Capabilities

The unit can be checked out at any time by issuing the TEST directive. This causes a self-test procedure to be executed. As this is done the unit draws a series of four boxes, one using each pen color.

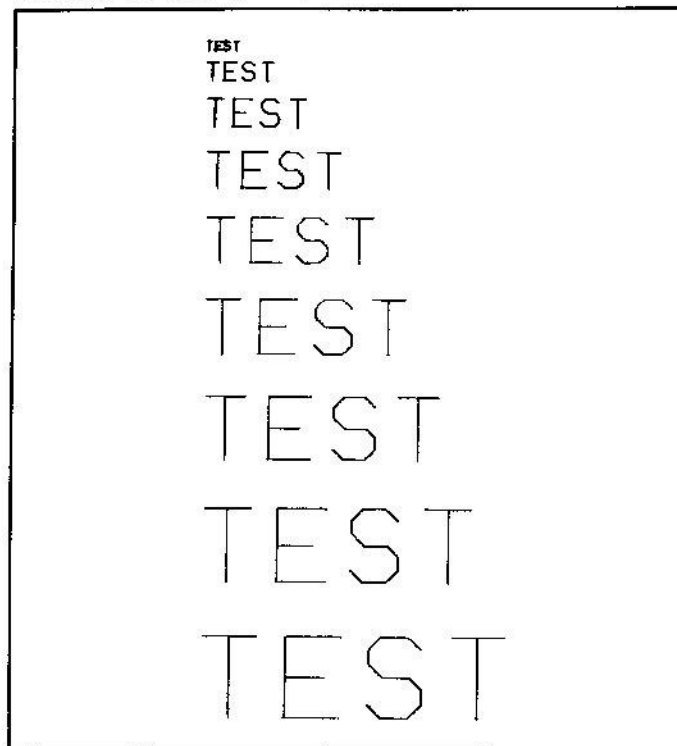
A switch on the front of the unit may be set by the operator to cause the printing of manual calculations or to only activate the unit under software (program) control.

The PC-1500 must be mounted on and hence plugged into the CE-150 when in operation. Rechargeable batteries power both the printer and the computer when so combined.

The CE-150 has two fundamental modes of operation. The TEXT mode is normally used for printing characters. Text can be printed in four directions: normal (left-to-right), inverted (right-to-left), and vertically (either from bottom-to-top or vice-versa). The ROTATE directive is used to select the direction in which text printing is to proceed.

The LPRINT command along with the USING, TAB, LCURSOR,

Sample CE-150 in TEXT Mode.



CSIZE, COLOR and LF directives is used to output text characters under program control. The LPRINT directive is similar in capability to the PRINT command that controls the PC's liquid-crystal display (LCD), but with extended capabilities through the use of the various supplemental commands.

To obtain program listings the LLIST command is utilized. This directive allows single lines, groups of lines or the entire contents of memory to be printed. Lines to be printed can be specified by line number or through the use of program labels.

You can obtain listings in the color of your choice by selecting a particular pen color prior to using the LLIST directive. This can be particularly useful in keeping listings in chronological order as a program is developed and upgraded.

An interesting capability of this unit is that a line feed (LF) command permits paper to be moved forwards or *backwards*! (There is, however, a limitation on how far you can back the paper. The maximum is about 10 centimeters which corresponds to approximately 24 lines when printing characters of size 18 to a line.)

Graphics Features

A group of commands when the unit is in the GRAPH mode enables the device to operate as a plotter.

An "origin" for graphic operations may be set using the "set origin" (SORGN) command. Once established, future pen moves in this mode are referenced to this position.

The GLCURSOR directive moves the pen to an X-Y coordinate without drawing a line (pen up).

The LINE statement enables the plotter to draw a line between two (or more) X-Y coordinates, using a specified type of line and pen color. The type of line possible ranges from a solid line through a series of dotted and dashed lines. A special variation of this instruction permits a box, square or rectangular, to automatically be drawn. All that is required are the coordinates of the diagonal across the box. Multi-segment lines are drawn by giving a series of X-Y coordinates in a single statement.

The RLINE directive is identical to the LINE statement except that X-Y coordinates are relative the current pen position rather than a previously set origin.

Pens Require Care

The 20 page instruction pamphlet is careful to warn the user to remove the pens from the unit when the system is not in use. The ink in the pens seems to be thinner than that of ordinary ballpoint writers and can apparently dry out at the tip fairly rapidly if the pens are not capped. After several weeks of operation I have not experienced any difficulty if each pen receives use on a daily basis. However, upon returning from a five day trip with the pens left uncapped in the unit, I found one of the pens would no longer write. Indeed, the tip had dried out just as the manual had warned. Fortunately, I found that the pen was rejuvenated by placing a drop of water in its storage cap and allowing the piece to rest for a day.

Removing/remounting the set of pens takes a minute or two. This can be a little annoying if you are used to "instant-on" appliances. But, my experience indicates you should heed the manual's advice and place the pens back in their storage capsules if you will not be using them for a period of several days.

Paper

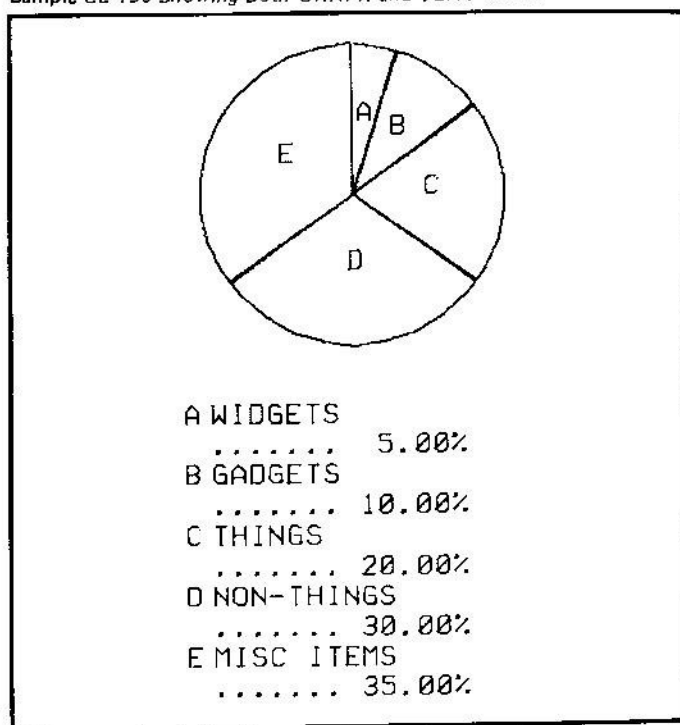
The paper used in the machine appears to be ordinary 2-1/4 inch wide adding- or calculator-machine tape. I have been replenishing the small inch-and-a-quarter diameter rolls supplied with the unit by winding such tape on the exhausted spools. Be sure to save the spools from the initial paper supply that comes with the unit if you plan to economize likewise. The spools are specially sized to accept a shaft that holds a spool in position and enables it to feed paper smoothly to the printer/plotter.

Summary

This appears to be a precision unit with remarkable printing capability considering its moderate price. I am initially intrigued and favorably impressed. Time will now tell if it maintains its precision under daily use.

Oh yes, the unit includes a tape cassette interface that can control two separate tape units. Its operation is straightforward and has a few features that make it a little snazzier than its predecessor, the CE-121/CE-122. For instance, it tells you the name(s) of the file(s) as it finds and loads them from tape!

Sample CE-150 Showing Both GRAPH and TEXT Modes.



PRODUCT REVIEW ELECTRICAL ENGINEERING I

Produced by: Radio Shack for the TRS-80 Pocket Computer.

List Price: \$24.95

Availability: Radio Shack Computer Centers and selected Radio Shack stores.

Reviewer: Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.

COMPLEX, the first of the six programs in the package, converts the PC to a complex-number calculator that uses reverse-polish notation.

I have some good news and some bad news.

The good news first: Radio shack has crammed a large number of operations and functions into this program. A nicely labeled keyboard template is provided to facilitate easy location of the appropriate user-defined keys. In addition to the four arithmetic operations, the operations X^Y and $\log X$ to base Y are provided. The programmed functions of complex numbers include $1/X$, \ln , e^X , \sin , \cos , \tan , ASN , ACS , and ATN as well as conversions between rectangular and polar forms. Following any calculation, the real and imaginary parts of the resulting complex number appear in the display.

The reverse-polish notation, familiar to users of Hewlett-Packard calculators, operates with a stack of twelve complex-number registers. (In the reverse-polish system, an operation on a pair of numbers requires first entering the pair, then specifying the operation to be performed.) The variable I is used as a pointer to indicate the location of the next-available stack position. There are "bells and whistles" to warn the user when the stack is depleted or overfilled.

The effect of the sequence SHIFT/SPC is determined by what has been entered into the display — a rather ingenious way of (in effect) extending the number of user-defined keys. If I is first put into the display, SHIFT/SPC will initialize the calculator by clearing the stack. With C displayed, the top of the stack is copied into the next stack level, making it convenient to multiply a complex number by itself without having to re-enter it. If P is displayed, the top of the stack is "pulled," that is, it is removed from the stack and displayed. This is useful in making corrections. Finally, if the display is blank, prompts for entering inputs into the stack will appear.

Fitting all of this into the available memory of the PC required heavy reliance on the use of subroutines. As a consequence, execution is slowed to some extent because of the search time needed for subroutine calls.

Now for the bad news. The subroutine used for rectangular-to-polar conversion produces incorrect results whenever the complex number used as input contains a negative real part. To make matters worse, this subroutine is also used by \ln , X^Y and $\log X$ to base Y . There is still more bad news. The routines for ASN , ACS and ATN (in some cases) give wrong answers!

Here are some examples: $-1 + 0j$ is converted to the polar form $r = 1$, $\theta = 0$. $\ln(-1 + 0j)$ is given as zero and $(-1)^{1/3}$ is given as 1. Although $\text{ATN}(\sqrt{3})$ equals $\pi/3$, it is given as $2\pi/3$. $\text{ASN}(0 - 1j)$ should be $0 - .881373587j$, but the program calculates the value as $0 + .881373587j$. These are not isolated examples of special cases giving incorrect results.

It is really a shame that such an otherwise well-thought-out program should be ruined because of incorrect algorithms. Anyone buying this package should plan on correcting the mistakes in it before putting it to productive use.

[Norlin has provided the necessary "fixes" — see the recommended corrections at the conclusion of this review. — N.W.]

The second program in the package, IMPCALC , is designed to perform impedance calculations. It determines the effective complex impedance of a network containing inductances, capacitances and resistances combined in series and parallel circuits. This program, which is only 606 bytes long, appears to be free of errors.

Program AMPDES determines the bias resistances and bias voltage for a transistor amplifier circuit. The instruction manual includes a diagram of the circuit modeled. A nice feature of this program is that the calculated resistor values are rounded to the nearest standard values. Thus, a resistance calculated as 35,800 ohms will be rounded to the

standard 33,000 ohm value (when 10% tolerance is specified). Although the instruction manual does not mention it, SHIFT/B can be used to re-display the calculated results. Another routine in the program provides the capability to determine the required heat sink transfer characteristic. The SHIFT/SPC key sequence provides a program menu.

SIMEQ is a program designed to solve a system of up to seven simultaneous equations. It is possible to display the entire matrix, one element at a time, after it has been entered. If used with a printer, a permanent record may be obtained. When the matrix is reviewed, the opportunity to correct the displayed entry is provided. This is a nice feature. It is also possible to change a specific entry without going through the entire matrix. This is accomplished by keying SHIFT/F and indicating the desired row and column when prompted.

However, a price must be paid for all this convenience. The program is so long that a system of seven equations is the largest system solvable within remaining memory. Execution is slow. A system of six equations takes about 3-1/2 minutes to solve. The speed could have been essentially doubled if less subroutine calls had been used. There also appears to be a minor error in this program. When the prompt asks for the order of the system, it will not accept an input of 2 even though the program is capable of solving such a system. When the order 8 is specified, it is accepted and the program proceeds to give an incorrect solution, instead of warning the user that the program's capabilities have been exceeded.

[Norlin has also provided a cure for this problem! — N.W.]

The next program, FILTERS, calculates the values for inductances and capacitances in derived highpass and lowpass filters. Input parameters are the cutoff frequency, maximum attenuation frequency and termination resistance. Active lowpass, highpass and bandpass filter circuits may also be designed. The computer determines the appropriate resistor and capacitor values from relevant inputs. The manual includes circuit diagrams for each of the filters covered, identifying the components specified as parameters.

The last program in the package is named TABLES. It may be used to determine (1) the resistance, per foot of copper wire at a given wire gauge; (2) the wire size needed to carry a specified current; (3) the value and tolerance of a color-coded resistor; (4) the value and tolerance of a color-coded capacitor; and (5) the number of turns of wire needed to make a RF coil having a specific inductance (in microhenries) at a given coil diameter and gauge of wire.

The routines that translate color codes allow either "grey" or "gray" as well as "purple" or "violet." However, the practicality of these programs is questionable. It seems that one could get results a lot faster consulting a chart rather than spending the time required to spell out the individual colors (to say nothing about the time needed to load the program from the cassette!).

There does appear to be a good deal of useful material in the package. It is unfortunate that the complex number calculator contains a number of bugs.

Bug Fixes

[PCN asked Radio Shack to comment on Norlin's findings. While expressing immediate concern, a Radio Shack representative was not able to provide a definitive corrective response as of press time. However, the following modifications to the Electrical Engineering I package are said to eliminate the bugs discovered in the review. These are the alterations suggested by the program's reviewer, Norlin Rober.]

Listing Correction to SIMEQ Program.

The program SIMEQ can be corrected to accept a 2nd order system and reject 8th order by replacing line 22 with the following:

```
22 "A" INPUT "ORDER?";A:B=A:IF (A < 2) + (A > 7)
    GOTO 22
```

Listing Corrections to COMPLEX Program.

The program COMPLEX requires considerable alteration to fix. If the original line numbers 25, 26, 28 and 39 are simply modified to eliminate wrong answers, then the program becomes so long that there is not enough memory left to hold a 12-register stack. However, it is possible to economize in other sections of the program in order to pick up some room in memory. The alterations settled upon include modifications to lines 25 — 28 and 39 — 44 as shown in the following listing:

```
25 "X" GOSUB 44:A=ASN E:I=I+2:GOTO 19
26 "C" GOSUB 44:A=ACS E:B=-B:I=I+2:GOTO 19
27 "V" C=AA+BB+2B+1:C=.25*LN(C/(C-4B)
28 G=A:B=1-AA-BB+E-90:A=ATN(2A/B)/2+(B<0)*pi/2*
    (SGN G)+(G=0)

39 G=√(AA+BB):B=SGN A*ASN(B/G)+(A<0)*pi*(SGN B+
    (B=0:A=G:RETURN
40 A=EXP A
41 B=E*SIN B:A=A*COS B:RETURN
42 F=EXP B:G=(F+1/F)/2:H=G-1/F:B=H*COS A:A=G*
    SIN A:RETURN
43 F=EXP B:G=(F+1/F)/2:H=G-1/F:B=-H*SIN A:A=G*
    COS A:RETURN
44 D=AA+2A+1+BB:G=√(D-4A):D=(√D+G)/2:E=D-G:
    B=LN(D+√(DD-1))*SGN B+(B=0:RETURN
```

PRODUCT REVIEW

PROBLEM-SOLVING ON THE TRS-80 POCKET COMPUTER

Authors: Don Inman and Jim Conlan

Publisher: John Wiley & Sons, Inc.

List Price: \$8.95

Reviewer: Nat Wadsworth, Editor

The authors of this book are associated with Dymax Corporation, a firm that has long been active in the personal computing educational field. This book is intended to educate first-time users in the operation and application of the Radio Shack TRS-80 Pocket Computer. This it will certainly do — and in a friendly and uplifting manner.

The 240+ page book is divided into 12 chapters plus 5 appendices. The chapters provide an overview of the computer itself, and then get right into teaching different capabilities of the little machine through a variety of practical application examples. There are sections dealing with data files, trigonometric functions, logic functions, random numbers, interest calculations, storing, sorting and searching, chaining programs together and using the optional printer unit.

The style is tutorial, friendly yet not condescending. There are lots of tables, charts and walk-through sequences — where each input and the expected display results are illustrated. A considerable number of practice examples are included whereby you are asked to fill in your own version of a program. The book then illustrates a suggested method of approaching the problem. There are additional self-test problems at the end of chapters, along with the answers or results one should obtain when a solution has been properly programmed.

The publishers of the book have declared it to be part of their STG ("Self-Teaching Guide") series. This it certainly appears to be. If you are a beginner to the use of PCs, I think you would enjoy this volume. At 240+ pages it is still slim enough (just 1/2-inch thick) to fit in your briefcase for ready reference or study in spare moments.

ELECTRONIC FILTER PROGRAM FOR THE CASIO FX-702P

This program determines the required filter order and component values to use in Butterworth or Chebyshev lowpass and highpass filters. An accompanying diagram indicates the circuit forms and positions of calculated components. The program only calculates odd order filters in order to maintain equal input/output impedances.

Mnemonic prompts used in the program are defined as follows:

R-IN = Filter input/output impedance in ohms.
 END-F = Filter passband end frequency (ripple bandwidth).
 REJ-F = Rejection frequency (determines filter slope/order).
 REJ-LOSS = Attenuation in DB at rejection frequency.
 RPL = Ripple in DB of Chebyshev filter.
 N = Filter order.
 3DB-F = Butterworth passband end frequency.

Sample Run

Check for proper loading of the program using the following example:

BUTWTH OR CHEB? C

HI OR LOWPASS? L

R-IN? 75

END-F? 5IE6

REJ-F? 7.5IE6

REJ-LOSS(DB)? 20

RPL? .5

N=5 REJ=26

C1=7.239730803IE-10

L2=2.935477964IE-06

C3=1.078383348IE-09

L4=2.935477964IE-06

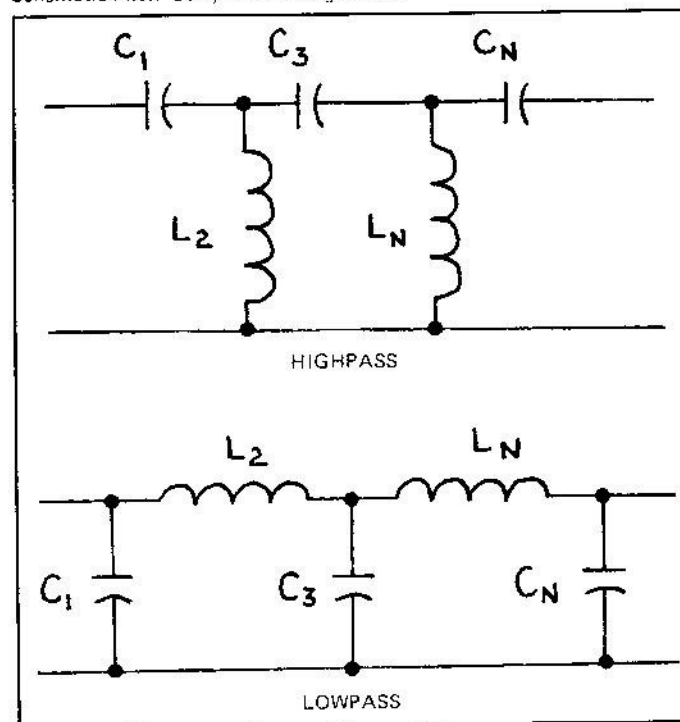
C5=7.239730803IE-10

PROGRAM COMPLETE

Program Filter Component Values

```
P0: 938 STEPS
50 INP "BUTWTH OR CHEB",S
60 IF MID(1,1)="B" THEN 3000
85 INP "HI OR LOWPASS",S
100 INP "R-IN",R,"END-F",U,"REJ-F",V,"REJ-LOSS
(DB)",A,"RPL",T
104 IF MID(1,1)="H" THEN 108
106 V=V/U:GOTO 110
108 V=U/V
110 GSB 1000
130 PRT "N=",N," REJ=",INT C
140 FOR K=1 TO N:GSB 2000
145 IF Q=1;G=(1/G)
150 IF FRAC (K/2)=0:PRT "C";K;"=";((1/R)/(2*PI*U))*G
160 IF FRAC (K/2)=0:PRT "L";K;"=";(R/(2*PI*U))*G
170 NEXT K
180 PRT "PROGRAM COMPLETE":END
1000 MODE 5:N=3
1010 D=(HCS (N*ABC (V)))*(HCS (N*ABC (V)))
1020 C=10*LOG (1+((10^(T/10))-1)*D)
1030 IF C > A THEN 1050
1040 N=N+2:GOTO 1010
1050 RET
2000 MODE 5:J=LN (1/(HTN (T/17.37)))
2010 Y=HSN (J/(2*N))
```

Schematic Filter Component Designations



```
2015 Z=SIN (((2*K)-1)*PI)/(2*N))
2020 B=Y+2+SIN ((K*PI)/N)*SIN ((K*PI)/N)
2030 IF K=1 THEN 2050
2040 G=((4*X)*Z)/(W*S):GOTO 2060
2050 G=(2*Z)/Y
2060 X=Z:W=B:S=G
2070 RET
3000 INP "HI OR LOWPASS",S
3010 INP "R-IN",R,"3DB-F",U,"REJ-F",V,"REJ-LOSS
(DB)",A
3014 IF MID(1,1)="H";V=U/V:GOTO 3020
3016 V=V/U
3020 GSB 4000
3030 PRT "N=",N," LOSS=",INT (C+.5)
3040 FOR K=1 TO N:GSB 5000
3045 IF Q=1;G=(1/G)
3050 IF FRAC (K/2)=0:PRT "C";K;"=";((1/R)/(2*PI*U))*G
3060 IF FRAC (K/2)=0:PRT "L";K;"=";(R/(2*PI*U))*G
3070 NEXT K
3080 GOTO 180
4000 N=3
4010 C=10*LOG (1+V*(2*N))
4020 IF C > A THEN 4040
4030 N=N+2:GOTO 4010
4040 RET
5000 MODE 5
5010 G=2*SIN (((2*K)-1)*PI)/(2*N))
5020 RET
```

PROGRAM YIELDS PHASES OF THE MOON

Input any date (month/day/year) and this program outputs the phase of the moon on that date. If the moon is not at any exact quarter phase (such as full, new, first quarter and so forth) then the program indicates the number of days into the lunar cycle.

The program is accurate for dates between March 1, 1900 and February 28, 2100. Starting with a lunar time of 14.60 days on March 1, 1900, this program adds 29.53059167 days for each lunar period. Need to know if there was a full moon when you were born... married... first became a computer programmer?!

Program submitted by: *Emerich Auersbacher, 41 King Street, Apt. 2, Belleville, NJ 07109.*

Program Moon Phases

```

10 CLEAR:BEEP 1:PAUSE "MOON PHASE CALCULATOR"
20 W=694098,X=29.53059167,Z=365.25:USING
30 BEEP 1:PAUSE "DATE IN QUESTION >> -->"
40 INPUT "MONTH:";M,"DAY:";D,"YEAR:";Y:IF Y < 100
   LET Y=1900+Y
50 IF M <= 2 LET S=((INT(30.6*(M+13))+INT(Z*(Y-1))+
   D-W)+14.6)/X:S=S-INT S:S=INT(SX+1):GOTO 70
60 S=((INT(30.6*(M+1))+INT(Z*Y)+D-W)+14.6)/X:S=S-
   INT S:S=INT(SX+1)
65 BEEP 1
70 IF (S=0)+(S=1)+(S=29)+(S=30) > 0 PRINT "PHASE:
   FULL MOON":GOTO 110
80 IF (S=14)+(S=15)+(S=16) > 0 PRINT "PHASE: NEW
   MOON":GOTO 110
90 IF (S=6)+(S=7)+(S=8) > 0 PRINT "PHASE: LAST
   QUARTER":GOTO 110
100 IF (S=21)+(S=22)+(S=23) > 0 PRINT "PHASE: FIRST
   QUARTER"
110 PRINT "CYCLE IS ";S;"DAYS INTO 30"
120 GOTO 30

```

PUTTING SOFTKEYS UNDER PROGRAM CONTROL

The softkeys on the new Sharp PC-1500 make selecting a particular program or routine the simple matter of pressing a single key. These keys can also be given labels that appear on the display. Using the label recall key a user can be reminded of the function of each key.

In normal use, the softkeys execute a programmed operation only when the PC is in the "executive" mode. That is, when the unit is not in the BUSY condition. Also, in order to see the softkey labels, it is necessary to first press the RCL key. Thus, while it is possible to construct a multi-part program and use the labeled softkeys as a menu selector, it is necessary to end each segment to return to the executive, have the user press the RCL key to bring up the softkeys menu and then punch the appropriate softkey.

However, putting the softkeys under program control makes it possible to display a menu and use the softkeys to direct program operation without ever returning to the executive mode. The accompanying listing illustrates how this can be accomplished.



35 Old State Rd, Oxford, CT 06483

Line 7000 presents a menu on the display.

Line 7010 uses the INKEY\$ function to examine the keyboard.

Line 7020 extracts the ASCII code for the key that has been pressed. It ignores all key codes outside the range (17 - 22) used by the softkeys.

Line 7030 subtracts 16 from the ASCII code of a softkey to leave a number in the range 1 to 6. This number then directs the ON...GOTO statement to the appropriate section of the program.

This routine can be implemented as a subroutine if desired. Just append a RETURN statement and change the directed GOTO statement to an ON...GOSUB directive.

Now a program can be kept entirely under programmed operation, menus will automatically be displayed as required and a single stroke of a softkey will take the user to the appropriate program section.

Program Program Control of Softkeys

```

7000 WAIT 0:PRINT "LB1 LB2 LB3 LB4 LB5 LB6"
7010 X$=INKEY$:IF X$="" GOTO 7010
7020 X=ASC(X$):IF (X < 17)+(X > 22) THEN 7100
7030 X=X-17:ON X GOTO 1000,2000,3000,4000,5000,6000

```

FROM THE WATCH POCKET

I attended the 7th West Coast Computer Faire in San Francisco this past month -- hoping to see the latest in PC wares. The only exhibitor in the PC department was a representative of Quasar showing (and selling) their HHC. I'll have more to say about the Quasar unit in a later issue. It is exactly the same machine as the Panasonic HHC which *David Motto* seems favorably impressed with as noted in this issue.

After over a month of extensive use of the Sharp PC-1500, my judgement is that it is a fine machine for the money. Fact is, I have yet to find a PC-1500 owner who is not basically pleased (or delighted!) with the unit. 4K memory expansion modules -- kicking the PC to 6K -- are already available in the U.S. and the 8K modules are due here by early April. The modules seem a little high priced (\$150.00 for the 8K unit), but I could sure use 10K (total) in my PC!

Radio Shack is reported to be planning to start selling their PC-2 (equivalent to the Sharp PC-1500) in May.

If you don't want to wait until then, the Sharp PC-1500 is available through a number of U.S. outlets. *Bob Hall* reports good service from Tam's, Inc., 14932 Garfield Avenue, Paramount, CA 90723. They take phone orders at 800-421-5188 and have the PC-1500 listed at \$249.95, the CE-150 at \$199.95 and the CE-151 (4k memory plug-in) at \$64.95. On the east coast, Atlantic NorthEast Marketing, PO Box 921, Marblehead, MA 01945, phone 617-639-0285, has a few interesting offers. If you buy a new PC-1500 and the CE-150 printer/plotter/cassette interface, you can trade in a Sharp PC-1211 or Radio Shack PC-1 (in working condition) for \$80.00 credit. On the other hand, if you are looking to pick up some used (but guaranteed to be in good working condition) PC-1211s and PC-1s and associated equipment, *Mort Rosenstein* says they have a good stock available. Call for prices and details. He (Mort) also says the PC-1500 Service Manual with all the technical details will be available shortly for \$10.00.

Send a self-addressed stamped envelope to *Walt Moffett, Box 1108, Sebastopol, CA 95472*, if you are interested in obtaining printed templates for your PC. He has a printing machine with just the right size characters. In addition to having several "standardized" versions available, he says he can make custom templates too.

The April issue of *BYTE Magazine* features a new pocket-sized telecomputing terminal from a new outfit called *IXO, Incorporated*. A lot of human-factors engineering reportedly went into the terminal which contains a built-in modem, system-connect protocol software, a 1K user RAM buffer and color-coded keyboard. Things are really starting to pop in the PC world!

— *Nat Wadsworth, Editor*

POCKET COMPUTER NEWSLETTER



© Copyright 1982 — Issue 15

May

GENERATING THE CHROMATIC SCALE ON THE SHARP PC-1500

As indicated on pages 84 and 85 of the *Sharp PC-1500 Instruction Manual* the BEEP statement allows the programmer to create tones for game playing, produce warning signals and serve other annunciator purposes. Three parameters specified as suffixes to the statement permit one to specify the number, frequency and duration of the tones emitted. Theoretically one should be able to play music. But what are the specifications needed to produce each tone on the chromatic scale?

Brian Peterson, 6807 N. Sheridan Rd. — Apt. 520, Chicago, IL 60626 has figured this information out and provided it for PCN readers.

Brian has also learned that the duration of a BEEP command varies with the frequency. The duration of all the tones can be made approximately equal by using the following relationship:

$$L = 10^{(N/48)}$$

In this formula L represents the length of the tone and N represents the number of the tone as shown in the accompanying table. The last column in the table shows the calculated value for L using this formula.

Table PC-1500 Chromatic Scale Using BEEP Function

#	KY	tone	L	UAL	26	A#	55	3.48
1	A	255	1.04	27	B	52	3.65	
2	A#	244	1.10	28	C	49	3.83	
3	B	229	1.15	29	C#	45	4.01	
4	C	219	1.21	30	D	43	4.21	
5	C#	203	1.27	31	D#	41	4.42	
6	D	195	1.33	32	E	37	4.64	
7	D#	184	1.39	33	F	35	4.86	
8	E	172	1.46	34	F#	32	5.10	
9	F	163	1.53	35	G	30	5.36	
10	F#	151	1.61	36	G#	28	5.62	
11	G	143	1.69	37	A	26	5.89	
12	G#	133	1.77	38	A#	24	6.18	
13	A	125	1.86	39	B	22	6.49	
14	A#	118	1.95	40	C	21	6.81	
15	B	111	2.05	41	C#	19	7.14	
16	C	105	2.15	42	D	18	7.49	
17	C#	98	2.26	43	D#	17	7.86	
18	D	94	2.37	44	E	15	8.25	
19	D#	89	2.48	45	F	14	8.65	
20	E	81	2.61	46	F#	12	9.08	
21	F	77	2.73	47	G	11	9.53	
22	F#	72	2.87	48	G#	10	10.00	
23	G	68	3.01	49	A	9	10.49	
24	G#	64	3.16	50	A#	8	11.00	
25	A	59	3.31	51	B	7	11.54	

Multiply this value by 10 to produce a note of short duration, 20 to get a longer note, and so forth.

By the way, middle C in the accompanying table is note number 28 (having a tone value of 49 and L value of 3.83).

Notes with tone values of less than 14 (notes numbered 46 and above in the table) are slightly out of tune. They may still be satisfactory in certain applications.

If you want to try your hand at playing a little music, load in the program shown in the accompanying listing and have at it on the number keys.

Brian recently sent in a striking rendition of Bach's *Prelude Number II* from *The Well Tempered Clavier*. He is willing to write a tutorial article on how to convert melodies to play on the PC-1500. Let us know if this strikes your fancy.

Program Musical Keyboard for the Sharp PC-1500

```

100: X$=INKEY$ : IF
    X$="" THEN 100
110: IF (X$<"0")+ (X
    $>"9") THEN 100
120: X=(ASC (X$)-47
    ) *10+190: GOTO
    X
200: BEEP 1, 52, 72:
    GOTO 100
210: BEEP 1, 49, 76:
    GOTO 100
220: BEEP 1, 43, 84:
    GOTO 100
230: BEEP 1, 37, 92:
    GOTO 100
240: BEEP 1, 35, 98:
    GOTO 100
250: BEEP 1, 30, 104:
    GOTO 100
260: BEEP 1, 26, 118:
    GOTO 100
270: BEEP 1, 22, 130:
    GOTO 100
280: BEEP 1, 21, 136:
    GOTO 100
290: BEEP 1, 18, 150:
    GOTO 100

```

STATUS 1

274

UNDERSTANDING THE SHARP PC-1500

This is the first of a series of columns being prepared by *Norlin Rober*, 407 North 1st Avenue, Marshalltown, IA 50158, with the object of the series being to help newcomers get acquainted with their PCs.

Hexadecimal Numbers

People ordinarily write numbers in *decimal* form. These numbers have a base of 10. The value 726 is understood to mean a total of 7 hundreds plus 2 tens plus 6 ones. Stated another way, it equals $7 \times 10^2 + 2 \times 10^1 + 6 \times 1$.

Computers use numbers based on the *binary* system which only has two digits, 0 and 1. It turns out that people can deal with groups of binary digits much more efficiently if they are grouped together and considered as numbers written with a base of 16. In this case, each of the digits is understood to be multiplied by a power of 16. Numbers in this format are referred to as *hexadecimal* numbers. To illustrate, the hexadecimal number 643 has a total value of $6 \times 16^2 + 4 \times 16^1 + 3 \times 1$. If you multiply this out, you will find that this is equivalent to the decimal (base 10) value 1603.

Decimal numbers are written using ten different digits (0, 1, 2, 3, . . . , 9). The hexadecimal system requires the use of 16 different digits. The letters A through F are used to represent the "digits" ten through fifteen. Thus, the hexadecimal digits are 0, 1, 2, . . . , 9, A, B, C, D, E and F. The hexadecimal number 2B represents $2 \times 16^1 + 11 \times 1$, which is 43 decimal.

Using the ampersand (&) on the Sharp PC-1500 as a prefix indicates that the number is to be interpreted as hexadecimal. PCN will use the same convention. Thus, &643 is understood to be the hexadecimal notation of 1603 (decimal).

Just to convince yourself, enter &643 on your PC-1500 and press ENTER. It will respond with the decimal equivalent: 1603.

Peeking at Memory

The BASIC interpreter in the PC-1500 contains a number of instructions that are not mentioned in the instruction manual. One of these is the PEEK function. It can be used to see what is stored at any location in memory. Among other uses, PEEK makes it comparatively easy to discover how the unit's memory is utilized.

To learn how PEEK works, try this example. First, execute the LOCK statement using the keyboard. This directive effectively locks the computer in its present mode of operation. Next, type in PEEK 31231 or PEEK &79FF and then press the ENTER key. The display will show the value 0. Now, execute the UNLOCK statement. Once again type in PEEK 31231 and press ENTER. Note that now the value 96 is displayed. What is going on?

The memory address 31231 (&79FF) is being used by the computer to store the information that the current mode is either locked or unlocked!

Although PEEK returns values in decimal notation, the computer can be considered to be storing numbers in hexadecimal format. The computer also refers to memory addresses as hexadecimal values. The decimal address 31231 is &79FF in hexadecimal and the decimal value 96 is &60. Speaking in hexadecimal terms, location &79FF holds &60 when the PC-1500 is locked and &0 when it is unlocked.

Each memory location is capable of storing a two-digit hexadecimal number ranging from &00 through &FF. A hexadecimal digit can represent the value of a group of 4 binary digits. Two hexadecimal digits can thus represent 8 binary digits. A group of eight binary digits in a computer is often referred to as a *byte*. So, two hexadecimal digits are needed to represent each byte of information in the PC-1500.

Taking a Look at Variable Storage

By doing a lot of PEEKing operations I have been able to ascertain where the variable values are stored in the PC-1500. The variable A is stored in memory locations &7900 through &7907. Notice that this amounts to eight bytes of storage. To experiment yourself, set the variable A to the value 74.685. Then PEEK at the eight locations &7900 through &7907. You should obtain the following results:

Location	PEEK Result (Decimal)	Actual Storage (Hexadecimal)
&7900	1	01
&7901	0	00
&7902	116	74
&7903	104	68
&7904	80	50
&7905	0	00
&7906	0	00
&7907	0	00

The value 01 stored in location &7900 is the exponent of ten for the stored number. (74.685 is stored as 7.4685E1.) Address &7901 is used to store the sign of the number. (Try repeating the above with a negative number and you will see how a negative sign is stored.) Note that (exclusive of the exponent) only the digits zero to nine are used. That is, the digits of the number are stored in decimal form. (Computers often store numbers in strictly binary or hexadecimal form.) However, if you do some experimenting, you will find that the exponent of a stored number uses hexadecimal notation. Furthermore, if the exponent is negative, it is stored as the sum of the negative exponent and the number 256. For instance, an exponent of -18, added to 256 yields a value of 238. When this is converted to hexadecimal the result is &EE. Of course, since PEEK converts to decimal automatically, you will simply see the value 238 when you PEEK at the address holding the exponent!

The string variable A\$ is stored in locations &78C0 through &78CF, allowing room for storage of 16 bytes. Each character of the stored string is stored as a single byte. The stored number for each character is the ASCII code for that character. The ASCII codes are tabulated in Appendix C of the PC-1500's Instruction Manual. Try PEEKing at a stored string variable.

Program Sharp PC-1500 Memory Dump to LCD

```
10 INPUT "STARTING LOCATION? ";A
20 A$=""0123456789ABCDEF"
30 WAIT 0:FOR B=0 TO 7:C=PEEK(A+B):D=1+INT(C/16)
40 PRINT MID$(A$,D,1);MID$(A$,C+17-16*D,1);" ";
   :NEXT B
50 WAIT:PRINT:GOTO 10
```

If you would like to do some exploring on your own and you are getting tired of typing in the command PEEK over and over, the accompanying short program may be used to display eight stored numbers in a row, all neatly converted to hexadecimal. [Another "memory dump" program elsewhere in this issue may be used to explore large areas of memory with output to the printer unit. --N.W.]

To use the memory display program, respond to the prompt by giving the memory address where you want to start examining a series of 8 bytes. You can also easily modify the program to automatically advance to the next 8 bytes by changing line 50 of the program to read:

```
50 WAIT:PRINT:A=A+8:GOTO 30
```

Next issue I plan to discuss the POKE directive which allows you to place values directly into memory locations. I'll have other goodies on the subject of understanding and utilizing the PC-1500 too!

PEEKING IN THE PC-1500

It didn't take users long to discover that there are a number of capabilities of the Sharp PC-1500 that are not mentioned in the instruction manual. Undocumented directives include PEEK, POKE, CALL and OPN. *George Fergus* was one of the first users to get a report in to PCN listing some of his discoveries using the PEEK instruction. This command permits one to examine the contents of memory by giving the directive: PEEK X, where X represents an address in decimal or hexadecimal (indicated by preceding the value with the "&" sign) format. Some of the information George has gleaned, such as the locations (addresses) of RAM elements in the basic unit and the tokens used in the BASIC interpreter, is shown in accompanying tables.

If you want to do some exploring on your own, an accompanying routine provides a means of "dumping" portions of memory using the CE-150 printer. In addition to providing a hexadecimal dump, the program prints ASCII-equivalent characters for appropriate values underneath each line of hexadecimal code. Examining memory starting at hexadecimal address C000 (49152 decimal) reveals the coding of the BASIC interpreter. If you use the dump to examine the first few hundred bytes of ROM code you will discover the token conversion table used by the interpreter.

For those of you who may not have had experience with the PEEK directive, *Norlin Rober* provides an introduction to the subject in an article elsewhere in this issue.

Table PC-1500 BASIC Token Codes

F1 50 AND	F1 81 ARUN	F1 B1 TO
F1 51 OR	F1 82 BEEP	F1 B3 WAIT
F1 58 MEM	F1 83 CONT	F1 B4 ERROR
F1 5B TIME	F1 86 GRAD	F1 B5 LOCK
F1 5C INKEYS	F1 87 CLEAR	F1 B6 UNLOCK
F1 5D PI	F1 8A CALL	
F1 60 ASC	F1 8B DIM	F0 84 CURSOR
F1 61 STR\$	F1 8C DEGREE	F0 85 USING
F1 62 VAL	F1 8D DATA	F0 88 CLS
F1 63 CHR\$	F1 8E END	F0 89 CLOAD
F1 64 LEN	F1 92 GOTO	F0 8F MERGE
F1 65 DEG	F1 94 GOSUB	F0 90 LIST
F1 66 DMS	F1 96 IF	F0 91 INPUT
F1 67 STATUS	F1 98 LET	F0 93 GCURSOR
F1 68 POINT	F1 99 RETURN	F0 95 CSAVE
F1 6B SQR	F1 9A NEXT	F0 97 PRINT
F1 6D NOT	F1 9B NEW	F0 9F GPRINT
F1 6E PEEK#	F1 9C ON	F0 B2 CHAIN
F1 6F PEEK	F1 9D OPN	F0 B5 COLOR
F1 70 ABS	F1 9E OFF	F0 B6 LF
F1 71 INT	F1 A0 POKE#	F0 B7 LINE
F1 72 RIGHT\$	F1 A1 POKE	F0 B8 LLIST
F1 73 ASN	F1 A2 PAUSE	F0 B9 LPRINT
F1 74 ACS	F1 A3 P	F0 BA RLIN
F1 75 ATN	F1 A4 RUN	F0 BB TAB
F1 76 LN	F1 A5 FOR	F0 BC TEST
F1 77 LOG	F1 A6 READ	
F1 78 EXP	F1 A7 RESTORE	E7 A9 RMT
F1 79 SGN	F1 A8 RANDOM	
F1 7A LEFT\$	F1 AA RADIAN	E6 80 CSIZE
F1 7B MIDS	F1 AB REM	E6 81 GRAPH
F1 7C RND	F1 AC STOP	E6 82 GLCURSOR
F1 7D SIN	F1 AD STEP	E6 83 LCURSOR
F1 7E COS	F1 AE THEN	E6 84 SORGN
F1 7F TAN	F1 AF TRON	E6 85 ROTATE
F1 80 AREAD	F1 B0 TROFF	E6 86 TEXT

Table PC-1500 Partial Memory Map

&4008 - 40C3	Reserve Program Area (188 bytes)
&40C6 - 47FF	User Main Program Area (1850 bytes)
&7050 - 70FF	Fixed Variables E\$ - O\$ (11 x 16 bytes)
&7150 - 71FF	Fixed Variables P\$ - Z\$ (11 x 16 bytes)
&78C0 - 78FF	Fixed Variables A\$ - D\$ (4 x 16 bytes)
&7900 - 79CF	Fixed Variables A - Z (26 x 8 bytes)
&C000 - FFFF	BASIC Interpreter (16K bytes in ROM)

Program PC-1500 Hexadecimal Memory Dump

```

10:CSIZE 1:INPUT      ,1);" ";
   "STARTING ADDR 30:WAIT 0:FOR B=0
   ESS? ";A          TO 7:C=(PEEK (
20:A$="0123456789    A+B))AND (&F0)
   ABCDEF"           :D=1+INT (C/16
25:WAIT 0:W=INT (    ):E=(PEEK (A+B
   A/4096)            ))AND (&0F)
26:X=INT ((A-(W*4    40:LPRINT MID$ (A
   096))/256)         $,D,1);MID$ (A
27:Y=INT ((A-(W*4    $,E+1,1);" ";:
   096)-(X*256))/    NEXT B
   16)                45:LPRINT :LPRINT
28:Z=INT ((A-(W*4    " ";:FOR
   096)-(X*256)-(    B=0TO 7:LPRINT
   Y*16)))            (CHR$ (PEEK (A
29:LPRINT MID$ (A    +B))); " ";:
   $,W+1,1);MID$     NEXT B:LPRINT
   (A$,X+1,1);       :LF 1
   MID$ (A$,Y+1,1   50:A=A+8:GOTO 25
   );MID$ (A$,Z+1

```

Table PC-1500 Key Memory Locations

&7600 - 764D	Display memory
&7700 - 774D	Display Memory
&764E - 764F	RUN/PRO/RESERVE settings, trig mode, SHIFT status, SML status
&7800 - 78BF	Memory pointers, format settings
&79D0 - 79FE	Settings primarily involving the CE-150
&79FF	LOCK/UNLOCK status
&7A00 - 7AFF	Calculation registers, data stack, FOR/NEXT stack, subroutine stack, etcetera
&7B00 - 7B07	Random number storage
&7B08 - 7B0F	Counter, used for automatic 7-minute shutoff, possibly other purposes
&7B10 - 7B5F	String buffer
&7B60 - 7BAF	Data to be outputted to display
&7BB0 - 7BFF	Input buffer
&A000 - BFFF	ROM in the CE-150

CURVE FITTING

The accompanying program will fit a series of X-Y data points to all of the following curves: linear, exponential, logarithmic and power. It is a good example of how a lot of practical computing power can be packed into a single program on a pocket computer. The program was crafted by: *Thomas S. Cox, 102 Evergreen Street, Easley, SC 29640.*

After loading the program, place the PC in the DEF mode.

Initialization

Use SHIFT/Space to initialize the program and/or to add additional data points. Answer appropriately to the initialization query. Enter X and Y data for each point. Press ENTER after reviewing data for correctness.

Deleting/Correcting Data

Press SHIFT/G to change or delete data points. Respond appropriately to prompts. Note that SHIFT/G must be used each time a data point is to be altered.

Obtaining Best Fit

Press SHIFT/F to determine which of the four curves yields the highest R^2 value. Information concerning the best fit is output in the sequence:

1. Curve type and descriptive equation, 2. Constant term A, 3. Vari-

able term B, 4. Correlation coefficient R, 5. Coefficient of determination R^2 .

Specific Curves

To view the parameters for any of the four curves: Press SHIFT/L for linear, SHIFT/A for exponential, SHIFT/S for logarithmic and SHIFT/D for the power curve. The parameter data is output in the same sequence as shown for the best fit curve.

Estimate of X or Y

To obtain an estimate of an X or Y value for the currently selected curve, press SHIFT/K (for X) or SHIFT/J (for Y). Enter the appropriate coordinate value when prompted. The program then calculates the predicted value.

Restrictions

If X and/or Y are negative the best fit procedure is skipped. The message "X OR Y OR BOTH NEG" is displayed. Use manual curve selection to obtain a fit. If not feasible, the same "X OR Y OR BOTH NEG" message will be shown.

Tight Fit

The version shown for the Sharp PC-1211/Radio Shack PC-1 will only have 2 program steps left when loaded with no extra spaces as shown in the listing. Variables A - Z are utilized by the program.

Program Curve Fitting (Sharp PC-1211/Radio Shack PC-1 Version)

```

1  " "USING:
   PRINT"CURVE FIT-V4":
   INPUT"CLR (Y/N)";Z$:
   IF Z$="Y"CLEAR
2  C=1:
   Z=N+1:
   PAUSE"PT#";Z:
   INPUT"X=";L:
   IF L<=0 LET J=1
3  X=L:
   INPUT"Y=";Y:
   IF Y<=0 LET T=1
4  GOSUB 5:
   BEEP 2:
   PRINT"X";N;"=";X;" Y";N;"=";Y:
   GOTO 2
5  D=D+CX:
   E=E+CX:
   F=F+CY:
   G=G+CY:
   H=H+CX:
   N=N+1:
   IF J*T=1 RETURN
6  IF J<>1 LET M=M+C*LN X:
   O=O+C*LN X*LN X:
   K=K+C*Y*LN X
7  IF T<>1 LET P=P+C*LN Y:
   Q=Q+C*LN Y*LN Y:
   I=I+CX*LN Y
8  IF J+T=0 LET S=S+C*LN X*LN Y
9  RETURN
10 "L"IF W=1 THEN 14
11 U=1:
   PRINT"LIN. Y=B*X+A":
12 B=(H-DF/N)/(E-DD/N):
   A=(F/N-BD/N)
14 R=(NH-DF)/SQR((NE-DD)*(NG-FF)):
   IF W=1 RETURN
16 GOTO 96
20 "A"IF W=1 THEN 24
21 U=2:
   PRINT"EXP. Y=A*EXP(B*X):
   IF T=1 GOTO 61
22 B=(I-PD/N)/(E-DD/N):
   A=EXP (P/N-BD/N)
24 R=(NI-DP)/SQR((NE-DD)*(NQ-PP)):
   IF W=1 RETURN
26 GOTO 96
30 "S"IF W=1 THEN 34
31 U=3:
   PRINT"LOG. Y=B*LN X+A":
   IF J=1 GOTO 61
32 B=(K-MF/N)/(O-MM/N):
   A=(F-BM)/N
34 R=(NK-FM)/SQR((NO-MM)*(NG-FF)):
   IF W=1 RETURN
36 GOTO 96
40 "D"IF W=1 THEN 44
41 U=4:
   PRINT"PWR. Y=A*X^B":
   IF T+J>0 GOTO 61
42 B=(S-MP/N)/(O-MM/N):
   A=EXP (P/N-(BM/N))
44 R=(NS-MP)/SQR((NO-MM)*(NQ-PP)):
   IF W=1 RETURN
46 GOTO 96
48 "F"IF J+T>0 THEN 61
50 W=1:
   V=0:
   U=0:
   FOR Z=1 TO 4:
     GOSUB (Z*10):
     C=RR:
     IF C>V GOSUB 99
55 NEXT Z:
   USING:
   W=0:
   BEEP 3:
   PRINT"BEST FIT WAS #";U
59 GOTO (U*10)
60 "K"PRINT"ESTIMATE OF X":
   INPUT"Y=";V:
   GOTO (U+61)
61 BEEP 1:
   PRINT"X OR Y OR BOTH NEG.":
   END
62 L=(V-A)/B:
   GOTO 97
63 L=(LNV-LNA)/B:
   GOTO 97
64 L=EXP((V-A)/B):
   GOTO 97
65 L=EXP ((LNV-LNA)/B):
   GOTO 97
67 USING"#####.###":
   RETURN
70 "G"BEEP 1:
   PRINT"DELETE":
   INPUT"DELETE LAST PT (Y/N)";Z$:
   IF Z$="Y"THEN 75
71 PRINT"PT# ";N:
   INPUT"X=";L:
   IF L<=0 LET J=1
72 X=L:
   INPUT"Y=";Y:
   GOSUB 98:
   GOTO 77
75 PRINT"X=";X;" Y=";Y:
   INPUT"OK TO DEL(Y/N)";Z$:
76 IF Z$="N"THEN 71
77 C=-1:
   IF Y<=0 LET T=1
78 GOSUB 5:
   END
82 "J"PRINT "ESTIMATE OF Y":
   INPUT"X=";L:
   GOTO (U+85)
86 V=BL+A:
   GOTO 97
87 V=A*EXP (BL):
   GOTO 97
88 V=B*LN L+A:
   GOTO 97
89 V=A*L^B:
   GOTO 97
96 GOSUB 67:
   PRINT"A=";A:
   PRINT"B=";B:
   PRINT"R=";R:
   C=RR:
   PRINT"R^2=";C:
   END
97 PRINT"X=";L;" Y=";V:
   END
98 PRINT"X=";X;" Y=";Y:
   RETURN
99 U=Z:
   V=C:
   RETURN

```

PC-1500 Version

implied multiplication, used extensively to fit the program in a PC-1, cannot be used in a Radio Shack PC-2/Sharp PC-1500. However, the additional memory in these models allows the program to fit with room to spare when implied directives are changed to explicit representations.

An accompanying listing illustrates how the program appears on a PC-1500/PC-1. Ordinarily, PCN readers will be left on their own to adapt PC-1211/PC-1 programs to PC-1500/PC-2 units. This might typically involve watching out for implied multiplication as well as dropped right hand parens, ending quotation marks, etc.

Program Curve Fitting (Sharp PC-1500/Radio Shack PC-2 Version)

```

1: " "USING :      . Y=B*X+A"
PRINT "CURVE F  12:B=(H-D*F/N)/(E
IT-U4"; INPUT "  -D*D/N):A=(F/N
CLR (Y/N)";Z$:   -B*D/N)
IF Z$="Y"CLEAR  14:R=(N*H-D*F)/J<
2:C=1:Z=N+1:      (N*E-D*D)*(N*G
PAUSE "PT#";Z:    -F*F)):IF W=1
INPUT "X=";L:      RETURN
IF L<=0LET J=1  16:GOTO 96
3:X=L:INPUT "Y="  20:"A"IF W=1THEN
;Y:IF Y<=0LET    24
T=1              21:U=2:PRINT "EXP
4:GOSUB 5:BEEP 2  . Y=A*EXP(B*X)
;PRINT "X";N;"   ";IF T=1GOTO 6
;"X; " Y";N;"=  1
;Y:GOTO 2        22:B=(I-P*D/N)/(E
5:D=D+C*X:E=E+C*  -D*D/N):A=EXP
X*X:F=F+C*Y:G=    (P/N-B*D/N)
G+C*Y*Y:H=H+C*   24:R=(N*I-D*P)/J<
X*Y:N=N+1*C:IF   (N*E-D*D)*(N*Q
J*T=1RETURN      -P*P)):IF W=1
6:IF J<>1LET M=M  RETURN
+C*LN X:O=O+C*   26:GOTO 96
LN X*LN X:K=K+   30:"S"IF W=1THEN
C*Y*LN X         34
7:IF T<>1LET P=P  31:U=3:PRINT "LOG
+C*LN Y:Q=Q+C*   . Y=B*LN X+A":
LN Y*LN Y:I=I+   IF J=1GOTO 61
C*X*LN Y         32:B=(K-M*F/N)/(O
8:IF J+T=0LET S= -M*M/N):A=(F-B
S+C*LN X*LN Y    *M)/N
9:RETURN         34:R=(N*K-F*M)/J<
10:"L"IF W=1THEN (N*Q-M*M)*(N*G
14               -F*F)):IF W=1
11:U=1:PRINT "LIN RETURN

```

STATUS 1 at end of listing indicates the number of bytes utilized.

```

36:GOTO 96      : INPUT "DELETE
40:"D"IF W=1THEN LAST PT (Y/N)
44              ";Z$:IF Z$="Y"
41:U=4:PRINT "PWR THEN 75
. Y=A*X^B":IF 71:PRINT "PT# ";N
T+J>0GOTO 61   : INPUT "X=";L:
42:B=(S-M*P/N)/(O IF L<=0LET J=1
-M*M/N):A=EXP 72:X=L:INPUT "Y="
(P/N-(B*M/N))  ;Y:GOSUB 98:
44:R=(N*S-M*P)/J< GOTO 77
(N*O-M*M)*(N*Q 75:PRINT "X=";X;"
-P*P)):IF W=1   Y=";Y:INPUT "
RETURN          OK TO DEL(Y/N)
46:GOTO 96      ";Z$
48:"F"IF J+T>0 76:IF Z$="N"THEN
THEN 61        71
50:W=1:U=0:U=0: 77:C=-1:IF Y<=0
FOR Z=1TO 4:    LET T=1
GOSUB (Z*10):C 78:GOSUB 5:END
=R*R:IF C>U    82:"J"PRINT "ESTI
GOSUB 99       MATE OF Y":
55:NEXT Z:USING : INPUT "X=";L:
W=0:BEEP 3:    GOTO (U+85)
PRINT "BEST F  86:U=B*L+A:GOTO 9
T WAS #";U     7
59:GOTO (U*10) 87:U=A*EXP (B*L):
60:"K"PRINT "ESTI GOTO 97
MATE OF X":    88:U=B*LN L+A:
INPUT "Y=";U:  GOTO 97
GOTO (U+61)    89:U=A*L^B:GOTO 9
61:BEEP 1:PRINT 7
X OR Y OR BOTH 96:GOSUB 67:PRINT
NEG."":END     "A=";A:PRINT "
62:L=(U-A)/B:GOTO B=";B:PRINT "R
97              =" ;R:C=R*R:
63:L=(LN U-LN A)/ PRINT "R^2=";C
B:GOTO 97      :END
64:L=EXP ((U-A)/B 97:PRINT "X=";L;"
):GOTO 97      Y=";U:END
65:L=EXP ((LN U- 98:PRINT "X=";X;"
LN A)/B):GOTO  Y=";Y:RETURN
97              99:U=Z:U=C:RETURN
67:USING "####.# STATUS 1
##":RETURN
70:"G"BEEP 1:   1677
PRINT "DELETE"

```

PUT THOSE SOFTKEYS TO WORK FOR YOU!

As George Fergus was quick to discover, you can quickly set up the Sharp PC-1500 to save yourself a lot of keying when developing programs. Just assign the most commonly used punctuation symbols that normally require two keystrokes (SHIFT and then the desired character) to a group of softkeys. For instance, using all six keys from left to right, you could define them to represent the symbols:



















! , ; ? # \$

Use the RCL key too, to display what you have assigned to each softkey to assist in remembering your assignments. The technique sure saves a lot of work.

LCD GRAPHICS FOR THE SHARP PC-1500

David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203, sent in the codes shown in the accompanying listing for producing a variety of graphics on the Sharp PC-1500 liquid crystal display.

The codes are presented within DATA statements. A simple routine at the start of the listing (lines 1 — 100) may be used to access the data statements and display the various patterns. Naturally, you can extend the concepts to produce your own new patterns and incorporate them in your own programs. Note the complete set of chess pieces provided by Dave. Does that give anyone ideas? See pages 91 and 92 of the Sharp PC-1500 Instruction Manual for details on producing LCD graphics.

1:RESTORE 1000	1340:DATA "434C34		1610:DATA "40787E
5:DIM X\$(0)*80	2A161961":		7F7E7840":
100:CLS :WAIT 100:	REM SPIRAL		REM BUILDI
GCURSOR 0:READ	1350:DATA "04324A		NG
X\$(0):GPRINT X	14292610":		
\$(0):GOTO 100	REM SPIRAL		
1000:DATA "081436	1360:DATA "1C3E7F		
49361408"	7F7F3E1C":		
1010:DATA "776355	REM BALL		
08556377"	1370:DATA "1C2241		1620:DATA "3F7F79
1020:DATA "553677	4141221C":		79393F3F3838
00773655"	REM CIRCLE		787C7C3838":
1030:DATA "1C0055	1380:DATA "63594F		REM LOCOMO
4955001C"	454F5963":		TIVE
1040:DATA "080014	REM BELL		
002A0055"	1400:DATA "0E7C7E		
1050:DATA "775577	7C0E":REM		
00775577"	ROOK (B)		
1060:DATA "7F2214	1410:DATA "080C4E		1630:DATA "3C7C7C
0814227F"	677E":REM		3C3C7C7C3C10
1070:DATA "63411C	KNIGHT (B)		":REM BOXC
141C4163"	1420:DATA "4C5E3F		AR
1080:DATA "493622	5E4C":REM		
49223649"	BISHOP (B)		
1090:DATA "775077	1430:DATA "427C7F		1640:DATA "020302
22775077"	7C42":REM		1C0C5C7C6C08
1100:DATA "364949	QUEEN (B)		10204040":
36494936"	1440:DATA "407A7F		REM KANGAR
1110:DATA "7F415D	7A40":REM		OO
555D417F"	KING (B)		
1120:DATA "1C2A49	1450:DATA "1C1E1F		
7F492A1C"	1E1C":REM		
1200:DATA "0E1121	PAWN (B)		
4221110E":	1500:DATA "0E7C42		1650:DATA "010101
REM HEART	7C0E":REM		0909191F1919
1210:DATA "081C4A	ROOK (W)		18181C1E1A06
7F4A1C08":	1510:DATA "080C4A		060706060202
REM CLUB	257E":REM		02":REM EN
1220:DATA "081422	KNIGHT (W)		TERPRISE
41221408":	1520:DATA "4C5233		
REM DIAMON	524C":REM		
D	BISHOP (W)		
1230:DATA "183C1E	1530:DATA "427C43		1660:DATA "1A1E1F
7F1E3C18":	7C42":REM		1C0C04040404
REM SPADE	QUEEN (W)		04040E0E0E6F
1300:DATA "1D2222	1540:DATA "407A4F		6F7F7B7B6860
4122221D":	7A40":REM		60":REM KL
REM TAURUS	KING (W)		INGON
1310:DATA "21322C	1550:DATA "1C1211		
20207C20":	121C":REM		
REM JUPITE	PAWN (W)		
R	1600:DATA "3F0B7F		1670:DATA "081412
1320:DATA "062979	1F1F1F7F":		122448102040
2906":REM	REM ELEPHA		000000402010
FEMALE	NT		102040000000
1330:DATA "304848			402010102040
48350307":			0000004020"
REM MALE			

FROM THE WATCH POCKET

This column will be somewhat longer than usual. I have been developing a list of matters to discuss as a result of inquiries, industry developments and my own findings. So, here goes . . .

The Panasonic/Quasar HHCs

I have been looking over a Panasonic HHC lately, equipped with 4K of user memory in the basic unit plus a 8K external module. Do you think that means I have a "12K" system? Not really, the memory is "paged" or partitioned so that you are either using the 4K section or the 8K block.

I have spent about 35 - 40 hours actually operating the unit, much of that devoted to keying in BASIC programs and running some evaluations and benchmarks.

The application programs supplied with the unit, such as the time-keeping and appointment system and the personal filing system, are functional and easy to operate.

But I am somewhat disappointed with the BASIC operating system. Panasonic, in my opinion, would be wise to study Sharp's approach in this area, starting right with the editing capabilities.

A feature I find particularly annoying on the Panasonic when listing a BASIC program is the following: on long lines the display automatically starts scrolling to the end of the line. It has been my experience when scanning program listings on a PC that I can generally follow the flow of a program by reviewing the beginning of each line (rather than the end of lines). The extra key pressing needed on the Panasonic to stop the across-the-line-scrolling and get back to the start of the line or merely abort the display to get to the next line is inconvenient.

If you edit a line on the Panasonic you must press the ENTER key to have the unit accept the changes. However, as soon as this is done, the line disappears from the display. To check the line again, often desirable after alterations have been made, you must re-enter the entire LIST directive with the desired line number.

Also, when reviewing a program, while it is possible to roll the display from one line to the next in ascending line number order, you cannot roll back. Once you have gone beyond a line, the only way to get back to it is to abort the listing and enter a new LIST directive.

Keying in, editing and debugging BASIC programs, based on my experience, requires considerably more work than on a Sharp PC-1500. There are several factors that combine to produce this effect.

First, I find that the lack of a numeric/operand keypad slows down proceedings. All mathematical operators such as +, *, /, etc., require the pressing of the shift key and then the appropriate symbol. This is also the case for other commonly used characters such as the colon and semi-colon. The extra keystrokes make a difference when keying in long programs. Being able to quickly set up softkeys to reduce the keystrokes in this situation (as can be done on the Sharp PC-1500) would be a big boon.

The Panasonic keyboard normally operates in the lower case mode. BASIC statements and expressions are automatically converted to upper case when listed by the system - except when you have enclosed something in quotes - then the characters remain in lower case. I found this convention disconcerting and confusing at times. This might not be the case for someone who has not used other PC systems.

The trace facility for debugging of programs is rudimentary and nowhere near as powerful and convenient to use as the Sharp PC's capabilities.

I don't like not being able to save programs and data on a mass storage device such as a tape cassette. Apparently no such devices are available for the Panasonic at this time. You can copy files from one memory bank to another. These banks are battery powered and the batteries can even be changed (quickly!) without losing the contents of memory. However, at a list price of \$325.00 for an 8K module, I don't want to use them for archive purposes!

It also appears that once you leave the BASIC operating system, all the variable values assigned to a BASIC program are lost. Going back, for instance, to check an appointment on the built-in appointment organizer, means terminating all calculations and values in whatever BASIC program you might be running.

Finally, there appears to be a slight bug in the BASIC interpreter on the unit I have been using. If you have a Panasonic HHC, try entering the following statements using the Microsoft BASIC capsule:

```
10 X=1.23:XS=STR$(X):X1=VAL(XS):PRINT X1
```

Execute the sequence and let me know if you get a strange result. Maybe we have something that should be reported to Friends Amis.

So what is the bottom line? I don't feel this is a particularly good unit for people who just want to putter around with a PC using BASIC. The machine does not appear to be well suited for this purpose. Pick something a little "friendlier" and less expensive. But, am I saying the Panasonic HHC is a bad unit? Nope, not at all. It is just that it is designed for certain applications more than others!

Where It Shines

This unit seems best suited for applications where the software has been designed by skilled professionals to operate in essentially a "canned" mode. Such programs can be provided on ROM capsules that are simply plugged into the back of the unit. An example of one such application was demonstrated recently to me by Dr. Brown of American Medical Instruments, PO Box 1080, El Cerrito, CA 94530, telephone (415) 525-1113. This firm has a program supplied on a ROM capsule that determines the correct I.V. (intravenous) drip rate for a variety of commonly used drugs based on the patient's body weight, desired dosage, etc. The program operates in a variety of modes. It can serve as a drug-determining calculator to provide the proper drip rate. It can be placed in a mode where it actually issues audible beeps so that the I.V. drip may be synchronized to a specific rate. Or, an operator can press a key each time a drop comes out of the I.V. and the unit will display the rate, etc. Presto, you have an application that is admirably suited for the machine. Complex calculations, often performed at times of stress (medical emergencies are not picnics for doctors and nurses), can now be performed without error using the proper dosages for each type of drug (as stored in the PC's memory).

The fact of the matter is that the Panasonic/Quasar HHC is really a high performance, industrial grade machine. It is designed to be customized to specific applications by professional software developers. The unit is fast. In a test I conducted using Microsoft BASIC, running a program that performed a wide variety of operations from string handling to number crunching, the Panasonic HHC was about three times faster than the equivalent program on a Sharp PC-1500. BASIC programs can be compiled to run even faster using a BASIC - SNAP translator about to be made available for the machine. In critical arenas and commercial or industrial environments this significant speed advantage may well be worth the price.

Friends Amis, 505 Beach Street, San Francisco, CA 94133, the HHC's system software developer, has a version of the SNAP compiler available for commercial software developers that runs on an Apple II computer. Programs for the HHC can thus be developed and program-mable ROMs burned in an environment that is more efficient than using the HHC itself to develop sophisticated software.

Panasonic is reportedly moving full steam ahead on developing additional peripherals to tie-in with the HHC. A four-color printer/plotter unit is reported to be scheduled for release this summer with a retail price in the vicinity of \$500.00. A 160K-byte 3-inch diskette cartridge unit is apparently in the works with initial deliveries due near the end of the year.

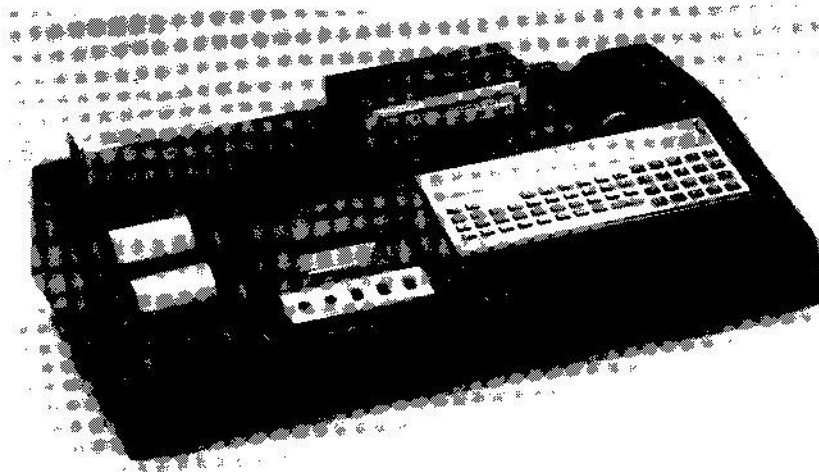
If you have a professional or business application and need a high-performance, quality HHC, take a good look at this unit. Software houses looking for a system conducive to being set up for specific applications might also be well advised to look the machine over carefully. The investment necessary to produce ROMs for serious applications will probably reduce the competition - maybe even keep it at a level where some people can turn a profit!

PC-1500 Tid-Bits

A lot of readers are hard at work analyzing the ROM code. It would be nice if key routines that could be used as subroutine calls could be located and identified. The recently released *Sharp PC-1500 Service Manual* does not contain information related to the ROM or even give the instruction set of the customized CPU. However, it does indicate

A UNIQUE SYSTEM ORGANIZER

Designed for the Sharp PC-1211 or Radio Shack TRS-80 Pocket Computer (PC-1) equipped with the printer/cassette interface, this *P/C Desk Console* has room for three cassettes, storage of 3 by 5 inch index cards, two rolls of paper and a spare printer ribbon as well as the cassette interface cables. It all measures just 8-1/2 by 16 by 2-3/4 inches ready to keep your system organized and easy to use. The price is \$19.95 (plus \$2.50 s/h and 6% tax if delivery is in California). Order from Fox/Walker, 4650 Arrow Highway, Building G-17, Montclair, CA 91763. Tell 'em you saw it in PCN.



that the CPU contains the following registers: a 16-bit program counter, stack pointer and "W" register, an 8-bit accumulator and an 8-bit "E" register (probably for flags or extension of the accumulator), and six more 8-bit registers UH and UL, XH and XL and YH and YL that are general purpose and can apparently be paired to form 16-bit data (pointer) registers.

The consensus amongst several experimenters so far is that the machine code BE represents a subroutine call and is followed by two bytes containing the absolute address (high then low address bytes). The code E2 is likely a subroutine return directive. The code AE seems to be "store the accumulator" at the address shown in the next two bytes. B5 apparently is a "load the accumulator" with the data in the next byte.

With what is now known it should be possible to eventually ascertain the machine code instruction set for this device. The key is to share your findings so that others can build upon earlier findings.

Lots of people are sending in information about undocumented capabilities. It is a little too early to attempt to summarize or publish all this material, but take notes and keep us informed of your discoveries. *Thomas S. Cox* has apparently figured one feature out

that had a number of users climbing the walls. His words are a lot more informative than the manual:

"Merging several programs having the same line numbers is possible if the programs are labeled and either the RESERVE or DEF modes are used to access the programs. When using this method, the programs can be run with no problems. I should emphasize that the labels must be present in the tape version of the program as it is not possible to edit the first programs in a series of MERGED programs unless the programs are entered such that the line numbers are in ascending order as the programs are loaded. By using non-conflicting labels, it is possible to merge and run programs without having to go to the trouble of renumbering the programs and entering them in precise order." - Thanks, Thomas, that is a nice clarification.

There is other information of this nature coming in and I will try to sort it out and get it disseminated to you as soon as practical.

This and That

Roland J. Saam sends word from England that Sharp has a new product for the PC-1500 designated the CE-153. It is called a "Software Board." It is really a big matrix of 140 keys arranged as 10 rows in 14 columns. You can place a template over it and use software to define each key.

The CE-155 8K RAM module is available now in the United States. I have one installed and it sure is nice to have 10K available on my PC. The list price is \$150.00. Shop around.

Sharp is beginning to advertise the PC-1500 in some of the computer and electronic trade journals. So is Panasonic. What took them so long?

Nanos Systems Corporation, PO Box 24344, Speedway, IN 46224, has introduced the TRS-80 BASIC Pocket Computer System Reference Card (which also is suitable for the Sharp PC-1211). The price is \$2.95 (plus 4% tax in Indiana). This is a reference card that folds to fit in a pocket. Expanded it measures 19 by 8-1/2 inches and is printed on both sides with information such as instructions, error codes, system limitations and so forth. Might be especially suitable for beginning programmers.

Radio Shack has announced the release of *Games II* for the TRS-80 Pocket Computer. The package includes eight games on two cassettes plus an instruction manual. The games are: Missile Marksman, Baccarat, Blackjack, Aceydeucey, One-Armed Bandit, Pokerslot, Numguess and Craps. The list price is \$14.95.

If the review of the Casio FX-702P in Issue 09 of PCN wasn't enough for you, *Bob McElwain* has written a splendidly thorough review of the unit and its capabilities in the May, 1982 issue of *Interface Age*. It starts on page 88 therein. Read it if you have any thoughts about obtaining the FX-702P.

Remember, the next issue of PCN will not be out until July. But, there is a good chance there will be a special Advertisers Edition in the interim and we are planning a few surprises for the July edition.

Help keep your fellow readers informed by keeping us informed.

- Nat Wadsworth, Editor

SUBSCRIPTION INFORMATION

Subscriptions are sold on a calendar year basis (January - December). There are ten issues per year. You get all issues published for the calendar year to which you subscribe.

☐ 1982 Regular Subscriber (Issues 11 - 20). \$30.00 in U.S. (U.S. \$36.00 to Canada. U.S. \$45.00 elsewhere.)

☐ Sample issue. \$3.00 in U.S. (U.S. \$4.00 elsewhere.) *Due to credit card minimum, this item cannot be charged.

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER. Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA # _____ Expires: _____

Signature: _____



35 Old State Rd, Oxford, CT 06483

POCKET COMPUTER NEWSLETTER



© Copyright 1982 — Issue 16

July

HEWLETT-PACKARD ANNOUNCES PROGRAMMER'S AID

Beginning July 1, 1982, Hewlett-Packard Company will be providing a new "engineering" calculator designed expressly for computer programmers and digital-design engineers. The unit marks a new trend in supporting a highly specialized market by the firm.

The new programmable model has been designated the HP-16C. It is able to handle a wide variety of computer-science problems such as format conversions, bit extraction and *simulation of selected micro-processor instructions!* The register size may be specified by the user, to a maximum of 64 bits, and thus the unit can be set to match the word size of virtually any commercial processor.

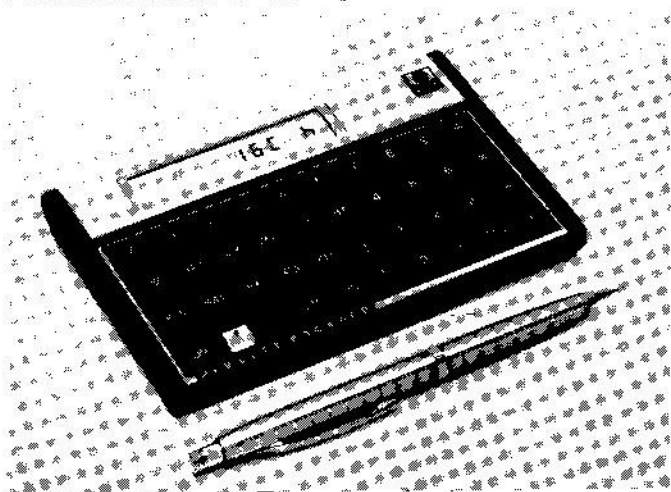
The unit may be placed in an "integer" mode and used to perform bit calculations. In this mode numbers may easily be entered in and converted between four different bases: hexadecimal, decimal, octal and *binary*. It is believed to be the first time that binary notation has ever been made available in a stand-alone handheld calculator. In the unsigned mode, set to a 64-bit word size, numbers up to $2^{64}-1$, which is 18,446,744,073,709,551,615 in base 10, may be represented. The liquid-crystal display scrolls right and left to permit viewing of large numbers.

Users can select whether numbers are to be interpreted as one's-complement, two's-complement or unsigned values. Among other benefits, this capability is said to be a boon to computer science students who sometimes have difficulty learning to distinguish among the three conventions.

The new unit utilizes Reverse Polish notation (RPN) logic and has the same floating-point decimal arithmetic features found in other HP machines. There are also 18 different bit-manipulation functions and four Boolean operators provided as special features in the unit. The programmable portion can retain up to 203 program lines or 101 sixteen-bit data registers.

The suggested U.S. retail list price has been announced as \$150.00.

Photo Hewlett-Packard HP-16C Programmer's Calculator



UPPER CASE CONVERSION ROUTINE

This routine was submitted by: *David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203*. It is particularly valuable on the Panasonic/Quasar HHC, but can also be applied on other PCs.

The routine shown in the listing will automatically convert characters in a string from lower case to upper case.

Line 10 accepts the input string (A\$) and clears out an area (B\$) for the conversion to upper case. Line 20 sets up a loop to extract the characters in the input string, one at a time. Line 30 extracts each character and stores the ASCII value.

Line 40 is the heart of the routine. It forces the ASCII value of the characters to stay in the range 32 to 95. The phrase "C AND 95" provides an upper limit of 95. If the initial value is less than 64, the remainder of the phrase within parenthesis adds 32 to the ASCII value. The resulting modified ASCII value is changed back to a character (now upper case) and is concatenated to the B\$ string.

Line 50 completes the scanning loop. Line 60 displays the original input and the converted output.

Some beginning BASIC programmers may not be familiar with the AND function used in line 40. Typically, this logic function is used in IF statements such as IF A=0 AND B=1. However, it may also be used as a numerical operator. This is how it is used in this application.

The number preceding the AND declaration and the number following it are considered to be in binary form (as far as the computer is concerned). Each bit of the two numbers is compared against each other. If both bits are a 1, the result is a 1. Otherwise, the result is 0. Stated logically: if A is true AND B is true, then the result is true. If either A or B (or both) are false, then the result is false.

For example:

0 1 1 1 0 0 1 0	represents lower case "r"
0 1 0 1 1 1 1 1	is 95 in binary format
0 1 0 1 0 0 1 0	the result of "r" AND 95

This result is indeed the ASCII representation of upper case R.

So, now you have one more little routine to put in your programming bag of tricks. Note that this routine is particularly valuable when inputting data from the normally lower case HHC as it converts letters of the alphabet, but it does not affect the inputting of numerical digits!

Program Upper Case Conversion

```
10 INPUT A$:B$ = ""
20 FOR I=1 TO LEN A$
30 C$=MID$(A$,I,1):C=ASC C$
40 B$=B$+CHR$(C AND 95 + 32 * (C < 64))
50 NEXT I
60 PRINT A$;" ";B$:GOTO 10
```


UNDERSTANDING THE SHARP PC-1500

This is the second article in a series being presented by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

An Introduction to POKE

In Issue 15 of *PCN I* explained that the PEEK function returned the byte that was stored in the address used as an argument. It is also possible to store a specified byte at a particular memory location, by using the POKE instruction.

For example, execution of POKE &79FF,0 will store a zero byte in the memory location &79FF and thus place the PC in the LOCK mode. (Try it!) As you would expect if you read the previous article, performing a POKE &79FF,&60 will cancel the LOCK mode.

A multiple POKE directive is possible on the PC-1500. When the directive POKE &7900,1,0,116,104,80,0,0 is executed, the byte 01 is stored into location &7900, 00 goes into &7901, and so on. This particular example will store the numerical value 78.685 into the variable A as discussed in the previous article.

Practical Applications

It may not be immediately obvious that there are a number of ways in which the PEEK and POKE instructions can be quite useful. The following examples should give you some notion of the kinds of possibilities.

For instance, certain memory locations act as *pointers*. That is, they contain the hexadecimal addresses of various memory locations needed by the computer, such as the location of the next instruction to be executed or the location of a stored variable.

Suppose you have executed the NEW command, effectively clearing your program from memory, and then realize you still want to use that program. Is it possible to recover the lost program? It is, provided that prior to having executed NEW you have saved the numbers obtained by PEEK &7867 and PEEK &7868. The contents of these locations act as a pointer to the memory address of the last byte in a program! The use of the NEW command does not actually erase a program from memory. It actually just resets some pointers so that the computer "thinks" that the memory is empty.

To test this, put a small program into the PC-1500 and write down the results of the two PEEKs just mentioned. Now execute NEW. To resurrect your program: (1) POKE the two previously saved PEEKed values back into their original locations and (2) POKE &40C5,0. (Note, if you have an 8K CE-155 memory module installed, step (2) should be POKE &38C5,0.) If the first line number in your program was larger than 255, you will find that it has been changed. You can edit it back to its original value.

You can also restore variables in main memory using a similar procedure. For instance, with some variables stored in main memory (created using two-character names), record the values obtained using PEEK &7899 and PEEK &789A. If these same values are POKEd back into those locations after main-memory variables have been "cleared" by CLEAR, RUN or NEW, the original variables will be restored. You see, the pointer in locations &7899 and &789A simply contains the starting address where variables are stored. (Note that this does not apply to the fixed-memory variables A through Z and A\$ through Z\$.)

Print formats may be set using the POKE directive instead of a USING statement, with the advantage that a calculated value may be used to determine the number of spaces formatted. Here is how the format settings are stored in memory:

- &7895 The byte stored here is the sum of the following format specifications: 128 for scientific notation, 64 for asterisk fill, 32 for forced sign, 16 for comma separation. (Note: when set by USING, the computer adds 1 to these values, except when the format is simply "^.")
- &7896 The number of positions reserved preceding the decimal point, including one for the sign.
- &7897 Length specified for strings, if any.
- &7898 The number of positions reserved following the decimal point, including one for the decimal point itself.

Thus, POKE &7895,32,7,0,3 would be equivalent to the statement USING "+#####.###".

Fooling the Printer

If you have the CE-150 printer connected, you can PEEK &79F4 to see what CSIZE is set. The largest size you can print is CSIZE 9, right? *Wrong!* Try POKE &79F4,50. Then LPRINT "G" and take a look at CSIZE 50! Now set the computer to GRAPH mode, execute the statement ROTATE 1, then POKE &79F4,36. Now execute LPRINT "HUGE". Be advised that if you continue this for very long, the PC will exceed its coordinate range resulting in ERROR 70. You can remedy this condition by executing SORGN so that it has a new origin. You might also discover that the computer interprets CSIZE 0 as CSIZE 256. The only thing printable in this size is an immense decimal point, obtainable by specifying LPRINT ".".

The COLOR setting is stored in location &79F3. Do *not* POKE into this location. You will only confuse the computer into thinking that the pen turret is positioned differently from where it actually resides.

[Note: Be careful using the POKE directive around memory locations used to control external units. It is conceivable that you might damage an external device if you inadvertently left a selenoid activated or otherwise interfered with the normal programmed operation of the device. At the very least, be prepared to immediately shut the unit off if strange sounds or activities come forth as the result of a POKE directive. — N.W.]

More PEEK and POKE Locations

I have located a number of memory addresses used by the PC-1500 using a process of trial and error. And, of course, there are quite a few addresses used for purposes that I have not yet figured out. The accompanying table lists those found recently. Be forewarned that if you poke values into certain locations that attempt to have the computer do something that is not possible, you may get a "crash." The PC may then completely lock up and you will have to use the ALL RESET switch to restore operation.

A Surprise PEEK

A number of PC-1500 fans are investigating the ROM coding, obtaining the hexadecimal codes and/or interpretations of these as ASCII characters. Here is another way of looking at the ROM that you may not have considered. The program illustrated can be used to display ROM codes as graphics on the LCD.

```
10 INPUT "BEGINNING ADDRESS?";A
20 WAIT 0:FOR C=0 TO 127:GPRINT PEEK(A+C):NEXT C:WAIT:
  PRINT
30 A=A+128:GOTO 20
```

Try using &C000 as the initial input to the beginning address prompt. When you get bored with this, take a look at the results when you use &FCA0 as a beginning address!

POINTERS

- | | |
|------------|---|
| &7863 | Beginning of memory. This location contains the first byte of the two-byte address of where memory begins. Only one byte is stored because the second byte is always zero. The beginning of memory can vary depending on what expansion module, if any, is plugged into the PC. |
| &7864 | Top of memory. Again, the second byte is always zero. This is the address of the first byte following the end of "main" (user) memory. |
| &7865 — 66 | Beginning of program. This is the address following RESERVE memory. It is where the first program line of a user program is stored. |
| &7867 — 68 | End of program. The computer stores a byte with the code &FF in the address pointed to by this pointer. Thus &FF is the byte immediately after the carriage-return (ENTER) code of the user's last program line. |
| &7869 — 6A | Beginning of merged program (a program loaded from cassette using the MERGE command). |
| &7899 — 9A | Beginning of variables. |

POINTERS USED DURING PROGRAM EXECUTION

- &789C — 9D Hexadecimal value of line number being executed.
- &789E — 9F Beginning of the program being executed. (It is the start of a merged program if that is what is being executed.)
- &78BE — BF Pointer to address at which next READ statement will obtain data. If none, the value C0C8 is held in the pointer.

SIX-BYTE POINTERS USED IN PROGRAM EXECUTION

Each of these pointers, starting at the location shown, contains six bytes. The first two bytes specify the storage address in memory; the next two contain the program line number; and the last two hold the memory address of the first byte of the program being executed.

- &78A0 Address of next program byte. It is the next byte to be executed unless a transfer (GOTO, etc.) has been specified.
- &78A6 Address of next byte to be executed after a transfer.
- &78AC Address where execution is to resume after a halt for input, etc.
- &78B2 Address of program location at which an error occurred.
- &78B8 Transfer destination specified by ON ERROR GOTO directive. When cancelled by RUN, &80 is added to the first byte of this stored address.

STORED MODES AND SETTINGS

- &764E Status of prefix keys; specification of RESERVE group.
- &764F Operating mode and trigonometric mode.
- &7863 Beeper on/off. Set to 0 for on, 1 for off.
- &7871 — 73 WAIT specification.
- &7874 — 75 Position of display cursor (G-CURSOR).
- &788D Trace mode. TRON = 96, TROFF = 0.
- &7895 — 98 Format (USING) specifications. (See text.)
- &79FF Lock mode. LOCK = 0, UNLOCK = 60.
- &7B0A — 0C Automatic 7-minute shutoff timer. Reset to value &FE 1D 1D whenever activity halts. Shuts PC off when value &FF FF FF is passed.
- &7B0D Timer for cursor flashing.
- &7B0E Type of program execution in progress. Continuous execution = 1, single-line execution (stepped by user with the line scroll-down key) = 193, and rapid execution when scroll-down key is held depressed = 225.
- &7B0F Keyboard matrix code for the key that initiated program execution. This would be ENTER or user-defined or a RESERVE key.

LOCATIONS RELATED TO PRINTER OPERATION

- &79E0 — E1 The X-coordinate relative to the origin when in the GRAPH mode. Negative numbers are stored in complemented form.
- &79E2 — E3 The Y-coordinate.
- &79E4 — E5 Paper reverse counter. Used to insure that paper does not get backed up more than about 4 inches.
- &79E6 Location (horizontal) of pen (0 — 216).
- &79E7 — E8 In GRAPH mode, the amount by which the horizontal coordinate exceeds the range in which printing may take place. This is in hexadecimal notation with the low byte first.
- &79E9 When paper is fed by the computer the value 15 is stored here. When fed manually, the value is 0.
- &79F2 ROTATE setting.
- &79F3 COLOR setting.
- &79F4 CSIZE setting.

CONVERTING PROGRAMS FOR THE CASIO FX-702P

Several PCN readers have expressed dismay over the lack of programs available for the Casio FX-702P. A recent survey of PC users leads us to believe that the situation is not likely to improve substantially in the immediate future. There are simply relatively few FX-702P users compared to Radio Shack and Sharp owners, according to our findings.

But, FX-702P owners need not despair. The fact is that most of the programs published to date in PCN are readily adapted for use on the Casio PC by making some editing changes as programs are keyed in.

The following suggestions and tips may be helpful to Casio FX-702P users who wish to make such program adaptations.

1.) Substitute appropriate statement/function mnemonics where applicable. Most of these types of alterations are pretty obvious. Thus, for instance, GOSUB is changed to GSB, PRINT to PRT, RETURN to RET, and so forth. A few may not be quite so apparent. The PAUSE statement becomes WAIT (x) where (x) specifies the length of the pause. CLEAR must be changed to VAC on the 702P. You need to type in SQR in place of the square root ($\sqrt{\quad}$) symbol. Also, remember that there is a special inequality symbol (\neq) on the 702P that takes the place of having to use the "less than" and "greater than" ($<$ $>$) symbols to express the inequality operation on a Sharp or Radio Shack PC.

2.) Forget about functions that do not exist on the FX-702P — such as BEEP.

3.) Shorten/abbreviate display messages to fit in the 20 character FX-702P display (versus the 24 characters on the PC-1).

4.) If subscripted variables are used in a program, remember the following:

A.) Subscripted variables having subscripts in the range 1 — 26 are really just another way of declaring the use of the variables A — Z on the Sharp/Radio Shack original PCs. Thus A(1) is the equivalent of specifying variable A on a FX-702P, A(2) = B, ..., A(26) = Z. Make the appropriate substitutions!

B.) Subscripted variables with subscripts above 26 can be converted to array elements on a FX-702P using the formula (I-26) where "I" represents the subscript on the Radio Shack/Sharp unit. I.e., A(27) = A(27-26) = A(1), A(28) = A(28-26) = A(2) on the 702P, etc. Simple, isn't it?

5.) Use separate IF statements in place of multiple logic expressions. An expression such as "IF (A < 52) + (B > 60) THEN 400" on a Radio Shack TRS-80 PC means "if A less than 52 OR B greater than 60 then go to line 400." On the FX-702P this must be precisely defined using two separate (consecutive) IF... THEN statements, such as:

```
IF A < 52 THEN 400
```

```
IF A > 60 THEN 400
```

The opposite logic operation (AND) would appear in a TRS-80 listing as "IF (A < 52) * (B > 60) THEN 400" which indicates that "if A is less than 52 AND B is greater than 60 then go to line 400." Such a statement can be re-arranged so that it can be stated using two consecutive IF statements on a 702P in this manner:

```
IF A >= 52 THEN *** {*** = condition not met}
```

```
IF B > 60 THEN 400
```

*** (line to which program goes when condition is not met!)

Note that the first IF statement in this example reverses the original test so that the program can "fall" directly into the second IF statement if the first part of the AND operation is met.

6.) Explicitly state all mathematical operations using their appropriate symbols. This means eliminating all instances of "implied" multiplication and parenthesis which sometimes are inferred in TRS-80 and Sharp PC programs. That is, an expression such as:

ATN((ABC + (D - E

should be completely specified by inserting appropriate multiplication signs and adding closing parentheses, thus:

ATN((A*B*C) + (D - E))

Use the six steps outlined in this article as a guide and you will find you can readily adapt most PC programs written for other machines so that they will execute on your FX-702P. In the process, you may even find you *understand* (and can thus adapt and customize) the program better than those who simply load the programs "to see what happens."

A COMPARISON OF PC MEMORIES

The Radio Shack TRS-80 and the Sharp PC-1211 pocket computers have 204 registers available for the storage of data and programs. Each register can hold eight bytes. The first 26 registers can only hold *data* (not program steps). Registers can hold numbers of up to 12 digits, a two-digit exponent (powers of ten) and the signs of the mantissa and exponent. Each digit or sign takes up 4 bits of space or a *nibble*. Alternately, a register may store up to seven alphanumeric characters. Each character is represented as 8 bits or a *byte*. The eighth byte in the register is used to indicate that the other bytes are representing *string* (alphanumeric) information.

The remaining 178 registers can hold data or program steps. Program steps begin at the last register (number 204) and proceed to be stored in descending (numbered) registers. Each program step uses one byte of storage. The *curtain* between program memory and data memory is moved automatically as program steps are created or deleted.

Registers may be cleared in two ways, depending on which type of register is involved. The NEW command effectively clears all program and data registers. The CLEAR command only affects the data registers.

If a data register contains a numerical value, such as 123, then trying to refer to that register as a string will result in an error condition.

A Look at the Casio FX-702P

This unit has 236 data and program registers. Twenty-six of these are reserved strictly for data purposes and 10 are reserved for program steps. The remaining 200 may be used for either data or program storage.

The Casio uses the same method of storage as the TRS-80 and 1211. The *curtain*, however, must be specifically specified by using the DEFM command. This directive allocates up to 200 registers (by groups of 10) for data storage. It will not permit allocation of memory that is already in use for programs. But, when the curtain is moved to create more data storage, that memory is cleared.

The VAC command clears data memory in the Casio unit and the CLR command clears program memory.

If a data register contains a number and an attempt is made to refer to it as a string, there is no error condition. Instead, the computer returns the "null" string (as though there was nothing there).

The Sharp PC-1500 & Radio Shack PC-2

These units are different than the earlier PC-1211 and PC-1. They have 26 fixed numeric registers, 26 separate fixed string registers, and 1850 bytes of program storage. The numeric data registers behave the same as described for the other units. However, the string registers are twice as long, holding 16 characters.

The 1850 bytes of storage memory may also be used for data when needed. To do so, a two-character variable name is created. The first character of the name may be any letter from A to Z. The second character may be a letter or a digit. Adding the \$ symbol creates a string variable.

Arrays of one or two dimensions may also be created in memory through use of the DIM statement. Both numerical arrays and string arrays having up to 80 characters-per-element may be defined. The assignment of an array name takes seven bytes of memory. Additional memory is then used by each element of the array.

The CLEAR and NEW commands operate similar to the earlier PCs by these firms. However, variables (including arrays) in the program workspace are not set to zero by the CLEAR directive; that memory space simply becomes available for other use.

The RUN command, used to start a program, clears all variables in the program workspace, but it does not alter those in fixed memory.

The Panasonic HHC

The Panasonic does things differently than the other machines. It does not have any fixed registers. Each variable, including arrays, uses up program workspace as it is needed. There are three types of variables and array elements: floating-point numbers, integers and strings. Floating-point numbers have up to 9 digits of accuracy and may range

from 2.93873588E-39 to 1.70141183E+38 (plus or minus). Integer values are restricted to the range -32768 to 32767. Strings may contain up to 255 characters.

Each simple variable name (versus array variables) uses up seven bytes of memory. Floating-point variables utilize the first two bytes for the name and the last five for the numeric value. Integers use the first two for the name, the next two for the value, and leave the last three unused!

String variables use two bytes for the name, one for the length of the string, and two more for the memory address of the location of the first character of the string. The last two (of the seven) bytes are not used in a string assignment.

Array names use two bytes for the name, two for the array length, one for the number of dimensions and additional bytes for the size of each dimension. Then the storage of array elements begins. Floating-point elements require five bytes, integer elements two bytes, and string elements use three bytes in the array (length and location) with the actual strings being stored separately.

Summary

Each PC or HHC handles certain aspects of memory utilization in its own way. However, there is a clear trend towards providing substantially more memory, both in the basic unit and as add-on or plug-in modules. For instance, the PC-1500 and PC-2 can be expanded using plug-ins to (eventually) 18 kilobytes. The Panasonic, using external modules, can be equipped with up to 56 kilobytes!

An accompanying table compares the basic memory configurations of the units discussed.

Thanks for this comparative article go to: David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203.

Table Basic Memory Supplied in Popular Units

Model	Fixed Data	Flexible Memory	Fixed Program
1211/PC-1	208	1424	0
FX-702P	208	1600	80
1500/PC-2	624	1850	0
RL-H1400	0	3108	0

Glossary PC Storage Terminology

bit	The smallest unit of programmable memory, able to represent a zero or a one.
nibble	A group of four bits, able to hold a number from 0 to 15.
byte	Two nibbles or eight bits, capable of representing numbers in the range 0 to 255.
BCD	Binary Coded Decimal, a method of representing the digits 0 through 9 in four bits.
curtain	Also known as a partition. It is the hypothetical boundary between program memory and data memory.
fixed memory	Memory that is available for only one purpose, be it storage of data or programs.
flexible memory	Memory that may be used for either data or programs.
program memory	Memory in which program steps are stored.
data memory	Memory in which numbers or character strings are stored.

MUSIC ON THE SHARP PC-1500

Incredible! J.S. Bach's *Prelude Number II* from *The Well Tempered Clavier* as programmed by: Brian Peterson, 6807 N. Sheridan Road - Apt. 520, Chicago, IL 60626. Brian says he will write a tutorial article showing the rest of us how he makes this kind of music if there is enough interest. So, if that is the case, write and give him some encouragement!

Program Music on the Sharp PC-1500

```

10: DIM A(51), B(51)
15: FOR X=1 TO 51
20: READ A(X): B(X)=10*(X/40)+8
25: NEXT X
30: WAIT 25: PRINT "ONE": PRINT "AND"
35: PRINT "TWO": PRINT "AND": CLS
50: READ N: IF NBEEP 1, A(N), B(N): GOTO 50
51: BEEP 1, A(1), B(1): Y4
52: READ N: IF NBEEP 1, A(N), B(N): GOTO 52
53: READ N: IF NBEEP 1, A(N), B(N): Y1.5
54: BEEP 1, A(1), B(1): Y4
55: READ N: IF NBEEP 1, A(N), B(N): GOTO 55
56: READ N: IF NBEEP 1, A(N), B(N): Y1.5
57: BEEP 1, A(1), B(1): Y4
58: READ N: IF NBEEP 1, A(N), B(N): GOTO 58
59: READ N: IF NBEEP 1, A(N), B(N): Y1.5
60: BEEP 1, A(1), B(1): GOTO 60
61: BEEP 1, A(16), B(16): BEEP 1, A(23),
    B(23): BEEP 1, A(26), B(26): BEEP 1,
    A(28), B(28)
62: BEEP 1, A(32), B(32): Y28
63: READ N: IF NBEEP 1, A(N), B(N): GOTO 63
64: READ N: IF NBEEP 1, A(N), B(N): Y1
65: READ N: IF NBEEP 1, A(N), B(N): Y4
66: READ N: IF NBEEP 1, A(N), B(N): GOTO 66
67: BEEP 1, A(16), B(16): BEEP 1, A(21),
    B(21): BEEP 1, A(24), B(24): BEEP 1,
    A(28), B(28)
68: BEEP 1, A(33), B(33): Y28
69: READ N: IF NBEEP 1, A(N), B(N): GOTO 69
70: BEEP 1, A(32), B(32): Y1
71: READ N: IF NBEEP 1, A(N), B(N): Y1.2
72: READ N: IF NBEEP 1, A(N), B(N): GOTO 72
73: READ N: IF NBEEP 1, A(N), B(N): Y4
74: READ N: IF NBEEP 1, A(N), B(N): Y6
75: READ N: IF NBEEP 1, A(N), B(N): Y8
76: BEEP 1, A(32), B(32): Y2: BEEP 1, A(38),
    B(38): Y2: BEEP 1, A(32), B(32): Y38
77: END
100: DATA 255, 244, 228, 219, 204, 195, 184
    : 122, 163, 151
101: DATA 143, 133, 125, 118, 111, 103, 99,
    : 94, 88, 82, 77
102: DATA 72, 68, 64, 59, 55, 52, 49, 45, 43,
    : 40, 37, 35
103: DATA 32, 30, 28, 26, 24, 22, 21, 19, 18,
    : 16, 15, 14
104: DATA 12, 11, 10, 9, 8, 7
200: DATA 40, 31, 30, 31, 28, 31, 30, 31
201: DATA 40, 31, 30, 31, 28, 31, 30, 31
202: DATA 36, 33, 32, 33, 28, 33, 32, 33
203: DATA 36, 33, 32, 33, 28, 33, 32, 33
204: DATA 39, 33, 31, 33, 30, 33, 31, 33
205: DATA 39, 33, 31, 33, 30, 33, 31, 33
206: DATA 40, 35, 33, 35, 31, 35, 33, 35
207: DATA 40, 35, 33, 35, 31, 35, 33, 35
208: DATA 43, 36, 35, 36, 31, 36, 35, 36
209: DATA 43, 36, 35, 36, 31, 36, 35, 36
210: DATA 42, 34, 32, 34, 38, 34, 32, 34
211: DATA 42, 34, 32, 34, 38, 34, 32, 34
212: DATA 42, 35, 34, 35, 38, 35, 34, 35
213: DATA 42, 35, 34, 35, 38, 35, 34, 35
214: DATA 48, 32, 38, 32, 28, 32, 38, 32
215: DATA 48, 32, 38, 32, 28, 32, 38, 32
216: DATA 48, 33, 32, 33, 28, 33, 32, 33
217: DATA 48, 33, 32, 33, 28, 33, 32, 33
218: DATA 38, 33, 31, 33, 38, 33, 31, 33
219: DATA 38, 33, 31, 33, 38, 33, 31, 33
220: DATA 38, 35, 33, 35, 31, 35, 33, 35
221: DATA 38, 35, 33, 35, 31, 35, 33, 35
222: DATA 36, 35, 33, 35, 31, 35, 33, 35
223: DATA 36, 35, 33, 35, 31, 35, 33, 35
224: DATA 36, 38, 28, 38, 26, 38, 28, 38
225: DATA 36, 38, 28, 38, 26, 38, 28, 38
226: DATA 35, 26, 24, 26, 31, 26, 24, 26
227: DATA 35, 26, 24, 26, 31, 26, 24, 26
228: DATA 33, 28, 26, 28, 25, 28, 26, 28
229: DATA 33, 28, 26, 28, 25, 28, 26, 28
230: DATA 33, 28, 28, 38, 27, 38, 28, 38
231: DATA 33, 30, 28, 38, 27, 38, 28, 38
232: DATA 33, 38, 28, 38, 27, 38, 28, 38
233: DATA 33, 38, 28, 38, 27, 38, 28, 38
234: DATA 31, 28, 27, 28, 23, 28, 27, 28
235: DATA 31, 28, 27, 28, 23, 28, 27, 28
236: DATA 21, 31, 38, 31, 33, 31, 38, 31
237: DATA 21, 31, 38, 31, 33, 31, 38, 31
238: DATA 22, 28, 27, 28, 21, 28, 27, 28
239: DATA 22, 28, 27, 28, 21, 28, 27, 28
240: DATA 31, 28, 27, 28, 23, 28, 27, 28
241: DATA 31, 28, 27, 28, 23, 28, 27, 28
242: DATA 34, 28, 27, 28, 25, 28, 27, 28
243: DATA 34, 28, 27, 28, 25, 28, 27, 28
244: DATA 35, 28, 27, 28, 38, 28, 27, 28
245: DATA 35, 28, 27, 28, 38, 28, 27, 28
246: DATA 36, 28, 27, 28, 38, 28, 27, 28
247: DATA 36, 28, 27, 28, 38, 28, 27, 28
248: DATA 8, 15, 18, 21, 8, 24, 21, 28, 21,
    : 21, 38, 27, 24, 21, 28, 21
249: DATA 8, 16, 19, 23, 8, 28, 23, 22, 23, 31
    : 28, 35, 31, 28, 24, 23, 24
250: DATA 8, 13, 22, 28, 8, 31, 28, 27, 28, 34
    : 28, 37, 34, 31, 28, 27, 28
251: DATA 11, 8, 42, 48, 42, 43, 48, 38, 48, 3
    : 40, 39, 48, 42, 39, 33, 39
252: DATA 35, 38, 37, 39, 48, 37, 35, 37, 34,
    : 37, 35, 37, 39, 35, 34, 35
253: DATA 38, 47, 45, 47, 48, 45, 43, 45, 42,
    : 45, 43, 45, 47, 43, 42, 43
254: DATA 48, 43, 42, 43, 45, 42, 48, 42, 38,
    : 42, 48, 42, 43, 48, 35, 48
255: DATA 35, 48, 39, 48, 36, 45, 43, 45, 35,
    : 43, 42, 43, 33, 42, 48, 42
256: DATA 31, 48, 39, 48, 36, 33, 31, 33, 35,
    : 31, 38, 31, 33, 38, 28, 38, 8
257: DATA 28, 38, 32, 8, 33, 35, 36, 38, 48, 3
    : 8, 36, 35, 8, 33, 8, 35, 32, 8
258: DATA 35, 33, 32, 33, 35, 36, 35, 8, 33, 3
    : 3, 38, 31, 33, 38, 21, 33, 8
259: DATA 27, 4, 15, 18, 21, 24, 23, 21, 27, 2
    : 1, 38, 21, 27, 24, 23, 21
260: DATA 28, 29, 26, 29, 26, 24, 21, 24, 23,
    : 26, 23, 28, 24, 21, 18, 21
261: DATA 28, 23, 28, 16, 21, 18, 15, 18, 8, 4
    : 11, 16, 18, 28, 23, 26, 23
262: DATA 24, 28, 33, 38, 33, 36, 8, 48, 39, 4
    : 8, 8, 35, 33, 8, 38, 8, 8
263: DATA 8

```

STATUS 1

3403

STATUS REPORT

A number of PC-1500 users have discovered that in addition to the STATUS 0 and STATUS 1 functions available on the machine, there are several other STATUS-related capabilities. Here is a summary of STATUS functions now reported to be available:

- STATUS 0** As indicated in the PC-1500 Instruction Manual, this function tells how many bytes of program memory are available for storage of programs and data. This function does not take account of memory assigned to two-character variables or array elements.
- STATUS 1** As reported in the manual, this function tells how many bytes of memory a user's program listing has

consumed. It does not include the space necessary for the storage of two-character variables or array elements.

STATUS 2 This function is not covered in the manual, but it apparently gives the actual address in memory where a user's program listing ends. If you enter

STATUS 2 - STATUS 1

you will obtain the memory address where user program storage actually begins. This address varies depending on which (if any) memory expansion modules are in use.

STATUS 3 Also not reported in the manual, this function returns the "bottom" of the two-character variables and/or array elements stack. Thus, to find out how much memory you have left when you want to include assignments for two-character variables and array elements, just enter:

STATUS 3 - STATUS 2

STATUS 4 Again is not reported in the manual, but it gives the last complete line number executed in a program. You can thus use this function to perform a lot of programming tricks - such as determining what line caused a jump because of an ON ERROR GOTO directive. You can also use this function to determine what part of a large program called a subroutine, etc. You just insert:

X = STATUS 4

in the first line of the subroutine and use X as desired to accomplish your programming objectives.

STATUS 5-9 Appear to duplicate the STATUS 4 function.

Thanks to readers such as Thomas S. Cox, Brian Peterson, Norlin Rober and others for reporting these latent capabilities.

Error Codes Summary courtesy of Thomas S. Cox

PC-1500 ERROR CODES	
CODE	MEANING
1	Syntax Error Ex: 10 GOTO: 10 5A=1: a=6: 10 NEW
2	No FOR to match NEXT or no GOSUB to match RETURN
4	Out of DATA
5	Double Dimension--Array already Defined
6	Trying to use an Array Variable before defining it
7	Improper Variable type (String or Numeric)
8	Attempt to Dimension array beyond 2
9	Subscript Range Error. Subscript exceeds Dimension.
10	Out of Memory. Not enough memory to create a new variable.
11	Undefined Line Number
12	Incorrect Format in a USING command
13	Out of Memory in Program or Reserve Area
14	FOR/NEXT nested too deeply or buffer space exceeded
15	GOSUB/RETURN nested too deeply or string buffer space exceeded
16	OVERFLOW/UNDERFLOW or HEX exceeds 65535 decimal
17	Mixed String and Numeric data in mathematical expression
18	Inappropriate arguments such as MID\$("jkl") or COS (45,80)
19	Dimension exceeds 255
20	When using Fixed Memory Array Variable (@), parenthesis not used
21	Variable required in expression
22	No memory available when program is loaded
23	Improper number entered for setting TIME
26	Command is not executable in current MODE
27	No program corresponding to given label: no printer
28	Command inserted inside " " or improper INPUT or AREAD
30	Line number exceeds 65539: Error 1 if line number 65280-65539
32	Graphic Cursor between columns 152-155 when executing INPUT
36	Data can't be specified with format given
37	Numeric calculation overflow
38	Division by zero
39	Negative Log or Negative SQR or improper trig. argument
CASSETTE RELATED ERRORS:	
40	Inappropriate Specification for expression
41	SAVE or LOAD specified for ROM area
42	Cassette file too large, can't be CLOADED
43	CLOAD? data format does not match file format
44	CHECKSUM error
PRINTER RELATED ERRORS:	
70	Pen has exceeded -2048<X,Y<2047 or will exceed if executed
71	Paper has or will back up more than 10.24 cm in TEXT mode
72	Inappropriate value for TAB
73	Wrong printer mode (GRAPH or TEXT)
74	Too many commas in LINE or RLIN command
76	Calculated result can't be LPRINTED on one line in TEXT mode
78	Pen being changed or LOW BATTERY state not corrected
79	COLOR signal has not been received
80	Low battery

UPDATES TO THE PC-1500 INSTRUCTION MANUAL

Add the following to page 109:

3. The CLOAD command

When using the CLOAD? command it is necessary that the tape be cued to the carrier tone which precedes the recorded data. The carrier tone consists of a steady, unmodulated signal of approximately 1 KHz. If the tape is not properly cued, no error will be indicated and no data will be presented on the display.

Replace paragraph 5 on page 111-112 with the following:

5. Use of the MERGE command

The MERGE command allows you to store many programs in the computer memory at the same time. For example, let's assume the computer memory contains the following program:

```
10: PRINT "DEPRECIATION ALLOWANCE"  
20: INPUT "Enter method: ";A
```

At this point you remember that you have a similar program portion on tape under the filename "DEP1". You will, of course, want to see if this program has sections useful in the program you are currently constructing. This first step is to find the tape with "DEP1" on it. Cue the tape to the place at which "DEP1" starts. Now type MERGE "DEP1" and press ENTER. The computer will now load "DEP1" into memory in addition to the above program. After "DEP1" is loaded, you might find something in memory similar to this:

```
10: PRINT "DEPRECIATION ALLOWANCE"  
20: INPUT "Enter method: ";A  
10: "DEP1":REM -- Second Module --  
20: PRINT "INTEREST CHARGES"  
30: INPUT "Amount Borrowed: ";B  
:  
(etc.)
```

Note that unlike the CLOAD command, the new program *did not* replace the existing one and that some line numbers have been duplicated. Also note that a "label" was used on the first line of the merged module. This allows *linking* of the modules together (see Linking Merged Modules -- below).

It is important that you review the following information before proceeding with any further editing or programming:

Important Notes

Once a MERGE is performed, no insertions, deletions or changes are allowed to previously existing program lines. Example:

```
10 "A" REM This is existing program  
20 FOR T=1 TO 100  
30 LPRINT T  
40 NEXT T  
(etc.)
```

Before doing a MERGE of the next program, make any necessary changes to this program. Then, MERGE the next program. For example: MERGE "PROG2":

```
10 "B" REM This is MERGED program  
20 INPUT "Enter depreciation: ";D  
30 INPUT "Number of Years: ";Y  
40 ... (etc.)
```

Now you may make changes to the above program since it was the last merged portion. If you need to make further changes to the first program then follow this procedure:

1. CSAVE what you have done to this point; use a new name when saving the tape, i.e., "PROG3" (as an example).
2. CLOAD the program saved in step 1. back into memory.
3. Now make any desired modifications or changes. However, keep this in mind: changes can be made *only* to the *first occurrence* of any duplicated line numbers. Thus, if program line 10 appears in the first program as well as the second, changes will only affect the first occurrence of line 10. If you attempt to edit the second occurrence of line 10, the change will be erroneously reflected in the first program portion only.

Adding Program Lines

Additional lines may be added at the end of the existing programs *only if they have line numbers greater than those previously used*. Note that additional program lines may not necessarily appear at the "bottom" of the listing. Since modules must be linked via labels (see below), this should not be of concern.

Linking Merged Modules (Programs) Together

Since the processor executes your program lines in logical sequence, it will stop when it encounters a break in the sequence in line numbering, i.e., if line numbers 10, 20, 30 are followed by duplicate line numbers in a second module, the processor will cease execution after line 30 in the first module.

To *link* your various modules together, the following techniques are valid: GOTO "B", GOSUB "B", IF ... THEN "B" (where "B" is used for example only; you may use any label except reserved words or letters which appear in row number two on the keyboard, i.e., Q through P).

CREDIT WHERE CREDIT IS DUE

The Electronic Filter Program published on page 7 in Issue 15 of PCN was submitted by: David L. Shuman, 1989 Altamont Place, San Diego, CA 92139. We regret inadvertently failing to acknowledge that fact at the time of publication.

PRINTER'S HELPER

The accompanying listing presents a program of particular use to printers, publishers, art studios, typesetters and anyone who specifies type or sizes artwork and photographs.

The program has two sections which may be selected using the DEFine mode. SHIFT/A calls up the typesetting portion, SHIFT/B the art sizing routine.

Selecting SHIFT/A starts the typesetting routine by asking the user to input point size, leading and line length of the proposed typeset copy. A typical input here might be 10 point type on 11 points leading with a 22 pica line length.

Line 80 of the program asks if the user wishes to "compensate." If the response is affirmative here, the program reduces the number of characters computed to fit on each typeset line by five. This factor, five, is the average number of characters that is typically lost when setting justified or ragged-right copy. If "compensation" is not requested, then the program emulates most manual typesetting routines.

Lines 130 - 200 provide the characters-per-pica (CPP) count for some standard text typefaces: English Times, Helios, Helios Condensed, Helios Light, Clearface, Korina, Oracle and Souvenir.

Be advised that CPP values for typefaces can vary among photo-typesetter manufacturers and typesetters. To customize the program for different CPP values, use the reciprocal of the 10 point value for the variable C. Thus, if 10 point Helios has a CPP of 2.43, set C equal to $1/(2.43 * 0.1)$. Incidentally, the values provided in the listing are for Compugraphic Corporation typefaces.

If the typeface is not one of those provided in the listing, use the MAN (manual) selection and input a CPP value to two decimal places, such as 2.43.

The program outputs the calculated number of typeset lines, copy depth in points and inches.

Line 280 causes the program to recycle without the need to re-specify parameters that are not being altered.

Sizing Artwork

SHIFT/B is a handy routine for sizing artwork. The program asks for original dimensions in inches and tenths. New width, height or percent reduction or enlargement can then be selected.

This program supplied by: Herb Griffiths, 129-1/2 Columbus Ave., Sandusky, OH 44870.


```

10 GRAPHIC
20 "A" CLEAR
30 INPUT "CHRS PER MMS LINE? ";B
40 INPUT "# OF MMS LINES? ";A
50 INPUT "POINT SIZE? ";S
60 INPUT "LEADING? ";L
70 INPUT "LINE LNGTH/PICAS? ";D
80 INPUT "COMPENSATE (Y/N)? ";Z$
90 INPUT "TYPE STYLE? ";Q$
100 IF Z$="Y" LET H=5
110 IF Z$="N" LET H=0
120 IF Q$="MAN" GOTO 290
130 IF Q$="ENG" LET C=1/(S*.03731)
140 IF Q$="HEL" LET C=1/(S*.04115)
150 IF Q$="HELC" LET C=1/(S*.03344)
160 IF Q$="HELL" LET C=1/(S*.03802)
170 IF Q$="CLE" LET C=1/(S*.03322)
180 IF Q$="KOR" LET C=1/(S*.04000)
190 IF Q$="ORA" LET C=1/(S*.03861)
200 IF Q$="SQU" LET C=1/(S*.03571)
210 IF C=0 PRINT "BAD ENTRY":GOTO 90
220 E=(A*B)/(C*D-H)+.9:E=INT E
230 F=L*E:G=F/72
240 PRINT "≡ TYPESET LINES ";E
250 PRINT "COPY DEPTH/PNTS ";F
260 PRINT "COPY DEPTH/INCHES ";USING "####.##";G
270 USING
280 GOTO 30
290 INPUT "CHRS. PER PICA? ";C:GOTO 220
305 "B" CLEAR
310 INPUT "ORIGINAL WIDTH? ";J
320 INPUT "ORIGINAL HEIGHT? ";K
325 L=0:M=0:N=0
330 INPUT "WIDTH WANTED? ";L
335 IF L LET Q=L/J:P=Q*K:O=L:Q=Q*100:GOTO 360
340 INPUT "HEIGHT WANTED? ";M
345 IF M LET Q=M/K:O=Q*J:P=M:Q=Q*100:GOTO 360
350 INPUT "% WANTED? ";N
355 N=N*.01:O=J*N:P=K*N:Q=N*100
360 USING "####.##"
370 PRINT "W=";O;" H=";P
380 PRINT "PERCENT=";Q
390 GOTO 325
400 END

```

USING THE PANASONIC HHC FOR TELECOMPUTING

"Telecomputing" is a term for which Friends Amis, Inc., claims a trademark. Essentially, it stands for a method of communicating with a computer by remote means. Recently, I checked out this method of remote computer communications using the Panasonic HHC equipped with the model RL-P4001 acoustic modem (controlled by a Telecomputing 2 ROM) and the RL-P2001 TV adaptor. While the TV adaptor is not necessary for "Telecomputing," it does make the process somewhat easier during lengthy sessions.

The RL-P4001 modem is controlled from the HHC's keyboard. You activate its send and receive capabilities by selecting the appropriate items from the I/O (input/output) menu which is always available to the user. Next you use the system's primary menu to indicate that you want to use the "Telecomputing 2" system package. This activates software stored in a ROM in the modem unit. This software takes care of all communications protocols.

The Telecomputing 2 software permits you to configure all the necessary parameters for communicating with a remote computer over telephone lines. For instance, you can specify the baud rate, number of data bits, start and stop bits, full or half-duplex modes, etc. Default values are set to those most commonly used by major remote computing services such as The SOURCE. In the tests I made with several local "bulletin board" services, I never had to change the basic configuration parameters. However, I did exercise the software to ascertain that it was easy to make such customized alterations if necessary.

To begin talking with a remote computer site, you just dial the telephone number of the "host" system and wait for the steady audio tone produced by the remote modem. Then you mount the telephone handset into the soft rubber cups of the modem. Next you select the "connect" option from the Telecomputing 2 menu. In a few seconds you see the beginnings of the "log-in" procedure come up on the LCD of the HHC and/or the TV monitor.

I tried the system both with and without the use of a TV monitor. Using a TV definitely makes things easier if you are having a lengthy Telecomputing session. But, without a large-screen display, the Telecomputing 2 system has features that make the use of the HHC practical. For example, you can select a mode called SLOW. When in this mode the HHC displays communications from the remote computer one line at a time. That is, one line of the HHC's display at a time. Whenever you are ready for the next line of information, you just press the NEXT key. That way, the information doesn't go zipping by you at 30 characters per second — unless you want it to. Press another button and the unit flips to the FAST mode whereby information is displayed as fast as it is received. Once you are familiar with a particular system you can use this mode to "log-in" or scan directories, etc.

I did find use of the HHC display alone somewhat tiring after extended use. The short lines, the effects of wordwrap, and constant pressing of the "next line" button soon seems to take the "fun" out of communicating using this method. However, there is no doubt that this is still a very viable method of getting a quick update or inputting data to a system. It also seems to me that familiarity with a particular system — getting to know what responses are expected to various prompts — undoubtedly eases the operators burden when using the HHC in a stand-alone fashion.

Adding the TV monitor to the system makes even extended sessions quite enjoyable. The video display produced by the RL-P2001 TV adaptor produces one of the crispest outputs I have seen on a color TV set. The unit produces a solid green background color and then displays white characters against this background. Sixteen lines with up to 32 characters-per-line (and wordwrap) provide sufficient information on the screen at all times and considerably lessens the mental strain of remembering what you are doing as you "Telecompute." Connection of the TV adaptor to the TV is accomplished readily through one cable. The TV interface is activated by a simple menu selection on the HHC.

The bottom line is that a highly portable system capable of easily communicating with a remote computer (while having its own computing capabilities) is here — and it works. Furthermore, it is easy to use. I like it.

— Nat Wadsworth, Editor

FROM THE WATCH POCKET

Where is Radio Shack's PC-2? As this issue went to press I had yet to find a R.S. store that would admit they had units for sale. Yet several readers say they have units in their hands? Chances are good they will be in the stores nationwide by early July.

There is a new PC program publisher on the scene and the initial releases indicate they are doing a good job. We have several of their new books containing numerous programs out for review. Write to: Micro Text Publications, Inc., One Lincoln Plaza, Suite 27C, New York, NY 10023, or phone (212) 877-8539 and ask for a catalog describing their publications: *Pocket Magic*, *Pocket Computer Primer* and *Science & Engineering Sourcebook*. You can also get cassettes containing all the programs (about 25 per book) at a most reasonable price when you purchase the relevant title.

Prices on PCs and HHCs are beginning to drop rapidly (as I told you would occur in my December, 1981, summary). For instance, Radio Shack's original TRS-80 Pocket Computer, introduced at a price of \$249.00 is now down to \$149.00. This price is likely to drop still further (possibly below \$100) with the introduction of the PC-2. Panasonic's 4K version of its HHC now has a list price of \$418.00, down from \$600.00 just six months ago. Rumors abound that Casio will soon have something similar to the FX-702P in capability at less than \$100.00. And how about that move by Sinclair? Dropped the price of the ZX81 (assembled) from \$149.95 to \$99.95, the 16K RAM add-on was cut in half from \$99.95 to \$49.95, and the ZX81 kit came down from \$99.95 to \$79.95! A fully assembled and tested 16K, highly portable computer can now be attained for less than \$150.00. Alas, the keyboard on that unit drives me bananas.

And where is HP's entry? I continue to hear talk of a powerful, multi-language capability HHC with 32K of memory (and more?) coming from that highly talented firm. End of the summer?

If you want to navigate by the stars in your yacht, be sure to read the article on using your TRS-80 PC for this purpose in the June, 1982, issue of *Microcomputing*, starting on page 112.

Cass Lewart has a nice article in the June issue of *Popular Electronics* that describes how to link up a PC-1 with an external printer.

I have heard that if you have a 2K Panasonic/Quasar HHC, you can expand it to 4K by buying a HM6116 memory IC at a cost of about \$9.00 and plugging it into the back of the unit yourself.

The next issue of *PCN* comes out in September. Hope everyone has a pleasant summer.

— Nat Wadsworth, Editor

SUBSCRIPTION INFORMATION

Subscriptions are sold on a calendar year basis (January – December). There are ten issues per year. You get all issues published for the calendar year to which you subscribe.

☐ 1982 Regular Subscriber (Issues 11 – 20). \$30.00 in U.S. (U.S. \$36.00 to Canada. U.S. \$45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 – 30). \$60.00 in U.S. (U.S. \$72.00 to Canada. U.S. \$90.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA # _____ Expires: _____

Signature: _____



P.O. Box 232, Seymour, CT 06483

AN INDEX FOR THE SHARP PC-1500 INSTRUCTION MANUAL

Howard R. Battan, 35432 44th Avenue South, Auburn, WA 98002, got fed up with the lack of an index in the PC-1500's manual -- so he created his own. Then, he sent in a copy to share with *PCN* readers!

&	98	logic operations	18
abbreviations	40, Apndx	lower case	12
ABS.	66	LPRINT	122
ACS.	65	MERGE	111
AND	98	MID\$.	76
AREAD	133	mode key	15
arrays.	67	multiple statements.	41
ARUN	134	NEW	29
ASC.	72	NEXT	51
ASN	65	NOT	99
ATN	65	ON ERROR GOTO.	84
BEEP.	84	ON GOSUB	83
BEEP OFF.	85	ON GOTO	83
BEEP ON	85	OR	99
CHAIN.	113	PAUSE	34
CHR\$.	73	Pi (π)	64
clear key	13	POINT	94
CLOAD	109	power on/off	63
CLOAD?.	109	PRINT	29
CLS.	88	PRINT=.	110
COLOR	121	printer/cassette int'face	101
concatenation (+).	70	RADIAN.	64
CONT	100	RANDOM	78
COS.	65	range of numbers	62
CSAVE.	109	READ	55
Csize	120	REM	57
CURSOR.	85	RESTORE.	55
DATA	55	RETURN	58
DEG	66	RIGHT\$.	77
DEGREE.	64	RLINE	127
DIM.	67	remote off.	114
DMS	66	remote on	114
editing	19	RND	77
END	43	ROTATE.	120
EXP.	65	RUN	44
FOR-TO-STEP.	51	scientific notation.	60
GCURSOR.	88	SGN	67
GLCURSOR.	124	SIN	65
GOSUB.	58	SORGN	124
GOTO	46	square root.	62
GPRINT	91	STATUS.	79
GRAD	64	STOP	100
GRAPH	118	STR\$.	78
hexadecimal (&).	98	TAB	124
IF-THEN	44	TAN	65
INKEY\$.	74	TEST.	116
INPUT	35	TEXT.	118
INPUT#.	110	THEN	44
INT.	66	TIME	79
LEFT\$.	75	TROFF.	96
LEN	75	TRON	96
LET.	29	two tape recorders	114
LF	121	UNLOCK	100
LINE	125	USING	80
line numbers.	27	VAL	79
LIST	43	variables	23, 135
LLIST	118	variables, string	23, 70
LN	65	WAIT.	54
LOCK	100	@().	135
LOG	65		

POCKET COMPUTER

NEWSLETTER



© Copyright 1982 — Issue 17

September

HEWLETT-PACKARD INTRODUCES PORTABLE COMPUTER

The rumors were true. Hewlett-Packard has been working on a highly portable personal computer. It will be made available to U.S. customers through HP and authorized HP dealers on September 15, 1982.

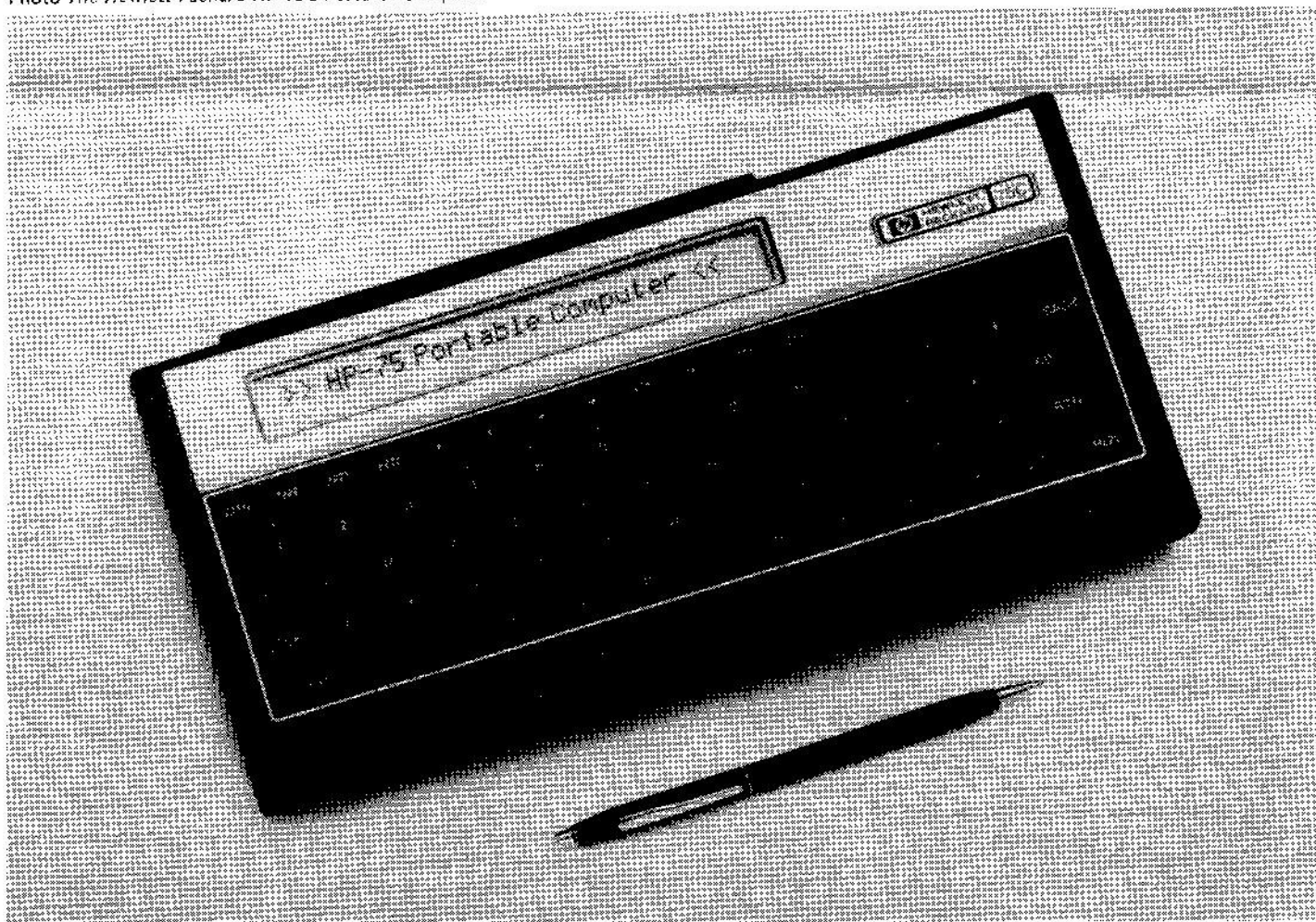
As is to be expected, the new machine appears to be potent in capability. It is small, slightly larger than "pocket-size, measuring 10 x 5 inches and 1-1/4 inch thick. It weighs one pound and ten ounces. For starters, it sports a 48 kilobyte combined BASIC interpreter and operating system in ROM along with an initial 16 kilobytes of user RAM. The RAM may be expanded internally by another 8 kilobytes. Furthermore, additional "software pacs" in 8 and 16 kilobyte ROM modules can be plugged directly into the unit giving it specialized capabilities. The combined memory (ROM and RAM) capability of the unit (all internal) is 120

kilobytes!

Another highly impressive feature of the unit is its built-in "mass storage" capability. A hand-pulled magnetic-card reader is provided as part of the unit. Magnetic strip-cards containing up to 1.3 kilobytes of programs or data are simply pulled through an integrated slot to transfer data or programs from and to the computer. This means a large number of programs can be conveniently carried by the user for instant loading into the machine as needed.

The unit has a typewriter-style keyboard that is large enough to permit touch-typing. Each key may be redefined by the user if desired. Keyboard overlays permit the customizing of the keyboard for special applications. A "hidden" numeric keypad is provided on the keyboard to aid in performing lengthy calculations.

Photo The Hewlett-Packard HP-75C Portable Computer



A single-line, 32 character liquid-crystal display provides a scrolling window on a 96-character line. The dot-matrix display provides for upper and lower case characters.

The rechargeable nickel-cadmium batteries can power the unit for approximately 30 hours of "computing" operations. When the computer is not in actual operation, its non-volatile memory maintains all programs and data. An unusual feature of the HP-75C is its ability, using its built-in real-time clock, to effectively "turn itself on" at a predetermined time, execute a program unattended, and then power back down.

Other sophisticated features include file-handling capabilities. A number of different files, such as program files, data files and even "appointment" files can be created, held in RAM, and directed to interact with one another.

The unit also comes equipped with the HP-IL (Hewlett-Packard Interface Loop) which enables it to communicate directly with a wide range of peripherals and even other computers.

And, a number of peripheral devices will be immediately available for

use with the new machine. These include a video/TV interface allowing a regular TV to be used as a monitor, mass storage devices such as a digital cassette drive, printers, a color graphics-plotter, acoustic modem, a digital multimeter, and other devices. Still further attachments will be available in the future.

The ROM-based operating system provides 147 BASIC commands, statements and functions. It also provides 22 system directives for a total high-level instruction set of 169 directives. With all of the operating system provided in ROM, the user's RAM complement is not diminished by operating system or other ROM plug-in modules.

The list price of the HP-75C is reported to be \$995.00. The optional 8 kilobyte RAM module (giving 24K total) is \$195.00.

For additional information on this unit and related equipment contact your local Hewlett-Packard sales office or mail inquiries to: Inquiries Manager, Hewlett-Packard Company, 1820 Embarcadero Road, Palo Alto, CA 94303.

PROGRAM LIBRARY SYSTEM FOR RADIO SHACK PC-1

How would you like to be able to place a whole series of programs for your Radio Shack PC-1 or Sharp PC-1211 on a single cassette tape. Then, have a system program that would automatically locate the program you want to use, load it into the computer and begin operation? Well, with the package provided here, you can have that sophisticated capability.

Background Information

The PC-1 has a CHAIN command that enables it to automatically load and begin the execution of a program. The simple standard explained here takes this basic capability and expands it into a convenient operating system. All it takes to implement the capability is the addition of a couple of lines of code in each of the programs you place in a "tape library" plus a "library" utility program to tie the whole thing together.

Essentially, the way the system works is to assign a name to the variable "N\$" in each program, which represents the program's identification. Additionally, the label "N" is reserved in each program for use by the "library" utility. The CHAIN command is executed by each program in turn (or the initial library utility program) and proceeds to load in the next program on the tape and auto-execute it beginning at the label "N". This location (label "N") checks to see if the name of the current program (in variable N\$) is the same as that being sought by the user (which the library utility stores as D\$). If not, the program displays its own name (for acknowledgement purposes) and proceeds to CHAIN in the next program in the tape. If it is the program being sought, then it simply proceeds to execute itself!

To keep the tape from running on indefinitely, a "terminating" program is placed on the tape at the end of all the "regular" programs. Its purpose is simply to notify the user that there are no more programs on the tape. The tape can then be rewound if further operations are to be performed. (Perhaps, in the future, PCs will have automatic tape-rewinding features that this program could activate!)

Thus, since each program in the tape library has the few special lines of code mentioned, the system will daisy-chain its way through the library until it finds the program desired by the user. It is a simple but elegant technique, especially if you are preparing a system for novice PC users.

The Library Utility

The library utility is the first program on a tape library. It has two parts. The first part is used to prepare the data file FLIBRY that is used to show the names of all the programs available on a particular tape. The second part uses this data to display the file names to the user when requested.

Just so you have a clear picture of how a tape cassette will be organized, this is what the arrangement will look like:

"PL" The library utility program.

"FLIBRY" The data file containing names of all user programs.

.... A gap on the tape to provide for expansion of the FLIBRY data

file above.

"PL" User program #1

"PL" User program #2

"PL" " " "

"PL" User program #N

"PL" The system "trailer" program to notify users that the tape must be rewound.

Note that all programs on the tape are CSAVED using the same name "PL". They are assigned their individual names internally using the variable N\$. This internal name is what is displayed to the user as each program is processed by the system. Also note that the data section of the tape is identified by the name FLIBRY.

Section "A" of the library utility program is used to specify how many user programs will be on a tape. You are asked to specify the item number (program number), a program name (limited to five characters) and a tape counter (location) value if desired. (This counter value must generally be estimated, based on experience, if programs are of varying length.)

The following procedure should be used to establish a tape using this system:

1. Load the library utility shown in the accompanying listing into your PC.

2. Prepare a fresh tape, zero your tape recorder counter, provide an optional voice message at the start of the tape, if desired.

3. CSAVE the library utility program as the first program on the tape using the tape file name "PL". Thus, the saving command should appear as: CSAVE "PL".

4. Make a list of all the programs that are to be placed on the cassette. The list should contain the tape sequence number, a five-character name, and the tape counter reading. Use this list during the next step:

5. Place the pocket computer in the DEF mode and press SHIFT/A. Respond to the first query from the computer with the total number of programs to be placed on the tape including the special ending program. Then respond appropriately to the queries for the sequence number, name and tape counter value for each program. At the end of each series for a particular program you are given the opportunity to verify that your entry was correct. If not, the program gives you a chance to start the process over. When all data for the programs in the library has been inputted, the utility asks for the current date. Respond in the format YYMMDD. Following this, the utility routine prints out a copy of the library as it will appear to the user. It then asks: "READY TO RECORD FLIBRY?"

6. Check to make sure your tape unit is connected properly and in the record mode. Press ENTER after responding to the above query to have the FLIBRY data saved on the tape.

7. When the FLIBRY data has been saved, advance the tape some on the recorder to leave a gap in case you later decide to add extra programs to the tape. You then have space to update the FLIBRY data without wiping out the first user's program you place on the cassette!

8. Now you sequentially record each user program on the cassette. Remember, each such program must have the special control lines, shown in an accompanying listing. These lines direct the program to check for the desired name and continue the CHAINing operation if it is not the one being sought.

9. The last program on the tape is the special "file terminating" program that reminds the user that the tape needs rewinding. This program is shown in an accompanying listing.

10. You have completed preparation of a tape!

Using the System

Once a tape has been prepared in the standardized fashion explained, operation is simplicity itself.

Just rewind a tape to the beginning and read in the utility program by using CLOAD "PL". Once loaded, with the PC in the DEF mode, press SHIFT/Z. The FLIBRY data will automatically load and show the programs available on the tape. Pick the desired program number, enter, it, and let the system take over. It will automatically find and begin executing the desired program.

The system does provide "a touch of class."

R.J. Saam, 149 Gloucester Road, London, England SW7 4TH, is the author of this program. He would like to hear from PCN readers interested in standards for passing data from one program to another. He also notes that this system of automating program loading has potential for application on other PC systems.

Programs: Directly below is the listing for the library utility program. Note that it has two major sections, starting at lines 790 ("N") and at line 850 ("Z"). At top of right hand column are the lines that should be in each user program in a tape library. Note that line 15 represents the beginning of the regular user program ("A"). A second routine also shown in the upper right hand column illustrates the library terminating program. You can make it as elaborate as you like!

```

790 "N" N$="LIBRY"
792 PRINT N$;" READY.":CLEAR:PRINT ":"
800 "A" D=1:PAUSE "LIBRY LOADER":PAUSE "INPUT #
      PROGS,NAME,LOCN."
802 INPUT "TOTAL # PROGRAMS?";F
803 FOR A=8 TO 3F+5 STEP 3
804 B=A+1:C=B+1
805 INPUT "ITEM #? ";A(A):INPUT "NAME(5CHAR)? ";
      A$(B):INPUT "TAPE COUNT:(.../.../...):";A$(C)
806 INPUT "WAS INPUT OK? Y/N ";B$:IF B$="N" PAUSE
      "RE-DO WHOLE ENTRY":GOTO 804
807 NEXT A
810 PAUSE "INPUT DONE":INPUT "FILE LIBRY DATE? ";
      G$
811 GOSUB "S"
812 "B" INPUT "READY TO RECORD LIBRY? ";B$:IF B$=
      "N" LET B$=" ":GOTO "B"
815 PRINT # "FLIBRY";F
820 PRINT "FLIBRY ";G$:PRINT "RECORDED":PRINT
      "NOW CSAVE YOUR"
821 PRINT "PROGRAMS WITH":PRINT "SAME NAME: PL"
822 PRINT "ALL PROGS NEED":PRINT "ENTRY ,N,"
      PRINT "WHERE N$=YOUR"
823 PRINT "FIVE CHAR LABEL"
824 END
  
```

```

10 "N" N$="BRKVN"
11 IF D$=N$ PRINT D$;" READY.":PRINT " ":CLEAR:
      GOTO "A"
12 PRINT N$:PRINT "LOOKING FOR":PRINT D$:
      PRINT " ":CHAIN "PL","N"
15 "A" CLEAR ... (Normal start of regular program)
  
```

```

10 "N" N$="LASTP"
15 PRINT N$
20 PRINT "END OF LIBRARY"
23 IF D$=N$ GOTO 30
25 PRINT D$;" NOT FOUND"
30 PRINT "IF YOU WANT TO":PRINT "ADD ANY
      PROGRAMS"
35 PRINT "PLEASE OVERWRITE":PRINT "THIS LAST
      PROGAM"
40 PRINT "THEN AMEND LIST:"
45 PRINT "A. REWIND TAPE"
50 PRINT "B. CLOAD <PL> TO OBTAIN LIBRY"
55 PRINT "C. THEN SHFT/A":PRINT "D. REVISE
      FLIBRY"
56 PRINT " ":PRINT " ":PRINT " "
57 END
  
```

```

850 "Z" D=2:PRINT "LIST PROGRAM LIBRARY"
860 INPUT "IS TAPE READY? Y/N";B$:IF B$="N"
      GOTO 860
865 INPUT # "FLIBRY";F
870 "S" PRINT "***A PERSONALIZED***":PRINT
      "FLIBRY: ";G$
871 USING " ##
872 PRINT F;" ENTRIES":PRINT " ":PRINT " # NAME
      FROM/TO
874 PRINT "== ==== ===/===
880 FOR A=8 TO 3F+5 STEP 3
882 B=A+1:C=B+1
884 PRINT A(A);" ";A$(B);" ";A$(C)
886 NEXT A
887 PRINT " ":PRINT " ":PRINT " "
888 IF D=1 RETURN
890 INPUT "WHICH PROGRAM NO.? ";A
891 IF A > F PRINT A;" GREATER THAN ";F;"#ITEMS
      ON FILE":GOTO 890
896 A=3A+5:B=A+1:C=A+2:D$=A$(B):PRINT D$;" # ";
      A(A)
897 PRINT "TAPE REPLAY ";A$(C):INPUT "TAPE
      READY? ";B$:IF B$="N" GOTO 897
898 CHAIN "PL","N"
  
```

TAX ASSESSMENT PROGRAM

This program can help you fill out your annual (local) tax assessment forms. It has the following features:

Maintenance of equipment purchase prices by year for the current year and eight prior years.

Maintenance of a total of purchases for the ninth prior and all earlier years.

Ability to update the values for any year at any time.

Maintenance of depreciated value factors for each year stored.

Ability to print a report showing the original purchase total and factors at any time.

Automatic shifting of values when a new year is entered.

The program is designed for the Radio Shack PC-1 with the printer, but it can be modified to run without the printer, if desired.

The program has four different functions. To select the appropriate function, put the computer into the DEF mode and press the shift key and the letter shown next to the function:

S - Set up current year and depreciation percents

A - Accumulate purchase prices of equipment

F - Final totals for the current year

D - Display stored values

Set Up

Your must run this first, before entering any values. This routine will prompt you for information, as described below. The program checks the values you enter for reasonability, and will print the message **ERROR** if the information you enter is incorrect. Press **ENTER** to return to the prompting message.

Here are the prompting messages for the set-up routine.:

CURRENT YEAR?

Enter the four-digit year. This should be the most recent year you want to process. In the sample reports 1977 was used. The program will automatically generate the eight preceding years (1976, 1975, . . . , 1969). To ensure that you have entered a four-digit number, the program checks to make sure the year is not less than 1950.

At this point, the program prompts you for the depreciated value factors for the current year, the eight prior years, and the ninth and prior years. Note: these factors are multiplied by the original purchase prices to get the depreciated values. For each one the program prints

YYYY PERCENT

on the printer. YYYY is the year, 1977 for example. The program then prompts

PERCENT?

Type the depreciated value factor and press **ENTER**. For example, if the factor is 90%, type 90 (NOT .90).

The program will ask for the years in descending order; for example, 1977, 1976, . . . , 1968. The final year is really going to be that year and all prior ones. In the samples below, the earliest year is 1969. 1968 and all preceding years are lumped together.

These factors remain constant regardless of the shifting of the years (unless, of course, you change them). In other words, the factors are really for the current year, whatever it may be, and the preceding years.

Accumulating Purchase Prices

To use this feature, press the **SHIFT/A**. The printer is not necessary for this processing. The program prompts you with

YEAR?

Type the four-digit year. The program checks to make sure it is not less than 1950. Processing continues depending on the value you entered:

If the year is less than or equal to the highest year stored, the program will prompt you for amounts with

AMOUNT?

Type in the original purchase price (dollars and cents). This will be added to the total for the specified year. The program then prompts you for the next amount. Continue entering information until you have no more for that year. Then press **ENTER** without entering a value and the program will stop.

Example Input and Output for the Tax Assessment Program

INPUT	
1977	5600.00
	95.00%
1976	5500.00
	90.00%
1975	5400.00
	80.00%
1974	5300.00
	70.00%
1973	5200.00
	60.00%
1972	5100.00
	50.00%
1971	5000.00
	40.00%
1970	8147.23
	30.00%
1969	12121.31
	25.00%
1968	15273.94
	25.00%

OUTPUT	
	1977
COST	5600.00
DEPR	5320.00
	1976
COST	5500.00
DEPR	4950.00
	1975
COST	5400.00
DEPR	4320.00
	1974
COST	5300.00
DEPR	3710.00
	1973
COST	5200.00
DEPR	3120.00
	1972
COST	5100.00
DEPR	2550.00
	1971
COST	5000.00
DEPR	2000.00
	1970
COST	8147.23
DEPR	2444.17
	1969
COST	12121.31
DEPR	3030.33
	BEFORE 1969
COST	15273.94
DEPR	3818.49
	TOTAL
COST	72642.48
DEPR	35262.99

If the year you entered is greater than the highest year stored, the program will ask

NEW YEAR (Y OR N)?

It wants to know whether you wish to start a new year or have just made an error. If you are starting a new year, type Y and press ENTER. The program will take about 12 to 15 seconds to shift its stored values and generate a new series of years. Then it will prompt you for amounts, as described above.

If you have made a mistake, type N and the program will prompt you for the year again.

NOTE: If you have made an error entering the year and the year is not greater than the highest year, the program won't know an error has occurred. It will proceed to accumulate amounts either in the specified year or in the total for all years prior to the earliest year. For your own application, you might want to change the program to test for some year other than 1950 as the earliest allowable year to help prevent this type of error. At any rate, be careful when entering the year!

Producing Totals for a Year

To use this feature, press SHIFT/F. The printer must be on. The program will produce a report. The original cost of equipment purchased during that year and the depreciated value are printed for each year. Years are listed in descending order, starting with the highest year and continuing through the eighth prior year (1977 through 1969 in the sample). Then, the sum of all prior years is printed (BEFORE 1969 in the sample). If there were no purchases during some year, that year will not be printed. At the end of the report, the word TOTAL is printed, followed by the totals for all years.

Note: The running of this section of the program doesn't initialize it for a new year. Initialization only happens when you enter a new year using

the accumulation portion of the program!

If you don't have a printer, you will want to change the program so that the values are displayed in a meaningful fashion. I recommend deleting the line of dashes, which separates the years, and adding the year itself to each line displayed.

Displaying Stored Values

To use this feature, turn the printer on, and press SHIFT/D. The program will list each year, the accumulated total, and the stored percent. This lets you see what the totals are and which percentages apply to each year.

Thanks for submitting this program go to: *Guerri F. Stevens, %General Business Systems, Inc., 1420 Main Street (Suite 130), Glastonbury, CT 06033.*

Table Variable Assignments In Tax Assessment Program

A	General — purpose
B	General — purpose (used for entry of percents)
C\$	Response to "new year" question
D	Depreciated total
I	Subscript
A(21) — A(30)	Years: A(21) is the latest; A(30) is the earliest, and represents that year and all prior.
A(31) — A(40)	Depreciation factors
A(41) — A(50)	Accumulated purchase totals
	Note that the values in A(31) through A(40) and in A(41) through A(50) correspond to the years in A(21) through A(30).

Program Local Tax Assessment

```

10:"S"FOR I=31
  TO 40:A(I)=1
  :A(I+10)=0:
  NEXT I
20: INPUT "CURRE
  NT YEAR? "A
30: IF A<1950
  GOSUB 90:
  GOTO 20
40:FOR I=21TO 3
  O:A(I)=A:A=A
  -1
50:PRINT USING
  "#####"A(I)
  : " PERCENT"
60: INPUT "PERCE
  NT? "B
70: IF (B>100)+(
  B<0)GOSUB 90
  :GOTO 60
80:A(I+10)=B/10
  O:NEXT I:END
90:PRINT "ERROR
  ":RETURN
100:"A"INPUT "YE
  AR? "A
110: IF A<1950
  GOSUB 90:
  GOTO 100
120: IF A>A(21)
  GOSUB 170
130: IF A>A(21)
  GOTO 100
140: I=21:GOSUB 2
      50:A=0
      150: INPUT "AMOUN
      T? "A:A(I)=
      A(I)+A:GOTO
      150
      160:END
      170:C$=" ":INPUT
      "NEW YEAR (Y
      OR N)? "C$
      180: IF C$="Y"
      GOSUB 210:
      GOTO 180
      190: IF C$="N"
      RETURN
      200:GOSUB 90:
      RETURN
      210:A(50)=A(50)+
      A(49):A(30)=
      A(29)
      220:FOR I=28TO 2
      1STEP -1:A(I
      +1)=A(I):A(I
      +21)=A(I+20)
      :NEXT I
      230:A(21)=A(22)+
      1:A(41)=0:IF
      A=A(21)LET C
      $="N"
      240:RETURN
      250: IF A<A(30)
      LET I=50:
      RETURN
      260: IF A=A(I)LET
      I=I+20:
      RETURN
      270: I=I+1:GOTO 2
      60
      280: "F"B=0:D=0:
      FOR I=21TO 2
      9: IF A(I+20)
      =0GOTO 330
      290:A=A(I+20):B=
      B+A:GOSUB 44
      O:PRINT "
      ":USING "#
      #####"A(I)
      300:PRINT "COST"
      :USING "####
      #####.##"A
      310:A=INT ((A(I+
      10)*A(I+20)+
      .005)*100)/1
      00:D=D+A
      320:PRINT "DEPR"
      :A
      330:NEXT I
      340: IF A(50)=0
      GOTO 390
      350: B=B+A(50):
      GOSUB 440:
      PRINT USING
      "#####" B
      EFORE"A(29)
      360:PRINT "COST"
      :USING "####
      #####.##"A(5
      0)
      370:A=INT ((A(40
      )*A(50)+.005
      )*(100)/100:D
      =D+A
      380:PRINT "DEPR"
      :A
      390:GOSUB 440:
      PRINT "
      TOTAL":PRINT
      "COST":B
      400:PRINT "DEPR"
      :D:GOSUB 440
      :PRINT " ":
      PRINT " ":
      PRINT " ":
      END
      410:"D"FOR I=21
      TO 30:A=A(I+
      20):PRINT
      USING "#####
      #####.##"
      :A
      420:A=A(I+10)*10
      O:PRINT "
      ":USING "#
      #####.##"A: "%
      :GOSUB 440
      430:NEXT I:PRINT
      " ":PRINT "
      ":END
      440:PRINT "-----
      -----"
      :RETURN
  
```

DECIMAL/BASES 2 - 16 CONVERSION PROGRAM

Thomas S. Cox, %Platt Saco Lowell, Drawer 2327, Greenville, SC 29602, sent in this program. You can use it to convert between decimal numbers and those in any base between 2 and 16 or vice versa.

Several versions of the program are shown. One for the Sharp PC-1500/Radio Shack PC-2. A second for the Casio FX-702P.

Program Number Conversion for the PC-1500/PC-2

```

10: "BC" CLEAR .          WAIT 0: GOTO 10
    WAIT 50: PRINT
    "Base Conversion"
12: A$="0123456789
    ABCDEF"
15: DIM A$(0)*50,
    A1$(0)*50
20: INPUT "B(N)-10
    or B10-BN for
    2? "; L
25: IF (L=1)+(L=2)
    <>1CLS: GOTO 2
    0
30: INPUT "BASE (N
    =?) (2-16) "; N
40: IF (N<2OR N>16
    )PRINT "BASE 0
    UT OF RANGE ";
    CLS: GOTO 30
50: CLS
60: ON LGOTO 100, 2
    00
100: WAIT 50: PRINT
    "BASE "; N; " TO
    BASE 10": A$(0)
    0)="": A1$(0)="
    "
105: WAIT 0: PRINT "
    # IN BASE"; N; "
    ": INPUT A$(0)
    )
110: CLS: A=LEN A$(
    0): B=0
115: FOR I=0 TO A-1:
    C$=MID$(A$(0),
    A-I, 1): G=ASC
    C$: IF (G=48)+
    (G=57)=2LET G
    =G-48: GOTO 130
120: IF (G=65)+(G<
    =70)=2LET G=G-
    55: GOTO 130
125: BEEP 1: CLS:
    WAIT 20: PRINT
    "INVALID ENTRY
    : TRY AGAIN":
    CLS: WAIT 0:
    GOTO 100
130: IF G=NWAIT 50
    : PRINT "DIGIT
    >= BASE": CLS:
    
```

```

140: B=B+G*N^I: NEXT
    I: WAIT: PRINT
    B: GOTO 100
200: WAIT 50: PRINT
    "Base 10 to Ba
    se "; N: A$(0)=
    "": A1$(0)="": B
    $=""
205: WAIT 0: INPUT "
    DECIMAL No.= "
    : D=E=0: IF D>9.
    999E10WAIT 50:
    PRINT "TOO BIG
    ": CLS: WAIT 0
    : GOTO 200
215: IF (N=2)+(D>2^
    26-1)=2WAIT 50
    : PRINT "TOO LA
    RGE FOR DISPLA
    Y": CLS: WAIT 0
    : GOTO 200
220: WAIT 0: F=0: CLS
230: FOR I=1 TO 26
235: B=(E/N)-INT (E
    /N): C=INT ((B*
    N)+.5): IF INT
    (E/N)=0LET F=1
240: E=INT (E/N)
250: B$=MID$(A$, C+
    1, 1): A$(0)=A$
    $(0)+B$
260: IF F=1GOTO 300
270: NEXT I
280: WAIT 100: PRINT
    "TOO LARGE FOR
    DISPLAY": CLS
    : GOTO 200
300: WAIT 0: FOR I=
    LEN A$(0) TO 1
    STEP -1: B$=
    MID$(A$(0), I
    , 1)
310: A1$(0)=A1$(0)+
    B$: NEXT I: WAIT
    : CLS
320: PRINT A1$(0):
    GOTO 200
    
```

1090

Program Number Conversion for the Casio FX-702P

```

VAR: 46 PRG: 1520
P0: 513 STEPS
5 WAIT 20
10 PRT "BASE CONVE
    RSION"
15 PRT "BASE(10) T
    O BASE(N)"
17 VAC: INP "BASE
    (N)", N
20 INP "DECIMAL NU
    MBER", D: E=0
25 F=0: PRT "CALCUL
    ATING":
30 FOR I=0 TO 19 S
    TEP 1
40 B=FRAC (E/N)
50 C=INT ((B*N)+.5
    )
55 IF INT (E/N)=0:
    F=1
60 E=INT (E/N)
70 GSB 200
75 IF D<(N-1): PRT
    "DIGIT > BASE":
    GOTO 20
80 B=B+D*N^I
90 NEXT I
100 WAIT 0
110 PRT $: " IN B("
    N): "="
115 PRT B: " B10": ST
    OP: GOTO 20
200 IF C$="0": D=0: R
    ET
201 IF C$="1": D=1: R
    ET
202 IF C$="2": D=2: R
    ET
203 IF C$="3": D=3: R
    ET
204 IF C$="4": D=4: R
    ET
205 IF C$="5": D=5: R
    ET
206 IF C$="6": D=6: R
    ET
207 IF C$="7": D=7: R
    ET
208 IF C$="8": D=8: R
    ET
209 IF C$="9": D=9: R
    ET
210 IF C$="A": D=10:
    RET
211 IF C$="B": D=11:
    RET
212 IF C$="C": D=12:
    RET
213 IF C$="D": D=13:
    RET
214 IF C$="E": D=14:
    RET
215 IF C$="F": D=15:
    RET
216 PRT "INVALID IN
    PUT": END
P1: 660 STEPS
5 WAIT 20
10 PRT "BASE CONVE
    RSION"
15 PRT "BASE(10) T
    O BASE(N)"
17 VAC: INP "BASE
    (N)", N
20 INP "DECIMAL NU
    MBER", D: E=0
25 F=0: PRT "CALCUL
    ATING":
30 FOR I=0 TO 19 S
    TEP 1
40 B=FRAC (E/N)
50 C=INT ((B*N)+.5
    )
55 IF INT (E/N)=0:
    F=1
60 E=INT (E/N)
70 GSB 300
80 IF F=1 THEN 100
90 NEXT I
100 PRT " ": PRT "TH
    E ANSWER IS: "
110 FOR I=0 TO 19: I
    F A$(I)="" THEN
    130
120 PRT A$(I):
130 NEXT I
135 STOP
140 GOTO 17
300 IF C=0: A$(19-I)
    ="0": RET
301 IF C=1: A$(19-I)
    ="1": RET
302 IF C=2: A$(19-I)
    ="2": RET
303 IF C=3: A$(19-I)
    ="3": RET
304 IF C=4: A$(19-I)
    ="4": RET
305 IF C=5: A$(19-I)
    ="5": RET
306 IF C=6: A$(19-I)
    ="6": RET
307 IF C=7: A$(19-I)
    ="7": RET
308 IF C=8: A$(19-I)
    ="8": RET
309 IF C=9: A$(19-I)
    ="9": RET
310 IF C=10: A$(19-I)
    )="A": RET
311 IF C=11: A$(19-I)
    )="B": RET
312 IF C=12: A$(19-I)
    )="C": RET
313 IF C=13: A$(19-I)
    )="D": RET
314 IF C=14: A$(19-I)
    )="E": RET
315 IF C=15: A$(19-I)
    )="F": RET
318 PRT "INVALID IN
    PUT": END
    
```


AMERICAN ROULETTE

Gary Heidbrink provided this version of a classic Las Vegas game. The listing shown will run on both PC-1/PC-1211 and PC-2/PC-1500 units.

The game is modeled on the American version of Roulette which has the numbers zero through 36 and a double zero (00). You are allowed to bet as follows.

Double zero: A bet that the marble will fall in the 00 slot.

Single number bet: Enter the number desired (0 - 36) followed by the amount of the wager.

Red: The numbers 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34 and 36 are red.

Black: The numbers 2, 4, 6, 8, 10, 11, 13, 15, 17, 20, 22, 24, 26, 28, 29, 31, 33 and 35 are this color.

Even: Any even number (excluding zero or 00).

Odd: Any odd number (not 0 or 00).

1 - 18: Any number within this range.

19 - 36: Any number within this range.

1 - 12: Inclusive!

13 - 24: Inclusive!

25 - 36: Inclusive!

Column 1 - 34: A bet that the number will be one of these - 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31 or 34.

Column 2 - 35: Any one of these - 2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32 or 35 is a winner.

Column 3 - 36: If a 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33 or 36 comes up then you win.

Bets continue to ride until they are removed by betting zero or you change the value of the bet. If you want to clear all bets, quit the game and start over.

The odds: Betting Even, Odd, Red, Black, 1-18 or 19-36 pays even money. Betting 1-12, 13-24, 25-36 or any Column pays 2:1. Single numbers (including 0 and 00) pay 35:1.

Good luck.

Program American Roulette

```

5  "A" B=3254:INPUT "RND#=";B
10  FOR E=6 TO 57:A(E)=0:NEXT E:PRINT "  AMERICAN
    ROULETTE"
15  INPUT "BET OR QUIT? (B/Q):";D$:IF D$="Q"
    GOTO 295
20  INPUT "CHANGING BETS? (Y/N):";D$:IF D$="N"
    GOTO 95
25  INPUT "DOUBLE 00:";H
30  INPUT "ENTER SINGLE BET #:";E:F=E+21:INPUT
    "AMOUNT BET:";A(F):GOTO 30
35  INPUT "RED:";I
40  INPUT "BLACK:";J
45  INPUT "EVEN:";K
50  INPUT "ODD:";L
55  INPUT "1-18:";M
60  INPUT "19-36:";N
65  INPUT "1-12:";O           (The letter O!)
70  INPUT "13-24:";P
75  INPUT "25-36:";Q
80  INPUT "COLUMN 1-34:";R
85  INPUT "COLUMN 2-35:";S

```

```

90  INPUT "COLUMN 3-36:";T
95  PAUSE "WHEEL IS SPUN"
100 E=E+1:B=439147*B+E:B=23*B-INT(B*.23)*100:A=INT
    (B*.38)
105 IF A=37 LET F=H*36:GOTO 270
110 IF A=0 LET F=U*36:GOTO 270
115 FOR E=1 TO 36:IF A=E LET F=A(E+21)*36
120 NEXT E
125 IF A=2 GOTO 220
130 IF A=4 GOTO 220
135 IF A=6 GOTO 220
140 IF A=8 GOTO 220
145 IF A=10 GOTO 220
150 IF A=11 GOTO 220
155 IF A=13 GOTO 220
160 IF A=15 GOTO 220
165 IF A=17 GOTO 220
170 IF A=20 GOTO 220
175 IF A=22 GOTO 220
180 IF A=24 GOTO 220
185 IF A=26 GOTO 220
190 IF A=28 GOTO 220
195 IF A=29 GOTO 220
200 IF A=31 GOTO 220
205 IF A=33 GOTO 220
210 IF A=35 GOTO 220
215 F=F+I*2:GOTO 225
220 F=F+J*2
225 IF A/2=INT(A/2) LET F=F+K*2:GOTO 235
230 F=F+L*2
235 IF A > 24 LET F=F+Q*3+N*2:GOTO 255
240 IF A > 18 LET F=F+P*3+N*2:GOTO 255
245 IF A > 12 LET F=F+P*3+M*2:GOTO 255
250 F=F+O*3+M*2
255 IF A/3=INT(A/3) LET F=F+T*3:GOTO 270
260 IF (A+1)/3=INT((A+1)/3) LET F=F+S*3:GOTO 270
265 F=F+R*3
270 FOR E=8 TO 57:F=F-A(E):NEXT E:G=G+F
275 BEEP 1:IF A=37 PRINT "BALL LANDS ON: 00":
    GOTO 285
280 PRINT "BALL LANDS ON: ";A
285 IF F < 0 LET F=-F:PRINT "YOU LOSE $";F:
    GOTO 15
290 PRINT "YOU WIN $";F:GOTO 15
295 IF G < 0 LET G=-G:PRINT "YOU LOST $";G:GOTO 10
300 PRINT "YOU WON $";S:GOTO 10

```

PROTECTION FOR YOUR PC!

Ever wish you could safely transport your entire PC system — complete with printer, cassette drive, and a few extra rolls of paper? Now you can. Shale Diversified Enterprises, 108 Ferris Avenue, Chardon, OH 44024, provides several styles for both PC-1/PC-1211 and PC-2/PC-1500 models. For details and information on pricing write directly to the firm and ask about their "PC Attache Cases."

Photo PC Attache Case Protects Entire System

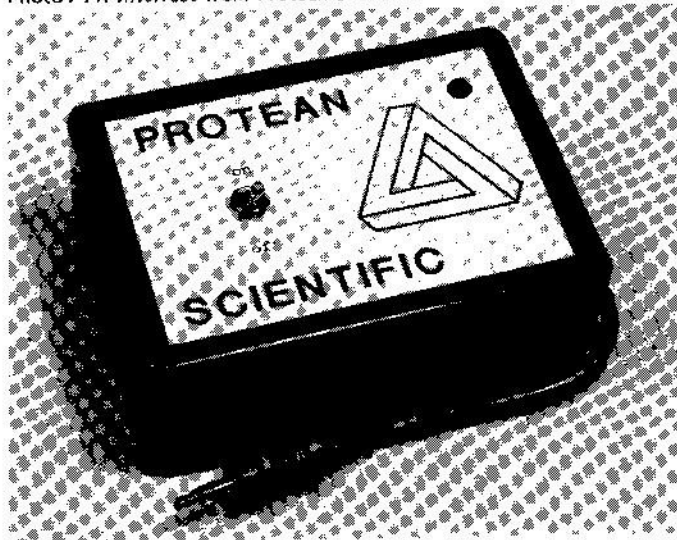


INTERFACE TRANSFERS DATA FROM PC TO DESKTOP UNIT

A new device dubbed the PTR Interface from Protean Scientific, Route 13, Lincoln, NE 68527, claims to provide the ability to transfer alpha and numeric data from a Radio Shack PC-1 to a TRS-80 Model I or III desktop unit.

The interface connects between the desktop unit and its cassette recorder and allows tapes created by the PC to be processed. You can transfer complete files or a specified number of memories. Checksum error detection is included. The unit is provided with software which includes an assembly language routine to facilitate the transfer process. Write directly to the company for additional information and pricing.

Photo PTR Interface from Protean Scientific



FROM THE WATCH POCKET

Sharp Electronics will begin releasing ROM Software Libraries during the fourth quarter of 1982. These modules, which will plug directly into the back of the PC-1500, each contain a whole series of routines specifically designed to solve problems commonly encountered in particular fields. Modules covering five general areas have been announced to date: Mathematics, Electrical Engineering, Statistics, Finance and Graphics. To give you an idea of the scope of each module, the Electrical Engineering Application Module is reported to be able to perform the following types of operations: Complex arithmetic with polar I/O capability. Fourier analysis. Network analysis. Bode/Nyquist calculations. Complex roots of a polynomial. Active filter design. Passive filter design. Phase-locked loop analysis. Series to parallel conversions. Smith Chart calculations. Signal detection determinations. Polynomial multiplication. Decibels, nepers, power, voltage and current ratio conversions. Reactance chart calculations and complex matrix arithmetic. Want more information on price and availability? Write Atlantic Northeast Marketing, P.O. Box 921, Marblehead, MA 01945 or phone (617) 639-0285.

Panasonic has announced further reductions in the list price of its HHC and accessories. A 4K RL-H1400 is down to \$380.00. They now have an 80-character, 4-color printer/plotter unit, an acoustic modem with cassette interface and other new peripherals available. For further information write to them at One Panasonic Way, Secaucus, NJ 07094.

Sanyo has been displaying a hand-held computer at recent trade shows. The model PHC8000 is said to contain a 24K operating system in ROM and have 4K of user RAM in the standard unit. It can mate with the PHC8010 to become a terminal, expand memory, etc. Look for U.S. deliveries later this year.

Now that Hewlett-Packard has come out of the woods, where is their old rival Texas Instruments? — Nat Wadsworth, Editor

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January — December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

- ☐ 1982 Regular Subscriber (Issues 11 — 20). \$30.00 in U.S. (U.S. \$36.00 to Canada. U.S. \$45.00 elsewhere.)
- ☐ 1982/83 Subscriber (Issues 11 — 30). \$60.00 in U.S. (U.S. \$72.00 to Canada. U.S. \$90.00 elsewhere.)
- ☐ 1983 Regular Subscriber (Issues 21 — 30). \$36.00 in U.S. (U.S. \$42.00 to Canada. U.S. \$50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

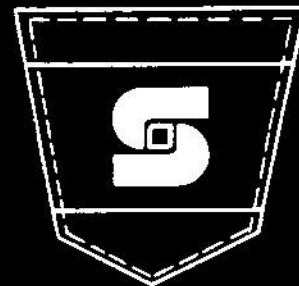
MC/VISA #: _____ Expires: _____

Signature: _____



P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER



© Copyright 1982 — Issue 18

October

NEW CASIO FX-700P CRACKS \$100.00 PC PRICE BARRIER

Just one year after Casio introduced the first "under \$200.00" PC, it has scored another price breakthrough. The new PC, designated the FX-700P is slated for U.S. deliveries in October. It is priced at \$99.95.

The unit appears to be a slightly smaller-sized version of the 702P, with some features of the earlier unit dropped.

One highly noticeable change is in the keyboard layout. It is now in the familiar QWERTY arrangement, with a numeric keypad along the right side. Another difference is a reduction in the size of the LCD, down to just 12 characters plus a battery of annunciators.

The unit measures 3/8 inches thick by 6-1/2 inches wide and 2-3/4 inches deep. It may also set a new weight record coming in at a mere 4.2 ounces when equipped with two CR2032 lithium batteries. Battery life is said to be about 300 operational hours.

User memory available in the unit is slightly less than the 702P. The FX-700P sports 1,568 program steps or 222 memories. Memory may be partitioned in steps to allocate variables storage space and program storage. Up to 10 programs may be identified and accessed within those memory boundaries. There is also room provided for a 30-character string element.

The BASIC language operating system includes the statements and commands: INPUT, PRINT, GOTO, FOR-NEXT, IF-THEN, GOSUB, RETURN, STOP, END, RUN, LIST, LIST A, MODE, SET, VAC, CLEAR, CLEAR A, DEFN, SAVE, SAVE A, LOAD, LOAD A,

PUT, GET and VER. Program functions include: KEY, CSR, LEN, MID and VAL. Program stacks permit up to 8 levels of subroutines, 4 levels of FOR-NEXT loops, 6 levels of numerical values and 12 levels of calculation elements.

The unit also sports a number of mathematical functions such as: choice of degree/radian/gradient angular units, log and exponentials, square root, powers, integer, absolute value, pi, random numbers and others.

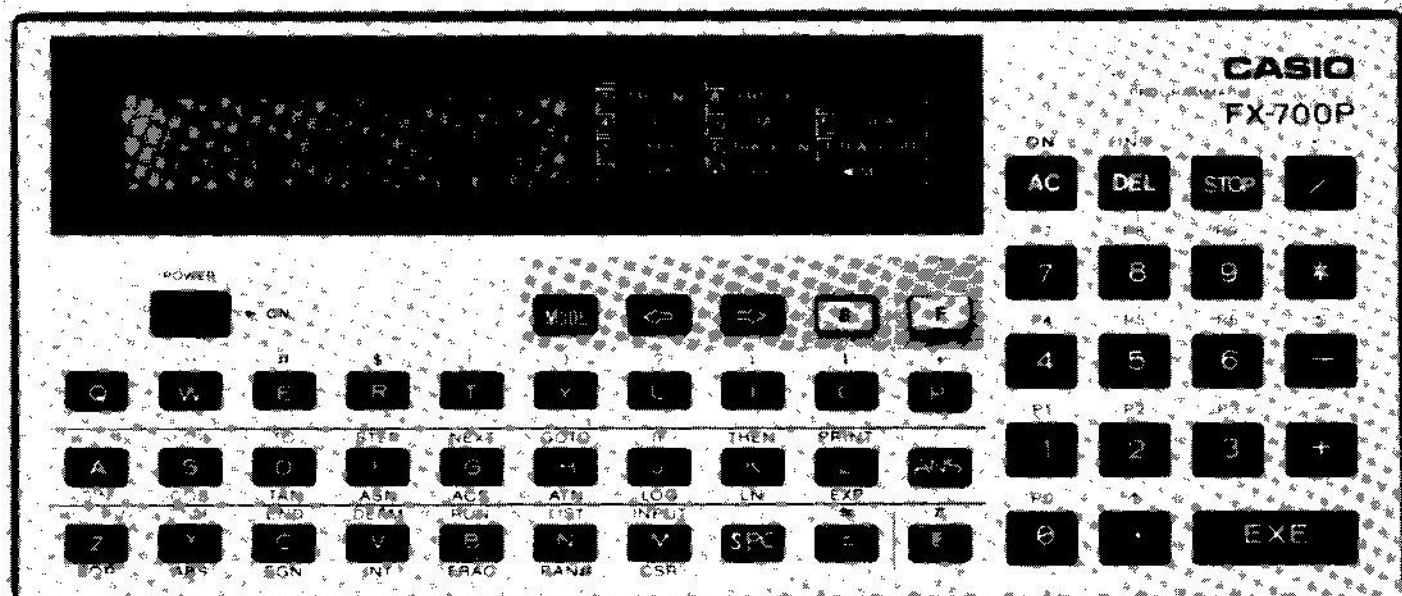
The liquid-crystal display with 12 character positions can display the mantissa of a number to 10 positions (including sign) or display an 8-digit mantissa plus a 2-digit exponent. The display also is capable of displaying both upper- and lower-case alphabetical characters plus a number of symbols. Special annunciators along the top of the display provide additional information about the status of the PC.

The keyboard achieves versatility by providing as many as three functions to a key. Special shift keys determine which function a key will serve when it is pressed. BASIC statement keywords have been assigned to keys to permit rapid entry of programs. The 54-key unit reportedly uses the standard ASCII character coding convention.

An optional tape recorder interface, called the FA-3 is reported as being available soon, at low cost.

A program library offering a selection of typical application programs is supplied with the unit. For more information contact: Casio, Consumer Products Division, 15 Gardner Road, Fairfield, NJ 07006.

Photo Casio FX-700P Pocket Computer Programs in BASIC.



THE HP-75: A FORMIDABLE COMPANION

I hardly know where to start telling you about Hewlett-Packard's entry into the true portable computer market. Let me simply preface my review by saying that this machine will very likely set a standard. One against which other makers of PCs will find themselves compared.

It is obvious that this machine was designed by a team that did their homework. They have managed to combine practically every desirable feature that experience has taught pocket computer users they need, into the HP-75. What is astounding is that they managed to do it on their first try!

Size

Hewlett-Packard does not call the HP-75 a *pocket* computer, they bill it as a highly *portable* computer. They are right. At 5 by 10 inches and 1-1/4 inch thick it is just a little too big to go in a pocket. But, at just a pound and a half, it feels and carries like a slim-line book. When snuggled safely in its leather carrying case (supplied as standard with the unit) you might think someone was toting an electronic multimeter, instead of a fully-functional computer! Oh yes, there is also room in the case to store a good supply of the little magnetic strip-cards which can hold programs or data. The card reader is built right in to the HP-75. This arrangement sure beats trying to stuff a pocket computer, a portable tape recorder, half a dozen cassettes and a bunch of connecting cords into a briefcase just so one can quickly exchange programs or data. In other words, the HP-75 adds a significant measure of *practicality* to the concept of portable computing — despite the fact that the unit doesn't actually fit into a pocket!

The size of the unit is just large enough so that the keyboard can be comfortably arranged for touch-typing. There are a few features of this keyboard that I personally consider drawbacks. One is that the keys are shaped like and feel like "calculator" buttons. I hope that in future models the designers can make the "feel" of this keyboard just a little better for touch-typists.

A second feature that I find a little disconcerting is the manner in which the shift key operates. You must hold it depressed while you press the key that is to be shifted. While this is exactly as one uses a typewriter, I guess I have been spoiled by the action of the Sharp/Radio Shack PCs. In those units, one press of the shift key means the next regular key pressed will be shifted — so you don't have to hold the shift key down. You hunt-and-peckers will have to get used to using both hands on a HP-75!

Taken in over-all context, however, these negative aspects of the keyboard's operation are minor. The fact is, you *can* touch-type on this keyboard with considerable speed — something you really have not been able to do on other PCs to date.

Features — Features — Features

There are so many features on the HP-75 that it will not be possible to cover them all (by a long shot) in this review. But, I will try to give you a feel for the type of capabilities that have been provided on this machine.

To start with, let me point out that the unit comes standardly equipped with 48 kilobytes worth of operating system in ROM plus 16K of user RAM. Now, 48K of ROM buys you a lot of operating system — and HP capitalized on every bit of it.

(One quick aside here. That 48K operating system in ROM *does* eat up some of the user's RAM — about 2K worth, typically. This is used for pointers, counters, I/O buffers, etc. This leaves about 14K for user programs in a standard system. You can add an optional 8K RAM module to form a 24K system, of which about 22K is available for user programs and data. I mention this because an HP press release I saw could leave the impression that the operating system does not require any user RAM.)

The Appointment Mode

To begin with, the system incorporates an extremely sophisticated "appointment reminder" capability. How sophisticated? Well, you can schedule appointments under the current one year period. Or, if that is not enough, you can use the extended 10,000 year (you should be so lucky!) calendar.

Pressing an 'APPT' key brings up a template on the LCD (liquid-crystal display) screen. You key in the date and time of your appointment using this template. If you don't know certain parameters (such as the day of the week on which a certain date falls), the operating system figures that information out for you. You also tell the system what type of audible alarm you want provided (there are ten possibilities) and whether the appointment is singular or repetitive. (If repetitive, you specify how often the appointment repeats.) You can also enter a message that describes the nature of the appointment as a reminder to yourself.

Once appointments have been keyed in, they are automatically announced *regardless of the state of the machine!* Yes, even if the machine is shut off. It will turn itself on, issue an audible alert (if you requested an audible option), turn on a visual annunciator on the LCD screen, and even proceed to execute a program, if that was desired!

The system operates such that it holds and alerts you to any appointments that are past-due every time you use the unit. It will do this until you acknowledge or otherwise remove the appointment from the system's memory!

Depending on which type of alarm you specify, an appointment reminder can range from the subtle (just a little annunciator displayed on the screen) to the not-so-subtle (how about a wailing siren every 15 seconds?!).

There are lots of features of this appointment system that I don't have the space to go into detail on. Let me say that I have spent several hours "playing around" with the various capabilities of just this mode of the unit and I still don't know all of what it is capable of doing.

Oh yes, you say you might have large programs to run and you do not want to waste any of your valuable memory with appointment schedules? No problem. All the appointments are held in a "file." That file, like any other file in the system, can be offloaded to a mass storage device and reloaded any time! I will have more to say about files....

The Time, Please!

How does the HP-75 keep track of the arrival of appointments? Why, with its real-time clock, of course. And what a clock. Do you know of any other PC that allows you to adjust the accuracy of its real-time clock?

This clock can be personalized to suit your own preferences. You can set it for 12- or 24-hour format. You can even arrange the Day/Month/Year template to your own liking. The clock can be used to activate the unit or external devices, called by programs to record actual times at which events occur, etc. And, as I said above, you can calibrate and increase or decrease the speed of the clock.

The Editing System

Up until the HP-75C, the Sharp/Radio Shack PCs had, in my opinion, the best editing capabilities of the portables. Virtually every other manufacturer of PCs had slipped in this area by leaving out vital editing functions. Such as, not providing the ability to scroll backwards (from a high numbered line to a lower numbered line). The HP-75C has virtually every feature of the, say, Sharp PC-1500, plus many more.

To start with, editing begins at the *file* level. Hewlett-Packard defines a "file" as "an area of memory that can be identified by name and manipulated as a unit." Oh yes, you can have many separate files stored in memory at any time.

To keep track of your files, there is a cataloging function built into the operating system. Any time you want to review what files you have in memory, you just ask for the system catalog. It will summarize all files by their names, type, amount of memory each consumes and the time and date at which each was established.

Files can be duplicated in memory or transferred (unless secured) to external devices (including magnetic strip-cards).

There are several different types of files. Two of the most commonly used types are termed BASIC and TEXT files. Yes, one is intended primarily for the development of programs, the other for the storage of plain text files. But, would you believe, there is a TRANSFORM command that enables you to convert one type of file into the other!

Once you have established a file, there are all kinds of editing functions at various levels. For instance, an automatic line numbering

option is available. Even more valuable to program developers is the BASIC renumbering capability. You can specify the renumbering of all or just part of a BASIC program. A very nice capability to have provided on a PC.

The sophistication continues down to the character level. A set of editing keys provide complete line and within-line scrolling capability. That is, you can scroll backwards and forwards through complete lines in a listing. Or, you can scroll across a single line. Shift options enable you to instantly jump from the beginning to the end of a line and vice versa. Similarly, a "fetch" key enables you to quickly specify a particular line within a listing or instantly return to the last line you were working on. Other editing keys provide the ability to insert or delete individual characters within a line.

In summary, the general file, text and program editing capabilities of this machine are superb! If you have any problem in this area, it will simply be in *learning* about all of the editing capabilities that are available.

My question, after examining the HP-75C, is why haven't the desktop computers had these kinds of capabilities provided as standard years ago? How many desktop units that you know of will let you merge files in memory? Yes, you can even do that (whole merges or partial-file merges) on the HP-75C!

It's a Calculator, too!

One hardly has to point out that any HP product would automatically include a calculator, but I will mention some of the types of functions that are provided as standard on this product.

You can assign values to variables directly from the keyboard. You can specify real, short form or integer variables. You can extract absolute, integer part, fractional part or integer values. You can set "floor" or "ceiling" values in calculations. You can find the square root, modulus, sign, maximum or minimum value or remainder of a number. You can generate random numbers and make the sequence repeatable or arbitrary. You can specify angles in radians or degrees. You can calculate the sine, arcsine, cosine, arccosine, tangent, arctangent, cotangent, secant and cosecant. You can use the Boolean operators AND, OR, EXCLUSIVE OR and NOT. In other words, you can just plain "calculate" using the HP-75C in its "keyboard calculator" mode. Of course, all these calculating functions are also available in the "program" mode.

Using the Magnetic Strip-Cards

The built-in ability of the HP-75C to store information on or read from magnetic strip-cards puts it a big step ahead in the area of portable practicality. No matter how portable you make a computer, if you cannot load different programs into the unit quickly and conveniently (and without having to carry a separate briefcase full of paraphernalia to accomplish the task!), then you simply cannot take advantage of the computer's principle asset: that it is able to perform a wide variety of tasks on a moment's notice. With the HP-75C, all you need carry with you to load in a new program (or data), or to save programs/data, are a few long, slim, magnetic strip-cards!

These strip-cards are about 3/8"ths of an inch wide and 10 inches in length. Frankly, I would prefer to see these cards about three inches shorter. At 10 inches they tend to be a little top-heavy when carried in a shirt pocket. (Please excuse me for being so picky.) But, compared to the alternatives (you know — the briefcase full of gear), they are an absolute dream come true. Each card can hold up to 1.3 kilobytes of data or program instructions. In fact, there are two magnetic tracks on a card. Each track holds 650 bytes plus "header" information. It takes two passes through the card reader, once for each track, to transfer all the information that can be stored on a strip-card.

Of course, with that 48 kilobyte operating system in ROM, all of the strip-card operations are prompted, supervised and verified by the HP-75C. Suppose you want to load a new program into memory? OK, you just give the system command COPY CARD TO "FILENAME". This command creates a new file in memory, ready to receive the information that is on the card(s) you are going to process. At this point, the operating system proceeds to prompt you through each step of the card reading procedure. You are told to insert and align a card in

the reader. (The reader is simply a slot provided on the front of the HP-75C.) Next, you are directed to pull the card through the slot. A nice steady pull is all it takes. (If you don't do it right, the computer tells you whether you were too fast or slow!) After one track has been read, the computer "knows" how many other tracks need to be processed. It then prompts you for further inputs as necessary. The order in which a multi-track file is processed is irrelevant. The operating system takes care of organizing the various tracks into a complete file. With a little practice handling the cards, a 2.6K program (spread over a set of two strip-cards) can easily be loaded into memory in less than 30 seconds.

Writing information on cards is similarly prompted and guided by the operating system. Indeed, the process includes verifying the validity of the information stored on the card(s).

Oh, by the way, the HP-75C has several levels of program security. For instance, invoking the *private card* option prevents a user from listing, altering or duplicating a program. Invoking the *password* option can restrict a file to only those who know a user-specified password. And, a *protect* option allows you to safeguard strip-cards from being accidentally overwritten. You can combine security options to increase the level of security, if desired. Such security provisions can, of course, provide a measure of comfort to those who need to create application packages that will be utilized by personnel to whom one does not wish to divulge proprietary algorithms, techniques or data.

Connecting to Peripherals

I have been suspicious for some time about the significance of Hewlett-Packard providing some rather powerful peripheral devices for their programmable HP-41 calculator. It has, frankly, seemed to me to be a bit of "overkill" to provide accessories such as the HP 82161A Digital Cassette Drive, capable of storing some 131,072 bytes of data, merely to provide storage for a calculator. After all, the drive costs roughly twice that of the HP-41 itself! Ah, but what if the real plan was to have a drive that could serve the calculator *as well as other devices*. And that, of course, was the plan.

You see, the HP-75C Portable Computer is equipped with HP-IL (Hewlett-Packard's Interface Loop). This makes it directly compatible with all those peripherals, such as the digital cassette drive just mentioned, and the HP-82162A Thermal Printer, that were recently made available for the HP-41.

The HP-IL system is a simple method (analogous, in some respects, to the old "teletype loop") for connecting a series of devices together. One device (the HP-75C, in this case) acts as the "talker," the others act as "listeners/relayers."

Part of the HP-75's operating system (remember that big 48K of ROM that I keep babbling about) contains all that is needed to quickly configure a system with peripherals. You just jumper the peripherals you want to use with HP-IL cables. (Two cables come with the HP-75, you get one more with each peripheral you purchase.) Then, you use an ASSIGNIO statement to assign device codes according to their serial position within the loop. From then on you can communicate with your peripherals through program or keyboard commands.

One other point I will make before moving on is this: The HP-IL system is physically neat. You can, for instance, set up a "home base" system of peripherals, such as a video monitor, tape drive and printer. When at your home base, you simply plug in the two ends of the HP-IL connectors into the top of the '75 (this takes about five seconds). Presto. You are operating as a "desktop" system. Need to go on the road with your PC? Unplug those two connectors (takes about two seconds!) and you are on your way.

The Keyboard is Infinite

Well, very close to it. Almost every key is redefinable. And not merely to be another character, but to be your choice of another character, an entire expression or a command (or series of directives)! Not only that; any keyboard configuration you create can be stored as a file in memory. You can set up keys to call whole "keyboard configuration" files. Thus, you can instantly change the definitions of virtually all of the keys. The variations are endless. Believe me, you can create some highly sophisticated (to say nothing of "personalized") application programs with this type of key-defining capability. (It is analogous to

having a couple of hundred "softkeys" on a Sharp PC-1500, with the added ability of being able to instantly re-define all those keys!)

Powerful HP BASIC

I am only going to point out some of the special features about Hewlett-Packard's version of BASIC, after noting that, with 48K of ROM, they have provided a rather complete implementation of the popular language.

People nurtured on Microsoft BASIC will find it takes some getting used to dealing with strings in HP BASIC. For instance, you cannot define two-dimensional string arrays on the HP-75. You may, however, define a string to be any length (up to available memory). You can then create a function that subdivides a string so that each section can be treated as a separate element. The net result is essentially the same as manipulating a string array or matrix, but it takes some extra thought and care, at least until one gets used to the concept.

On the other hand, there are some string handling functions available in HP BASIC that simply are not available on many other machines. For instance, the UPRC\$ function can convert lowercase strings to their uppercase representations. And, the POS string function can be used to locate the position of one string inside another.

A particularly strong area of HP BASIC is that it provides the option for users to create their own single-line or multiple-line functions. Thus, for example, if you frequently need to perform a particular operation, such as converting degrees Centigrade to Fahrenheit, you can just define the function. You give it a function name, such as FNC, and embody it in the form FNC(C), where the (C) indicates the parameter that is to be passed (Centigrade degrees, in this case) to the function. Any time you need to perform the conversion, you just invoke the FNC(C) function. It is a little neater than having to call a subroutine to perform such a procedure. What is more, variables used within a function definition are *local to that function*. However, variables used outside a function are available (global) to all functions. You can use function definitions to perform some pretty fancy and powerful programming.

The '75 has a rather interesting way of storing and recalling data in "external" files. (A file separate from the program being executed.) Access to such a file is essentially treated as though you were creating or reading from DATA statements! You can specify line numbers for the DATA statements as you create and write to them. Alternately, you can use default methods to have line numbers automatically assigned as needed. To recover information from such external files, you use READ # statements (as opposed to just plain READ statements when the data is within the same file as the program itself).

This method of processing external files is intriguing. Files created in this manner are accessible by the normal editing capabilities of the system. Thus, they are easily reviewed. In fact, they can be manually modified, if desired. I think most people will find this method of handling data in files easier than traditional computer methods. This is primarily because it provides for easier conceptualization of the external filing process.

The HP-75 also provides the ability to process this data serially or using random access techniques. Furthermore, special forms of the data-handling statements enable one to perform operations such as overwriting a particular line of data. It is thus possible to organize a random access file where the data elements vary in length! Other directives permit entire numeric arrays to be processed as a unit. Finally, ability to access plain text files under program control is provided, using similar types of DATA statements.

However, perhaps the most exciting capability of HP BASIC as implemented on the '75 is the ability to perform what HP terms a *program call*. Now, this "program call" capability is not the same as the old "chaining" method of loading in another program, with which you may be familiar. It is much more powerful!

First of all, the program CALL statement passes control to another program just as though that program were a subroutine. The calling program expects to eventually have control returned to itself.

However, the called program is executed as though it was an entirely separate unit. Variable values are not passed to the called program

unless an external file is established for this purpose.

Never-the-less, the calling program *retains all its variable value assignments* (even those with names identical to ones used in the called program) and resumes operations with its retained variable values when control is returned!

The net result is this: You can create an entire series of programs, each to perform a specific operation or function. You build each of these programs in its own file and without regard to line numbering, variable assignments, etc., (unless you intend to pass specific parameters between programs). Then you can construct a master program (or programs) to call upon the various smaller programs as required to perform desired tasks.

And, if that form of operation in itself is not enough, *these program calls can be recursive!*

Now that, readers, is an example of providing real personal computing power.

There Is Still More

Of course, there are other (more mundane?) capabilities that one *might* expect in a PC. You can format your output to just about any layout desired using IMAGE and USING statements. These directives permit the mixing of numerical and string values. You can even specify the type of radix or separator symbol desired (European or American).

Even the beeper is programmable - and directly by specification of the frequency and duration. (All you music lovers: take note!)

Finally, I will mention that good debugging features are also provided. You can trace variables and/or program flow (branches). You can execute a program a step at a time. And, you can set up error traps.

Is There Anything Missing?

The only thing I could think of after spending a considerable amount of time reviewing this machine is that there is no provision (that I could find) for operating the unit's liquid-crystal display (LCD) in a bit-mapped mode. Ah well, most applications I have seen for utilizing bit-mapped graphics on a single-line display have been rather trivial. So, I am really not going to fault HP for leaving such capability out. You will just have to be satisfied with the 128 characters and their underlined counterparts that can be shown on the 32-character screen.

Styling

My compliments to HP's designers on the overall layout and design of this machine. It feels nice and it looks nice. The position of, for instance, the strip-card reader is practical and convenient. And, the accessibility of the ROM drawers (right on the left-hand front of the unit) is splendid. One gets the distinct impression that the design group put a great deal of thought into this model. It will be appreciated by discerning users, I am sure.

By the way, whether designed for use in this position or not, I have found that the HP-75 fits quite comfortably (while touch-typing, at that) in my lap. If you travel a lot, such as by plane, the ability to work in this position (comfortably) could be of value.

The Manuals

You receive two manuals with the HP-75C. One is a huge, 360+ page, 8-1/2 by 11 inch volume entitled the *HP-75 Owner's Manual*. It is intended to serve as both a tutorial introduction and a reference manual. It is nicely organized, contains numerous illustrative examples, has a number of useful appendices, and a comprehensive index.

The second is named the *HP-75 Reference Manual*. It contains approximately 70 pages (8-1/2 by 11 inches). The purpose of this manual is to serve as a quick refresher and guide once you are basically familiar with the operation of the unit.

I rate this set of manuals: *superb*.

The Bottom Line

This gem sells for \$995.00. I predict many discerning professionals will consider it a worthwhile investment. It is a pleasure to work with a tool that exhibits this level of overall quality and excellence of design.

Well done, HP.

— Nat Wadsworth, Editor

ROBER MNEMONICS UPDATE

Norlin Rober has some additions and corrections to the material that appeared in the recent Special Edition of *PCN*. Plus, he has gleaned a lot of new knowledge. Here is a compilation of the latest findings by the Master Sleuth (on the Sharp PC-1500/Radio Shack PC-2):

1. Since the CPU does not use true indexed addressing, registers X, Y and U should more properly be referred to as Pointer Registers, not Index Registers.

2. The description of the operation of flag V contains an error. With reference to subtractions it should read "... is the same as that of the operand subtracted" rather than "is opposite to that of the operand subtracted."

3. The instruction designated INXY (opcode F5) does more than just increment X and Y. Prior to the incrementing, it transfers the contents of the address pointed to by X into the address pointed to by Y. The mnemonic should be changed to: STI (X) (Y).

4. The ADD instructions with opcodes EF, 4F, 5F and 6F are "Add without carry." That is, the condition of the carry flag prior to the addition has no effect. (The result of the addition, however, does affect the flags as usual.)

5. The condensed ROM map on page 4 of the Special Edition of *PCN* lists D5BF - DCAD as containing code. It should be D6BF - DCAD. Also, there is another brief lookup table located at E4E3 - E4EA.

6. Some additional instructions have been determined:

PWR DOWN (Code FD 4E) switches off the computer.

DSP OFF (Code FD C0) turns off the display.

DSP ON (Code FD C1) turns on the display.

7. New information on interrupts: A maskable interrupt (usable only when flag I is set, enabling the interrupt) pushes E and P onto the stack, then transfers control to the interrupt service routine that begins at E171 (the address stored in FFF8 & FFF9). The RTI instruction at the end of the interrupt routine pops P and E from the stack.

It appears that the BREAK key does not use the interrupt routine. When this key is pressed, bit 1 of alternate memory address F00B is set to 1. This address is checked routinely by instructions in ROM.

The manual paper advance apparently produces an interrupt.

It appears that the instruction represented by opcode FD CE sets the timer to produce a "timer interrupt" after a specified length of time. This time is related to the contents of the accumulator at the time FD CE is used, but the exact connection is not clear. If the accumulator contains zero, there is no interrupt at all. In other cases, the timer produces an interrupt after a length of time that depends on the accumulator contents. This length of time seems to be about 25 milliseconds at the longest. The timer interrupt routine begins at E22C, the address obtained from ROM locations FFFA & FFFB. It will occur only if flag I is set.

The FD CE instruction would presumably be followed by a WAI instruction to produce a programmable delay lasting until the interrupt occurred.

The FD DE opcode, not used in ROM, is similar to FD CE in effect, but the resulting delay times are different.

8. New information on inputting to the CPU from the keyboard: Connections to the 64 keys (exclusive of the ON key) form the electrical equivalent of an 8 by 8 matrix. The keyboard is polled as follows: The presence of a 1 bit in a particular position in the contents of alternate memory buffer address F00C specifies the corresponding column

of the matrix. The CPU instruction LDA KB loads the accumulator with the byte having a zero bit in only the position corresponding to the row (of the specified column) in which a key is pressed. The byte loaded into A is then used to determine a location in the lookup table (FE80 - FEFF), from which the appropriate ASCII code is obtained.

The ROM routine that performs this operation begins at address E42C.

9. The power-up routine begins at E000, the address stored in bytes FFFE & FFFF of ROM.

10. Although the CPU has a non-maskable interrupt capability, it apparently is not used. The circuit diagram in the Service Manual shows the NMI pin as being grounded. The interrupt routine for NMI is located at E22B, which is the address stored in FFF8 & FFF9. This routine contains the RTI instruction.

11. According to the service manual, a third kind of interrupt, designated as a "Timer Interrupt" exists. It seems likely that the address stored in FFFA & FFFB, which is E22C, is where the Timer Interrupt begins. Note that at E22C there is a short routine ending with RTI and that it contains the code FD CE.

12. The addresses following the tokens in the list of BASIC words (ROM addresses C054 to C34D) are the starting addresses of the routines that execute the associated BASIC statements. For example the BEEP routine begins at E5C1. Note that words which may not begin a BASIC statement, such as THEN, TO, AND and OFF, are followed by CD89. This is the address at which the routine for ERROR 1 begins.

The routines executing BASIC statements end with CALL E2, where the stack pointer is reset and the next BASIC statement is read. If an error condition is to occur, CALL E4 (for ERROR 1) or CALL E0 (other ERROR) ends the BASIC statement routine.

The tokenized words representing functions, however, call on subroutines ending with RTS.

13. Revised versions of the ROM in the PC-1500 may differ from the ROM described above, particularly in regards to exact addresses of the various addresses.

14. Thanks to James Stutsman for pointing out that two separate ROMs may occupy the address space 8000 to BFFF at one time. The opcode B8, which I am giving the mnemonic ROM1, selects the ROM used by the printer/cassette interface. Opcode A8 with the mnemonic ROM2 selects the other. The "other" could be the RS-232 interface promised by Sharp.

15. You can upgrade the disassembler program to include the items discussed in this column with the following lines:

```
345 LPRINT "STI (X) (Y)";RETURN
```

```
476 LPRINT "PWR DOWN";RETURN
```

```
592 LPRINT "DSP OFF";RETURN
```

```
593 LPRINT "DSP ON";RETURN
```

16. If you would like the printout of disassembled ROM to include the addresses of subroutines called from the base page, make the following modifications to the disassembler program:

A. End line 50 with GOTO 53 (instead of GOTO 54).

B. Add line 53 to the program as shown here:

```
53 LPRINT " ";TAB 14;C=D+65280:D=PEEK C:GOSUB 24:  
D=PEEK(C+1):GOSUB 24:D=PEEK A
```

[Norlin indicates that he is getting quite a bit of mail. Please remember when writing to him or any author, that it is always a nice courtesy to include a S.A.S.E. (self-addressed, stamped envelope) if you would like a reply. — N.W.]

PC-1500/PC-2 RENUMBERING PROGRAM

Milt Sherwin, 8602D E. Amherst Drive, Denver, CO 80231, provides this sought after utility. Space limitations in this issue restrict us to the presentation of brief operating instructions and the program listing. However, Milt has prepared a nice article explaining how the program works. If you are interested, let *PCN* know and we will try to present the details in an upcoming issue.

You need at least a 4K memory module in your PC to use this program. Use the program as follows:

1. Clear memory then load the program.

2. If you wish to renumber a previously developed program, then MERGE it into memory at this point, before trying to run the renumbering program. If you are going to develop (key in) a program that you may decide to renumber later, then immediately execute the renumbering program by going to the RUN mode and issuing a RUN command. This operation simulates a MERGE and protects the renumbering program from itself!

3. Develop your program. Assign a label to the start of the program under construction or modification so that you can test it by reference

to the label. This is necessary because the renumbering program simulates the MERGE operation. You will not be able to access your new program if you do not give it a label.

4. Anytime you want to renumber the program under development, go to the RUN mode and issue a RUN command. This activates the renumbering program. Respond to the prompt: ST,IC,EN,LN with the appropriate values as indicated here:

ST = Starting line number (defaults to first line of user's program).

IC = Line increment (defaults to an increment value of 10).

EN = Ending line number (defaults to last line of user's program).

LN = Initial line number assignment for renumbering (default is 100).

Use a comma after inputting each value or if you want to use the de-

fault value. (Thus, entering three commas [, , ,] would result in the renumber program using all its default values.)

5. At the end of each RUNNING of the renumbering program, it will ask if it should be "unlinked." Answer Y for yes if you no longer wish to use the renumbering program. Answer N for no if you plan further development work. Repeat step 4 as necessary.

6. Once the renumbering program has been unlinked, you may LIST or CSAVE the user's program in a normal manner. The renumbering program is no longer accessible.

7. When you are all through, use NEW0 to normalize your PC. (The renumbering program accesses various pointers at the machine language level. NEW0 assures that all pointers are re-initialized.)

100: IF PEEK 30821< =100	0	=DTHEN 720
>PEEK 30825AND 270: I=VAL B\$: IF I=	490: INPUT "UNLINK	700: NEXT X
PEEK 30822<> 0THEN LET I=10	RENUM? ";A\$	710: GOTO 760
PEEK 30826THEN 280: E=VAL D\$: IF E=	500: IF LEFT\$ (A\$, 1	720: D=PEEK (&7150+
150 0THEN LET E=65	><)"Y"THEN 520	X)*256+PEEK (&
110: IF PEEK 30824+	510: POKE 30821,	7150+X+1)
1=256THEN POKE 290: L=VAL E\$: IF L<	PEEK 30825:	730: D\$=STR\$ D: R=
30826, PEEK 308	POKE 30822,	LEN D\$: IF R=L>
24-255: POKE 30	PEEK 30826	0GOSUB 790:
825, PEEK 30823	520: END	GOTO 760
+1: GOTO 130	530: C=PEEK M: D=INT	740: IF R-L<0GOSUB
120: POKE 30826,	(C/16): B=D*409	830: GOTO 760
PEEK 30824+1:	6+(C-D*16)*256	750: GOSUB 880
POKE 30825,	:A=C	760: A\$="": K=0: IF A
PEEK 30823	540: C=PEEK (M+1): D	=&2CTHEN 620
130: POKE 30823,	=INT (C/16): B=	770: GOTO 580
PEEK 30825:	B+D*16+(C-D*16	780: RETURN
POKE 30824,): D=C	790: I=PEEK (30823)
PEEK 30826	550: C=PEEK (M+2): Z	*256+PEEK 3082
140: POKE PEEK 3082	=Z+1	4: H=I
3*256+PEEK 308	560: RETURN	800: POKE (1+R-L),
24, 255: GOTO 52	570: A\$="": J=M+3: K=	PEEK I
0	0	810: I=I-1: IF I>K
150: CLEAR : DIM A\$(580: A=PEEK J: G=	THEN 800
0)*26: C=1: P=&7	PEEK (J+1)	820: GOTO 860
050: R=&7150: M=	590: IF A=&0DTHEN 7	830: I=K: H=PEEK (30
PEEK 30825*256	80	823)*256+PEEK
+PEEK 30826	600: IF A=&F1AND G=	30824
160: INPUT "ST, IC, E	&92OR G=&94OR	840: POKE I, PEEK (I
N, LN "; A\$(0)	G=&AEOR G=&AB	-R+L)
170: FOR J=1TO LEN	THEN LET J=J+1	850: I=I+1: IF I<H
A\$(0)	: GOTO 620	THEN 840
180: C\$=MID\$ (A\$(0)	610: J=J+1: GOTO 580	860: C=C+R-L: POKE (
, J, 1)	620: J=J+1: A=PEEK J	M+2), C: J=J+R-L
190: IF C\$="," THEN	630: IF A<&30OR A>&	870: H=H+R-L: POKE 3
LET C=C+1: GOTO	39THEN 660	0823: INT (H/25
250	640: IF K=0THEN LET	6): POKE 30824,
200: ON CGOTO 210, 2	K=J	H-INT (H/256)*
20, 230, 240	650: A\$=A\$+CHR\$ A:	256
210: A\$=A\$+C\$: GOTO	GOTO 620	880: FOR X=1TO R
250	660: L=LEN A\$: IF L=	890: A\$=MID\$ (D\$, X,
220: B\$=B\$+C\$: GOTO	0THEN 580	1): POKE K, ASC
250	670: D=VAL A\$	A\$: K=K+1
230: D\$=D\$+C\$: GOTO	680: FOR X=0TO P	900: NEXT X
250	STEP 2	910: RETURN
240: E\$=E\$+C\$	690: E=PEEK (&7050+	STATUS 1
250: NEXT J	X)*256+PEEK (&	
260: S=VAL A\$: F=0: N	7050+X+1): IF E	

1935

RPN CALCULATOR FOR PC-1/PC-1211

Here is *Norlin Rober*, 407 North 1st Avenue, Marshalltown, IA 50158, with another one of his creative endeavors. This program converts the Radio Shack PC-1 or Sharp PC-1211 to a double-precision, Reverse-Polish Notation (RPN) calculator (similar in operation to Hewlett-Packard machines). The program permits addition, subtraction, multiplication and division. All inputs and outputs use scientific notation. There are four "guard digits" calculated with each operation, and the 20 most significant digits are displayed.

The "stack" consists of four registers, referred to as X, Y, Z and T. Register X uses variables A, B, C and D. Register Y uses E, F, G and H. Register Z uses variables I, J, K and L. Register T is stored in M, N, O and P. The first variable associated with each register contains the first 8 digits of the mantissa, the second variable holds the next 8 digits. The third variable holds the last 8 digits. The fourth variable stores the decimal exponent.

For those unfamiliar with RPN, its operation is described briefly as follows: When an input occurs, it is stored in register X, with the previous contents of X shifted into Y, the previous contents of Y into Z and those of Z into T. This shifting of the register contents is called a *stack lift*.

An arithmetic operation is performed using the contents of X and Y as operands. The calculated result is placed into register X. Such an operation is accompanied by a *stack drop*. That is, register Z is shifted into Y and T drops into Z (as well as remaining in T).

The four registers may be viewed as "stacked" on top of each other in the following form:

```
REGISTER T
REGISTER Z
REGISTER Y
REGISTER X
```

The RPN system is efficient as it is very convenient to use previously calculated results in further operations. The primary principle to keep in mind is that an operation (add, subtract, multiply or divide) is given *after* the numerical entries for that operation have been made and not *between* operands as is done with an algebraic calculator.

The following operations are performed using the PC when it is set to the DEF mode:

- SHIFT/L Lift stack, with input to replace X. Note: If ENTER is pressed without any input being keyed, the previous contents of X remain there. This is useful, for example, in squaring a number as the number need not be entered twice.
- SHIFT/N Non-lifting input. The input simply *replaces* the contents of X. No stack lift occurs.
- SHIFT/X Exchange the contents of X and Y. Useful when it is desired to exchange the operands prior to division or subtraction.
- SHIFT/A Add X and Y. Result in X, with stack drop.
- SHIFT/S Subtract X from Y. Result in X, with stack drop.
- SHIFT/M Multiply X by Y. Result in X, with stack drop.
- SHIFT/D Divide Y by X. Result in X, with stack drop.

Inputs must be in scientific notation, including a decimal point after the first digit of the mantissa. Negative numbers should only have the minus sign entered in the first 8-digit block.

The prompt **READY** is displayed after numerical or operand inputs. Keying **ENTER** (alone) shows the current contents of register X as a 20-digit mantissa. Keying **ENTER** a second time displays the exponent.

Example: To calculate $(2.176513 + 408.99716238)/(79.312294758 - .4851339176288)$, proceed as shown below.

With the program loaded satisfactorily, set the PC to the DEF mode.

(1) SHIFT/L. Enter inputs as follows:

```
1ST 8 DIGITS? 2.1765130
NEXT 8?       0
NEXT 8?       0
EXP?          0 (Note: Inputs in scientific notation.)
```

(2) SHIFT/L (again) and enter inputs of the second operand:

```
1ST 8 DIGITS? 4.0899716
NEXT 8?      23800000
```

Program RPN Calculator for PC-1/PC-1211

```
1 "L" M=I:N=J:O=K:P=L:I=E:J=F:K=G:L=H:E=A:F=B:
   G=C:H=D
2 "N" Q=E-4:R=IE8:S=IE-8
3 INPUT "1ST 8 DIGITS? ";A:A=IE7A:B=0:C=0:D=0:
   INPUT "NEXT 8? ";B;"NEXT 8? ";C;"EXP? ";D
4 PRINT "READY":X=IE-6B:Y=X-INT X+IE-10*INT
   (CQ+.5:X=IE-9*(ABS IE2A+INT X+INT Y:Y=IE10*
   (Y-INT Y:Z=D
5 IF X=10 LET X=1:Z=Z+1
6 PRINT USING "########";X*SGN A;USING
   "#####";Y:PRINT USING;"EXP ";Z:GOTO 6
7 "" E=I:F=J:G=K:H=L:M=N:K=O:L=P:GOTO 4
8 "X" W=A:X=B:Y=C:Z=D:A=E:B=F:C=G:D=H:E=W:
   F=X:G=Y:H=Z:GOTO 4
20 "S" A=-A
21 "A" U=SGN AE:IF U=0 LET A=A+E:B=B+F:C=C+G:
   D=D+H:GOTO 4
22 Z=D-H:IF Z*IE16+(ABS A-ABS E)R+B-F+(C-G)S LET
   W=ABS E:X=F:Y=G:GOTO 24
23 W=ABS A:X=B:Y=C:A=E:B=F:C=G:D=H:Z=-Z
24 IF Z > 25 GOTO 4
25 IF Z > 7 LET Z=Z-8:Y=X+YS:X=W:W=0:GOTO 25
26 IF Z LET Z=10^-Z:Y=YZ+(XZ-INT XZ)R:X=INT XZ+
   (WZ-INT WZ)R:W=INT WZ
27 V=SGN A:A=ABS A+WU:B=B+XU:C=C+YU:GOTO 50
30 "M" V=SGN AE:A=ABS IE-3A:B=IE-3B:C=10C:D=D+H:
   E=ABS EQ:F=FQ:W=A-INT A:X=B-INT B:Y=E-INT E:
   Z=F-INT F
31 T=W*INT E+Y*INT A:U=W*INT F+X*INT E+Y*INT
   B+Z*INT A:C=(BG+CF)QS+(U-INT U+XY+WZ)R+AGQ+
   BF+CEQ
32 B=INT A*INT F+INT B*INT E+INT U+(T-INT T+WY)R:
   A=INT A*INT E+INT T:GOTO 50
40 "D" V=SGN E/SGN A:D=H-D-1:H=ABS A+BS:A=ABS
   AQ:B=BQ:C=CQ:T=A-INT A:U=B-INT B:W=INT((ABS
   ER+F)/H)*Q
41 X=W-INT W:Y=U*INT W+X*INT B:E=(ABS E-INT A*
   INT W-T*INT W-X*INT A-TX)R+F-INT B*INT W-INT Y
42 F=G-(Y-INT Y)R-UXR-C*INT W-CX:X=INT((ER
   +F)/H)*Q:Y=X-INT X
43 C=((E-INT A*INT X-T*INT X-Y*INT A-TY)R+F-
   INT B*INT X-U*INT X-Y*INT B-UY)R-CX)/H:A=IE4W:
   B=IE4X
50 C=CS+1:B=(B-1+INT C)S+1:A=A-1+INT B:B=(B-INT B)R
   :C=(C-INT C)R
51 IF A=0 IF B=0 IF C=0 LET D=0:GOTO 4
52 IF A=0 LET A=B:B=INT C:C=(C-INT C)R:D=D-8:
   GOTO 52
53 IF A < IE7 LET U=1+INT LOG A:D=D+U-8:U=10^-U:
   C=CU:B=BU:A=AUR+INT B:B=(B-INT B)R+INT C:
   C=(C-INT C)R
54 IF A >= R LET A=.1A:B=.1B+(A-INT A)R:C=.1C+(B-
   INT B)R:B=INT B:A=INT A:D=D+1
55 A=AV:GOTO 4
```

NEXT 8? 0
EXP? 2

(3) SHIFT/A, X and Y are added. The result is in X. You can view X by keying ENTER (after READY appears). It should be 4.1117367538000 000000 EXP 2.

(4) SHIFT/L. Now entering the denominator of the original expression:

1ST 8 DIGITS? 7.9312294
NEXT 8? 75800000
NEXT 8? 0
EXP? 1

(5) SHIFT/L (once more!) and enter:

1ST 8 DIGITS? 4.8513391
NEXT 8? 76288000
NEXT 8? 0
EXP? -1

Right at this point, the internal stack contents are:

REGISTER Z 411.17367538
REGISTER Y 79.312294758
REGISTER X 4851339176288

(6) SHIFT/S. This calculates Y-X and drops the stack one notch. You can examine the result of the subtraction by pressing ENTER after the READY prompt appears. The stack now contains:

REGISTER Y 411.17367538 (previously in Z)
REGISTER X 78.827160843712 (after subtracting)

(7) SHIFT/D. This performs the division operation to yield the final desired result for the original expression. Key ENTER to display it:

5.216142139 2893562452 EXP 0

When Keying in the Program

Remember, the character "E" stands for the EXponent key, not the letter E.

Also, be sure and enter the program exactly as shown. A number of "tricks" to cram a lot of operations into the PC, such as implied multiplication, are used. Just enter the program *exactly* as shown, then try the example to see if you get the same results. If an error shows up, examine the line indicated in the error message for a keying mistake.

[So what else can you wring out of a little old PC-1? Nice work, Norlin. Some of you programming pro's might want to try a hand at converting this one for the PC-2/PC-1500. Looks like it would come in mighty handy in many applications. The speed of the PC-1500 should bring up results in the blink of an eye! — N.W.]

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January — December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

☐ 1982 Regular Subscriber (Issues 11 — 20). \$30.00 in U.S. (U.S. \$36.00 to Canada. U.S. \$45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 — 30). \$60.00 in U.S. (U.S. \$72.00 to Canada. U.S. \$90.00 elsewhere.)

☐ 1983 Regular Subscriber (Issues 21 — 30). \$36.00 in U.S. (U.S. \$42.00 to Canada. U.S. \$50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA #: _____ Expires: _____

Signature: _____



P.O. Box 232, Seymour, CT 06483

FROM THE WATCH POCKET

If you are interested in closely following the emergence of Hewlett-Packard's PCs, I suggest you send a large 9 by 12 inch, self-addressed envelope with 2 ounces of first class postage on it, to: PPC, 2545 West Camden Place, Santa Ana, CA 92704. The organization puts out several journals of interest to HP-oriented PC enthusiasts. Sending in that stamped envelope addressed to yourself will reportedly bring you samples of the material put out by this organization.

Casio PC users might want to inquire about a newsletter being published by Department 'G' Software, 212 South Oak Street, Owatonna, MN 55060.

Sharp Electronics will be providing U.S. deliveries of its new RS-232 interface beginning in November. The list price will be \$225.00. The unit can be connected directly to the PC-1500 or the printer/cassette interface. It is reported to be capable of providing two-way, asynchronous, communication signals, programmable at rates of 50, 100, 110, 200, 300, 600, 1200 and 2400 baud. Other parameters, such as character size, parity and stop-bit length, are also programmable. A set of four AA nickel-cadmium batteries will power the unit for up to two hours in portable operation. The unit is also reported to contain firmware that converts the PC to a communications terminal. The system can then be connected (through an appropriate modem) into communications networks via telephone, connected to external devices such as full-sized printers or plotters, used to monitor instruments, talk with a desktop computer or drive a video display device.

Sharp is also due to begin deliveries of its ROM software modules this month. Mort Rosenstein of Atlantic Northeast Marketing, PO Box 921, Marblehead, MA 01945, telephone (617) 639-0285, says he will have them as soon as they become available.

With the Casio FX-700P coming in at \$99.95, how long do you suppose Radio Shack will try to hold a \$149.95 price on the PC-1? I bet they move to give it *and* the PC-2 a substantial reduction in time for the Holiday Season this December.

By the way, I hear now that the price of the FA-3 cassette interface for the FX-700P will be \$49.95. I am surprised. I figured a \$99.95 PC would have, say, a \$29.95 cassette interface. After all, even the "high priced" Radio Shack PC-1 can be equipped with a cassette interface at \$29.95. Oh yes, word is that Casio *will* bring out a little printer for their newest PC. Probably won't be on the shelves, though, until the first quarter of '83.

Several people have pointed out that the PC-1500 has a CPU register designated as register W in the PC-1500 Service Manual. No machine codes for the use of this register were given in PCN's Special Edition. That is right, they are missing at this point. But, as the saying goes, "People are working on it."

If anyone is still trying to load tapes for a PC-1 into a PC-2, *forget it!* Several people have complained that some advertisements seem to infer that "programs written for a PC-1 can be loaded directly into a PC-2." They have then spent hours trying to load tapes created for the PC-1. While technically the PC-2 is capable of executing the language statements of the PC-1, this can only be done if no programming shortcuts (such as implied multiplication) are used in the PC-1 version. But, irrespective of this, *you cannot load a PC-1 tape into a PC-2!* The tape recording characteristics are simply not compatible!

Martin Shapiro has been having a lot of fun building "humungous" (that is an acronym for "very large") programs for his PC-2. He buys TDK endless loop cassettes, (the kind that do not have any end of tape foil on them). Then he loads 'em up with programs. He can chain endlessly from program to program without pausing to fool around with the recorder controls. Do you see possibilities for your own applications from that idea?

While on cassettes, several readers have cautioned that attempting to use most microcassette recorders for storing programs requires use of top-of-the-line tape. The lower grade ones just can't seem to hack it.

Walter Tolbert, P.O. Box 1465, South Gate, CA 90280, telephone (213) 566-7449, wants to form a club of PC and HHC users in his area. Contact him if you are interested in forming such a group.

You can expect to see a flood of PC-2/PC-1500 software hitting the market in the next few months.

— Nat Wadsworth, Editor

POCKET COMPUTER NEWSLETTER



© Copyright 1982 — Issue 19

November

PANASONIC ESTABLISHES HHC SERVICE ON COMPUERVE

Panasonic Company, A Division of Matsushita Electric Corporation of America, One Panasonic Way, Secaucus, NJ 07094, has announced the creation of the Panasonic Users Group (PUG) bulletin board on the nationwide CompuServe network.

The bulletin board service is available to any member of CompuServe. The PUG bulletin board operates as one of a number of Special Interest Groups (SIGs) now available on the network. The service is intended to serve Panasonic's HHC users *as well as users of other hand-held and pocket computers.*

The board already contains a number of programs that can be downloaded directly into HHC units. Of course, a HHC must be equipped with a communications modem in order to connect to the network via telephone.

According to *Edward J. Gelb*, National Sales Manager of HHCs for Panasonic, his company will be providing one hour of free introductory service on the CompuServe network to Panasonic consumer customers who purchase a modem for their HHC. Users can then extend their one hour of introductory time into a regular CompuServe service contract if they are pleased with the world of network information services. (The standard night-time connect charge is \$5.00 per hour.)

Current users of the CompuServe network can access the PUG board by using the command GO PCS106 at the command prompt (!) which is given at the end of page CIS-1 after sign-on. Operation of the board is similar to that of other SIG services offered by the network.

One of the primary purposes of PUG is to provide a rapid means of information exchange among HHC and other pocket computer users. This is accomplished through the use of an on-going message bulletin board. Up to 256 messages can exist on this board at one time. Anyone having news of general interest to other users can leave a message on this board. Other members can review current messages and add additional comments, if desired. The medium thus serves as a constant source of up-to-the-minute news on using HHCs, a way to stay in touch with Panasonic software personnel, and share ideas and software accomplishments with other users.

The PUG service may also be used to download programs and documents directly to Panasonic HHCs. Some of the material is also applicable to other brands of hand-held and pocket computers. Current offerings (at no charge, by the way, other than normal CompuServe connect time) include: a program and documentation (operating instructions) for a four-function calculator that can operate in several numeric bases, a program to map memory in a HHC, documentation for operating the new HHC Plotter using BASIC, and a program in BASIC for solving up to 20 simultaneous equations (it appears that this program could readily be adapted to run, say, on a Sharp PC-1500 or Radio Shack PC-2 equipped with an 8K memory module).

The PUG board may also be used to communicate directly with members of the Panasonic HHC staff to obtain prompt responses to user questions. At this time, the PUG SYSOP (system operator) appears to be none other than Mr. Gelb, National Sales Manager for HHCs. That should guarantee pretty prompt responses to any Panasonic HHC users who are in need of assistance.

ROUTINE CONVERTS STATEMENTS

Sharp PC-1500 and Radio Shack PC-2 users will appreciate the use and operation of this small routine. It will automatically scan a program in memory and convert all PRINT statements to LPRINT or vice versa.

An understanding of its operation will enable readers to apply the principles used to other conversion tasks.

Essentially, what the program does is search all of user memory for the exact two-byte code representing the token for a PRINT or an LPRINT statement (depending on which option has been selected). If found, it replaces the second byte with the desired alternate code.

The address at which to begin the search (the start of user program storage) is obtained by examining the contents of memory locations &7865 and &7866. (Remember, the prefix '&' means the number is in the hexadecimal base.) The end of user program storage is available through the STATUS 2 function. These values are used to establish the boundaries of memory that are to be searched. This area is then examined on a byte-by-byte basis for an exact match with the two-byte token being sought. If a match is found (say, with the token for PRINT) then the second byte of the token is changed (to, for instance, represent the LPRINT operation).

To use the program, just load it into memory and type RUN. Respond appropriately to the initial query. (Enter 1 if you want all occurrences of LPRINT to be changed to PRINT, enter 2 if you want the opposite conversion.) Allow the program to run until it signals it has completed the task by beeping. You may then list or CSAVE the converted program. Note that this program is particularly valuable if you developed a lot of programs before you obtained a CE-150 printer/plotter and want to update your library to make use of your new equipment.

This routine was submitted by: *Thomas S. Cox, %Platt Saco Lowell, P.O. Drawer 2327, Greenville, SC 29602.*

Program Tokens Converter

```
1: "LP" INPUT "LPT      :END
   -PRT(1) OR P-L      6: POKE (I+1), 185
   PT(2)"; A: IF A=    :RETURN
   1 GOTO 7             7: C=PEEK &7865*2
2: IF A=2 GOTO 4        56+PEEK &7866:
3: END                 FOR I=CTO (
4: C=PEEK &7865*2      STATUS 2-1): A=
   56+PEEK &7866:      PEEK I: B=PEEK
   FOR I=CTO (        (I+1): IF (A=24
   STATUS 2-1): A=    0)+(B=185)=2
   PEEK I: B=PEEK    GOSUB 9
   (I+1): IF (A=24   8: NEXT I: BEEP 20
   0)+(B=151)=2      :END
   GOSUB 6           9: POKE (I+1), 151
5: NEXT I: BEEP 20     :RETURN
```


UNDERSTANDING THE SHARP PC-1500

This is the third article in a series being presented by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

More Hidden Instructions

In Issues 15 and 16 of *PCN*, the uses of PEEK and POKE were discussed. Here are a number of other instructions of which you may not be aware.

The PC-1500's processor can address two separate banks of memory, each of which could conceivably contain as much memory as 64K. The first bank addresses the System ROM, CE-150 (Printer/Cassette) ROM, System RAM and user RAM. At present, at least, the "alternate buffer" (mentioned in Radio Shack's Instruction Manual, but not Sharp's) contains only the input/output (I/O) ports. The PEEK# and POKE# instructions operate in the same way as PEEK and POKE, except that they address the memory in this second bank of memory.

The use of POKE# can affect the buzzer, clock, remote switches that control the cassette recorders, pen solenoid, paper feed, pen position, and other I/O functions. Be forewarned that using POKE# can lead to serious problems — not only system crashes, *but possibly physical damage to connected devices.* I'll give you an example: While experimenting with POKE#, I had turned on the pen solenoid, but had not realized that it remained on. A few minutes later I somehow happened to notice that the solenoid had become quite hot! As it turned out, no permanent damage had been done, but luck must have been with me.

Thus, unless you know what you are doing (or you like to live dangerously!), you would be well advised not to experiment with the POKE# instruction.

The use of CALL will cause execution of a machine-language program beginning at a specified memory address. Those who choose not to delve into machine language can still use CALL in certain ways, provided the desired address to call is known. I'll give you an example that you might find interesting and useful.

The power-up routine checks certain flags to determine whether the cause of the last power-down was the automatic 7-minute shutoff. If the automatic shutoff *had* occurred, certain parts of the power-up routine are skipped over. The operations skipped include clearing the display, canceling execution of a program not finished, re-seeding the random number generator, and running the pen through its color change cycle.

You can perform the equivalent of a 7-minute shutoff (without having to leave the computer on for seven minutes!) by executing the instruction CALL &E33F. This instruction may be used as part of a BASIC program too, in case you want the computer to turn itself off under program control. *[It is a nifty command to assign to a softkey in the Reserve mode, such as: F1: CALL &E33F@. Remember, the "@" gives you immediate execution of a softkey! — N.W.]* It is also possible to obtain the equivalent of having used the OFF key, by execution of CALL &CD71.

It is possible to clear the variables in "main" memory (without clearing those in "fixed" memory, that is A — Z and A\$ — Z\$), by execution of CALL &D091. This would permit re-dimensioning of variables without loss of the fixed variables, within a program.

Another instruction not mentioned in the manuals is the NEW command followed by an expression. Nor is the difference between just plain NEW and NEW0 made clear.

NEW will effectively clear a program (by setting the End Program pointer at &7867-68 to coincide with the Start Program pointer at &7865-66) without changing the starting location. On the other hand, NEW0 resets both pointers to &C5 plus the beginning address of RAM physically present in the computer. Another difference is that NEW0 will cancel LOCK mode, whereas NEW will not.

When NEW is followed by an expression (within the permissible range), the value of that expression determines the memory location at which program lines will be entered. Thus, execution of the directive NEW &4800 will take address &4800 as the beginning location at which program lines will be placed into memory, whether entered manually (in PRO mode) or loaded from cassette. Addresses below

&4800 are then protected.

Machine-Language Programmers Take Note!

One use for this NEW <expression> command is to reserve space for a machine-language program in such a way as to protect it from being changed by storage of data or BASIC program lines.

Another use is reserving a segment of memory for the efficient storage of data with PEEK and POKE. (Such data would need to consist of integers from 0 to 255 or, with appropriate programming, one could store larger integers.)

It is not possible, however, to execute NEW &4000 (NEW &3800 if an 8K module is used) to absorb Reserve memory into program space. A safeguard is built into the NEW routine to prevent this. Attempting to do so results in the appearance of ERROR 25. That same message appears if you attempt starting your program at an address higher than that of existing memory (e.g., NEW &6800).

If the Start of Program pointer has been relocated by, for example, NEW &4800, executing NEW will leave it there (while effectively clearing the program, of course); but, NEW0 will set it back to its normal location.

Special Cassette Commands

A considerable amount of versatility is added to the cassette storage system by the use of the CLOAD M and CSAVE M instructions. Execution of CSAVE M permits saving memory contents between two specified memory addresses. CLOAD M reloads what has been stored on a tape by CSAVE M. Machine-language users will immediately recognize this as a very convenient method for saving machine-language routines.

To illustrate: Suppose you have an 8K expansion RAM module. You have placed a machine-language program into the area &38C5 to &39D0, having left Reserve memory (&3808-C4) to operate in the normal manner. Suppose further, that you have executed NEW &3A00 to separate your BASIC program from the portion reserved for the machine-language program. You can now save the machine-language program by executing CSAVE M "NAME"; &38C5, &39D0. (If you would like to save Reserve memory along with it, then just use the directive CSAVE M "NAME"; &3808, &39D0.) The BASIC portion can be CSAVE'd in the usual manner.

When you want to reload this program at some later time, you execute CLOAD M "NAME" or simply CLOAD M. This will place the machine-language portion back into its original location in the computer's memory. Of course, NEW &3A00 must be executed before CLOADing the BASIC program or else the machine-language routine would be obliterated.

CLOAD M "NAME"; <address> is also a valid command. Its effect is to load the contents of the cassette into memory beginning at the address given, rather than at the same location it had when CSAVE'd.

If you are using two cassette recorders, you may find it of interest to note that CLOAD M-1 and CSAVE M-1 are valid instructions as well!

Unfortunately, there appears to be no way to verify a recording made using CSAVE M. As you might guess, there are no MERGE M or CHAIN M instructions.

The idea might occur to you that we have here a way to save both Reserve memory contents and a BASIC program simultaneously with CSAVE M &3800, STATUS 2. Later CLOAD M might be used to load both back into the computer. But, there is a hitch! It does get loaded back into the computer all right, but the End of Program pointer does not get set. Thus, the computer is misled into thinking that it doesn't contain a program at all.

CLOAD M will load only a recording made by a CSAVE M instruction. A recording made with CSAVE M can be CLOADed by only the CLOAD M command.

[That is some super information. Thanks, Norlin. If anyone is looking for a good project to apply their machine-language programming skills on, take a look at the next article. With all of the m.l. material that has been supplied through the pages of PCN, I expect to see lots of clever m.l. and hybrid programs appearing on the scene in the near future! — N.W.]

RENUMBERING PROGRAM REVISITED

It soon became apparent from the mail received after Issue 18 of *PCN*, that many readers could use an explanation of the renumbering program provided by Milt Sherwin, 8602D East Amherst Drive, Denver, CO 80231. Here is how Milt describes its development and operation:

Program Concepts and Development

In developing a renumbering program, it is necessary to know the format of the BASIC lines (line number storage, linkage arrangements, BASIC text representation, etc.). Once this information is obtained, the choices revolve around how sophisticated the program should be, e.g. (1) how much flexibility will the user be provided in choosing the parameters, (2) does the program renumber line numbers following GOTOs, GOSUBs, THENs, and REMs, and (3) how easy the program is to use in general. By far the most difficult item is (2), the renumbering of GOTOs, GOSUBs, THENs and REMs. Of course, this directly influences item (3), how easy the renumbering program is to use!

BASIC program storage begins at hexadecimal address &38C5 (which is decimal 14533) in a PC-1500 equipped with an 8K model CE-155 RAM module. (The address is &40C5 if there is no memory module or only a 4K module is installed.)

BASIC lines are stored in memory with the following format:

1 byte	1 byte	1 byte	N bytes	1 byte	1 byte
Line Number (high)	Line Number (low)	Pointer	BASIC text	Carriage Return "&0D"	*

The byte denoted by an asterisk here contains the high byte of the next line number or &FF if this is the last line of the program. This works OK since the highest line number allowed by the interpreter is 65279 (hexadecimal \$FEFF).

Line numbers at the beginning of statement lines are stored as two bytes (high, low). Thus, line 10 decimal equals bytes 00 0A in hexadecimal. Likewise, 65279 converts to FE FF.

The value stored in the byte marked "pointer" indicates how many bytes to the line-ending carriage return (taken from the pointer location itself).

BASIC lines are made up of BASIC tokens and ASCII coded characters. The code for carriage return indicates the end of a BASIC line. Thus, the line "300 GOTO 90" would be stored in the PC-1500 as:

01 2C 05 F1 92 39 30 0D

Knowing this format enables a programmer to "step" through a program, line-by-line, using PEEKs and a few program loops.

Once the program starting point and the format of BASIC lines are known, it is easy to find the line number locations and appropriate tokens (for GOTO, GOSUB, THEN or REM) that would be associated with renumbering. It is also possible to determine the line numbers that follow those tokens, using the string-manipulating functions CHR\$, LEN and VAL, etc. However, the line numbers following tokens are stored as ASCII digits, and line numbers such as 5, 50, 500 and 5000 all require a different number of bytes for storage (1, 2, 3 and 4 respectively). A comprehensive renumbering program must take this into consideration. The plot, thus, thickens! If a program to be renumbered contains a line such as "300 GOTO 90" and renumbering causes it to become "310 GOTO 100", then an additional byte is needed because the number following the GOTO statement has now expanded to three digits! That means everything stored after that location (perhaps an entire program!) must be relocated by one byte to create room for the new digit. This is not an easy undertaking. A similar problem occurs in the opposite direction when a referenced number decreases, such as when "310 GOTO 100" becomes "300 GOTO 90". Now one byte must be removed and everything above that point moved "down" appropriately. One more little point: If the line length changes because digit(s) are added/subtracted, then the line length "pointer" must also be altered to reflect the new line length!

Note that these problems do not occur with statement line numbers. They are always stored in hexadecimal, two-byte notation. Hence,

statement line numbers are easily modified without encountering expansion/contraction problems.

There is another aspect of program renumbering which must be considered during program design. The relationship of the original line numbers to the renumbered line numbers must be kept track of in order to renumber the line numbers following the GOTOs, GOSUBs, THENs and REMs. The location of this line number relationship table could be critical to the program's operation. During the operation of the renumbering program, the variables storage area could be disturbed if the program size changes due to the altering of line numbers following tokens. One way to avoid this potential problem is to POKE (store) and PEEK (read) numbers directly into "safe" areas in memory. I selected two areas which should be OK for this purpose. Namely, the fixed variable storage areas for character variables E\$ through O\$ (at addresses &7050 through &70FF) and P\$ through Z\$ (at addresses &7150 through &71FF). None of these character variables are used by the main part of the renumbering program. [And, it makes no difference if these areas are used by the program being renumbered. After all, you cannot execute the program being renumbered at the same time that the renumbering is occurring! — N.W.] Thus, this area is effectively protected whenever it is being used for storage of the number relationship table.

The first area, &7050 through &70FF is used to hold the original line numbers. The 176 bytes here can store 88 line numbers. The second area, &7150 through &71FF can also hold 88 line numbers too. So, that is where the renumbered line values are held.

If you have more than 88 lines in a program, then renumber it in sections! [Or, use the information provided by Norlin Rober in this issue to relocate the line-number relationship tables to a larger, protected area of memory. After all, now we know how to set aside as much memory as we want for special purposes! — N.W.]

Next, there is the problem of "where does the renumbering program itself reside in memory?" If it is made a part of the main program (numbered so that it resides at the end of the program being developed), then the dynamics of the changing line lengths in the program being renumbered will effect the renumbering program as it tries to operate. That won't work.

A separate renumbering program could be written which could be MERGED with the program under development. Would this approach work? The MERGE command is indeed interesting. It allows a user to have more than one program in memory at one time. If each program uses alphabetical labels, it can be accessed and executed. However, only the last program MERGED can actually be modified by the system's editor. Furthermore, no selectivity is provided if a program is LISTED or CSAVED. Everything in memory is simply listed or saved. Furthermore, in order to develop a new program one must initially MERGE a program from tape. You cannot just start typing in line numbers and BASIC statements. And, even if the renumbering program is MERGED after program development and is used to renumber the program already in memory, there is still a problem. While the renumbering program could then be removed line-by-line, the program under development would still not be accessible to the editor. This is because the &FF byte which terminated the MERGED renumbering program would still be in memory following the &FF byte that terminated the original program under development. That is, there would be two consecutive &FF bytes in memory which would prohibit alterations being made by the editor. Of course, the program could be CSAVED and then CLOADED. But, that takes a lot of time and is, frankly, cumbersome. But, is there light at the end of the MERGE tunnel?

What if the renumbering program were loaded first? Then, development work could take place in memory beyond the renumbering program. The program under development could also be expanded or contracted without disturbing the renumbering program. The renumbering program would not have to be MERGED for each use or taken out to continue the development process. But, what about overcoming the previously mentioned problems with MERGE?

There happen to be three pointers (taking a total of six bytes, each has a high and low part) which indicate the memory address of what I call the "start of BASIC" (SOB), the "end of BASIC" (EOB) and the

"start of program" (SOP) locations.

30821	30822	30823	30824	30825	30826
Start of BASIC (SOB)		End of BASIC (EOB)		Start of Program (SOP)	
High	Low	High	Low	High	Low
&7865	&7866	&7867	&7868	&7869	&786A

In the non-MERGE mode of operation, pointers SOB and SOP are the same. When there is a MERGE, pointer SOP contains the starting address of the MERGED program. This is the reason that additions, changes and deletions only work on the last program MERGED. Pointer EOB points to the &FF byte of the last program in memory. A &FF byte also terminates every program that is MERGED into memory. The SOB pointer normally remains fixed at &38C5 (14533 decimal) in a PC-1500 containing an 8K memory module. Would it be possible for a renumbering program to modify these pointers?

Yes indeed! And, that is just what my renumbering program does. The methodology is as follows:

- 1) The renumbering program is CLOAded into memory.
- 2) A RUN command causes the renumbering program to alter the SOP and EOP to simulate the performance of a MERGE.
- 3) Program development may then take place. The program being developed should be given a label for use when testing. (Thus, one can say LIST "A" or RUN "A" as appropriate.)
- 4) Subsequent RUN directives cause the renumbering program to renumber the program that is under development.
- 5) At the end of a renumbering pass, the renumbering program asks whether it should be "unlinked." If so, the program modifies SOB so that it equals SOP, thereby effectively removing the renumbering program from further access.
- 6) Once the renumbering program has been unlinked, the program under development can be LISTed, CSAVED or executed using normal methods.
- 7) Execution of the command NEW0 restores all three pointers (SOB, EOB and SOP) to their pristine states.

Furthermore, if a previously developed program needs to be renumbered, it can simply be MERGED in place of step 2 above. Doing so causes SOP and EOB to be set appropriately by the MERGE operation instead of by the renumbering program.

For reference purposes, the behavior of the SOB, EOB and SOP pointers are summarized here:

DURING REGULAR OPERATIONS

NEW0/START SOB = EOB = SOP = 14533
 PROGRAMMING/CLOAD SOB = SOP = 14533, EOB > 14533
 MERGE SOB = 14533, SOP > 14533, EOB > SOP

DURING RENUMBERING OPERATIONS

NEW0/START SOB = EOB = SOP = 14533
 CLOAD RENUM SOB = SOP, EOB > 14533
 INITIAL RUN/MERGE SOB = 14533, SOP = EOB > SOB
 DEVELOPMENT SOB = 14533, SOP > SOB, EOB > SOP
 UNLINK SOB = SOP(Development), EOB > SOP
 NEW0 SOB = EOB = SOP = 14533

(Value of 14533 is for a PC-1500 with an 8K memory module installed.)

Operation of the Renumbering Program

Now that you are familiar with the concepts behind the design of the program, the actual implementation of the program can be described. Please remember that the program only renumbers statement line numbers and line numbers referenced by GOTO, GOSUB, THEN and REM statements. *It will not recalculate line numbers that are obtained by manipulating variable names or performing similar types of programming tricks!*

Lines 100 - 140: The values of SOB and SOP are compared to determine if they are equal, indicating that a MERGE has not been performed. If equal, a MERGE is simulated, program execution terminates and the user can undertake program development. If SOB and SOP are not equal, then the program assumes that program development has taken place (either through an actual tape MERGE or a previous exe-

cution of this program).

Lines 150 - 250: Variables are initialized and renumbering parameters are requested from the operator. There are four parameters used by the program:

PARAMETER	DEFINITION	DEFAULT VALUE
ST	Starting line number	First line of the development program
IC	Line increment	10
EN	Ending line number	Last line of the development program
LN	Initial line number when program renumbered	100

Commas must be used to separate the parameters during input *even if values are not entered for some parameters*. Thus, inputting ", , , 200" will result in the current development program being renumbered in its entirety, with line increments of 10, and the first line of the renumbered program being 200. However, if the RETURN key is pressed without entering any parameter values or commas, then default values will be used to renumber the program. The parameters are input as a single character variable, then split into four individual variables for use within the program.

Lines 260 - 300: The variables used to store the parameters are set for operation at their default values or the values inputted by the user.

Lines 310 - 360: The starting location is found.

Lines 370 - 450: Line numbers are renumbered. The original line number/renumbered line number table is built. The subroutine at line 530 is used to determine the original line numbers.

Lines 460 - 480: Variables are set for renumbering line numbers that follow GOTOS, GOSUBs, THENs and REMs. Each line is searched for the corresponding tokens using the subroutine at line 570.

Lines 490 - 520: The operator is provided the option of unlinking the renumbering program. If unlinking is to occur, SOB is set equal to SOP. If not, the program is terminated. Further program development may then take place.

Lines 530 - 560: Subroutine to determine a line number. Line numbers, stored as two bytes, are converted to single line number values.

Lines 570 - 610: First part of the subroutine that searches for GOTO, GOSUB, THEN and REM tokens. (Renumbering line numbers after a REM only works if the line number follows immediately after the REM.) If no appropriate token is found, detection of the code for carriage return in line 590 causes the subroutine to terminate (via line 780).

Lines 620 - 650: Second part of the token-searching subroutine. If an appropriate token is found, the ASCII representation of the associated line number(s) is(are) located. As they are found, they are converted so they may be stored in a character variable for later use.

Lines 660 - 710: Part three of the subroutine evaluates the line number stored as a character variable. The previously created line number table is used to determine the referenced renumbered line number.

Lines 720 - 780: The fourth part tests to determine if the renumbered line number has the same number of digits as the original line number. If the renumbered line number has more digits than the original line number, the subroutine starting at line 790 is used to create more space. If the new number has less digits, the subroutine at line 830 is used to remove the excess space. If the line numbers have the same number of digits, the renumbered line number is placed in position by the subroutine at line 860. If no match is found in the table, no renumbered line number is required (as the original line number has not changed). The variables are then reset. The search for more line numbers or tokens resumes.

Lines 790 - 820: Subroutine to create additional space in a program.

Lines 830 - 870: Subroutine to remove excess space in a program.

Lines 880 - 910: Subroutine to place a renumbered line number into memory.

[OK, all you machine language programmers. Here is an opportunity for you to really have some fun. Milt has described in detail how his renumbering program operates. Machine-language speed would be nice in this application. Who is going to tackle the project? - N.W.]

MEMO PRINTER PROGRAM

Here is a program for Radio Shack PC-2 and Sharp PC-1500 users who want to keep notes or text in memory. This program was created by: *Brian Peterson, 6807 N. Sheridan Road, Chicago, IL 60626.* One of the interesting features about this program is that *it prints sideways!* Thus, it accepts and prints out full 80-character lines. Brian calls his program "Memo Printer." Here is what he has to say about using it:

It Operates In Three Modes

These modes are referred to as: Input, Edit and Print.

The Input mode allows the user to enter characters into memory. You need at least a 4K memory expansion module to use the program. With a 4K module, it can accept approximately 33 lines of text. An 8K module allows about 88 lines. Theoretically, a 16K expansion unit would allow up to about 187 lines. Each line may contain up to 80 characters.

If you want to leave a blank line between paragraphs, just enter a single 'space' character on the line. To get out of the Input mode, press the ENTER key without putting any characters in the line.

Editing

You use the Edit mode to go back and change text that was originally entered while in the Input mode. The Edit mode allows you to examine the text buffer and make alterations. You can insert characters and entire lines, if desired.

Several keys have different functions than normal when in the Edit mode. The DEF key is used to activate the ability to insert or delete lines. After the DEF key is pressed, the PC will ask whether the user wishes to insert or delete. Enter 'I' to insert. Enter 'D' to delete. When in the delete phase, the CL key deletes individual characters. Holding the key down will cause it to repeat and delete a series of characters. The MODE key enables the user to insert characters at the current cursor position. When this insert function is in effect, the symbol > will appear in its inverse form (filled-in) on the screen. The key normally used to select the desired Reserve mode (<) is used to append lines to

the end of the text. The scroll line up (^) key and the scroll line down (v) key are used to examine lines above or below the current text line. The symbols <, >, ?, :, ; and comma are available through the use of the SHIFT key. Use the SML key to enter lower case characters. Tip: When entering text, put the unit in the lower case mode and use the SHIFT key when a capital letter is needed.

The six softkeys can be customized to allow the use of characters not usually available on the keyboard. Program lines 247 - 252 can be altered for this purpose.

Press the keys firmly when using the Edit mode to allow time for the character to be recognized. Most keys will repeat if held down in this mode.

To exit the Edit mode, press the RCL key.

Printing

This program prints "lengthwise" on the paper. Options for selecting the pen color as well as single- or double-spacing are provided.

Use GOTO

Once you have text in memory, always enter the program using a GOTO directive, not RUN. Using RUN will clear out any text that you have placed in memory.

There Is No Stopping

This program disables the ON/BREAK key during operation, except when printing.

[While this last feature is interesting, I am not convinced that it is desirable. I was recently using the program while away from the office, got into the Edit mode, and forgot how to get out. With the BREAK key disabled, I had no choice but to leave the unit on until I got back to the office, as I did not want to reset the PC and lose everything that I had in the unit. I don't think the risk of accidentally striking this key, nor the possible consequences of doing so, are worth not being able to shut the unit off if I forget how to operate the program. In any event, a word to the wise: Keep the operating instructions handy until you have memorized all of the options available!— N.W.]

Program Memo Printer (Part 1)

5: ARUN : WAIT 0:	End"	STATUS 3-23	55: G=0
POKE# &F00D,	16: V=ASC INKEY\$:	42: IF VAND ASC	60: FOR X=1 TO 61
PEEK# &F00DOR	IF VCALL	INKEY\$ THEN 42	STEP 20
128	STATUS 3-23	43: C=V-17: IF V<17	70: FOR Y=T-1 TO 0
10: IF STATUS 3=&5	17: IF VAND ASC	OR V>20 GOTO 40	STEP -1
800DIM P1\$(0)*	INKEY\$ THEN 17	44: PRINT " SNG	75: X\$=MID\$ (A\$(Y+
80, P2\$(0)*80, A	18: ON V-16 GOTO 20	DBL -SPA	G), X, 10)
\$(32)*80: N=33	, 16, 200, 16, 40,	CING-"	80: IF X\$="" GOTO 1
11: IF STATUS 3=&6	600	45: V=ASC INKEY\$:	10
000DIM P1\$(0)*	19: GOTO 15	IF VCALL	90: GLCURSOR (200-
80, P2\$(0)*80, A	20: LOCK : L=0:	STATUS 3-23	19*Y*S, -(X*12)
\$(87)*80: N=88	USING	46: IF VAND ASC	100: LPRINT X\$
12: IF STATUS 3=&6	25: ON ERROR GOTO	INKEY\$ THEN 46	110: NEXT Y
400DIM P1\$(0)*	0	47: IF V<>17 AND V<	120: FOR Y=0 TO T-1
80, P2\$(0)*80, A	30: PRINT "Line"; L	>19 GOTO 44	125: X\$=MID\$ (A\$(Y+
\$(186)*80: N=18	; ">"; : INPUT "	48: S=(V=17)+2*(V=	G), X+10, 10)
7	; A\$(L): CLS : L=	19): T=11*(V=17	130: IF X\$="" GOTO 1
13: POKE STATUS 3-	L+1: IF L<NGOTO	>+6*(V=19)	60
23, 72, 118, 74, 0	30	49: CLS : GRAPH :	140: GLCURSOR (200-
, 5, 189, 255, 65,	35: M=L-1: UNLOCK :	LINE -(220, 0),	19*Y*S, -(X+10)
78, 78, 153, 8	BEEP 1: CLS :	0, C: TEXT : LF 5	*12)
14: POKE STATUS 3-	GOTO 15	50: GRAPH : SORGN :	150: LPRINT X\$
11, 76, 119, 139,	40: PRINT " BLK BL	ROTATE 1: CSIZE	160: NEXT Y
6, 72, 119, 74, 0,	U GRN RED -CO	2	170: NEXT X
158, 18, 154	LOR-"	52: POKE# &F00D,	
15: PRINT "Input	41: V=ASC INKEY\$:	PEEK# &F00DAND	
Edit Print	IF VCALL	127	

program continued
on next page


```

175: G=G+1: IF M>=G      PEEK &764E)):      0: CLS : GOTO 40      D: FOR X=DT0 M:
TEXT :LF 5:      GOTO 225      0      A$(X)=A$(X+1):
GRAPH :SORGN : 232: W=8: GOSUB 550: 261: Z=(Z=0): GOSUB      NEXT X: A$(M)="
ROTATE 1: GOTO      F=(8=(8AND      ": M=M-1: L=D:
60      PEEK &764E)): 262: GOTO 272      GOTO 210
180: TEXT :LF 5:      GOTO 225      270: IF E+F=1 IF U>6 420: INPUT "Insert
BEEP 1: POKE# & 238: P=P-(P<>1):      4LET K$=CHR$ (      line after lin
F00D, PEEK# &F0      GOTO 223      UOR 32)      e#"; I: I=I+1
00OR 128: GOTO 239: M=M+(M<N-1): L=      271: IF U>39 IF U<48 430: FOR X=M-(M=N-1
15      M: GOSUB 500      IF E=1 GOSUB U-      )TO 1STEP -1: A
200: ON ERROR GOTO 240: L=L+(L<M): P=1:      39+600      $(X+1)=A$(X):
350: L=0      Q=1: R=5: GOTO 2      NEXT X: A$(1)="
210: CLS : WAIT 0: P=      21      ": M=M+1: L=I+1:
1: Q=1: R=5: IF D 241: L=L-(L<>0): P=1      GOTO 210
LET L=L-(L>0)      : Q=1: R=5: GOTO      500: IF ASC INKEY$
220: USING "####":      221      THEN 500
GOSUB 580      242: P=P+(P<=LEN A$      510: RETURN
221: Z=0: E=0: F=0:      (L)): GOTO 223      550: U=PEEK &764E:
GOTO 224      243: P=1: Q=1: R=5:      IF UAND WPOKE
223: Q=1: R=P+4: IF P      GOSUB 580:      &764E, U-W:
>21LET Q=P-20:      GOSUB 500: GOTO      GOSUB 500:
R=25      221      RETURN
224: PRINT L; ">"; 247: K$="!": GOTO 27      555: POKE &764E, U+W
MID$ (A$(L), Q,      2      : GOSUB 500:
21): IF ZCURSOR 248: K$="," : GOTO 27      RETURN
4: GPRINT 0; 127      2      580: POKE &764E, 245
; 62; 28      249: K$="#": GOTO 27      AND PEEK &764E
225: Y=0: CURSOR R:      2      : E=0: F=0:
PRINT CHR$ 127 250: K$="$": GOTO 27      RETURN
226: K$=INKEY$ : U=      2      600: POKE# &F00D,
ASC K$: IF U>32 251: K$="%": GOTO 27      PEEK# &F00DAND
GOTO 270      2      127: END
227: IF UGOTO 230+U 252: K$="&": GOTO 27      601: K$="<": RETURN
228: X=X+1: IF X<6      2      602: K$=">": RETURN
GOTO 226      254: P1$(0)=LEFT$ (      603: K$=" ": RETURN
229: X=0: Y=(Y=0): IF      A$(L), P-1): P2$      604: K$="," : RETURN
YCURSOR R:      (0)=MID$ (A$(L      606: K$="," : RETURN
PRINT CHR$ 127      ), P+1, 80-P): A$      608: K$="?": RETURN
: GOTO 226      (L)=P1$(0)+P2$
230: CURSOR R: PRINT      (0): GOTO 223      STATUS 1
MID$ (A$(L), P, 255: GOSUB 500:      3376
1): GOTO 226      GOSUB 580: L=M:
231: W=2: GOSUB 550:      GOTO 15
E=(2=(2AND 257: W=128: GOSUB 55

```

STOPWATCH PROGRAM FOR R. S. PC-1/SHARP PC-1211

Radio Shack PC-1 and Sharp PC-1211 users can use this program to precisely time events down to 1/10th of a second intervals!

Timer Design

Two conditions (rules) should be met when designing an accurate time keeping device: 1) There should be a large number of cycles ("ticks") per second, and 2) The regularity of each cycle must be as stable as possible. Both of these rules are met quite nicely by the Cesium Atomic Clock. It "ticks" at a frequency of 9,192,631,770 Hertz (cycles per second). Its regularity is within 1 part in 10,000,000,000,000. That makes it the most accurate timing device known to man.

The design rules mentioned, of course, can be applied to the design of computer "clock" programs. Unfortunately, many programmers

do not follow these rules. Consequently, inaccurate (errors up to several minutes each hour) times results.

Environmental temperatures can also have an effect on accuracy. Laboratory experiments performed on the Radio Shack PC-1 and the Sharp PC-1211 indicate that program execution speed increases slightly with increasing temperature. In equation form, it was found that:

$$S(T)/S(T=0) = 2.0261E-5*(T+273.15)+0.994466$$

when T is between 0 and 40 degrees Centigrade. S(T) is the relative execution speed, T is the temperature in Celsius of the PC, and S(T=0) is the reference speed at 0.0C (defined, arbitrarily, as 1.000) in this equation. Using this equation, one can observe that at 40 degrees C., a program executes about 0.08% faster than the same program at a temperature of zero. Though these effects are relatively small, they can be significant over an extended period of time. The program provided

in this article utilizes a temperature compensator that may be utilized in extreme temperature ranges.

Other experiments indicated that battery condition had little effect on the execution speed. Using very weak batteries in the PC (with the display barely visible) gave the same results as fresh batteries. This is good news. It means you can run the accompanying program for extended periods of time without fearing a loss in accuracy due to battery condition.

The stopwatch program provided here has been pre-calibrated against a laboratory chronometer. Chances are, it will not require any additional calibration in your PC. However, if you should find the program slightly fast or slow on your particular PC, it contains an easy to use calibration routine. This routine calculates the exact calibration constant required for an individual PC. The routine, if used at all, only needs to be run once after loading the program into memory.

A Stopwatch Program

This program makes a good substitute for an ordinary 1/10th second stopwatch. It can be calibrated to give errors of less than about 0.2 seconds per day. While this is not quartz accuracy, it is better than some mechanical spring-wound watches. The output is given in hours, minutes, seconds and tenths of a second.

The design rules discussed above were used in designing this program. For instance, examine the basic time-keeping routine in line 8 of the program.

To satisfy the first rule, this line is made to execute as fast as possible by making it short and simple as well as using a high speed variable (Z).

The second rule (perhaps more important), is satisfied by assigning Z a large initial value (see line 4). This was done because tests have shown that a calculation of $Z=Z+1$ takes slightly longer as the order of magnitude of Z increases. Thus, it would not be wise to start off with $Z=0$, as the "clock" would "tick" slower and slower as Z rapidly increased several orders of magnitude. Instead, an initial value of $Z=800000$ is used. It then takes several days before the order of magnitude changes and each "tick" is thereby made constant.

Operation

The program is designed to be executed in the DEF mode. You press SHIFT/S to place the PC stopwatch into the start position. Then, to actually start timing an event, just press the ENTER key. Use the BREAK key as the stop button. To display the elapsed time, simply press SHIFT/D. An example output might appear as:

TIME (H.MMSST) = 2.07583

This would be interpreted as 2 hours, 7 minutes and 58.3 seconds.

Calibration

If you should find the stopwatch program to be slightly off on your particular unit, you can execute the calibration routine. To obtain a calibration you need to first run the program, as described above, then press SHIFT/C. An example illustrates the process:

You time an interval of exactly 4 hours, 0 minutes and 0 seconds. But, when you press SHIFT/D to display the elapsed time, you find that the stopwatch is off by a significant amount. So, you run the calibration routine:

SHIFT/C

CALIBRATION -->

TRUE (H.MMSST): 4.0000

NEW Y = 0.17658008

You would have entered the value '4.0000' in the above example. The NEW Y value would then have been computed by the calibration routine. The calculation uses the difference between the actual elapsed time and that originally obtained using the stopwatch program.

You complete the calibration process by placing the PC in the PRO mode, then changing the constant Y in lines 4 and 32 to the new value. You then use this value in the future on your PC.

Note that a calibration run should be made over a several hour period so as to keep relative errors to a minimum. Also, you need run this procedure only once (if at all) for a particular PC.

If you are using your PC at temperature extremes that are greater than 85 degrees Fahrenheit or less than 55 F., then you might also

want to run the temperature calibration routine. This routine assumes you have already calibrated your PC for normal temperatures. Press SHIFT/X to perform this calibration and respond to the prompt for temperature and the scale. The routine will provide a new value of Y for you to use in lines 4 and 32.

Summary

The program is designed to be used in the DEF mode. Here is a list of the DEFINED keys and their functions:

SHIFT/S — Puts the PC into the 'ready to start' position.

ENTER — Starts the timing of an event.

BREAK — Stops the timing of an event.

SHIFT/D — Displays the elapsed time.

SHIFT/C — General calibration procedure.

SHIFT/X — Extreme temperatures calibration procedure.

This program provided by: Emerich Auersbacher, 41 King Street #2, Belleville, NJ 07109.

Program Stopwatch

```
1 BEEP 1:PAUSE " STOPWATCH PROGRAM"
2 PRINT "USE (DEF) MODE PLEASE.":END
4 "S":Z=800000:Y=0.1766080237:BEEP 1
6 PRINT "READY: (ENTER) TO START"
8 Z=Z+1:GOTO 8
10 "D":T=DMS((Z-800000)*Y/3600):T=T+0.000005
12 PRINT USING "###.####";"TIME(H.MMSST)=";T:
   GOTO 4
14 "C":PAUSE "CALIBRATION -->"
16 INPUT "TRUE (H.MMSST):";T:T=DEG T
18 T=3600:T=Y=T/(Z-800000)
20 PRINT USING;"NEW Y = ";Y:END
22 "X":PAUSE "TEMP. ADJUST -->":INPUT "TEMPERA
   TURE:";T
24 INPUT "DEGREES C OR F?";D$
26 IF D$="F" LET T=5/9*(T-32):GOTO 32
28 IF D$="C" GOTO 32
30 GOTO 24
32 Y=0.1766080237:Y=1.000416768*Y/(0.000020261*
   (T+273.15)+0.994466)
34 PRINT USING;"NEW Y = ";Y:END
```

A FEW OP-AIDS FOR THE PC-1500/PC-2

A fellow by the name of Rich Hart, 2019 N. Kessler, Wichita, KS 67203, recently sent in a few routines to share with PCN readers. They come in pretty handy, especially if you are inclined towards "exploring" the Sharp PC-1500 or Radio Shack PC-2.

The first routine, shown in lines 1 — 3 of the accompanying listing, just combines the use of the STATUS 1 and STATUS 2 commands to automatically display the total number of bytes used for program storage. It uses the ARUN command to display this information each time the PC is turned on.

The first routine then calls part of the next routine to obtain the address range being utilized for program storage. This second routine comprises lines 10 — 50 of the accompanying listing. It converts the value in variable N to hexadecimal notation. It is particularly handy when examining the contents of memory locations using the PEEK instruction. Using the DEF J keystroke sequence, decimal values on the display are immediately converted to their hexadecimal equivalents. Note that this routine can be used as a subroutine by entering

it at line 12. However, when this is done, you must set up variable SF=1 as a "flag" so that the latter part of line 45 will be executed.

The third routine finds the absolute hexadecimal address of the first byte of code for a given BASIC program line number. It is useful in a number of situations, such as when you want to perform an ASCII dump of just a portion of a large BASIC program. Those of you puttering around with machine language or considering hybrid program (part BASIC, part machine code) will also find this utility quite useful. This routine occupies lines 200 - 230 in the listing.

It operates as follows: After inputting the statement line number that is to be located, the user presses DEF/K. Line 220 of the routine examines the first two bytes of a BASIC source line (which holds the line number) and converts it from its two-byte hexadecimal format so that it can be compared with the decimal value specified. If a match is found, then the current address is displayed. Otherwise, the routine takes the third byte as a displacement value to get to the start of the next BASIC line.

By the way, these routines were designed for use in PCs having an 8K memory module installed. If you have an unexpanded unit or a 4K module, then substitute the value &48C5 wherever &38C5 appears in the listing.

Program Op-Aids

```

1: ARUN : WAIT 200 30: R=N-16*Q
   : PRINT STATUS 40: B$=MID$ (A$, R+
   1; " BYTES NOW 1, 1)+B$: N=Q
   RESIDENT" 45: NEXT X: IF SF
2: N=(STATUS 2)-1 THEN RETURN
   : SF=1: GOSUB 12 50: WAIT : PRINT B$
3: WAIT : PRINT "P : END
   ROGRAMS FROM " 200: "K" INPUT "LINE
   ; "38C5 TO "; B$ NUMBER?", N
   : END 210: X=&38C5
10: "J"AREAD N: SF= 220: IF 256*(PEEK X
   0 )+PEEK (X+1)=N
12: A$="0123456789 PRINT X: END
   ABCDEF": B$="" 230: X=X+3+PEEK (X+
15: FOR X=1 TO 4 2): GOTO 220
20: Q=INT (N/16)

```

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January - December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

- ☐ 1982 Regular Subscriber (Issues 11 - 20). \$30.00 in U.S. (U.S. \$36.00 to Canada. U.S. \$45.00 elsewhere.)
- ☐ 1982/83 Subscriber (Issues 11 - 30). \$60.00 in U.S. (U.S. \$72.00 to Canada. U.S. \$90.00 elsewhere.)
- ☐ 1983 Regular Subscriber (Issues 21 - 30). \$36.00 in U.S. (U.S. \$42.00 to Canada. U.S. \$50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____
 Addr: _____
 City: _____ State: _____ Zip: _____
 MC/VISA #: _____ Expires: _____
 Signature: _____



P.O. Box 232, Seymour, CT 06483

FROM THE WATCH POCKET

My sources indicate that Hewlett-Packard is running a bit behind in getting the new HP-75 into the hands of users. The original announced date for initial deliveries was September 15th. One exasperated dealer, unable to get a single unit to date, while being circled by cash-in-hand buyers, quipped that, when it came to initial deliveries, HP stood for "Have Patience." So be it. At least they have a cadre of interested buyers standing around waiting for it!

Seems HP isn't the only firm running a little behind on anticipated delivery dates. I have yet to hear of any Sharp ROM modules being delivered in the U.S. I have also heard that the RS-232 interface may be slipped to December.

It also seems that Sharp does not plan to market some of its accessories for the PC-1500 in the U.S. At least, not at the present time. This goes, apparently, for the CE-153 Software Board (a matrix of 140 switches that can be used as an input device). I have also heard that Sharp now has a video display unit available for the PC-1500 in Japan, but has no plans for introducing such a product to the U.S. market.

There is little talk these days of a disk or other high speed mass memory device being planned for the PC-1500. Earlier, I had reports that such units were being studied. It seems to me though, that such devices should be possible. Maybe some independents will take a look at the potential for such accessories. I am, frankly, surprised that no one is yet producing any kind of peripheral device for the unit. Is this because Sharp has been very closed-mouthed about technical information?

The new on-line service being provided by Panasonic on the CompuServe network seems to be worthwhile and timely. I plan to keep an eye on it for awhile and will let you know how it develops. Remember, anyone on CompuServe can access this facility. There are programs on the bulletin that can be adapted to PCs other than Panasonic's HHCs.

Speaking of networks: PCN is on the Source (ST4322) as well as CompuServe (74445,1123). I currently check for mail on those networks a few times a week.

There is an outfit called Elek-Tek, Inc., 6557 N. Lincoln Avenue, Chicago, IL 60645, phone (800) 621-1269, advertising the PC-1211 for \$94 and the CE-122 Printer/Cassette interface for \$64! They also carry the PC-1500 as well as Hewlett-Packard equipment. You might want to check 'em out if you are in the market. Also, I continue to receive favorable reports on the speedy service and low prices provided by: Tam's, Inc., 14932 Garfield Avenue, Paramount, CA 90723, with an order phone number of (800) 421-5188, as well as by Atlantic NE Marketing, PO Box 291, Marblehead, MA 01945, phone (617) 639-0285.

By the way, if anyone is wondering what kind of program submission to PCN would really catch my eye, let me tell you: There is a very serious need at this point in the development of machine-language capabilities on the PC-1500/PC-2 for at least a rudimentary Monitor program. By that, I mean a program, using hexadecimal notation, that will enable a m.l. programmer to perform such basic tasks as examine specific memory locations, alter their contents if desired, perform block memory-to-memory transfers, allow at least the examination of the CPU registers (and possibly the alteration of some), permit the establishment of "breakpoints" in memory, and so forth. While such a program could readily be done in BASIC, a version of such a utility tool done in m.l. has obvious appeal. Everything that is needed to produce such a practical tool has been published in PCN during the past few months (particularly in the Special Issue). The existence of such a Monitor program would make it much easier for others to create hybrid or pure machine-language programs. It is virtually a necessity for going on to produce and check out a comprehensive assembler.

Of course, any practical embellishments you might want to add, such as decimal-to-hexadecimal conversion, would be "frosting on the cake" provided they were memory-efficient, etc. I would tend to think that a nice little Monitor with a few bells and whistles could be packed into 1.5 to 2K of memory.

I am pleased to report that a number of people are making progress finding key ROM entry points. Stay tuned! — Nat Wadsworth, Editor

POCKET COMPUTER

NEWSLETTER



© Copyright 1982 — Issue 20

December

EPSON AMERICA ANNOUNCES HX-20 PORTABLE COMPUTER

The company that has marketed the highly successful MX-series of low cost computer printers for several years has now announced a major entry in the burgeoning portable computer market. The new model is designated the HX-20. Its weight, including a built-in 24-column dot matrix microprinter is under four pounds. The unit also includes a full-sized keyboard containing 68 keys and a four-line by 20-character LCD (liquid-crystal display) that can also display simple graphics in a 120 by 32 dot matrix mode. Overall size of the unit is 11-3/8 by 8-1/2 by 1-3/4 inches. The area it occupies on a table is just about exactly that of an ordinary piece of writing paper.

Other features of the unit include: dual CMOS 8-bit type 6301 microprocessors operating at 614 KHz; 16K of RAM provided with

the basic unit (expandable to 32K); 32K of system ROM supplied as standard (expandable to 40K internally, 64K in an expanded mode); built-in timer/clock/calendar with separate battery backup; four octave, half-tone, programmable tone generator; RS-232C communications port at full or half duplex and 110 to 4800 baud rate; built-in serial interface; ability to interface with HP barcode reader (through a special connector); standard audio cassette interface; optional microcassette unit can be installed in the basic unit; rechargeable Nickel-Cadmium batteries can power the unit for up to 50 hours of completely portable operation and recharge within 8 hours.

The company is reportedly making initial deliveries to selected, authorized dealers. The list price for the basic unit is \$799.95. For additional information contact: Epson America, Inc., 3415 Kashiwa Street, Torrance, CA 90505. The phone number is (213) 539-9140.

Photo: Epson HX-20 Portable Computer



Another Personal Information™ product from SCELBI Publications.

A WEATHER FORECASTING PROGRAM

The accompanying program predicts the local, short term (12 -- 24 hours) weather for any region in the Northern Hemisphere above the Tropic of Cancer.

To use the program, obtain the barometric pressure reading (inches). Try to interpolate readings to the nearest hundredth of an inch. An example reading might be: 29.95.

You must observe barometer movements over several hours to determine the barometric tendency. For application of this program, the following definitions apply: The barometric tendency is moving fast if it is changing more than 0.05 inches per hour. It is slow if the change is between 0.02 and 0.05 inches per hour. It is considered steady if less than or equal to 0.02 inches per hour.

Enter wind direction as N, NE, E, SE, S, SW, W or NW depending on which direction the wind is blowing *from* (not the direction that it is going towards!).

The program is based on meteorological rules of thumb. To obtain highest possible accuracy, the program should be run at frequent intervals as atmospheric conditions change.

The listing shown is for a PC-2/PC-1500. The program will fit in a PC-1/PC-1211 but you will need to change the variable @\$(Z) in line 210 to read A\$(Z). This program submitted by: *Emerich Auersbacher, 41 King Street - Apt. 2, Belleville, NJ 07109.*

```

10: BEEP 1: PAUSE "
    WEATHER FORECA
    STER"
20: A$="S": B$="SW"
    : C$="W": D$="NW"
    : E$="N": F$="N
    E": G$="E": H$="
    SE"
30: INPUT "BAROMET
    ER (IN): "; P: IF
    P>30.2 LET S=0
40: IF P<30.2 LET S
    =1
50: IF P<30.1 LET S
    =2
60: IF P<30.0 LET S
    =3
70: IF P<=29.8 LET
    S=4
80: INPUT "(R)ISE,
    (F)ALL, (S)TEAD
    Y?": T$: X=0
90: IF T$="R" INPUT
    "RISING (F)AST
    , (S)LOW?": R$:
    IF R$="F" LET X
    =1: GOTO 150
100: IF T$="R" LET X
    =2: GOTO 150
110: IF T$="F" INPUT
    "FALLING (F)AS
    T, (S)LOW?": R$:
    IF R$="F" LET X
    =3: GOTO 150
120: IF T$="F" LET X
    =4: GOTO 150
130: IF T$="S" GOTO
    150
140: GOTO 80
150: INPUT "WIND FR
    OM THE: "; W$:
    GOSUB 200
160: IF Y=0 BEEP 3:
    PAUSE "ERROR":
    GOTO 150
170: GOTO 300
200: Y=0: FOR Z=1 TO
    8
210: IF W$=Q$(Z) LET
    W=Z: Z=8: Y=1:
    GOTO 220
220: NEXT Z: RETURN
300: T=600: Y=ABS (W
    -3)<2
310: IF Y*(S<3)*(X=
    0) LET Z=1: GOTO
    T
320: IF Y*(S=1)*(X=
    1) LET Z=6: GOTO
    T
330: IF Y*(S=0)*(X=
    4) LET Z=3: GOTO
    T
340: Y=(W=1)+(W=8)
350: IF Y*(S=1)*(X=
    4) LET Z=4: GOTO
    T
360: IF Y*(S=1)*(X=
    3) LET Z=5: GOTO
    T
370: Y=ABS (W-7)<2
380: IF Y*(S=1)*(X=
    4) LET Z=4: GOTO
    T
390: IF Y*(S=1)*(X=
    3) LET Z=5: GOTO
    T
400: IF Y*(S>2)*(X=

```

PRODUCT REVIEW

99 TIPS & TRICKS FOR THE NEW POCKET COMPUTERS

Author: *Jim Cole*

Publisher: *ARCsoft Publishers, POB 132, Woodboro, MD 21798*

List Price: \$7.95 (add \$1 for s/h charges on mail orders).

Availability: *Directly from publisher, computer & book stores.*

Reviewer: *Nat Wadsworth, Editor.*

At first I thought "maybe old Jim just re-arranged some of his previous routines from his earlier books?" But no (how could I think such a thing?), Jim has indeed come up with a genuine potpourri of new ideas and original programs for the Radio Shack PC-2/Sharp PC-1500 Pocket Computers!

These routines range from the almost ridiculously simple (such as laying down a string of "#" signs to portray railroad tracks!) all the way up to complete programs that allow you to do such things as simulate a RPN calculator, make an automatic Morse Code practice oscillator (really neat!) and convert world currencies at the touch of a few buttons. There are also quite a few graphics oriented routines, including one that draws a pair of little dice on the LCD and makes them appear just as though you had rolled 'em!

While this 128 page booklet is intended primarily for beginners, it has enough interesting ideas so that even those with plenty of experience might pick up a trick or two. The price makes it a bargain.

```

4) LET Z=6: GOTO
    T
410: IF Y*(S>2)*(X=
    3) LET Z=7: GOTO
    T
420: Y=(W=6)+(W=7)
430: IF Y*(S<2)*(X=
    4) LET Z=8: GOTO
    T
440: IF Y*(S<2)*(X=
    3) LET Z=9: GOTO
    T
450: IF ((W=1)+(W=2
    ))*(S>2)*(X=2)
    LET Z=10: GOTO
    T
460: IF ((W<2)+(W>6
    ))*(S=4)*(X=3)
    LET Z=11: GOTO
    T
470: IF (ABS (W-6)<
    2)*(S=4)*(X=3)
    LET Z=12: GOTO
    T
480: IF (ABS (W-3)<
    2)*(S=4)*(X=1)
    LET Z=13: GOTO
    T
490: IF (X<3)*(ABS
    (W-5)<3) LET Z=
    2: GOTO T
500: Z=1: IF Z=3 LET
    Z=8
600: BEEP 3: PAUSE "
    FORECAST -->"
    : GOSUB T+10*Z:
    GOTO 30
610: PRINT "FAIR, L
    ITTLE CHANGE."
    : RETURN
620: PRINT "FAIR, A
    LITTLE COOLER
    .": RETURN
630: PRINT "FAIR, A
    LITTLE WARMER
    .": RETURN
640: PRINT "RAIN WI
    THIN 24 HRS.":
    RETURN
650: PRINT "WINDY,
    RAIN BY 12 HRS
    .": RETURN
660: PRINT "INCREAS
    ING CLOUDS, RA
    IN.": RETURN
670: PRINT "RAIN AN
    D HIGH WINDS."
    : RETURN
680: PRINT "UNSETTL
    ED, CHANCE RAI
    N.": RETURN
690: PRINT "RAIN/SN
    OW, WINDY.":
    RETURN
700: PRINT "CLEARIN
    G, BECOMING FA
    IR.": RETURN
710: PRINT "SEVERE
    STORM IMMINENT
    .": RETURN
720: PRINT "HEAVY R
    AIN/SNOW, WIND
    S.": RET
730: PRINT "CLEARIN
    G AND COOLER."
    : RETURN
STATUS 1 1608

```


GETTING STARTED WITH MACHINE LANGUAGE

Perhaps some of you who are wondering what all the fuss over the machine codes for the PC-1500/PC-2 is about, might be able to satisfy your curiosities somewhat by trying your hand at a few simple programs written in machine language.

Let's start with a program so simple that it is of strictly of educational value. The program to be illustrated first will merely perform the equivalent of the BASIC statement $C = A + B$, and that is all! In fact, it will use routines that are already stored in ROM for the purpose of performing floating-point arithmetic.

Three things need to be done: 1) Write the program; 2) Put the program into memory; and 3) Execute the machine-language program.

We will begin by writing the program. First we need to transfer the contents of the variable named A into the floating-point accumulator, which is CPU register R0. The ROM routine for doing this requires that CPU register YH contain the code &7A. Furthermore, CPU register X must contain the address at which variable A is stored. Variable A is stored starting at address &7900. Here is what the start of the program looks like:

Opcodes	Mnemonics	Comments
58 7A	LDYH #7A	Contents of variable A
48 79	LDXH #79	placed into register R0
4A 00	LDXL #00	
BE F7 3F	JSR F73F	

Next, we will move the contents of register R0 into register R2. This needs to be done so that variable B may be placed into R0;

E6	CALL E6	Stores R0 into R2
----	---------	-------------------

Remember that the opcode E6 used alone is equivalent to the instruction CALL E6. (We could have used opcodes CD E6 for this directive.)

Now we need to put variable B into R0. (Since the previous instruction has left &7A in CPU register YH, we do not need to include the LDYH #7A directive this time.)

Opcodes	Mnemonics	Comments
48 79	LDXH #79	Contents of variable B
4A 08	LDXL #08	stored into R0
BE F7 3F	JSR F73F	

The multiplication itself can be performed by simply using the ROM routine for floating-point multiplication:

CD 7E	CALL 7E	Replace R0 with $R0 * R2$
-------	---------	---------------------------

Finally, the calculated product must be placed into variable C, whose beginning address is &7910:

58 79	LDYH #79	Contents of R0 put into
5A 10	LDYL #10	variable C
BE F7 11	JSR F711	

We must end the machine-language program with:

9A	RTS	Return from subroutine (to BASIC)
----	-----	-----------------------------------

That is the program! Now, how do we load it into an appropriate section of memory?

First, we need to reserve some space for the machine-language routine. We can do this by executing NEW &3900. (If you are not using an 8K CE-155 module then use NEW &4100 instead.) This action protects the memory area from &38C5 to &3900 (or &40C5 to &4100) from being used by a BASIC program.

The opcodes derived above must now be put into memory using a POKE statement. This needs to be done just once. It may be accomplished either in manual mode or in a program line. The following statements will perform this operation:

POKE &38D0, &58, &7A, &48, &79, &4A, 0, &BE, &F7, &3F, &E6, &48, &79, &4A, 8, &BE, &F7
POKE &38E0, &3F, &CD, &7E, &58, &79, &5A, &10, &BE, &F7, &11, &9A

If you are not using an 8K expansion module, then begin the two POKE sequences at &40D0 and &40E0 respectively.

Now you are in a position to test the program. Put a couple of your favorite numbers into variables A and B. Then, either in manual mode or in a program line, execute the BASIC statement: CALL &38D0. (Use CALL &40D0 if you do not have an 8K module). This will cause the machine language program that you stored at &38D0 (or &40D0) to be

executed. To see whether things worked as expected, display variable C. It should contain the product of A and B.

If errors are made in a machine-language routine, there are no "error messages" provided by the PC to assist the programmer. Programming errors in machine-language frequently lead to a "crash," disabling the keyboard or disrupting operations in other ways. In such cases, you may be forced to use the "All Reset" button on the back of the PC, then start all over again! Anyone who takes up experimenting with machine-language programming can expect this to happen often!

A More Useful Machine-Language Routine

The factorial function is not a part of the Sharp PC-1500/Radio Shack PC-2's repertoire. Of course, it can be programmed in BASIC and one efficient algorithm that will calculate $N!$ (factorial) when N is an integer in the range 1 to 69 may be programmed as follows:

```
10 F=1: FOR J=N TO 2 STEP -1: F=F*J: NEXT J
```

The result is stored in variable F. The length of time required to perform this calculation using the BASIC routine depends on the value of N. When N is specified as 69, it takes approximately 2.4 seconds.

A machine-language program that accomplishes the same result is shown below. It will reduce the time required to calculate $69!$ to just 0.6 seconds. If this increase in speed does not seem that impressive, it should be remembered that there is a lot of arithmetic being done here and that the same floating-point routines are being used whether BASIC or machine language is used. All of the savings in time is being achieved from the simple elimination of the need to interpret the BASIC statement line!

There is even another advantage to using the machine-language routine for this type of calculation. In the BASIC version, each intermediate result is rounded to ten digits. This roundoff accumulates. The machine-language version maintains twelve digits throughout the computation, rounding to ten digits only after the final result is obtained. As an example, the calculated result for $56!$ errs by 8 in the tenth digit when using the BASIC program, but the machine-language program is correct to all ten digit positions.

The routine shown is accessed by a CALL instruction from BASIC in which the input, N, is passed to the machine-language program. This input is placed into CPU register X by the call routine. However, when this is done it is in hexadecimal format. It must be converted to decimal before the floating-point arithmetic routines may be applied.

Fortunately, a ML program for making this conversion already exists in ROM. The base page addressed subroutine 10 will store the BCD equivalent of the contents of register U in R0. (This subroutine requires the passing of one byte. The desired effect can be achieved by using &80.)

Here is the program. The memory addresses assume storage in a PC having an 8K RAM module.

Address	Opcodes	Mnemonics	Comments
&38D0	FD 6A	STX U	X stored into U
&38D2	FD A8	PSH U	Save in stack
&38D4	CD 10 80	CALL 10 (PASS 80)	Store into R0 (in BCD)
&38D7	E6	CALL E6	R0 into R2
&38D8	FD 2A	POP U	Get U from stack
&38DA	62	DEUL	Decrement UL
&38DB	6E 02	CPUL #02	Compare UL to 2
&38DD	81 09	FBNC #09	If $UL < 2$, branch to &38E8
&38DF	FD A8	PSH U	Save U in stack
&38E1	CD 10 80	CALL 10 (PASS 80)	U stored into R0 (in BCD)
&38E4	CD 7E	CALL 7E	R0 replaced by $R0 * R2$
&38E6	9E 11	RB #11	Branch to &38D7
&38E8	BE F9 32	JSR F932	Round R0 to ten digits
&38EB	58 79	LDYH #79	R0 is copied
&38ED	5A 28	LDYL #28	into variable F,
&38EF	BE F7 11	JSR F711	which begins at &7928
&38F2	9A	RTS	Return to BASIC

The program may be loaded into memory using the POKE statements:

POKE &38D0, &FD, &6A, &FD, &A8, &CD, &10, &80, &E6, &FD,

&2A, &62, &6E, 2, &81, 9, &FD
 POKE &38E0, &A8, &CD, &10, &80, &CD, &7E, &9E, &11, &BE,
 &F9, &32, &5B, &79, &5A, &28, &BE
 POKE &38F0, &F7, &11, &9A

Execution of CALL &38D0, N will place the value of N! into the variable named F. Of course, the number (1 to 69) whose factorial is to be calculated must be assigned to the variable N beforehand.

As in the previous example, if you do not have an 8K memory expansion module, the addresses in the POKE statements should be changed to &40D0, &40E0 and &40F0 with CALL &40D0 being used to execute the routine.

One More Example

This final example program makes use of the floating-point stack. This stack should not be confused with the stack used by machine-language PUSH and POP instructions.

The ML (machine-language) instructions PSHA, PSIH, etc., will save one or two bytes obtained from a CPU register in a stack located in RAM at addresses &784F and below. The instructions POPA, POPX, etc., remove one or two bytes at a time from this stack and store them back into the corresponding CPU register(s). This same stack is used to save machine-language subroutine return addresses.

A completely separate stack, similar in principle, is created by ROM routines in the PC. This stack is used to save an 8-byte block of data, obtained from floating-point register R0. This data can later be popped back into R0. Just as in the case of the CPU stack, this software-created stack operates on a LIFO (last-in, first-out) basis. The floating-point stack is located between &7A38 and &7AFF. It is also used to save BASIC subroutine return addresses and in keeping track of FOR/NEXT operations.

The program shown next will perform calculations equivalent to the BASIC statement: $C = \sqrt{(A^2 + B^2)}$.

Address	Opcodes	Mnemonics	Comments
&38D0	58 7A	LDYH #7A	Contents of
&38D2	48 79	LDXH #79	variable A
&38D4	4A 00	LDXL #00	stored
&38D6	BE F7 3F	JSR F73F	into R0
&38D9	BE F0 19	JSR F019	Contents of R0 squared
&38DC	BE DB F5	JSR DBF5	R0 pushed onto F-P stack
&38E0	48 79	LDXH #79	Contents of
&38E2	4A 08	LDXL #08	variable B
&38E4	BE F7 3F	JSR F73F	stored into R0
&38E7	BE F0 19	JSR F019	Contents of R0 squared
&38EA	E6	CALL E6	Store R0 into R2
&38EB	CD 30	CALL 30	Pop from F-P stack into R0
&38ED	F0	CALL F0	Add R2 to R0
&38EF	BE F0 E9	JSR F0E9	Replace R0 by its square root
&38F2	BE F9 32	JSR F932	Round R0 to ten digits
&38F5	58 79	LDYH #79	Store contents
&38F7	5A 10	LDYL #10	of R0 into
&38F9	BE F7 11	JSR F711	variable C
&38FC	9A	RTS	Return to BASIC

As in the previous examples, use POKE statements to load the opcodes into memory. Assign values to variables A and B then execute the routine by CALL &38D0 (or CALL &40D0 if you do not have an 8K module). The variable C will contain the computed result.

These examples provide a rough idea of how floating-point routines provided as part of the ROM can be incorporated into ML programs. Of course, there are many applications of machine language that would not involve the use of these floating-point ROM algorithms. Remember, essentially anything that can be done in BASIC can be done in ML, but the process can be a lot more complicated. The payback is in operating speed and increased flexibility.

[Sleuth Norlin Rober provides a wealth of information regarding the location of ROM floating-point routines in the following tables. Just at press time he notified PCN that he had located one more key F-P entry point: Calling &EFB6 will perform floating-point subtraction between F-P registers R0 and R2 (as R0 - R2) with the result left in R0. That provides a pretty complete F-P set of ROM routines for you to work with. Hats off to Norlin once again! - N.W.]

FLOATING-POINT ARITHMETIC IN THE PC-1500

Floating-point arithmetic is performed using 12-digit binary-coded-decimal mantissas, with exponents of 10 stored in two's-complement binary. Each floating-point number is stored in 8 bytes, as follows:

BYTE	CONTENTS (NORMALIZED FORM)
0	Exponent, in two's-complement binary
1	Sign of mantissa; 0 for positive, &20 for negative
2	1st and 2nd digits of mantissa, in BCD
3	3rd and 4th digits
4	5th and 6th digits
5	7th and 8th digits
6	9th and 10th digits
7	11th and 12th digits

BASIC routines round calculated results to ten-digit mantissas prior to display, printing, or storage into a variable.

Computations are carried out using seven floating-point registers, each comprised of 8 bytes, located in RAM at the following addresses:

R0:	7A00 to 7A07 (Floating-point Accumulator)
R1:	7A08 to 7A0F
R2:	7A10 to 7A17
R3:	7A18 to 7A1F
R4:	7A20 to 7A27
R5:	7A28 to 7A2F
R6:	7A30 to 7A37

In ROM routines, CPU Registers X and Y act as pointers to the addresses used as floating-point registers.

In the major routines, calculated results are placed in R0. In binary operations, the operands taken as input arguments are obtained from R0 and R2; operations using a single operand use the contents of R0 as input.

MAJOR SUBROUTINES INVOLVING FLOATING-POINT ARITHMETIC

The ROM subroutines tabulated below are useful in machine-language programming where floating-point arithmetic is used.

Where applicable, base-page call numbers are tabulated also.

CALL	ADDR	RESULT IN R0	REGISTERS USED	NOTES
FE	EFBA	R0 + R2	R0, R2	
7E	F01A	R0 - R2	R0 to R5	
5B	F084	R0 / R2	R0 to R4	
6E	F019	R0 * R2	R0 to R5	
	F080	1 / R0	R0 to R4	XH and YH must contain &7A
	F0F9	SQR R0	R0 to R2	
	F161	LN R0	R0 to R5	
	F165	LOG R0	R0 to R5	
	F1C3	EXP R0	R0 to R5	
	F1D4	10 ^ R0	R0 to R5	
	F391	COS R0	R0 to R6	
	F39E	TAN R0	R0 to R5	
	F3A2	SIN R0	R0 to R6	
	F492	ACS R0	R0 to R6	
	F496	ATN R0	R0 to R5	Flag C must be clear
	F49A	ASN R0	R0 to R6	
	F531	DSG R0	R0 to R4, R6	
	F564	DMS R0	R0 to R6	
	F597	ABS R0	R0	
	F590	SGN R0	R0	
	F5B5	#	R0, R2	# placed in R2 also
	F5B7	INT R0	R0, R2	R2 = fractional part (not normalized)
	F5D0	RND R0	R0 to R6	
2C	F757	zero	R0	
	F89C	R0 ^ R2	R0 to R6	Calculated using logarithms
	F932	R0, rounded	R0	Rounded to ten significant digits

Trigonometric functions and their inverses are calculated in degrees, radians, or grads, according to the mode set.

Except as noted, all results are normalized. A result less in absolute value than 1 E-99 is replaced by zero.

Illegal operations and the subroutine with Flag C set; CPU register YH will contain the error code, as listed below:

UH = &25:	ERRR 37, overflow
UH = &26:	ERRR 38, division by zero
UH = &27:	ERRR 39, illegal function argument

ROUTINES FOR TRANSFERRING FLOATING-POINT DATA

A floating-point stack shares RAM addresses 7A38 to 7AFF with the BASIC loop and subroutine stacks.

The ROM subroutines listed below include data transfers involving the floating-point registers, floating-point stack, and numeric variables.

CALL	ADDR	EFFECT OF SUBROUTINE	NOTES
80	F707	R0 stored into R6	
8E	F70D	R0 stored into R2	
	F737	R6 stored into R0	
56	F730	R2 stored into R0	XH and YH must contain &7A
64	F7B5	R0 and R6 exchanged	XH and YH must contain &7A
66	F7B9	R0 and R2 exchanged	XH and YH must contain &7A
	DBF5	Push R0 onto FP stack	If capacity exceeded, Flag C set and UH = &26.
30	DC16	Pop R0 from FP stack	
	F711	R0 stored into variable	XH must contain &7A, Y the variable address
	F73F	Variable stored into R0	YH must contain &7A, X the variable address

ADDRESSES OF FIXED VARIABLES A TO Z

The addresses given below are those of the initial bytes of the respective variables. They are the addresses that must be in CPU register X or Y, in the above routines for data transfer between R0 and variables.

A	7900	D	7930	N	7968	T	7998
B	7908	E	7938	O	7970	U	79A0
C	7910	I	7940	P	7978	V	79A8
D	7918	J	7948	Q	7980	W	79B0
E	7920	K	7950	R	7988	X	79B8
F	7928	L	7958	S	7990	Y	79C0
		M	7960			Z	79C8

NORMALIZING ROUTINES

The subroutines for transforming calculated results into normalized form are executed by the major floating-point routines.

CALL	ADDR	EFFECT	NOTES
58	F65D	R0 normalized	XH and YH must contain 47A
59	F661	R0 normalized and given a positive sign	XH and YH must contain 47A
52	F663	R0 normalized; contents of A used as sign	XH and YH must contain 47A

SHORT SUBROUTINES USED WITH FLOATING-POINT ARITHMETIC

In certain subroutines (as indicated) CPU Registers XH and/or YH must contain 7A prior to execution of the subroutine.

Unless otherwise stated, the initial byte of a floating-point register, representing the exponent, is ignored.

CALL	ADDR	7A REQ'D	EFFECT OF SUBROUTINE
7C	F63C		11th and 12th digits of R0 cleared
6C	F616	XH, YH	Sign of R2 and sign of product R0*R2 PUSHed; both signs cleared
	F618	XH	Sign of R0 into A; sign cleared
80	F701	XH, YH	R2 copied into R6, including exponent
26	F707		R0 copied into R6, including exponent
68	F700		R0 copied into R2, including exponent
	F715	XH, YH	R6 copied into R2, including exponent
	F718	XH, YH	R2 copied into R0
	F71F	XH, YH	R1 copied into R0
	F725	XH, YH	R2 copied into R1
82	F729	XH, YH	R0 copied into R2
78	F72F	XH, YH	R0 copied into R1
	F733		Addresses X to X-6 copied into Y to Y-6
	F737		R6 copied into R0, including exponent
56	F73D	XH, YH	R2 copied into R0, including exponent
70	F747	XH	R2 cleared
	F748	XH	R1 cleared
	F74F	XH	R5 cleared
	F753	XH	R2 cleared, including exponent
9C	F757		R0 cleared, including exponent
	F758		Addresses X to X-7 cleared
76	F75F	XH	R0 cleared
	F761		Addresses X to X-6 cleared
8A	F763		Addresses X to X-6 cleared
	F769	XH	Digits of R5 shifted right
	F76D	XH	Digits of R1 shifted right
	F771	XH	Digits of R2 shifted right
74	F775	XH	Digits of R0 shifted right
	F777		Digits of addresses X to X-6 shifted right
	F780	XH	Bytes of R1 shifted right
	F78C	XH	Digits of bytes 2 to 6 of R6 shifted left
	F794	XH	Digits of R5 shifted left
	F798	XH	Digits of R1 shifted left
EA	F79C	XH	Digits of R0 shifted left
52	F7A7	YH	Random number copied into R0
54	F7B8		XH and YH loaded with 47A
64	F7B5	XH, YH	R0 and R6 are exchanged, including exponent
66	F7B9	XH, YH	R0 and R2 are exchanged, including exponent
EE	F7C8	XH, YH	R3 is added to R0
72	F7CC	XH, YH	R2 is added to R6
	F7D2	XH, YH	Register that ends with address Y is added to R0
7A	F7D9	XH, YH	R4 is subtracted from R0
	F7DD	XH, YH	R2 is subtracted from R0
	F7DF	XH	Register that ends with address Y is subtracted from R0
	F7E1	XH	Addresses Y-6 to Y subtracted from addresses X-6 to X
	F7E8	XH	R0 is replaced by its ten's complement

SHORT SUBROUTINES USED WITH FLOATING-POINT ARITHMETIC (CONT'D)

CALL	ADDR	7A REQ'D	EFFECT OF SUBROUTINE
F7F7			Digits of Register ending with X shifted right A times
F805			Bytes of Register ending with X shifted right A times
F81C		XH, YH	2*R2 into R3, 4*R2 into R4
F820		XH, YH	2*R2 into R3, 4*R2 into R4, 8*R2 into R5
F83C		XH, YH	LDA 7A08. If A=0, digits of R1 are shifted right A-1 times, and R1 is added to R2
F84D		XH, YH	LDA 7A08. If A=0, bytes of R1 are shifted right A-1 times, and R1 is added to R2
F85A		YH	IF GRAD Mode, R2 = 0.9; IF RADIANT, R2 = 57.2957795131
F866		YH	R2 set equal to 57.2957795131 ($\pi/180$)
F875		YH	R2 set equal to 3.14159265359 (π)
F87B		YH	R2 set equal to 0.434294481903 (LOG e)
F877		YH	R2 set equal to 0.9
F883		YH	R2 set equal to 90
F887		YH	R2 set equal to 180
F88B		YH	R2 set equal to 0.6
6A	F887	YH	R2 set equal to 1

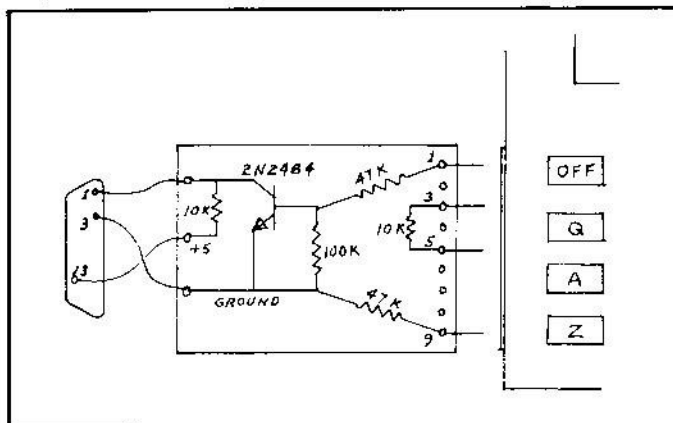
Note: In the last nine subroutines listed above, XH is loaded with 47A, and Flag C is set. The constants stored into R2 are obtained from a ROM table located F8B9 to F89F.

THE RADIO SHACK PC-1 AS A DATA COLLECTING TERMINAL

One logical use for the Radio Shack TRS-80 PC-1 and Sharp PC-1211 is as a portable data entry device. The units have a keyboard, a readout, a reasonable amount of memory and the ability to retain information with the power off. Unfortunately, there is no built-in provision to transfer data to a larger computer!

This article describes a method of transferring data directly from a PC through a simple interface to a larger system. An accompanying drawing shows an interface that uses a single transistor. It can be connected between a PC-1/PC-1211 and a parallel input port of a larger computer that uses a CPU such as an 8080. The drawing illustrates the electronic schematic as well as the actual physical layout I have utilized. The pins that plug into the PC-1 are made out of number 24 tinned

Diagram PC Data Transfer Interface



Program PC Transfer Routines in 8080 Assembler Code

```
00 00 3E 08 MVI A,08 ;END OF BUFFER
00 02 21 00 01 LXI H,01 00 ;START OF BUFFER
00 05 36 00 CLEAR, MVI M,00 ;CLEAR MEM LOC
00 07 23 INX H ;INCR HL PAIR
00 08 8C CMP H ;COMPARE H WITH ACC
00 09 C2 D5 00 JNZ CLEAH ;CLEAR NEXT BYTE
00 0C 26 01 MVI H,01 ;RESET BUFFER
00 0E 00 NOP
00 0F 00 NOP
00 10 16 00 CHAR, MVI D,00 ;CLEAR REG D
00 12 CD 96 00 CALL START ;LOOK FOR START BIT
00 15 AF XRA A ;CLEAR ACC
00 16 B8 CMP B ;COMPARE WITH B
00 17 CA 04 2A JZ 2A04 ;BACK TO BASIC
00 1A CD 5D 00 CALL NIBL ;GET FIRST NIBBLE
00 1D 50 MOV D, B ;STORE IN D
00 1E CD 36 00 CALL START ;LOOK FOR START BIT
00 21 CD 5D 00 CALL NIBL ;GET SECOND NIBBLE
00 24 78 MOV A, B ;MOVE NIBBLE TO ACC
00 25 0F RRC ;SHIFT TO FAR RIGHT
00 26 0F RRC
00 27 0F RRC
00 28 0F RRC
00 29 E6 0F ANI 0F ;NEED 4 LSB ONLY
00 2B 82 ADD D ;COMBINE BOTH NIBBLES
00 2C 77 MOV M, A ;STORE CHAR IN BUFFER
00 2D 23 INX H ;INCR HL PAIR
00 2E C3 10 00 JMP CHAR ;GET NEXT CHAR

00 36 0608 START, MVI B, 08 ;TIME CONSTANT
00 3B 0E FF MVI C, FF ;TIME CONSTANT
00 3A AF XRA A ;CLEAR ACC
00 3B DB 00 LOOK, IN 0 ;READ PAR INPUT PORT
00 3D A7 ANA A ;SET FLAGS
00 3E FA 4B 00 JM TIMER ;IF NO START BIT
00 41 DB 00 IN 0 ;CHECK FOR TRUE BIT
00 43 A7 ANA A ;SET FLAGS
00 44 FA 4B 00 JM TIMER ;IF NO START BIT
00 47 CD 82 00 CALL DELAY ;HALF BIT DELAY
00 4A C9 RET ;BACK TO MAIN PROGRAM

00 4B 1D TIMER, DCR E ;KILL TIME
00 4C C2 3B 00 JNZ LOOK ;LOOK AGAIN
00 4F 0D DCR C ;KILL TIME
00 50 C2 3B 00 JNZ LOOK ;LOOK AGAIN
00 53 05 DCR B ;KILL TIME
00 54 C2 3B 00 JNZ LOOK ;LOOK AGAIN
00 57 C9 RET ;EXIT IF TIMED OUT

00 5D 06 00 NIBL, MVI B,00 ;CLEAR REG B
00 5F 0E 04 MVI C,04 ;SET BIT COUNTER
00 61 CD 82 00 BIT, CALL DELAY ;SKIP FULL BIT
00 64 CD 82 00 CALL DELAY
00 67 AF XRA A ;CLEAR ACC
00 68 DB 00 IN 0 ;READ PAR PORT
00 6A 00 NOP
00 6B E6 80 ANI 80 ;USE BIT 7 ONLY
00 6D 80 ADD B ;ADD BITS IN REG B
00 6E 0D DCR C ;DECR BIT COUNTER
00 6F CA 77 00 JZ END ;TEST FOR 4 BITS
00 72 0F RRC ;SHIFT TO RIGHT
00 73 47 MOV B, A ;STORE IN REG B
00 74 C3 61 00 JMP BIT ;GET NEXT BIT
00 77 47 END, MOV B, A ;STORE IN REG B
00 78 CD 82 00 CALL DELAY ;MOVE PAST LAST BIT
00 7B CD 82 00 CALL DELAY
00 7E C9 RET ;BACK TO MAIN PROGRAM

00 82 1F 9C DELAY, MVI E,9C ;NO. OF LOOPS FOR HALF BIT
00 84 1D DEL, DCR E ;KILL TIME
00 85 1C INC E ;KILL TIME
00 86 1D DCR E ;KILL TIME
00 87 08 RZ ;END OF DELAY
00 88 C3 84 00 JMP DEL ;DELAY ANOTHER LOOP
```

LIST command is then used when transferring the data to my desktop computer. I restrict the use of line numbers to the range 10 to 99 to provide a constant line format. A decimal point is used as a separator between a line number and data and between fields of data.

Other ways to enter data are as PC BASIC strings. However, this method requires a short program in PC BASIC. Also, PC-1/PC-1211 strings are limited to seven characters. This complicates the program if several strings must be used for each record.

Data can also be stored as BASIC variables. This, too, requires a program. One advantage of this method is that the data can be pre-processed, if desired, with only the results stored for later transfer to the main computer. A disadvantage is that variables are transferred in the same format as they normally appear on the PC-1 display. Therefore a considerable number of space characters may also be processed.

If lines of data are limited to 20 characters each, the PC-1/PC-1211

will hold about 70 lines.

After the information has been collected and keyed into the PC, the procedure for passing data to a larger computer is as follows: Connect the pocket computer and interface to the larger system. Power up the large system and load the required programs. Enter the necessary commands to start the transferring program *except for the last key to be pressed*. The PC is then powered up by pressing the ON key twice. It is then placed in the RUN or LIST configuration, as required, except for the final press of the ENTER key. After this, press the last key needed to start the program in the large computer and then immediately press the ENTER key on the PC. When the data has been processed, the PC will display the > prompt. (The reason for starting both computers at the same time is so that the large unit will receive a start bit within five seconds. If it does not, it will jump back to its BASIC executive.)

After the data has been transferred to my Horizon, I use the accompanying program written in NorthStar BASIC to convert each line of data in the input buffer to string format. It is necessary to add other statements to this routine (after line 1190), if you want to move the strings to a file, convert them to numeric variables, etc. The function EXAM in lines 110 and 1820 is similar to PEEK in other BASICs. The routine could also be used to transfer PC BASIC programs. However, I have not completed the routine for such use. If desired, all of the PC BASIC tokens could be converted in statements like those beginning at line 300.

This method for collecting data with a pocket computer and transferring them to the Horizon for further processing has been a great time saver. It is often faster to enter numeric data in the field directly on the PC's keyboard rather than use pencil and paper. The automation also means fewer errors. Perhaps the greatest advantage of the system is the fact that the information collected is immediately available for further processing. Final results of a research project can be quickly and easily ascertained using the powers of the desktop computer.

Thanks go to: S. J. Toy, Route 3, Box 73, Chico, CA 95926, for this informative description of the PC as a data input terminal.

Program Converting Raw Data to Strings

```
40 DIM T$(80),A$(1),B$(80),D$(80)
45 DIM M$(80),L$(80),P$(80)
78 P1=0:REM OUTPUT TO CRT
79 REM ***** CONVERT DATA TO STRINGS
80 M$="E.9876543210 /°-+*[]{}"
85 L$="ZYXWVUTSRQPONMLKJINGFEDCBA"
95 P1$="":$X$!!"
100 A1=0255:REM STARTING ADDR OF BUFFER
102 D$="":REM INITIALIZE FOR CONCATENATION
105 A1=A1+1
110 C1=EXAM(A1):REM GET CHARACTER
115 IF C1=141 THEN 105:REM IGNORE LINE START CHAR FOR LIST
120 IF C1=142 THEN 105:REM IGNORE LINE START CHAR FOR PRINT
130 IF C1=0 THEN 1100:REM LINE END CHAR
140 IF C1[76 AND C1]47 THEN 185:REM MATH SYMBOLS AND NUMBERS
150 IF C1[107 AND C1]80 THEN 195:REM LETTERS
160 IF C1[30 AND C1]18 THEN 205:REM PUNCTUATION MARKS
162 IF C1=18 THEN 175:REM QUOTATION MARK
165 IF C1=17 THEN 210:REM SPACE CHAR
170 IF C1[106 THEN 300 ELSE 900:REM BASIC TOKENS
175 T1$=CHR$(34):GOTO 1000:REM QUOTATION MARK
185 T1$=M$(76-C1,76-C1):GOTO 1000
195 T1$=L$(107-C1,107-C1):GOTO 1000
205 T1$=P$(30-C1,30-C1):GOTO 1000
210 T1$=" ":GOTO 1000:REM SPACE CHAR
300 IF C1=193 THEN T1$=" PRINT "ELSE 310:GOTO 1000
310 IF C1=194 THEN T1$=" INPUT "ELSE 900:GOTO 1000
315 REM ADD OTHER KEYWORDS OF PC BASIC IN SAME FORMAT
900 T1$=" ":GOTO 1000:REM PRINT SPACE IF NO CONV CHAR
1000 D$=D$+T1$:GOTO 105:REM ADD CHAR TO STRING
1100 REM STRING CONVERSION COMPLETED
1190 IF P1,D$:REM PRINT THE CONVERTED STRING
1810 A1=A1+1:REM INCR CHAR LOCATION
1820 C1=EXAM(A1):REM GET NEXT CHAR
1830 IF C1[141 OR C1]142 THEN 5000:REM TEST FOR LINE START CHAR
1900 D$=B$:REM CLEAR DATA STRING
1910 D$="":REM INITIALIZE DATA STRING
2000 GOTO 105:REM PROCESS ANOTHER LINE
5000 I:!" DATA ALL PROCESSED":!
```


FROM THE WATCH POCKET

A number of readers have asked me to comment further on the HP-75C that I reviewed several months ago. There is insufficient space for extensive comment in this issue, but if interest continues I shall plan a column later. Let me just say that after several hundred operating hours with the HP-75C, my respect and admiration for the machine continues to grow. It is a very pleasurable machine to utilize. At this point, I know that the feature I enjoy the most is its file handling capabilities. That is, the ability to instantly switch between and access many different files within its memory. It is a powerful unit and I continue to believe that it has set a standard against which other entries to the field will have to be compared. Yes, it is a relatively expensive unit. The challenge for others is whether they can begin to match it and cause a lowering of its price? In summary, at this point, I am far more impressed with, say, HP's entry to the portable computing market as evidenced by their HP-75C, than I was a few years ago when they entered the personal computing market with their HP-85. They could, in fact, improve the HP-85 by adding some of the capabilities of the 75!

The Year in Review

There was nowhere near the momentum or sales that could well have been expected in the pocket computer market. Here are my year-end "guess-estimates" of U.S. sales for some of the leading PCs.

Sharp PC-1211/Radio Shack PC-1 (combined) 1982 sales of 100,000. Down from perhaps 175,000 in 1981. This model is rapidly being outdated, they are being discounted heavily by just about everyone except Radio Shack. I doubt that you will be able to even buy one by the end of 1983.

Sharp PC-1500/Radio Shack PC-2 does not seem to be as doing as well as might have been expected. Perhaps 40,000 - 50,000 units sold so far. Many reasons, in my opinion, for the relative softness: Initial poor marketing, followed by better (but too little and too late). A person who views the powerful one-minute television ads for a Timex-Sinclair at \$100.00 has no comprehension of how a PC-1500/PC-2 compares when observing a single-page magazine ad. The essential key asset of a PC (its battery powered portability) is never properly highlighted nor the ramifications explained. The distribution channels were not properly chosen for this type of product. There seems little motivation for R.S. Computer Centers to push this item when a salesman can make a lot more commission promoting a desktop unit. Sharp apparently has missed out on a once-in-a-lifetime opportunity to be a major force in the U.S. market. In fact, I tend to think they never fully understood the opportunities that they had here. It seems obvious that they could have done much better. There are a lot of competitors coming along that will probably be able to avoid their mistakes. If Sharp does not develop a better understanding of the U.S. market in the near future, they may quickly lose their status as being a front-runner in this volatile field.

Panasonic/Quasar is apparently not pressing for the consumer market. It is hard to get a handle on their success in the industrial arena. I would be surprised if they did more than about 30,000 units overall (remember, I am considering only the U.S. market). I hear rumors of some sizable sales (a few hundred units) to firms that will develop special application modules. But, the relative price and lack of certain basic amenities seems to have kept them out of the so-called "popular" market. I don't expect much out of them in 1983.

Casio apparently never even knew what hit them. The FX-702P appears to have lacked a few fundamental features desired by PC users. Despite its touted special function capabilities, it never appeared to gain any marketing momentum. (One thing that probably hurt them was their inability to come up with the promised plug-in modules. A number of people told me they were waiting to see what those provided. When they never appeared, the interest in the product died.) I figure they would have been lucky to sell 20,000 units in 1982. I do not expect to see much in the way of wonders from Casio in 1983.

It really seems that PC manufacturers, as a group, are being creamed by the marketing clout of those such as Commodore, Atari and Texas Instruments. These latter firms are convincing newcomers that the way to join the computer revolution is through an attachment to their TV.

Play games, they say, and do it all in living color. The Timex-Sinclair unit is reported as being manufactured at a rate approaching 75,000 per month and the Commodore VIC-20 at around 40,000 per month. Look at what the PC manufacturers have missed!

Where the Market is Headed

People definitely want personal, portable computing power. They also want convenience. PC makers had better plan on offering at least two features if they want to stay in the game: The first is improved display capability (multiple-lines). The second is a comfortable keyboard. A keyboard such as that on a PC-1500, which was acceptable to the pioneering 1982 buyers, will not be tolerated by the mass-market consumers. They have better sense!

Two companies are already showing the trend. Hewlett-Packard, with its well-designed HP-75 (though they could soften up their keyboard), and Epson with their 4-pound HX-20.

I would tend to think that Epson, with their several years of experience marketing their popular line of printers in the U.S., would be in a position to exploit a wide-open opportunity. However, they too may blow it. I have learned that they have recently been shaking up and re-arranging their entire U.S. distribution system. Apparently they are going to try to keep the HX-20 out of the hands of mail order suppliers, a la Apple Computer Company's course. And, they may try to restrict access to technical information a la Sharp. Adopting either of these philosophies could hurt their penetration and long term sales potential, just as it has done to their predecessors.

PCs and Telecomputing: Not Good Enough — Yet!

A lot of companies appear to be trying to sell the idea that a PC is just what one needs to do telecomputing. (That is, have the PC act as a remote terminal for a large computer.) They are coming out with modems, RS-232C interfaces and "terminal software" (you could sure read that with more than one meaning) so that you can hook a PC to a telephone and talk to telecomputing networks such as The Source and CompuServe. After trying it out for awhile, I am thoroughly disappointed.

There are several problems. The two most poignant are: The PC keyboards, for the most part, are too small to permit touch-typing and so you spend too much time inputting commands and/or data. And, the displays are too small to permit comfortable viewing of the volumes of information that one has access to. I defy anyone to spend an hour trying to use CompuServe using a one-line PC display and come away without either having a severe headache or the feeling that your eyeballs had been through an experience for which they were not designed.

Oh yes, the idea is great. It would be nice to be able to access some of those data bases at anytime and from any location. It is the present implementation of the idea that is sorely lacking.

Here is what is needed to rectify the situation: A display that will allow at least 10 to 15 seconds of transmitted data to appear on the screen at one time. That is just enough time for you to make a rapid scan and consider whether the information has lasting value to you personally. (This could be achieved by using a TV or video interface. Some units, such as Panasonic's HHC and HP's 75C have this capability.) A printer unit that will allow you to make a hardcopy of important information. (Again, a number of PCs can be connected to printers. Some of these, however, may not operate fast enough for typical telecomputing applications.) And, most importantly (which no manufacturer that I know of has yet provided), a control arrangement that would allow you to turn the printer on and off as needed. It is important to realize that the printer must be capable of picking up what is currently in the PC's display buffer. Thus, if you were scanning data and you suddenly realized that you wanted to retain the information you were viewing, you would have several seconds in which to activate the printer and capture the relevant data!

These are all simple concepts, easy to implement with today's technology. The first company to be able to supply such a unit could really sock it to those firms that haven't figured on the importance of human factors engineering.

Thank you for supporting PCN during 1982. Best wishes to everyone for prosperity during 1983 and beyond. — Nat Wadsworth, Editor

1982 INDEX TO ARTICLES AND PROGRAMS

ENTERTAINMENT PROGRAMS

Auersbacher, Emerich:	
Electronic Card Deck	11 - 8
Heidbrink, Gary:	
American Roulette	17 - 7
Motto, David:	
Animated Runner	14 - 3
LCD Graphics for the PC-1500	14 - 5
Shifty Words	12 - 8
Peterson, Brian:	
Generating the Chromatic Scale	15 - 1
Music on the PC-1500	16 - 6

GENERAL APPLICATIONS

Auersbacher, Emerich:	
Amortization	13 - 3
Calendar	12 - 4
Coding/Decoding Messages	14 - 1
Phases of the Moon	14 - 8
Stopwatch	19 - 6
Weather Forecasting	20 - 2
Griffiths, Herb:	
Printer's Helper	16 - 6
Motto, David:	
Time	14 - 3
Nath, H.:	
Accounts Receivable	12 - 6
Peterson, Brian:	
Memo Printer	19 - 5
Robertson, Gene:	
Aviator	11 - 6
Saam, R. J.:	
Tape Library System	17 - 2
Slaughter, Ken:	
Determine Tab Settings	13 - 2
Supermarket Companion	11 - 4
Sparkman, James K.:	
Relative Humidity & Dew Point	12 - 4
Stevens, Geurri F.:	
Tax Assessment	17 - 4
Stonebrook, Kendal B.:	
Indirect File Names on Tape	11 - 1
Toy, S. J.:	
PC-1 as a Data Collecting Terminal	20 - 5

OPERATING INFORMATION

Auersbacher, Emerich:	
Tips & Tricks	13 - 8
Battan, Howard R.:	
Index for the PC-1500 Manual	16 - 8

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January-December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

1982 Regular Subscriber (Issues 11 - 20) \$30.00 in U.S.

(U.S. \$36.00 to Canada, U.S. \$45.00 elsewhere.)

1982/83 Subscriber (Issues 11 - 30) \$60.00 in U.S.

(U.S. \$72.00 to Canada, U.S. \$90.00 elsewhere.)

1983 Regular Subscriber (Issues 21 - 30) \$36.00 in U.S.

(U.S. \$42.00 to Canada, U.S. \$50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not

issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA #: _____ Expires: _____

Signature: _____

Chamier, A. D.:	
X-Y Plotter Interface	13 - 6
Cox, Thomas S.:	
Converting Statements	19 - 1
Error Codes for the PC-1500	16 - 5
Status Report	16 - 5
Fergus, George:	
Put Softkeys to Work	14 - 5
Speed Insertion of Statements	11 - 8
Hart, Rich:	
Op-Aids for the PC-1500/PC-2	19 - 7
Miller, Ron:	
PC Interface Signals	13 - 5
Motto, David:	
Comparison of PC Memories	16 - 4
Upper Case Conversion	16 - 1
Rober, Norlin:	
Exploring the PC-1/PC-1211	13 - 4
Getting Started With Machine Language	20 - 3
Hexadecimal Memory Dump	15 - 3
PC-1 Instruction Execution Times	11 - 2
PC-1500/PC-2 Disassembler	SE - 4
PC-1500/PC-2 Machine Language Codes	SE - 1
Peeking in the PC-1500	15 - 3
Rober Mnemonics Update	18 - 5
Understanding the PC-1500 Part 1	15 - 2
Understanding the PC-1500 Part 2	16 - 2
Understanding the PC-1500 Part 3	19 - 2
Use of Memory in the Radio Shack PC-1	13 - 4
Sherwin, Milt:	
PC-1500/PC-2 Renumbering Program	18 - 5
Renumbering Revisited	19 - 3
Staff, PCN:	
Converting Programs for the FX-702P	16 - 3
Softkeys Under Program Control	14 - 8
Updates to the PC-1500 Manual	16 - 6

PRODUCT ANNOUNCEMENTS

Staff, PCN:	
Epson Announces HX-20 Portable	20 - 1
FX-700P Cracks \$100.00 Barrier	18 - 1
IIHC Service on Compuserve	19 - 1
HP Announces Programmer's Aid	16 - 1
HP Introduces Portable Computer	17 - 1
Protection For Your PC	17 - 8
Radio Shack Announces New PC	12 - 2
Radio Shack Four-Color Printer/Plotter	12 - 2
Sharp Prepares New PC-1500	11 - 1
Sharp Announces PC-1500	12 - 1
Transfer Data from PC	17 - 8

PRODUCT REVIEWS

Motto, David:	
Panasonic IIHC	14 - 2
Sharp PC-1500	14 - 3
Rober, Norlin:	
Electrical Engineering 1	14 - 5
Staff, PCN:	
Sharp PC-1500	13 - 1
Wadsworth, Nat:	
Aviation Program	11 - 2
HP-75: Formidable Companion	18 - 2
Problem Solving on the PC-1	14 - 6
Sharp CE-150 Printer/Plotter	14 - 4
Telecomputing with a IIHC	16 - 7
99 Tips & Tricks for the New PCs	20 - 2

SCIENTIFIC/ENGINEERING PROGRAMS

Auersbacher, Emerich:	
Any Base Conversion	11 - 7
Chamier, A. D.:	
Fast-Fourier-Transform	13 - 3
Cox, Thomas S.:	
Curve Fitting	15 - 4
Decimal to Bases 2 - 16 Conversion	17 - 6
Librach, Hank:	
Low Pass Filter Design	12 - 5
Meredith, Paul T.:	
Interpolation Package	12 - 3
Rober, Norlin:	
Linear Regression Analysis	11 - 6
Polynomial Arithmetic	12 - 5
RPN Calculator for PC-1/PC-1211	18 - 7
Triangles	11 - 5
Shuman, David:	
Electronic Filter Design on the FX-702P	14 - 7



P.O. Box 232, Seymour, CT 06483