POCKET COMPUTER NEWSLETTER

 — Issue 21

January

SHARP TO INTRODUCE WALLET-SIZE PC TO U.S. MARKET

Sharp Electronics Corporation plans to introduce a new pocket unit dubbed the PC-1250 to the U.S. market in late January/early February. The small 5-5/16 by 2-3/4 by 3/8 inch computer is slated to retail at $109.95. It is reported to contain approximately 2K of RAM memory, including 1438 bytes of user program memory and 48 bytes of reserve user memory. In some respects an apparent upgrade of its earlier model PC-1211, the new unit has an enhanced BASIC. While claiming to be "upwards compatible" with the PC-1211, the BASIC will be capable of handling two-dimensional arrays and dealing effectively with strings.

Other enhancements over the PC-1211 include the BASIC statements DATA, READ and RESTORE, plus RANDOM and several commands for use with an optional cassette/printer interface.

The unit retains the excellent program editing features of its predecessor, such as cursor shifting, character insertion and deletion and line scrolling forwards or backwards. The display is a 24-digit dot matrix liquid-crystal display without "graphics" capability.

The new PC is reported to operate for about 300 hours of "display on" time when powered by a pair of lithium cells. Of course, memory is retained when the unit is in the powered-down non-display mode.

An optional printer/cassette unit, designated the CE-125 will also be made available for the new pocket computer. It will combine a 24-character thermal printer and a complete microcassette storage system. Note that the thermal printer width matches the width of the 24-character LCD of the PC-1250. The thermal printer will also be much quieter than Sharp's earlier ink printer for the PC-1211. The CE-125 measures 8-1/16 by 5-7/8 by about 1 inch. With the new PC mounted on it, Sharp reports it is about the size of an average hardcover book. The unit includes a complete microcassette recorder with which data and programs may be saved and restored to the PC. The tape unit is equipped with a tape counter to facilitate locating information rapidly. The optional CE-125 peripheral is reported to have a U.S. retail price of $169.95.

For more information, contact: *Sharp Electronics Corporation, 10 Sharp Plaza, Paramus, NJ 07652.*

Another *Personal Information*™ product from SCELBI Publications.

A FIRST LOOK AT THE EPSON HX-20

David G. Motto, 3639 Roosevelt Circle, Jackson, MI 49203, passes along these first-hand comments about his experiences with the new Epson HX-20 portable computer:

Never buy the first one of anything. I always say that, but then I do it anyway. I recently ordered an Epson HX-20 from Elek-Tek in Chicago. It took about a week to arrive from the time they said they would ship it.

The unit is not much bigger than an HP-75C. *[Sorry, Dave, but I think that 8-1/2 by 11-3/8 by 1-3/4 is substantially larger than 5 by 10 by 1-1/4 — N.W.]* It contains a noisy dot-matrix printer, a 20 by 4 (characters) liquid-crystal display screen, a full-size keyboard, a couple of weird-looking plugs for RS-232C and serial interfaces, and other assorted goodies.

Alas, when I opened the package, I noted that something was missing. The computer was shipped without any BASIC manuals. It did come with a nice owner's manual that tells how to charge the battery, install the paper and ribbon, move the cursor around the screen, and so forth. But, only one BASIC command is mentioned in that book. It would be rather difficult to write a program using just one command!

A notice on the shipping invoice said that the manuals would be shipped soon. But, waiting wouldn't be any fun. I decided to dive in on my own. I charged up the batteries, then turned the machine on. (The on/off switch is a recessed lever on the right side of the unit.) Then, I selected "2" from the menu on the LCD and the display responded with:

```
EPSON BASIC V-1.0
Copyright 1982 by
Microsoft & EPSON
P1:          0 Bytes
```

Then I started trying things. PRINT "DAVE" worked. I started making a list of directives that worked: MID$, LEFT$, RIGHT$, SOUND 5,1 (made a beep!). I tried the five keys labeled PF1 through PF5 that are right underneath the display screen. Each one caused a keyword to be printed: AUTO, LIST, LLIST, STAT, and RUN. All except AUTO and STAT execute immediately. However, if the printer is off, then LLIST won't cause any activity.

After spending several days experimenting, I found over 150 BASIC commands and functions. PEEK helped me to write a program that found itself in memory, but POKE would not allow my program to modify itself. I had to turn to the built-in Monitor program to help me on that project.

You can get back to the main menu just by pressing the Menu key. I would rather they make it a little harder, because it will interrupt whatever is going on.

The main menu consists of three choices: CTRL/@ Initialize. If you press the @ key while holding the control key, you can reset memory and the real-time clock. Another option puts you in the Monitor mode. The third is BASIC.

Ever try to operate a monitor program when you do not know the instructions? It is fun, but *dangerous!*

I went into the monitor mode and started trying letters of the alphabet. The letter "A" resulted in the contents of four items named T, L, O and E being displayed. Guess I will have to wait for the manuals for an explanation of those. The letter "D" followed by an address resulted in the display of 15 bytes of memory, but the cursor could not be moved to type over them (as can be done, on say, a Commodore PET). The letter "X" displayed the registers A, B, C, P, S and X. These are the accumulator, two single-byte registers (with C being, I think, the flag register), and 3 two-byte registers (possibly the program counter, stack pointer and index registers?). The letter "B" returned control to the main menu or back to BASIC if the MON directive had been used to get into the Monitor mode. The operation I was really looking for is invoked by the letter "S". It displays data and lets me change it!

I then located my program in memory and started trying to enter numbers larger than 127 on the first line of it. Complications arose. The keywords are stored as single-byte or two-byte codes. It appears that if they are double-byte, the first part is always 255. I think I have found most of the keyword codes.

Many of the built-in commands are used to access the not-built-in microcassette recorder. Did I mention that you can record programs on a regular cassette with the built-in interface?

Armed with my self-made list of BASIC instructions and some knowledge of Microsoft BASIC for CP/M computers, I worked on a few tiny programs. There are 12,891 bytes available to the user in the basic non-expanded system. I consider the BASIC to be very good -- better than Sharp's — with more commands, double precision (16 digits) arithmetic, some other capabilities. The number range, however, is only 10 to the 39th power, but how often do you deal with numbers greater than that?

The screen contents can be copied to the printer either with a couple of keystrokes or under program control. Pictures are possible as any dot on the 120 by 32 matrix can be turned on, off or tested for status. You can have five separate programs in memory at one time, with the same line numbers. You can run them from the main menu at the press of a button.

Disaster, however, struck my unit a few days after I received it! The first, second, seventeenth and eighteenth lines on my dot-matrix display stopped working. I had to package the unit up and send it back for repairs or replacement. Despite this early failure, however, I still like it!

Some other noteworthy observations: None of the promised accessories, such as the microcassette player, cables for RS-232C and serial interfaces, extra ribbons, paper, the CX-20 communications adaptor, modules, memory expansion or carrying case are available yet. The display has a control on it (similar to the Casio FX-702P contrast adjuster) that allows the display to be effectively "tilted." Sound volume is not adjustable. The computer beeps twice when power is first applied. Of course, memory contents are preserved when the unit is turned off. The HX-20 will run BASIC programs for about 50 hours (continuously) between an 8-hour recharge of its batteries. The keyboard is springy and feels good. There is a hidden numberic keypad for performing arithmetic calculations when in the BASIC direct execute mode. Error codes consist of two letters, as in the Panasonic HHC.

Even though you cannot fit an Epson in your pocket, I recommend *PCN* readers take a look at it.

SOME DETAILS ON EPSON'S BASIC

The Epson HX-20 has a lot of BASIC commands and statements that will be familiar to Sharp PC-1500 users, such as: END, FOR, NEXT, DATA, DIM, READ, LET, GOTO, GOSUB, RUN, IF, RESTORE, RETURN, REM, STOP, TRON, TROFF, ON, LPRINT, LLIST, PRINT, CONT, LIST, CLEAR, NEW, LOAD, SAVE, MERGE, PUT, GET, TAB, TO, SPC, USING, OFF, THEN, NOT, STEP, AND, OR, SGN, INT, ABS, SQR, LOG, EXP, COS, SIN, ATN, PEEK, LEN, STR$, VAL, ASC, CHR$, LEFT$, RIGHT$, RND, and the usual mathematical operators. These commands and statements (except for RND, which gives a random number in the range 0 to 1) are similar to those on the Sharp PC.

But, there are a host of BASIC statements, functions and commands on the HX-20 that give greatly enhanced capabilities to this portable unit. Here is a list of some of these new instructions with a brief explanation of their powers:

SWAP x,y	Exchanges the values in the variables x and y. Great in sorting situations.
DEFSTR, DEFINT, DEFSNG, DEFDBL	Used to define variables to be STRing, INTeger, SiNGle precision or DouBLe precision. Variables may be over 11 characters long and may contain embedded keywords.
RENUM m,n	Renumbers the program segment beginning with line m using an increment of n line numbers.

(continued on next page)

EPSON'S BASIC
(continuation from previous page)

ERROR n	Forces the error number n to occur, used for debugging or other purposes.
RESUME 0	Resumes program execution after an ON ERROR GOTO error trap. Can specify resumption with the same line, the next line or any specified line.
AUTO m,n	Begins automatic line numbering at line m with an increment of n.
DELETE m-n	Delete program lines m through n.
DEF FN x(y,z)	Define a function (similar to a one-line subroutine) named x with parameters y and z or more, if desired.
CLEAR n	Reserve n memory locations for string space. Default is 200 bytes.
OPTION BASE n	Set the first array element to be at 0 or 1.
RANDOMIZE n	Provide a seed to the random number generator.
ERASE x	Erases the array x to open up memory space.
LINE INPUT x	Accepts a line of input, but will not stop execution if only the RETURN key is used.
SCROLL n	Sets up the number of lines that the display will roll in either direction when display control keys are pressed.
SOUND x,y	Plays sound tone n (1 -56) for duration y.
MON	Passes control to the machine language monitor.
CLS	Clears the screen.
WIDTH x,y	Sets up the number of columns (x) and rows (y) which may be used by the virtual screen. Only 20 by 4 lines may be viewed at one time. Default value is 40 by 8.
COLOR n	Selects color from 0 to 7. (Plotter, TV?)
LOGIN x	Places you into workspace number 1 through 5. Each program workspace is separate, with its own line numbers. Like Casio's P0 through P9.
TITLE x	Gives the current workspace a title, Prevents the space from being accidently erased. Title appears on the main menu.
FRE(x)	Gives the number of bytes remaining in memory.
POS(x)	Yields the current column position.
EOF(x)	Provides the end-of-file status for file x.
CINT	Convert numbers from other forms to INTeger,
CSNG	SiNGle-precision or
CDBL	DouBLe-precision format.
FIX(x)	Returns the fixed-point portion of the number x. Is not the same as INT(x) when dealing with negative numbers.
SPACE$(n)	Provides n spaces.
HEX$	Converts numbers to hexadecimal notation.
OCT$	Converts numbers to octal notation.
MID$	Can be used on the left side of an equation to change the characters in a string at specified locations.
INSTR(x,y)	Locates where substring y begins in string x.
VARPTR x	Shows where the variable x is located in memory.
TIME$	Gives the time as hh:mm:ss
DATE$	Gives the date as mm/dd/yy
DAY	Gives the day of the week, Sunday = 0 through Saturday = 6.
INPUT$(x)	Waits for the receipt of x characters. Does not need RETURN.
POINT(x,y)	Give the status of the dot at position x,y.
PSET(x,y)	Turn on the dot at position x,y.
PRESET(x,y)	Turn off the dot at position x,y.
SCREEN n	Select screen 0 or 1.
COPY	Puts a copy of the screen on the printer.
TAPCNT	Displays the microcassette tape counter value.

UNDERSTANDING THE PC-1500

This is the fourth article in a series being presented by *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* Unless otherwise noted, the information presented by Norlin in this series also applies to the Radio Shack PC-2 unit.

Since it was first introduced, the Sharp PC-1500 has undergone a number of changes. Most of these are nearly unnoticeable to the average user. Many of these changes have been to correct minor errors. A few of the improvements, however, are rather significant.

The integrated circuit containing the ROM chip is stamped with one of the following identifying numbers:

A01 Original version.
A03 First revision. Also used in Radio Shack's PC-2.
A04 Latest version.

Apparently, there is no version A02 in widespread use.

You can readily identify the version in your machine (without physically opening your unit) by peeking at a few locations and using the following information:

Version	PEEK &C443	PEEK &C5BD
A01	56	129
A03	59	129
A04	59	74

I have identified many of the effects (but not all) of the modifications that have been made by various revisions to the ROM. Here are comments regarding these alterations.

Minor Improvements

1.) In version A01, when program execution is begun automatically using an ARUN statement, the BUSY signal in the display fails to turn on. This was corrected in version A03.

2.) In the original version, a program line may not consist of only a label. If you attempt to use such a line, ERROR 21 will result whenever the line is reached during program execution. A label alone may be used as a program line when ROM versions A03 and A04 are installed.

3.) In version A01, a statement containing USING, followed by the name of a numeric variable, will produce either a fatal crash or an execution of the power-up routine (with NEW0? :CHECK displayed). One may demonstrate this phenomena by typing USING A, followed by ENTER. This defect was corrected in version A04.

4.) In the REServe mode, a "template" of exactly 26 characters will place a zero byte in the location immediately following the template. This cancels a previously defined REServe key or template. (This only applies to the A01 ROM version). To illustrate, do the following:

A. Set the REServe mode, clear REServe memory with NEW.
B. Set to Group III. Assign something to one of the REServe keys.
C. Enter a template containing exactly 26 characters. (Note that now the assignment made to the REServe key has been lost.)
D. Set to Group II. Enter a template of exactly 26 characters. Now the template previously set for Group III has disappeared!

This defect was corrected in version A03.

5.) In REServe mode, with a version A01 ROM, it is not possible to clear the phrase assigned to a REServe key by overwriting with spaces.

6.) Under certain circumstances, when the input buffer is filled, the display continues on to include the contents of &7800 and beyond. This particular quirk exists in version A01 and in at least some PCs with version A03. In at least some cases, it does not occur with version A04. To demonstrate the problem, clear the display, then fill it with the token for RETURNs by repeatedly keying in DEF Y. When the end of the input buffer has been reached, there will probably be something unexpected appearing at the right end of the display. (If nothing happens, try POKE &7800, 87, 72, 65, 84, 63 and then repeat the above.)

7.) Another oddity, occurring only in version A01, can be observed by doing the following:

A. Enter the program line: 10 PRINT "ANYTHING"
B. In the RUN mode, key the following:
RUN
ENTER
1

ENTER
Up arrow
Cursor left

Note that the program line has been placed into the display, without the usual colon following the line number.

8.) Here is another quirk involving the display that appears in version A01 only. Enter the program line:

10 INPUT "A";A

Now RUN the program. In response to the prompt, enter just a plus sign (rather than a legal input). The display will show: ERROR 1. Note that using the CL key will not completely remove the error message.

9.) Enter the program line:

10 GCURSOR 152:INPUT A

When the program is executed, the display will show ERROR 32 IN 10, as expected. However, in version A01, the use of CL or Up arrow will not clear the error message. In version A03, the error message is cleared, but the computer continues to wait for input, with a partial question mark displayed at the right end of the LCD. (Version A04 works fine.)

Major Improvements

1.) In version A01 only, it is possible to completely mess up a program when variable storage in main memory occupies exactly one more byte than for which there is room. (Thanks to *Stan Kameny* for noting this problem.) What happens is that the "stop byte" following the last program line is overwritten by the variable. To illustrate the defect:

A. Clear using NEW0 and enter the program line:
10 ABCDEFGH
B. In RUN mode, execute one of the following, depending on how much RAM expansion you have installed in your PC:
No expansion: DIM A(228)
4K expansion: DIM A(2, 246)
8K expansion: DIM A(6, 178)
C. Now set PRO mode and LIST the program. Use the Down arrow to observe that the stop byte has been lost. Entering more program lines now will cause additional confusion, extending to the contents of REServe memory. Enter the program line:
60000 ABCDEFGH
D. Note the effect on REServe memory.

2.) In versions A01 and A03, the use of Boolean operators will produce incorrect results when inputs are negative numbers that are represented internally in binary form. Since ordinarily these Boolean operators are used with positive inputs only, the errors are of less consequence than it might seem at first. To illustrate:

A. NOT NOT 1 returns -31233. The correct result is 1.
B. 1 OR (-1 OR 1) returns 31233. The correct result is -1.
C. 1 AND (-1 OR 1) returns 0. The correct result is 1.

In version A04, none of these errors occur.

3.) In the RADIAN and GRAD modes (only), certain input arguments used with trigonometric functions produce incorrect results. This bug was corrected in version A04. In RADIAN mode, any input whose mantissa digits are 174532925199 will be treated as zero. In GRAD mode the same applies to mantissas having digits: 111111111111. To illustrate:

A. In RADIAN mode, SIN(π/18) returns 0. The correct result is .1736481777.
B. In GRAD mode, TAN(100/9) returns 0. The correct result is .1763269807.

Some Changes in the Operation of BASIC

In version A04 of the ROM, two changes were made to make Sharp BASIC more compatible with other BASICs. These changes must be taken into account if a program is to operate correctly amongst all PC-1500 and PC-2 computers.

1.) A change was made in the interpretation of a statement beginning with an IF expression. (Example: IF A THEN 50.) In versions A01 and A03, such a statement was regarded by the PC as true if the value of the expression was positive and false if zero or negative. In A04 it is considered as true if non-zero, false if zero.

2.) A change in the interpretation of FOR/NEXT loops affects the number of passes made through the loop for cases in which the final value of the index variable does not exactly equal the test value. In versions A01 and A03, when NEXT is executed the index variable is first compared with the test value. If it is less than the test value, it is incremented by the step size and the loop is repeated. In version A04, NEXT always increments the index variable and then compares the incremented result to the test value. The loop is repeated unless the new (incremented) index variable exceeds the test value. (Of course, if a negative number is being used as a step value, then this description must be modified.) The following summary should clarify the differences:

	A01, A03	A04
A. Loop: FOR X=0 TO 3		
Values of X used in loop:	0, 1, 2, 3	0, 1, 2, 3
Value of X after loop finished:	3	4
B. Loop: FOR X=0 TO 5 STEP 2		
Values of X used in loop:	0, 2, 4, 6	0, 2, 4
Value of X after loop finished:	6	6

A Warning!

Do not use the Boolean NOT operator in an IF statement. The use of NOT in a conditional test is not interpreted by the PC-1500 as meaning the negation of a logical statement. When Sharp chose to make the logical value of a true statement 1 and a false statement 0, they automatically ruled out the possibility of allowing NOT to mean negation. (Note that in Sharp's system, 0 and 1 may not have the same truth value. But, NOT 0 and NOT 1 are both calculated as negative numbers, thereby making NOT 0 and NOT 1 identical in truth value!) Perhaps this should be treated as a choice made by the designers, rather than as an error?

Here are a few examples you may use to convince yourself that NOT doesn't belong in an IF statement.

A. With ROM versions A01 or A03, enter the program line:
10 IF NOT (1=2) BEEP 1
Since it is not the case that 1=2, one would expect to hear a beep. There won't be one if you have ROM A01 or A03. However, the corrections made in A04 (regarding Boolean operators) affect the the use of NOT in IF statements. Thus, this example when run on a PC having ROM version A04, does produce a beep. However, the program line:
10 IF NOT (1=1) BEEP 1
does yield a beep with an A04 ROM, contrary to expectations. (There is no beep in the earlier ROMs with this statement.)

In Conclusion

It is to Sharp's credit that they are making efforts to constantly upgrade the PC-1500. However, anyone considering purchasing a new unit might first wish to make certain that they are getting the latest version of the PC. Users of earlier units may take care when programming to avoid the pitfalls discussed in this article.

HISTOGRAM PROGRAM

Russel Doughty, 3604 Northwick Place, Bowie, MD 20716, developed the program shown in the accompanying listing. He also provided the following comments regarding its use and application.

If you have ever used statistics, you know that they can often be misleading. Suppose you have a group of one hundred people with an average income of $20,000 ($20K). This could mean that they all make $20K or that fifty make $30K and fifty make $10K or that ten make $200K and the rest are unemployed!

In short, that single "average" number does not tell you much about the true distribution of incomes. It takes a look at the standard deviation, variance, minimum and maximum values to begin to get a true picture. Even then it can take a fair amount of interpretation to sort things out. But, your trusty Radio Shack PC-2 or Sharp PC-1500 and the plotting capabilities of its printer/cassette interface are a natural for creating "histograms." This is a statistical method that literally draws a picture of the data distribution.

You create a histogram by sorting your data into equally spaced "cells." Then you count the number of data points in each cell. For

the income example just mentioned, you might find (using your PC to analyze the data, of course!) that the minimum and maximum incomes were $10K and $100K respectively. You might choose the width of cells, then, to be $10K. You would then sort the data into nine cells: $10–20K, $20–30k, . . . $80–90K and $90 100K. You would then count the number of data points falling within each cell and plot the results. You end up with a visual representation of the data distribution. Interpretation becomes easy.

That is essentially what the accompanying program does. You will need a memory expansion module to use the program. It is written to utilize an 8K module. I believe the program will work satisfactorily in a system having only a 4K module, provided that you reduce the array named X that is DIMensioned in line 100.

Start the program using a RUN directive. The program will ask if you need instructions. Responding "Y" for yes results in the printout of a summary of the DEFined keys that have been established to facilitate selecting various program operations.

Initially, the program prompts for data inputs. When you have

Program *Histogram*

```
100:CLEAR :H=1:DIM
    X(255):DIM F(5
    1)
108:WAIT 100:PRINT
    "          HIST
    OGRAM"
112:INPUT "NEED IN
    STRUCTIONS(Y/N
    )? ";Z$:IF Z$=
    "Y"GOSUB 2000
114:PRINT "ENTER D
    ATA"
200:"A"WAIT 5
210:PRINT "X(";H;"
    )":CURSOR 8:
    INPUT X(H):CLS
220:H=H+1:GOTO 210
500:"S"TEXT :
    LPRINT :COLOR
    0:LPRINT "STAT
    ISTICS":LPRINT
510:FOR J=1TO H:
    FOR Q=1TO H-J
530:A=X(Q):C=X(Q+1
    )
550:IF A<CTHEN 580
560:X(Q)=C:X(Q+1)=
    A
580:NEXT Q
590:NEXT J
600:Z=0:S=0:FOR I=
    2TO J:S=S+X(I)
    :NEXT I:M=S/(J
    -1)
605:FOR I=2TO J:Z=
    Z+X(I)*X(I):
    NEXT I
607:V=(Z-(J-1)*M*M
    )/(J-2):CSIZE
    1
610:LPRINT USING "
    #######.##";"DA
    TA MIN = ";X(2
    )
620:LPRINT "DATA M
    AX = ";X(J)
630:LPRINT "MEAN
       = ";M:
    LPRINT "STD DE
    V  = ";√V
640:LPRINT "VARIAN
    CE = ";V
645:LPRINT "ENTRIE
    S  = ";H-1
650:WAIT :PRINT
805:"H"INPUT "CELL
     WIDTH ";W
810:Y=INT ((X(J)-X
    (2))/W)+1
815:FOR I=0TO Y:F(
    I)=0:NEXT I
820:FOR I=0TO Y
830:FOR Q=2TO J
835:R=INT (X(2))
840:IF X(Q)>=(R+(I
    *W))AND X(Q)<(
    R+((I+1)*W))
    THEN 848
844:GOTO 850
848:F(I)=F(I)+1
850:NEXT Q:NEXT I
905:"G"TEXT :LF 2:
    GRAPH :
    GLCURSOR (30,-
    150)
910:SORGN :LINE -(
    180,170),0,1,B
920:A=0:M=0
930:FOR I=0TO Y
940:IF A<F(I)THEN
    LET A=F(I)
945:M=M+F(I)
950:NEXT I
960:D=A
970:M=INT (D/M*100
    )
 1000:A=150/D:E=14
      0/(Y+1)
 1010:GLCURSOR (6,
      0):SORGN
 1020:FOR I=0TO Y
 1030:LINE ((I*E),
      0)-(((I+1)*E
      ),(F(I)*A)),
      0,3,B
 1040:NEXT I
 1043:GLCURSOR (-6
      ,0):SORGN
 1044:FOR I=0TO 7
      STEP 2
1045:LINE (0,(10*
     D*A-I*D*A)/1
     0)-(180,(10*
     D*A-I*D*A)/1
     0),2,1:NEXT
     I
1050:TEXT :CSIZE
     1:COLOR 0:LF
     4:LPRINT
     USING "#####
     .##";INT (X(
     2));TAB 26;
     INT (X(2))+W
     *(Y+1)
1055:LF -18:
     LPRINT TAB (
     15);"HISTOGR
     AM":LF -2
1056:LPRINT "%":
     LPRINT "TOT"
     :USING "###"
     :LPRINT TAB
     (1);M
1058:FOR I=2TO 7
     STEP 2:LF 2:
     LPRINT TAB (
     1);INT ((10*
     M-I*M)/10):
     NEXT I
1060:USING "#####
     .##";LF 5:
     LPRINT TAB (
     6);"MAX FREQ
     . = ";D:
     LPRINT TAB (
     6);"CELL WID
     TH= ";W
1065:WAIT :PRINT
1070:"C"LPRINT :
     LPRINT :
     LPRINT TAB 1
     5;"CELL DATA
     "
1080:LPRINT :FOR
     I=0TO Y:
     USING "#####"
     :LPRINT TAB
     (5);"CELL ";
     I+1;" = ";:
     USING "####.
     ##";LPRINT F
     (I);
1081:LPRINT INT (
     X(2)+I*W):
     NEXT I
1082:WAIT :PRINT
1085:"L"LPRINT :
     LPRINT :
     LPRINT TAB (
     12);"RAW DAT
     A LISTING":
     LPRINT
1090:FOR I=2TO H:
     USING "####"
     :LPRINT TAB
     (9);"DATA PT
     .";I-1;" = "
     ;:USING "###
     .##";LPRINT
     X(I)
1091:NEXT I
1092:WAIT :PRINT
     :GOTO 100
2000:CSIZE 1:
     LPRINT "DEF/
     S = STATISTI
     CS":LPRINT "
     DEF/H = HIST
     OGRAM"
2010:LPRINT "DEF/
     C = HISTOGRA
     M CELL DATA"
     :LPRINT "DEF
     /L = RAW DAT
     A LISTING"
2015:LPRINT "DEF/
     A = ADD MORE
      DATA"
2020:LPRINT "YOU
     CAN ENTER UP
      TO 255 DATA
      POINTS"
2030:LPRINT "YOU
     MUST USE DEF
     /S & H BEFOR
     E C OR L"
2040:LF 7:RETURN
STATUS 1        1853
```

entered a sufficient number of points, select the following options using DEFined keys:

DEF/S: Performs statistical analysis on the data and prints minimum, maximum, standard deviation and variance figures.

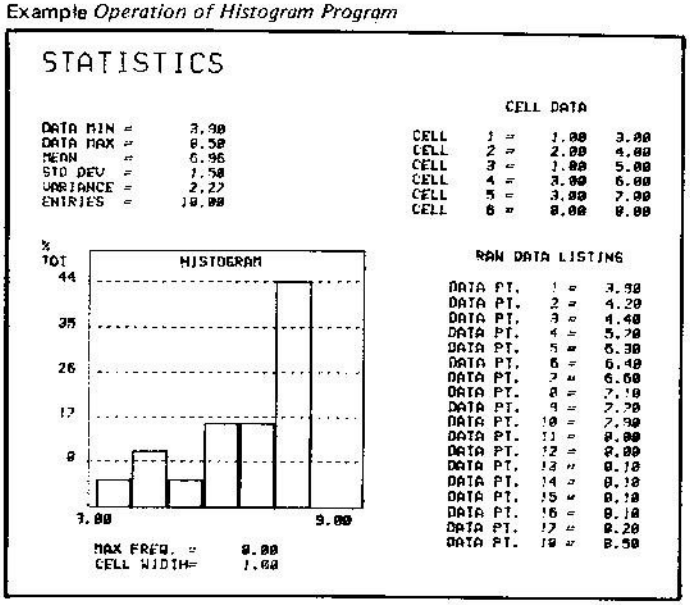
DEF/H: Draws a histogram of the data using the data previously entered and your response to the prompt for the cell width to use. The drawing is done using three pen colors. The vertical axis presents the number of data points in each cell expressed as a percentage of the total number of data points in all cells.

DEF/C: Gives a listing of the counts by cell.

DEF/L: Provides a listing of the data sorted in ascending order.

Have a little patience when working with a lot of data points. Your PC is pretty fast, but it cannot match an IBM 370. Oh yes, the program is designed to accept data having up to two signficant digits on either side of the decimal point. You may change this by altering the USING statements in the program.

Example *Operation of Histogram Program*



THREE-DIMENSIONAL PLOTS

Here is a program that makes it possible to represent three-dimensional surfaces. Curves consisting of the interesection of the surface with a number of equally-spaced "cutting planes" are drawn by the plotter.

To use the program with your PC-2/PC-1500 and plotter unit, line 100 must contain the formula for computing Z as a function of X and Y. A second function beginning at line 200 may be added, if desired.

Once the program has been started, prompts are given for the minimum and maximum values of X, Y and Z. You must also indicate the number of sections (cutting planes) to be drawn and the "increment size" (distance between the cutting planes).

It can take a number of minutes to produce one drawing since the number of points to be plotted can be quite large. Using relatively large increment sizes and reducing the number of cutting planes can speed up the process. However, this results in less detail being shown. Suitable compromise values for the number of sections seems to be in the range of 7 to 11 planes.

The X, Y and Z axes are not drawn by this program in order to avoid unnecessary clutter.

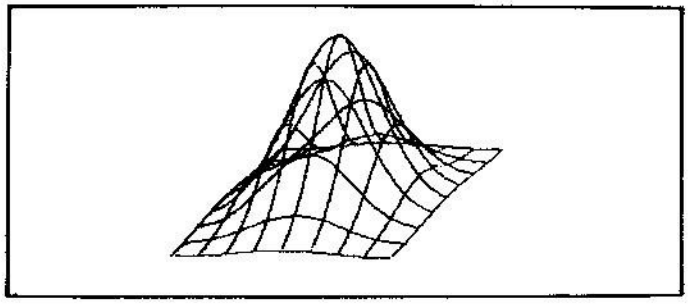
Here are the values used to draw the accompanying sample (for the function included at line 100 in the program listing):

```
MIN X? -2.5
MAX X? 2.5
MIN Y? -2.5
MAX Y? 2.5
MIN Z? 0
MAX Z? 1.3
NUMBER OF SECTIONS? 9
INCREMENT SIZE? 2
```

For variety, you might want to experiment with using different colors to draw the cutting planes. The effects could be interesting.

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

Example *Three-Dimensional Plot*



Program *Three-Dimensional Plotting*

```
10:INPUT "MIN X?
   ";A
11:INPUT "MAX X?
   ";B
12:INPUT "MIN Y?
   ";C
13:INPUT "MAX Y?
   ";D
14:INPUT "MIN Z?
   ";E
15:INPUT "MAX Z?
   ";F
16:INPUT "NUMBER
   OF SECTIONS? "
   ;G:G=G-1
17:H=1:INPUT "INC
   REMENT SIZE? "
   ;H
18:N=1:INPUT "NUM
   BER OF FUNCTIO
   NS? ";N
20:CLS :TEXT :
   CSIZE 2
21:LPRINT "GRAPHE
   D:":FOR I=1TO
   N:LLIST 100*I,
   100*I+99:LF -3
   :NEXT I:LF 1
22:LPRINT "MIN X=
   ";A:LPRINT "MA
   X X=";B:LPRINT
   "MIN Y=";C:
   LPRINT "MAX Y=
   ";D:LPRINT "MI
   N Z=";E:LPRINT
   "MAX Z=";F
30:XS=(B-A)/144,Y
   S=(D-C)/144,ZS
   =144/(F-E),GS=
   144/G
40:GRAPH :
   GLCURSOR (0,-2
   20):SORGN
50:FOR I=1TO N
51:FOR J=0TO G:K=
   1:XC=J*GS:FOR
   YC=0TO 144STEP
   H:GOSUB 60:
   NEXT YC:NEXT J
52:FOR J=0TO G:K=
   1:YC=J*GS:FOR
   XC=0TO 144STEP
   H:GOSUB 60:
   NEXT XC:NEXT J
53:NEXT I
54:GLCURSOR (0,-7
   0):END
60:X=B-XS*XC:Y=C+
   YS*YC:GOSUB 10
   0*I:L=.5*XC+YC
   :M=.5*XC+ZS*(Z
   -E)
61:IF Z<EOR Z>F
   LET K=1:RETURN
62:IF KGLCURSOR (
   L,M):K=0:
   RETURN
63:LINE -(L,M):
   RETURN
100:Z=EXP (-(X*X+Y
   *Y)/2)
199:RETURN
200:REM  2ND FUNCT
   ION
299:RETURN
STATUS 1          735
```

Program *HP-75C Weather Forecaster*

```
10 DELAY 1 @ DISP "WEATH
ER FORECASTER"
20 A$="S. SW W. NW N. NE
 E. SE "
30 INPUT "BAROMETER (IN)
:"; P @ IF P)=30.2 THEN
S=0
40 IF P<30.2 THEN S=1
50 IF P<30.1 THEN S=2
60 IF P<30 THEN S=3
70 IF P<=29.9 THEN S=4
80 INPUT "(R)ISE, (F)ALL
, (S)TEADY?"; T$ @ X=0
90 IF UPRC$(T$[1,1])#"R"
 THEN 130
100 INPUT "RISING (F)AST
, (S)LOW?"; R$ @ IF UPRC
$(R$[1,1])="F" THEN X=1
@ GOTO 190
110 IF UPRC$(R$[1,1])#"S
" THEN BEEP @ GOTO 100
120 X=2 @ GOTO 190
130 IF UPRC$(T$[1,1])="S
" THEN 190
140 IF UPRC$(T$[1,1])#"F
" THEN BEEP @ GOTO 80
150 INPUT "FALLING (F)AS
T, (S)LOW?";R$
160 IF UPRC$(R$[1,1])="F
" THEN X=3 @ GOTO 190
170 IF UPRC$(R$[1,1])#"S
" THEN BEEP @ GOTO 140
180 X=4
190 INPUT "WIND FROM THE
:"; W$ @ IF LEN(W$)#2 TH
EN BEEP @ GOTO 190
200 W=POS(A$,UPRC$(W$))
@ IF W=0 THEN BEEP @ GOT
O 190
210 W=INT(W/3)+1
300 Y=ABS(W-3)<2
310 IF Y*(S<3) AND X=0 T
HEN Z=1 @ GOTO 600
320 IF Y*(S=1) AND X=1 T
HEN Z=6 @ GOTO 600
330 IF Y*(S=0) AND X=4 T
HEN Z=3 @ GOTO 600
340 Y=W=1 OR W=8
350 IF Y*(S=1) AND X=4 T
HEN Z=4 @ GOTO 600
360 IF Y*(S=1) AND X=3 T
HEN Z=5 @ GOTO 600
370 Y=ABS(W-7)<2
380 IF Y*(S=1) AND X=4 T
HEN Z=4 @ GOTO 600
390 IF Y*(S=1) AND X=3 T
HEN Z=5 @ GOTO 600
400 IF Y*(S)2) AND X=4 T
HEN Z=6 @ GOTO 600
410 IF Y*(S)2) AND X=3 T
HEN Z=7 @ GOTO 600
420 Y=W=6 OR W=7
430 IF Y*(S<2) AND X=4 T
HEN Z=8 @ GOTO 600
440 IF Y*(S<2) AND X=3 T
HEN Z=9 @ GOTO 600
450 IF (W=1 OR W=2) AND
S)2 AND X=2 THEN Z=10 @
GOTO 600
460 IF (W<2 OR W)6) AND
S=4 AND X=3 THEN Z=11 @
GOTO 600
470 IF ABS(W-6)<2 AND S=
4 AND X=3 THEN Z=12 @ GO
TO 600
480 IF ABS(W-3)<2 AND S=
4 AND X=1 THEN Z=13 @ GO
TO 600
490 IF X<3 AND ABS(W-5)<
3 THEN Z=2 @ GOTO 600
500 Z=1 @ IF X=3 THEN Z=
8
600 DISP "FORECAST -->"
@ K$=""
610 ON Z GOSUB 710,720,7
30,740,750,760,770,780,7
90,800,810,820,830
620 K$=KEY$ @ IF K$="" T
HEN 610 ELSE 30
710 DISP "FAIR, LITTLE C
HANGE."; @ RETURN
720 DISP "FAIR, A LITTLE
 COOLER." @ RETURN
730 DISP "FAIR, A LITTLE
 WARMER." @ RETURN
740 DISP "RAIN WITHIN 24
 HRS." @ RETURN
750 DISP "WINDY, RAIN BY
 12 HRS." @ RETURN
760 DISP "INCREASING CLO
UDS, RAIN." @ RETURN
770 DISP "RAIN AND HIGH
WINDS." @ RETURN
780 DISP "UNSETTLED, CHA
NCE RAIN." @ RETURN
790 DISP "RAIN/SNOW, WIN
DY." @ RETURN
800 DISP "CLEARING, BECO
MING FAIR." @ RETURN
810 DISP "SEVERE STORM I
MMINENT." @ RETURN
820 DISP "HEAVY RAIN/SNO
W, WINDS." @ RETURN
830 DISP "CLEARING AND C
OOLER." @ RETURN
```

A FEW MORE WORDS ON THE HP-75C

A number of readers have expressed an interest in the HP-75C since its announcement and subsequent review in *PCN*. (See Issues 17 & 18.) Indeed, a few readers have advised us they have purchased the top-rated machine. To date *PCN* has not received any complaints as to its performance or perceived value. Several major magazines, including McGraw-Hill's *Popular Computing* have recently reviewed and warmly praised the unit. It is high priced, but many professionals appear to be quite enamored by the machine.

We are including a re-write of the Weather Forecasting program developed by *Emerich Auersbacher* that was originally published in Issue 20 of *PCN* for Sharp and Radio Shack PCs. This will give new owners of the HP-75C something to load and use right away. Those of you who might be contemplating the purchase of the HP-75C might want to peruse the program and compare it with the earlier version. There are some substantial differences between this HP version and the original Sharp/Radio Shack version. Remember, HP uses its own proprietary implementation of BASIC. It is quite different in some respects, especially as regards the handling of character strings.

The program uses 2076 bytes of memory as listed. About the only operating note regarding its use concerns the inputting of the wind direction. The wind direction must be inputted as one of eight major compass points: N., NE, E., SE, S., SW, W., NW. Note that the abbreviation of the four major points (North, East, South and West) must be followed by a period. Also, make note of the fact that the wind direction you must input refers to the direction the wind is blowing from, not towards.

We have noted one anomoly in the HP-75's operation. If you try to terminate a displayed line by pressing the RTN (return) key in order to defeat a set DELAY time, the computer apparently also accepts that input as an entry to the next INPUT statement it encounters! One solution to the problem seems to be to insert a dummy statement, such as W$=KEY$ just before the next INPUT statement that will be encountered following such an operation. This will "intercept" the key used to terminate the DELAY operation.

AVOID FROSTBITE!

January and February are predicted to be colder than average in much of the northern part of the U.S. this year. However, you can avoid frostbite by using this program to obtain the wind chill equivalent temperature. This program should run on just about any Sharp/Radio Shack PC, including the new 1250/1251 models! It was submitted by *Emerich Auersbacher* who included these comments and information:

The wind chill equivalent temperature is a useful indicator of how cold a person actually feels when exposed to various combinations of temperature and wind velocity. The accompanying program is especially useful to skiers or people who need to work or otherwise be outside during cold weather.

The program is based upon heat gain/loss equations originally developed by Siple and Passel from wind chill experiments performed at the Antarctic in 1941. The equation takes into account heat loss from radiation, conduction and convection, and is based upon an average skin temperature of 33C (91.4F).

Inputs required to run the program are the temperature (in degrees Fahrenheit) and the wind velocity in miles-per-hour. If you do not have a mechanical indicator of wind velocity, you may use the accompanying table to obtain an approximation of the wind velocity in miles-per-hour.

The program includes a frostbite alarm/alert feature. The message:
DANGER: FROSTBITE ALERT
is displayed when weather conditions combine to give a wind chill of minus 20 degrees Fahrenheit or less. Under those conditions, exposed flesh is subject to frostbite at a rapid rate. The message:
FROSTBITE WARNING
is flashed on the display when the wind chill drops below minus 10 degrees Fahrenheit. Frostbite is possible at this temperature if one is not careful.

* * * THE BEAUFORT SCALE OF WIND FORCE * * *

WIND	VELOCITY (mph)	DESCRIPTION
Calm	< 1	Smoke rises straight up.
Light	1 – 3	Smoke drifts, wind vanes motionless.
Slight	4 – 7	Felt on face, leaves rustle.
Gentle	8 – 12	Constant motion of leaves and twigs.
Moderate	13 – 18	Dust raised, small branches sway.
Fresh	19 – 24	Small trees sway, wavelets form.
Strong	25 – 31	Large branches sway, wires whistle.
High	32 – 38	Large trees sway, walking difficult.
Gale	39 – 46	Twigs break off trees.

Program *Wind Chill Factor*

```
 5:BEEP 1:PAUSE "
   WIND CHILL FAC
   TOR"
10:INPUT "WIND SP
   EED (MPH): ";U
   :U=.44704*U
15:IF U<1.78816
   LET U=1.78816
20:INPUT "TEMPERA
   TURE (F): ";T:
   T=5/9*(T-32)
30:A=10.45:B=10:C
   =-1
40:H=(A+B*SQR(U)+C*U)
   *(33-T)
50:T=33-(H/22.034
   ):T=9/5*T+32
60:U=5/9*(T-32):U
   =INT (10*U+.5)
   /10:T=INT (10*
   T+.5)/10
70:IF T<-19BEEP 5
   :FOR Z=1TO 4:
   PAUSE "DANGER:
    FROSTBITE ALE
   RT";PAUSE " ":
   NEXT Z:GOTO 90
80:IF T<-10BEEP 3
   :FOR Z=1TO 4:
   PAUSE "  FROST
   BITE WARNING":
   PAUSE " ":NEXT
   Z
90:PRINT "WIND CF
   = ";T;"F (";U;
   "C)"
100:GOTO 10
STATUS 1          424
```


FROM THE WATCH POCKET

A warm welcome to all of our new and renewing 1983 subscribers. We are now in our third year of publication. It looks as though there will be a lot of new PC entries this year....

On the top of the rumor mill is the buzzing over several initial entries about to be released by Texas Instruments. Could be as many as three models. It appears they are waiting for the CES exposition in Las Vegas to unveil?

Sharp appears to be aiming for the educational market with their PC-1250 in the U.S. They have opted to keep the cost low and will apparently not make the PC-1251 available in the United States. The PC-1251 is identical to the PC-1250 except that it has about twice as much user RAM. It has been available for several months in Japan, parts of Europe and Canada. The retail price is about $50.00 higher than that of the 1250. I think Sharp is making a mistake in not making the unit available to the U.S. market. Many customers who cut their teeth on the PC-1211 would love to upgrade to a wallet-size PC that had twice as much memory and six times the speed. They will think twice about buying a unit with all those nice enhanced BASIC instructions and yet not enough memory to really make good use of capabilities such as 2-dimensional arrays.

On the positive side, Sharp seems to be moving in the right direction with its 1250/1251 expansion equipment. I used to point out that it was inconvenient to have to carry a whole passel of equipment to have a practical system. That is, with the PC-1211 you had the printer/cassette interface, plus a tape recorder, plus connecting cables, plus regular-size cassette tapes, etc. The net result was that you had to carry essentially a whole briefcase full of gear in order to have a useful system.

Now, at least, the amount and size of the accessories have been cut substantially. The combination cassette/printer unit now includes the tape recorder (in microcassette form). When the PC-1250 is mounted on this unit, the entire system is about the size of a typical textbook. No more messy cables. You can stuff a handful of microcassettes in your pocket. Recording reliability should improve. It sure is a lot neater.

That the U.S. educational market is upper in Sharp's mind is apparent by the *Student Computer Handbook* they are providing with the PC-1250. It is the first PC book I have seen them deliver that has a chance of receiving some praise from users. I guess they finally figured out U.S. customers wanted to read English. This "self-teaching tutorial operator's manual" was developed for Sharp by the New York Institute of Technology. Compared to previous manuals, this one exhibits a little bit of class. You can actually understand what they are talking about! Bravo, Sharp, please keep commissioning those type of supportive publications.

If you have been sitting around waiting for all those nice software modules for the PC-1500 that have been promised for some time, just keep waiting. The latest story I have heard is that the developers were told they had to use a different type of ROM than originally planned on. Scratch three months of the delivery schedule. Current promises are for modules being available, perhaps, in February. I have seen some of the outputs (they make extensive use of the optional printer/plotter) from these modules. They look good. If they can ever get around to delivering them (before the PC-1500 is obsolete), they will probably move pretty well. The anticipated pricing is about $69.95 per 16K (ROM) module.

Speaking of software, Radio Shack will soon be releasing its first cassettes containing programs for the PC-2. Four have been announced so far: Personal Finance ($19.95), Business Finance ($19.95), Games Pak [sic] ($14.95) and Invasion Force ($9.95). The latter requires the addition of a 4K RAM module to the PC. It appears to be similar in some respects to the old classic Startrek game. Keep an eye out at your local Radio Shack Computer Center for the arrival of these packages.

If you have been getting some strange weather forecasts from the Weather program in Issue 20 of *PCN*, try correcting line 500 to read:
500 Z=1:IF X=3 LET Z=8

A number of people sent in samples of Christmas and Holiday cards they they generated on their PCs. It was quite enjoyable seeing the talent exhibited by so many PC users.

Thank You!

– Nat Wadsworth, Editor

POCKET COMPUTER NEWSLETTER

 — Issue 22 February

TEXAS INSTRUMENTS JOINS FRAY WITH THE CC-40

Calling its entry to the "portable" computer market the Compact Computer 40 (abbreviated CC-40), Texas Instruments joined the throng of traditional calculator manufacturers who are providing true computer power in miniaturized form.

The TI CC-40 weighs in at 22 ounces with dimensions of 9-1/2 by 5-3/4 by 1 inch (remarkably close to Hewlett-Packard's HP-75). It can be powered for approximately 200 hours on a set of four AA alkaline batteries. An optional 115-volt ac adapter provides desktop operation if desired.

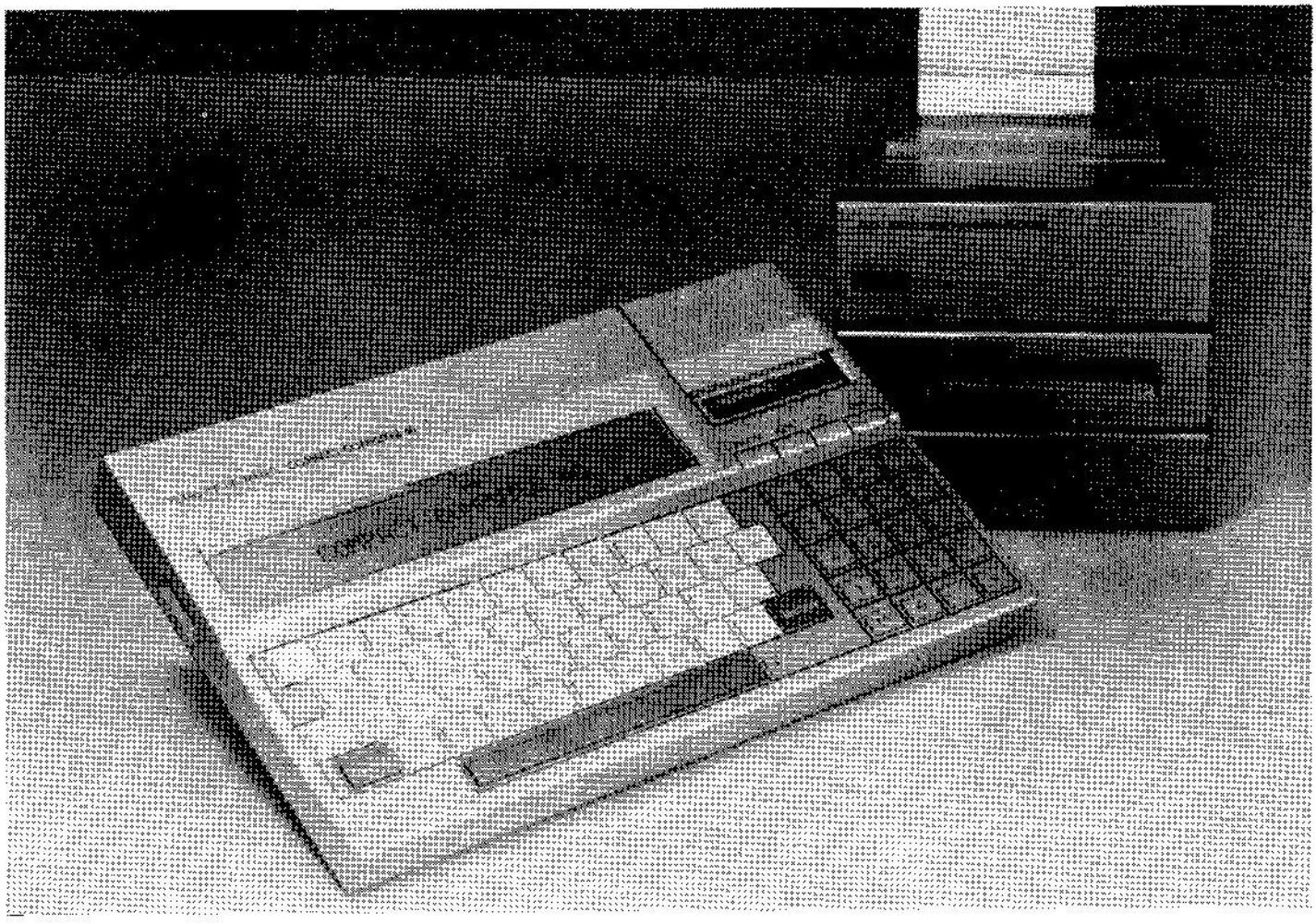
The little computer has a 31-character, scrollable liquid crystal display that presents both upper and lower case letters. There are 18 built-in annunciators including 6 user-settable flags. The staggered keyboard layout in QWERTY format and a numeric keypad provide for ease in keying. A tilt stand is built into the back of the console to provide an optimum viewing and keying angle.

The unit is programmable in Enhanced BASIC, similar to that used by TI's family of home computers. BASIC and the operating system are provided in 34 kilobytes of built-in ROM. The unit comes standardly equipped with 6K of user RAM, expandable internally to 16K. An additional 128 kilobytes of ROM containing advanced professional and personal programs may be plugged into a port provided for that purpose.

The list price of the basic CC-40 is $249.95. Deliveries are to start during the first quarter of 1983.

Three low-cost peripherals will be made immediately available to complement the computer. These are: an RS-232 interface, a printer/

(continued on page 7)



Another *Personal Information*™ product from SCELBI Publications.

EPSON DELIVERS MANUALS FOR THE HX-20

Epson is now delivering BASIC programming manuals for their HX-20 portable computer along with a couple of "bonus" program packages.

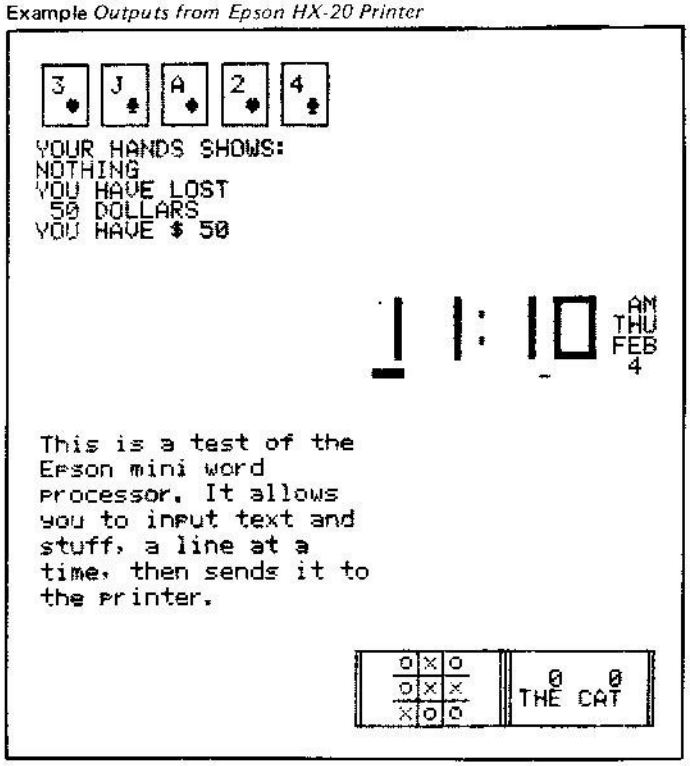
The two-volume BASIC manual consists of a 250 page "tutorial" for those who have not had previous programming experience and a second 250 page "reference" that is chock full of information to aid more experienced users.

The "bonus" packages consists of two cassettes containing software. The "Personal Entertainment" cassette has 10 "standards" such as Blackjack, Poker, Artillery, Hangman and Tic-Tac-Toe. The "Personal Productivity" cassette includes programs that provide a Clock, Event Timer, Memo Writer, and Ten Key Calculator among others. While all of the programs are relatively simple and exhibit signs of having been hastily prepared, most users will appreciate the good will gesture of at least providing some ready-to-run programs for the HX-20. A few samples of the output from these programs on the built-in HX-20 printer are provided here for your perusal.

For those looking for something else to load into a new HX-20, we are including our standard "benchmarking" Weather program for use and programming comparison purposes. This is an adaptation of the program originally supplied by *Emerich Auersbacher* for the Sharp PC-1500.

The Epson HX-20 has a lot of nice features and appears to be gaining a warm reception from the buying public. *PCN* plans to provide further reviews of the machine's capabilities in upcoming issues.

Example *Outputs from Epson HX-20 Printer*



Program *Epson HX-20 Weather Forecaster*

```
10 DIM A$(8):CLS:PRINT "
  PRESS RETURN FOR":PRIN
T :PRINT " WEATHER FOREC
ASTER";
15 X$=INKEY$:IF X$=""THE
N 15
20 A$(1)="S":A$(2)="SW":
A$(3)="W":A$(4)="NW":A$(
5)="N":A$(6)="NE":A$(7)=
"E":A$(8)="SE"
30 CLS:PRINT "BAROMETRIC
 PRESSURE":INPUT"(INCHES
)";P:IF P>30.2 THEN S=0
40 IF P<30.2 THEN S=1
50 IF P<30.1 THEN S=2
60 IF P<30.0 THEN S=3
70 IF P<29.8 THEN S=4
80 PRINT "(R)ISE, (F)ALL
, ":INPUT "OR (S)TEADY";
T$:X=0
90 IF T$="R"THEN PRINT "
RISING (F)AST OR":INPUT
"RISING (S)LOW";R$:IF R$
="F" THEN X=1:GOTO 150
100 IF T$="R"THEN X=2:GO
TO 150
110 IF T$="F" THEN PRINT
 "FALLING (F)AST":INPUT
"OR (S)LOW";R$:IF R$="F"
 THEN X=3:GOTO 150
120 IF T$="F"THEN X=4:GO
TO 150
130 IF T$="S" GOTO 150
140 GOTO 80
150 PRINT "WIND BLOWING"
:INPUT "FROM THE";W$:CLS
:GOSUB 200
160 IF Y=0 THEN SOUND 25
,1:PRINT "ERROR":GOTO 15
0
170 GOTO 300
200 Y=0:FOR Z=1 TO 8
210 IF W$=A$(Z)THEN W=Z:
Z=8:Y=1
220 NEXT Z:RETURN
300 Y=ABS(W-3)<2
310 IF Y*(S<3)ANDX=0 THE
N Z=1:GOTO 600
320 IF Y*(S=1)ANDX=1 THE
N Z=6:GOTO 600
330 IF Y*(S=0)ANDX=4 THE
N Z=3:GOTO 600
340 Y=W=1 OR W=8
350 IF Y*(S=1)ANDX=4 THE
N Z=4:GOTO 600
360 IF Y*(S=1)ANDX=3 THE
N Z=5:GOTO 600
370 Y=ABS(W-7)<2
380 IF Y*(S=1)ANDX=4 THE
N Z=4:GOTO 600
390 IF Y*(S=1)ANDX=3 THE
N Z=5:GOTO 600
400 IF Y*(S>2)ANDX=4 THE
N Z=6:GOTO 600
410 IF Y*(S>2)ANDX=3 THE
N Z=7:GOTO 600
420 Y=W=6 OR W=7
430 IF Y*(S<2)ANDX=4 THE
N Z=8:GOTO 600
440 IF Y*(S<2)ANDX=3 THE
N Z=9:GOTO 600
450 IF (W=1 OR W=2) AND
S>2 AND X=2 THEN Z=10:GO
TO 600
460 IF (W<2 OR W>6) AND
S=4 AND X=3 THEN Z=11:GO
TO 600
470 IF ABS(W-6)<2 AND S=
4 AND X=3 THEN Z=12:GOTO
 600
480 IF ABS(W-3)<2 AND S=
4 AND X=1 THEN Z=13:GOTO
 600
490 IF X<3 AND ABS(W-5)<
3 THEN Z=2:GOTO 600
500 Z=1:IF X=3 THEN Z=8
600 PRINT "    FORECAST
-->":PRINT:K$=""
610 ON Z GOSUB 710,720,7
30,740,750,760,770,780,7
90,800,810,820,830
620 X$=INKEY$:IF X$=""TH
EN 620 ELSE 30
710 PRINT "FAIR, LITTLE
CHANGE.":RETURN
720 PRINT "FAIR, LITTLE
COOLER.":RETURN
730 PRINT "    FAIR, WAR
MER.":RETURN
740 PRINT "RAIN WITHIN 2
4 HOURS":RETURN
750 PRINT "WINDY/RAIN IN
 12 HRS":RETURN
760 PRINT " CLOUDS/RAIN
COMING.":RETURN
770 PRINT " RAIN & HIGH
WINDS.":RETURN
780 PRINT "UNSETTLED/MAY
BE RAIN":RETURN
790 PRINT" RAIN/SNOW & W
INDY.":RETURN
800 PRINT " CLEARING & F
AIRER."
810 PRINT "SEVERE STORM
WARNING":RETURN
820 PRINT "HEAVY RAIN/SN
OW/WIND":RETURN
830 PRINT " CLEARING & C
OOLER.":RETURN
```

A MACHINE-LANGUAGE MONITOR FOR THE PC-1500/PC-2

This Monitor for the Sharp PC-1500/Radio Shack PC-2 is designed to make it easier to write, load and debug machine-language programs. The program was created by *Norlin Rober.* Norlin is the originator of the mnemonics which *PCN* uses when discussing these units' machine-language. (The instruction set was presented in the Special Edition of *PCN* published in September of 1982. A summary of the mnemonics and machine codes, on a pocket-size card for easy reference, is being distributed with this issue of *PCN.*)

This is what is sometimes referred to as a "hybridized" program. The portion of the program involving operator inputs and outputs is written in BASIC. The remainder is in machine language. One reason for using the latter is that execution is much faster, particularly for operations such as MOVE and HUNT.

The Monitor includes the commands shown in the accompanying list. They are executed by the indicated user-defined keys.

List *Monitor Commands*

DEF/S	STORE codes into memory.
DEF/V	VIEW memory contents.
DEF/L	LIST memory contents using printer.
DEF/H	HUNT for a specified byte.
DEF/M	MOVE a block of stored codes.
DEF/B	BREAKPOINT insertion in user's m.l. program.
DEF/J	JUMP to specified address in user's m.l. program. (If a breakpoint is reached, contents of CPU registers and stack may be printed.)
DEF/A	ALTER contents of CPU registers at breakpoint.
DEF/C	CONTINUE execution following breakpoint.
DEF/F	FIX program (remove breakpoint).
DEF/X	EXIT from STORE, VIEW or HUNT mode.
DEF/=	Convert decimal number in display to hexadecimal.

Loading the Monitor into Memory

This program is intended for use in a PC having an 8K memory module. The first step in loading the program is to execute the command: NEW &4000. This reserves a portion of memory for machine-language purposes. There are then two parts to entering the actual Monitor:

1). The accompanying machine-language codes must be entered into an appropriate section of memory using POKE statements. Although this is a tedious process, once it is finished the codes may be stored on cassette for easy loading in the future. It is important that the codes be stored in the exact locations specified.

2). The accompanying BASIC portion of the program is then keyed into memory in the usual way. It too may be saved on cassette for ease in future loading of the Monitor.

It is a good idea to save the program on cassette *before* testing it! Then, in the event you have made any keying errors that result in a bad program crash, you will not have to key everything all over again.

Saving/Loading the Monitor Program Using a Cassette

After the two parts of the program have been keyed into memory, they should be recorded on cassette as follows:

1). Execute CSAVE M "MONITOR PART 1"; &3E4F, &3FFF
2). Execute CSAVE "MONITOR PART 2"

To reload the program from cassette, use:

1). NEW &4000
2). CLOAD M
3). CLOAD

Monitor Memory Map

The Monitor program uses memory areas as follows:

3800 – 38C4 REServe memory area, not used by Monitor.
38C5 – 3DFF Available for user's m.l. routines (1339 bytes).
3E00 – 3E4B Monitor temporary registers and stack area.
3E4F – 3FFF Monitor machine-language routines.
4000 – 5FFF BASIC storage area. Monitor uses 929 bytes.

The Monitor also uses variables A, B, C, A$, B$, C$ and D$. Thus, those memory areas (&7900 – &7917 and &78C0 – &78FF) may be altered when the Monitor is in operation.

Operating the Monitor

Place the PC in the RUN mode and use the DEFined keys to select the various operations. Always enter addresses as four hexadecimal digits (using 0 – 9 and A – F). Machine codes should be entered as two-digit hexadecimal numbers. Note that (for example) the code 0B should be entered by keying both 0 and B, not simply B. The & prefix normally used to indicate hexadecimal values to the PC is not necessary when responding to Monitor prompts.

The following discussion provides information on the use of each of the Monitor commands:

STORE: DEF/S places the computer in the mode used to load and store machine code. The Monitor will prompt for the beginning address at which codes are to be stored. Respond to this prompt with a four-digit hexadecimal address (such as 38C5) and then press the ENTER key. The program will then show the address into which the next byte of code will be placed. The code must be entered as a two-digit hexadecimal number (for example, B5) which is terminated by the ENTER key. After each byte has been accepted, the display shows the next sequential address. You may thus continue entering object code into successive memory locations. If you want to skip over an address, then just press ENTER without any input. This advances the Monitor to the next address, leaving the contents of the skipped address unchanged. To exit this mode, key DEF/X.

VIEW: Key DEF/V to place the Monitor in the mode used to examine the contents of memory. Enter a beginning address when prompted. The computer will then display the contents of memory in groups of four bytes. Press the ENTER key to advance to the next group of four bytes. Use DEF/X to terminate this mode.

LIST: Initiated by DEF/L, this command operates similarly to VIEW, except that the memory contents are printed instead of being displayed on the LCD. However, two address prompts will appear: one to indicate the starting address, the other denoting the ending location.

HUNT: Key DEF/H. Enter the byte being sought at the prompt. A second prompt asks for the address at which the Monitor is to begin searching. When a byte having the indicated value is located, the display will show the address, the byte itself, and the three bytes that follow it. Press ENTER to continue the search for another occurrence of the same byte.

As an example, the Monitor command makes it easy to locate the codes used at the beginning of BASIC program lines. Simply HUNT for the byte 0D! (The code 0D is the ASCII representation for "carriage return" used to terminate lines of BASIC.) Given 4000 as a beginning address (assuming this Monitor program is in memory), the display will show 401E 0D 00 0C 08. This indicates that the first BASIC program line (of the Monitor) terminates at address 401E. The next line number (using two bytes starting at 401F) is 00 0C (line 12 in decimal). The last byte (08) in the display represents the "link" byte. It indicates the number of bytes that remain in that BASIC program line. If you press the ENTER key, you will find where the next line ends. Use DEF/X to exit from this mode.

The HUNT command is also highly useful as an aid to locating specific instruction codes when investigating ROM.

The speed at which HUNT operates dramatically illustrates the advantage of using machine language.

MOVE: DEF/M initiates this mode. Using it, a block of stored codes of any size may be moved, in either direction, to any specified RAM location. The new location may overlap the original. Unless there is overlapping, the original block of code remains unchanged by the operation.

Prompts request the beginning and ending addresses of the original block of memory, and then the beginning address of the location to

which it is to be transferred.

Be careful when using this command. If an incorrect address is inadvertently given, you may destroy parts of memory that you did not want altered! A wrong move operation can alter the Monitor program itself or disturb pointers in system RAM, possibly resulting in a system crash.

The following example may be used to familiarize yourself with the command. The variable J is stored (by BASIC) in RAM addresses 7948 to 794F. Variable M begins at address 7960. You can copy the contents of variable J into M using MOVE. Use BASIC to store some arbitrary test number in J. Key DEF/M to initiate MOVE. Enter the address 7948 in response to the prompt BEGIN BLOCK and address 794F for END BLOCK. In response to MOVE TO, key in the address 7960. Press ENTER at the end of this address and the MOVE command will be executed. You can then use BASIC to display the contents of the variable M. It should contain the same value as that originally placed in J.

BREAKPOINT: This command is useful in debugging machine-language programs. What it does is remove three bytes from a program in RAM, at whatever address is specified, and save them elsewhere. The three locations are then loaded with opcodes for an instruction to jump to a register-saving routine, which saves the contents of the CPU registers, flag status register and the stack. If execution is resumed later, (for example, by the Monitor's CONTINUE command), the CPU registers, flags and stack will be restored.

Note that the BREAKPOINT command executed by DEF/B simply inserts an instruction to jump to the register-saving routine. It does not execute the user's machine-language program. Also note that only one breakpoint should be in a program at any given time.

As a note of caution: be sure that the address entered for insertion of a breakpoint is the address of the *first byte* of a machine-language instruction!

JUMP: Key DEF/J to use this routine. The address entered in response to the prompt will be taken as the beginning address of the code to be executed.

This command may be used in lieu of a BASIC "CALL" instruction to begin executing a machine-language program. It *must* be used when debugging a program containing a breakpoint.

If a program containing a breakpoint is executed (using the JUMP command), BREAKPOINT REACHED will be displayed when the breakpoint is found. The user may then initiate a printout of the CPU registers, flags and stack by pressing ENTER.

ALTER: DEF/A will begin the routine for altering the contents of the CPU registers following the reaching of a breakpoint. When the prompt for a particular CPU register appears, the new value should be entered. Note that CPU register A contains one byte, requiring inputting of two hexadecimal digits. Registers X, Y and U, on the other hand, each hold two bytes, hence four-digit values should be entered for those registers.

If you want to leave a particular register undisturbed, press ENTER alone when the prompt for that register appears.

After all desired alterations have been entered, the three bytes that had been removed from the program by the BREAKPOINT command will be replaced. Execution of the machine-language program then continues from where it left off, with the CPU registers now containing the altered values specified.

CONTINUE: Execution of DEF/C will continue execution of a program following a breakpoint, with no alteration of the CPU registers.

FIX: The DEF/F command replaces the three bytes that were removed by a breakpoint directive. It is only needed if neither ALTER or CONTINUE are used following establishment of a breakpoint, since those two commands make the same replacement.

EXIT: Use DEF/X to exit from the STORE, VIEW or HUNT modes. The command is provided as a convenience, rather than a necessity. (The use of SHIFT/CL would accomplish the same effect.)

HEXADECIMAL equivalents of calculated results may be obtained using DEF/=. This is useful for such things as calculating the required size of a relative branch. An example illustrating hexadecimal conversion follows:

Enter 35 * 61 into the display. Now key DEF/=. The product will be shown as 0857 which is the hexadecimal equivalent of the product of the decimal numbers 35 and 61. Enter &3A56 - &39EB and key DEF/=. You will have an instant answer for a hexadecimal subtraction!

You may also simply convert a decimal number to hexadecimal by entering the number into the display and using DEF/=.

The BREAKPOINT Illustrated

To amplify how BREAKPOINT and related commands operate, I will illustrate their use with a specific example.

To begin, input a short, simple machine-language program. Put the codes into memory using the STORE command. Here is such a routine:

Address	Opcodes	Mnemonics
38C5	48 12	LDXH #12
38C7	4A 34	LDXL #34
38C9	58 56	LDYH #56
38CB	5A 78	LDYL #78
38CD	68 9A	LDUH #9A
38CF	6A BC	LDUL #BC
38D1	B5 00	LDA #00
38D3	FD 88	PSHX
38D5	FD 98	PSHY
38D7	FB	SETC
38D8	FD 1A	POPY
38DA	FD 0A	POPX
38DC	AE 39 00	STA 3900
38DF	A4	LDA UH
38E0	AE 39 01	STA 3901
38E3	24	LDA UL
38E4	AE 39 02	STA 3902
38E7	9A	RTS

A careful examination of this example routine will reveal that it does not accomplish anything of any practical importance. The purpose of the routine is simply to illustrate how some of the Monitor features perform.

After storing the example codes, use VIEW to make certain that they have been entered correctly. Then try executing the program with the JUMP command, giving address 38C5 as the response to the BEGIN prompt. To see whether the program did what it was designed to do, use VIEW again to see if addresses 3900 to 3903 contain appropriate values.

Try a breakpoint at the address 38D8 by keying DEF/B and entering 38D8 at the prompt. Then execute JUMP again using 38C5 as the starting location. The BREAKPOINT REACHED message should soon appear. Assuming that your PC is connected to its printer, press the ENTER key. Compare the printout against what the machine-language routine should have accomplished to that point. (The "CIZ" in the printout means that flags C, I and Z were set at the time the breakpoint was encountered.)

You can then alter the contents of, say, registers A and U, then see whether the altered values eventually find their way to addresses 3900 to 3902. First, key DEF/A. The prompt "A:" appears to request a new value for register A. Enter, for trial purposes, "BB" and press the ENTER key. Skip over changing X and Y by just pressing ENTER when their prompts come up. When "U:" appears, input FFEE as a new value and then press ENTER.

Use VIEW starting at address 3900 to verify that the CPU alterations took effect.

Key DEF/F to repair the breakpoint. You will be informed that there is no breakpoint to be fixed! Remember, using the ALTER feature automatically "fixes" the breakpoint. If you try either ALTER or CONTINUE at this point (without putting in another breakpoint), you will get the same response.

However, there is no built-in protection against your entering a second breakpoint without having fixed the first one. If you put in more than one breakpoint at a time, the FIX routine will only correct the last one entered. (The Monitor will have then lost track of where you placed the first breakpoint and what the original bytes were.)

Program *Machine Language Portion of Monitor*

```
3E4F BE 3F DE 58
3E53 3F 5A 19 84
3E57 51 04 51 F5
3E5B F5 F5 5B FF
3E5F 46 B5 71 43
3E63 B5 3E 43 B5
3E67 BA 0E 9A EB
3E6B 3E 70 FF FD
3E6F 5E FF FD C8
3E73 FD 88 FD 98
3E77 FD A8 FD AA
3E7B FD C8 FD 48
3E7F FD 88 46 58
3E83 3E 5A 00 F5
3E87 4E 49 91 05
3E8B AA 78 48 E9
3E8F 3E 70 00 9A
3E93 48 3F 4A 19
3E97 BE 3F 9A 5A
3E9B D0 48 3E 4A
3E9F AF A5 3E 02
3EA3 6A 04 D5 81
3EA7 02 F5 46 44
3EAB 88 08 8E DE
3EAF 43 49 5A 56
3EB3 48 48 3E 4A
3EB7 03 BE 3F 9A
3EBB 5A F0 BE 3F
3EBF 9C 5A C0 BE
3EC3 3F 9C 8E E3
3EC7 48 3F 4A 19
3ECB 45 18 45 1A
3ECF F5 F5 F5 49
3ED3 00 9A 48 3E
3ED7 4A 09 8E FC
3EDB 4A 07 8E 06
3EDF 4A 05 8E 02
3EE3 4A 03 48 3E
3EE7 BE 3F D7 8E
3EEB ED AA 3D FF
3EEF FD 0A FD 4E
3EF3 44 58 3E 5A
3EF7 02 55 41 4E
3EFB 49 91 06 FD
3EFF 8A FD EC FD
3F03 2A FD 1A FD
3F07 0A 8B 08 FD
3F0B 8A ED C0 00
3F0F FF 8E 06 FD
3F13 8A ED C0 00
3F17 00 BA 00 00
3F1B 00 00 00 00
3F1F A5 79 15 F7
3F23 99 03 46 BE
3F27 3F 8F 46 46
3F2B 46 9A BE 3F
3F2F DE FD 5A F4
3F33 79 05 FD 28
3F37 FB A5 79 0E
3F3B 00 2A A5 79
3F3F 0D 80 28 84
3F43 96 81 0E 89
3F47 04 04 16 81
3F4B 08 F5 88 03
3F4F FD 62 93 07
3F53 9A 24 FD CA
3F57 FD DA 84 A2
3F5B 08 94 A2 18
3F5F 47 53 88 04
3F63 FD 62 93 08
3F67 9A F2 48 79
3F6B 4A 10 BE DC
3F6F 20 58 78 5A
3F73 E0 48 7A CD
3F77 6C B7 00 8B
3F7B 02 B5 2D BB
3F7F 20 51 D0 00
3F83 04 FD 28 8E
3F87 32 56 CD 2A
3F8B 65 05 8E 44
3F8F BE 3F B8 6A
3F93 03 BE 3F AE
3F97 88 05 9A 5A
3F9B E0 58 78 B5
3F9F 20 51 8E 02
3FA3 5A E0 BE 3F
3FA7 B5 8E 0B 58
3FAB 78 5A D0 B5
3FAF 20 51 8E 02
3FB3 5A D0 45 8E
3FB7 07 5A C0 84
3FBB BE 3F BF 04
3FBF 58 78 28 F1
3FC3 BE 3F C7 A4
3FC7 F9 B9 0F B3
3FCB 30 B7 3A 81
3FCF 02 B3 06 51
3FD3 59 00 FB 9A
3FD7 5A D0 8E 3F
3FDB E9 41 9A 5A
3FDF C0 BE 3F E9
3FE3 08 BE 3F E9
3FE7 0A 9A 58 78
3FEB BE 3F F6 F1
3FEF 28 BE 3F F6
3FF3 A2 FB 9A 55
3FF7 B7 40 81 02
3FFB B3 08 B9 0F
3FFF 9A
```

Program *BASIC Portion of Monitor*

```
10:"="AREAD C:
   CALL &3F68:
   CURSOR 21:
   PRINT C$:END
12:"S"GOSUB 68
14:CALL &3FB8,A:
   WAIT 0:PRINT A
   $;:WAIT :INPUT
   ": ";B$:CLS :
   CALL &3FD7,A:
   GOTO 14
16:CLS :A=A+1:
   GOTO 14
18:"V"GOSUB 68
20:CALL &3F8F,A:
   PRINT A$:GOTO
   20
22:"L"GOSUB 72
24:CALL &3F8F,A:
   LPRINT A$:IF A
   >BEND
26:GOTO 24
28:"H"CLS :INPUT
   "BYTE SOUGHT:
   ";A$:CALL &3FD
   E,C:GOSUB 68
30:CALL &3F1F,A:
   PRINT A$:GOTO
   30
32:"M"CLS :INPUT
   "BEGIN BLOCK:
   ";A$:CALL &3FD
   E,A:INPUT "END
    BLOCK: ";A$:
   CALL &3FDE,B
34:INPUT "MOVE TO
   : ";A$:CALL &3
   F2D:END
36:"X"CLS :END
38:"B"CLS :INPUT
   "BREAKPOINT: "
   ;A$:CALL &3E4F
   :END
40:GOTO 38
42:"J"GOSUB 68:
   CALL &3E6A,A:
   IF PEEK &3E70
   END
44:PRINT "BREAKPO
   INT REACHED"
46:CALL &3E93:
   LPRINT "ADDRES
   S:";C$:LPRINT
   "FLAGS: ";B$
48:CALL &3EB4:
   LPRINT "A:";B$
   :LPRINT "X:";A
   $:LPRINT "Y:";
   D$:LPRINT "U:"
   ;C$
50:B=64-PEEK &3E0
   1:IF B>0LPRINT
   "STACK:";:A=&3
   E0A:FOR C=1TO
   B:CALL &3FAA,A
   :LPRINT B$;:
   NEXT C:LPRINT
52:LPRINT :END
54:"A"GOSUB 76:
   INPUT "A: ";B$
   :CALL &3ED5
56:INPUT "X: ";B$
   :CALL &3EDB
58:INPUT "Y: ";B$
   :CALL &3EDF
60:INPUT "U: ";B$
   :CALL &3EE3
62:CALL &3EEC:END
64:"C"GOSUB 76:
   GOTO 62
66:"F"GOSUB 76:
   END
68:CLS :INPUT "BE
   GIN: ";A$:CALL
   &3FDE,A:RETURN
70:GOTO 68
72:GOSUB 68:INPUT
   "END: ";A$:
   CALL &3FDE,B:
   RETURN
74:GOTO 72
76:CLS :IF PEEK &
   3F1ECALL &3EC7
   :RETURN
78:PRINT "NO BREA
   KPOINT":END
STATUS 1          929
```

Crash Recovery

Anyone experimenting with machine language soon learns that the PC (or any computer!) is totally unforgiving of errors. There are none of the friendly "ERROR" messages as found in BASIC. In many cases, the simplest mistakes will be punished by "crashes" which result in the PC becoming completely locked up. The only thing to be done in such situations is to use the "All Reset" key on the back of the PC. But, all is not necessarily always lost!

Instead of following the instructions given in the manual when a crash occurs, try using the following approach. Nine times out of ten your program will still be intact: First, do *not* hold in the ON key while pressing "All Reset." Secondly, after you obtain "NEW0? CHECK" on the display, do *not* execute NEW0! Instead, key SHIFT

and then CL (SHIFT/CL). Then, survey the damage, if any, to the contents of memory. If you find things messed up beyond repair, then you will have to key in NEW0 and start over. Often, however, you will find memory contents retained with little alteration.

The Monitor program does contain a few safeguards. If you neglect to key in an *address* before pressing ENTER, the computer will repeat the prompt. If you forget any of the three addresses required for a MOVE, it will restart the entire MOVE routine from the beginning. And, if you use the hexadecimal conversion routine with a result exceeding FFFF (or less than -FFFF), the Monitor will put the word ERROR into the display. Otherwise, you are on your own!

Combined Monitor/Disassembler

The use of this Monitor program can be greatly enhanced by coupling it with the Disassembler program published in *PCN* (Special Edition accompanying Issue 17). Those who have the Disassembler on a cassette can simply MERGE that program with this Monitor. To facilitate joint use of the Monitor and Disassembler, change the Disassembler as follows:

1). Insert "D" as a label at the beginning of line 10.
2). Delete lines 90, 91 and 92 as they are no longer needed.
3). It would also be advisable to eliminate lines 12, 80 and 82.

Now DEF/D may be used to DISASSEMBLE! A printed list of mnemonics generated by the Disassembler is often of considerable help in finding elusive machine-language programming errors!

If you are interested in doing machine-language programming on your PC-2/PC-1500, you should find this Monitor of considerable value. The 1339 bytes (from address &38C5 through &3DFF) that are available for machine-language routines should be adequate for most purposes. (If you are able to write longer routines, you should have little difficulty figuring out how to relocate the Monitor program!)

SIDELISTER

Mel Beckman, 717 West Broadway, Winona, MN 55987, developed this program and the following narrative:

One inconvenience of pocket computers that I had resigned to put up with is the hard to read program listings produced on those teeny weeny printers. Recently it occurred to me that Sharp had nearly provided an alternative to this in the form of sideways printing. The ability to print sideways going up, down or inverted is provided by the ROTATE command. ROTATE 0 causes text to print normally. Using ROTATE 1 turns the output 90 degrees clockwise; ROTATE 2, 180 degrees; and ROTATE 3, 270 degrees. After a ROTATE has been executed, all subsequent LPRINT directives will print in that orientation. You might expect that entering ROTATE 1 followed by LLIST would cause the program listing to print sideways. The nice thing about such a listing would be the ability to print long lines without those eye-jogging continuations that occur when long lines have to be printed down a column. Alas, it is not that simple.

Where There Is a Will. . .

A somewhat clumsy method of obtaining a sideways program listing is to write a small basic program that PEEKs at the entire program and prints it using ROTATE and LPRINT. That is what this program does. It is really not an ideal solution to the problem, but it does produce a sideways listing. The program is designed to be appended to the end of a user's program. Thus, it uses high line numbers (64000+) to avoid conflict with the user's code. The output may be printed in either CSIZE 1 or CSIZE 2. The former permits up to 84 characters per line and is nice for archival listings. CSIZE 2 prints up to 42 characters per line and is good for debugging sessions. The program automatically advances to a new "page" (after 20 lines at CSIZE 1 or 10 lines at CSIZE 2). It also correctly continues unusually long statements onto the next line when necessary.

Program *Sidelister*

```
64000:"L"REM  SIDE
      LISTER
64025:Z=2:X=212-Z*
      6
64050:GRAPH :
      GLCURSOR (X,
      0):SORGN :
      ROTATE 1:
      CSIZE Z
64100:X=0:L=0:C=0:
      M=0:Q=STATUS
      2-STATUS 1:
      LPRINT "*TOP
      *";TIME
64105:GOSUB 64200
64110:IF PEEK (Q)=
      &FFEND
64115:LPRINT USING
      "######";
      PEEK (Q)*256
      +PEEK (Q+1);
      ":";
64120:Q=Q+3:C=C+7
64125:IF PEEK (Q)=
      &0DLET Q=Q+1
      :GOTO 64105
64130:GOSUB 64400
64135:GOTO 64125
64200:IF C>MLET M=
      C:IF M>(84/Z
      )LET M=84/Z
64205:C=0:X=X-Z*10
      :GLCURSOR (X
      ,0)
64210:L=L+1:IF L=
      INT (20/Z)
      GOSUB 64300
64215:RETURN
64300:L=0:X=0:Y=-(
      Z*M*6.4):
      GLCURSOR (X,
      Y):SORGN :M=
      0:RETURN
64400:IF PEEK Q>&E
      5GOTO 64450
64405:IF C>(84/Z)
      GOSUB 64500
64410:LPRINT CHR$
      PEEK Q:C=C+1
      :Q=Q+1:
      RETURN
64450:U=PEEK (Q):W
      =PEEK (Q+1):
      U=&C053
64460:N=PEEK UAND
      &0F
64470:IF (PEEK (U+
      N+1)=U)AND (
      PEEK (U+N+2)
      =W)GOTO 6449
      0
64475:U=U+N+5:IF U
      >&C348LET U=
      &B054:GOTO 6
      4460
64480:IF (U<&C053)
      AND (U>&B0E2
      )LPRINT "? "
      ;:Q=Q+2:C=C+
      2:RETURN
64485:GOTO 64460
64490:IF (C+N+1)>(
      84/Z)GOSUB 6
      4500
64495:FOR J=1TO N:
      LPRINT CHR$
      (PEEK (U+J))
      ;:NEXT J
64498:LPRINT " ";:
      C=C+N+1:Q=Q+
      2:RETURN
64500:GOSUB 64200:
      LPRINT "
         ";:C=C+7:
      RETURN
STATUS 1        739
```

How It Works

The first task of Sidelister is to determine where the program it is going to list starts in memory. STATUS 2 less STATUS 1 provides that value. Sidelister thus starts PEEKing at that address and stops when it finds a hexadecimal value of FF as that is the value used to signify the end of a program.

BASIC statements are not actually stored character-for-character in the PC-1500/PC-2. To conserve space the line number is saved as a two-byte binary value and keywords are converted into two-byte "tokens". The format of such a compressed BASIC statement line is:

2-byte line #	length byte	statements	&0D

The text length byte shown in the diagram is used by the BASIC interpreter to enable it to rapidly scan forward through a program. Since the Sidelister program will be processing every program line, it can ignore this byte. The code &0D at the end of every program line represents an ASCII carriage return. It is used by Sidelister to advance to the next line to be printed.

Tokens are expanded back into keywords using the keyword/token table stored in ROM. This is the same table that the BASIC interpreter uses. There are actually two tables. One starts at address &C053, the other at &B054 (for the CE-150 printer/cassette statements). Each table entry is formatted as follows:

length byte	keyword	token code	address

The first byte of the table entry contains the length of the keyword in the four low-order bits. The high-order bits are apparently used as some sort of flag. Extracting the four low-order bits is accomplished by using an AND with the value &0F. The expanded keyword appears next in the table, followed by the two-byte token value used to represent the keyword. Tokens always begin with a byte greater than &E5 which makes them easy to identify. The last two bytes of each entry contain the address of the ROM routine that processes the statement.

A simple BASIC loop is used to scan these tables and translate the tokens into corresponding keywords for the listing. A blank is added to the end of each keyword for spacing.

Referring to the Sidelister program listing, lines 64000 – 64100 perform "housekeeping". Note variable Z in line 64025. Set this to select the CSIZE value (1 or 2) desired. While values of 3 through 9 could conceivably be used, the listings would be unduly large. Lines 64105 - 64135 constitute the main program loop. It extracts and prints the line number, checks for a carriage return code, and terminates upon encountering an &FF byte. The "new line" routine begins at line 64200, the "new page" at 64300, and the print routine at line 64400. The print routine fetches a byte, determines if it is a token that needs expansion, expands it if required, then prints appropriately. Line 64500 is used to continue long statements onto the next line.

Is It Practical?

After comparing various program listings in vertical and sideways formats, I strongly prefer sidelistings. The elimination of a lot of continuation lines improves readability. Debugging a program from a sideways listing is less strenuous on the eyes. The logical flow of a program seems more apparent. It is nice to be able to fold a listing between "pages" thereby eliminating annoying creases in the middle of a line. The horizontal arrangement is easier to scan than a skinny vertical printout. I would use this listing format exclusively if it could be obtained without undue effort.

Earlier I stated that this was not the ideal solution to the sidelisting problem. The best solution would be to have the capability provided in ROM. Maybe some future version of the unit will include this feature. The program suffers from a number of disadvantages that make practical application doubtful in its present form. Not the least of these is its slow execution speed. Looking up tokens with a BASIC search routine is not quick (taking 1 – 8 seconds each). Another drawback is having to append the program at the end of each program that is to be so listed. This burden can be eased by merging the routine from tape, but the code still utilizes valuable memory space.

However, I did not write Sidelister as a practical answer to the problem. It was an experiment; a tool to investigate the usefulness of such listings. In this capacity it has served well. The program is functional, but not efficient. It was developed using the simplest methods available without regard to execution speed or memory usage. This is a viable way to approach many software projects: programs can always be "tuned up" later; get them running first!

Immediate improvements readily come to mind. Somewhere in ROM is a routine that will return a keyword when given a token as input. Finding this routine would considerably reduce the execution time and cut the program size perhaps in half. There may well be a ROM routine that returns expanded statements, line-by-line. If not, machine-language routines could accomplish tremendous speed enhancements. I do hope that other ingenious persons will take the outline presented here and produce a Son of Sidelister. Perhaps additional features, such as the ability to print just selected lines or skip to a new page could be added.

If nothing else, perhaps the information disclosed about the internal storage of the token tables and BASIC program storage may inspire others to create additional programming tools!

Example *Output from Sidelister Program Using CSIZE 1*

```
*TOP* @1814.5888
 64000:"L"REM SIDELISTER
 64025:Z=1:X=212-2*6
 64050:GRAPH :GLCURSOR (X,8):SORGN :ROTATE 1:CSIZE Z
 64100:X=0:L=0:C=0:M=0:Q=STATUS 2-STATUS 1:LPRINT "*TOP*";TIME
 64105:GOSUB 64200
 64110:IF PEEK (Q)=&FFEND
 64115:LPRINT USING "######";PEEK (Q)*256+PEEK (Q+1);" ";
 64120:Q=Q+3:C=C+2
 64125:IF PEEK (Q)=&0DLET Q=Q+1:GOTO 64105
 64130:GOSUB 64400
 64135:GOTO 64125
 64200:IF C>MLET M=C:IF M>(84/Z)LET M=84/Z
 64205:C=0:X=X-Z*10:GLCURSOR (X,0)
 64210:L=L+1:IF L=INT (28/Z)GOSUB 64300
 64215:RETURN
 64300:L=0:X=0:Y=-(Z*7*6.4):GLCURSOR (X,Y):SORGN :M=0:RETURN
 64400:IF PEEK Q>&E5GOTO 64450
 64405:IF C>(84/Z)GOSUB 64500
 64410:LPRINT CHR$ PEEK Q;:C=C+1:Q=Q+1:RETURN
 64450:U=PEEK (Q):W=PEEK (Q+1):U=&C053
 64460:N=PEEK (U)AND &0F
 64470:IF (PEEK (U+N+1)=U)AND (PEEK (U+N+2)=W)GOTO 64480
 64475:U=U+N+5:IF U>&C348LET U=&B054:GOTO 64460
 64480:IF (U<&C053)AND (U>&B0E2)LPRINT "? ";:Q=Q+2:C=C+2:RETURN
 64485:GOTO 64460
 64490:IF (C+N+1)>(84/Z)GOSUB 64500
 64495:FOR J=1TO N:LPRINT CHR$ (PEEK (U+J));:NEXT J
 64498:LPRINT " ";:C=C+N+1:Q=Q+2:RETURN
 64500:GOSUB 64200:LPRINT "          ";:C=C+2:RETURN
```

TI CC-40 *(concluded from page 1)*

plotter and a Wafertape digital tape drive. Other add-ons, scheduled to be available by the third quarter of 1983 include: a wand input device, modems, printers and a black and white TV monitor. These devices are also said to operate with the new TI-99/2 Basic Computer, and, using an appropriate adapter, will work with the TI-99/4A Family Computer.

List prices of the three immediately-available peripherals are: $99.95 for the RS-232 interface, $199.95 for the 4-color printer/plotter, and $139.95 for the Wafertape drive.

The Wafertape drive will be capable of transferring data in digital format at approximately 1 kilobyte per second. This is considerably faster than most other portable systems now on the market.

Texas Instruments has also announced an array of 22 software application packages as being immediately available, with another 53 packages planned for introduction by the third quarter of 1983. Some software will be supplied on Wafertapes at $19.95 each. These include: Elementary Dynamics, Regression/Curve Fitting, Pipe Design, Production and Planning, Inventory Control, Electrical Engineering, Thermodynamics, Photography and more. Other software packages will come in plug-in ROM cartridges. These include: Mathematics, Finance, Perspective Drawing, Statistics, Business Graphics, Nonparametric statistics, and Advanced Electrical Engineering (at $59.95 each); an Editor/Assembler ($124.95) and two games packages (at $39.95 each).

THE SHARP PC-1251

While Sharp Electronics is showing off the PC-1250 to U.S. customers, (see page 1, Issue 21 of *PCN*), the rest of the world is apparently being treated to the model PC-1251. What is the difference between the two models? Just this: the 1251 leaves the user about 3,700 bytes of memory for program and data storage, while the 1250 provides a mere 1,400. Otherwise, the two units are identical, separated only by approximately $50.00 in price.

Does the extra memory make a big difference? You bet it does! The 3,700 bytes in the 1251 gives room to utilize the two-dimensional arrays that can be defined on both it and the 1250. The extra memory also permits about 2-1/2 times as many programs, routines or data to be instantly available!

During the past few weeks *PCN* has had the opportunity to try out a PC-1251. It does appear to be upwards compatible with the original PC-1211. In particular, the designers have handled the arrays properly (something they failed to do in the PC-1500) so that variables A – Z may also be referenced as array elements A(1) – A(26). This array can then be extended using additional elements starting with A(27), etc. Additionally, you can get away with using implied multiplication (with AB being interpreted as A times B) since the machine only provides for single character variable names. (Now, however, you can also have similarly named array elements, such as B(0), B(1), B(2), . . . , etc.) The upshot of this is that you can simply load in programs originally created for the PC-1211 without alteration and have them execute 5 to 6 times as fast! Previous PC-1211 owners should be delighted. Of course, you can then expand those programs to take advantage of some of the enhanced capabilities of the 1251. Perhaps most notable amongst the enhancements are the DATA, READ and RESTORE statements that so many people complained about missing on the earlier PC-1211.

Another nice feature of the 1251 is that it really does fit in a pocket, even a shirt pocket, quite comfortably. The keys are tiny, but spaced far enough apart so as to permit relative ease in the inputting of data or programs. The numeric keys are larger than the alphabetical buttons and are arranged in the traditional calculator format to facilitate rapid keying of numeric information.

For *PCN* afficionados, at least, the PC-1251 at a list price of $159.95 would seem to be a better deal than the PC-1250, even though the 1250 has a list price of just $109.95. It is too bad that the 1251 is not currently sold in the U.S. Could Sharp be persuaded to change their minds?

P.O. Box 232, Seymour, CT 06483

FROM THE WATCH POCKET

If you are having any problems with the Histogram program published in Issue 21 of *PCN*, then chances are you have a version A04 ROM. Drop *Russ Doughty* a SASE (self-addressed, stamped envelope) if you would like a new listing of the program that includes tape save/retrieve capability. His address is given at the start of the text portion of the article in Issue 21.

Jon Heath has a tip especially for PC-1 users: If you run out of memory when writing a lengthy number-crunching program, leave off the usual PRINT statements and their memory-consuming captions. Use the ability of the PC-1 to output data when you key in a variable name, such as X, Y, Z or A(12) followed by the ENTER key. It is almost as convenient, since you usually know what each variable represents. And, you can often save enough memory to squeeze in an extra formula or two.

Thomas Cox also wants to remind PC-1211/PC-1 users that you can merge programs using the CLOAD1 "NAME" command. However, these merged programs can only be accessed using unique labels, such as A, B, etc., whether being executed (in the DEF mode) or referenced by other routines (such as GOSUB "A"). I seem to recollect that their were some early units that did not have this capability. I guess the best way for you to find out if yours does is to give it a try!

If the new PC-1250/1251 turns out to be popular, there should be lots of software ready to go. Virtually all PC-1211/PC-1 software should run without modification on the new units. What is more, PC-1/1211 programs saved on your own cassette system should load through the CE-125 right into a 1250/1251. (Note that Sharp cautions that tapes prepared on other machines may not load. Thus, the spec's must be pretty tight or they are guarding their flanks!)

Some people seem to be less than happy with the information that comes with the CE-158 RS-232C interface. Seems the instructions on preparing connectors for various types of equipment are less than clear. On the other hand, those that have managed to get gear connected say the unit works great. Maybe a few of those that have gotten good results can share their findings for the benefit of those still struggling?

If you need to do some experimenting with the signal wires going to/from the CE-158 via DB-25 connectors, you might be interested in the Model 51 MINI-PATCH Box from **Remark Datacom, Inc.**, *4 Sycamore Drive, Woodbury, NY 11797*, phone (516) 367-3806. They make a "poormans" breakout box priced at $37.95 that allows you to reconfigure RS-232 connectors using jumper plugs.

Steven Tripp reports that over in Japan the PC market is really getting hot with a number of Japanese firms such as Toshiba, Epson and NEC busy supplying the field. One of the most interesting units available over there is NEC's PC-2001 which touts a 2-line, 40-character liquid-crystal display! The unit has a 36K ROM operating system and 8K of RAM expandable to 16K. Some of these units may eventually be exported to the U.S. Steven also indicates that Epson has a dual floppy disk drive available for the HX-20 over in Japan.

Hewlett-Packard has dropped prices on several peripherals that can be used with the HP-75C. The HP-82161A Digital Cassette Drive has been reduced from $550 to $450, the HP-82162A Thermal Printer/Plotter declined from $495 to $450 and the HP-82163A Video Interface (for U.S. equipment) from $295 to $225. HP has also announced the introduction of the HP-75 Text Formatter software module to be made available in March at less than $100.00. The software is intended to make it easy to produce short letters and memos using the HP-75C. In addition to providing formatting options, such as left-and-right justifying, letters may be customized by inserting names and addresses anywhere in the letter. Contact you local HP sales office for additional information regarding this software module.

No new word yet regarding the release of ROM modules for the Sharp PC-1500. About seven modules are past-due and lots of *PCN* readers are anxiously awaiting their arrival.

Radio Shack has reduced the price of the PC-2 and the Printer/Plotter Interface. However, Sharp's list price for the PC-1500 is still lower. It looks like the usual price reduction curve is well under way.

– Nat Wadsworth, Editor

POCKET COMPUTER NEWSLETTER

 — Issue 23 March

RADIO SHACK OFFERS NEW LOW-PRICED PC-4

Radio Shack has expanded its trade marked TRS-80 brand of pocket computers with a new low-cost model designated the PC-4. The list price of the 3/8 by 6-1/2 by 2-3/4 inches pocketable unit is just $69.95!

The PC-4 comes equipped with 544 bytes of memory programmable by the user in a dialect of BASIC offering 25 commands and 15 functions. Up to 10 different user-labeled programs can reside in memory at one time. It is claimed that the new PC can also accept character strings of up to 30 characters.

The pocket computer, which looks as though it may have been custom designed by Casio, sports a QWERTY style keyboard with 53 keys for alphabetic input including a 10-key numeric pad. The short 12-character liquid-crystal display can scroll horizontally to present up to 62 characters in a line. Lower case characters are said to be accessible through the use of a special mode. The PC includes a number of mathematical functions including trigonometric and reverse-trigonometric, radians, gradians, log, exponent, square root, angular conversions and absolute operations. The device is provided with Edit and Debug modes to facilitate efficient programming.

The PC-4 will operate from power provided by a pair of lithium batteries which are not included in the price of the basic unit. An automatic shut-off feature is built-in to maximize battery life.

Users who wish to extend the capabilities of the PC-4 can install an optional one kilobyte RAM module.

A cassette interface priced at $39.95 will be available for storing/retrieving programs using magnetic audio tape. Another peripheral, the PC-4 Printer, enables the PC-4 to operate as a printing calculator and print information at 20 characters to a line. A printing rate of 60 lpm is claimed for the PC-4 Printer which has a list price of $79.95. The PC-4, cassette interface and printer can be joined together to form a single, easily-transported unit measuring approximately 1-1/2 by 6-3/4 by 7 inches. The system can be protected in a padded vinyl carrying case available as an accessory at a list price of $7.95.

Another *Personal Information*™ product from SCELBI Publications.

EPSON HX-20 FOLLOW-UP

Here is *David Motto*'s follow-up to the preliminary report he gave on the Epson HX-20 which appeared in Issue 21 of *PCN*.

I have now received the entire basic package for the Epson HX-20. The BASIC manuals arrived about a week ago, and the carrying case a few days after that. I also received a couple of "Special Free Introductory Offer" program cassettes.

There were a few surprises in the BASIC manuals. There are two manuals; the first is a Tutorial and second is a Reference manual. They are of about equal thickness. Each contains an index to both volumes.

Unlike the Operations manual, the Tutorial is written in a fairly concise (not condescending or cute) style. There are about 24 program examples to help teach the novice how to program. Experienced BASIC programmers should study this text to learn some of the idiosyncracies of this version of Microsoft BASIC. The Reference manual contains 14 appendices. Appendix L is over 170 pages long. It contains a description of each BASIC keyword and its use.

Some of the commands and functions are different than I expected from this machine. For instance, the RAM file area commands are welcome (and vital).

The HX-20 will let you put up to five separate programs in memory at one time (depending upon space requirements). You can change from one area to another by using the LOGIN command. The problem with this is that variables assigned to a particular program are wiped out when you LOGIN to the new area.

The solution to this problem is the RAM file space, an area which is not cleared by a LOGIN directive. Values stored there can be passed to another program. To set up the RAM file space for input or output, all you have to do is define the record size and the offset within RAM space where the first record starts. Then, the GET% and PUT% commands will allow input and output, respectively, to the file. If the current RAM file space isn't big enough to hold all of the records, it can be changed by the CLEAR command.

Other nice features include: the LINE command, which draws a line between any two points on the screen. Lines other than vertical or horizontal are stepped. The LOCATES command locates the visible screen within the virtual screen area. The SCROLL command has three parameters: the speed at which to scroll the screen (similar to the PAUSE key), the horizontal-scrolling enable flag, and the x-step and y-step values that are executed when hitting a cursor key while holding down the CTRL key.

The KEY LIST and KEY LLIST commands display or print the current definitions of the 10 Program Function keys. It was nice of Epson to include these features. They could save a lot of trouble.

In the MON commands, the K instruction allows you to define several keystrokes which will be executed each time you turn on the unit. WIND and TAPCNT allow you to move the optional microcassette tape drive to any tape counter reading.

The manuals were written by Kenneth Skier of SkiSoft, Inc. I compliment him on them, but I have one small complaint. The page numbers in Volume 2 pick up where the numbers in Volume 1 leave off. Sometimes it is difficult to find a page as easily as it would if they were numbered in a different manner.

While on the subject of software, I should mention the cassettes I received with my manuals. The programs were written by ArtSci, Inc. Some of them show the useful applications of the Epson BASIC command set and the four-line display. The cassette of games has a lot of good graphics demonstrations. The other cassette of "Personal Productivity Software" shows what can be done with the printer, clock, string and calculation commands. I personally don't like any of the programs in either package. I think I could write them better. But, they are good for demonstration purposes and some techniques may be learned from them.

In conclusion, I am pleasantly surprised by the Epson entry into the portable computer marketplace. They seem to be going after a specific market. I think their unit is admirably suited for first-time users and students. I am looking forward to the industry's response to the HX-20. It should be very interesting.

THE SHARP CE-125 PRINTER/MICROCASSETTE INTERFACE

For awhile I thought that Sharp may have produced the first perfect peripheral for pocket computers. The CE-125 combination printer and microcassette recorder is attractive, small and lightweight. It certainly beats the earlier printer/cassette interface for the PC-1211. However, it does fall short of being perfect.

The microcassette portion of this unit is quite nice. The controls operate with a light touch. An accurate (mechanical) tape counter makes it easy to relocate programs on a tape (provided you make note of the counter value). A switch (REMOTE ON/OFF) is provided so that you may place the tape unit under manual or computer control.

The microcassettes used in this machine measure about 1-3/8 by 2 by 1/4 inches. A surprising amount of programs and/or data can be stored on a single cassette. For example, 20 programs ranging in length from several hundred to over a thousand bytes are provided on the cassette that comes with the interface. They all are stored on one side of the cassette, with perhaps 1/3 of that side of the tape remaining unused! Thus, some 50 to 60 typically-sized programs can be stored using both sides of a single cassette. You can shove five or six of these cassettes into a shirt pocket; thus, you can travel around with two or three hundred programs in your pocket! You can protect one or both sides of a cassette from accidental erasure by removing a plastic tab. Should you ever want to re-record after the tab has been removed, you can just place a piece of tape over the knock-out.

Another nice feature about the cassette interface is that you can use it to recover programs from a regular cassette. That is, programs recorded using the earlier PC-1211 on a standard cassette recorder can be transferred to the PC-1250/1251 through this interface. Note that this is a "playback only" feature. You cannot record programs onto an external recorder. While the manual warns that you should only attempt to recover tapes using the same machine as that used to record, it appears that you can often recover data produced by other recorders. For example, I succeeded in recovering a number of programs from Radio Shack program libraries originally produced for the PC-1 as well as my own tapes that had been made with a variety of recorders. Apparently, Sharp doesn't want to guarantee successful program loading across the gamut of available recorders (probably because of potential tape head skewing problems), although chances are good that many professionally duplicated tapes can be loaded from an external recorder. Thus, most of the material previously provided for the Sharp PC-1211 and the Radio Shack PC-1 will be immediately usable on the PC-1250. That is sensational news! (Remember, the PC-1250 typically executes PC-1211 code five to six times faster. Those programs that seemed interminable on the earlier model are really cut down to size by this new machine.)

The printer portion of the interface consists of a 24-character wide thermal printer. The rolls of paper are small. Don't plan on listing a whole bunch of programs while you are on the road unless you stuff a handful of paper rolls in your pocket. Since the paper is specially treated thermal material, it will be difficult to find cheap substitutes. Considering the fact that you no longer have to buy inked ribbons, however, it is probably a fair trade-off.

The dot-matrix printing is clearly legible. Indeed, there is less variation in character definition than on the PC-1211's mechanically driven printing head. The 24-character wide column makes listings a little easier to read when compared to a 16-column wide listing. While Sharp uses the advertising ploy that the printer width, being the same as that of the LCD display, makes it easier to use the unit, this only applies to program data outputs. Program listings of long lines still must wrap to several lines and cannot be directly compared to the scrolling LCD when checking listings.

The reason I was not able to rate the unit as "the first perfect PC peripheral" is because, surprisingly, the printer is not really "silent." The mechanism used to advance the printer to the next line makes enough noise so that I would be embarrassed to use it, say, in a public library or a classroom. Too bad. If they could just quiet down that linefeed mechanism, you could use the unit unobtrusively just about anywhere.

(continued on page 8)

26K OF RAM IN THE SHARP PC-1500 OR RADIO SHACK PC-2

The project discussed in this article should only be attempted by experienced electronic technicians. While the staff of PCN has successfully implemented this memory expansion on a Sharp PC-1500, it is our opinion that the procedure could best be described as risky. The amount of risk is related to the degree of experience one has had working with micro-electronics. Anyone attempting this project should be well aware that a mistake could result in damage to the pocket computer and/or memory module(s). Furthermore, the very act of attempting to install this modification will void any warranty on the part of Sharp or Radio Shack. Thus, if after reviewing the article, you do not feel you have the experience to perform the operations required, we suggest you have patience. Real 16K memory modules may be available sooner than you think!

It is possible to install 16K bytes of CMOS RAM inside the Sharp PC-1500 or Radio Shack PC-2. The installation does not require cutting any circuit traces or changing any chips. The module slot on the back will still accept any RAM/ROM plug-ins as long as they do not use any addresses below &4800. Thus, a 4K or (slightly modified) 8K module can be plugged in, making available up to 26,426 bytes of BASIC program space. The installation requires about 25 hours work and two 8K modules (CE-155).

One of the very nice features of the Sharp is that, each time a hard (15 second) reset is done, the computer automatically checks itself to see how much contiguous memory is installed and sets the pointers accordingly. There are no switches to set manually. Thus, memory can start at address 0 and extend to at least &6800 (hexadecimal 6800 or 26,624 decimal).

The unexpanded Sharp has 2K of user RAM at address &4000 to &47FF. Adding a 4K module extends the range to &57FF. You would expect the 8K module to address the range &4000 to &67FF. It does not. Instead, the 8K module addresses a range from &3800 to &5FFF. For some unknown reason, Sharp did not connect the internally decoded Y4 NOT (S4) signal from the 40H138 integrated circuit to any pin of the module socket. Therefore, the upper 6K of an 8K module is selected by S1, S2, and S3, while the lowest 2K is addressed by another 40H138 (1 of 8 decoder chips) inside the module. This apparent oversight and resulting correction is very fortunate for us. The module decoder chip will be used to decode the addresses from 0 to &3FFF. This makes implementation of the 16K expansion easy.

In the Radio Shack PC-2 version, the S4 signal is connected by a jumper to the NC pin of the module socket. Never-the-less, the Radio Shack 8K modules which I have examined do not use it. They contain the same 40H138 for decoding, just as the Sharp version does.

The 8K RAM modules are not sealed shut. The upper and lower halves of the cases will readily snap apart to reveal the multilayer printed circuit board inside. The covers can be snapped back together with no damage. These covers should be kept on as much as possible to avoid damage caused by static electricity. Also, when making the modifications described here, you should use a 3-wire grounded soldering iron (Weller model WP-25-3 or equivalent).

The description that follows assumes that you want to put 16K inside the Sharp and plug in an additional 8K (giving a total of 26K). Before starting, be sure to test each of the 8K modules to verify that they work. You might find it useless to complain about a defective one after you have soldered it.

Opening the Case

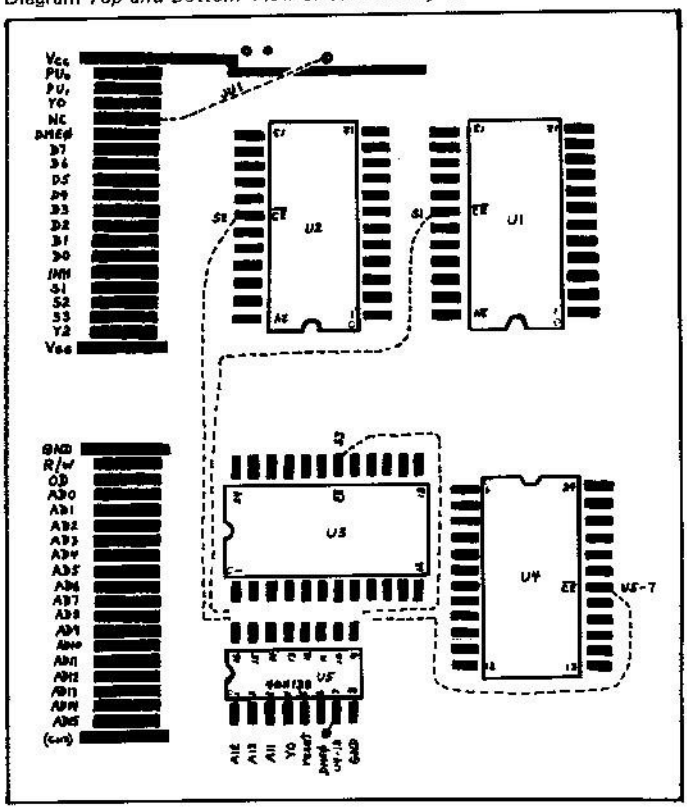
To access the internal circuit boards of the PC it is necessary to remove the back cover. This may be accomplished by removing a total of eight screws from the back of the unit. Five of these screws are visible on the back cover. Three more may be found inside the battery compartment. Once the screws have been removed you can gently separate the back of the unit from the front. They remain hinged together by a piece of flexible circuitry.

Add the S4 Jumper

This change is required only if you plan to use plug-ins in addition to the 16K being added internally. However, I strongly recommend doing it because the Radio Shack PC-2 version has it and some of their plug-in ROM's may not work in the PC-1500 if you don't put it in. If you have the Radio Shack PC-2, you can skip this step as a jumper will already be installed, though not at the same place.

On the main circuit board of the Sharp solder a jumper from the eyelet next to the screw of the 40 pin module socket to pin 11 (Y4 NOT) of the 40H138. This eyelet connects to the "NC" pin of the module socket. See the accompanying diagram.

Diagram *Top and Bottom View of 8K Memory Module*



Change Plug-In Module Addressing to &4800 to &67FF

An accompanying illustration shows both sides of a module printed circuit board. After opening a module, very carefully unsolder and lift pin 7 of the 40H138 (U5) chip away from the printed circuit board until it is parallel to the board. The easiest way that I have found to do this is to loop a piece of bare number 30 wire under the pin and pull gently on it while heating the solder joint. Next, connect a jumper (JU1) from the eyelet above U2 to the module pin designated "NC" in the diagram. Use solder sparingly and don't overlap the pin much. Most of the pin should remain shiny gold. Make a small V-notch in the cover so as to avoid pinching the wire. Replace the module covers.

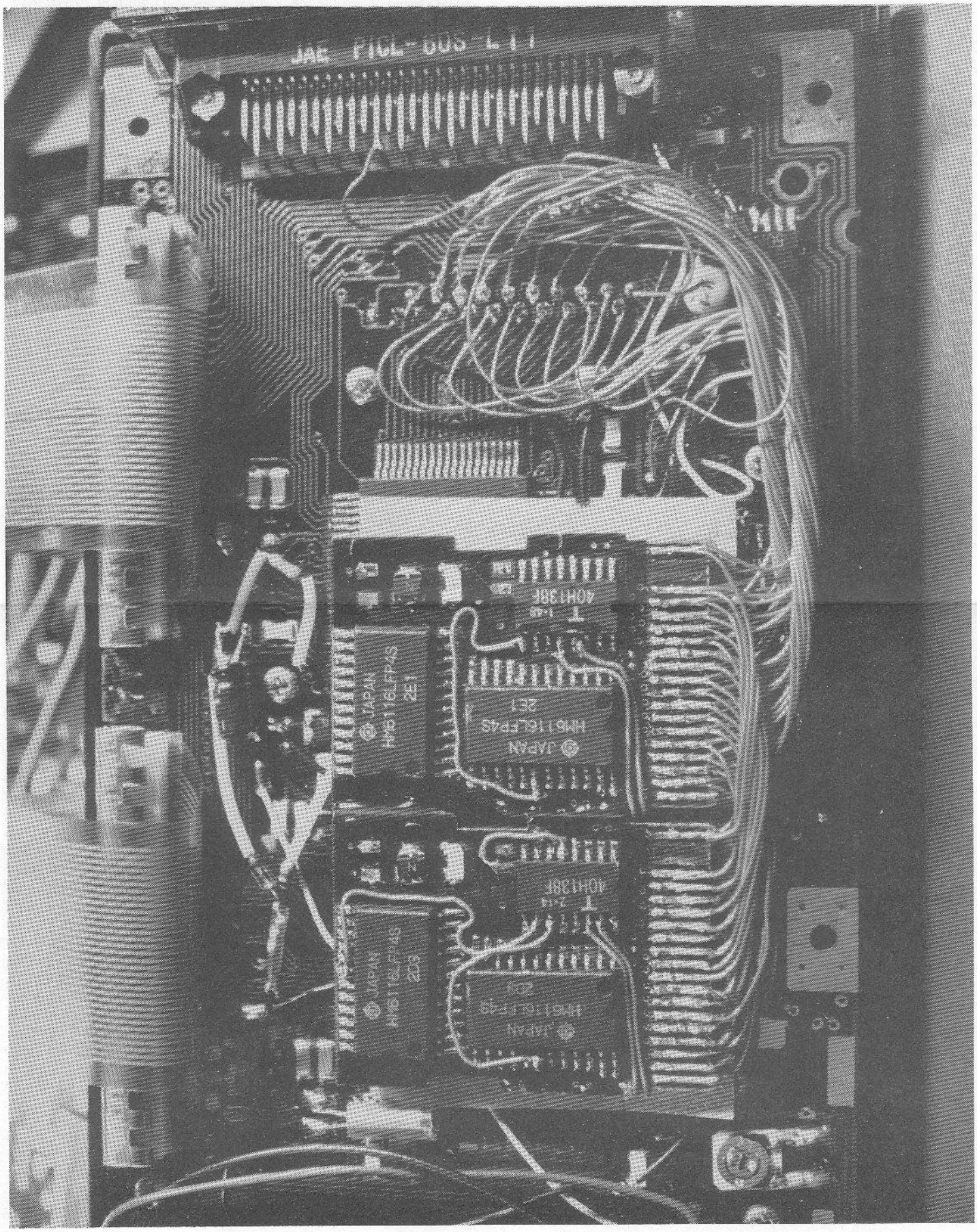
Plug the modified module into the module slot. (Always follow the usual precautions of taking a battery out and holding the reset for 15 seconds.) After NEW0, MEM should now give 10042 as it did before these changes. However, PEEK &7863 should now show &40 (64) instead of &38. STATUS 3 should yield &6800 (26624). These tests will show whether the address range has been moved.

When the tests have been satisfactorily completed, take out the module. Write the new address range on the label (&4800-&67FF). Put the module aside until all the rest of the modifications have been made.

Change a Module to Address &2000-&3FFF

Open another module. Unsolder and lift the CE NOT pins (pin 18) of U1, U2, and U3. Add three jumpers to connect the CE NOT pins (not the foil) of U1, U2, and U3 to decoder U5 pins 12, 13, and 9 respectively. Route the jumpers as shown by the dotted lines in the diagram. (Note that two of the jumpers will wrap around the edge of the board

Photo *Installation and Wiring of Memory Modules to the CPU Board*



from the bottom to the top sides.)

Without replacing the covers, plug the module into the module slot (U3, U4, and U5 visible). Install the batteries and execute a NEW0. MEM should now read 10042. PEEK &7863 should give &20 (32) and STATUS 3 should show &4800 (18432). Remove the module from the slot and set it aside. You may leave the covers off.

Change a Module to Address &0000-&1FFF

Open a third module. Unsolder and lift the CE NOT pins of U1, U2, U3, and U4. Add four jumpers to connect the CE NOT pins of U1, U2, U3, and U4 to the decoder U5 pins 14, 15, 11, and 10 respectively.

Once again, leaving the covers off and being careful to have the right side up, plug in this modified module.

MEM should now yield 7994. This is because the automatic memory check will count only contiguous memory. It will not see the 2K of memory at &4000 to &4800. PEEK &7863 should give 0. STATUS 3 should show &2000 (8192). Remove the module, leave the covers off and set it aside.

Prepare the 16K Assembly

Needless to say, space inside the PC is limited. There is not enough room for even one module with the module covers on as the covers add a significant amount of thickness. Without covers, however, two modules will fit comfortably side-by-side in the space between the CPU and ROM and the flexible printed circuits which connect to the second board.

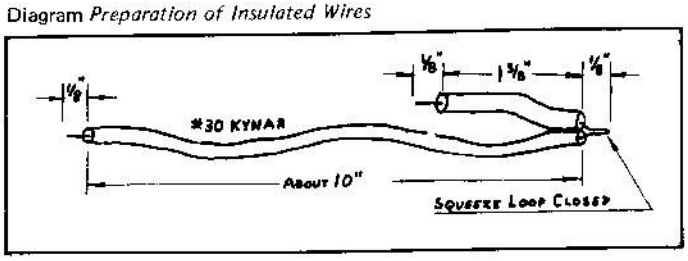
Epoxy the two coverless modules together, edge to edge, same sides up, so that both rows of connector traces are along one side as shown in the accompanying photograph. It doesn't matter which module is at the left or right. This step will make handling and soldering the 16K assembly a neat package which can be removed and transferred to a different computer. Remember, in a year or two, a new, shinier toy may come out.

When the epoxy has hardened, tin the eighty gold-plated connector traces. Use plenty of rosin flux to ensure good wetting and to avoid bridging between traces. The assembly will now be completely wired together before any of the leads are connected to the computer.

Wire the 16K Assembly

Using number 30 insulated wire (preferably Kynar or Teflon), prepare 29 wires, each about 8 inches long. Do this by stripping a one-fourth inch section of each wire, one and three-fourths inches from one end. Then strip one-eighth inch from each of the ends. Fold the quarter-inch section back on itself and squeeze together with pliers to form a tight hairpin as shown in the accompanying diagram.

Diagram *Preparation of Insulated Wires*



Place the epoxied module assembly in front of you with the U3, U4, U5 side up and the connectors facing you. Solder the hairpin of a prepared wire to pin AD13 of the righthand module. Solder the short end of the same wire to pin AD13 of the lefthand module. Continue soldering hairpins and short ends to the remaining sixteen pins to the left of AD13. (The three pins to the right of AD13 will not be used.)

At the end of each of the wires coming from GND, R/W, OD, AD11, AD12, and AD13, fasten a small piece of masking tape with the wire identification written on it.

Dress the wires neatly out to the right and twist them around each other so they form a neat spiral bundle for a couple of inches. This will keep the bundle flexible and compact. Tie a string or tape around the bundle to hold it together while you work on the other side.

Turn the assembly over and solder the remaining twelve prepared wires to pins Vcc, Y0, DME0, D7 through D0, and Vgg on both modules. This time mark the four long wires going to Vcc, Y0, DME0, and Vgg with masking tape. Finish this side the same way you did the first side, and make a separate spiral bundle. Arrange the bundles side-by-side so they are no thicker than the modules.

That completes the module assembly. A good way to protect and insulate it is to cut two pieces of plastic packaging tape measuring approximately 1½ by 2½ inches, and place one on each side of the assembly.

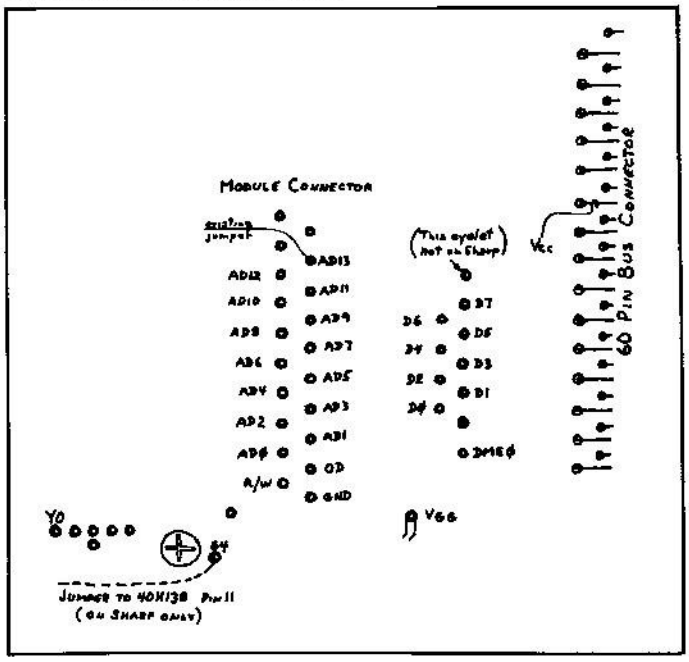
Connect the Assembly to the Computer

Notice that you were not directed to mark the individual data wires nor the address wires AD0 through AD10. Any data wire from the assembly can be connected to any data eyelet in the computer in mixed order. The same is true for the address lines, except for the three highest AD11, AD12, and AD13. These latter three should go to their corresponding points in the computer to facilitate troubleshooting if necessary.

Position the assembly over the CPU and ROM with the wire bundles on the side away from the flex circuit connections as shown in the photograph. Temporarily fasten it there with masking tape.

Start with the lower bundle of wires. Route each wire as neatly as possible, cut it to the right length, and strip 1/8 inch of insulation from the end. Then solder it to the proper eyelet, as shown in the accompanying photograph and diagram. Start with DME0, Vgg, and YO, then the eight data wires D0 through D7, and finally connect Vcc to the wire on the sixty pin connector.

Diagram *Connections to CPU Board*



Connect the wires from the upper bundle next. Starting with GND, R/W, and OD, cut, strip, and form a small loop around the appropriate pin. Squeeze the loop with needle nose pliers to hold it in place while you solder it. Next, connect the address lines AD0 through AD10 in any order. Finally, connect the last three address lines AD11, AD12 and AD13 to the proper pins. Pin AD13 already has one jumper connected to it. Be sure it stays in place when you solder your wire.

Remove the masking tape which temporarily held the assembly in place. Make sure nothing will be scraped or pinched as you reassemble the computer. There will be a very light pressure required as you push the two halves back together. This is the flex circuit pushing down on the assembly and will not cause any problem.

Checkout the Installation

After NEW0, MEM should now read 18234. PEEK &7863 should give 0. STATUS 3 should show &4800 (18432).

You may now plug in the modified 8K module that you prepared earlier. After NEW0, MEM should read 26426. PEEK &7863 should show 0 and STATUS 3 should yield &6800 (26624).

With the 8K plug-in module installed, and including the 1-½K of permanent memory and 16K of ROM, you now have 43-½K of memory or 51-½K if the printer-plotter is connected. So the next time someone asks how much memory your "calculator" has, I guarantee you'll see a surprised reaction when he finds out. You may have much more in a one-pound package than he has in his desktop computer.

Using the Expanded Memory

Entering NEW0 always sets the SOB (Start of BASIC) pointer at &C5 bytes above the lowest memory address (leaving 197 bytes for the soft keys). After you have installed this 16K memory expansion, NEW0 will set SOB to &00C5.

Most programs can be entered or CLOADed without regard to where the SOB pointer is set. However, if some of the programs you now use or acquire in the future address specific memory locations, but have not been written to be relocatable, then they may not run properly. For example, if a line renumbering program expects to read the value of the first BASIC line number of a program by using a statement such as:

```
10 N=PEEK &38C5 * 256 + PEEK &38C6
```

or some variation of this, then it will be looking at the wrong address in a computer that has this 16K memory addition installed.

In general, any time a program has statements containing PEEK, POKE, or CALL, you should examine it to see whether the author allowed for different memory starting addresses.

If the program is not relocatable, you can always run it by entering NEW followed by either &40C5 or &38C5, whichever is appropriate for the program, before you load it. This will cause the computer to effectively ignore the added memory.

A much better solution is to write new programs and to modify existing ones so that they are relocatable. One way to do this is by utilizing address &7863. This address (after a hard reset) always contains the upper byte of a two byte value equal to the lowest address available in the PC. For instance, in an unexpanded PC-1500, PEEK &7863 will show &40, indicating that the lowest memory address is &4000. After the 16K expansion, PEEK &7863 will give 0 because memory starts at &0000.

You can make a program relocatable by substituting PEEK &7863 for &40 or &38, wherever they occur. Thus, &38C5 becomes:

```
PEEK &7863 * 256 + &C5
```

Line 10 in the example could be replaced by:

```
10 S=PEEK &7863 * 256 + &C5
11 N=PEEK S * 256 + PEEK (S+1)
```

or, if you must put it all on one program line, it could be written:

```
10 N=PEEK (PEEK &7863 * 256 + &C5) * 256 + PEEK (PEEK
   &7863* 256 + &C6)
```

This method will always work if the computer is initialized with a NEW0. However, it may not work if some area of memory has been set aside for another purpose by using NEW followed by some value other than 0.

A more general method of making a program completely relocatable is to use the SOB pointer as a reference. You can always obtain the value of SOB by:

```
SOB=PEEK &7863 * 256 + PEEK &7866
```

or by:

```
SOB=STATUS 2 - STATUS 1.
```

Line 10 in the example could thus be replaced with:

```
10 S=PEEK &7865 *256 + PEEK &7866
11 N=PEEK S * 256 + PEEK (S+1)
```

or by:

```
10 S = STATUS 2 - STATUS 1
11 N = PEEK S * 256 + PEEK (S+1)
```

This is not as awkward or extravagant with memory as it might at first appear, because the line to calculate SOB (line 10) has to be used only once in the entire program. Thereafter, the program can simply refer to S whenever SOB is needed. This will probably shorten the overall program.

When this latter method is used to make the program relocatable, any legal value can be entered following NEW and the program will load and run properly.

If program authors will begin using this technique, they will not have to specify special instructions on how to load their programs, according to how much memory a user has installed. They can merely specify the minimum amount of RAM required by their programs. Then the programs themselves can take care of any variations in memory configuration. The programs could even automatically check to see how much memory was installed, and notify the user if there was not enough!

For the Future...

You may have noticed that U5 in the plug-in module now serves no purpose. It could be removed altogether. Perhaps someone will want to figure out how to put a tiny battery in that space, so that a module could be removed, put on a shelf, then reinstalled without losing its memory.

Having this much memory will make you painfully aware of just how slow the cassette interface is. It takes about forty minutes to load or save the entire memory. I hope someone discovers the secret of changing the baud rate. And wouldn't a three-inch disk drive be a perfect match?

Another possibility as the price of modules comes down would be to build two more 16K assemblies, mount them in the printer, and address them as &0 to &7FFF in the alternate address range. Programs could be moved from there to main memory as needed.

This impressive memory enhancement article was designed and described by: *Don. L. Carter, Boeing JW-26, 220 Wynn Drive, Huntsville, AL 35805.*

MEMORY MAPS

The Sharp PC-1500/Radio Shack PC-2 come standardly equipped with 2K of RAM (random access memory) at the hexadecimal addresses &4000 - &47FF. Part of this block is set aside for use by the Reserved Softkeys. The remainder is available for user programs.

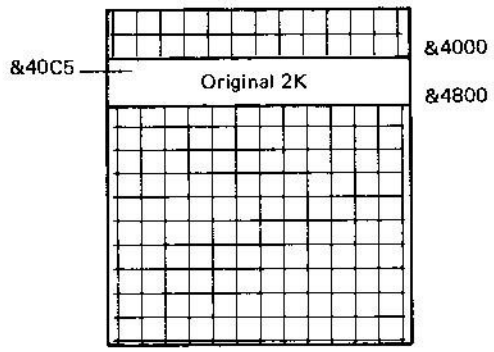
If you plug in a CE-151 (or equivalent) 4K memory module into the back of the PC, you will find that the new memory is addressed in the range &4800 - &57FF. No surprises there as the memory block simply picks up where the original 2K block ends.

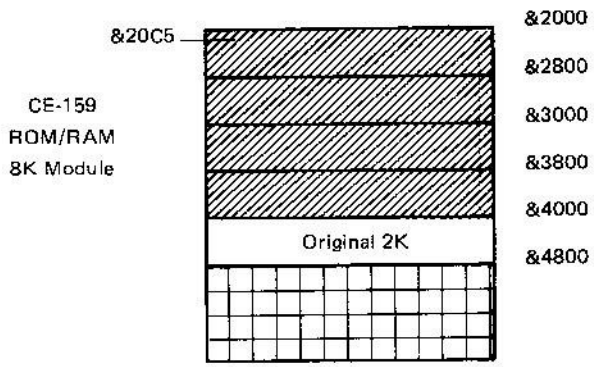
However, if you plug in a CE-155 8K memory module, you may be a little surprised to learn the following: the module effectively splits its memory into two separate address ranges to surround the 2K block provided in the basic unit! A 2K section of the module occupies the address range &3800 - &3FFF. Then, a 6K block picks up at &4800 and extends through &5FFF. The operating system provided in the PC's ROM is able to tell where RAM begins and ends. Thus, when a CE-155 module is installed, the PC changes pointers so that the area set aside for the Reserved Softkeys starts at the address &3800 instead of &4000.

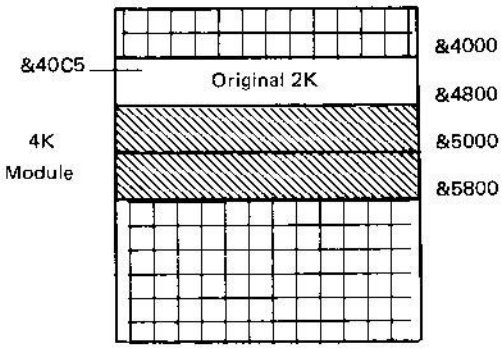
In the memory expansion article provided in this issue of *PCN*, Don Carter shows how a CE-155 memory module can be modified so that it operates as one continuous 8K block accessed as addresses from &4800-&67FF. This is done so that the 16K that is to be installed inside the PC will not overlap in the range &3800 - &3FFF. Again, the operating system of the PC is able to determine where RAM begins and to set aside an area for the Reserved Softkeys accordingly.

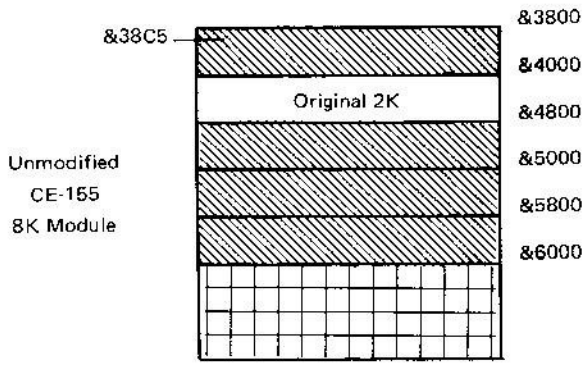
Sharp Electronics, however, has sprung another surprise with its CE-159 memory module. This is a special battery-equipped 8K module that can be configured to operate effectively as ROM or partitioned to operate partially as RAM and partially as ROM. Want to guess at the address range for which this module was designed? It is not the same as that of the CE-155, which one might presume. No, it uses the range &2000 - &3FFF!

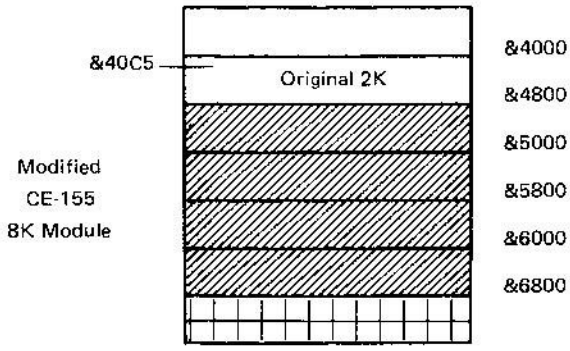
Figure *Memory Maps*











Now, what difference does it make as to what address range the various memory expansion modules might utilize? Well, if you only program in BASIC it will not make any difference. If you start using machine language routines, however, it can make a big difference. Many machine language programs are address dependent. For instance, the machine language portion of Norlin Rober's monitor program published in Issue 22 of *PCN* is designed to reside specifically in the addresses &3E4F - &3FFF. Once installed in memory at those addresses, the machine language instructions must be protected from interference by future BASIC programs, accomplished by using the NEW&4000 directive. The memory is effectively partitioned so that all BASIC statements are stored above the address &4000 and thus above the machine language routines.

Note that when this is done with a CE-155 module installed in the system, there will still be about 8K of RAM available for BASIC because the RAM elements extend through address &5FFF. Alas, however, if one performs the same directive with a CE-159 module installed, then a mere 2K of memory is left for BASIC! This is because all of memory in that module is addressed below &4000, leaving just the original 2K of PC memory available once the partitioning NEW&4000 directive has been given!

The upshot of these addressing differences among the various memory expansion modules is that users planning on employing address-specific machine language routines will have to plan their module purchases carefully.

EXPANDED DISPLAY ROUTINE

The accompanying program uses the graphics capability of the PC-2 to increase the display from 26 to approximately 40 characters. The message can be up to 80 characters long in which case it will wrap to the beginning of the screen. The message is placed in variable M$(1) and the extended display is invoked by GOSUB 100. ASCII characters 32 to 90 are covered. This includes A-Z, 0-9, and most of the special characters. Characters outside this limit, such as lower case, will cause an error. Numeric information must be in alpha format so STR$ X should be used to insert numeric data into the string.

Statements 1, 10 set up the graphic matrix. 100, 105 is the expanded display routine. 1000-1120 contains the graphic data (but can be put anywhere at the end -- or in a preprocessor that loads the matrix and then CHAINs in the program containing the expansion subroutine). The sample has a simple input routine at 500, 510 but the actual program can be placed where desired with the appropriate adjustment to statement 10.

This program submitted by: *H. David Jackson, 126 Smithfield Drive, Endicott, NY 13760.*

Program *Expanded Display*

```
  1:WAIT 0:DIM M$(
    1)*80
 10:DIM C$(58):FOR
    I=0TO 58:READ
    C$(I):NEXT I:
    GOTO 500
100:FOR I=1TO LEN
    (M$(1)):GPRINT
    C$(ASC (MID$ (
    M$(1), I, 1))-32
    )+"00";:NEXT I
105:GPRINT "":
    RETURN
500:INPUT M$(1):
    GOSUB 100
510:FOR I=0TO 400:
    NEXT I:GOTO 50
    0
 1000:DATA "0000",
      "2E","060806
      ","143E14"
 1010:DATA "246B12
      ","120824","
      142A34"
 1020:DATA "362236
      ","1C22","22
      1C","1C1C1C"
 1030:DATA "081C08
      ","3070","08
      0808","3030"
      ,"300806","3
      E223E"
 1040:DATA "3E","3
      A2A2E","2A2A
      3E","0E083E"
      ,"2E2A3A"
 1050:DATA "3E2838
      ","02023E","
      3E2A3E","0E0
      A3E"
 1060:DATA "14","2
      014","081422
      ","141414"
 1070:DATA "221408
      ","022A04","
      3A2A3E"
 1080:DATA "3C0A3C
      ","3E2A14","
      1C2214","3E2
      21C","3E2A22
      "
 1090:DATA "3E0A02
      ","1C2234","
      3E083E","3E"
      ,"10201E","3
      E1826"
 1100:DATA "3E2020
      ","3E043E","
      3E1C3E","1C2
      21C","3E0A04
      "
 1110:DATA "1C223C
      ","3E1A24","
      242A12","023
      E02","3E203E
      "
 1120:DATA "1E201E
      ","3E103E","
      360836","0E3
      80E","322A26
      "
STATUS 1          736
```


CE-125 *(concluded from page 2)*

That is about the only drawback I could come up with, however. The unit is quite comfortable to use. It is so lightweight that I often use the PC away from a desk by holding the entire printer/cassette interface with one hand while I punch in data or commands with the other. The rechargeable battery pack (capable of providing about 4 hours of continuous tape drive operation or some 2000 printed lines) takes me through a typical day with plenty to spare. The size of a relatively thin textbook, the unit fits comfortably in a briefcase, making it easy to keep at hand.

The unit is supplied with what I call a "miniature suitcase." It is a multi-compartmented affair with room for two accompanying manuals, extra cassettes, rolls of paper, the battery charger and a cable for use with an external recorder. Frankly, the carrying case is so large (relatively speaking) that I just leave it at home as a "supplies base" while I tuck the actual unit into my briefcase for normal transporting.

One of the manuals provides general operating instructions. The other contains twenty application programs, about half of which make use of the printer interface and all of which are supplied on an accompanying microcassette for easy loading. All-in-all, at a list price of $169.95 in the United States, the CE-125 is a nice package. Not perfect, mind you, but they are getting close!

THE ULTIMATE ROM?

The new PC-1250 has an interesting security feature: a PASS command that allows you to secure the contents of its memory. Anyone can use the program(s) stored therein, but only those with the appropriate password (containing up to seven characters) can list, modify or save the contents of memory on tape!

One enterprising person we have heard about is using this capability to provide "customized" computers. He sells the entire PC-1250 already loaded with specialized software that has been effectively "locked" using the PASS command.

The party we have heard about apparently only distributes his package by directly loading the program into the PC and "locking" it at the time of sale. He then provides a warranty regarding reloading of the program should accidental erasure occur. There is, however, perhaps an even better way to deliver the software. While the manual for the PC-1250 may be a little confusing in this regard to some, tests have indicated that you can write a program to a tape in a "protected" form!

The CSAVE command can be issued in the following format:

CSAVE "NAME","XXXXXXX"

where "NAME" indicates the file name assigned to the program and "XXXXXXX" signifies a password. A tape written using this directive can be loaded into a PC just by knowing its file name. However, it cannot be listed, modified, *or saved on another cassette* unless the password is known. Thus, by declining to give the password used at the time the tape is written, a vendor can distribute software while providing a measure of protection against casual unauthorized distribution!

This alone should have enormous implications for developers of software. When one considers one more factor, the future prospects become even more exciting! Consider the fact that within about a year, as manufacturing efficiencies increase, a PC such as the 1250 can probably be made to retail for about $50.00. That means a volume wholesale buyer will probably be able to purchase the basic unit at $25-$30 each. By loading in a customized program, the wholesale operator can, in effect, deliver an entire computer, complete with customized software, for about the same amount of money as it used to cost to provide a small amount of ROMs (when dealing in quantities of about one thousand pieces)! This should open the door to many smaller operators—the door that really needs to be opened in order to spill forth a great variety of practical uses for PCs.

So what are you waiting for? You now have the capability of providing customized ROMs, complete with the entire computer, input, output, display and operating system and delivering it at a price that many people are willing to pay for software alone! Talk about opportunity knocking...

POCKET COMPUTER NEWSLETTER

Issue 24 — April

THE RADIO SHACK TRS-80 MODEL 100 PORTABLE COMPUTER

Radio Shack, a division of Tandy, may have scored a considerable marketing scoop with the introduction of their new portable TRS-80 Model 100.

In a break from traditional marketing methods, whereby products are announced months before they become readily available, the firm kept details of the new portable under tight wraps. Meanwhile, in a carefully orchestrated fashion, the company arranged to get substantial supplies of the new product into its outlets around the country. On March 25, 1983, the company unleashed its tongue via a newspaper media blitz and announced that the new product was in stock at local Radio Shack outlets.

The slightly under four pounds (3 lbs., 13-1/2 oz.) portable measures 11-5/8 by 8-1/4 by 2 inches, covering an area approximately equal to a standard-sized piece of writing paper. Its most prominent feature is a large 240 by 64 pixel liquid-crystal display that measures close to 2-1/8 by 7-1/2 inches. This display can present text and special graphic characters in 8 rows by 40 columns. It can also be operated in a pure pixel-by-pixel graphics mode to create pictures, charts and graphs.

The area that remains beneath the display is occupied by a full-sized touch-typing keyboard. The five-row keyboard is further complimented by 16 special function keys directly beneath the display. Additionally, a subset of the regular keyboard can be designated to serve as a numeric input keypad.

Perhaps the most surprising feature of the Model 100 is a complete built-in modem and comprehensive terminal software which permits automated telephone number lookup and dialing, user-defined log-on procedures, file up-loading and down-loading, and other aids to facilitate its use as an intelligent remote terminal.

In addition to the communications software, the unit includes a BASIC operating system, a text editor complete with word-wrap, an address-maintenance program and an appointments scheduler. The unit is supplied with a real-time clock that provides the year, month, day, and time and is accessible by software. A sophisticated memory management system permits multiple user files to reside in memory simultaneously. These user files may be accessed by the unit's operating system or by other user-created programs.

The Model 100 is pleasingly designed to facilitate its use by novices. Whenever power is turned on the display is filled with a directory giving the date and time, the names of up to 24 programs or data files currently residing in memory, and the amount of memory that is still available for general use. To access a program or data file, the user simply positions a cursor over a directory entry and presses the ENTER key. Once a file has been selected, legends at the bottom of the display, associated with special function keys directly beneath the LCD, offer options to the user in menu fashion. (The menu offerings can be toggled on and off using a function key. Thus, experienced users can free up a line of the display by not using the prompts.)

For instance, when the user initially calls up the text editing program, the system prompts for a new file name. Once given, a press of the "label" key offers choices of saving or loading a new file, finding a point in the text, selecting a portion of the text for manipulations, copying or cutting text or exiting to the main directory. In an apparent effort to make computer operations more consistent with every day experience, commands like "cut" and "paste" are used instead of "delete" or "move".

The new Radio Shack entry has a powerful BASIC interpreter, perhaps the most powerful amongst portables at this point. Tailored by Microsoft, it contains such keywords as: ATN, BEEP, CALL, CDBL, CHR$, CINT, CLEAR, CLOAD, CLOAD?, CLOADM, CLOSE, CLS, COM ON/OFF/STOP, CONT, COS, CSAVE, CSAVEM, CSNG, CSRLIN, DATA, DATE$, DAY$, DEFDBL, DEFINT, DEFSNG, DEFSTR, DIM, EDIT, END, EOF, ERL, ERR, ERROR, EXP, FILES, FIX, FOR. . .NEXT, FRE, GOSUB, GOTO, HIMEM, IF. . .THEN. . . ELSE, INKEY$, INP, INPUT, INPUT #, INPUT$, INSTR, INT, IPL, KEY, KEY LIST, KEY ON/OFF/STOP, KILL, LCOPY, LEFT$, LEN, LET, LINE, LIST, LLIST, LINE INPUT #, LINE INPUT, LOAD, LOADM, LOG, LPOS, LPRINT, LPRINT USING, MAXFILES, MAXRAM, MDM ON/OFF/STOP, MENU, MERGE, MID$, MOTOR, NAME. . .AS, NEW, ON COM GOSUB, ON ERROR GOTO, ON KEY GOSUB, ON MDM GOSUB, ON TIME$ GOSUB, ON. . .GOTO, ON. . . GOSUB, OPEN, OUT, PEEK, POKE, POS, POWER, POWER CONT, POWER OFF, PRESET, PRINT, PRINT #, PRINT USING, PRINT # USING, PSET, READ, REM, RESTORE, RESUME, RIGHT$, RND, RUN, RUNM, SAVE, SAVEM, SCREEN, SGN, SIN, SOUND, SOUND ON/OFF, SPACE$, SQR, STOP, STR$, STRING$, TAB, TAN, TIME$, TIME$ ON/OFF/STOP, VAL, and VARPTR.

Many of these extended BASIC keywords are used to control the built-in communications and input/output capabilities. In addition to the modem, the Model 100 includes interfaces and connectors to: store and retrieve programs and data on an audio cassette recorder, drive an external printer (parallel "Centronics" interface), and connect to serial devices via an RS-232C (25-pin) connector.

The Model 100 is currently being supplied in two standard initial configurations: equipped with 8K of user RAM or 24K of user RAM. Both versions are said to be internally expandable to a maximum of 32K of RAM. The 8K unit is retailing for $799.00. The 24K unit is retailing for $999.00.

The unit is powered by four AA cells, which are said to power an 8K unit for approximately 30 days or a 32K unit for approximately 8 days. A low power light-emitting diode turns on when batteries need replacement. The unit may also be powered by an optional a.c. power pack.

The machine is supplied with an impressive 224-page (8-1/2 by 11 inch) operating manual. The manual is well organized, illustrative yet concise. A soft carrying case is also provided with the computer.

Surprisingly, batteries are not supplied with the unit. You will have to spring for 4 AA cells or an a.c. adaptor to power up the machine. Additionally, while the direct-connect modem and terminal software is supplied as part of the computer, the connector that goes to a telephone is an "optional accessory". Radio Shack is charging $19.95 for this special connector, but includes manuals and introductory time on the Compuserve and Dow Jones information networks as part of the package. You will also need to shell out $5.95 for a connector that mates to a cassette recorder if you want to store/retrieve information on magnetic tape.

The Model 100 appears to be an impressive entry to the compact portable computer market. Take a look at it.

PC-1250/51 PROGRAMMABLE IN MACHINE LANGUAGE?

It does appear that the new PC-1250 (and the PC-1251) may be programmable in machine language! If so, this may help boost the low-priced unit to the stratosphere of sale levels.

Preliminary explorations have revealed that the unit does respond to PEEK and POKE directives, even though these capabilities are not indicated in instructions accompanying the unit. The PEEK directive takes the format: PEEK X where X is an address in the range 0 - 65,535. The POKE directive has the form: POKE X,Y where X represents an address value and Y indicates a value in the range 0 - 225 which is to be stored at the stated address. An alternate form of this directive is: POKE X,Y, . , . , . , where successive values are stored in consecutive locations in memory.

Memory Mapping

If you use PEEK to start examining areas in memory, you will find some surprises. For instance, peeking into an address that does not contain any RAM or ROM results in the "page" of the address used being echoed. Thus, PEEK 0 will return the value 00. (Oh, yes, you can use decimal or hexadecimal notation. Preceding a value by an ampersand (&) causes it to be interpreted as a hexadecimal value, just as is the case on a PC-1500.)

There does not appear to be any RAM or ROM in the address range 0 - 16383 (0 - &3FFF). Addresses 16384 - 32767 (&4000 - &7FFF) appear to contain ROM directives. Addresses 32768 - 34815 (&8000 - &87FF) apparently access user RAM. However, the same memory elements can also be reached using address &8800 - &8FFF or &9000 - &97FF, etc.: in other words, at 2K intervals. (Thus, it appears that RAM does not use complete address decoding.) This continues until address 49152 (&C000) is reached. In a PC-1251, additional RAM appears at 49152 (&C000) and this block is duplicated beginning at 53248 (&D000).

Theoretically, at least according to the advertising literature, there should be another 8K of ROM somewhere in the unit. (The literature clearly says the PC has 24K of ROM.) At this time, the search for that ROM is still in progress. It is quite likely that this block of coding handles the CE-125 device. Unlike the PC-1500, the PC-1250 properly tokenizes commands such as LPRINT whether or not it is connected to the printer unit. Provisions must have been made to interpret such statements within the basic PC.

ROM Coding

Peeking at ROM (beginning at address &C000) reveals quite a bit of information. First of all, it quickly becomes evident (to those that may have delved into the PC-1500) that the CPU chip is *not* the same as that used in the PC-1500! The machine language instruction set is different. It is a little early to tell just how much different, in terms of architecture and so forth, but at this point it appears that it certainly does not use the same machine codes!

Furthermore, scanning the ROM for familiar ASCII characters (to locate token tables, for instance) won't do much good. You see, the PC-1250 (like its predecessor the PC-1211 and Radio Shack equivalent PC-1) doesn't use ASCII codes at the machine level! The letter A is represented by &51 (instead of &41), digits are in the range &40 - &49 (instead of &30 - &39) and symbols and punctuation marks are different, too (space = &11 instead of &20). You need to remember this if you examine BASIC source code stored in user's RAM. Can you imagine the gyrations the BASIC interpreter has to go through when dealing with strings? Remember, the specifications say that string functions (such as ASC) return true ASCII values! Whew.

In any event, at address &412A in ROM you can pick up what appears to be the main token table. The first entry there is: B5 51 62 55 51 54 DC 51 26. This can be decoded as: a classifier byte (B5) with the 5 part indicating that the keyword has 5 characters. The next five bytes, 51 62 55 51 54, represent the letters A R E A D, which is simply the AREAD keyword. Note the non-ASCII representation of the characters. Next comes the token value for that keyword (DC). Finally, the next two bytes probably represent the address in ROM (51 26) where the routine to process the statement begins. The table continues in this fashion and contains a number of functions as well as commands and statements. If you are inclined toward working on cracking the machine codes used in this PC, this is a good place to start getting a feel for the layout of the ROM. Not surprisingly, the tokens for all of the statements that appeared in the original PC-1211 are the same on the PC-1250. Of course, new tokens appear to represent the upgraded capabilities. (The fact that this would virtually have to be the case is hinted at in that the PC-1250 can interpret source code stored on tape by a PC-1211. This source code is in tokenized form!)

Do Your Own Dump(s)

If you want to do your own snooping around, you can begin using the routine shown in the accompanying listing. It is a modification of an earlier routine used to print out the contents of memory in a PC-1500. (Refer to *PCN* Issue 15, page 3.) In order to conserve paper, locations are dumped in blocks of 64. Even so, a complete roll of thermal paper will only display a few kilobytes of memory. Choose your targets carefully! Happy hunting!

Program *Memory Dump for PC-1250*

```
10:"D" CLEAR : INPUT "S
   TART ADDR? "!A
12:INPUT "ENDING ADDR?
   "!F
15:DIM B$(1)*24
20:B$(0)="0123456789ABC
   DEF"
25:W= INT (A/4096)
26:X= INT ((A-(W*4096))
   /256)
27:Y= INT ((A-(W*4096)-
   (X*256))/16)
28:Z= INT ((A-(W*4096)-
   (X*256)-(Y*16)))
29:LPRINT "ID$ (B$(0),W
   +1,1);"ID$ (B$(0),X
   +1,1);"ID$ (B$(0),Y
   +1,1);" ID$ (B$(0),Z
   +1,1);"  "
30:FOR B=0 TO 7:C=(
   PEEK (A+B)) AND (&F0
   ):D=1+ INT (C/16):E=
   ( PEEK (A+B)) AND (&
   0F)
40:B$(1)=B$(1)+ MID$ (B
   $(0),D,1)+ MID$ (B$(
   0),E+1,1)+" ": NEXT
   B
45:LPRINT B$(1):B$(1)="
   ": IF INT ((A+8)/64)
   > INT (A/64) LET G=2
   5: GOTO 55
50:G=30
55:IF A+7<F LET A=A+8:
   GOTO G
```

FILE MAINTENANCE PROGRAM

The file maintenance program allows the setup of a filing system to a user's specific needs with regard to record size, number of fields, and field length. The complete file is memory resident for purposes of access. The record size is dependent upon the number of fields and their length. Once a file has been created, records may be added, changed or retrieved and the file may be saved on cassette. The program was developed on the Radio Shack PC-2, and I suspect it will run on the Sharp PC-1500. Using an 8K memory module, the following conditions are possible:

Record Length	80 Bytes Max
Number of Records	80 Max (if record length is 80)
Fields per Record	20 Max
File Name	16 Bytes Max (fixed)
Field Name	16 Bytes Max (fixed)
Field Length	80 Bytes Max
Field Search	5 Fields Max

To run the program, execute a run command and a continuous menu will be displayed. To select a menu item, a single key stroke of the appropriate letter or number is all that is necessary. The program prompts the user and validates all inputs. An invalid input will cause a return to the prompt. Capacity overflows are indicated by error messages. The execution of the menu subroutine executes a clear command; therefore, using the Break key and restarting the program will erase any previous files and data in memory. Upon the completion of a menu function, the menu will be displayed. The following menu functions are available:

S – Setup new file
O – Output file
I – Input file
1 – Print file parameters
A – Add a record
P – Print a record
F – Find record(s)
C – Change a record

A delete function was not incorporated in order to conserve program space. If a record is created and is no longer needed, label one of its fields deleted. When a new record is desired, use the Find function to locate the record and the Change function to edit its contents to the new data.

If peripheral programs are to be written on the data base, the following file information will be helpful:

F$	=	File name
N$ (array)	=	Field names
R$ (array)	=	Records
L (array)	=	Length of fields
F	=	Number of fields in record
R	=	Number of records in file
X	=	Length of record
U	=	Number of records used

This program submitted by: *Stephen Tomback, 19 Maplewood Way, Pleasantville, NY 10570.*

Program *File Maintenance*

```
 1:REM File maint
   enance
 2:REM FM 10/11/8
   2  12/04/82
10:REM MENU
15:CLEAR :LOCK :
   DIM U$(0)*80
20:CLS :WAIT 0:E=
   8:K=0:RESTORE
   :FOR I=1TO E:
   READ U$(0):
   PRINT U$(0)
25:FOR J=1TO 60:U
   $=INKEY$ :IF U
   $<>""LET J=60
30:NEXT J
35:IF U$=""THEN 5
   5
40:FOR J=1TO E:IF
   MID$ ("SOIIAPF
   C", J, 1)=U$LET
   K=J:J=E
45:NEXT J
50:IF KCLS :ON K
   GOSUB 100,400,
   500,600,700,85
   0,900,1000:
   GOTO 20
55:NEXT I:GOTO 20
60:DATA "S - SETU
   P NEW FILE","O
    - OUTPUT FILE
   ","I - INPUT F
   ILE"
65:DATA "1 - PRIN
   T FILE PARAMET
   ERS","A - ADD
   A RECORD","P -
    PRINT A RECOR
   D"
70:DATA "F - FIND
    RECORD(S)","C
    - CHANGE A RE
   CORD"
100:REM NEW FILE
105:CLEAR :INPUT "
   NEW FILE NAME:
    ";F$
110:IF LEN F$<1
   THEN 105
115:INPUT "HOW MAN
   Y RECORDS: ";R
120:IF R<1OR R>255
   THEN 115
125:INPUT "FIELDS
   PER RECORD: ";
   F
130:IF F<1OR F>20
   THEN 125
135:DIM N$(F-1):
   DIM L(F-1)
140:FOR I=1TO F
145:CLS :PRINT "FI
   ELD";I;" NAME:
   ":CURSOR 14:
   INPUT "";N$(I-
   1)
150:IF LEN N$(I-1)
   <1GOTO 145
155:CLS :INPUT "HO
   W MANY BYTES:
   ";L(I-1)
160:IF L(I-1)=0
   THEN 155
165:IF X+L(I-1)<81
   LET X=X+L(I-1)
   :GOTO 175
170:PRINT 80-X;" B
   YTES LEFT":
   BEEP 15:GOTO 1
   55
175:NEXT I
180:IF STATUS 3-
   STATUS 2-R*X-1
   81<1THEN 190
185:DIM R$(R-1)*X:
   R$(0)="":DIM I
   $(0)*80:DIM U$
   (0)*80:RETURN
190:PRINT "NOT END
   UGH MEMORY":
   BEEP 15:GOTO 1
   05
400:REM OUTPUT
405:IF LEN F$=0
   GOSUB 2000:
   RETURN
410:INPUT "TAPE RE
   ADY? Y to outp
   ut ";I$
415:IF I$<>"Y"THEN
   410
```

Program *File Maintenance (conclusion)*

```
420:PRINT #F$;F,R,
    X,U
425:I$=F$+"*"
430:PRINT #I$;N$(*
    ),R$(*),L(*):
    RETURN
500:REM INPUT
505:CLEAR :INPUT "
    INPUT FILE NAM
    E: ";F$
510:INPUT #F$;F,R,
    X,U
515:DIM N$(F-1):
    DIM R$(R-1)*X:
    DIM L(F-1)
520:I$=F$+"*"
525:INPUT #I$;N$(*
    ),R$(*),L(*)
530:DIM I$(0)*X:
    DIM U$(0)*X:
    RETURN
600:REM PARAMETERS
605:IF LEN F$=0
    GOSUB 2000:
    RETURN
610:TEXT :CSIZE 1:
    LF 5
615:LPRINT "FILENA
    ME : ";F$
620:LPRINT "NO OF
    RECORDS :";R
625:LPRINT "RECORD
    S USED :";U
630:LPRINT "BYTES
    PER RECORD :";
    X
635:LPRINT "MEMORY
     LEFT :";
    STATUS 3-
    STATUS 2:LF 1
640:FOR I=1TO F:
    LPRINT "FIELD"
    ;I;" ";N$(I-1)
    ;TAB 28;USING
    "###";"BYTES";
    L(I-1):USING
645:NEXT I:LF 6:
    RETURN
700:REM ADD
705:IF LEN F$=0
    GOSUB 2000:
    RETURN
710:FOR I=0TO R-1:
    REM FIND 1ST A
    VAILABLE RECOR
    D
715:IF LEN R$(I)=0
    THEN 725
720:NEXT I:PRINT "
    NO MORE RECORD
    S!":BEEP 15:
    RETURN
725:FOR J=0TO F-1
730:CLS :I$(0)="":
    PRINT N$(J);":
     ":CURSOR LEN
    N$(J)+2:INPUT
    "";I$(0)
735:IF LEN I$(0)>L
    (J)THEN 760
740:IF LEN I$(0)=L
    (J)THEN 750
745:I$(0)=I$(0)+"
    ":GOTO 740
750:R$(I)=R$(I)+I$
    (0):I$(0)=""
755:NEXT J:U=U+1:
    CLS :PRINT "AD
    D RECORD";U:
    BEEP 15:RETURN
760:GOSUB 3000:
    GOTO 730
800:REM OUT TO PRI
    NTER
805:TEXT :CSIZE 1:
    LF 1:J=0
810:LPRINT "RECORD
    :";N:LF 1
815:FOR I=0TO F-1
820:IF I=0LET J=1
825:LPRINT N$(I);"
    : ";MID$ (R$(N
    -1),J,L(I)):J=
    J+L(I)
830:NEXT I:LF 1:
    RETURN
850:REM PRINT
855:IF U=0GOSUB 20
    00:RETURN
860:N=0:INPUT "REC
    ORD NUMBER: ";
    N
865:IF N<1OR N>U
    THEN 860
870:GOSUB 800:LF 5
    :RETURN
900:REM  FIND
905:IF U=0GOSUB 20
    00:RETURN
910:CLS :Q=0:INPUT
    "HOW MANY FIEL
    D SEARCH? ";Q
915:IF Q<1OR Q>5OR
    Q>FTHEN 910
920:RESTORE 990:
    FOR I=1TO Q:
    READ W$:Z$=
    STR$ I
925:CLS :@(I)=0:
    CURSOR :WAIT 0
    :PRINT Z$;W$:
    CURSOR 4:INPUT
    "FIELD NUMBER:
     ";@(I)
930:IF @(I)<1OR @(
    I)>FTHEN 925
935:CLS :@$(I)="":
    CURSOR 0:PRINT
    "SEARCH";@(I);
    " FOR: ":
    CURSOR 14:
    INPUT "";@$(I)
940:IF LEN @$(I)>L
    (@(I)-1)GOSUB
    3000:GOTO 935
945:IF LEN @$(I)<1
    THEN 935
950:NEXT I
955:FOR P=0TO U-1:
    FOR I=1TO Q
960:CLS :PRINT "RE
    CORD";P+1:BEEP
    1
965:S=@(I):GOSUB 4
    000
970:T=LEN @$(I)
975:IF MID$ (R$(P)
    ,Z,T)<>@$(I)
    NEXT P:RETURN
980:NEXT I
985:N=P+1:LF 2:
    GOSUB 800:NEXT
    P:RETURN
990:DATA "st","nd"
    ,"rd","th","th
    "
1000:REM CHANGE
1005:IF U=0GOSUB
     2000:RETURN
1010:N=0:INPUT "R
     ECORD NUMBER
     : ";N
1015:IF N<1OR N>U
     THEN 1010
1020:P=0:INPUT "F
     IELD NUMBER:
      ";P
1025:IF P<1OR P>F
     THEN 1020
1030:I$(0)="":
     INPUT "CHANG
     E TO: ";I$(0
     )
1035:IF LEN I$(0)
     >L(P-1)THEN
     1050
1040:IF LEN I$(0)
     =L(P-1)THEN
     1055
1045:I$(0)=I$(0)+
     " ":GOTO 104
     0
1050:GOSUB 3000:
     GOTO 1030
1055:IF P=FTHEN 1
     080
1060:S=P+1:GOSUB
     4000:T=X-Z+1
1065:U$(0)=MID$ (
     R$(N-1),Z,T)
1070:I$(0)=I$(0)+
     U$(0)
1075:IF P=1LET R$
     (N-1)=I$(0):
     RETURN
1080:S=P:GOSUB 40
     00:R$(N-1)=
     MID$ (R$(N-1
     ),1,Z-1)+I$(
     0):RETURN
2000:PRINT "MEMOR
     Y EMPTY!":
     BEEP 15:
     RETURN
3000:CLS :PRINT "
     FIELD LENGTH
      EXCEEDED!":
     BEEP 15:
     RETURN
3336:+6701
4000:REM START PO
     S OF FIELD I
     N R$
4001:REM  ENTER/F
     IELD # IN R$
4002:REM  EXIT/PO
     S IN Z
4005:Z=1:IF S=1
     RETURN
4010:FOR K=1TO S-
     1:Z=Z+L(K-1)
     :NEXT K:
     RETURN
STATUS 1      3256
```

THE CE-158 RS-232C AND PARALLEL INTERFACE

After quite a few hours of staring at the manual, it became apparent that you needed more than just the CE-158 in order to make anything exciting occur! Things began to happen when I spotted my idle Epson MX-80 line printer. But wait! The connector on it didn't look anything like the parallel output on the CE-158. No kidding! They are not the same. One is a 25-pin EIA connector, the other is a 36-pin "alligator" job. Whew. That seems ridiculous. What do you want to bet you have to buy a cable that has the two different mating connectors on each end just to make it work? Would you believe Sharp is retailing it for $75.00? Believe it.

Of course, if you do not want to shell out that kind of money for a connecting cable, take a look at the accompanying table. If you can find an ordinary female 25-pin RS-232C connector and a male 36-pin alligator ("Centronics") connector, then you can wire up your own cable. Of course, it is still going to cost you a few bucks because those connectors and multi-wire cable are not cheap. But, seventy-five bucks?

Table *Cable Wiring from CE-158 Parallel Port to Centronics Connector*

EIA	Centronics
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	11
11	31
12	nc
13	nc
14	19
15	20
16	21
17	22
18	23
19	24
20	25
21	26
22	27
23	29
24	30
25	16

The cable, though, is just about half of the trick. The other half is simply using the statement: OPN "LPRT". This short statement is all it takes to redirect data from the CE-150 printer to your external unit! Once issued, a command such as LLIST will cause programs to be listed on your full width printer. Execution of LPRINT statements similarly are routed through the parallel port to your desktop clunker.

There are a few other special directives to ease the path. The FEED statement causes your external printer to linefeed. If you want 8 linefeeds in a row, just use: FEED 8. Another directive, CONSOLE, is used to format the maximum lengths of lines outputted on the external device. This appears to be settable from 16 characters to 80 on my system. (Remember, it is an Epson MX-80). You use this directive once when you first start using the interface. That format will remain until you change it (or power down). There is a suffix to this directive that is supposed to allow alteration of the "end of line" character. However, the details of this specification are unclear at this point. I have been getting away with copying a statement example wherein this suffix value was 1. Thus, to set a line length of 40 characters, I just execute: CONSOLE 40,1.

While the CE-158 Instruction Manual is pretty much a disaster, the section relating to the parallel port (pages 64 - 71) should be generally decipherable by those with an electronic technician's bent. This is primarily because it consists of charts and tables. There is even a timing diagram so that designers can take a whack at the using the port for other output purposes. (It sure looks as though it should be possible for someone to design a video interface that accepted data from this port.)

The RS-232C Port

Trying to get anything to happen with the RS-232C port can take a lot of patience. It turns out, I think, that most people experience a lot of difficulty with this part of the interface because of one particular operational aspect. (That is, however, after discounting the effects of the woeful manual, which can cause enough grief all by itself!)

The CE-158 is capable of communicating through the RS-232 port using a set of special BASIC statements. Or, it can be placed in a powerful TERMINAL mode. (No, it doesn't cause the unit to "drop dead", but you might tend to wish it would while you are struggling to understand its operation using the manual supplied.) When in the terminal mode, the PC-1500 will effectively emulate a terminal with many operational features. All of the programming for accomplishing terminal operations, including a variety of options, are on ROM inside the CE-158. It is all very nice, once you get it working for you.

Where Beginners Have a Problem

Naturally, most people, when they get a new item, like to "take it for a spin". Trouble is, if you don't know what you are doing with the CE-158, you can not really tell whether its wheels are turning! The trouble centers around that pesky little RS-232C port.

An EIA RS-232C port is a two-way interface. It is designed to both transmit and receive data. It is also quite flexible. Indeed, this flexibility can cause a lot of problems for the uninitiated. Through the use of sometimes complex arrangements of signals, the interface can be adapted to a number of communication "protocols" (systems or methods of passing information).

The RS-232C interface includes a number of control signals with names such as Request To Send (RTS), Clear To Send (CTS), Data Set Ready (DSR) and Carrier Detect (CD). These signals are used to tell the interface when it is appropriate to transmit data to another device. Of the four signals just listed, one is an "output" signal from the CE-158 (in this case) which is used to tell another device (such as a modem) that the CE-158 has information to transmit. This is the RTS (Request To Send) signal. The other three signals are "inputs" to the CE-158. They must all be in the proper state (a logic one or "marking") before the unit will begin electronically transmitting information on pin 2 (TD - - Transmit Data) of the interface.

Thus, before the software that controls the CE-158 will allow it to send information out of the RS-232 port, all three of the following RS-232 pins must be at a logic one: pin 5 — the Clear To Send (CTS) lead, pin 6 — the Data Set Ready (DSR) lead, and pin 8 — the Carrier Detect (CD) lead. (Remember, a logic one at an EIA RS-232C interface is defined as a distinctly positive voltage. For the CE-158 it is spec'd to be between +5 and +15 volts relative to the ground reference on pin 7 of the port.)

In a typical, full-dress communications system, those pins would be fed signals from a device, such as a modem, in such a manner that when it was time for the CE-158 to transmit, it would be able to do so. That is, the logic levels would be appropriate at those inputs.

But, what happens if someone just wants to "fool around" with their CE-158 to "see if it works"? What happens if you try to get the CE-158 to send some data "to nowhere"? In other words, what will happen if someone tries to check out the RS-232C port when nothing is connected to it?

It turns out that the CE-158 is designed so that nothing will happen under this condition! And, I would not be surprised to learn that more than one purchaser has returned their CE-158 to the factory after they mistakenly thought it was malfunctioning because they attempted to perform such a test!

Unless you fool the CE-158 into thinking that it is connected to a device in a communications system, the unit will just cause the PC-1500 to "hang up" (go into an endless loop) when you try to execute directives intended to have it transmit data!

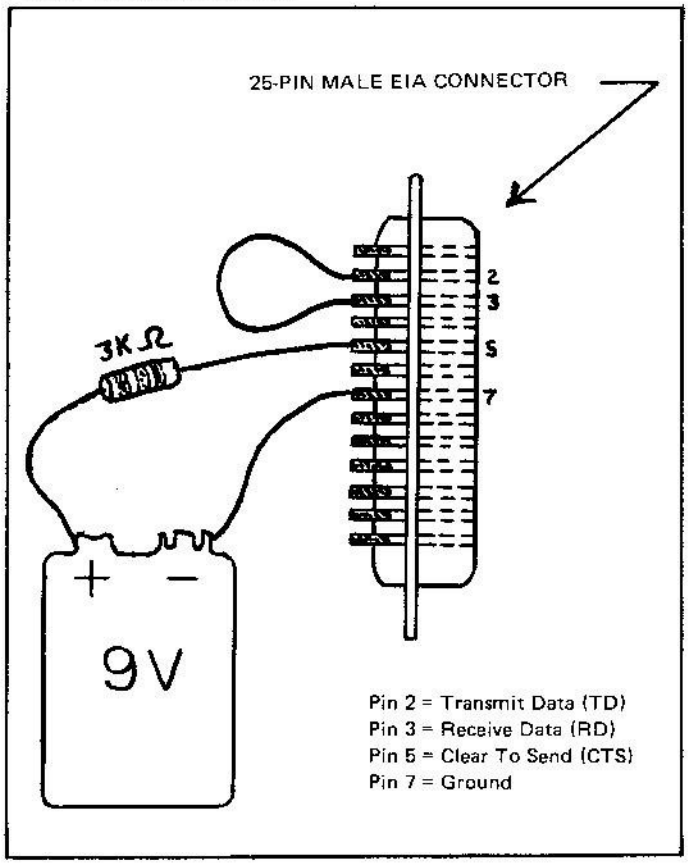
However, tricking the interface into thinking it is connected to something is not difficult, once you understand the situation. It turns out that two of the control pins on the RS-232 port effectively "float" to the logic high state when they are not connected to another device. Pins 6 and 8 (Data Set Ready and Carrier Detect) automatically assume this logic one condition when they are not connected elsewhere. However, the Clear To Send signal on pin 5 is designed to "sink" to a logic zero level when it is disconnected. That one signal will prevent the interface from transmitting data. When it is at a low state, it will cause any software that attempts to execute a data transmit directive to effectively "hang up". In short, it can stop the CE-158 interface in its tracks!

Make Your Own Test

If you are one of those people who have been pounding your head against the wall because, for instance, you could not get a short test program (such as that at the bottom of page 24 in the CE-158 Instruction Manual) to complete execution or couldn't make the TERMINAL mode operate properly when hooked to a simple modem, you might want to try the following test.

Obtain a small 9-volt radio battery, a couple of clip leads, a 3000 ohm resistor, and a male 25-pin EIA connector. Clip the minus terminal of the battery to pin 7 of the connector. Attach the 3000 ohm resistor to the plus lead of the battery. Clip the other end of the resistor to pin 5 of the connector. Plug the male connector into the RS-232C port (lower connector) on the CE-158.

Pictorial *CE-158 Test Circuitry*



Try executing a few of those directives or short programs (such as the one on page 24 of the manual) that used to get hung.

Not convinced? Do you really want to see your RS-232 port "talk to itself"? Try this: Add a jumper between pins 2 and 3 of the test connector that you have connected to the port. This will effectively "loop-back" data from pin 2 (output) to pin 3 (input). Place the CE-158 in the terminal mode by typing the TERMINAL command (or the similar DTE directive) on the keyboard of the PC-1500 (when it is in the RUN mode). When the menu:

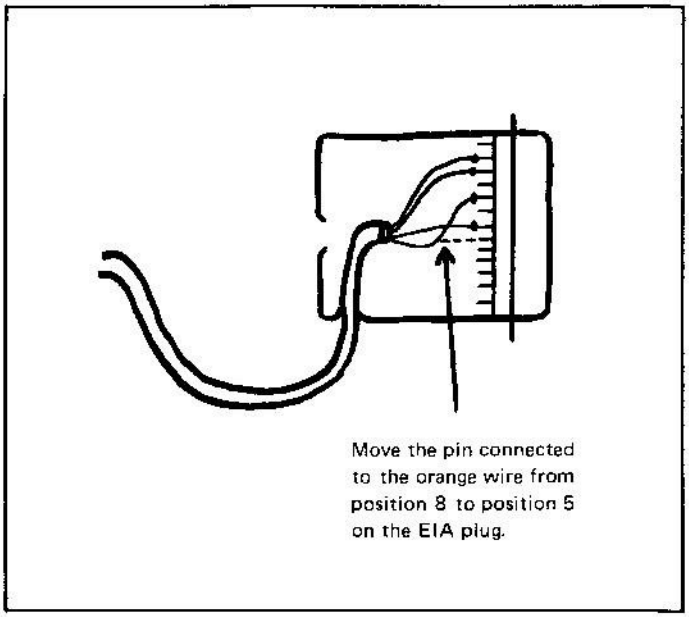
Terminal: Ent Aut Quit

comes up, select the "Ent" option. Now type some data. You should see the characters you input reflected on the LCD of the PC-1500. (If you had been fooling around with various terminal options and had placed ECHO ON, then you would see double sets of characters each time you pressed a key.) This demonstrates that what you are typing on the keyboard is being transmitted out pin 2 and is coming back in pin 3 in the test arrangement. If you want to see the effect of the artificial signal level at pin 5 (Clear To Send), remove the clip lead between the battery, the 3K resistor, and pin 5. Try typing some more data on the keyboard. What happened? Replace the clip lead to raise the signal level back to a logic one. Can you see the keyboard characters echoed again?

Hooking Up a J-CAT Modem

Of course, a real fun way to use the CE-158 is to hook it up to a communications modem so that you can talk with a big computer over the phone lines. It can be a bit incongrous these days to have a system wherein the PC itself is half the size of a modem, but that does not have to be the case any longer. A company by the name of Novation, Inc., is now producing a miniature modem that they call the J-CAT. It is a type 103 (originate/auto answer) model. It is not fancy, but it should be adequate for many PC users. The list price is $149.95 in the United States. Remember, you need this type of device (a modem) in addition to the CE-158 if you want to communicate over the phone lines using your PC.

Pictorial *Modifying the EIA-to-Modem Cable for the J-CAT Modem*



The J-CAT modem comes from the factory configured as a very simple communications modem. It is not wired up to perform fancy protocols via Request To Send, Clear To Send, Data Set Ready signals, etc. Nope, it is supplied with just four wires going to a 25-pin EIA connector: pin 2 (Transmit Data), pin 3 (Receive Data), pin 7 (Ground) and pin 8 (Carrier Detect). Alas, if you simply plug this connector from the J-CAT to the CE-158, you will find that you do not have a working

communications system.

But, by now you should know why! Yep, it is because of that pin 5 on the RS-232 port of the CE-158 which sinks to a logic low level when nothing is connected to it. Remember, the connector from the J-CAT modem does not normally use pin 5.

However, there is a quick and easy remedy to the situation. Since pin 8 (Carrier Detect) on the CE-158 port will float high if nothing is connected to it, the signal coming from the J-CAT to that pin is not really needed in this system. Instead, the Carrier Detect signal from the modem can be used to activate pin 5 (Clear To Send) on the CE-158.

The EIA connector supplied with the J-CAT has removable pins. It is a simple matter to push out the connector pin on the cable supplied with the J-CAT at pin 8 (which is connected to the wire carrying the Carrier Detect signal) and move it over so that it is at pin position 5. (Refer to the accompanying diagram.) That is all it takes to construct a viable modem system for use with networks such as The Source or Compuserve!

Communicating Using the CE-158

Once a modem such as the Novation J-CAT is successfully connected to the CE-158 and the phone lines, you are ready to tap the built-in communications capabilities of the CE-158. (It is assumed at this point that you are familiar with networks such as Compuserve and The Source and want to use such services. By the way, the J-CAT modem is currently being delivered with a package that enables you to try out The Source for one hour at no charge.)

Typing in the keyboard statement TERMINAL or DTE will automatically place the CE-158 into its terminal communications mode. This is initially indicated by a brief display of the message:

---ENTER MENU SELECTION. . .

followed by the beginning of what is a multi-part menu:

Terminal: Ent Aut Quit

You access various options by pressing the softkeys underneath the menu legends. Or, you can bring up another part of the primary menu by, for instance, pressing the line scroll (down) key. Doing so at this point, for instance, would reveal a new selection:

Setup: Aut Fnc Com

Another press of the scroll (down) key would yield:

Operate: Nrm A/P A/L

and yet another would reveal:

Protocol: XO/O Echo

At this point, using the line scroll key will cause the original menu selection to reappear. In other words, this portion of the communications menu cycles in the display (in response to pressing the line scroll keys) as though it were on a cylinder or drum.

Another set of options is accessed by pressing the SHIFT and line scroll (up) keys:

Output: Ext Trc Dsp Etx

This separate group of options is related to output or display options when in the communications mode.

To give you an idea of how various communications options may be selected, consider the output group just mentioned. If the softkey underneath the legend "Ext" was pressed, the PC display would change to query:

EXT. PRINTER OFF ?(Y/N)

This query indicates that the external printer (fed through the parallel port of the CE-158) is currently considered (by the communications package) to be off. Thus, data received via the RS-232 port will be routed to the PC-1500 for display on its LCD (or to the CE-150 if it is in use). The various option queries usually operate in a flip-flop mode. Responding (Y)es to the above query or merely pressing the ENTER key would leave the present condition unchanged. However, responding (N)o would cause the condition to switch to the opposite state (external printer considered activated). If this was done, then pressing the "Ext" softkey would result in the option now appearing as:

EXT. PRINTER ON ?(Y/N)

Note that now the printer is considered to be activated and the query is to confirm or alter this status.

Pressing the softkey under the "Trc" legend provides the opportunity to set or clear what Sharp refers to as a communications "trace" mode. When selected, this mode directs keyboard inputs to be reflected on the device connected to the parallel port. Otherwise, such inputs are shown on the LCD of the PC.

The softkey under "Dsp" is used to select whether horizontal scrolling of input lines is to occur or whether data is to essentially be displayed in maximum of 26-character blocks. This latter option is called the "clean text" mode. This mode of operation seems somewhat easier to read than a continuously scrolling line, at least when using a time-sharing service. Its selection is confirmed by the message:

CLEAN TEXT ON ?(Y/N)

Finally, in the output section, "Etx" is used to label the softkey that permits the designation of the red CL key for special uses. When this option is activated, depression of the CL key will cause the unit to transmit the ETX code (&03). Depression of SHIFT/CL will cause a 240 millisecond spacing condition to be transmitted (sometimes known as a "line break"). If the "Etx" option is not selected, then the CL key is essentially idle during communication operations. As with other options, the mode toggles between selected and non-selected as indicated by the displayed queries:

CL=ETX ON ?(Y/N)

or

CL=ETX OFF ?(Y/N)

To get out of the "output parameters" mode, you need simply press the line scroll up (or down) key. This will bring up one of the other previously mentioned menus. To review some of the other options (and thus capabilities) that can be selected, let us examine those other menus in greater detail.

The "Setup" menu has legends over three softkeys. You can make selections that enable you to establish an "automatic sign-on buffer", to define operations using the Reserve keys, to specify the length of a memory-resident communications storage area, and to customize communication parameters such as baud rate, number of data bits and number of stop bits in a character.

The "Operate" menu provides a means to select from among several operational modes such as "paged", line-at-a-time or continuous reception.

The "Protocol" menu enables one to select whether the system will automatically transmit XON and XOFF codes and how information typed on the keyboard is "echoed" on the system display device.

The communications software also has other features. There is not sufficient space to go into all the details and capabilities in this article. However, a significant capability is that of being able to "capture" information in a communications buffer. This buffer can then be scanned using "window" directives and played back on an external printer through the parallel port, if desired.

Alas, this scanning/playback operation can only take place when incoming data has been halted. Additionally, it takes a significant amount of time to search and locate a particular piece of data (or even just to get to the start of a long buffer).

I have yet to see a system that would enable one to buffer, say, 10 to 15 seconds of real time data so that pressing a switch whenever desired would permit one to obtain a "lagging" hardcopy of pertinent data. Of course, with what is now known about machine language programming on the PC-1500 and the BASIC communication statements provided in the CE-158, it might be possible for some enterprising souls to put together a truly customized communications package. There is no doubt that machine language methods would have to be used in order to provide the kind of real-time buffering capabilities necessary to implement a practical system.

It Is a Start

The CE-158 does has a lot of capability and offers unlimited potential for users to call upon the enhanced BASIC statements to tailor their own communications programs. Until such time as Sharp sees fit to produce a somewhat more readable manual, however, users should be prepared to do a lot of experimenting on their own in order to fully understand and utilize its capabilities. The material in this article should assist newcomers in getting a handle on this PC peripheral.

FROM THE WATCH POCKET

The new Radio Shack Model 100 Portable Computer is quite impressive. Putting a large display in the unit makes it a lot easier to perform word processing tasks and to follow the continuous stream of data one receives when connected to a timesharing network. Leaving out a printer, but providing several interfaces for connecting an external one seems like a sensible, cost-effective trade-off. It is a functionally attractive unit and I predict it will do well. While it is similar in size and some other respects to the Epson HX-20, it does not utilize the same kind of CPU chip. Speculation by others that it was produced by NEC seems doubtful. Radio Shack declines to identify the subcontractor.

Sharp CE-158 Needs a New Manual

After having seen the Instruction Manual for the CE-158 interface, I do not wonder that confusion abounds concerning this RS-232 device. In many places the manual is more of a hindrance than a help. Take, for example, this small quotation:

"Because this unit is a data terminal, connector signals are designed for transmission between data terminal and data communication unit. Therefore, it needs to use the cable of which signal flow is inverted, if the peripheral unit is also a data terminal because it differes (sic) in signal flow."

The sad fact is that one must first interpret the stilted English before one can begin to extract the desired technical information. With a device as complicated as the CE-158, capable of being used in a variety of arrangements, the added burden of literally having to translate each sentence makes practical application of the device quite difficult for many potential users.

That is indeed too bad, for the unit adds significant capability to a PC-1500. Not only can it give the PC the ability to communicate with a larger computer (such as via phone lines using a modem), but it can also enable it to dump listings and data to a full-sized printer. Theoretically, it can even serve as a link between the PC and a mass storage memory device such as a floppy disk. The quick loading of programs from such a peripheral is sorely needed by many PC users.

The problem is that using it with a wide variety of devices requires that one configure it in different ways. Indeed, even the wiring of the connecting cables must be altered depending upon each specific application. In other words, you have to know what you are doing. For this you need, at the very least, a comprehensible instruction manual.

Sharp had better get busy and have this manual re-written if they want to see decent sales of this device in the U.S. market. I don't think the average citizen will tolerate this kind of gibberish to describe such a sophisticated peripheral. Come on, Sharp, this could be a great product if people understood how to make it work!

Here and There

If anyone is still interested in a PC-1, Radio Shack is selling them at $99.95. Gee, is that supposed to be a bargain? The new Sharp PC-1250 with all the capabilities of the PC-1 and more has a list price of just $109.95 and can be purchased at many U.S. outlets for less than what Tandy wants for a PC-1. Matter of fact, some discounters in the New York area are trying to unload PC-1211s (the Sharp equivalent of the PC-1) complete with the optional printer/cassette interfaces for less than $100.00!

ARCsoft Publishers, PO Box 132, Woodsboro, MD 21798, has produced a new book carrying the title *Practical PC-2/PC-1500 Pocket Computer Programs*. Authored by Jim Cole, the new title is said to contain a collection of 36 "new and practical" programs for these PCs. Price is $7.95 and should soon be available in many computer-oriented bookstores. Write to the company for a current catalog as they carry a number of titles for PC-1/PC-2 and PC-1211/PC-1500 models. Many of their titles are especially good for beginners.

Robert Sutliff of New York points out that the problem with the NOT function outlined by Norlin Rober (Issue 21 of *PCN*) can be circumvented by using the form:

NOT (comparison) + 2

Thus, a statement such as IF NOT(1=2)+2 BEEP 1 will produce a beep, while IF NOT(1=1)+2 BEEP 1 will not produce a beep. He says this format will work regardless of the ROM version being used in the PC. Thanks for that tip, Robert.

Battery Lane Publications is said to be preparing a guidebook for the new Epson HX-20 Portable Computer. HX-20 users are invited to share their experiences on that machine with the guidebook editor. Product developers (both software and hardware) are invited to submit material for review consideration. Correspondence should be addressed to: HX-20 Editor, Battery Lane Information Services, 14704 Seneca Castle Court, Gaithersburg, MD 20878.

Scuttlebutt running around now says that Sharp may have 16K RAM modules that plug into the back of the PC-1500 (where 8K modules now go) available by September.

The ROM software modules for the PC-1500, which have a long history of delays, still are not available. However, there are rumors that production evaluation samples are currently being examined. A limited supply of the modules may be available in April. I will believe it when I actually see one.

I will bet it does not take long for people to crack the machine codes used in the PC-1250/1251. Although it is a different CPU than that used in the PC-1500, many *PCN* readers got plenty of "cracking" experience working on the PC-1500/PC-2. It would be nice, however, if Sharp just published the codes as an appendix in their operating manuals. It would save people a lot of work and quite possibly gain them some customers. They say they want to crack the educational market. Well, lots of Computer Science instructors would love to be able to give students a course in machine level programming while having the pupils practice on a low cost PC. The capability is there, why not tell people enough to enable them to play with it if they really want to?

A numbers of readers have inquired as to whether anyone was installing anything similar to the RAM expansion package described in Issue 23 of *PCN* on a commercial basis. I recently received word of an outfit in West Germany that apparently has a number of memory kits for the PC-1500. You might try writing for additional information to: Walter Speidel, Pfarrstr. 10 Postfach 1168, D-7320 Goppingen, West Germany. Watch out, though, the literature you receive may be entirely in German!

By the way, that new Radio Shack Model 100 contains an 80C85 CPU chip. This is believed to be a CMOS version of the popular 8085. With PEEK and POKE provided on the machine, much will be possible!

The next edition of *PCN*, Issue 25, is scheduled for publication in June. (Remember that during May!) — *Nat Wadsworth, Editor*

POCKET COMPUTER NEWSLETTER

Issue 25 — June

NEW RADIO SHACK MODEL 100 SETS TREND

In case you have not yet seen one, the accompanying photograph illustrates what appears to be the hottest portable computer introduced to date: the new Radio Shack Model 100. Technical information on this unit was provided in Issue 24 of *PCN*. While this computer can not be classified as "pocket-sized", the compact portable fits easily in a briefcase or handbag. The unit is being warmly received by the buying public, apparently even beyond the expectation of Radio Shack's proud marketing department. It provides a lot of features that newcomers and old computer hands want, including a comfortable keyboard, large display and built-in modem complete with terminal (communications) software.

While Tandy Corporation normally does not release product-specific sales figures, *PCN* estimates that the firm has probably sold approximately 50,000 units in the first month following its introduction. Radio Shack may find itself hard-pressed to meet the demand for this model in the coming months as its popularity steadily mounts.

An indication of its extremely fast acceptance by the public is the fact that the Compuserve computer network has already created

(continued on page 4)

Another *Personal Information*™ product from SCELBI Publications.

PC-1500/PC-2 CASSETTE STORAGE

The cassette storage capabilities of the Sharp PC-1500 and Radio Shack PC-2 pocket computers are only partially detailed in the manuals supplied with those units. Furthermore, the manuals are confusing in some respects when describing these operations. This article is intended as a summary and clarification of the cassette operations. Thanks for this article go to: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

BASIC Programs

The CSAVE statement, which is well covered in the manuals, may be used to save BASIC programs manually (in the RUN or PRO mode) or as a statement within a program. Its execution records the information beginning at the address pointed to by the START OF BASIC pointer and ending at the address pointed to by the END OF BASIC pointer. (These pointers are located at &7865-66 and &7867-68.) When the saved program is reloaded into the computer using CLOAD or CHAIN, these pointers are automatically reset to the values they had when the program was CSAVEd.

The CLOAD statement may be executed manually in either RUN or PRO mode. CHAIN is similar to CLOAD in effect, except that it is usable only within a program and it automatically begins execution of the program loaded from cassette. (The manuals cover CHAIN well.)

RESERVE Memory

If CSAVE is executed when the computer is set to REServe mode, the contents of REServe memory are recorded. A CLOAD statement, executed in REServe mode, loads the stored codes back into REServe memory.

Machine Language Codes

Machine language programs may be saved with the CSAVE M statement. This is not discussed in the instruction manuals. This command takes the form: CSAVE M "FILE NAME"; expr 1, expr 2, expr 3. The first and second expressions are interpreted as the beginning and ending addresses of the code that is to be recorded. The third expression, which is optional, is a memory address at which the computer will automatically begin execution of machine language codes. This will occur as soon as the recorded material has been loaded into memory using an appropriate CLOAD M directive.

This auto-execution feature may be used, for example, to save both the machine language and BASIC portions of a "hybrid" program so as to permit loading of both portions with a single operation. This is accomplished by the inclusion of a short machine language program that sets the BASIC pointers as soon as loading from the cassette has been completed.

An example should help clarify this operation. Suppose your program includes a machine language portion in the area &38C5 to &38FF and the START OF BASIC pointer is set to &3900. Assume also that your END OF BASIC pointer is set to &3B85 (as determined by peeking at &7867-68). In this situation we will need a machine language routine that will be automatically executed after a CLOAD M directive has been completed. This machine language routine must set the START OF BASIC, END OF BASIC and START OF EDIT pointers. (The START OF EDIT pointer is discussed later in this article.)

An ideal location for this machine language routine is in the area immediately following the BASIC program. Thus, for this example, it would begin at &3B86. The machine language routine will end with a call to the base-page-addressed routine 42. This routine sets the PC to the "READY" state. Such a routine, beginning at &3B86 might appear as shown in the accompanying listing. It would be loaded into memory using POKE directives. (Remember that the example listing is just that! The actual pointer settings you would use would depend on the size and location of your BASIC program, etc.)

Example *Routine Sets BASIC Pointers Then Exits to PC "Ready" State.*

Opcodes	Mnemonics	Comments
B5 39	LDA #39	Set START OF BASIC
AE 78 65	STA 7865	pointer
AE 78 69	STA 7869	as well as the
B5 00	LDA #00	START OF EDIT
AE 78 66	STA 7866	pointer to the
AE 78 6A	STA 786A	value &3900
B5 3B	LDA #3B	Set END OF BASIC
AE 78 67	STA 7867	pointer to the
B5 85	LDA #85	value &3B85
AE 78 68	STA 7868	
CE 42	CALL #42	Set computer to "Ready"

Continuing to discuss the example, the cassette recording in this case would be made by execution of the directive:

CSAVE M "FILE NAME"; &38C5, &3BA1, &3B86

Thus, all information for both the machine language and BASIC portions of the program, along with the pointer-setting routine, would be recorded.

Loading this recording at a later time using the CLOAD M command will result in the BASIC pointers being set automatically as soon as the loading is completed.

If you wanted to save the contents of REServe memory also, then the only change needed in the preceding example would be to have the first expression in the CSAVE M directive specify &3800 instead of &38C5. (It is not necessary to set pointers for REServe memory in such a situation.)

Merged Programs

The instruction manuals can easily lead one to believe that the use of MERGE is a lot more trouble than it is worth. Such is really not the case.

Suppose you have several programs already saved on cassette tape and you would like to be able to use all of them by having them available in memory at one time. The MERGE directive permits placing any number of previously recorded programs into the computer at once (assuming, of course, that you do not run out of memory). No problems are caused by the overlapping or repetition of line numbers in the various programs as each merged program operates independently. The programs that are MERGEd together are separated from each other by a "stop" byte (code &FF) that effectively blocks searches for line numbers beyond the end of each program section. These stop bytes, however, *do not block searches for labels!* This feature of PC-1500/PC-2 BASIC is what makes it possible to execute a number of different programs that are stored in memory at one time.

It is important to note, however, that when two or more programs are in memory at one time, that the following two rules must be observed:

1. Execution of any program (other than the first one) *must* be begun with a *label* address. Execution of the program or routine may then be initiated by manually telling the computer to RUN "LABEL" or GOTO "LABEL" or, when appropriate, by using the DEFine key. (Of course, care must be taken to ensure that each label only appears in the combined programs one time. If there are multiple occurrences then the first appearance of the specified label will be utilized.)

2. The only program that may be edited is the last one loaded using the MERGE directive. Thus, any programs that do not begin with a label should be modified to begin with one. This can be done immediately after the program has been loaded, but *before* merging another section.

The pointer located at address &7869-6A has been identified previously in *PCN* as the BEGINNING OF MERGED PROGRAM or START OF PROGRAM marker. I hereby propose calling it START OF EDIT since its function is to locate the place at which editing of program lines may begin. When merging is not involved, its contents are

identical to the START OF BASIC pointer. After a MERGE it is automatically set to point to the beginning of the program that has just been loaded by the MERGE directive.

Knowing this information it becomes possible to alter memory so that two merged programs can be combined into a single "editable" program. Note that this is only possible if the lines in the combined programs are numbered consecutively. Here is how it could be done:

1. CLOAD the first program (if not already in memory).
2. Determine (using PEEK) the address contained in the END OF BASIC pointer (&7867-68). Reduce this value by one and POKE it into &7867-68.
3. Load the second program using MERGE.
4. Execute POKE &7869, PEEK &7865, PEEK &7866.

Recording of Data

There are six types of variables that can be recorded with PRINT # statements:

1. A single undimensioned numeric variable.
2. A single undimensioned string variable.
3. The complete set of "fixed" string variables (A through Z), specified as @(*).
4. The complete set of "fixed" string variables (A$ through Z$), specified as @$(*).
5. A complete numeric array (excluding "fixed" variables), specified by using * as a subscript.
6. A complete string array (excluding "fixed" string variables), specified by using * as a subscript.

During execution of a PRINT # statement, the type of variable in each case is recorded along with the data. (The name of the variable being recorded is not saved on tape.)

When INPUT # is used to load data from tape, each specification must match the type of the corresponding recorded data. Thus a recording made using the statement:

PRINT #"FILE NAME"; AC, D(*), @$(*), J$

could be loaded with:

INPUT #"FILE NAME"; K, A(*), @$(*), D2$

provided that A() was dimensioned with the same parameters as D() and that D2$ was properly defined.

Whenever variables in "main" memory (that is, other than variables A through Z and A$ through Z$) are named in an INPUT # statement, those variables must have been defined prior to execution of INPUT #. Furthermore, if dimensioned, they must have the same dimensions as those previously recorded. If they are dimensioned string variables, they must also agree in length. Any mismatch will produce an ERROR 43 message.

It is of interest to note that CSAVE M may be used to record variables. Of course, if this is done then CLOAD M must be used to recover those variables from tape. As an illustration, you could use:

CSAVE M "FILE NAME"; &7900, &79CF

to save variables Z through Z. A CLOAD M command would then load them back into the computer. The advantage of using this method is that it is faster in operation than using PRINT # and INPUT #.

Assorted Information

As a summary, the four kinds of recordings possible are as follows:

1. REServe memory, saved with CSAVE, loadable only by CLOAD.
2. BASIC program, saved with CSAVE, loadable using CLOAD or MERGE or CHAIN.
3. Coded data (machine language routines, etc.) saved by CLOAD M, loadable using CLOAD M.
4. Variables, saved using PRINT #, loadable using INPUT #.

Remember that the PC distinguishes amongst all four of the above when reading from a tape, so that the same file name may be used for each different type of recording.

CLOAD? may be used to verify recordings made by CSAVE (either REServe memory or BASIC programs), but it cannot be used to verify recordings made using CSAVE M or PRINT #.

Both CSAVE M and CLOAD M may be included as statements within a program.

A second cassette recorder may be addressed using CSAVE-1, CSAVE M-1, PRINT #-1, CLOAD-1, CLOAD?-1, MERGE-1, CLOAD M-1, INPUT #-1, or CHAIN-1 statements.

The inclusion of a file name in cassette-related statements is optional except in the two following instances:

1. PRINT # and INPUT # statements in which the first variable named is a string variable.
2. CHAIN statements in which a line number or label is specified.

When a file name is left out or "" is used as a file name, the CLOAD, CLOAD M, CLOAD?, MERGE, CHAIN and INPUT # statements cause initiation of a search for a recorded file of the appropriate type. The first such file found is loaded.

If a CHAIN statement specifies a line number then it must end the program line in which it appears.

NORMAL PROBABILITY USING MACHINE LANGUAGE

Norlin Rober submitted this program that illustrates an example of the use of machine language to calculate a fairly complicated mathematical function. If you need to perform such calculations frequently, the time saved by using a machine language routine can be quite significant.

A number of short algorithms are commonly used for calculating Normal (Gaussian) probability. The most commonly used procedures are those developed by Hastings. For example, Sharp's POCKET COMPUTER APPLICATIONS MANUAL for the PC-1211 uses the "best" of these. It correctly provides the first seven places following the decimal point. Note though, that for values of Z greater than 5.2,

Program *Normal Probability in PC-1500/PC-2 Machine Language.*

```
38C5 48 79 4A C8  3949 41 DD 8E 0F
38C9 BE DC 20 BE  394D EB 7A 12 36
38CD F5 97 CD 80  3951 CD 7E CD 6A
38D1 BE F0 19 FD  3955 4A 12 B5 44
38D5 58 B5 4B FD  3959 41 B5 66 0E
38D9 CA BE F7 17  395D F0 CD 68 F0
38DD FD 88 CD 7E  3961 E6 F4 7B 60
38E1 BE F1 D4 FD  3965 CD 10 00 CD
38E5 0A BE F7 17  3969 58 A5 7B 61
38E9 CD 7E 58 79  396D DF AE 7B 61
38ED 5A 70 BE F7  3971 99 15 CD 68
38F1 11 BE DB F5  3975 F0 CD 6E E6
38F5 CD 70 EB 7A  3979 CD 30 CD 7E
38F9 10 FF EB 7A  397D A5 79 C9 8B
38FD 11 80 68 00  3981 07 CD 6A EF
3901 6A 15 F6 7B  3985 7A 01 80 F0
3905 60 BE F7 37  3989 5A 78 58 79
3909 A5 7A 00 B7  398D BA F7 11 BE
390D 80 83 80 B7  3991 F0 19 CD 80
3911 01 83 39 A5  3995 8E 05 CD 68
3915 7A 02 B7 16  3999 CD 7E E6 F4
3919 83 16 EF 7B  399D 7B 60 CD 10
391D 61 0A 8E 6F  39A1 00 CD 66 CD
3921 FF 80 21 71  39A5 58 CD 6A F0
3925 47 24 09 52  39A9 A5 7B 61 DF
3929 FF 00 39 89  39AD DF AE 7B 61
392D 42 28 04 01  39B1 DF 99 1D E6
3931 B7 25 83 18  39B5 48 79 4A C8
3935 EF 7B 61 10  39B9 BE DC 20 CD
3939 EB 7A 12 42  39BD 7E E6 CD 30
393D EB 7A 13 50  39C1 CD 7E CD 62
3941 CD 7E CD 6A  39C5 EF 7A 12 F0
3945 4A 12 B5 60  39C9 9E 47
```

where probabilities are less than 1E-7, the calculated results are meaningless.

The routine used in the accompanying machine language program gives a full ten digits of accuracy, regardless of the value of Z. (When Z exceeds 21.165, the probability is less than 1E-99, so the result underflows to zero.) Probabilities are calculated with a Taylor series for absolute values of Z less than 1.6, and with a continued fraction for larger values of Z. Execution is fast, taking less than a second.

You can enter the machine language program using a Monitor program or by using BASIC POKEs. (The program is *relocatable*, so if you wish you may start it at some address other than &38C5 without making any modifications. Just remember to call the right address when accessing the routine if you do move it elsewhere.)

To use the program, just follow these steps:

1. The value of Z to be used must be assigned to the variable Z.
2. Execute CALL &38C5 (or the starting address if you relocate the machine language routine).
3. Display the variable P to obtain the probability that a normally distributed random variable exceeds Z.

As a bonus, the variable O contains the ordinate of the normal curve for the given Z!

You can test the program's operation (to verify proper loading of the machine language codes) with the following examples:

Value of Z	Resulting P	Resulting O
-2	9.772498681E-01	5.399096651E-02
1.3	9.680048459E-02	0.171368592
4	3.167124183E-05	1.338302256E-04

ROUTINE DELETES LINES FROM BASIC PROGRAMS

John Norton, % Pencept, Inc., 39 Green Street, Waltham, MA 02154, submitted this program that makes it easy to remove groups of lines from a BASIC program. Here is what John has to say about his routine:

I have often wished for a simple way to eliminate sections of a BASIC program in a Sharp PC-1500/Radio Shack PC-2. Trying to remove a number of lines by specifying each line number is just too slow, cumbersome and subject to error. By capitalizing on the material that has appeared in *PCN*, I have been able to create a routine that will permit the deletion of a group of lines from a BASIC program.

To use this routine, it must be loaded into memory prior to your development of a BASIC program. Your BASIC program is then created (starting with a line number greater than 14 as this is the highest line number in the deletion routine). The routine uses a JUMP statement so that a RUN directive causes the PC to skip over it and execute the user's program.

The deletion routine may be utilized at any time by pressing the DEFine and D keys. It begins by prompting for two items: the line number at which to begin deleting and the last line number to be deleted. If, for example, you had a program starting at line 500, with a last line number of 850, and you wanted to delete all the lines from number 722 through 751, you would proceed as follows:

What you type	What the display shows
DEF/D (ENTER)	DELETE FROM:
722 (ENTER)	DELETE TO:
751 (ENTER)	WORKING ...

That is all there is to it! After these inputs, the routine will issue a beep each time it encounters a line less than 722. It will signal when it has found line 722, when it finds line 751, and when it does the deletion of the material between those lines.

The routine operates as follows: it marks the memory location at which line 722 starts. It then re-writes every byte from the first line after 751 to the end of the program, into memory starting from the marked byte (where line 722 started).

It is important to note the following: The routine will not attempt to delete itself. It does this be starting to look for lines in memory after itself. It does this by "knowing" exactly how long it is. Thus, the routine should not be altered by as much as a single character! Make sure you load it exactly as shown in the accompanying listing!

The routine uses default values as follows: If no value is given in response to the prompt "DELETE TO", then the "DELETE FROM" value is used. Thus, to delete a single line from memory, just type: DEF/D ### (ENTER) (ENTER), where ### represents the line number to be deleted.

In the event that the starting line is specified as larger than the ending line, the starting line is not found or the ending line is not located, the routine will display an error message such as "START NOT FOUND" or "END NOT FOUND" and cease operating.

To delete the routine itself from memory, type an asterisk (*) immediately before entering DEFinc/D. This will cause the computer to restructure pointers so that the deletion routine is effectively erased!

Program *Line Deletion*

```
1:GOTO 16
2:"D":AREAD DP$:
  IF DP$="*"LET
  D1=59:D2=149:
  GOTO 14
3:CLEAR :WAIT 0:
  INPUT "DELETE
  FROM:";DF:DT=D
  F:INPUT "DELET
  E TO:";DT:
  PRINT "WORKING
  ..."
4:DM=256*PEEK &7
  865+PEEK &7866
  +720:DB=DM:IF
  PEEK DM=255LET
  DM=DM+1:DB=DM
5:DN=256*PEEK DM
  +PEEK (DM+1):
  IF PEEK DM=255
  OR DN>DFBEEP 5
  ,60:WAIT .
  PRINT "NO STAR
  T":END
6:BEEP 1,40:IF D
  N<DFLET DM=DM+
  PEEK (DM+2)+3:
  DB=DM:GOTO 5
7:BEEP 5,30:
  PRINT "START F
  OUND"
8:IF DN=DTLET DM
  =DM+PEEK (DM+2
  )+3:GOTO 11
9:IF PEEK DM=255
  OR DN>DTBEEP 5
  ,60:WAIT :
  PRINT "NO END"
  :END
10:BEEP 1,20:DM=D
  M+PEEK (DM+2)+
  3:DN=256*PEEK
  DM+PEEK (DM+1)
  :GOTO 8
11:BEEP 5,10:
  PRINT "...DELE
  TING":DE=256*
  PEEK &7867+
  PEEK &7868:FOR
  D1=0TO DE-DM:
  BEEP 1,0
12:POKE DB+D1,
  PEEK (DM+D1):
  NEXT D1:POKE &
  7867,INT ((DB+
  D1)/256),((DB+
  D1)AND &FF)
13:FOR D1=20TO 0
  STEP -1:BEEP 1
  ,D1:NEXT D1:
  END
14:D2=D2+((PEEK &
  7865<>PEEK &78
  69)OR (PEEK &7
  866<>PEEK &786
  A))
15:POKE &7865,D1,
  D2:POKE &7869,
  D1,D2:END
16:REM *MAIN PROG
  RAM*
STATUS 1          721
```

(Model 100 . . . continuation from page 1)

a "SIG" (Special Interest Group) devoted exclusively to Model 100 enthusiasts. In the first two weeks following its activation, some 1600 users have joined this SIG in order to use the Model 100-related information base. The Compuserve Model 100 SIG (which can be accessed by Compuserve users typing GO PCS-154 at the command (!) prompt) has already had some 30 programs contributed to its user library. These range from several simple "demo" programs on up to useful utilities such as memory dumps, disassemblers, printer formatting

routines (that will format the output from text files). There are also programs that generate graphics and produce music.

Another indicator of the success of this unit is that already three (that's right: 3!) organizations have announced that they will be publishing magazines that will cater to Model 100 afficianados. The three firms that have announced such publications to date are:

PORTABLE 100 MAGAZINE
21 Elm Street
Box 597
Camden, ME 04843

PORTABLE COMPUTING MAGAZINE
9529 U.S. Highway 42
P.O. Box 209
Prospect, KY 40059

BRIEFCASE PORTABLE
560 South Hartz Avenue - Suite 447
Danville, CA 94526

If you are interested in any of these publications, it is suggested that you contact them right away. Some of them are offering special "pre-publication" subscription rates that may be withdrawn soon.

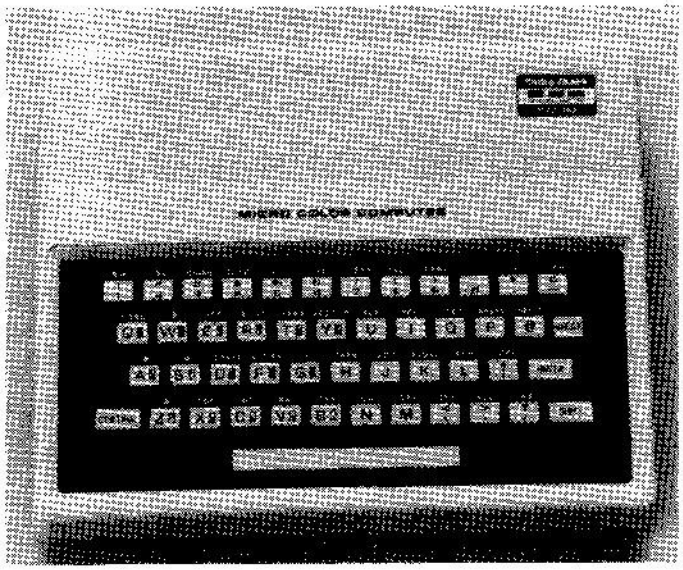
RADIO SHACK ANNOUNCES MICRO COLOR COMPUTER

While it might not technically be termed "portable" because it requires the use of 110 volt a.c. for power, the new MC-10 from Radio Shack is pretty small. Measuring just 7 by 8 by 2 inches and weighing 29.5 ounces (a shade under two pounds), the unit is easily transported from one location to another.

While equipped with a full keyboard and an RF moudulator, the user must provide a television set as a display. The MC-10 can generate eight different colors for graphics and display text in a 32-character, 16-line format. The unit is programmable in BASIC and comes equipped with 4K of memory, expandable to 20K. It also is equipped with a serial I/O port to provide connection to modems or printers as well as a cassette I/O port for loading and saving programs on cassette tape.

An easy-to-read operator's manual is provided with the MC-10. It includes instruction in the use of Microsoft color BASIC especially aimed at beginners as well as a quick-reference guide for more experienced programmers. It is claimed that almost all standard color BASIC programs that work on a 4K Radio Shack TRS-80 Color Computer can be keyed in and run on the new MC-10 with minor changes.

The MC-10 is low-priced: just $119.95 at U.S. Radio Shack Computer Centers and other Radio Shack outlets!



GRAPH PROGRAM FOR THE PC-1500/PC-2

This program, provided by *John Norton*, enables the Sharp PC-1500 or Radio Shack PC-2 equipped with a printer/plotter to draw a variety of graphs. Straight lines such as that expressed by the equation Y=3*X+7, curves such as Y=SIN X, functions with extremes such as Y=1/X, functions with asymptotes and even pseudo-functions such as Y=RND 100 can all be drawn through the use of this program. Want to plot using polar coordinates? It can do that, too! The program allows you to define the region of the coordinate plane to be drawn, to select rectangular or polar coordinates, the increment (accuracy) size, and X and Y scaling factors.

The program that accomplishes all this is shown in the accompanying listing. To utilize the program you must substitute the function that you wish to graph in line 50. For example, if you wanted to examine the function Y=SIN X, you would have line 50 read:

```
50 Y=SIN(X)
```

Notice that this example assumes a rectangular coordinate system. That is, the distance vertically is a function of the horizontal distance. If you were interested in a polar coordinate graph of the SIN function, you would have line 50 appear as:

```
50 R=SIN(T)
```

In this case, R would represent the radius and T would indicate the angle (theta). When using polar (as contrasted to rectangular) coordinates, the distance of the plotted point from the origin is a function of the angle. Therefore, for polar plotting express equations using R = (a function of T). When using rectangular coordinates, express Y = (a function of X).

Once you have placed the desired function in line 50, the program is ready to be used. Just issue the RUN command. Respond to the prompts for rectangular or polar coordinates (R or P), minimum and maximum values of X and Y (or theta) and the X (or theta) increment size. You will also be prompted for X and Y scaling factors.

Once the prompts have been responded to, the program will display pertinent information about the graph it is to draw. It then draws a window of the graph, X and/or Y axes (if they occur within the window), places scaling marks as appropriate and then graphs the function.

If the plot does not appear as you expected it to, you might want to check the accompanying table that lists typical graphing problems. A second table indicates default values that will be used if you respond to a prompt by just pressing the ENTER key.

A selection of drawings produced by the program and the functions and parameters used to generate these drawings is also provided for checking purposes (and as a general illustration of the program's capabilities). Try experimenting. You can get a lot of interesting results!

Table *Typical Graphing Problems*

PROGRAM OUTPUT	POSSIBLE PROBLEM
1. Nothing plotted in window.	1a. Window too small or large or is not over the proper portion of the graph.
	1b. RAD/DEG/GRAD not set properly.
	1c. Increment value too large.
2. Lines appearing where they should not exist.	2. Increment value too large.
3. Curves appear squared.	3. Increment value too large.
4. Program seems to take too long to plot.	4a. Increment value too small.
	4b. Function is producing an error causing program to continuously loop.

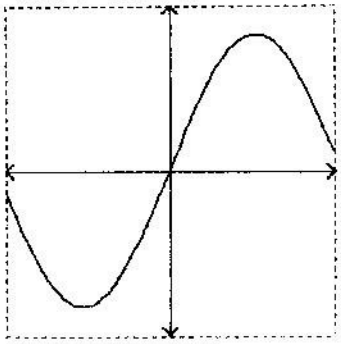
Program *Graph*

```
 10:REM  ** GRAPH
    -- Written by
    John Norton **
 20:"A":WAIT 0:F1=
    0:F2=0:RF=1000
    000:E=2.718281
    828:GOTO 100
 30:BEEP 1,1
 40:ON ERROR GOTO
    850
 50:REM -FUNCTION-
 60:IF G$="P"LET X
    =R*COS T
 70:IF G$="P"LET Y
    =R*SIN T
 80:PRINT "("+STR$
    (INT (RF*X+.5)
    /RF)+","+STR$
    (INT (RF*Y+.5)
    /RF)+")"
 90:RETURN
100:G$="R":INPUT "
    Type of graph
    (R or P): ";G$
110:X1=-10:INPUT "
    X-minimum:";X1
120:X2=-X1:INPUT "
    X-maximum:";X2
130:I=(X2-X1)/75:
    IF G$="R"INPUT
    "X-increment:"
    ;I
140:Y1=X1:INPUT "Y
    -minimum:";Y1
150:Y2=-Y1:INPUT "
    Y-maximum:";Y2
160:T1=0:IF G$="P"
    INPUT "Theta m
    inimum:";T1
170:T2=2*π:IF G$="
    P"INPUT "Theta
     maximum:";T2
180:IF G$="P"LET I
    =(T2-T1)/75:
    INPUT "Theta-i
    ncrement:";I
190:S1=0:INPUT "X-
    scaling:";S1
200:S2=S1:INPUT "Y
    -scaling:";S2
210:CLS :TEXT :
    COLOR 3:LLIST
    50:LF 10:COLOR
    0
220:CSIZE 1
230:IF G$="R"
    LPRINT "RECTAN
    GULAR COORDINA
    TES"
240:IF G$="P"
    LPRINT "POLAR
    COORDINATES"
250:LF 1:CSIZE 2
260:LPRINT "X: ";X
    1;",";X2
270:LPRINT "Y: ";Y
    1;",";Y2
280:IF G$="P"
    LPRINT "Θ":LF
    -1:LPRINT "-:
    ";T1;",";T2
290:LPRINT "INC: "
    +STR$ (INT (RF
    *I+.5)/RF)
300:IF S1>0LPRINT
    "X-S: ";S1
310:IF S2>0LPRINT
    "Y-S: ";S2
320:LF -6-(G$="P")
    -(S1>0)-(S2>0)
330:GRAPH :SORGN :
    COLOR 1
340:LINE (0,0)-(21
    6,216),2,1,B
350:PY=X1/(X1-X2)*
    216
360:IF (PY<0)OR (P
    Y>216)GOTO 440
370:F1=1
380:LINE -(PY,216)
    ,9
390:RLINE -(-5,-5)
    ,0:RLINE -(5,5
    ),0
400:RLINE -(5,-5),
    0:RLINE -(-5,5
    ),0
410:LINE -(PY,0)
420:RLINE -(-5,5),
    0:RLINE -(5,-5
    ),0
430:RLINE -(5,5),0
    :RLINE -(-5,-5
    ),0
440:PX=Y1/(Y1-Y2)*
    216
450:IF (PX<0)OR (P
    X>216)GOTO 530
460:F2=1
470:LINE -(0,PX),9
480:RLINE -(5,5),0
    :RLINE -(-5,-5
    ),0
490:RLINE -(5,-5),
    0:RLINE -(-5,5
    ),0
500:LINE -(216,PX)
510:RLINE -(-5,5),
    0:RLINE -(5,-5
    ),0
520:RLINE -(-5,-5)
    ,0:RLINE -(5,5
    ),0
530:IF (S1+S2=0)OR
    (F1+F2=0)GOTO
    700
540:COLOR 3
550:IF (S2*F2)=0
    GOTO 620
560:K=0
570:K=K+S2
580:IF K<=Y2LINE (
    PY-2,(K-Y1)/(Y
    2-Y1)*216)-(PY
    +4,(K-Y1)/(Y2-
    Y1)*216),0:
    GOTO 570
590:K=0
600:K=K-S2
610:IF K>=Y1LINE (
    PY-2,(K-Y1)/(Y
    2-Y1)*216)-(PY
    +4,(K-Y1)/(Y2-
    Y1)*216),0:
    GOTO 600
620:IF (S1*F1)=0
    GOTO 690
630:K=0
640:K=K+S1
650:IF K<=X2LINE (
    (K-X1)/(X2-X1)
    *216,PX+3)-((K
    -X1)/(X2-X1)*2
    16,PX-3),0:
    GOTO 640
660:K=0
670:K=K-S1
680:IF K>=X1LINE (
    (K-X1)/(X2-X1)
    *216,PX+3)-((K
    -X1)/(X2-X1)*2
    16,PX-3),0:
    GOTO 670
690:COLOR 2
700:CLS :RANDOM :X
    =X1:T=T1:GOSUB
    30
710:G2=(Y-Y1)/(Y2-
    Y1)*216
720:IF G$="P"LET G
    1=(X-X1)/(X2-X
    1)*216
730:IF (G1<0)OR (G
    1>216)LET F=1:
    GOTO 760
740:IF (G2<0)OR (G
    2>216)LET F=1:
    GOTO 760
750:LINE -(G1,G2),
    9
760:COLOR 2,X=X+I:
    T=T+I
770:IF (X>X2)AND (
    G$="R")GOTO 86
    0
780:IF (T>T2)AND (
    G$="P")GOTO 86
    0
790:GOSUB 30
800:P2=(Y-Y1)/(Y2-
    Y1)*216:P1=(X-
    X1)/(X2-X1)*21
    6
810:IF (P1<0)OR (P
    1>216)OR (P2<0
    )OR (P2>216)
    LET F=1:GOTO 8
    40
820:IF F=1LET F=0:
    GOTO 840
830:LINE (G1,G2)-(
    P1,P2),0
840:G1=P1:G2=P2:
    GOTO 760
850:F=1:BEEP 1,20:
    GOTO 760
860:LINE -(0,0),9
870:TEXT :LF 10+(S
    1>0)+(S2>0)
880:END
STATUS 1        2263
```

Table *Default Values for Graph Program*

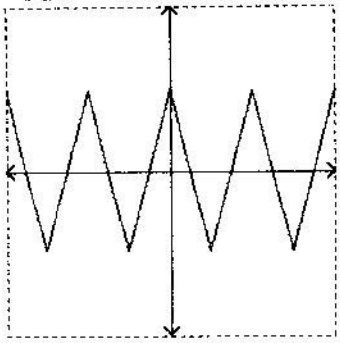
INPUT REQUESTED	DEFAULT VALUE
Rectangular/Polar Coord's	Rectangular (R)
X - minimum	-10
X - maximum	- X - minimum
X - increment	(X max - X min)/75
Y - minimum	Y minimum
Y - maximum	- Y - minimum
Theta min/max	0, 2π
Theta increment	(T max - T min)/75
X / Y scaling	0, X

```
50:Y=SIN X
```



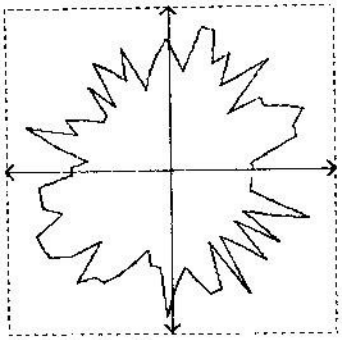
```
RECTANGULAR COORDINATES
X:  -3,  3
Y:  -1.2,  1.2
INC:  0.1
```

```
50:Y=COS X
```



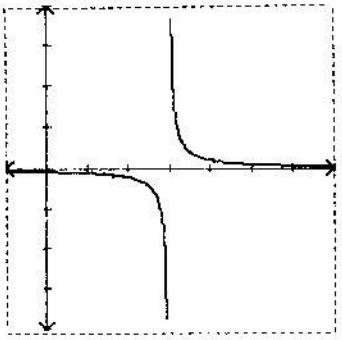
```
RECTANGULAR COORDINATES
X:  -12.56,  12.56
Y:  -2,  2
INC:  1.57
```

```
50:R=10-RND 5
```



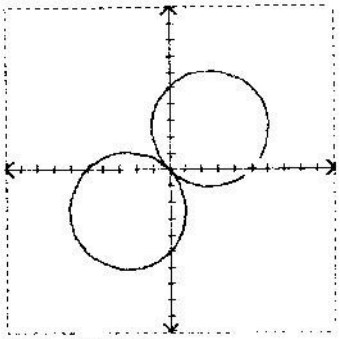
```
POLAR COORDINATES
X:  -10,  10
Y:  -10,  10
θ:  0,  6.283185307
INC:  0.083776
```

```
50:Y=1/(X-3)
```



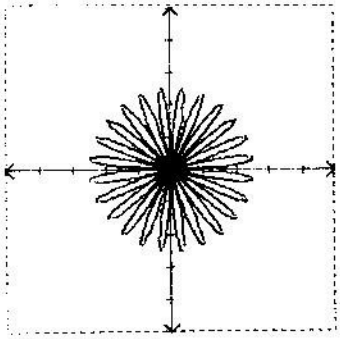
```
RECTANGULAR COORDINATES
X:  -1,  7
Y:  -20,  20
INC:  0.021333
X-S:  1
Y-S:  5
```

```
50:R=ABS (SIN T+
   COS T)
```



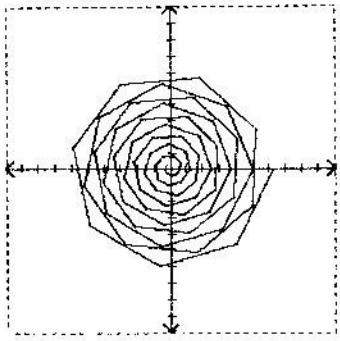
```
POLAR COORDINATES
X:  -2,  2
Y:  -2,  2
θ:  0,  6.283185307
INC:  0.083776
X-S:  0.2
Y-S:  0.2
```

```
50:R=SIN (6*T)*
   COS (6*T)
```



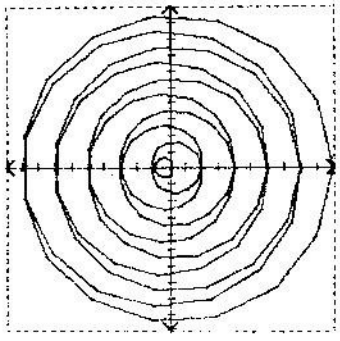
```
POLAR COORDINATES
X:  -1,  1
Y:  -1,  1
θ:  0,  6.283185307
INC:  0.013963
X-S:  0.2
Y-S:  0.2
```

```
50:R=SIN T+T
```



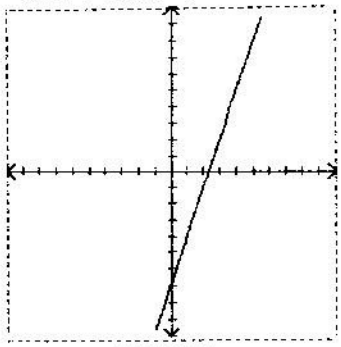
```
POLAR COORDINATES
X:  -100,  100
Y:  -100,  100
θ:  0,  62.83185307
INC:  0.837758
X-S:  10
Y-S:  10
```

```
50:R=ABS (10*Π-T)
```



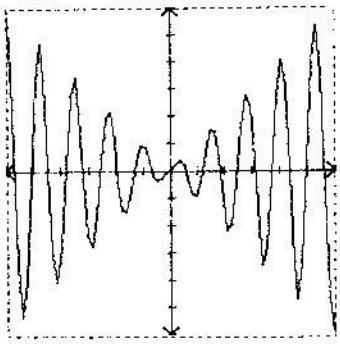
```
POLAR COORDINATES
X:  -31.416,  31.416
Y:  -31.416,  31.416
θ:  0,  62.84
INC:  0.418879
X-S:  3.141592654
Y-S:  3.141592654
```

```
50:Y=3*X-7
```



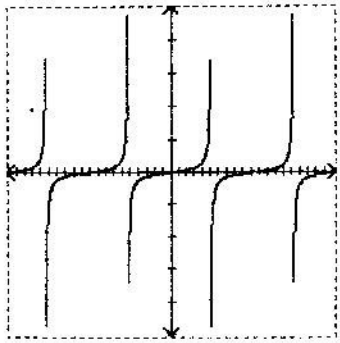
```
RECTANGULAR COORDINATES
X:  -10,  10
Y:  -10,  10
INC:  0.266667
X-S:  1
Y-S:  1
```

```
50:Y=SIN X*ABS X
```



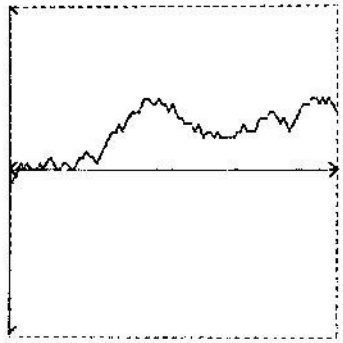
```
RECTANGULAR COORDINATES
X:  -30,  30
Y:  -30,  30
INC:  0.8
X-S:  5
Y-S:  5
```

```
50:Y=TAN X
```



```
RECTANGULAR COORDINATES
X:  -6.283,  6.283
Y:  -50,  50
INC:  0.016755
X-S:  0.31
Y-S:  10
```

```
50:Y=RND 3-2+LY:L
   Y=Y
```



```
RECTANGULAR COORDINATES
X:  0,  100
Y:  -25,  25
INC:  1
```

FROM THE WATCH POCKET

Well, they (Sharp) finally broke down and released information on machine language programming of the PC-1500 (which should also apply to the Radio Shack PC-2). *PCN* will take the credit (thank you) for virtually embarrassing them into acknowledging such capability. Remember, we (thanks particularly to *Norlin Rober*) published the machine codes for almost the entire instruction set about a year ago and have provided a good deal of supplemental material on various aspects of the machine over the past year.

In any event, the new *Sharp Pocket Computer PC-1500 Technical Reference Manual* verifies a lot of what *PCN* has published during the past year. And, it adds a lot more information that will be of interest to PC-1500/PC-2 afficionados.

The book starts off with an overview of the LH5801 CPU, including a block diagram, a summary of its internal registers and status flags and the CPU chip pin-out. There is a detailed description of each signal line. Information is provided on the internal timer, the interrupt system, and special functions (such as reset and wait). CPU execution sequences and timing diagrams are also provided. The instruction set is discussed in detail. A diagram illustrating how CPU registers are affected by each type of instruction is provided in this section. Sharp provides their own set of mnemonics to describe the instruction set.

Note: Be advised that PCN *will continue to use* Rober Mnemonics *in its articles and machine language programs.*

There is a substantial chapter that describes and discusses the application of the LH5810/5811 I/O port controller. Much of this information is new material (even for *PCN* subscribers).

There is a chapter devoted to describing the overall PC-1500 system. This includes information on power supplies (such as in the actual PC-1500, the CE-150 printer/cassette interface, the CE-158 RS-232 interface and the CE-159 RAM/ROM memory module). There is also information on the connectors used in the system (signal names on a pin-out basis), comprehensive system memory maps, and keyboard matrix and key code charts.

Another chapter discusses BASIC commands that were not mentioned in the original PC-1500 operating manual. *PCN* subscribers are already aware of most of the information provided in this section relating to such capabilities as PEEK, POKE, STATUS, CALL, CSAVE M and CLOAD M. Later in this section there is information on how variables are stored in memory and some system subroutine calls. A number of these machine language calls are already known to *PCN* subscribers, but there is some new information of interest to machine language programmers here, too.

Another chapter provides some machine language programming examples. There is a binary to hexadecimal conversion routine, a display inversion routine, a routine that will shift the display right or left by one dot and a few other relatively simple procedures.

The appendices includes schematic diagrams of the PC-1500, the CE-150, CE-151, CE-153, CE-155, CE-158 and CE-159.

While this book has a lot of information (about 160 pages in large 8-1/2 by 11 inch format), it is not particularly easy to digest. Much of the English appears as though someone was trying to translate directly from Japanese. You will have to re-read many passages in order for the material to make sense. *This is not a tutorial on machine language programming!* It is assumed that you already know how to perform such programming. However, if your interest lies in the area of performing such programming on a PC-1500/PC-2, then this book is practically a must as a reference manual. Ditto if you want to do interfacing to equipment of your own design, etc.

Where can you get a copy? My understanding is that they are only being made available through authorized Sharp distributors. Mort Rosenstein at Atlantic NorthEast Marketing, *P.O. Box 921, Marblehead, MA 01945*, (phone (617) 639-0285) says he has copies available. The price from him is $20.00 for delivery in the United States. Contact him for pricing information on deliveries outside the U.S.

First ROM Program Library Available

The first of the long-awaited ROM software libraries distributed by Sharp are now available in the United States. I have actually seen a CE-502A General Statistics Module in operation. My first impression was that it looked pretty good. We plan on having a thorough review of the package in the next issue of *PCN*. Check with your local Sharp distributor on pricing. I understand the list price of the CE-502A is $69.95, but you may be able to find it for $10 or $15 less.

Incidently, the rest in the series of modules are due to be released this summer. The Finance package is due at any moment. A Circuit Analysis, Business Graphics and a second Statistics module are due in June. July should yield general Graphics, Electrical Engineering and Mathematics modules. Keep checking with your local distributor.

Here Come More Portables

Sharp has announced that it will begin delivering a portable computer named the PC-5000 this fall. The battery operated unit, about the size of a small portable typewriter, will sport an unusual liquid-crystal display. It will be capable of presenting 8 lines of 80 characters in the text mode or 640 by 80 dots in a bit-mapped graphics mode.

There are other unique features. It will be equipped with 128K of RAM expandable to 256K! It also has 128K (bytes) of bubble memory and can be equipped with up to 128K of ROM.

The machine will use an 8088 microprocessor. It can be fitted with a high-density dot-matrix thermal/impact printer. This unusual printer can operate in a strictly thermal mode using thermal paper. Alternately, when a carbon ribbon is inserted, it can print in an impact mode on plain bond paper! The unit can also be equipped with a modem/autodialer.

Optional double-sided, double-density disk drives, capable of storing up to 320 kilobytes (each) will also be made available for the unit.

The PC-5000 will apparently be equipped with Microsoft Basic and a bundle of software from Sorcim, including a spreadsheet package, word processing, data base management and a communications program.

The PC-5000 will probably be priced at about $2000.00 initially in the United States. The optional printer will probably go for about $400. RAM expansion modules are reported to be in the vicinity of $150.00 for 64K.

I will have more information on the PC-5000 in the next issue.

Casio also plans to introduce a notebook-sized portable computer near the end of this summer. I bet there will be lots of others!

New Sinclair ZX-81/Timex 1000 Software

If you are looking for ZX-81/Timex 1000 software, write to American Micro Products, *705 North Bowser, Richardson, TX 75081*. They have just announced a new Timex/Sinclair line. – *Nat Wadsworth, Editor*

POCKET COMPUTER NEWSLETTER

Issue 26 — July

Photo *Sharp PC-5000 Portable Computer*

Another *Personal Information*™ product from SCELBI Publications.

SHARP STEPS UP IN SIZE

This fall Sharp Electronics Corporation will begin delivering a portable computer that it claims will combine the advantages of big computer capability with small computer convenience. The new machine, dubbed the PC-5000, weighs less than 11 pounds and can operate from a built-in rechargeable battery or from an a.c. adaptor. It is said to fit within an attache case.

It is supplied with 128K bytes of RAM memory, internally expandable to 256K (in 64K increments). Its typewriter-style keyboard is designed to facilitate word processing. Using an optional modem, the unit will function as a remote terminal.

The PC-5000 uses an 8088 microprocessor. In addition to the regular RAM memory, the unit has 128K of bubble memory storage. Another 128K of ROM in cartridges will be used to implement a disk operating system and BASIC.

The liquid crystal display folds over the keyboard when the unit is being transported. When open, the display presents up to eight lines of 80 characters of text or 640 by 80 dots (in excess of 51,000 pixels) for graphics in a bit-mapped mode. The display supports a full 256-character/symbol set.

A number of options may be added to the basic unit. A combination high-density dot-matrix thermal/impact printer inserts into the back of the PC-5000. It is claimed that this unit produces near letter quality copy. In the thermal mode it utilizes thermal paper. When a carbon ribbon is inserted, the printer will output on plain bond paper. The printer produces 80 characters per line at 12 characters per inch. The speed is approximately 37 characters per second. Optional 10 pitch formatting is also possible.

An optional 10-key modem/auto dialer can connect the computer to telephone lines through a standard Bell jack. The modem permits two-way conferencing through a loudspeaker provided on the face of the unit. Auto dialing of up to 10 different numbers can be accomplished using the manual keys. A communications software package can be used to access an essentially unlimited network. The modem allows up to 16 digits per number, has both ringer and receiver and can operate with touchtone or telepulse systems.

Double sided/double density disk drives, each providing up to 320K bytes of storage, will also be available. Alternately, programs can be loaded/saved on standard cassettes using a cassette interface.

Sharp promises initial software offerings including word processing, communications, spreadsheet analysis, executive planning and data base management which will be provided by arrangement with major software houses. Additional software compatability is said to be provided through the use of a MS-DOS system that will permit the use of existing MS-DOS software packages.

For additional information on the new Sharp PC-5000 Portable Computer, contact: *Sharp Electronics Corporation, Systems Division, 10 Sharp Plaza, P.O. Box 588, Paramus, NJ 07652.*

INSIDE THE SHARP PC-1250

Here is sleuth *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158,* with his report on the Sharp PC-1250:

Many users of Sharp's PC-1250 will have discovered by now that it has a number of BASIC statements and functions that are not covered by the instruction manual. A few of these were mentioned in Issue 24 of *PCN*.

The POKE, CALL, CSAVE M, CLOAD M and PEEK functions and the & prefix to indicate hexadecimal numerals operate in the same fashion as they do in the PC-1500. Statements in the RSV (Reserve) mode terminated by the @ symbol will also automatically execute as though they had been terminated by pressing the ENTER key when they are called into the display.

BASIC also contains COM$, ROM, INSTAT, OUTSTAT, SETCOM and KEY. These may appear in various valid statements such as SETCOM OFF, KEY OFF, KEY ON, OUTSTAT(numeric expression) and SETCOM, , , ,KI. These statements produce ERROR 8 ("device not present"). Does Sharp plan on making a modem for the PC-1250?

Although I have not been able to determine how they are used yet, the words ERROR and DEBUG are also tokenized by the 1250.

A Few Words on the CE-125 Printer

If you are trying to economize on paper, thermal rolls made for Texas Instruments calculators work fairly well. Rolls two inches in diameter can simply be laid in the paper chamber with the lid left open. The ones I have tried print in blue and they are not as legible as the ones furnished with the computer.

Keying SHIFT 6 will enter the symbol for the Japanese yen into the display.

A few additional characters may be obtained with the printer, but codes must be POKEd into a string variable to accomplish such output. Execute POKE &C698,&F5,1,0. This makes variable A a string variable containing the code 1. The zero byte terminates the string. If you now display A$, you will see some meaningless dots. But, LPRINT A$ will print the "not equal to" (≠) symbol. If the preceding POKE statement has the number 1 replaced by another value, other characters are printable, such as:

2 Greek sigma (Σ)
3 Boldface letter C
4 Boldface letter P
6 What is it?
7 And what is this?
9 Uppercase Greek pi (Π)
11 Hollow triangle
12 Solid triangle
14 Greek theta (θ)
15 Arrow pointing to the right
19 Character insertion marker
23 Symbol for yen
78 Underscore

Incidently, I hope that 1250 owners noticed the PRINT–LPRINT and PRINT=PRINT statements as described in the manual. (Too bad they did not incorporate that into the PC-1500!)

Memory Map

The accompanying table summarizes the accessible memory in the PC-1250. This accounts for 16K of ROM. It is believed that another 8K of ROM is located on the CPU chip and is not accessible to the PEEK directive.

Table *RAM Memory Map for PC-1250*

0000 – 3FFF	not used
4000 – 7FFF	ROM: operating system and BASIC interpreter
8000 – BFFF	not used
C000 C7FF	RAM: including RSV, user's BASIC program, variables and system registers.
C800 CFFF	not used in PC-1250
D000 – DFFF	duplicate of C000 CFFF (where memory exists)
E000 – EFFF	duplicate of F000 – FFFF (where memory exists)
F000 - F800	not used
F800 – F8BF	system RAM (F8C0 – F8FF filled with FF)
FA00 – FAFF,	FB00 – FBFF . . . FF00 – FFFF are duplicate addresses for F800 – F8FF

A complete dump of the ROM from 4000 – 7FFF indicates the absence of the tables of BCD numbers required in calculations of transcendental functions. Thus, it must be assumed that these tables constitute part of the 8K of ROM believed to be on the CPU chip.

Machine Language

The CPU machine language opcodes used by the PC-1250 do not bear much resemblance to those of the PC-1500.

Table *ROM Memory Map for PC-1250*

4000 – 4020	JUMP addresses
4021 – 402F	Messages: BREAK, IN, ERROR
4030 – 4063	Addresses of tokenized words in initial-letter order
4064 – 4129	Addresses of tokenized words in token-code order
412A – 43DF	Table of BASIC words with tokens and ROM routine addresses
43E0 – 4414	Table of codes for shifted keys
4415 – 446B	Table for keys, used to convert row/column signals from keyboard into codes
446C – 4503	JUMP addresses
4504 – 45E2	Continuation of codes for character generation
45E3 – 7FFF	Appears to be all machine language code
C000 – C02F	Reserve memory area
C030	Stop byte (FF) marking beginning of user program area
C031 – C5CF	BASIC program area. Another FF byte is used as a end of program marker. Variables A(27) and above, as well as dimensioned arrays are also stored in this area. They begin at C5CF and work down towards the user's BASIC program. Each dimensioned variable has a 6-byte header that identifies it by name, type, length, dimensions and number of bytes per array element.
C5D0 – C5D7	Variable Z (or Z$)
C5D8 – D5DF	Variable Y (or Y$)
. . . – . . .	etc.
C698 – C69F	Variable A (or A$)
C6A0 – C70F	System RAM. Pointers in this area give the low order byte first. A number of pointers in this area are used by BASIC to keep track of program locations and locations of variables. Some of the more important ones are: C6E1 – C6E2 START OF BASIC C6E3 – C6E4 END OF BASIC C6FC – C6FD START OF VARIABLES There is no pointer to START OF EDITABLE (MERGED) PROGRAM. MERGE operates differently from in a PC-1500. C6E5 C6E6 WAIT setting (low byte first) C6F8 – C6FA USING format settings C710 – C75F FOR/NEXT stack
C760 – C7AF	80-byte buffer used to form character strings
C7B0 – C7FF	80-byte buffer
F800 – F83B	Half of display memory
F83C – F83F	Mode settings. Bit 1 of F83C is Auto Print. DEGREE is set by bits 2 and 3 of F83C. GRAD by bits 2 of F83C and F83D. RADIAN by bit 2 of F83D. RUN, RRO and RSV are set by bits 0, 1 and 2 of F83E.
F840 – F87B	Other half of display memory
F87C – F87F	?
F880 – F8A7	Subroutine return address stack
F8A8 – F8AF	PASSword string
F8B0 – F8B7	Current value of random number
F8B8 – F8BF	Various settings. (F88A – F8B8 is the countdown timer storage area for the automatic 9-minute powerdown sequence.)

The information that follows is only a partial listing of the machine language instructions. The choices I have made for the names of the CPU registers are purely arbitrary since I have no technical information concerning the actual architecture of the CPU.

I have detected the presence of two flags, a zero flag (Z) and a carry flag (C). It is likely that other flags are also used.

The Accumulator

An 8-bit register, an accumulator, will be designated in my discussions by the letter A. The following list shows opcodes for operations that involve only this register:

OPCODE	MNEMONIC	OPERATION
02	LDA #nn	Accumulator loaded with immediate data nn
42	INA	A is incremented
43	DEA	A is decremented
58	RDA	Exchange digits of A
5A	RLA	Rotate bits to left through carry
D2	RRA	Rotate bits to right through carry
64	AND A #nn	Contents of A are ANDed with immediate data nn
65	OR A #nn	Contents of A are ORed with immediate data nn
66	BIT A #nn	Bit test. Flag Z is set or reset based on result of A being ANDed with nn
67	CPA #nn	Comparison. Flags C and Z set or reset based on the results of the subtraction of nn from A

Register W

This 16-bit register is used by some of the opcodes as a pointer to a location in memory. The symbol W in parentheses (W) will designate the contents of the location whose address is contained in register W. The high and low bytes of register W are designated WH and WL.

OPCODE	MNEMONIC	OPERATION
10	LDW #nnnn	W is loaded with the immediate value nnnn
11	LDWL #nn	WL is loaded with the immediate value nn
52	STA (W)	Contents of A stored into the location pointed to by W
57	LDA (W)	Contents of location pointed to by W loaded into A
D4	AND (W) #nn	Contents of (W) ANDed with the immediate value nn
D5	OR (W) #nn	Contents of (W) ORed with the immediate value nn
D6	BIT (W) #nn	Bit test. Flag Z set or reset based on result of (W) being ANDed with nn

Registers X and Y

Two more 16-bit registers, with high and low bytes designated as XH, XL, YH and YL are assumed to exist. The use of parentheses indicates the contents of the location pointed to by the X or Y registers.

OPCODE	MNEMONIC	OPERATION
03	LDXH #nn	Load XH with immediate value nn
06	INXL	Increment XL
07	DEXL	Decrement XL
C2	INXH	Increment XH
C3	DEXH	Decrement XH
26	STI A (X)	Store with increment. X is first incremented, then the contents of A are stored into (X)
27	STD A (X)	Store with decrement. X is first decremented, then the contents of A are stored into (X)
DA	EXA XH	Exchange A with XH
04	INY	Increment Y
05	DEY	Decrement Y
24	LDI A (Y)	Increment Y then load A with (Y)
25	LDD A (Y)	Decrement Y then load A with (Y)

Register S

The following discussion concerning the register arbitrarily designated S is somewhat speculative. However, repeated experiments have consis-

tently produced these results.

Register S appears to act as an 8-bit register that points to a stack located on the CPU chip. The effect of any opcode in the range from hexadecimal 80 to BF is to load S with a byte that causes S to point to one of the locations in that stack. I am arbitrarily calling these 1-byte addresses on the CPU chip 80 to BF. (They might actually, for instance, be 00 to 4F.) This stack does not appear to operate with PUSH or POP instructions.

OPCODE	MNEMONIC	OPERATION
80	LDS 80	S loaded with 80. (S) will designate contents of address 80
81	LDS 81	S loaded with 81. (S) will designate contents of address 81
..	.....	etc.
BF	LDS BF	S loaded with BF. (S) will designate contents of address BF

It appears that CPU registers A, X and Y are located in certain of the CPU stack locations as follows:

ADDRESS	REGISTER
82	A
83	XH
84	YL
85	YH
86	XL

Register W, however, is not located in this stack. Furthermore, subroutine return addresses do not appear to be stored here. Location 96 in this stack is used to save the TRACE mode setting!

The following instructions use (S), the contents of the address pointed to by register S:

OPCODE	MNEMONIC	OPERATION
55	STA (S)	Contents of A into (S)
59	LDA (S)	Contents of (S) into A
44	ADD (S) A	Contents of A added to (S) (without carry)
45	SUB (S) A	Contents of A subtracted from (S) (without carry)
46	AND (S) A	Contents of A and (S) are ANDed with the result left in (S)
47	OR (S) A	Contents of A and (S) are ORed with the result left in (S)
60	AND (S) #nn	(S) ANDed with immediate value nn
61	OR (S) #nn	(S) ORed with immediate value nn
62	BIT (S) #nn	Bit test. Flag Z set or reset as a result of (S) ANDed with nn
63	CP (S) #nn	Flags C and Z set or reset based on results of subtracting nn from (S)
C4	ADC (S) A	A added to (S) (with carry)
C5	SBC (S) A	A subtracted from (S) (with carry)
C6	BIT (S) A	Bit test. Z set or reset based on result of (S) AND A.
C7	CP (S) A	Flags C and Z set or reset based on subtraction of A from (S)
DB	EXA (S)	Exchange contents of A and (S)

One more instruction involving register S is:

OPCODE	MNEMONIC	OPERATION
30	STA S	A stored into S. (Values 00 to 7F have the same effect as 80 to FF.)

Flags

Flag C is set when the result of an addition produces a carry or when the result of a subtraction produces a borrow. It is cleared when no carry or borrow results. Note that in the case of subtraction, this is the opposite of the operation of the C flag in the PC-1500.

The Z flag is set whenever the result of an operation is zero. It is not affected by load and store instructions.

OPCODE	MNEMONIC	OPERATION
D0	SETC	Set flag C. (Flag Z is also set.)
D1	CLRC	Clear flag C. (Flag Z is set.)

Instructions for Branching and Subroutining

Jumps to absolute addresses are as follows:

OPCODE	MNEMONIC	OPERATION
79	JMP nnnn	Jump to address nnnn
7C	JZC nnnn	Jump if flag Z is clear
7E	JZS nnnn	Jump if flag Z is set
7D	JCC nnnn	Jump if flag C is clear
7F	JCS nnnn	Jump if flag C is set

Relative branch instructions exist for branching forward or backward through memory. The address reached by the branch is determined by counting from the final byte of the instruction, not from the initial byte of the following instruction.

OPCODE	MNEMONIC	OPERATION
2C	FWD nn	Forward branch nn bytes
28	FZC nn	Forward if flag Z is clear
38	FZS nn	Forward if flag Z is set
2A	FCC nn	Forward if flag C is clear
3A	FCS nn	Forward if flag C is set
2D	REV nn	Reverse branch nn bytes
29	RZC nn	Reverse if flag Z is clear
39	RZS nn	Reverse if flag Z is set
2B	RCC nn	Reverse if flag C is clear
3B	RCS nn	Reverse if flag C is set

Subroutine operations are controlled with the following directives:

OPCODE	MNEMONIC	OPERATION
78	JSR nnnn	Jump to subroutine at address nnnn
37	RTS	Return from subroutine

There Are More . . .

There are quite a few more opcodes whose operations have not been described in the preceding lists. You are invited to pursue those on your own. The PC-1250 does not appear well-suited towards machine language programming for a variety of reasons. However, the above information may be sufficient to allow those interested to dabble in the subject and proceed on their own. Good luck!

Compatability of BASIC

It is claimed that programs written for the Sharp PC-1211 (and the equivalent Radio Shack PC-1) may be CLOADed directly into the PC-1250 without modification. Although this is generally true, there are a few exceptions.

A common technique used for conserving memory in the PC-1211 is to omit closing parentheses in expressions. The PC-1250, however, will not permit the use of this technique. The omitted parentheses will need to be inserted, if there is room! (Remember, though, the PC-1250 does have room for an additional 14 bytes over a PC-1211.)

Another possible incompatibility can cause problems when using a printer. Any programs for the PC-1211 that used the "wrap-around" based on a 16-character line in order to produce a particular printed format will need to be modified.

The PC-1250 uses the same unconventional FOR/NEXT looping as the PC-1211 (and early versions of the PC-1500). That is, the control variable is tested *before* it is incremented. The 1250 can also use implied multiplication.

Some Speed Comparisons

In an effort to see just how much faster the PC-1250 is than the 1211, I ran the following "benchmark" program:

```
10 A=1
20 FOR X=1 to 999
30 GOSUB 70
40 NEXT X
50 PRINT A
60 END
70 A=A+A/√(A+X)
80 RETURN
```

This program is an attempt to give a somewhat balanced comparison in that it contains a loop, a subroutine and some computation. The results I obtained were as follows:

PC-1211 693 seconds
PC-1250 218 seconds
PC-1500 107 seconds

It might be of interest to note that the same program took 57 seconds on a Commodore desktop microcomputer.

One reason the PC-1250 is slower than the PC-1500 is that the 1250 doesn't use link bytes (following line numbers) to point to the beginning of the next line of BASIC. It has to examine each byte of code to determine where one line ends and the next line begins.

In comparing available program space in two computers, it is not enough to just count the number of bytes. One computer may use more memory than another for tokens and for line numbers. To illustrate, the above program required 72 bytes in the PC-1500, but only 56 bytes in the PC-1211 and PC-1250. At that rate, the PC-1250's 1438 bytes can handle almost exactly the same size program as the (unexpanded) PC-1500 with its 1850 bytes!

Table *PC-1250 Character Codes and BASIC Token Values*

00	End of line	51	A	90	TO	C0	GRAD
		52	B	91	STEP	C1	PRINT
10	␣ (insertion symbol)	53	C	92	THEN	C2	INPUT
11	Space	54	D	93	RANDOM	C3	RADIAN
12	"	55	E			C4	DEGREE
13	?	56	F	95	WAIT	C5	CLEAR
14	!	57	G	96	ERROR		
15	#	58	H			C9	CALL
16	%	59	I	99	KEY	CA	DIM
17	¥	5A	J			CB	DATA
18	$	5B	K	9B	SETCOM	CC	ON
19	π	5C	L			CD	OFF
1A	√	5D	M	9E	ROM	CE	POKE
1B	'	5E	N	9F	LPRINT	CF	READ
1C	;	5F	O				
1D	:			A0	SIN	D0	IF
1E	@	60	P	A1	COS	D1	FOR
1F	&	61	Q	A2	TAN	D2	LET
		62	R	A3	ASN	D3	REM
30	(	63	S	A4	ACS	D4	END
31	)	64	T	A5	ATN	D5	NEXT
32	>	65	U	A6	EXP	D6	STOP
33	<	66	V	A7	LN	D7	GOTO
34	=	67	W	A8	LOG	D8	GOSUB
35	+	68	X	A9	INT	D9	CHAIN
36	-	69	Y	AA	ABS	DA	PAUSE
37	*	6A	Z	AB	SGN	DB	BEEP
38	/			AC	DEG	DC	AREAD
39	^	7D	ASC	AD	DMS	DD	USING
		7E	VAL	AE	RND	DE	RETURN
40	Ø	7F	LEN	AF	PEEK	DF	RESTORE
41	1						
42	2	81	AND	B0	RUN	E0	Line 0--
43	3	82	>=	B1	NEW	E1	Line 1--
44	4	83	<=	B2	MEM	E2	Line 2--
45	5	84	<>	B3	LIST	E3	Line 3--
46	6	85	OR	B4	CONT	E4	Line 4--
47	7	86	NOT	B5	DEBUG	E5	Line 5--
48	8	87	SQR	B6	CSAVE	E6	Line 6--
49	9	88	CHR$	B7	CLOAD	E7	Line 7--
4A	.	89	COM$	B8	MERGE	E8	Line 8--
4B	E	8A	INKEY$	B9	TRON	E9	Line 9--
		8B	STR$	BA	TROFF		
4E	_ (underscore)	8C	LEFT$	BB	PASS		
		8D	RIGHT$	BC	LLIST		
		8E	MID$	BD	PI		
				BE	OUTSTAT	FF	Stop byte
				BF	INSTAT		

THE CE-502A GENERAL STATISTICS MODULE

This plug-in ROM module for the Sharp PC-1500/Radio Shack PC-2 contains 16K of statistical programs, all in BASIC, located in memory in the addresses zero to &3FFF.

Since the module is plugged into the slot that is also used by the RAM expansion modules (such as the CE-151, CE-155 or CE-159), only the 2 kilobytes of unexpanded RAM is left available for user programs when the CE-502A is installed. A user program in RAM must begin with an executable label. This is because such a program is separated from the ROM module by the equivalent of a MERGE. It should also be noted that REServe memory is taken over by the CE-502A ROM module.

The module contains seven programs:

1. Means and Moments
2. t-Test, Paired
3. t-Test, Unpaired
4. One-Way ANOVA
5. Two-Way ANOVA
6. Contingency Table
7. Ranked Sum Test

Execution of any of these may be initiated by use of a REServe key. The correct REServe key can be conveniently identified by displaying the REServe memory template.

To get an idea of the amount of flexibility provided by the programs in this module, consider the options given the user in the Means and Moments program. Data may be entered either from the keyboard or from cassette. If the printer is connected, a choice is offered as to whether input data (and output) are to be printed. Inputs may be in the form of either grouped or ungrouped data. Editing of data is permitted. Data can be recorded on tape and a backup copy made. The second moment (variance) may be calculated with either n or n-1 used in the denominator.

The outputs provided by this program include the arithmetic, geometric and harmonic means, the second, third and fourth moments about the mean, the skewness and kurtosis and the number of entries made. However, it will not give you the standard deviation.

The t-tests for paired and unpaired data are used for determining the level of significance of the difference between measurements taken from two populations. The use of the test for differences between population means (the unpaired t-test) is based on the assumption that the two populations are normally distributed, with equal variances. Again, there are options for printing or not printing, entering data from the keyboard or cassette tape, inputting grouped or ungrouped data and recording data onto tape. The computer requests input of the hypothesized difference in means, with a default value of zero. The output includes the value of t, but the program does not calculate the level of significance. You will need to use tables for that. The number of degrees of freedom, together with the number of x entries and y entries, is also given.

The unpaired t-test (also known as the paired-difference test) requires only the assumption that the *differences* are normally distributed. The outputs here include the mean and standard deviation of the differences, the value of t and the number of degrees of freedom. Again, however, you will need to use a t-distribution table to interpret the results.

The programs for one-way and two-way analysis of variance (usually shortened to "ANOVA") provide for a variety of ways of handling the input data. Outputs include all necessary information, including the value of the F statistic. The use of an F-distribution table is required to interpret the results.

The two remaining programs involve nonparametric tests. The contingency table program gives the user the usual cassette options involving data. In this program there is a pleasant surprise. Not only is the value of chi-squared calculated, but its probability is determined too! This avoids the necessity of having to check a table to determine the level of significance. This is particularly useful in view of the fact that almost all chi-squared tables contain only certain selected values.

The final program, the Mann-Whitney Ranked Sum test, provides a way of comparing two means *without* the need for the assumptions required by the t-test. A Shell sort incorporated within the program is used to rearrange the input values, from each of the two samples, into numerical order. (It is possible to display or print the rankings after they have been sorted.) Outputs include the rank sum, rank mean and rank variance for both samples. The test statistics U (indicated as w) and z are given, but you will have to provide your own tables again. (Although the module contains a routine for determining approximate values of normal probability in lines 28205 – 28340, for some strange reason it is not used by *this* routine!)

There are many statistical tests in existence. The writers of the programs contained in this module had to make some choices. They wisely decided to omit some of the common tests that require a relatively small amount of computation, such as tests of hypotheses and determination of confidence intervals involving means, variances and proportions for single populations. Some users, however, might like to have seen inclusion of the chi-square test, linear regression and correlation. But, even 16K of memory can not hold everything, particularly when so many bells and whistles are included in the package.

One nice feature used in all of the programs in this module is a flashing of the default value in the display when yes or no inputs are requested. For example, the display of:

RECORD DATA (Y/N)?

is shown with the letter N flashing on and off. This tells the user that if ENTER is pressed without an input having been made, the computer will interpret the response as having been N for no! This feature is easy to get used to and it is convenient.

I would rate the manual as excellent. Although such a manual obviously can not include an entire course in statistics, the writers did take the trouble to include at least a brief explanation of what each of the tests is about. The manual ends with a list of examples of applications for which each of the tests in the module would be appropriate.

The manual documents the programs well. There are listings of the variables used by each program. In many cases the formulas used are also provided. This manual was obviously written in the U.S.A. It is not an atrocious translation from some other language!

I have not found any real bugs in the program, but there is a matter that comes pretty close to qualifying as such. The choice of algorithms used and the way in which they are implemented can produce some inaccurate results in certain circumstances.

The results frequently obtained for moments M3 and M4, as well as those for skewness and kurtosis, appear to be the worst. To illustrate: If the data values 865, 866 and 867 are entered, the following results are obtained:

RESULT	GIVEN BY PROGRAM	CORRECT VALUE
M2	1.00005333	1
M3	0.10000000	0
M4	-37.0370370	0.66666666
Skewness	0.09999200	0
Kurtosis	-37.0330867	0.66666666

A negative value for kurtosis is, of course, impossible. Note that the second moment also is not extremely accurate.

One cause of the errors is that during the accumulation of the sums ΣX^2, ΣX^3 and ΣX^4, the 10-digit capacity of the computer places limits on the accuracy of the sums. Then the required subtraction of nearly equal quantities having small relative errors produces a difference value having a large relative error.

About all you can do about this problem is code your data. Unfortunately, no provision appears to have been made for this in the program. (In the example presented, if you subtract 866 from each of the given input values and use the results as inputs, you will get correct values for M2, M3, M4, skewness and kurtosis.)

It seems to me that a better solution would have been for the programmer to have used another algorithm. The mean, $\bar{X}$, could have been calculated first. Next the numerator of M2 could have been calculated as $\Sigma(X-\bar{X})^2$ rather than as $\Sigma X^2 - (\Sigma X)^2/N$. (With this approach it would not have been possible to have accumulated the combined data from several tape files. There would have been ways to have gotten around that too, but it would have been rather complicated.) Similar approaches to the calculation of M3 and M4 would have eliminated the

significant errors noted in the illustration.

There is another inaccurate procedure used in these programs. The use of exponentiation is detrimental to the accuracy of the t-test and ANOVA programs as well as the Means and Moments program. Using, for example, X^2 in place of X*X has two bad effects: it is much slower and there is a loss of accuracy. The exponentiation operation uses logarithms to calculate an *approximate* result. This often contains slight inaccuracies. These inaccuracies are greatly magnified when subtraction of nearly equal quantities is required. That is exactly what happens in many statistical calculations. For instance, the error in M2 in the earlier example is entirely due to the use of X^2 in the program.

The programs could have been designed to operate somewhat faster. Part of the blame for a loss of speed has to go to the use of the exponentiation operation. The computer's response sometimes seems irritatingly slow when YES/NO inputs are required. While the sorting routine used in the Mann-Whitney test is quite efficient, a machine language sort would have really moved things along.

I have one more rather minor complaint. In four of the programs, data inputs are first placed into a string variable. Their numerical values are then later determined by use of the VAL function. Besides slowing down operation somewhat, this prevents one from using expressions such as 2/3, π or 968-750 as inputs. As a consequence, if you want to code input data, it is not nearly as easy to do. Their reason for using this approach was apparently to permit the input of E to signal the end of data. It would not have required a great deal of imagination to have achieved such a result another way.

However, my overall assessment of the CE-502A module is that it is a powerful and useful package for anyone doing much analysis of statistical data. Just be sure to take the skewness and kurtosis results with a grain of salt!

This review presented by: *Norlin Rober.*

Program *RPN Octal--Decimal Hexadecimal Calculator*

```
100: CLS :CLEAR :A$
     ="0123456789AB
     CDEF":BA=16:BB
     =58:U=33:T$="&
     ":WAIT 0
110: ON ERROR GOTO
     730:GOTO 600
120: X$=""
130: K$=INKEY$ :IF
     K$=""THEN GOTO
     130
140: BEEP 1
150: K=ASC K$:IF (K
     >47AND K<BB)OR
     (K>64AND K<71
     AND BA=16)THEN
     LET X$=X$+K$:
     CLS :PRINT T$+
     X$:GOTO 130
160: IF X$=""THEN
     GOTO 240
170: IF X<>0THEN
     LET T=Z:Z=Y:Y=
     X
180: IF BA=10THEN
     LET X=VAL X$:
     GOTO 230
190: L=LEN X$:X=0:B
     =1
200: FOR M=0TO L-1:
     0$=MID$ (X$,L-
     M,1):P=VAL 0$
210: IF P=0THEN LET
     P=ASC 0$-55:IF
     P<0THEN LET P=
     0
220: X=X+P*BT:BT=BT
     *BA:NEXT M
230: PRINT T$+X$,X:
     GOTO 250
240: IF K=13THEN
     LET T=Z:Z=Y:Y=
     X:GOTO 120
250: IF K=13AND X=0
     THEN LET T=Z:Z
     =Y:Y=X:GOTO 12
     0
260: IF K=43THEN
     LET X=Y+X:Y=Z:
     Z=T:GOTO 610
270: IF K=45THEN
     LET X=Y-X:Y=Z:
     Z=T:GOTO 610
280: IF K=42THEN
     LET X=Y*X:Y=Z:
     Z=T:GOTO 610
290: IF K=47THEN
     LET X=Y/X:Y=Z:
     Z=T:GOTO 610
300: IF K=24THEN
     LET X$="":X=0:
     PRINT T$+X$,X:
     GOTO 120
310: IF K=09THEN
     LET K=X:X=Y:Y=
     K:GOTO 610
320: IF K=10THEN
     LET K=X:X=Y:Y=
     Z:Z=T:T=K:GOTO
     610
330: IF K=11THEN
     LET K=T:T=Z:Z=
     Y:Y=X:X=K:GOTO
     610
340: IF K=02THEN
     LET X=-X:GOTO
     610
350: IF K=15THEN
     END
360: IF K=31THEN
     GOTO 470
370: IF K=01THEN
     GOTO 500
380: IF K=78THEN
     LET X=-X-1:
     GOTO 610
390: IF K=22THEN
     GOTO 550
400: IF K=79THEN
     GOTO 570
410: IF K=83THEN
     LET S=X:GOTO 6
     10
420: IF K=25THEN
     LET T=Z:Z=Y:Y=
     X:X=S:GOTO 610
430: IF K=61AND X<0
     AND BA<>10THEN
     LET N=UL+X:S$=
     "":GOTO 640
440: IF K=85THEN
     GOTO 590
450: IF X$=""THEN
     GOTO 610
460: PRINT T$+X$,X:
     GOTO 120
470: IF BA=16THEN
     LET BA=10:BB=5
     8:T$="%":PAUSE
     "DECIMAL":GOTO
     610
480: IF BA=10THEN
     LET BA=08:BB=5
     6:T$="#":PAUSE
     "OCTAL":GOTO 6
     10
490: IF BA=08THEN
     LET BA=16:BB=5
     8:T$="&":PAUSE
     "HEXADECIMAL":
     GOTO 610
500: K$=INKEY$ :IF
     K$=""THEN GOTO
     500
510: BEEP 1:K=ASC K
     $:IF K=11THEN
     LET X=INT X:GOTO
     610
520: IF K=32THEN
     LET X=Y^X:Y=Z:
     Z=T:GOTO 610
530: IF K=24THEN
     LET X=0:Y=0:Z=
     0:T=0:X$="":
     PRINT T$+X$,X:
     GOTO 120
540: GOTO 120
550: GOSUB 670:X=(X
     2AND Y2)*B15*B
     15+(X3AND Y3)*
     B15+(X1AND Y1)
560: Y=Z:Z=T:GOTO 6
     10
570: GOSUB 670:X=(X
     2OR Y2)*B15*B1
     5+(X3OR Y3)*B1
     5+(X1OR Y1)
580: Y=Z:Z=T:GOTO 6
     10
590: PRINT "# BITS=
     ";U;" ";:
     INPUT U
600: BEEP 1:UL=1:
     FOR I=1TO U:UL
     =UL*2:NEXT I:
     CLS :PAUSE U;"
      BITS=";UL;" U
     P LIM":GOTO 61
     0
610: IF BA=10THEN
     PRINT T$,X:X$=
     "":GOTO 120
620: IF ABS X>UL
     THEN LET X=SGN
     X*UL
630: N=ABS X:S$="":
```

program listing
continued on next page

continuation of program listing from previous page

```
    IF X<0THEN LET
    S$="-"
640:X$=""
650:O=INT (N/BA):X
    $=MID$ (A$,N+1
    -BA*O,1)+X$:N=
    O:IF N<>0THEN
    GOTO 650
660:PRINT I$+S$+X$
    ,X:GOTO 120
670:B15=32768:IF
    ABS X>ULOR ABS
    Y>ULTHEN PRINT
    "TOO LARGE!":
    BEEP 2:GOTO 12
    0
680:IF X=0THEN LET
    X1=0:X2=0:X3=0
    :GOTO 700
690:X0=INT (X/B15)
    :X1=X-X0*B15:X
    2=INT (X0/B15)
    :X3=X0-X2*B15
700:IF Y=0THEN LET
    Y1=0:Y2=0:Y3=0
    :GOTO 720
710:Y0=INT (Y/B15)
    :Y1=Y-Y0*B15:Y
    2=INT (Y0/B15)
    :Y3=Y0-Y2*B15
720:RETURN
730:PAUSE "ERROR !
    !":BEEP 3:
    PRINT "":GOTO
    120
STATUS 1
                    2014
```

RPN CALCULATOR FOR PROGRAMMERS

This program simulates a RPN hexadecimal, decimal and octal calculator on the PC-1500. It was provided by: *Michael H. Frey, 1536 Pine Street, Philadelphia, PA 19102.*

The program allows the following functions: addition, subtraction, multiplication, division, square root, Y to the X power, NOT, OR, AND, 2's complement and one memory with recall.

The upper limit for calculations is initially set at 8589934592 (33 bits for a word length) with an accuracy of +/- one digit in the last place.

The simulator has a four register stack labeled X, Y, Z and T. The functions of the program are summarized in the accompanying table. Numbers are entered as in a normal calculator. Note that the hexadecimal mode also uses the "digits" A through F. Illegal characters are treated as an ENTER. The program recognizes that A – F are valid in the hexadecimal mode and that 8, 9 and A – F are not valid in the octal mode. Two numbers are normally displayed at a time. The exceptions are when actually entering a number or when in the decimal mode, the left hand number is hexadecimal or octal and the right hand number is the decimal equivalent.

The various numerical modes are indicated by preceding the left hand number by a & for hexadecimal, a % for decimal and a # for octal. Inputs must be integer numbers and the left hand number is always displayed as the truncated integer of the right hand number.

Table *Functions of RPN Calculator for Programmers*

KEY SEQUENCE/ FUNCTION	REGISTER CONTENTS AFTER OPERATION X	Y	Z	T	M=MEMORY
ENTER	X	X	Y	Z	M
CL	0	Y	Z	T	M
SHIFT CL	0	0	0	0	M
↕	Y	X	Z	T	M
↓	Y	Z	T	X	M
↑	T	X	Y	Z	M
S=STORE	X	Y	Z	T	X
RCL	M	X	Y	Z	M
+-*/=[]	Y[]X	Z	T	T	M
SML*change sign	-X	Y	Z	T	M
SHIFT ↑	Y↑X	Y	Z	T	M
SHIFT ^	Y^X	Z	T	T	M
N=NOT	NOT X	Y	Z	T	M
O=OR	X OR Y	Z	T	T	M
&=AND	X AND Y	Z	T	T	M
=	x(2'c) displays 2's complement of x if x is negative	Y	Z	T	M
MODE	hexadecimal->decimal->octal->hexadecimal->...				
0	SET NUMBER OF BITS				
OFF	RETURNS TO PC-1500 RUN MODE				

FROM THE WATCH POCKET

PC-1/PC-1211 owners with an interest in astronomy might want to write to *Fred Klein, 12225 Magdalena, Los Altos, CA 94022,* for information about a collection of programs he is self-publishing. The 100 page (8-1/2 by 11 inches) publication is described as "a collection of 13 programs for observers, astrophotographers and Dobsonians." Mr. Klein, holder of a Ph.D., appears to have written an astronomical works that will be of interest to serious astronomical enthusiasts.

Speaking of serious works, Inframetrics, Inc., *25 Wiggins Ave., Bedford, MA 01730,* has announced a "radiometric computer" that performs "non-contact temperature measurement calculations in less than one second." The unit appears to be a Quasar HHC packaged with special software and a custom calibration module. Write to the firm for pricing, (about $650.00 for the basic package), literature and additional technical details.

Prices are starting to drop on add-on memory modules for the PC-1500/PC-2. Radio Shack is selling their 4K modules for $39.00. Atlantic N.E. Marketing, *P.O. Box 921, Marblehead, MA 01945,* telephone (617) 639-0285, says they have a limited supply of CE-151 4K modules available at $25.00! Word is that the talked-about 16K RAM module coming this fall (September) from Sharp will be designated the CE-161, will have a list price of $170.00 and will be battery-backed. (The 8K battery-backed CE-159 has been dropped to $110.00.)

PCN is currently published 10 times a year. There will not be any August issue! Have a pleasant summer. – *Nat Wadsworth, Editor*

POCKET COMPUTER NEWSLETTER

Issue 27 - September

RADIO SHACK ANNOUNCES THE PC-3

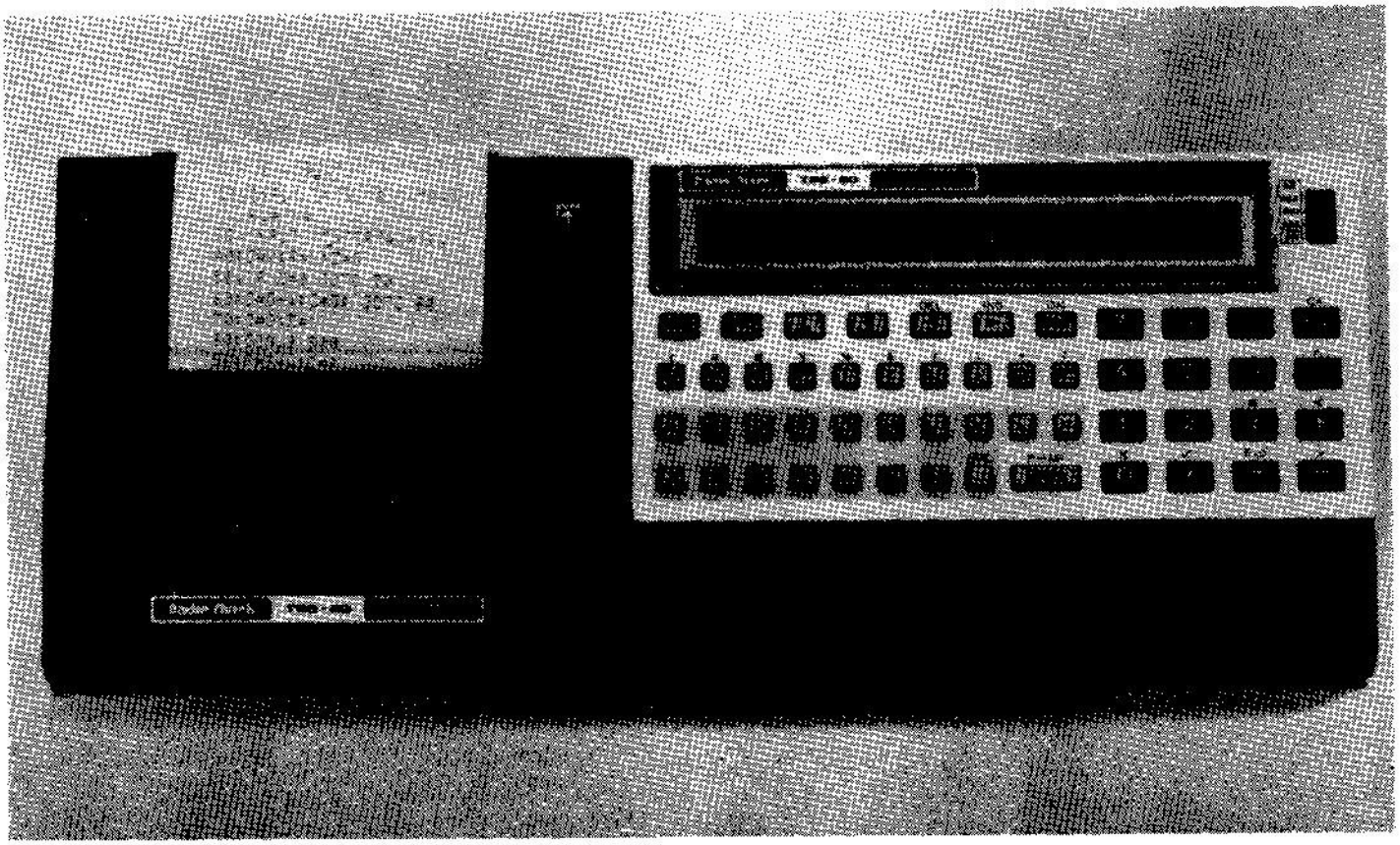
Tandy Corporation has announced the availability of a 4-ounce shirt-pocket computer that is programmable in BASIC.

The new PC has a 24-character liquid crystal display, a display accuracy up 10 digits and 1.4K of user-programmable memory. Besides operating in a programmable mode, it can be used as a direct entry calculator.

The PC-3 is said to be software compatible with the earlier PC-1. Thus, Radio Shack claims that their existing library of programs produced for the PC-1 will be immediately available for use on it.

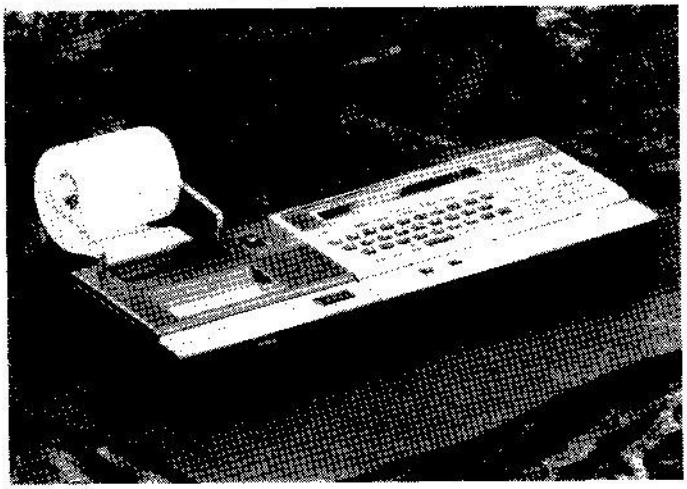
The U.S. price of the PC-3 is set at $99.95 from Radio Shack computer centers and stores. It is supplied with batteries and an instruction manual. *(Note: the PC-3 appears to be a custom-labeled version of the Sharp PC-1250.)*

A matching accessory, termed the PC-3 Printer/Cassette Interface is reported to be available at a U.S. price of 119.95. It enables the PC-3 pocket computer to load and store data using a cassette tape recorder. A thermal dot-matrix printer capable of printing 24 characters to a line at a rate of one line per second, provides for hardcopy of programs and data. The unit comes supplied with rechargeable batteries, an A.C. adaptor, some paper, cables to connect to a recorder unit and an operating manual.



PAPER HOLDER KIT FOR PC-1500 & PC-2

The *Bisi Beaver Paper Holder Kit* is an attachment for Radio Shack PC-2 and Sharp PC-1500 computers that will accomodate a standard 2-1/4 inch roll of calculator paper. The kit includes the paper holder, a template for proper alignment and the drilling of holes in the printer cover, and instructions for making the modifications. When not in use the paper holder and paper may be stored in the computer's carrying case.

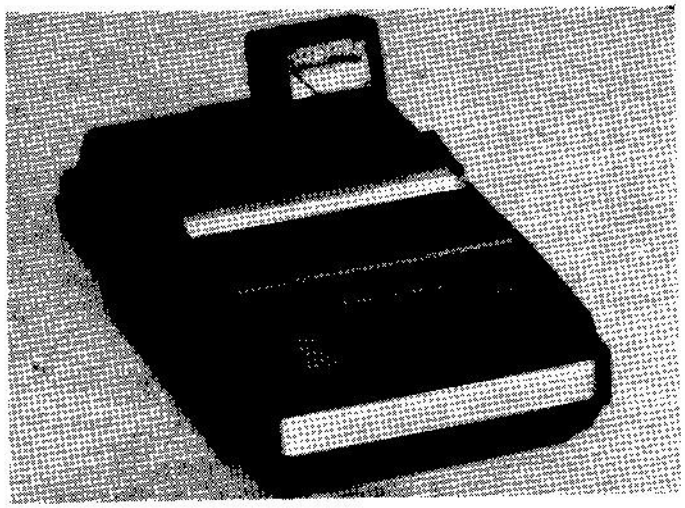


The kit retails in the U.S. for $24.95 (plus $1.56 sales tax in the state of Washington). Kits are normally shipped by UPS with shipping charges collect. Orders may be placed with: *Bisi Beaver Software Company, P.O. Box 53, Vashon, WA 98070.*

CASSETTE VOLUME MONITOR

If you use a standard audio cassette recorder to store computer programs, you know it can sometimes be difficult to get the volume adjusted properly when loading programs.

A new electronic device named *VU-LOAD* may be what you need to eliminate such problems. It enables you to "observe" the signal level at the ear jack of the cassette recorder. You may then adjust the level for the optimum volume setting.



If your cassette does not have an automatic level control, it is claimed that the *VU-LOAD* monitor will enable you to find the optimum volume setting at which your cassette recorder will "save" your programs. And, it will give a positive "save" indication with most other cassette players.

VU-LOAD may be mounted anywhere within reach of its 16 inch connector, which plugs into the ear jack of a cassette recorder.

The U.S. price is quoted as $20.95 plus $2.50 for postage/handling. It is claimed to be fully guaranteed and to be available from: *L & G Enterprises, Box 6854, Silver Spring, MD 20906.*

VARIABLES LISTER

This is a cross reference program for the Radio Shack PC-2 and Sharp PC-1500. It will list all the variables used in a program, showing them in alphabetical order (ascending ASCII). After each variable name is listed, the line(s) where that variable is (are) used will be shown. If a variable is used several times within a line (such as in A=A+1) then the line number is repeated the appropriate number of times.

The program distinguishes between all four possible variations of a variable assignment (i.e., A, A(), A$() and A$).

A *linked list* was used to conserve memory. Three major arrays are used by the program. T$ holds the variables used in the cross referenced program. M stores the starting node in the linked list for each variable name. O stores the linked list.

T$(n) = **Name of variable n.**

M(n) = **Index in array O of the first entry for the line numbers associated with variable n.**

O(1,M(n)) = **First line number associated with variable n.**

O(2,M(n)) = **Index in O of the next reference in O associated with variable n. That is, the next line number associated with variable n will be stored at O(1,O(2,M(n))). O(2,O(M(n))) holds the index in O of the next reference to the variable n. A value of zero in O(2,M(m) indicates the last reference to that variable.**

Consider the short example program here:

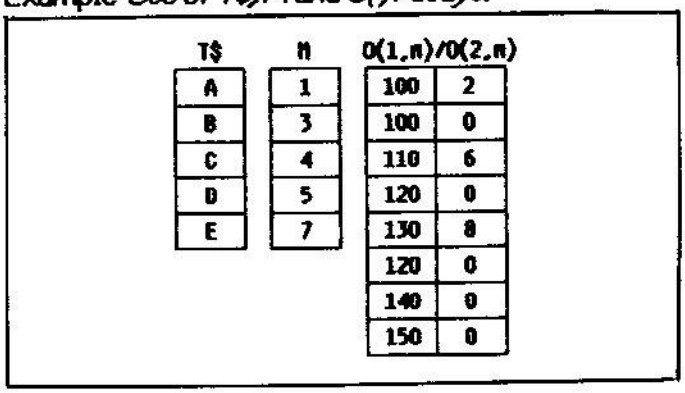
```
90 : REM TEST
100 : A = A + 1
```

```
110 : B = 12
120 : C = B + 13
130 : D = 14
140 : E = 15
150 : D = 0
160 : END
```

The accompanying diagram illustrates how the T$, M and O$ arrays in the Variables Lister program would appear when processing the above example program.

The arrays T$ and M have been dimensioned to 25 and the linked list to 250. This currently limits

Example *Use of T$, M and O() Arrays.*



T$	M
A	1
B	3
C	4
D	5
E	7

O(1,n)	O(2,n)
100	2
100	0
110	6
120	0
130	8
120	0
140	0
150	0

Program *Variables Lister.*

```
1000:REM  "XREF"
2010:CLEAR :T=1:N
     =1:WAIT 10
2020:DIM M(25),O(
     2,250),T$(25
     )*4
2030:DIM N$(0)*36
2040:FOR I=1TO 25
     :M(I)=0:T$(I
     )="":NEXT I
2050:FOR I=1TO 25
     0:O(1,I)=0:O
     (2,I)=0:NEXT
     I
2080:Q=PEEK &7869
     *256+PEEK &7
     86A
2120:Q=Q+5
2140:REM
2160:N$(0)=N$(0)+
     CHR$ (PEEK Q
     )
2170:Q=Q+1
2190:IF PEEK Q<>&
     0DGOTO 2140
2220:LPRINT N$(0)
     :LF 1:Q=Q+1:
     W$=""
2250:REM
2260:IF PEEK Q=&F
     FGOTO 2610
2280:L=PEEK Q*256
     +PEEK (Q+1):
     Q=Q+3:PRINT
     L
2290:IF PEEK Q=&F
     1AND PEEK (Q
     +1)=&ABTHEN
     LET Q=Q+PEEK
     (Q-1)-1
2310:REM
2320:IF PEEK Q=&0
     DGOTO 2560
2350:IF PEEK Q>&E
     5THEN LET Q=
     Q+2:GOTO 254
     0
2380:IF PEEK Q<&4
     1OR PEEK Q>&
     90GOSUB 6000
     :Q=Q+1:GOTO
     2540
2410:REM
2430:W$=W$+CHR$
     PEEK Q:Q=Q+1
2450:IF (&41<PEEK
     QAND PEEK Q<
     &90)OR PEEK
     Q=&24GOTO 24
     10
2480:IF PEEK Q=&2
     8THEN LET W$
     =W$+CHR$
     PEEK Q:Q=Q+1
2510:P=1:GOSUB 30
     00:W$=""
2540:REM
2550:GOTO 2310
2560:REM
2580:Q=Q+1
2600:GOTO 2250
2610:REM
2630:T=T-1:LPRINT
     "WORDS FOUND
     :";T
2640:N=N-1:LPRINT
     "REFERENCES:
     ";N
2650:GOSUB 4000:
     GOSUB 5000
2670:END
3000:REM "STOTRE"
3020:IF T>1GOTO 3
     080
3040:T$(T)=W$:M(T
     )=N:O(1,N)=L
3050:T=T+1:N=N+1
3070:GOTO 3400
3080:REM
3090:REM
3100:IF (P>T)OR (
     W$=T$(P))
     GOTO 3150
3120:P=P+1
3140:GOTO 3090
3150:REM
3170:IF P>TGOTO 3
     340
3200:P=M(P)
3220:REM
3230:IF O(2,P)=0
     GOTO 3280
3250:P=O(2,P)
3270:GOTO 3220
3280:REM
3300:O(2,P)=N:O(1
     ,N)=L:N=N+1
3330:GOTO 3380
3340:REM
3350:T$(T)=W$:M(T
     )=N:O(1,N)=L
3360:T=T+1:N=N+1
3380:REM
3400:REM
3410:RETURN
4000:REM "SORT"
4020:IF T<=1GOTO
     4230
4050:FOR I=1TO T-
     1
4070:K=I:C=I+1
4100:FOR J=CTO T:
     IF T$(K)>T$(
     J)THEN LET K
     =J
4110:NEXT J
4140:IF I=KGOTO 4
     190
4160:U$=T$(I):U=M
     (I)
4170:T$(I)=T$(K):
     M(I)=M(K):T$
     (K)=U$:M(K)=
     U
4190:REM
4200:NEXT I
4230:REM
4240:RETURN
5000:REM "PRTLIN"
5020:FOR I=1TO T
5040:P=M(I):J=1:
     LPRINT T$(I)
5050:REM GOSUB 70
     00
5070:REM
5090:LPRINT USING
     "######";O(1
     ,P);:P=O(2,P
     ):J=J+1
5110:IF P<>0GOTO
     5070
5130:LPRINT
5140:NEXT I
5160:RETURN
6000:REM "SP CHR"
6020:IF PEEK Q<>&
     22GOTO 6110
6040:REM
6060:Q=Q+1
6080:IF PEEK Q<>&
     22GOTO 6040
6110:REM
6130:IF PEEK Q<>&
     26GOTO 6250
6150:REM
6170:Q=Q+1
6190:IF (PEEK Q>=
     &30AND PEEK
     Q<=&39)OR (
     PEEK Q>=&41
     AND PEEK Q<=
     &46)GOTO 615
     0
6220:Q=Q-1
6250:REM
6260:RETURN
STATUS 1        1542
```

Example *RUN of Variables Lister.*

```
          TEST

        WORDS FOUND: 5
        REFERENCES: 8
        A
            100    100
        B
            110    120
        C
            120
        D
            130    150
        E
            140
```

the program to processing 25 different variables with no more than 250 total references. If these dimensions are not satisfactory, they may be altered. Remember, however, that increasing the size of these arrays will decrease the amount of memory left available for an application program. To change the dimension of T$ and M, change lines 2020 and 2040. Lines 2020 and 2050 are modified to change the dimension of O().

The program assumes that the first line of the application program begins with a REM statement. The contents of this REM are printed as a heading on the variables listing. It is a good idea to place the name of the program in this line.

The LCD contains the line number being processed during operation of the program.

Operation is straightforward:

1. Load XREF into memory.
2. MERGE your program.
3. RUN.

Thanks for this valuable utility program go to: *Diane Campbell, 220 Houston, League City, TX 77573.*

VERTICAL LISTER

The SIDELISTER program published in Issue 22 of *PCN* introduced the concept of vertical printing of BASIC program lines. As *Mel Beckman*, author of the article pointed out, his pioneering effort was slow in operation.

This program uses machine language to expand tokens into a string of characters. It will provide a vertical listing at about the same speed as a regular LLIST command.

Making the Vertical Lister Cassette

First enter the machine codes shown in the accompanying listing using either POKEs or a monitor program. (As with any machine language program, the correct entry of these codes is absolutely crucial. Check your work.) Save the ML program on cassette by executing:

CSAVE M "VERT LISTER (ML)";&7650,&76EE

Next, clear memory with the NEW command. Now enter the BASIC portion of the Vertical Lister as shown in the accompanying listing. Add this to the same cassette using the standard procedure:

CSAVE "VERTICAL LISTER"

Program *ML Portion of Vertical Lister.*

```
7650 B5 2A ED 79
7654 F4 01 8B 02
7658 B5 50 AE 77
765C 4F FD 88 CC
7660 99 B5 07 FD
7664 CA FD 5A 04
7668 AE 77 4E 6A
766C FB CD BA FD
7670 0A 45 28 45
7674 2A FD 88 CD
7678 10 40 FD 0A
767C B5 3A 51 44
7680 45 B7 E0 83
7684 0D B7 0D 8B
7688 07 FD 88 46
768C B5 01 8E 24
7690 FB 9A 28 45
7694 2A FD 88 48
7698 B0 4A 54 45
769C B9 0F 8B 44
76A0 FD CA 45 A6
76A4 89 39 05 26
76A8 89 35 46 46
76AC 46 05 D9 91
76B0 05 45 B9 0F
76B4 2A DF 28 A5
76B8 77 4E F9 A3
76BC 77 4F 16 91
76C0 07 20 16 89
76C4 02 68 00 83
76C8 08 22 1A B5
76CC 20 51 51 51
76D0 51 62 F5 88
76D4 03 A4 8B 03
76D8 B5 20 51 FD
76DC 0A 9E 5F 44
76E0 44 44 9E 49
76E4 84 B3 08 08
76E8 B7 C8 91 53
76EC FD 1A E4 00
```

Using the Vertical Lister

Load the BASIC program you desire to have listed into the computer. Make sure that STATUS 0 yields a value of at least 455. That is the amount of space required by the Vertical Lister program, including the space it uses for the dimensioned variable A$().

Program *BASIC Portion of Vertical Lister.*

```
1:"VL"INPUT "SIZ
  E? ";S:GRAPH :
  CSIZE S:ROTATE
  1:A=STATUS 2-
  STATUS 1:DIM A
  $(3*S-1)*76/S+
  4:X=216
2:CALL &7650,A:C
  =0
3:IF X=0GLCURSOR
  (0,-576):SORGN
  :X=216
4:X=X-9*S:
  GLCURSOR (X,0)
  :LPRINT A$(C):
  C=C+1:IF C<3*S
  IF A$(C)GOTO 3
5:IF PEEK A<255
  GOTO 2
6:GLCURSOR (0,-5
  76):TEXT
STATUS 1
                196
```

Insert the cassette containing the two parts of the Vertical Lister program into the tape player. Execute CLOAD M. After the machine language portion as loaded, execute MERGE to bring in the BASIC part of Vertical Lister. Now execute the command RUN "VL" and enter the desired print size (1 or 2) in response to the prompt. Your BASIC program will then be listed in vertical format. (Of course, the Vertical Lister program itself is not listed.)

Some Alternatives

The procedure described above makes it easy to produced vertical listings of programs that you have previously saved on tape. The Vertical Lister program itself takes only a small amount of memory and thus takes less than a minute to load. Furthermore, locating the ML portion in the strings variable area means it is unnecessary to have previously reserved memory space for it.

However, when used in the above manner, the VL program can not be used as an aid in debugging a program that is in the process of being developed. This is because the MERGE of the BASIC portion of the VL program prevents editing of the lines of the BASIC program that is under development. Also, string variables E$ to N$ must be left alone, since the area they reside in is occupied by the ML portion of Vertical Lister. (Note that the use of CLEAR would completely erase the ML portion of VL, too!)

One alternative to this situation is to locate the ML portion of the program elsewhere. Indeed, the ML routine has been written so that it is relocatable without any modification to it. (Of course, the CALL &7650 statement in line 2 of the BASIC part would have to be changed to conform to the new address of the ML routine.) If Reserve memory is not being used for anything else, it would be a good place for the ML routine. Another possibility would be to relocate the beginning of the BASIC storage area using the NEW XXXX statement. (The ML code for VL requires 159 bytes.)

If you desire to avoid the use of a MERGE in order to leave the BASIC program in a form whereby it can be edited, then the six lines of BASIC programming used by VL may be loaded prior to development of the program. (As an alternate, the BASIC portion of VL can manually be typed in at any appropriate point.) Naturally, when a MERGE command is not used to load the BASIC portion of VL, then Vertical Lister itself *will* be included in the vertical listing.

A closing note: This program only searches the token tables of the PC-1500 and CE-150. It will not look for tokens in the CE-158 or other devices. Hence BASIC programs that use statements unique to such peripherals can not be listed by VL.

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

THREE-DIMENSIONAL PROJECTIONS

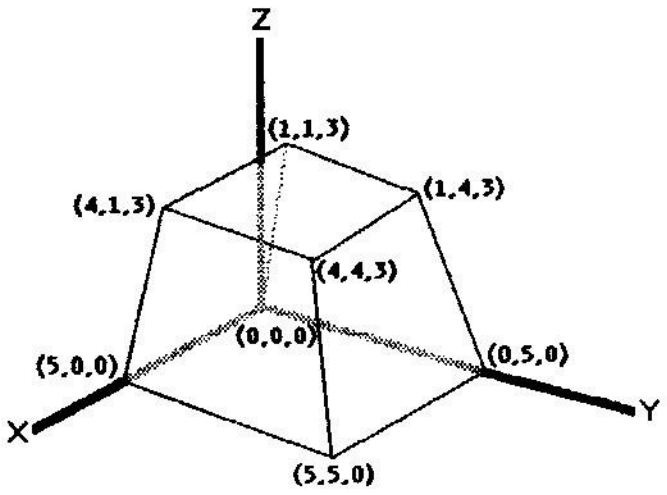
This program uses the PC-1500/PC-2 with its printer to plot the projection of a three-dimensional object onto a plane. Two kinds of projections are available. One is an *oblique* projection, in which the plane of projection is parallel to the XY plane. In the second type, an *axonometric* projection, the projecting rays are perpendicular to the plane of projection.

In either type of projection, the user selects the direction of the projecting rays by inputting values of L, M and N. These three values represent, respectively, the X-, Y- and Z-components of the direction that the projecting lines are to have. You may think of (L,M,N) as a vector in the direction of the projecting lines.

Taking the points in the order in which they are to be connected, the X-, Y- and Z-coordinates of each point must be specified in DATA statements. The letter S should appear in the DATA list to indicate the *start* of each sequence of points to be connected. (Its effect should become clear when the accompanying example is studied.) Use the letter T as the last item in the DATA list to signal *termination* of the drawing.

An Example

The truncated pyramid in the accompanying diagram is shown with the X-, Y- and Z-coordinates labeled at each of the points to be connected. These coordinates appear in the DATA statements in lines 100 to 103 of the program listing. Note, also, the use of S and T in these program lines. The letter S always precedes a sequence of points that are to be joined by line segments.



When the program is RUN, it first requests the input of the letter A or O. This indicates whether an *axonometric* or *oblique* projection is to be drawn. Next, the direction of the projecting lines must be specified by L, M and N. Note that L, M and N should be positive numbers. For an oblique projection, the use of 7, 4 and 3 results in a drawing with a fairly pleasing appearance. With axonometric projections, the choice 4, 3, 2 works well. (The "ideal" direction will depend on the object being represented.)

Finally, the program requests maximum and minimum values of X, Y and Z. These values determine the portion of three-dimensional space that is to be represented in the projection drawn. They are not necessarily the largest and smallest values of a coordinate used in the DATA statements. To retain correct scaling, the same size interval should be used in all three dimensions. To illustrate: the minimum values of X, Y and Z might all be zero, with the maximum values stated as 20, even if all three do not attain that value. (Of course, there may be cases where it is desirable *not* to use the same scale in all three directions. Perhaps a magnification is wanted in one dimension, for example.)

If you desire to re-execute the program, it is not necessary to re-enter those values that are not being altered. For instance, if you want to change just the direction of view, simply key ENTER without data when the prompts appear for the type of projection and the maximum and minimum values.

After the drawing has been completed, you have the option of keying ENTER to print out a listing of the parameters used in the drawing.

Program *Three-Dimensional Projections.*

```
10:INPUT "Type of
    proj (A/O)? "
   ;A$
11:INPUT "Directi
   on: L? ";L,"M?
    ";M,"N? ";N
12:INPUT "Min X?
   ";A,"Max X? ";
   B
13:INPUT "Min Y?
   ";C,"Max Y? ";
   D
14:INPUT "Min Z?
   ";E,"Max Z? ";
   F
20:CLS :IF A$="A"
   GOTO 30
21:K1=216/(L+M):K
   4=0
22:XH=K1*M/(B-A):
   YH=K1*L/(D-C):
   XU=K1*N/(B-A):
   ZU=K1*L/(F-E):
   YU=0:GOTO 40
30:K1=L*L+M*M:K2=
   216/√(2*K1):K3
   =K2/√(K1+N*N):
   K4=108-K2*(L+M
   )/2
31:XH=K2*M/(B-A):
   YH=K2*L/(D-C):
   XU=K3*L*N/(B-A
   ):YU=K3*M*N/(D
   -C):ZU=K1*K3/(
   F-E)
40:S=1E99:T=2E99:
   GRAPH :
   GLCURSOR (K4,-
   270):SORGN
41:READ X:IF X=T
   GOTO 44
42:IF X<SREAD Y,Z
   :GOSUB 60:LINE
   -(HC,VC):GOTO
   41
43:READ X,Y,Z:
   GOSUB 60:
   GLCURSOR (HC,V
   C):GOTO 41
44:GLCURSOR (0,-8
   0):TEXT
50:PRINT "Key ENT
   ER to list inp
   uts"
52:LPRINT "L=";L:
   LPRINT "M=";M:
   LPRINT "N=";N:
   LPRINT
53:LPRINT "Min X=
   ";A:LPRINT "Ma
   x X=";B:LPRINT
   "Min Y=";C:
   LPRINT "Max Y=
   ";D:LPRINT "Mi
   n Z=";E:LPRINT
   "Max Z=";F
54:END
60:HC=XH*(B-X)+YH
   *(Y-C):VC=XU*(
   B-X)+YU*(D-Y)+
   ZU*(Z-E):
   RETURN
100:DATA S,0,0,0,5
    ,0,0,5,5,0,0,5
    ,0
101:DATA 0,0,0,1,1
    ,3,4,1,3,4,4,3
102:DATA 1,4,3,1,1
    ,3,S,4,1,3,5,0
    ,0
103:DATA S,4,4,3,5
    ,5,0,S,1,4,3,0
    ,5,0,T
STATUS 1        878
```

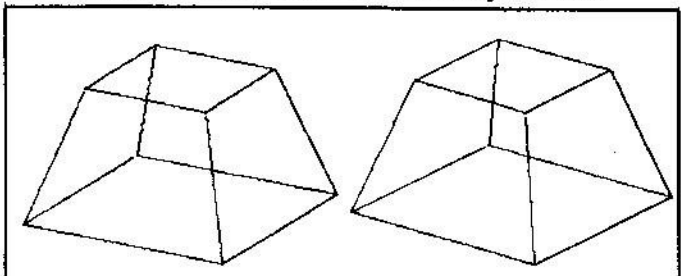
After the drawing has been completed, you have the option of keying ENTER to print out a listing of the parameters used in the drawing.

Special Cases

An oblique projection having a foreshortening factor of 1/2 is called a *cabinet* drawing. An example of a cabinet drawing may be produced by using L = 10, M = 4 and N = 3 in the program.

An *isometric* projection is an axonometric

Example *RUNs of 3-Dimensional Projections.*



```
L= 7                    L= 4
M= 4                    M= 3
N= 3                    N= 2

Min X= 0                Min X= 0
Max X= 5                Max X= 5
Min Y= 0                Min Y= 0
Max Y= 5                Max Y= 5
Min Z= 0                Min Z= 0
Max Z= 5                Max Z= 5
```

projection in which the projecting lines make equal angles with the X-, Y- and Z-axes. This kind of projection results when L, M and N are equal.

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

COUNT-DOWN TIMER PROGRAM

OK, all you PC-1/PC-1211 fans, here is a little routine that will enable you to use your PC as a timer that beeps when a specified interval has elapsed:

```
10  BEEP 1:PAUSE " TIMER PROGRAM"
20  Y=12099.24205
30  INPUT "TIME (H.MMSS):";T:W=INT(Y*DEG
    T + 80000 - 2)
40  W=W-1:IF W>80000 GOTO 40
50  BEEP 3:PAUSE "TIME IS UP!":GOTO 50
100 "C":PAUSE "CALIBRATION -->":USING
110 INPUT "TRUE TIME(H.MMSS):";S
120 X=DEG T/DEG S:X=X*Y
130 PRINT "NEW Y= ";X:END
```

Example Run

Want to know when exactly 14 minutes and 35 seconds have elapsed? Start the program with a RUN command:

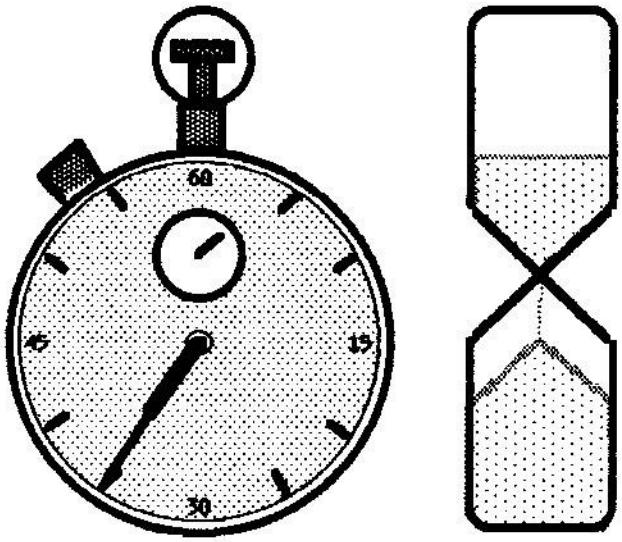
```
RUN
TIMER PROGRAM
TIME (H.MMSS): 0.1435
. . . . 14 minutes, 35 seconds later . . . .
Beep . . . Beep . . . Beep
TIME IS UP!
```

Calibration

If, after a trial run of a few hours, you find that the timer is off, you may calibrate the program against another clock or watch. Say that on a test run of exactly 3 hours, you find that the program starts to beep after a true time of 2 hours, 59 minutes and 37 seconds. Just run the calibration program as shown here:

```
RUN 100  (or SHIFT/C in the DEF mode)
CALIBRATION -->
TRUE TIME (H.MMSS): 2.5937
NEW Y= 12125.06395
```

After executing the calibration routine, you would need to place your PC into the PRO mode and change the calibration constant Y in line 20 to the new value. Your PC will now be custom-calibrated!



Sharp PC-1250/1251 and Radio Shack PC-3 owners: setting Y = 46141.17733 in line 20 seems to yield pretty good results for this program when installed in one of these models. You should be able to tweek the timing up, if necessary, using the calibration subroutine. Try it!

You can use your portable timer to do a three-minute egg, remind you when your parking meter is about to expire or when it is time to gracefully exit from the speaker's podium!

This program was skillfully crafted by: *Emerich Auersbacher, 41 King Street #2, Belleville, NJ 07109.*

FROM THE WATCH POCKET

Where are the 16K RAMs? On the way, is the latest word, and expected to be available in the U.S. by mid-September. The list price is still quoted as $169.95. Be careful if you plan to upgrade to a 16K module and are using machine language routines. It may be that the 16K module is addressed such that it would not be compatible with your present address-dependent machine language programs! If you use strictly BASIC programs, however, you should be able to simply plug in the new 16K module and load and execute programs with nothing to worry about except how to use up all that lovely RAM!

Of course, the introduction of the new 16K modules is likely to put the heat on anybody thinking about producing some kind of better program/data loading device. Filling up all that memory using an audio tape recorder can take 20 - 30 minutes. That can begin to get a bit aggravating for those who are trying to use PCs to save time! Where is a nice wafer drive or *anything* that will quickly load stuff into a PC-2/PC-1500?

Interested in trying to make money with your computer? **Computer Business Information Exchange,** *P.O. Box 4759, Santa Barbara, CA 93103,* has a catalog describing some 100 reports that give information on using a computer to make money. Write to them and ask for their catalog. Or, you might try phoning them at (805) 963-9580.

Rudy M. Hunger at **RMH Computers,** *P.O. Box 421, CH-6030 Ebikon, Switzerland,* says they have kits available that will act as an interface between the CE-158 RS-232 connector and a Brother EP-20. The EP-20 can then serve as a computer-driven printer.

The Radio Shack Model 100 portable continues to receive much praise. A college in Texas, **Dallas Baptist College,** has issued a requirement that *all* entering students (in all disciplines, not just science and engineering) be equipped with their own Model 100. Students pay for the unit through a special registration surcharge that is spread over four semesters. A college spokesman said in adopting the measure that: "Computer literacy is not an option for the educated man or woman of the 1980's. It's a requirement." A college spokesman also added: "We looked at eight or nine other computers that either fit in the pocket or in the briefcase. Radio Shack just simply did the best all-around job for the price."

Speaking of briefcase portables, it seems that the Sharp PC-5000 will not arrive in September. The latest word I have gleaned from unofficial sources is that it may be available by December. That should be just in time for it to land smack in the middle of a lot of other rumored entries to the field. The grapevine is now whispering that even outfits like Apple and Commodore are feverishly working on candidates for the "notebook" class.

I have heard that Sharp will soon be releasing a ROM programmer that will enable small businesses to "burn" their own PROMs. No solid information on capability and prices yet, but I have heard figures of $1500 for the system, $50 to $60 each for the PROMs. Figuring a markup of 2:1, that would mean software suppliers would have to be reselling the ROMs (with software installed) for $150 - $180 each. Whew! Can the market bear it? If you think so, ask a Sharp representative about it.

Sharp is also apparently on the verge of releasing a combination calculator/computer with a lot of transcendental functions available directly from the keyboard. I understand it will be called the model EL-5500. That indicates that it is considered part of their "calculator" (versus "computer") line. It seems to be similar to a 1250 with souped-up capabilities. The price will likely be $99.95 in the U.S. It seems there will also be a little printer made available for the EL-5500 that may sell as low as $70.00. Could it be that this little unit may also work with the PC-1250 in a pinch? It would sure save some money over the CE-125 if that turns out to be the case.

Say, did Sharp score one on Tandy? Radio Shack's PC-3 is a custom copy of the PC-1250. Now word is that Sharp will provide U.S. customers with a PC-1250A, which is known as a 1251 outside the states. This 4K version of the 1250 may be sold at about the same price as the Shack's PC-3. That should do wonders for their PC-3 sales.

By the way, note the $119.95 price on Radio Shack's Printer/Cassette Interface for the PC-3. Say, do you suppose you could just plug a 1250 into that and save $50 over the list price on the CE-125?

Next issue *PCN* will increase to 16 pages. This will necessitate our switching from first class to third class mailing. If you want to receive your copy of *PCN* in a timely manner, then please make sure that the address shown on the latest mailer is current. Third class mail is returned to the sender for remailing rather than automatically being forwarded by the postal service. Thus, it is vital that we have your correct address and that you keep us informed of any changes. Please forward any change of address notice to the attention of our Subscription Department.

P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

Issue 28 - October

Photo *Gavilan Mobile Computer with Revolutionary Touch Control Panel.*

Another *Personal Information* (tm) product.

GAVILAN READYING MOBILE COMPUTER

Gavilan Computer Corporation, of Campbell, California, is completing 18 months of development work on an exciting new highly portable "mobile" computer. Self-contained, and weighing just 9 pounds complete with batteries that will provide eight hours of operation, the unit is described by the firm as "a complete mobile office in a briefcase for executives and professionals." Measuring just 11.4 by 11.4 by 2.7 inches, the unit does indeed fit with room to spare inside a standard business briefcase.

Photo *Gavilan and Printer Fit in a Briefcase.*

Perhaps the most exciting feature about the unit is the simplified human interface. The user commands the machine through an integrated touch panel which is described as a "solid state mouse." Merely touching a finger to this panel causes a pointer on the display screen to follow its motion. When the pointer highlights a desired choice, the operator just lightly taps the panel to invoke the selection. The ability to visually maneuver objects on the screen by using a single finger gives users the power to quickly become the system's boss.

The $3,995.00 basic system includes 32K of user RAM memory, 48K of system software, a built-in 3.5 inch floppy disk (320K formatted), a built-in modem, full keyboard and 10-key numeric pad, 8 line by 80 column LCD display, an external battery charger, and the revolutionary "finger mouse" panel. A strong complement of software is also supplied at no additional charge. A Gavilan operating system, 5 Gavilan application modules and the industry standard MS/DOS operating system are included. The Gavilan application modules, supplied on plug-in capsules that are about one-half the size of a package of cigarettes, include a spreadsheet, a forms processing data manager, word processor, communications controller, and a so-called "portable secretary."

The system can be expanded to include two disk drives, have up to 336 kilobytes of memory and may be equipped with a portable printer unit that weighs just five pounds. Other options include an accoustic coupler for communicating over standard telephone lines, a vehicle power adapter for operating the unit from a car, and plug-in memory expansion capsules that may contain 32 kilobytes of RAM, ROM, PROM or EPROM or combinations thereof. The capsules also hold their own lithium batteries which provide up to one year of RAM backup capability.

Photo *Gavilan Computer, Printer, Modules.*

The Software is Revolutionary

Working with computers can be a complex and frustrating experience, especially for beginners. The special human interface software and finger mouse built into the Gavilan mobile office makes using the computer easy for everyone. Here is how the system operates:

The user plugs a small module, called a capsule, containing the application program (actually on an integrated circuit) that he or she wants to use (or, alternately, inserts a microfloppy diskette into the built-in disk drive).

The user presses a "view" button on the control panel. This immediately causes a "desk-top" to appear on the screen. The desk-top uses icons, (symbols) and names to show the files and documents available to be worked on.

To indicate a selection from the desk-top, the user simply places a finger on the touch panel that is immediately beneath the display screen. Moving the finger along this panel causes a pointer to move on the display. When the pointer highlights the item wanted, a tap on the touch pad causes the document to be activated. It is not necessary to use cursor keys or memorize codes, document or file names in order to get the computer to operate.

The 8-line liquid-crystal display permits a data file to be easily scrolled left, right, up or down using scroll buttons on the control panel. There is also a system-wide command called *zoom* that is rather unique. It condenses an entire document so that it fits on the screen providing the user with an overview of the document's format. Any section of the document may then be accessed by using the finger mouse to move a "window" over the desired area and then tapping the control panel.

Help is available at any time from the computer itself rather than having to refer to a manual or instruction book. Touching the HELP button on the control panel invokes an overlay that presents four basic questions: What just happened? Where am I? What is this I am seeing? What do I do next? The finger-controlled pointer may then be used to select any of those questions for answering. The computer responds appropriately for the context. The system can also provide definitions of the function buttons and menu selections, if needed. The net result of this constant on-line assistance is that it is not necessary to carry along a user's manual or operating guide.

The Gavilan Capsuleware application programs provided with the unit are fully integrated. That means that the interface between the user and the programs operates similarly for all programs. As long as all the desired application modules are on-line (up to four capsules may be installed at one time), information may be transferred from one application to another. For instance, if the user is writing a financial report, spreadsheet values may be written directly into the text without having to load a separate program. Alternately, if the spreadsheet figures are updated, the corresponding figures in the text report will be revised automatically!

The five fully integrated application packages provided with the Gavilan mobile computer are briefly summarized as follows:

CapsuleWord -- provides full word processing capability including text editing, formatting and printing. Standard text or letters may be stored, recalled and modified. Features include global word search and replace, move/copy block, word, sentence and paragraph. Multiple fonts, page and paragraph formatting.

CapsuleCalc -- provides spreadsheets for laying out financial and other calculations, automatic recalculations, ability to include and print results in reports and other documents.

CapsuleOffice -- provides "to do" list, time recording, expense reporting, call reports, routes and schedules, appointments, tickler file and activity reports.

CapsuleComm -- provides ability to access commercial and proprietary data bases to inquire and select standard news, stock, library, events, airline schedules, hotel availability, games, corporate mail, etc. Allows communications between similar or dissimilar computers.

CapsuleForm -- provides the ability to design user-specific forms for data entry and printing. The fields associated with the form may be simply data entry fields or fields that are created as a result of arithmetic operations with other fields.

High-Powered Development Team

The Gavilan machine has been put together by an impressive development effort. Its design has been spearheaded by the founder and President, Manny A. Fernandez, who was formerly head of world-famous Zilog, Inc., of Z80 chip fame. F. John Zepecki, Vice President of Hardware Engineering, previously directed system development at Magnuson Computer Systems. John P. Banning, Ph.D., Vice President of Software Development, who also was a manager at Zilog, was instrumental in developing new computer architect for Amdahl Corporation in the past.

The development work has been fully supported by some 8.5 million dollars in venture capital financing and a 12 million dollar credit line with Bank of the West (of San Jose, California).

Competitive

The U.S. list price of $3995.00 complete with the Capsuleware software, the built-in floppy disk, communications modem, and the remarkable "finger mouse" control panel makes this an attractively priced unit. While priced modestly above a comparably equipped portable, such as the Sharp PC-5000, the highly integrated software, ease of use, compact size, and fact that the manufacturer is U.S. based, will likely make it highly attractive to many prospective portable computer users in the United States as well as elsewhere.

To obtain additional information write to: *Gavilan Computer Corporation, 240 Hacienda Avenue, Campbell, CA 95008.*

CASIO FX-802P INCLUDES PRINTER

Casio, Inc., 15 Gardner Road, Fairfield, NJ 07006, has recently introduced several pocket computers that sport a built-in 20-character thermal printer. Of interest to scientist and engineers is the model FX-802P. This unit is similar in most respects to a model dubbed the PB-300, except that it makes a number of transcendental functions available at the stroke of a special function key.

The unit weighs in at a reported 9.1 ounces with batteries installed. The printer section is powered by its own rechargeable nickel-cadmium batteries that can print about 3,000 lines per charge. The computer is powered by a pair a lithium cells that provide roughly 100 hours of operation. The PC measures 3/4 inch high by approximately 7 inches in width and 3-1/2 inches in depth. At this size it pretty well "stuffs" a pocket.

The 5 by 7 dot-matrix LCD display has 12 character positions for displaying text or numbers in 10-position signed mantissa format or 8-position mantissa, 2-digit exponent fashion. (All internal calculations are to 12 mantissa digits.) A total of 114 characters, including upper and lower case letters of the alphabet can be represented on the display. In addition, the display has individually controlled legends for indicating the status of the unit. These legends include: EXT, RUN, WRT, DEG, RAD, GRA, TR, PRT and STOP.

Fifty-five keys, including function and shift buttons, make up the ASCII encoded keyboard. Ten of the keys are arranged as a numeric keypad and are bordered by the most commonly used mathematical operators: plus, minus, multiply and divide. Alphabetical keys are arranged in standard typewriter QWERTY format. There is a large space bar at the bottom of the keyboard, which is unusual for a PC.

The unit is programmable in a version of BASIC that includes many functions such as square roots, powers, trigonometric and inverse trigonometric, logarithmic, exponential and generation of random numbers. User memory provides for 1,568 program steps and may be partitioned into 10 separate programs.

An excellent introductory BASIC programming manual is provided with the PC. This is a well written publication, not a stilted translation that sometimes accompanies imported PC models. There are about 15 simple program included in this publication. A second, more sophisticated manual of about 150 pages filled with applications programs is also included with the unit.

A rapidly expanding libary of programs is being made available on cassette tapes. These are loaded via the optional model FA-3 tape recorder interface.

The U.S. list price of the FX-802P is $150.00.

Photo *Casio FX-802P Pocket Computer with Built-in Thermal Printer.*

CASIO SOFTWARE MODULES

Casio has announced the availability of an entire series of programs on cassette for the FX-700P and FX-802P. The following is a list and brief summary of the capabilities of those said to be on the market. Casio claims more programs are under development.

Business/Finance

Travel Pak: Accumulate expenses in up to 10 categories. Prints out each entry and category totals. Organize travel expenses. $19.95.
Decision Evaluator: Computer will analyse your problem based on given variables. Output preducts your probability of success and control. $24.95.
Days & Dates: Calculates the day, date or calendar in the future or past. $19.95.
Financial Analysis I: A series of 9 programs may be used to calculate present and future values, annuities, periodic withdrawel and term of IRA investments. $24.95.
Financial Analysis II: Borrowing analysis using bank methods. Bankers and bank customers would use this program to calculate loan values, yielding same values as bank's computer. $24.95.
Mortgage Amortization: Determine payments of loan using amortized values. $29.95.
Mortage Analysis: Shows mortgage acceleration obtained by adding payments that reduce total payout. Also compares time payments to yield best mortgage value. $29.95.
Installment Sales Package: Calculates premiums related to the purchase of cars, houses, businesses, etc. $39.95.
Retail Sales Package: Tracks commissions, daily sales totals, product discounts. $19.95.
Matrix-Calc: Spreadsheet information from rows and columns is input and "what-if" variations may be made to obtain various outputs. $29.95.
Word Frame: General purpose filing system that allows sorting of long fields of data. $24.95.
Bank Package: Calculates loan values, early withdrawels, loan payments, penalties, etc. $39.95.
Auto Dealers Package: Calculates leasing rates, pricing of cars with loan variations, down payments and financing terms, etc. $39.95.
Real Estate Package: Calculates retail value of houses, buildings, businesses based on square footage. May be used on commercial or residential property. Calculates payments. $29.95.
Yardstick: Analyze sales performance against monthly and yearly sales goals/quotas. For sales people and management. $24.95.

Educational

Guess Again: Science, geography, trivia, history. An accompanying book gives the questions, the computer checks your answers. $14.95.
Alphabetizer: Learn alphabetizing. Store your favorite names in proper order. Presented in game form. $14.95.
Words: Hangman and word scrabble provide a way for children to learn as they play. $14.95.
Math for Kids I: Four different levels of difficulty for grade levels three through six. Randomly presented examples teach and test in one operation. $14.95.
Math for Kids II: More questions and higher difficulty levels. $14.95.
Scramble: Advanced word game helps teach spelling and tests powers of concentration. $19.95.
Understanding Inflation: Calculates value of money yesterday, today and tomorrow. Shows loan paybacks in terms of real dollars. $19.95.
Electronics: Learn principles of electronic calculations. Presents useful equations such as Ohm's Law, log functions, power calculations, etc. Can also be used in designing and solving electronic circuits. $24.95.

Games/Entertainment

Coder/Decoder: Secret code generator puts your message in a special code. Information is retrieved using your own password. $19.95.
Executive Decision Maker: The computer helps you with decisions. $14.95.
Game Pak 1: Vegas-type games - jackpot slots, Hi-Lo, Mindbender. $14.95.
Game Pak 2: Blackjack, dice games, golf and rocket game. Uses LCD computer graphics. $14.95.
Reflex: The computer tests your reactions. $14.95.
Puzzles: Cryptogram, assorted puzzles, thinking games. Used with accompanying booklet. $14.95.
Advanced Puzzles: Concentration and other board games to test your brain power. $19.95.

General Interest

Personal Finance: Checkbook, simple loans, calculation of monthly payments. $19.95.
Loan Analysis: General loan analysis, add-on interest, declining balance loans. $14.95.
Budget: Provides for twenty-two categories, shows under/over budget, year-to-date, adjusts budget to real spending, creates end-of-year report that may be used on income tax form. $19.95.
File: General purpose file system allows user to set up any file structure for phone numbers, products, names, etc. $24.95.
Biorhythms: Generates a chart of predicted good, bad and marginal days. $19.95.
Smart Shopper: Allows price versus quantity evaluation. Also useful for shopping, car loan, credit card payment comparisons. $24.95.
Coupon Clipper: Know what coupons you can use

by asking your computer. Tallies coupon values and tells how much is saved. $24.95.

Metric Converter: Converts between measuring systems. $19.95.

Score Keeper: Keep score on the golf course, at the ball game, playing cards, etc. Works with almost any kind of scoring game. $14.95.

Home Inventory: Maintain a computerized list of valuable items, serial numbers, date of purchase, value. $24.95.

Aviation: Complete flight package. Weight and balance calculations, general navigation, fuel consumption, flight time, drift angle, true air speed, corrected heading and more. Use the computer print out to fill out flight plan. $24.95.

Teachers Pet: Monitor class performance and that of individuals. Computer plots grade/performance curve. Data stored on tape for updating. $24.95.

Boot Strap: Compares tax-free investments versus self-funded non tax-free investments. Comparison shows real yield after taxes. $24.95.

Programming the FX-700P/FX-802P: Instruction book, data cassette and voice tape introduce programming. Examples in the book are followed by the computer. $29.95.

Advanced Applications and Concepts: Utility routines are provided and explained. Use as is or modified in your own programs. $29.95.

Math Frame: Flexible formula format allows variations in formula inputs. Results are then calculated. Up to three variables may be altered simultaneously. $24.95.

Base Conversion: Convert one base value to any other base (in the range 2 - 20) via base 10. Can be loaded with other programs. $19.95.

USING THE CE-158 (RS-232C) INTERFACE

Robert E. Fieux, P.O. Box 72, West Barnstable, MA 02668, has gained some valuable experience with the Sharp RS-232C interface (CE-158). This interface enables the Sharp PC-1500 or Radio Shack PC-2 to operate an external printer or to connect to a remote computer via telephone. This latter procedure permits linking into such networks as The Source or CompuServe. Many people feel the manual furnished with the CE-158 is difficult to understand. This article is intended to assist those who may have problems getting the RS-232C peripheral to operate. It is based on practical experience.

Compatability

When purchasing an RS-232C device, it is advisable to buy one of the same brand as your computer. Although the RS-232C ports on both the Sharp and Radio Shack are the same and will work interchangeably, the computer keyboards differ in layout. A keyboard template is provided with each brand-name RS-232C peripheral which is intended to fit the keyboard of the matching computer. If you cross brands, the template will not fit your computer keyboard. *[Note that while the RS-232C ports are identical, the Sharp unit provides an additional parallel port for directly driving a Centronics type printer. -N.W.]*

The RS-232C port will route information to a full-size external printer without additional accessories. However, if you wish to use the unit to communicate via telephone, you will also require a modem. The J-Cat Modem by Novation, Inc., will be discussed in this article.

Operating an External Printer

Connect the RS-232C device to your computer following the instructions in the manual, which are clear. Be sure your computer is turned off. Plug your printer into the top (parallel) connector. If you intend to use a modem, plug it into the RS-232C (bottom) connector. If you are using a J-Cat modem, be sure to make the connector pin change described on page 6 in Issue 24 of *PCN*. Always apply power to the computer first, then turn on the remaining peripherals.

The computer is only able to direct an external printer to output a line length of 74 characters. While a full line actually consist of 80 characters, the software uses some bytes for program line and print directions. The program permanently stored in the CE-158 ROM controls the length of the printed line as well as whether lines are single or double spaced. Other options are possible as discussed in the manual. The essential element in routing print instructions to an external printer instead of the CE-150 printer is the use of the command OPN "LPRT" with the computer in the RUN mode.

A Simple Word Processor

Switch to the PRO mode to type in the desired text. In this mode the computer is acting as a word processor. Each line must be numbered, as in writing a program, and each line number should be followed by a print command. For example, LPRINT" will cause the printer to start a line at the left hand margin. The command LPRINT TAB 5;" will cause the line to be indented five spaces.

These print command may be set up on the softkeys under one of the three Reserve Mode sections. To skip one line, as between paragraphs, use the command FEED 1.

For additional control over the printing layout, use the CONSOLE command. This command can only be issued after OPN "LPRT". When the PC is turned on, the default state of the CONSOLE command is 80,1. This means that the line length is set at 80 characters and lines are single spaced. For shorter lines, substitute the number desired in place of 80. To double space lines, issue the command CONSOLE 80,1,1.

When you are ready to print your text, switch to the RUN mode. Type RUN and press ENTER. (Be sure all peripherals are turned on and that you have entered the OPN "LPRT" command.)

Telephone Communications

Install the J-Cat modem according to the manufacturer's instructions (with the addition of the pin change mentioned earlier). When the modem is not in use, keep the Manual/Auto selector switch on Manual. If this is not done, the telephone may not receive normal incoming calls or be able to properly make outgoing voice calls.

Since everything is reset to default values when the PC is turned off, it is necessary to set up the computer for terminal operations each time you wish to use the communication facilities. I believe it is best to print the received data by dumping it from the receiving buffer (where it is stored as it comes in). Otherwise, the material will be lost when you end the terminal operation. This method also eliminates the problem of missing out on information that crossed the display too fast to be properly comprehended. The following initial entries should be made with the computer in the RUN mode in order to prepare the PC for terminal mode operation. (Conclude each directive by pressing the ENTER key.)

1. Enter OPN "LPRT".
2. Enter the CONSOLE command if line spacing is to be other than single spaced and 80 characters per line.
3. Enter the communications directive for the computer as follows: SETCOM 300,7,E,1. (Verify by entering COM$.)
4. Enter the printer control instruction as: SETDEV PO. (Check by entering DEV$.)
5. Enter TERMINAL to switch to the Terminal mode. (Do not enter the Terminal mode by using the DTE command as this appears to set the PC improperly.)

When the Terminal mode has been entered the display flashes the message:

ENTER MENU SELECTION

This is followed by:

Terminal: Ent Aut Quit

Now press the SHIFT key, then the scroll up key. A menu should appear on the LCD that reads:

Output: Ext Trc Dsp Etx

Press the softkey under Ext. The display will now query: EXT. PRINTER OFF? (Y/N). Type N and press ENTER to select the external printe. Then tap the scroll down key until the original menu (**Terminal: Ent Aut Quit**) appears again.

Press the softkey underneath Ent and the cursor (a dash) will appear on the LCD. This indicates that you are now in the Terminal mode and that the PC is ready to operate as a terminal. After setting the J-Cat modem appropriately, you may dial a distant computer.

To set up the J-Cat, place the selector switch on Automatic. Pick up the telephone handset and dial the number of a remote computer. After the normal ring tone, you should receive a connect tone which is usually a high-pitched steady whistle. When you hear this connect tone, press the Connect key on the J-Cat. This will produce a loud hum and both indicators should light. Hang up the phone.

You should now be able to receive messages from the computer you dialed. Additionally, messages that you type on your keyboard should be transmitted to the remote computer. If you are on a network such as CompuServe, follow their sign-on and operating protocols.

Dumping Received Data to a Printer

All of the data received from a remote computer goes into a buffer. (When the buffer overflows, the earliest data is dropped.) When you disconnect from the remote computer, but while still in the Terminal mode, you can recall the contents of the buffer by using the scroll keys. If you press SHIFT and the scroll up arrow, you will see the top line stored in the buffer. From this position you can dump all the data in the buffer to the printer. Just press the key marked PRINT on the template (softkey number three).

Exiting from Terminal Operation

When you want to exit the Terminal mode, press the BREAK (ON) key. The original Terminal menu will appear on the LCD:

Terminal: Ent Aut Quit

Press the softkey underneath Quit (F6) and the PC will return to the RUN mode.

Don't forget to set the select switch on the J-Cat back to Manual!

Once you have used this procedure a few times, go back and re-read the manual. You will probably find that many parts of it will become more understandable as you gain experience.

RS-232C INTERFACE PRINT FORMATTER

A limitation of the Radio Shack RS-232C Interface is the inability to change the 18 column, CSIZE 2 output format while in the Terminal mode. This can be particularly frustrating when dealing with numeric data formatted for 40 or 80 column printers because of the wrap-around feature. However, the data stored in RAM can eventually be accessed using BASIC. From there, it can be output to a printer in any CSIZE or format desired.

This program, provided by *Robert Stock, 304 Horseshoe Lane, Downingtown, PA 19335,* serves to simulate a print spooler. (A "print spooler" is a portion of a program that handles data as it is being prepared for output. It places the data into a buffer. On large computer systems the buffer is generally in disk memory.) The program permits scrolling through the data, forwards or backwards, by full and half screens or by single characters. The user may print at CSIZE 1 in the normal fashion. Alternately, "sideways" printing may be invoked at a 42 character width for CSIZE 2 or 84 character width using CSIZE 1. (The sideways printing routine is derived from Mel Beckman's *Sidelister* program that appeared in Issue 22 of *PCN*.) This program also enables the user to save and load data using cassette tape.

Program *RS-232C Print Formatter.*

```
 10:" "TEXT :CSIZE
    1:INPUT ">>>?
    ";L$:GOTO L$
 15:END
 20:"SET"CLS :K=
    STATUS 2:GOTO
    10
 25:"GO"CLS :WAIT
    0:PRINT "  ==>
     <<1 <13 <26 P
    RT SIDE"
 30:F=ASC INKEY$ -
    16:IF F<1OR F>
    6THEN 30
 35:CLS :GOTO 100*
    F
100:N=ASC INKEY$ :
    P=PEEK K
110:IF N=17THEN
    PRINT CHR$ P;:
    K=K+1
120:IF N=13THEN 25
130:GOTO 100
200:K=K-1:GOSUB 45
    0:GOTO 25
300:K=K-13:GOSUB 4
    50:GOTO 25
400:K=K-26:GOSUB 4
    50:GOTO 25
450:S=K:FOR J=1TO
    25
460:Q=PEEK S:PRINT
    CHR$ Q;:S=S+1
470:NEXT J:WAIT :
    PRINT CHR$
    PEEK (S+1):
    RETURN
500:P=PEEK K:
    LPRINT CHR$ P;
510:IF P=13THEN LF
    1:TAB 0
520:K=K+1:GOTO 500
600:INPUT "CSIZE =
     ? ";Z:CLS :X=
    212-Z*6
610:GRAPH :
    GLCURSOR (X,0)
    :SORGN :ROTATE
    1:CSIZE Z
620:X=0:L=0:C=0:M=
    0:GOTO 680
630:IF C>MLET M=C:
    IF M>(84/Z)LET
    M=84/Z
640:C=0:X=X-Z*10:
    GLCURSOR (X,0)
650:L=L+1:IF L=INT
    (20/Z)GOSUB 67
    0
660:RETURN
670:L=0:X=0:Y=-(Z*
    M*6.4):
    GLCURSOR (X,Y)
    :SORGN :M=0:
    RETURN
680:IF C>(84/Z)
    GOSUB 630
690:P=PEEK K:
    LPRINT CHR$ P;
700:IF P=13GOSUB 6
    30
710:C=C+1:K=K+1:
    GOTO 680
720:"SAVE"CLS :
    INPUT "Filenam
    e? ";F$:CSAVE
    MF$;STATUS 2,
    STATUS 3:GOTO
    10
730:"LOAD"CLS :
    INPUT "Filenam
    e? ";F$:CLOAD
    MF$;STATUS 2:
    GOTO 10
STATUS 1
                    766
```

Operating Instructions

Run the program by pressing DEF/SPACE. At the >>>? prompt, input SET and press ENTER to zero the spooler and initialize the printer. Alternately, type GO and press ENTER to see the Command Menu:

==> <<1 <<13 <<26 PRT SIDE

Select from the menu as follows:

Press F1 (==>) to scroll forward through the data in the buffer. Hold the F1 key down for continuous scrolling. Press ENTER to exit to the Command Menu.

Press F2 (<<1) to move backwards by one character. Press ENTER to return to the Command Menu.

Press F3 (<13) to move backwards by a half screen (13 characters). Press F4 (<26) to move back by a full screen. Either function may be terminated by pressing ENTER.

Press F5 (PRT) to print normally at CSIZE 1. This gives a 36 column format which is great for networks such as Compuserve, The Source or Dow Jones.

Press F6 (SIDE) to print sideways. Respond with a 1 or 2 to the prompt for CSIZE. A 1 will start the printer in an 84 column by 20 lines per page format. A 2 will print in a 42 column by 10 lines per page arrangement.

Once in the print mode (initiated by either F5 or F6), press BREAK to stop printing. DEF/SPACE may then be used to re-initialize the printer or re-run the program.

If you want to save data on cassette tape for future use, set up the recorder as if to CSAVE in the usual manner. At the >>>? prompt type SAVE and press ENTER. Respond to the prompt with an appropriate file name. Press ENTER at the end of the name to send date to the tape recorder. If the data buffer is not completely filled, listen to the audio sound made by the PC as the data is transmitted. When the sound becomes a steady tone, no more data is being sent. Press BREAK and then the CLear key to end the SAVE operation.

(When RAM is full, the tape SAVE operation will end automatically.)

To recover previously saved data, set up the recorder as if to CLOAD in the usual manner. At the >>>? prompt, type LOAD and press ENTER. Input the desired filename when prompted and conclude the name with the ENTER key. The data will be fed to the PC. If the file was made from a full RAM, then the loading operation will conclude automatically. Otherwise, a steady tone will indicate the end of data. Press BREAK and CLear to end the load operation when this tone is heard.

A Tip

If you use an acoustic modem, such as the Radio Shack Model AC-3, you can tap into the data going over the telephone lines. This can be accomplished through the use of an inexpensive in-line telephone pickup. Record the data on tape. The recorded data can then be played back through an earphone that has been mounted in foam rubber and inserted into the modem microphone cup. With the modem in the originate mode and the tape recorder volume set properly, the data can then be fed into the RS-232C interface. With the PC in Terminal mode, you can receive this data as though it was an actual telephone communication. Practically unlimited amounts of data can be stored on tape and played back as desired for viewing and/or printing!

FRACTIONAL BASE CONVERSION

Here is a program that will convert between binary, octal, decimal and hexadecimal. It does it without resorting to the use of double-digits (such as 10 and 15 in place of A and F). It has fractional

Program *Fractional Base Conversion*

```
 1:REM S.ETHERIDGE 6-83
10:PAUSE "  BASE CONVER
   TER W/F": CLEAR :
   DIM W$(1)*24,F$(1)*2
   4,T$(24)*2,D$(12)*2
20:INPUT "  BASE: ";B:
   INPUT "  CONVERT TO:
    ";C: INPUT "  NUMBE
   R: ";W$(1):L= LEN (W
   $(1))
30:FOR S=1 TO L:T$(S)=
   MID$ (W$(1),S,1): IF
   T$(S)="." LET R=S:S=
   L
40:NEXT S
50:IF R<>0 LET F$(1)=
   RIGHT$ (W$(1),L-R):W
   $(1)= LEFT$ (W$(1),R
   -1)
60:L= LEN (W$(1)):E=L:P
   =0:X=0:Y=0: GOTO 80
70:L= LEN (F$(1)):E=0:P
   =1:X=0:Y=1
80:FOR S=1 TO L:H=1
90:IF P=0 LET T$(S)=
   MID$ (W$(1),S,1):E=E
   -1
100:IF P=1 LET T$(S)=
   MID$ (F$(1),S,1),E=S
   *(-1)
110:IF B<>8 THEN 130
120:IF T$(S)="8" OR T$(S
   )="9" LET S=L: GOTO
   10
130:IF B=16 GOSUB 330
140:X=X+ VAL (T$(S))*B^E
   : NEXT S
150:IF P=0 LET W$(1)=
   STR$ (X)
160:IF P=1 LET F$(1)=
   STR$ (X)
170:IF Y=1 THEN 200
180:GOTO 70
190:IF R<>0 THEN 140
200:W= VAL (W$(1)):F=
   VAL (F$(1)):A=W+F:
   IF C=10 LET W$(1)=
   STR$ (A): GOTO 320
210:L= LEN (W$(1)):W$(1)
   ="":X=0:Y=0: IF L=0
   THEN 250
220:X= INT (W/C):Y=(((W/
   C)-X)*C):W=X:H=2:T$(
   S)= STR$ (Y): IF C=1
   6 GOSUB 330
230:W$(1)=T$(S)+W$(1):
   IF W=0 THEN 250
240:GOTO 220
250:IF VAL (F$(1))=0
   THEN 320
260:W$(1)=W$(1)+".":M=
   LEN (F$(1)):P=M:X=0:
   Y=0
270:P=P-1
280:X= INT (F*C):Y=(F*C)
   -X:F=Y:H=2:T$(S)=
   STR$ (X): IF C=16
   GOSUB 330
290:W$(1)=W$(1)+T$(S):
   IF Y=0 THEN 320
300:IF P=1 AND X=0 THEN
   280
310:IF P<>-1 THEN 270
320:PAUSE "ANSWER BASE "
   ;C: PRINT W$(1):
   GOTO 10
330:IF D$(1)<>"" THEN 36
   0
340:IF H=1 LET I=1,J=12,
   K=1
350:IF H=2 LET I=12,J=1,
   K=-1
360:FOR D=I TO J STEP K:
   READ D$(D): NEXT D:H
   =0: RESTORE
370:IF T$(S)=D$(1) LET T
   $(S)=D$(2): RETURN
380:IF T$(S)=D$(3) LET T
   $(S)=D$(4): RETURN
390:IF T$(S)=D$(5) LET T
   $(S)=D$(6): RETURN
400:IF T$(S)=D$(7) LET T
   $(S)=D$(8): RETURN
410:IF T$(S)=D$(9) LET T
   $(S)=D$(10): RETURN
420:IF T$(S)=D$(11) LET
   T$(S)=D$(12): RETURN
430:RETURN
440:DATA "A","10","B","1
   1","C","12","D","13"
   ,"E","14","F","15"
```

capabilities. And, it does not require that you enter each digit separately.

Steven L. Etheridge, 14451 Tyler Street, Sylmar, CA 91342, submitted this program and adds the following general comments.

The program is designed to run on a Sharp 1250 or 1500. (It should do nicely on a Radio Shack PC-2 or PC-3.) To use the program, just load it and type the command RUN. Respond to the queries for the base (of the number you will be inputting), the "convert to" base and then the actual number that you wish to have converted. The number may be up to 24 characters and may be integer, fractional or a combination. (The limit of 24 characters includes the radix point!) That is all there is to it!

The program is broadly organized as follows:

Lines 10 & 20	**Initialization and data input.**
Lines 30 & 40	**Scan for fractional number.**
Lines 60 - 200	**Convert input to decimal.**
Lines 210 - 310	**Convert decimal to final base.**
Line 320	**Display answer.**
Lines 330 - 440	**Hexadecimal translation.**

The program only has a few error traps, due to the limited memory in a PC-1250. Those with a PC-1500 may want to add to the trapping that was included to catch the use of 8 and 9 while working in octal.

HEXADECIMAL LOADER

William Delinger, Northern Arizona University, Department of Physics, Box 6010, Flagstaff, AZ 86011, submitted this program. It may be used to load information that is in hexadecimal notation directly into memory. While designed primarily for the PC-1500 and PC-2, changing the line numbers to a lower range should also make it useful to PC-1250 and PC-3 users who are dabbling with machine language.

Program *Hexadecimal Loader.*

```
1000 REM  HEXADECIMAL LOAD PROGRAM
1010 INPUT "STARTING ADDRESS=?";S
1020 INPUT "HEX=?";H$
1030 IF H4="STOP"THEN STOP
1040 L=LEN(H$)
1050 IF L<>2THEN 1130
1060 D1=ASC(LEFT$(H$,1))
1070 D2=ASC(RIGHT$(H$,1))
1080 IF D1>64AND D1<71LET D1=(D1-55)*16:
     GOTO 1110
1090 IF D1>47AND D1<58LET D1=(D1-48)*16:
     GOTO 1110
1100 GOTO 1130
1110 IF D2>64AND D2<71LET D2=(D2-55)+D1:
     GOTO 1150
1120 IF D2>47AND D2<58LET D2=(D2-48)+D1:
     GOTO 1150
1130 PAUSE "WHAT?"
1140 GOTO 1020
1150 POKE S,D2
1160 S=S+1
1170 GOTO 1020
1180 END
```

METEORS GAME FOR PC-2/PC-1500

PCN rarely publishes games. However, we are making an exception for a submission by *David Hergert, 4714 Chickering Avenue, Cincinnati, OH 45232.* His combination BASIC and machine language (hybrid) program illustrates how ML routines can be called upon to enhance critical portions of a program. The program is elegant in its playing simplicity, yet can be rather challenging -- just try getting a perfect score (10 hits) when the game is set for its highest skill level (1).

To play the game, imagine that you are Captain of the spaceship that appears on the display. As you are traveling along you suddenly encounter a meteor shower. There are 15 meteors in the shower but you only have 10 laser-missiles with which to defend yourself. (You have learn to judge which meteors will probably miss your craft!)

Once the program has been loaded, use a RUN command to start the action. Respond to the skill level query according to courage. Level 4 is for beginners. Level 1 brings on the meteors at their fastest rate. Watch out! Any meteor landing a solid hit on your spaceship ends the game. A successful mission on your part, brings a visual reward.

If you want to do some tinkering with the program, variable G in line 40 controls the display time of the laser. The laser firing is displayed via a ML routine that uses an exclusive OR instruction to invert the center row of dots. Line 190 of the BASIC program calls a ML counter that displays the contents of the accumulator for a short amount of time.

Good luck, Captains!

Program *Meteors*

```
10:POKE &47C0,&48
   ,&76,&4A,&0,&5
   ,&BD,&88,&41,&
   4E,&4E,&99,&8,
   &4C,&77,&8B,&6
20:POKE &47D0,&48
   ,&77,&4A,&0,&9
   E,&12,&9A,&FD,
   &6A,&FD,&A8,&C
   D,&10,&80,&FD,
   &2A
25:POKE &47E0,&62
   ,&6E,&01,&81,&
   07,&FD,&A8,&CD
   ,&10,&80,&9E,&
   E,&9A
27:WAIT 50:PRINT
   "PRESS + TO FI
   RE"
30:INPUT "ENTER S
   KILL LEVEL(1TO
   4) ";M:CLS
40:S=0:W=0:P=0:G=
   50
50:A=1:Z=((RND 6-
   1)/2+1)*10
60:WAIT M:P=P+1
70:FOR C=155TO 6
   STEP -Z
80:GCURSOR 0:
   GPRINT "08080C
   0C1E1E3F1E1E0C
   0C0808"
90:IF P=16THEN 27
   0
100:GCURSOR ABS (C
   -1):GPRINT A;A
110:IF INKEY$ ="+"
   AND S<10THEN 1
   80
120:GCURSOR ABS (C
   -1):GPRINT 0;0
130:A=A*2:IF A=128
   THEN 50
140:NEXT C
150:WAIT 50:
   GCURSOR 3:
   GPRINT "412241
   08412241"
160:CLS
170:GOTO 250
180:CALL &47C0
190:CALL &47D7,G
200:CLS :S=S+1:IF
   A=8THEN 220
210:GOTO 80
220:GCURSOR 0:
   GPRINT "08080C
   0C1E1E3F1E1E0C
   0C0808"
230:WAIT 50:
   GCURSOR C:
   GPRINT "412214
   08142241"
240:CLS :W=W+1:
   GOTO 50
250:WAIT 100:PRINT
   "YOU HIT ";W;"
    METEORS."
260:CLS :GOTO 30
270:CLS :WAIT 0
280:FOR X=1TO 155
290:GCURSOR X:
   GPRINT "000008
   080C0C1E1E3F1E
   1E0C0C0808"
300:NEXT X
310:WAIT 50:PRINT
   "YOU MADE IT!"
   :GOTO 30
STATUS 1          829
```

FROM THE HIP POCKET

Our readers write to share information with others.

Time

The Sharp PC-1500/Radio Shack PC-2 TIME function is a good calendar clock. But it may take a moment or two to interpret a display like 70403.0017 at 3 AM or when there are distractions. One remedy is to use the Instruction Manual's clock simulation program. But it is a nuisance to type this program back in each time after one has deleted a long program using NEW.

A better remedy is also an interesting example of the sort of small program that can be executed via Reserve keys. Program any two adjacent Reserve keys as follows:

```
Left: Z=(INT(TIME*100+.7)-INT(TIME/100)*1E4)/
      100
Right: PRINTUSING"###.##";"Date =";INT(TIME/
      100)/100;" Time =";Z-12*INT(Z/13)
```

(The use of .7 instead of .5 in the first INT expression is not a misprint. With the one exception that is mentioned below, it rounds base-60 minutes and seconds properly for a base-10 display.)

Pressing LEFT alone gives the time, but a time such as 2:10 PM will appear as 14.1. Pressing LEFT then RIGHT gives a properly formatted date and modulus-12 time. For instance, at 2:10 PM on July 4th the display would read:

Date = 7.04 Time = 2.10

There is one minor bug. Whenever the time is 30 or more seconds past 59 minutes after an hour, screen output will display that hour and 60 minutes instead of the next hour (3.60, for instance, instead of 4.00). Unfortunately, the programming also uses up most of the reserve area memory, so that eliminating this bug requires simplifying the display. Use three Reserve keys:

```
Left: (As above)
Next: Z=Z+.4*(Z-INTZ=.6)
Right: PRINTUSING"###.##";"D";INT(TIME/100)/
      100;" T";Z-12*INT(Z/13)
```

I prefer the first program.

Arthur L. Thomas
University of Kansas
307 Summerfield Hall
Lawrence, KS 66045

I/O Ports

I have found the handshake signals of the CE-158 serial interface make convenient one-bit input and output ports for the Sharp PC-1500.

The example provided here uses the Request-To-Send (RTS) signal from pin 4 on the RS-232

connector. This pin is connected to a diode and a 2.2K resistor and then to a solid state relay as shown in the accompanying diagram. In this way the low voltage output of the PC-1500 can be used to control a 110 V.A.C. device. The BASIC command OUTSTAT 0 will turn the relay on and the command OUTSTAT 2 will turn it off. Or, equivalently, these values may be POKEd at hexadecimal location D00E in the alternate memory of the PC. Machine language commands may also be used to control the relay.

As indicated in the diagram, the output from pin 4 on the RS-232 connector is used to control the solid state relay. A suggested relay is the Crydom Model D2402 (priced at about $16.00 at Newark Electronics). This relay will control A.C. currents up to 2.5 amperes. With the arrangement shown, about 1.6 milliamperes is drawn from the CE-158 interface when the potential difference at the D.C. input to the relay is about 3.6 volts. This current is well below the limiting value of the SN75188 driver chip in the interface, which is approximately 10 milliamperes.

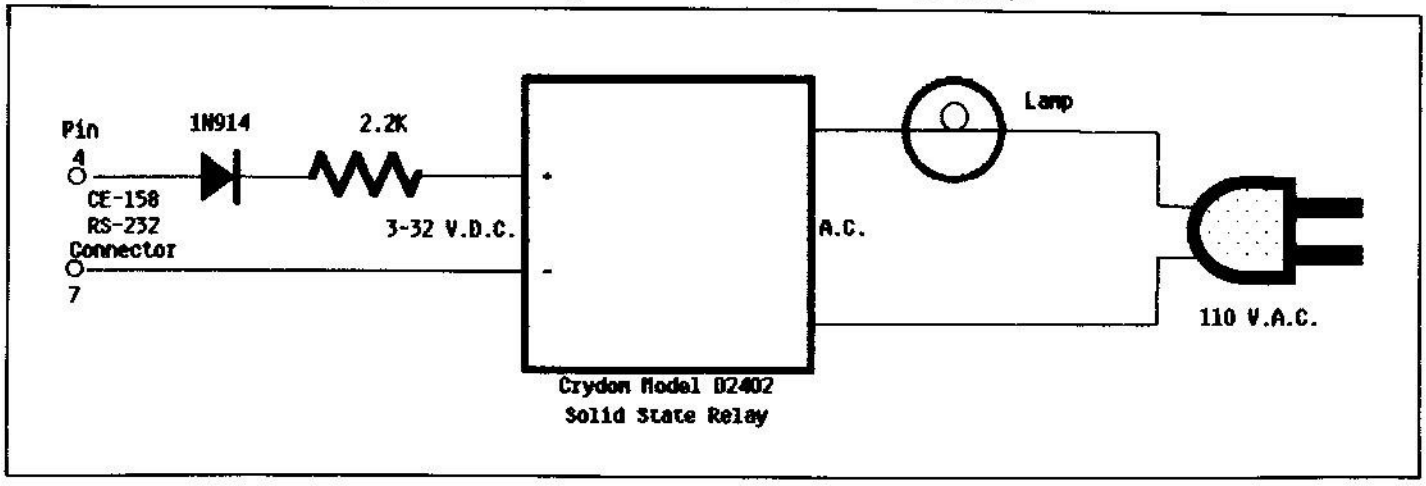
The BASIC program is a simple example illustrating that the latched port can be used to turn on a 110 V.A.C. light after an elapsed time. The first three lines of the program prompt the user for the hour, minute and second. The program then goes into a loop and will turn on the light at the requested time. Of course, a more sophisticated type of control program could be written, but this illustrates the method.

```
10 : INPUT "HOUR=?";H
20 : INPUT "MINUTE=?";M
30 : INPUT "SECOND=?";S
40 : CLS
50 : C=10000*H+100*M+S
60 : A=TIME/100
70 : B=(A-INT(A))*1000000
80 : IF B=C THEN OUTSTAT 0:STOP
90 : GOTO 60
100 : END
```

I have also used this general scheme to run a stepper motor. Similarly, other signals can be input or output to connect the PC with the outside world.

William G. Delinger
Northern Arizona University
Department of Physics
Box 6010
Flagstaff, AZ 86011

Diagram *Solid State Relay, Controlled by RS-232C Port, Activates Lamp.*



ALTERNATE CONTROL OF THE PC-1500 LCD

The liquid-crystal display (LCD) on the Sharp PC-1500 and Radio Shack PC-2 consists of 156 columns by 7 rows for a total of 1092 points or pixels. If you have studied the Instruction Manual, then you know that one way to access individual pixels is to use the GCURSOR and GPRINT statements. The GCURSOR directive will select the starting "column" while GPRINTprovides for the turning on (or off) of selected dots within a column.

Selecting a particular column is not difficult because the GCURSOR parameter allows this to be directly specified: GCURSOR 10 directs the PC to start displaying in column 10 (with the first column being referred to as column zero).

Selecting a particular row within a column can be somewhat more complicated. Rows are numbered from top to bottom, starting with row zero. Rows are assigned values according to a binary positioning scheme. The top row (row 0) is activated with a value of 1. The next row (row 1)

needs a value of 2, the next (row 2) requires the value 4, and so forth. This continues down to the seventh row (labeled row 6) which is activated by a value of 128. Note that this is 2 raised to the 7th power. Dealing with this kind of scheme can become complicated for anyone, never mind just beginners.

For instance, what do you do if you want to, say, turn on a single pixel *without disturbing any of the other points* within a column? You cannot just blindly invoke the GPRINT statement (even if you have figured out how to easily turn on the desired row position), for to do so will knock out whatever else was already activated in that column.

Well, there is also a POINT statement in the PC's repertoire. It can be used to find out what points within a column are activated at any given time. So, you use it. What do you get back? Why a value. Properly decoded it will tell you what points are turned on. Now what? You have to do something that will preserve those points while also turning on the pixel that you want to put up at the present time. Have you figured out a way to do it?

```
5900 "GR"WAIT 0
5910 E=POINT X:D=
     (2^Y):D=DOR
     E
5920 GCURSOR X
5930 GPRINT D
5940 RETURN
```

The variables X and Y in the above subroutine are the LCD matrix points (column number going from left to right starting at zero, row number going from top to bottom, starting at zero). Variable E is used to temporarily store the current contents of column X as retrieved by the POINT statement. The little trick of raising 2 to the power of Y (where Y is the desired row in the range 0 to 6) yields a value that turns on a single pixel within a column. This value, representing a single point, is retained in variable D and then logic-ORed with the contents of E. The result is whatever was already in the column with pixel Y also being on (if it wasn't already). The GPRINT D directive throws the whole thing back up on the screen.

Thank you, but I don't feel that little routine is non-trivial or very obvious to those that have not been around programming for awhile. The gist of what I am saying is that using LCD graphics in many types of applications can hardly be called a snap on the PC-1500.

Ah well, one can always learn, right? That is what this little spiel is about. Just to make sure you are with me, you can couple the little subroutine just presented with the following series of directives and do some experimenting.

```
100 INPUT "X-COORD
    : ";X
110 INPUT "Y-COORD
    : ";Y
120 CLS:GOSUB "GR"
130 A$=INKEY$:IF
    A$=""THEN 130
140 GOTO 100
```

With the PC in the RUN mode, you may enter X coordinates (in the range 0 - 155) and Y coordinates (0 - 6) in response to the prompts. You will then see a single dot appear on the screen at the position selected. Play around for awhile until you become a believer and are in sync with the coordinate grid being used. (The X axis goes the same old way. The Y axis has increasing values going down, which may perturb you a bit if you are not used to dealing with computer graphic displays. Never mind, you will get used to it!)

Exciting, huh. You can put a whole dot at any position you want on the display....

But, A String of Dots....

Makes a *line!* So, when you feel up to it, add in the next couple of routines to your growing library:

```
200 INPUT "X1-COOR
    D: ";R
210 INPUT "Y1-COOR
    D: ";U
220 INPUT "X2-COOR
    D: ";S
230 INPUT "Y2-COOR
    D: ";V
240 CLS :GOSUB "LI
    NE"
250 A$=INKEY$ :IF
    A$=""THEN 250
260 GOTO 200
5000 "LINE"IF S>R
    LET P=1
5010 IF S<RLET P=
    -1
5020 IF S=RTHEN 5
    070
5030 FOR M=RTO S
    STEP P
5040 Y=INT ((((V-
    U)/(S-R))*(M
    -R))+.5)+U
5050 X=M:GOSUB "G
    R"
5060 NEXT M
5070 IF V>ULET O=
    1
5080 IF V<ULET O=
    -1
5090 IF V=UTHEN 5
    140
5100 FOR N=UTO V
    STEP O
```

```
5110 X=INT (((N-U
     )*(S-R)/(V-U
     ))+.5)+R
5120 Y=N:GOSUB "G
     R"
5130 NEXT N
5140 RETURN
```

Note that the variable at the end of lines 5070 and 5080 (and the STEP value in 5100) has the name "O" (as in "Oh, my goodness!"). Don't go crazy trying to redefine the value zero.

When you have those routines installed, you can use the command RUN 200 to give your new capabilities a whirl. Respond to the queries for X1,Y1 and X2,Y2 coordinates as the beginning and ending points of a line on the LCD. Press ENTER after inputting Y2 and you should see your little PC laboriously draw a line between the points you specified. I say laboriously, because the process is rather slow. Never-the-less, chances are you haven't seen the likes of that before on your PC. If you do some experimenting, you will discover that the line can go from left to right or right to left or down to up and vice-versa or straight across, etc. You can now draw any straight line you want, just give the start and end coordinates. (Yeh, I know you could do that on the printer, but had you done it on your LCD?)

When you get tired of inputting coordinates to see your little machine draw lines, you can create your own line doodler with these lines of code:

```
300  CLS
310  R=RND (155)+1:
     S=RND (155)+1:
     U=RND (6)+1:V=
     RND (6)+1
320  GOSUB "LINE"
330  GOTO 310
```

Executing RUN 300 will send the PC off on the task of filling up its screen with lines. Remember, this routine is only turning points on, not off, so any time it intersects with a previously drawn line you won't see anything happen. After awhile you won't see much of anything going on because the screen will be filled with overlapping lines. Of course, then you can press BREAK and start the process all over again.

When you have had enough of that, you can get to work thinking of some practical applications for the line drawing routine. If, that is, you can tolerate the snail's pace at which it operates. (Of course, sometimes slow is a lot better than nothing at all!)

You Can Stop Here

If you don't want to learn any more about how the LCD really operates. The going starts to get a little rougher....

While what has been presented certainly works, it sort of lacks something -- mainly speed! If we are going to go on to see if this problem can be surmounted, then it will be necessary to get right down to the real nitty-gritty of how the LCD is controlled on a pixel-by-pixel basis. (That's right, we are heading toward machine language routines. If that turns you off, you should have stopped when given the opportunity!)

The liquid-crystal display is always controlled by the contents of a particular block of RAM. Precisely, the locations from &7600 - &764D and &7700 - &774D. These are the locations that control the individual pixels. (Two other memory locations, &764E and &764F control the LCD legends. Those are the little indicators that say RUN, BUSY, DEG, etc. This discussion will not be concerned with the control of those signs.)

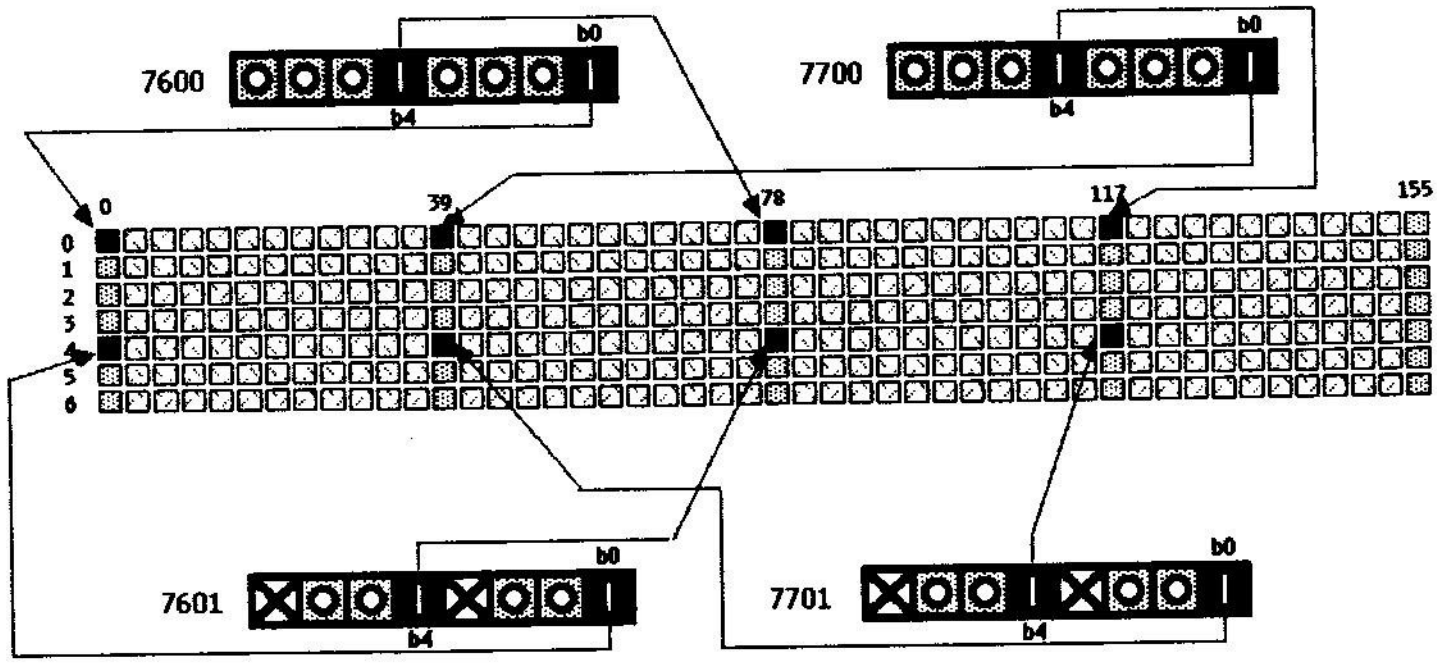
Alas, the correlation between the controlling locations in memory and the points on the display is not exactly what one would call a straightforward transformation! A more likely terminology would be piece-wise linear -- with a lot of pieces!

Now it starts out easy enough. Using the coordinate system previously defined for pixels in the LCD, location 0,0 is controlled by the contents of bit 0 (the least significant bit) in the byte of memory that resides at the address &7600.

It doesn't stay neat for very long. Bits 1, 2 and 3 of that memory location do control pixels in rows 1, 2 and 3 of column zero on the display. But, does bit 4 control row 4? Noooo! What is controlled by bit 4 at address &7600? Would you believe *row 0 in column 78?* Believe it. And where on earth is the memory location that controls cell 4 in column 0? How about the zero bit in the next memory address at &7601? Well, that's the way it works.

The scheme is somewhat illustrated by the accompanying diagram. There is a pattern to it, but it will seem pretty obtuse to those uninitiated in the craft of computer design. The gist of the layout is: the lower-half bytes of even-valued memory addresses in the range &7600 - &764C control the pixels in the zone X=0 to X=38 and Y=0 to Y=3, the lower-half even-valued addresses from &7700 - &774C control pixels in the zone X=39 to X=77 and Y=0 to Y=3, the upper-half bytes of even-valued addresses &7600 through &764C dictate the status of pixels in the range X=78 to X=116 and Y=0 to Y=3, the upper-half bytes of even-valued locations &7700 - &774C relate to the LCD cells in the range X=117 through X=155 with Y=0 to Y=3, the three lowest-ordered bits of the bytes having odd addresses in the range &7601 - &764D control row points 4 to 6 (that is, Y=4 to Y=6) in the range X=0 to X=38 (note that bit position b3 is not used in the odd-valued address bytes), similarly the lower

Diagram *Memory Locations That Control Individual LCD Pixels.*



three bits of the odd addresses in the range &7701 - &774D control pixels in the range X=39 to X=77 with Y=4 to Y=6, while the upper-half of bytes (with bit b7 unused) in the odd-valued addresses in the range &7601 - &764D evoke responses in the range X=78 to X=116 and Y=4 to Y=6, and similarly the upper-half bytes (b7 not used) of odd addresses from &7701 through &774D control pixels having the coordinates X=117 to X=155 and Y=4 to Y=6.

Whew. Behold the diagram for clarification. (Oh yes, the diagram is compressed horizontally, so that not every column is shown. OK?) The following routine can be used to verify our understanding of these matters:

```
1000 "ON"A=&7600:
     B=256
1010 "COL"F=0:IF
     X<39LET C=A+
     X*2:GOTO "RO
     W"
1020 IF X<78LET X
     =X-39:C=A+B+
     X*2:GOTO "RO
     W"
1030 F=1:IF X<117
     LET X=X-78:C
     =A+X*2:GOTO
     "ROW"
1040 X=X-117:C=A+
     B+X*2
1050 "ROW"IF Y<4
     GOTO "FLG"
1060 C=C+1:Y=Y-4
1070 "FLG"IF F=1
     LET Y=Y+4
1080 E=PEEK C:D=(
     2^Y):D=DOR E
1090 POKE C,D
1100 RETURN
```

All this routine really does is give us another way of controlling the individual pixels of the LCD. You can substitute it for the first subroutine (labeled "GR") that was presented near the start of this discussion. Just change the references to "GR" in lines 120, 5050 and 5120 to "ON" to incorporate this new subroutine. Try it out using the entry routines (RUN 100, RUN 200 or RUN 300).

You may not be all that thrilled at this point. The speed of this routine seems to be about the same as the GCURSOR/GPRINT method. What has been gained? Little more than understanding, hopefully, at this juncture. But, at least now we have a method of dealing with those little pixels *that correlates with locations in memory.*

That means we are in a position to start working directly in machine language if it appears that it might do us some good. If speed is a criteria, then there is no doubt that MLP (machine language programming) could serve us in good stead. How would you like to see any line you wanted drawn on the LCD in a fraction of a second?

That shall be the goal of the next installment. In the mean time, perhaps you will want to try your hand at converting the above "ON" subroutine to machine coded directives. Challenging?

FROM THE WATCH POCKET

Hope you like our new 16 page format. It will enable us to broaden our scope somewhat. We plan to keep a closer eye on some of the PC models that seem go be gaining in popularity, such as the new Casio models. And, in response to popular requests, we will be tracking the introduction of some of the compact notebook and briefcase computers that offer substantial capability.

Speaking of the latter, I am most impressed with the operating system that is being touted for the Gavilan. This type of system, particularly when implemented in a manner such as theirs, makes it very easy for anyone to use the machine. One of the biggest drawbacks to dealing with a computer on a regular basis is having to organize all the data. By that I mean having to remember which files are on what diskette, what data is in which files, what files relate to one another, and so forth.

Up until recently, the user was forced to do a good portion of this organizing mentally. You had to devise a file naming system that would (you hoped) help you recall what you were dealing with as you used the computer. For instance, you might name a word processing file "Jones:ltr." If you also had a spreadsheet document associated with the information you provided in the Jones letter, you might call it "Jones:calc." A third file for your data base program might be named "Jones:info."

The problem comes in when the number of such documents you may be dealing with mounts into the hundreds. It soon becomes a chore just turning the computer on and mentally struggling to recall all the file relationships, names, etc. Sure, menu and catalog systems help. But, unless the software is fully integrated, you can't even tell, for instance, that the Jones letter refers to another spreadsheet file and/or entries in another data base.

A fully integrated system such as the Gavilan greatly relieves the mental strain associated with using a computer. Pictorials of files, folders and documents, are arranged so that they intrinsically show you the organization of a project. At a glance you can see that a "folder" contains a letter, a spreadsheet and a list. Then you need simply "point" to the image of the folder using the touch control panel to tell the computer that you want to work with its contents. Another touch (pointing to, say, a "letter" document that is within a folder) and you are ready to modify the document.

There can be little argument that merely pointing a finger and tapping a screen to make a selection from a graphically organized menu sure beats having to recall and type something such as: CATALOG, LOAD JONES.LTR, EDIT. Repeated tens and hundreds of times during a working day, the latter method becomes a joke (and even a cruel one) in comparison.

A Plug

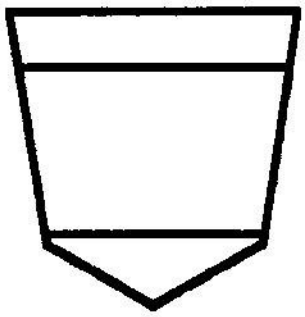
If you are interested in keeping up with the "desktop" computer scene without having to read a dozen magazines a month, I can recommend the **Jeffries Report,** *Box 6838, Santa Barbara, CA 93111,* phone (805) 967-7167. This 8-page monthly newsletter is tightly written by Ron Jeffries (who also bylines a number of magazine columns). He does a good job and makes interesting reading. It is one of the better industry newsletters I have seen over the years. Current subscriptions are $30.00 for 12 issues. No kickback involved here, just genuine respect.

Thank You!

As many of you know, *PCN* periodically surveys a randomly-selected sample of its readership. The results of these polls are used to provide guidance as to the type of material you wish to see. To those of you who have responded in the past and to those who may be asked to do so in the future, I wish to extend a sincere "Thanks!".

By the way, if you have material you believe might be of interest to other *PCN* readers, write for a copy of our *Author's Guidelines.* It provides a brief outline of our article review policy, desired manuscript format and payment rates.

POCKET
COMPUTER
NEWSLETTER



Issue 29 - November

PORTABLE COMPUTER TRAVELS LIGHT

MicroOffice Systems Technology has announced a portable computer designed for travelers. The five pound unit can serve in the office and on the road as a remote terminal or stand-alone personal computer.

The machine has been given the trademarked name "RoadRunner." It weighs just five pounds, operates from self-contained rechargeable batteries, and is notebook-sized. It measures a trim 7-3/4 by 11-1/2 by 3 inches. At this size it fits in one-half of a standard business briefcase. The other half of the briefcase can be used for the invariable array of documents that typically accompany the harried traveler. The company believes the RoadRunner is ideal for use by writers, salespeople and others who need a small, yet powerful, portable office work station.

Using appropriate software modules, the unit permits users to create account files, write letters and reports, analyze financial models, track expenses, follow orders and proposals, and access remote databases, if desired.

The mobile unit features a standard-sized 73-key keyboard in the usual QWERTY format. There are 18 functions keys. Eight of these are used to select operations from a menu that can be shown on the system's built-in LCD screen.

In fact, the 8-line by 80-column LCD screen is built as part of the unit's cover. The cover flips up to automatically turn the computer on and provide a convenient viewing angle of the screen. When this cover is closed, the system's keyboard and plug-in software modules are fully protected from accidental damage in a traveling environment.

A series of plug-in modules provide a wide range of capabilities. Modules may contain up to 32 kilobytes of RAM or 128 kilobytes of ROM. The modules are said to provide advantages over the use of sensitive disk drives in portable situations.

The unit utilizes a RS-232C interface for communicating with external devices. An internal 300 baud auto-dial, auto-answer modem may be used for communicating with other terminals or for utilizing remote computers and databases.

The RoadRunner can be operated for more than eight hours before its battery pack is exhausted.

Software packages to be made available for the machine include: a text editor, spreadsheet, BASIC, phone list, appointment manager and terminal communications package. The company also plans to provide a variety of CP/M-compatible programs through its plug-in modules.

The single-quantity price for RoadRunner ranges from $1,600 - $2,000 depending on options selected. Productions units are scheduled for delivery in late November, 1983.

The RoadRunner has been under development since the company's founding in late 1981. The firm, MicroOffice Systems Technology, Inc., was founded as an organization dedicated to the development, manufacture and marketing of advanced portable information products.

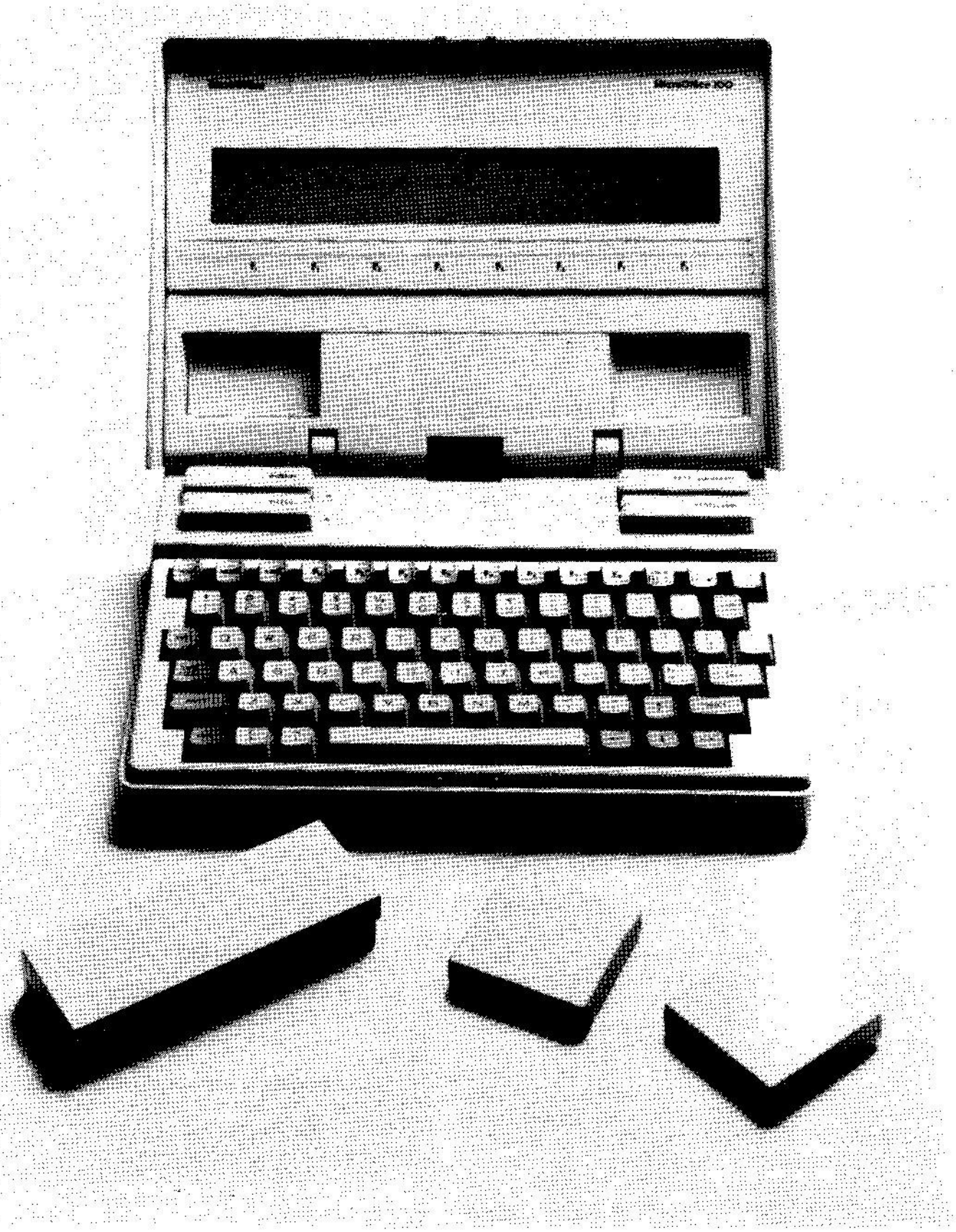
The company is headed by Robert F. Weltzein who serves as Chairman of the Board and CEO. He served formerly as President of Citizen Watch. Prior to that, he was Chairman and CEO of Timex Corporation.

The firm's president is James P. Dunn. Mr. Dunn was formerly with Exxon Enterprises, Hendrix Electronics and IBM.

John W. Chapman is Secretary and Treasurer. Mr. Chapman is also Vice President and General Counsel for the Computer and Communications Industry Association. Previously he has served on the corporate staff of Xerox, Honeywell Information Systems and California Computer Products.

For additonal information on the RoadRunner portable computer, contact: *Marvin Preston, Director of Marketing, MicroOffice Systems Technology, Inc., 35 Kings Highway East, Fairfield, CT 06430.* Telephone (203) 367-2525.

Photo *The RoadRunner portable computer is shown with its removable battery pack and several of its plug-in modules. Up to four modules may be installed at one time. The flip-up cover contains the display and provides protection for the keyboard when closed.*



THE CASIO FX-700P

At first feel, the FX-700P lies somewhere between a Sharp PC-1250 and the earlier Radio Shack PC-1 in size. That is, it is certainly bigger than the 1250, but slimmer, shorter and not as "chunky" feeling as the PC-1. It does, indeed, slip into a shirt pocket and is light enough (at just 4.2 ounces) so that one feels comfortable with it there.

There is more room for the keys on the Casio 700P than there is on the Sharp PC-1250. They are larger and spaced further apart. Thus, keying in programs is not as much of a chore as it sometimes feels when keying in a large program on the tiny keys of the 1250.

I would say that the 700P might have special appeal to engineers and scientists. For one thing, it sports a good selection of mathematical functions. These include: SIN, COS, TAN, ASN, ACS, ATN, LOG, LN, EXP, SQR, ABS, SGN, INT, FRAC and RAN#. All of these are available at the stroke of two keys: a special (F) function key and the key that is sublabeled with the corresponding legend. For another, most of the commonly used BASIC statements are also available for two keystrokes. The keywords GOSUB, FOR, TO, STEP, NEXT, GOTO, IF, THEN, PRINT, RETURN, STOP, END, DEFM and INPUT, as well as the commands LIST and RUN, are easily brought to the display. Just press the special shift (S) key and the appropriately superlabeled key. This feature can be particularly nice if you are not accustomed to doing a lot of typing on a QWERTY keyboard, which is the layout used on this machine.

Another factor in favor of the FX-700P if you plan to do a fair amount of number crunching, is its relative speed. Based on a simple loop test, it is more than twice as fast as its predecessor (the FX-702P). In another comparison using a simple benchmark that loops, calls a subroutine and performs standard mathematical operations, the little FX-700P beat the PC-1250 by 2:1 and was virtually tied with the PC-1500. Quite impressive.

At first I was a little dubious about the practicality of having just a 12-character LCD display. Experience has shown that, at least in this implementation, it is acceptable. Numbers may be displayed in 10-digt signed mantissa format or an 8-digit mantissa with 2-digit exponent. One reason Casio can get away with such a short display is the wise manner in which they implemented its operation. When displaying a long message, such as via a PRINT statement, the screen automatically scrolls at a comfortable rate.

The little LCD can depict a full complement of alphanumeric characters. In addition to uppercase representations, a special EXTension mode allows lowercase characters and even a variety of special mathematical symbols and a few graphics (such as representations of card suits: the spade, heart, diamond and club symbols) to be displayed.

The 700P seems especially well-suited for people who need to repetitively perform a number of relatively limited procedures. You can store up to ten different programs in its memory at one time. Each program area (designated P0 - P9) is accessed independently (even though all programs are sharing the same block of user memory). Thus, you can re-use the same group of line numbers in each program, if desired.

It is as if you had 10 separate little PCs built into one keyboard/display package. Each program area can be edited or executed at will. You can allocate the amount of memory assigned to each program from amongst the total of 1568 program steps (or 222 variables). If you divided this into equal portions , each program area could contain about 156 program steps (bytes). Put your ten most frequently used programs in memory and solve at will!

The BASIC interpreter is similar in overall capabilities to most other PCs. There are some special features that deal with the partitioning capability of the machine. For instance, the use of the statement GOSUB #1 enables a user to call a complete program module as a subroutine. While the machine does not provide for the use of labels on program lines, GOTO and GOSUB statements can use variable values to represent line numbers.

As might be expected on a machine having only a 12-character display, string capabilities are a bit limited: about on a par with a Radio Shack PC-1. Except for one special 30-character ($) string storage area, standard string variables are limited to just seven characters in length. However, the Casio FX-700P can test for greater than/less than when processing string variables. The PC-1 could only check for equality.

The 700P does not have any audio capability. If you are going to let the PC crank away on projects that take a lot of time, you will have to periodically check the display for results. You cannot generate a cheery little beep to alert you when the task has been completed.

Still, all-in-all, it is a nice, slim, comfortable, light-weight, fully-functional PC. At a U.S. list price of $99.95, it is competitive with other brands.

Programs can be saved to or restored from an audio cassette recorder using an optional recorder interface dubbed the FA-3. This item has a U.S. list price of $39.95. An optional thermal printer, the FP-12, at $69.95, provides capability for obtaining hardcopy of programs and data.

FUEL CONSUMPTION ANALYST

R. M. Organ, 9501 Nowell Drive, Bethesda, MD 20817, who describes himself as an "absolute beginner" to the world of programming, has come up with this analyzer for the fuel-conscious motorist. You can use it to generate a chart that can reveal the influence on fuel consumption caused by such factors as highway versus city driving, winter versus summer, one driver versus another, the consequences of mechanical defects such as brakes that rub or fuel swilling out of a loose cap on the gas tank, etc.

You use the program by transferring data from a logbook. This book should record the number of gallons put in and the odometer mileage at each fill-up. As each pair of entries is inputted, the plotter unit draws a bar representing the miles-per-gallon achieved from the previous tankful. Successive entries result in more bars. A graph builds up as paper rolls through. The program automatically extends divisions on the MPG scale as needed and periodically repeats the MPG scale. You can build a chart that covers 100,000 miles or more, if you don't mind two feet (plus) of paper! When all of the desired data has been fed in, the program does still more: it prints a table of frequencies, listing the number of occurrences of each integer MPG, and plots a histogram. This makes it easy to spot which MPG occurs most often or determine whether there are several peaks. Those might represent the results of two drivers, different seasons of the year, etc.? The user is also given the option of having a summary of overall average MPG calculated for the entire period covered by the chart.

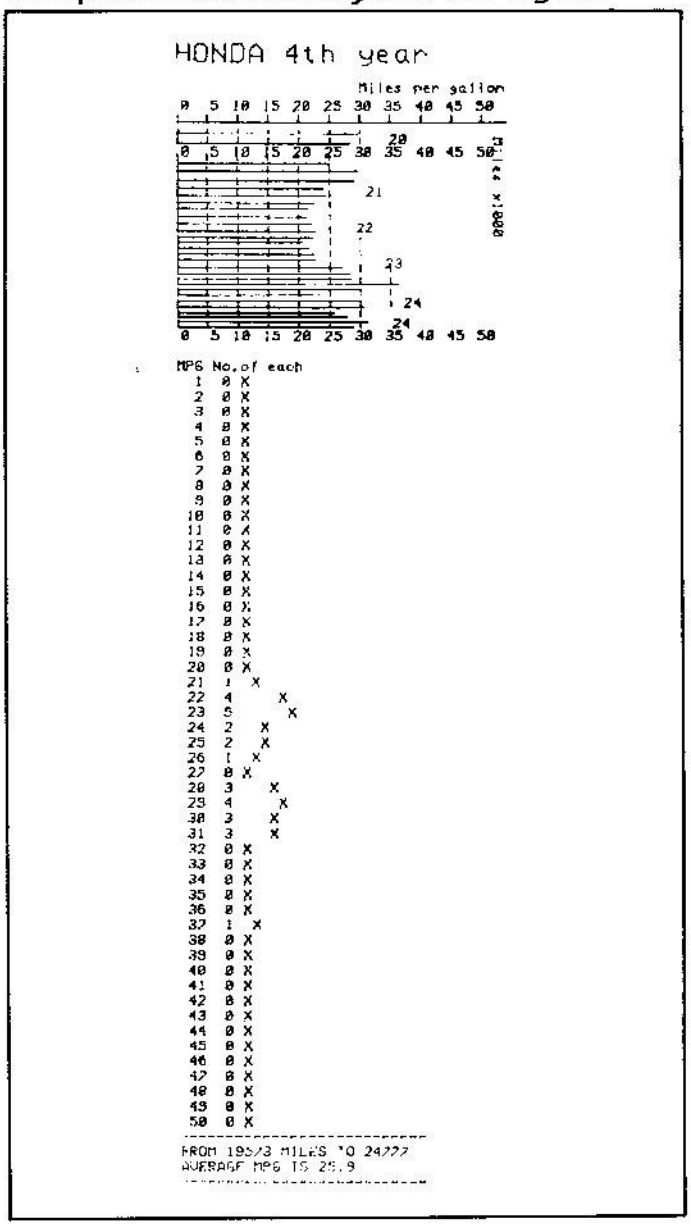
Overview of the Program

The program will execute in an unexpanded Radio Shack PC-2 or Sharp PC-1500. It does a lot of work for the user. After the user has entered the EPA estimated MPG, the program selects its own scale for the MPG axis. It averages the MPG from successive entries, thereby taking care of the times when the tank was underfilled/overfilled, which could produce an unsatisfactory appearance on the chart. It shows the MPG as closely as the plotter allows. This is one two-hundredth of the maximum MPG value shown on the scale. For a 30 MPG scale, it would be shown to within 0.15 MPG. (Of course, calculations maintain a higher degree of accuracy.) The histogram value is rounded to the *nearest* whole number, instead of always being rounded downwards. The MPG scale will accept any maximum value in the range 10 to 100 in tens. The program stops if any figure over 99 (MPG) is produced. This is an unlikely event. The program also has other safeguards.

Tips On Inputting Data

Make sure the entries in your log are legible, do not have gross errors, and are in proper sequence *before* you commence inputting to the computer. It is a good idea to lay a ruler or piece of paper beneath the entry line you are reading to help keep track of your place and coordinate gallon and mileage data pairs. Check the LCD after each entry to verify proper keying *before* you press the ENTER key. Wait for the computer to generate its beeps before starting to input the next data item. The computer and printer sometimes have quite a bit to do between entries. If a premature entry is attempted, the program may halt. It is also advisable to cover up the arithmetic operator keys (/ to +) or take great care not to inadvertently press those buttons while inputting numbers. Failure to

Example *Chart Produced by the Fuel Program.*



heed this advice may bring the operation of the program to an untimely halt.

Program Description

The program runs in several successive phases as indicated by the accompanying flowchart. The first phase accepts the name of a car plus notes to a total of 16 characters.

Line 710 zeros some important variables and prompts for the maximum MPG expected. Lines 720 and 730 round this answer up to the next largest decade as the maximum scale value. Line 740 calculates distances for the pen to move and the numbers to be printed on the axis. Lines 750 to 780 draw and print. Line 790 is used to title the mileage axis (which will develop as the program proceeds). Line 795 dimensions an array (that will be used by the histogram later) and fills it with zeros.

Line 800 accepts the first pair of entries. One of these is remembered for use in the final summary. The other is set to zero since it will not be needed in the first calculation but it is convenient to establish the habit of keying in pairs. Line 805 foreshadows serious operations: O, M and N are variables to contain mileage. N will always hold the most recently entered, M the earlier and O the one before. Thus, for every new entry its two predecessors must have been moved back one along the series. Later, in line 830, the existence of O will enable the program to average the mileage from two fills.

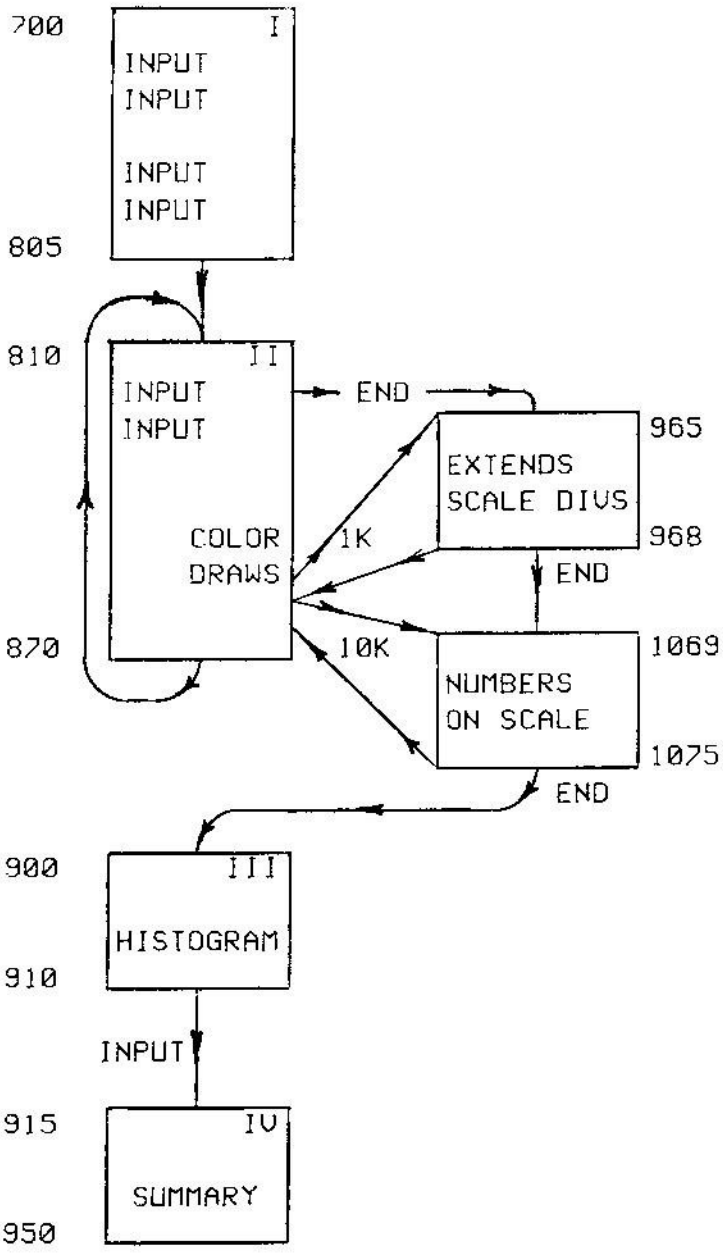
J is a variable paired with I which contains the number of ten-thousands of miles. At the end of the second phase J will become I and S will become R. However, they are each calculated here in order to avoid having extra scales printed immediately next to the axis, as would otherwise happen if the second mileage to be entered introduced a new thousand or ten-thousand. This line, 805, ends the first phase of the program, which is used only once.

The next phase is repeated for every pair of entries. Its objectives are: to draw a bar of the correct length, to test whether the newest entry has changed its number into the next thousand or ten thousand, to begin an ending-off operation if the prompt to "end" has been responded to affirmatively, or to return control back to its beginning in order to repeat the loop.

Line 810 accepts the next entry for mileage or an indication to end. Line 820 accepts an entry for gallons and adds it to an accumulating total. Lines 830 and 840 make a number of calculations: notably average MPG, the distance the printer pen must move for this and for miles run, the number of thousands and ten thousands in the new mileage, of MPG as an integer, and pen color.

Color was changed each thousand miles as a way of avoiding construction of the usual miles axis. This would have taken up much needed space at the edge of the narrow paper tape. Considerable thought was needed to find a formula that would convert successive thousands into a repeating sequence of 0 to 3. In line 850 that particular MPG in the array that corresponds with the MPG just calculated is increased in number by one. The printer is prepared to draw the bar and the actual printing takes place in line 860. Lines 863 and 865 test whether the mileage has increased into a new thousand or ten-thousand. If so, direct control is passed to the appropriate subroutines starting in lines 965 and 1069.

Flowchart *Operation of the Fuel Consumption Program.*



The final line of this phase is number 870. It moves numbers back along their storage sequences, leaving variables free to accept new entries, then loops back to the start of this phase.

Line 900 starts a third phase which constructs a histogram as the first part of an end-off sequence. In the text mode, two columns are headed, one for the name of MPG and the other for how many of each MPG occur. The baseline for the histogram is defined as the seventh column from the edge of the printout. Lines 905 and 910 print the data stored in the array into the selected format to complete phase three.

Line 915 begins with a prompt for the second part of the end-off, which is phase four. Intellectually, this is the least satisfying part of the program, as it ought to continue automatically. Every attempt to cause this failed because various PRINT USING statements are included in the earlier phases for good reasons. These are not appropriate to the format used in phase four. The only way that this programmer could find to eliminate memory of the earlier PRINT USINGs was to include an instruction to RUN. Apparently this clears all variables except those designated by letters, which, by great fortune, are those used here.

[The use of a USING statement without any parameters will return the PC to its normal "unformatted" state of operation. - Editor.]

Line 920 formats the calculated overall MPG into a single decimal place. It was invented to avoid use of the PRINT USING statement, which would have clouded the running of lines 930 and 935. These define strings and then print them with numbers interpolated. It is the fact that these numbers can vary in length from two-digits to six-digits, dependent on the miles entered, that causes

Program *Charting Fuel Consumption.*

```
700:"FUELUSE":WAIT
    130:PRINT "FUE
    LUSE by R.M.OR
    GAN":CLEAR
705:INPUT "NAME OF
     CAR? ";E$:
    CSIZE 2:LPRINT
    E$
710:TEXT :CSIZE 1:
    LCURSOR 20:
    LPRINT "Miles
    per gallon":
    LET B=0, D=0, I=
    0, J=0, N=0, Q=0,
    S=0
715:LET W=0, Z=0:LF
    1:INPUT "EPA M
    PG (Hwy) is ";
    E
720:IF (E/10-INT (
    E/10))=0THEN
    GOTO 740
730:IF (E/10-INT (
    E/10))<>0THEN
    LET E=E+1:GOTO
    720
740:GRAPH :T=20:
    FOR X=0TO 200
    STEP T:Z=X/200
    *E
750:LINE -(X,0)-(X
    ,5),0,0
760:GLCURSOR ((X-1
    5),8):CSIZE 1:
    LPRINT USING "
    ####";Z:
    GLCURSOR (X,0)
770:NEXT X
780:LINE -(220,0),
    0,0
790:ROTATE 1:
    GLCURSOR (207,
    -10):LPRINT "M
    iles x1000":
    GLCURSOR (0,-5
    ):ROTATE 0
795:DIM A(E):FOR H
    =0TO E:LET A(H
    )=0:NEXT H
800:TEXT :BEEP 1,8
    ,50:INPUT "MIL
    EAGE=";M:BEEP
    2,8,50:INPUT "
    GALLONS=";K:
    LET B=M,K=0
805:LET O=M:LET J=
    INT (M/10000):
    LET S=INT (M/1
    000)
810:BEEP 1,8,50.
    INPUT "NEXT MI
    LEAGE(1 for EN
    D)=";N:IF N=1
    THEN 965
820:BEEP 3,8,50:
    INPUT "NEXT GA
    LLONS=";L:LET
    W=W+L
830:LET U=(N-O):
    LET G=U/(K+L),
    X=G*200/E:LET
    Y=U/75:LET Q=Q
    +Y:LET R=INT (
    N/1000)
840:LET I=INT (N/1
    0000):LET C=
    INT (4*(N/4000
    -INT (N/4000))
    ):LET H=INT (G
    +0.5)
850:LET A(H)=A(H)+
    1:GRAPH :CSIZE
    1
860:LINE -(0,-Y)-(
    X,-Y)-(0,-Y),0
    ,C:GLCURSOR (0
    ,-Y):SORGN
863:IF S<>RGOSUB 9
    65
865:IF J<>IGOSUB 1
    069
870:LET O=M,M=N,K=
    L,J=I:IF N<>1
    THEN 810
900:TEXT :CSIZE 1:
    COLOR 0:LF 2:
    LPRINT "MPG No
    .of each":FOR
    H=1TO E:LET F=
    A(H)+7
905:LPRINT USING "
    ###";H;A(H)
910:LF -1:LCURSOR
    F:LPRINT "X":
    NEXT H
915:BEEP 5,210,100
    :WAIT :PRINT "
    Now RUN 920 or
     OFF"
920:LET Z=0.1*INT
    (10*(M-B)/W):
    COLOR 3:LPRINT
    "--------------
    --------------
    "
930:A$="FROM":B$="
     MILES TO":C$=
    "AVERAGE MPG I
    S":LPRINT A$;B
    ;B$;M
935:LPRINT C$;Z
940:LPRINT "-------
    --------------
    -------"
950:COLOR 0:
    LCURSOR 0:END
965:LET U=INT X/T:
    FOR P=0TO (U*T
    )STEP T:LINE (
    P,0)-(P,Q),3:
    NEXT P
966:GLCURSOR ((X+1
    5),0):LPRINT R
    :GLCURSOR (0,0
    ):SORGN :LET S
    =R,Q=0
967:IF N=1THEN 106
    9
968:RETURN
1069:FOR X=0TO 20
     0STEP T:Z=X/
     200*E:
     GLCURSOR ((X
     -15),-8):
     LPRINT USING
     "####";Z:
     NEXT X
1070:IF N<>1GOTO
     870
1075:IF N=1GOTO 9
     00
```

a fixed PRINT USING statement to be unsatisfactory unless one is prepared to accept unsightly variable numbers of spaces in a sentence. Line 950 brings the program to an end.

The subroutine contained in lines 965 to 968 serves to put dotted lines across the bars at intervals (T) corresponding with the intervals between divisions on the MPG axis. The lengths of these lines usually represents 1000 miles, because they are added every time the pen changes color. However, the program accumulates the actual distance since the pen last changed, in variable Q. This is because the run can be terminated at any mileage, probably less than 1000, and part of the end-off operation is to extend these dotted lines. Line 966 moves the pen to a position 15 units beyond the tip of the bar. The number of thousand miles is printed where it is usually clearly visible on a white background.

After the printing has been done, the various variables involved are either moved back along the series (S) or set to zero again (Q) ready to be used in the next series.

The subroutine in lines 1069 and 1070 repeats the printing of numbers used for the MPG axis in lines 740 to 760. However, it is done without the associated plain line. The string of numbers is printed every time the mileage changes into a new ten-thousand. An associated plain line might be confused with a MPG bar.

Use of Variables

Here is a list of the variables used in the program:

A is the title for the array of integers for MPG used in the histogram.

B is the store for the very first entry of mileage, called upon in phase four.

C is for the color of the pen.

E begins as the EPA estimate of MPG but is raised by the program to the length of the MPG scale.

F is the tab value of the X that is printed at the end of each histogram bar.

G is the MPG calculated from the most recent two pairs of entries.

H is the integer MPG calculated for use in the histogram.

I is the number of ten-thousand miles calculated for the current entry.

J is the I of the previous run through phase two.

K is the number of gallons used in the first tank filling. On later runs this becomes the stored value of the preceding filling.

L is the current number of gallons put in the tank.

M is the mileage.

N is the next mileage to be entered.

O is a store for M so that M can store N when a new N is to be entered.

P is the number of printer units between dotted lines of the scale markers.

Q is the number of printer units that have accumulated along the miles axis since the pen last changed color.

R is the newly calculated ineger number of thousand miles.

S is a store for the value of R calculated for the previous bar.

T is the number of printer steps in each division of the MPG axis.

U is the number of divisions of the MPG axis included in the length of the previous bar. The bars vary in length. In order not to clutter the chart in the area where the miles numbers will be printed, the dotted lines are only printed where needed, up to division U.

V is the mileage run between fills.

W is used to accumulate the number of gallons used since the first entry.

X is the distance in printer units that the pen moves parallel with the MPG axis.

Y is the distance in printer units that the pen moves parallel with the miles axis.

Z is a number, wherever it may be printed.

Possible Alterations

For those whoe would like to experiment, there are some improvements that could be made.

The number 75 in line 830 can be changed to 50 or 100 in order to expand or contract the miles axis.

If additional memory is available, the following lines may be added:

```
845 ON ERROR GOTO 1080
1080 WAIT:PRINT "RESTART - ADD 10 TO EPA MPG"
```

These changes will prompt action if an excessive MPG turns up and stops the run. The extra memory needed to accomodate this addition will be available already if the highest MPG does not exceed 30. However, if you should then have to take the advice and raise MPG to 40, there may be too little memory left for the array.

If your Radio Shack PC-2 accepts the statement DIM A(E)*2 in line 795, as the insert in the manual suggests it should (but mine does not), then the total memory needed will shrink considerably and should find few restrictions. The array cannot be made exactly the right size because it must accept unusually large and small MPG that happen sporadically, otherwise the program will stop.

[Thanks, R. M. Organ, for showing us how a beginning programmer tackles a less than trivial programming project. - Editor]

ALTERNATE CONTROL OF THE PC-1500 LCD

This is the second part of the discussion on this subject matter that was started in Issue 28 of *PCN*. You may want to refer to that article as I will be picking up where I left off. At that point I had presented a general line drawing routine in BASIC and a few supporting routines. These were used to demonstrate that a line could be drawn between any two points on the LCD. I suggested, at the close of the article, that those interested might find it challenging to work on converting the pixel control routine (labeled ON) into machine language. This would enable the drawing of a line to occur at a faster rate. Did you work on the problem?

Creating A Hybrid Program

Mixing machine language and BASIC routines (or any high level language routines) results in what I like to refer to as a *hybrid* program. The use of this technique on the PC-1500 has special advantages. For one thing, it makes it easy to control, for example, input operations -- which could get pretty hairy trying to do in machine language -- as these operations can be left in BASIC. But, when you want to maximize the performance in some area -- such as we are working on doing with the LCD graphics -- then you can go right in and tweak things up using super-fast MLP techniques.

The BASIC used in the PC-1500 makes it easy to construct hybrid programs. You can cross from BASIC to machine language by using the CALL statement. Using CALL &XXXX, where &XXXX represents an address in hexadecimal notation, will simply direct the PC to call a machine language routine at the given address as a subroutine. When a machine language return instruction (RTS) is found, program control will pass back to the BASIC program, just as though you had used a GOSUB statement.

If you want to pass a value from BASIC to a machine language routine, then you may use a variation of the CALL statement. The statement form CALL &XXXX,N will pass the value of variable N to the machine language routine. It does this by placing the variable value into the 16-bit X register of the CPU where it may be picked up and further manipulated by the machine language routine. There are, however, a few restrictions that must be observed: only integer values that fit within 15 bits plus a sign bit, (+32,767 to -32,768) may be passed. If you try to pass a non-integer value using this technique, the BASIC interpreter will effectively integerize it. If you try to pass a value outside this range, you will generate an error message.

You can also pass a value back to BASIC from a machine language routine. If you use the format CALL &XXXX,N then the value of variable N will automatically become the value contained in the 16-bit X register of the CPU when the machine language routine is exited *provided that the carry flag is set at the time of exiting!* You must be careful when exiting a machine language routine back to BASIC when you use this calling format *especially when you do not want the value of the variable to be altered by the ML routine!* Make sure, in such cases, that you clear the carry flag just prior to exiting. Otherwise, *whatever garbage is in the X register of the CPU will end up as an (unwanted!) value for your BASIC variable.* (I speak from experience having once spent some time tracing down a bug caused by my forgetting this point!)

Hybridizing (instead of going entirely to ML routines) does, of course, reduce the overall speed of a program. The BASIC interpreter introduces a lot of overhead in the process of passing control back and forth from it to a machine language routine. The amount of speed sacrificed depends on how many times the transfer between BASIC and ML takes place and other factors. However, even with such handicaps, speed improvements in the order of 10 to 1000 (over programming entirely in BASIC) are generally possible using the technique. You will get a good feel for this aspect if you are following along and actually implementing the routines presented in this series.

The Conversion

The first step in our goal of obtaining speedy line drawing capability is to convert the pixel control routine (that was labeled ON in the preceding article) into machine language. Did you take a try at doing this? Did you learn anything? I bet you did!

The accompanying listings show the routines I came up with using a hybridized approach. I split the process into two parts. One for handling the X coordinate and another for the Y coordinate. That is, I first passed the X value to the ML routine and massaged it into the desired form. When the first ML routine returns to BASIC, another CALL is used to pass the Y value to a second ML routine. This second ML routine combines the Y coordinate with the previously converted information and turns on the desired LCD pixel.

One reason for using this technique at this point was so that ML students could observe the transformation from BASIC to ML coding. That is, to illustrate how BASIC can be used to develop an algorithm (the ON routine in the preceding article) and then show how ML can be used to obtain speed enhancements. The ML routines presented here closely follow the structure of the original BASIC pixel ON routine. The labels used in the machine

Listing *ML Routine to Convert X-Coordinate Value into Display Buffer Address Value.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
70	50	58	76				ON	LDYH #76	Place base address of LCD buffer into Y register.
70	52	5A	00					LDYL #00	Base address is 7600.
70	54	B5	00				COL	LDA #00	Load accumulator with zero.
70	56	AE	78	C0				STA 78C0	Stuff it into 78C0 to clear location used as F flag.
70	59	04						LDA XL	Pull X value (passed from BASIC) into the accumulator.
70	5A	B7	27					CPA #27	Compare it with 39 (decimal). I.e., is X<39?
70	5C	83	05					FBC 1020	Skip if have carry as that means XL=>39.
70	5E	D9						SLA	Multiply X value by 2.
70	5F	12						ADDA YL	Add (X*2) to base address value.
70	60	1A						STA YL	Store the result in Y.
70	61	8E	25					FB ROWSAV	Go to process pixel row value.
70	63	B7	4E				1020	CPA #4E	Compare X value with 78 (decimal). Is X<78?
70	65	83	09					FBC 1030	Skip if have carry as that means XL=>78.
70	67	B1	26					SUBA #26	Subtract 38 + complement of carry so have (X-39).
70	69	D9						SLA	Multiply the result by 2 so have (X-39)*2.
70	6A	12						ADDA YL	Add to base address value.
70	6B	1A						STA YL	Store the result in Y.
70	6C	FD	50					INYH	Form (X-39)*2+256 in Y.
70	6E	8E	18					FB ROWSAV	Go to process pixel row value.
70	70	B5	FF				1030	LDA #FF	Load accumulator with all ones.
70	72	AE	78	C0				STA 78C0	Stuff this into 78C0 to set location used as F flag.
70	75	04						LDA XL	Pull original X value (passed from BASIC) into A register.
70	76	B7	75					CPA #75	Compare it with 117 (decimal). Is X<117?
70	78	83	07					FBC 1040	Skip if have carry as that means XL=>117.
70	7A	B1	4D					SUBA #4D	Subtract 77 + complement of carry so have (X-78).
70	7C	D9						SLA	Multiply the result by 2 so have (X-78)*2.
70	7D	12						ADDA YL	Add to base address value.
70	7E	1A						STA YL	Store the result in Y.
70	7F	8E	07					FB ROWSAV	Go to process pixel row value.
70	81	B1	75				1040	SUBA #75	Form (X-117).
70	83	D9						SLA	Multiply by 2 so have (X-117)*2.
70	84	12						ADDA YL	Add to base address value.
70	85	1A						STA YL	Store in Y.
70	86	FD	50					INYH	Form (X-117)*2+256 in Y.
70	88	68	78				ROWSAV	LDUH #78	Set U to point to temporary storage location.
70	8A	6A	C1					LDUL #C1	Using 78C1 & 78C2 to store converted pixel column address.
70	8C	94						LDA YH	Get high portion into the accumulator.
70	8D	61						STAI (U)	Store into 78C1 with auto-increment of U pointer.
70	8E	14						LDA YL	Get low portion into the accumulator.
70	8F	2E						STA (U)	Store into 78C2.
70	90	F9						CLRC	Clear carry before departing subroutine.
70	91	9A						RTS	Exit back to BASIC to fetch pixel row (Y) value.

language version -- especially the reference to line numbers as ML labels -- are to assist the reader in following the conversion process.

The ML routines are highly commented so I will not use a lot of space discussing them in detail. You will have to do some studying of the listings if you really want to understand the routines in detail. I will, however, point out a few items that highlight differences between BASIC and ML programming.

Note, for instance, that at the start of the ML routine (labeled ON to correspond with the original BASIC version), the display base address &7600 is set up in CPU register Y. There is no need to assign it to a variable as was done in BASIC. The BASIC variable B can be eliminated from the ML version because advancing an address value by 256 is readily accomplished just by incrementing the high order byte of a two-byte address. Remember, too, that the X-coordinate value will be residing in CPU register X when this routine is entered, having been placed there by the BASIC CALL statement.

The ML routines use three designated memory locations as storage places. The flag F (in the BASIC program) is represented by the contents of location &78C0. A value of 00 here means the flag is cleared. A value of &FF represents a set state of the flag. Locations &78C1 and &78C2 are used to hold information contained in a CPU register. This information might be lost when the PC was interpreting BASIC statements, as it must do when

Listing *ML Routine to Convert Y-Coordinate Value to Display Buffer Address and Bit Value.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
70	A0	68	78				ROW	LDUH #78	Set U to point to temporary storage location.
70	A2	6A	C1					LDUL #C1	Starts at 78C1, also uses 78C2.
70	A4	65						LDAI (U)	Fetch high portion into accumulator.
70	A5	18						STA YH	Store pixel column address (high part).
70	A6	25						LDA (U)	Fetch low portion into accumulator.
70	A7	1A						STA YL	Store pixel column address (low part).
70	A8	04						LDA XL	Fetch pixel row value (passed from BASIC).
70	A9	B7	04					CPA #04	See if less than 4.
70	AB	81	04					FBNC FLG	Skip ahead on non-carry as Y<4.
70	AD	B1	04					SUBA #04	Subtract 4 so have Y=Y-4.
70	AF	50						INYL	If Y=>4 then advance display buffer pointer.
70	B1	40					FLG	INXL	Add 1 to pixel row value (0 to 6 becomes 1 to 7).
70	B2	B5	01					LDA #01	Set least significant bit in accumulator.
70	B4	42					LOOP	DEXL	Decrement adjusted row pixel value.
70	B5	8B	03					FBZ OUT	Jump out of loop when XL reaches zero.
70	B7	D9						SLA	Shift bit in accumulator one position to the left.
70	B8	9E	06					RB LOOP	Go back to continue bit-positioning operation.
70	BA	0A					OUT	STA XL	Save positioned bit back in XL.
70	BB	A5	78	C0				LDA 78C0	Fetch F flag from its storage location.
70	BE	00						INA	Exercise the accumulator (in order to set flags).
70	BF	DF						DEA	Restore the accumulator (flags reflect original value).
70	C0	0A						STA XL	Save Y in XL.
70	C0	8B	06					FBZ 1080	If F flag was cleared, then skip ahead.
70	C2	04						LDA XL	Fetch positioned bit from XL.
70	C3	D9						SLA	Rotate the bit into the upper nibble.
70	C4	D9						SLA	Ditto.
70	C5	D9						SLA	Ditto.
70	C6	D9						SLA	Ditto.
70	C7	0A						STA XL	Restore positioned bit to XL.
70	C8	04					1080	LDA XL	Fetch positioned bit from XL.
70	C9	1B						ORA (Y)	Mesh the current bit with whatever is already in display.
70	CA	1E						STA (Y)	Update the display buffer. (At last, at last!)
70	CB	F9						CLRC	Clear the carry flag before departing.
70	CC	9A						RTS	Exit back to BASIC.

It goes back to pick up the Y-coordinate value.

The ML routines and the storage locations just discussed have been assigned to the area in RAM that is normally used to hold string variables A$ - O$. Avoid using a CLEAR directive when these ML routines are installed!

Loading and Experimenting

You can load the ML routines directly into memory from the accompanying listing. You can make up a BASIC program that uses POKE directives to set up the ML routines. Alternately, use a ML monitor program (such as Norlin Rober's LMD package sold by *PCM*) to store the code. Note that the first two columns in the listings provide the address. The next several columns indicate the actual machine code bytes that make up each ML directive. These are the hexadecimal values that must be placed into memory to form a sequence of ML directives.

Once the machine language routines are in memory, you need a BASIC routine that calls them and passes the appropriate X- and Y-coordinates. Here is how that can be done efficiently:

```
3000 "ON"CALL &70
     50,X:CALL &7
     0A0,Y:RETURN
```

To test your new capability you can use a simple input routine such as:

```
1000 "DOT"INPUT "
     X-COORD: ";X
1010 INPUT "Y-COO
     RD: ";Y
1020 CLS :GOSUB "
     ON"
1030 B$=INKEY$ :
     IF B$=""THEN
     1030
1040 GOTO 1000
```

This will allow you to specify display coordinates and observe that the appropriate pixel turns on.

You can combine the LINE routine and the input routines that started at lines 200 and 300 in the previous article to provide complete line drawing

capability. (If you are using the LMD package, you should renumber the input routines so they are above line number 999. Then they will not be mistaken as lines of source code. Label the start of each routine so you can access it from the RUN mode using a labeled directive.)

If you do this you will note that while lines can be drawn considerably faster than when the ON routine was entirely in BASIC, the process is still relatively slow. In order to obtain impressive speed it will be necessary to also convert the LINE routine from BASIC to machine language. Are you ready to tackle that task? Why not give it a try. You can compare your method with that presented in the next installment (which will conclude this series).

A RENUMBERING UTILITY

The Sharp PC-1500 and Radio Shack PC-2 are tremendously powerful devices in the hands of even a modest programmer. However, the recent introduction of the long awaited 16K RAM modules has reminded me of the major stumbling block in writing long application programs on these PCs: the lack of an easy to use renumbering utility. The purpose of this article is to describe an extremely fast "renumbering" routine written in only three lines of BASIC code.

Why do I consider the availability of a renumbering utility to be so important? The BASIC programming language itself is at fault. When a program branches through a GOTO or GOSUB it is necessary to specify a destination line number. Unless you are one of the very few who completely plans out programs in advance on paper, you will find yourself needing to decide what line number your program should jump to as you create code.

Can this be a problem? Suppose that you are going to frequently be calling an as yet unwritten subroutine that will, say, evaluate the polynomial P(x). You make a note on paper that the subroutine will be at line 2000. Every time you need this routine you must either remember it or else look up the the subroutine's starting line number. If you come back to update the program at a later date to modify it you will find yourself in the unpleasant situation of having to trace each subroutine reference in order to understand the program (unless you have otherwise documented the program in a thorough fashion). Things can rapidly get out of a hand in a long program where there may be dozens of GOTOs and GOSUBs. Remember, six months after the fact, a reference to line number 2000 might not do much to remind you that it handles P(x).

Fortunately, Sharp designed the PC-1500 with an unusually powerful feature that is lacking in many home computers: labeling. Any line may be labeled. All that is needed is to place the label in quotes immediately after the line number. You do not even have to place a colon between the label and the first statement in a line.

To jump to the line, you only need to state GOTO or GOSUB followed by the label of the desired line. This label reference must be enclosed in quotation marks. Thus, GOSUB 2000 might be replaced by GOSUB "P(x)" which is much more descriptive than GOSUB 2000. In fact, it is a fair example of self-documenting code.

A programmer has a virtually unlimited range of choices for labels since a label on the PC-1500 can be up to 73 characters in length and include blanks, upper and lower case letters, numerals and any other symbols except the quotation mark.

Since the computer supports this powerful feature, I strongly recommend that PC-1500 programmers use labels for all references to other routines and subroutines.

If the previously cited reasons are not sufficient to convince you of the worthiness of using labels, I can offer one more tantalizing appeal: the following program can renumber a 300 line program in about one minute. This program resides in just three lines of code. But, it only operates on programs wherein all GOTO and GOSUB references are by labels, not line numbers!

Here is the magic routine:

```
65024 END
65025 "L"B=STATUS
      2-STATUS 1,L
      =0
65026 IF PEEK B>25
      3THEN END
65027 L=L+10:POKE
      B,L/256,L-25
      6*INT (L/256
      ):B=B+3+PEEK
      (B+2):GOTO 6
      5026
```

The program may be executed by typing DEF L or RUN 65025. The program then renumbers all lines lower than 65024, assigning the first line number the value of 10 and using an increment between lines of 10. Note that the program does *not* change GOTO and GOSUB statements! Those statements *must* refer to labeled lines.

Here is an outline of how the renumbering program operates:

The first line is just a precaution that prevents other programs from accidently dropping down to the renumbering program.

Line 65025 finds and stores the address of the first byte of your program. This address is stored in variable B. It utilizes the STATUS function so the procedure works regardless of the amount of memory installed in your PC. This line also initializes the variable L which is used to store new line numbers.

The best way to understand the rest of the routine is to recall how programs themselves are stored in the PC-1500. Each BASIC line starts with a two-byte value, most significant byte first, that represents its line number. The third byte is equal to N+1 bytes, where N is the number of bytes in the body of the tokenized BASIC line. The next N bytes are those comprising the tokenized line. The byte after these N bytes (N+1) is the ASCII code for a carriage return (13). All line numbers must be greater than zero. The most significant byte of a line number must be less than 255. This latter restriction limits the programmer to using line numbers that are less than 65280. The last user line in the computer is terminated by the code 255.

The renumber utility ignores all line numbers above 65023. Thus, you can store other commonly used utility routines, such as a base converter, above the renumbering routine, etc. Note that line 65026 in the program terminates its own operation when a line number greater than 65023 is found.

Line 65027 increments the line number (variable L) by 10. It then POKEs it into the proper two bytes as indicated by variable B. Variable B is then incremented by N+4 so that it points to the address of the next BASIC line number. The program then goes back to line 65026 to determine whether it is to continue renumbering.

You may customize this program. The line increment set in line 65027 by the statement L=L+10 may be replaced with whatever increment value suits you. And, instead of setting L=0 in line 65025, you may set L equal to whatever offset you desire, such that the first line number becomes that offset value *plus* the line increment value.

Thanks for this utility procedure and routine go to: *Norman E. Beam, 21051 Gresham Street #103, Canoga Park, CA 91304.*

STAR TARGETS

This is a graphics game for the Radio Shack PC-2 and Sharp PC-1500. It was designed by *David A. Cloutier, Bullard Road, North Brookfield, MA 01535.* You need at least a 4K RAM module in your PC in order to use this program.

To play the game, imagine that you are a ball turret gunner in a space freighter. Your job is to eliminate all enemy fighters trying to destroy the freighter. When the title flashes on the screen, press ENTER *twice* and you will be all set for battle. Press the left arrow key to rotate your turret to the left. Use the right arrow key to rotate it to the right. (Note that the enemy will shift in the opposite direction.) Use the keys to position the enemy into the sights. When the sights start blinking, press the spacebar to fire!

```
3005:REM Star Tar
     gets
3010:CLEAR :CLS :
     DIM S$(0)*18
     ,SU$(0)*18,S
     D$(0)*18,C$(
     0)*18
3015:DIM D1$(0)*2
     2,D2$(0)*22,
     V$(0)*22
3020:S$(0)="120C1
     A2E2A2E1A0C1
     2"
3025:D1$(0)="0008
     2255361D6E10
     244B3A"
3030:SU$(0)="0906
     0F1517150F06
     09"
3035:D2$(0)="0002
     28120D3E0C32
     040012"
3040:SD$(0)="6418
     3C343C343C18
     64"
3045:V$(0)="63410
     000081C08000
     04163"
3047:GOSUB 3300
3050:T=TIME :WAIT
     0:G=60:
     GCURSOR G:
     GPRINT S$(0)
     ;
3060:C=0:RANDOM :
     R=RND 3:IF R
     =1THEN LET C
     $(0)=S$(0)
3070:IF R=2THEN
     LET C$(0)=SU
     $(0)
3080:IF R=3THEN
     LET C$(0)=SD
     $(0)
3090:BEEP 1,15,50
     :GR=(RND 8)-
     4:IF GR=0
     THEN LET GR=
     -4
3095:G=G+GR:IF G>
     155THEN LET
     G=0
3097:IF G<0THEN
     LET G=155
3098:GOSUB 3500:
     GOSUB 3400
3100:BEEP 1,20,50
     :CLS :
     GCURSOR G:
     GPRINT C$(0)
     :GOTO 3060
3300:WAIT 0:PRINT
     CHR$ 127;" S
     tar Targets
      V1.8 !";:
     CURSOR 25:
     PRINT CHR$ 1
     27
3302:IF INKEY$ =
     CHR$ 13THEN
     3302
3304:A=RND 200:B=
     RND 50
3306:GCURSOR 1:
     BEEP 1,A,B:
     GPRINT A,B;:
     GCURSOR 151:
     GPRINT A,B;
3308:IF INKEY$ <>
     CHR$ 13THEN
     3304
3310:CLS :PRINT "
     By David A.
     Cloutier":
     GOSUB 3850
3320:RETURN
3400:P=RND 7:IF P
     <>1THEN
     RETURN
3410:CLS :GCURSOR
     G:GPRINT "08
     04";C$(0);"0
     408";
3420:BEEP 5,7,200
     :P=RND 10:IF
     P<>1THEN
```

Program *Star Targets.*

```
      RETURN
3430: G1=78:G2=79
3440: BEEP 1,G2-70
      ,25:GCURSOR
      G1:GPRINT 25
      5-POINT G1:
      GCURSOR G2:
      GPRINT 255-
      POINT G2
3445: G1=G1-1:G2=G
      2+1
3450: IF G1<0OR G2
      >155THEN 345
      5
3452: GOTO 3440
3455: PRINT "You w
      ere blown to
       pieces!":
      GOSUB 3850
3460: PRINT "Score
       =";SC:GOSUB
      3850
3470: PRINT "Ships
       =";WS:GOSUB
      3850
3480: END
3500: GCURSOR 73:
      GPRINT "41";
      :GCURSOR 82:
      GPRINT "41"
3505: A$=INKEY$ :
      GOSUB 3590
3510: IF G<>74THEN
      RETURN
3520: CLS :GCURSOR
      73:GPRINT U$
      (0);:BEEP 1,
      8,100:A$=
      INKEY$
3530: IF A$=" "
      THEN 3600
3540: CLS :GCURSOR
      G:GPRINT S$(
      0);:BEEP 1,8
      ,200:C=C+1
3550: IF C=3THEN
      RETURN
3560: GOTO 3520
3590: IF A$=CHR$ 1
      2THEN LET G=
      G-5
3592: IF G<0THEN
      LET G=155
3595: IF A$=CHR$ 8
      THEN LET G=G
      +5
3596: IF G>155THEN
      LET G=0
3597: IF A$=" "
      THEN 3599
3598: RETURN
3599: A$=INKEY$ :
      GOTO 3597
3600: GCURSOR 73:
      GPRINT D2$(0
      );:BEEP 5,9,
      200
3610: CLS :GCURSOR
      73:GPRINT D1
      $(0);:BEEP 5
      ,20,50
3620: G1=72:G2=84
3630: GCURSOR G1:
      GPRINT (RND
      128)-1:
      GCURSOR G2:
      GPRINT (RND
      128)-1:BEEP
      1,G2,5
3640: G1=G1-4:G2=G
      2+4:IF G1<=0
      OR G2>=155
      THEN LET WS=
      WS+1:GOSUB 3
      700:GOTO 305
      0
3650: GOTO 3630
3700: SC=INT (SC+1
      00-(100*(
      TIME -T)))
3720: PRINT "Score
       =";SC:GOSUB
      3850
3730: PRINT WS;" e
      nemy ships d
      estroyed":
      GOSUB 3850
3840: GOTO 3860
3850: IF INKEY$ =
      CHR$ 13THEN
      3850
3855: IF INKEY$ <>
      CHR$ 13THEN
      3855
3857: IF INKEY$ <>
      CHR$ 13THEN
      3857
3860: CLS :RETURN
```

REDEFINING THE PC-1500 KEYBOARD

If you have POKEd around in the Sharp PC-1500 you may have noticed a little Japanese symbol appear in the display. This occurs when bit 2 of location &764E is set. (You can see it by entering: POKE &764E,PEEK &764E OR 4.) When the PC is in this mode, keyboard codes may be fetched from a user-built table. To do this, a pointer stored at &785D must be set to &80 when using a CE-150 or to &00 when using a CE-158 (RS-232) interface.

In this discussion, the use of a CE-150 will be assumed. For this case the user-built table must start at an address: &nn80. The high byte of this address (nn) is indicated to ROM by setting the pointer at &785E to (nn+1).

The first keyboard table contains 128 bytes. Sixty-four are used for the regular keyboard and 64 for when the shift key is utilized. Keys to be redefined must have ASCII codes greater than A0.

A second table holds GPRINT codes -- five per symbol -- that are used to define each character. This table is located at address &(nn+1)A0. The length of this second table depends on the number of keys redefined. It takes five bytes to define each key.

Thus, the directive POKE &785D,&80,&3A will indicate to ROM that a keyboard table starts at &3980 (going to &39FF) and that there is a GPRINT table starting at &3AA0.

(Keys having ASCII codes less than &80 will be fetched from the regular ROM table at addresses &FCA0 through &FE7F.)

The program shown in the accompanying listing capitalizes on this information. It *adds* a set of Greek alphabet characters to the repertoire of the PC-1500. Note that I said *adds* and not *substitutes*. All other capabilities remain. Thus, by a single keystroke, you may switch from one keyboard representation to another.

Actually, the program only redefines 35 keys. This provides 24 lowercase Greek letters, 10 uppercase Greek symbols (as the other 14 uppercase Greek characters are identical to English ones), and the apostrophe which Sharp did not bother to provide. To facilitate using these new symbols within BASIC programs, only lowercase keyboard characters were redefined.

The program consist of two parts. One POKEs the two necessary tables into the low portion of memory. The other is a little machine language routine that activates the Greek mode whenever desired.

Of course, these keyboard tables must be protected by an appropriate NEW statement to isolate them from the user's BASIC storage area. However, the program works with any RAM memory expansion (4K, 8K or 16K modules). This is

because the first available address is computed after examining the SOM (Start Of Memory) pointer that is maintained in address &7863.

Installation and Use

The first step you must take is to protect a section of low memory where the table will be stored. Do this by executing the command:

```
NEW(PEEK &7863+2)*256+336
```

(If you are using an 8K RAM module, then you could just enter: NEW &3B50.)

Next, load the entire BASIC program shown in the accompanying listing into memory.

Execute this program by typing RUN in the RUN mode. This causes the two tables to be placed into the protected area of low memory. After a few seconds the > prompt will come back up on the LCD. Now you can eliminate all of the BASIC lines *except for line 300.*

To save the keyboard tables on tape for future use, set up two variables by executing these statements:

```
A=(PEEK &7863+1)*256+128
B=(PEEK &7863+2)*256+335
```

Now place the tape unit in the record mode and execute the command:

```
CSAVE M "GREEK KB TABLE";A,B
```

(Again, if you are using an 8K RAM module, then use: CSAVE M"GREEK KB TABLE",&3980,&3B4F.)

Then, in the future you need only reserve the protected area in memory and load the keyboard table directly into memory using the command:

```
CLOAD M "GREEK KB TABLE";(PEEK &7863+2)*
256+128
```

You also need to restore line 300 (but you may renumber that as desired). Alternately, you may assign the contents of that line to a Reserved key:

```
(CALL(PEEK&7863+2)*256)
```

However, make sure you erase this assignation when the Greek table is not in memory!

That is all there is to it. Press the DEF/G key (or the Reserved key if you set it up as suggested) when you want to use Greek symbols. The Japanese

Diagram *Symbols Assigned to Numeric Keys.*

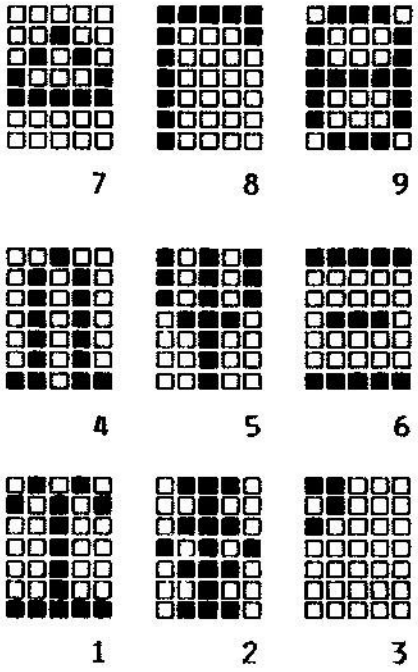
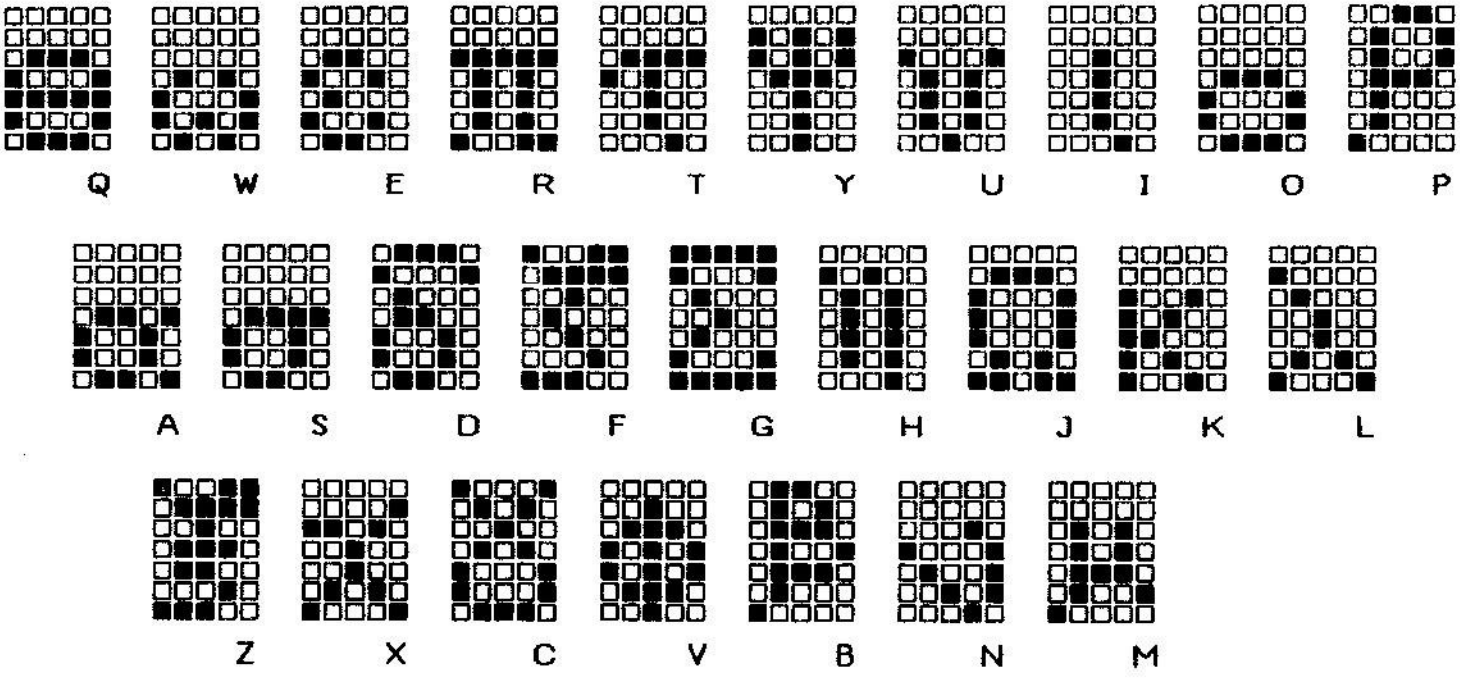


Diagram *Greek Symbols Assigned to Alphabetical Keys.*



symbol will come up on the display (to the left of the SMALL legend) to indicate that you are in this special keyboard mode. Remember these symbols are accessed using lowercase, so press the SHIFT key before pressing the desired alphabetical or numeric key. An accompanying diagram shows the Greek keyboard layout.

To switch back to the regular keyboard, press the SML key twice.

If you turn the PC off using the BREAK key, you will automatically quit the Greek mode. However, if the PC turns off by powering-down after the 7 minute time-out, you will remain in whatever mode you were using.

Of course, you may use these new Greek symbols just as you would any other characters: they can serve in labels, PRINT, PAUSE and INPUT statements, as well as file names.

Naturally the technique demonstrated here may be modified to enable your PC to communicate in whatever language you desire, be it Russian, Hebrew, or Sanscript. How about converting your PC to French, complete with accentuated vowels?

This powerful technique for customizing your PC keyboard was provided by: *Patrick L. Pollet, P.O. Box 144, UPM Box 298, Dhahran International Airport, Saudi Arabia.*

Program ***Greek Keyboard***

```
1:REM greek alp
  habet keyboard
2:DATA &0B,&4E,&
  59,&01,&48,&38
  ,&35,&32,&09,&
  58,&57,&11,&53
  ,&1F,&2D,&2E
3:DATA &30,&4D,&
  55,&15,&4A,&37
  ,&34,&31,&0D,&
  28,&49,&16,&4B
  ,&4F,&4C,&29
4:DATA &19,&43,&
  45,&12,&44,&2F
  ,&2A,&2B,&20,&
  56,&52,&13,&46
  ,&50,&08,&3D
5:DATA &02,&5A,&
  51,&1B,&41,&18
  ,&1F,&0C,&0A,&
  42,&54,&14,&47
  ,&39,&36,&33
6:DATA &5B,&A0,&
  A1,&01,&A2,&A3
  ,&A4,&A5,&09,&
  A6,&A7,&21,&A8
  ,&0F,&2C,&2E
7:DATA &30,&A9,&
  AA,&25,&AB,&AC
  ,&AD,&AE,&0D,&
  3C,&AF,&26,&B0
  ,&B1,&B2,&3E
8:DATA &19,&B3,&
  B4,&22,&B5,&3F
  ,&3A,&3B,&5E,&
  B6,&B7,&23,&B8
  ,&B9,&1D,&40
9:DATA &02,&BA,&
  BB,&1B,&BC,&1A
  ,&1E,&1C,&5D,&
  BD,&BE,&24,&BF
  ,&C0,&C1,&C2
19:REM machine c
  ode routine to
   activate tabl
  es
20:DATA &B5,&80,&
  AE,&78,&5D,&A5
  ,&78,&63,&DD,&
  DD
21:DATA &AE,&78,&
  5E,&EB,&76,&4E
  ,&04,&9A,&38,&
  38
29:REM table ppr
  int codes of t
  he special cha
  racters
30:DATA &08,&10,&
  20,&44,&38,&06
  ,&08,&7E,&08,&
  06
31:DATA &02,&3C,&
  02,&7C,&00,&7F
  ,&01,&01,&01,&
  03
32:DATA &07,&08,&
  7F,&08,&07,&08
  ,&55,&7F,&55,&
  08
33:DATA &44,&24,&
  18,&24,&42,&30
  ,&48,&20,&48,&
  30
34:DATA &30,&48,&
  48,&38,&08,&40
  ,&3C,&10,&1C,&
  20
35:DATA &04,&38,&
  40,&38,&04,&5C
  ,&62,&02,&62,&
  5C
36:DATA &18,&14,&
  12,&14,&18,&40
  ,&7E,&01,&7E,&
  40
37:DATA &42,&41,&
  7E,&41,&42,&00
  ,&3C,&40,&00,&
  00
38:DATA &7C,&10,&
  28,&44,&00,&30
  ,&48,&48,&48,&
  30
39:DATA &42,&24,&
  18,&20,&40,&31
  ,&4A,&44,&4A,&
  31
40:DATA &28,&54,&
  44,&28,&00,&32
  ,&4D,&49,&31,&
  02
41:DATA &18,&24,&
  7E,&24,&18,&44
  ,&3C,&04,&7C,&
  44
42:DATA &41,&4A,&
  56,&23,&03,&40
  ,&3E,&09,&09,&
  06
43:DATA &41,&5A,&
  5E,&2B,&03,&38
  ,&54,&54,&54,&
  38
44:DATA &30,&48,&
  48,&30,&48,&40
  ,&3F,&15,&16,&
  08
45:DATA &08,&04,&
  3C,&44,&04,&63
  ,&55,&49,&41,&
  63
46:DATA &3E,&49,&
  49,&49,&3E,&41
  ,&49,&49,&49,&
  41
47:DATA &00,&05,&
  03,&00,&00
200:"D"A=(PEEK &78
  63+1)*256+&80
210:FOR I=0TO 127:
  READ M:POKE A+
  I,M:NEXT I
220:A=(PEEK &7863+
  2)*256
230:FOR I=0TO 19:
  READ ML:POKE A
  +I,ML:NEXT I
240:A=(PEEK &7863+
  2)*256+&A0
250:FOR I=0TO 174:
  READ M:POKE A+
  I,M:NEXT I
260:END
300:"G"CALL (PEEK
  &7863+2)*256:
  END
STATUS 1
                1758
```

FROM THE WATCH POCKET

I receive a lot of mail here at *PCN*. There is no way that I could respond to every inquiry. Over the years I have developed some basic rules or policies that I follow in dealing with this ongoing deluge.

The first is that I do not even look at letters that are rude and/or obnoxious. Should, by chance, one get by my secretarial staff, (and it is trained to remove letters from people with personal problems that have nothing to do with our line of work), I drop it into the wastebasket the moment it becomes evident that the intent of the letter is to vent someone's grief, hostility, anger or frustration. I am not a social or medical counselor nor a receptacle for abusive thoughts and ravings.

Many of the letters that I do review contain technical questions relating to subjects I have written about. If I am able to quickly respond to the questions by jotting down a brief reply (a few words to a few phrases - not a few paragraphs or pages!) and, *if the inquirer has had the courtesy of enclosing a self-address, stamped envelope,* then I often do forward a reply. (Not always, there is no guarantee, even with a S.A.S.E.)

But you can bet that if you do *not* include a self-addressed, stamped envelope, the chances of receiving a reply are approximately zero.

I do use comments received in letters to help shape the contents of *PCN*. For instance, if I receive a lot of letters from all around the country indicating that people are having difficulty using the CE-158 RS-232C interface, I assume there is a fair amount of interest in the subject. I thus select one or two articles dealing with that subject for publication.

Recently it seems that new purchasers of the CE-161 16K RAM module sometimes experience difficulty trying to use their Reserve keys. Despite the warnings published in the instruction manual, people are attempting to use these modules by entering NEW 0 after installation. The manual states to use NEW 256. Do what the manual says. If you don't, the system cannot set aside the proper storage space for the Reserve keys.

I frequently get letters asking about how to improve the performance of the tape cassette program/data storage and recall system used with the Sharp and Radio Shack PCs. The consensus seems to be that top quality tape is a great help. Can I name a brand that provides ultra-reliability? NO! I wouldn't attempt to do so, there are too many variables affecting the outcome. I can give some practical advice. Don't give too much credence to the word "certified" on a package of tape. Certified by whom, for what, under which conditions? A good rule of thumb is that if the brand you are using isn't working, then switch!

Can I name a tape recorder that works well with PCs? Sure. The Radio Shack Minisette-9, which is the model originally recommended by Radio Shack for use with their PCs, seems to do very well for the staff at *PCN*. Furthermore, we have received good reports about its use in this application from many other users. That does not mean you *have* to use a Minisette-9.

Do not expect perfection from any audio tape cassette recording system. They were never really meant to store computer data. It is not appropriate to attempt to store large amounts of critical data using these systems. The technology being used is such that one can expect to encounter an error in every 30 minutes or so of tape recorder use. It is wise to keep such limitations in perspective when you consider your applications!

From the Rumor Mills

It does appear as though 1984 will be the year of the notebook computer. Besides the ones already announced in *PCN*, the streets are rife with talk of entries from many more firms. It seems Casio may be one of the next to announce. Those already in the running are apparently planning on spending megabucks to promote their entries. Watch the magazines, TV and listen to radio for the barrage Sharp is expected to release touting the PC-5000.

I also expect to see a barrage of exciting new software offerings and PC add-ons in the coming year. Most of these will be from small companies that have figured the PC base is now big enough to take a gamble on.

POCKET
COMPUTER
NEWSLETTER

Issue 30 - December

CARLETON 16K EPROM MODULES

Carleton Micro Systems has announced the availability of a 16K EPROM Module for the Sharp PC-1500 and PC-1500A or the Radio Shack PC-2 computers. Using this new module, program source codes can be stored permanently. Furthermore, Carleton states that the source code in these modules can be protected against any kind of viewing, copying or editing. This is accomplished by disabling the LIST, LLIST, CSAVE and editing functions.

The firm claims that these modules are much more reliable than battery backup RAM modules. Using these modules, complete software packages may be quickly interchanged by simply plugging them into the computer hardware.

Using a Carleton EPROM Module with a Sharp PC-1500A results in a system containing 16K of program ROM and 8K of data RAM.

The company can either burn a program into the module or assist software vendors in programming the modules on their own. Normally, software

received on cassette or battery backup RAM is transferred to the Carleton ROM Modules. Carleton works under standard non-disclosure terms to protect vendors software. Orders for as many as 250 ROM Modules can be processed in less than a week.

The service is primarily designed to support software vendors who wish to produce modules in quantity. A minimum module order is normally 25 pieces. At this quantity, a 16K EPROM Module is priced at $150.00 each, plus an initial programming setup fee. The price declines to the range of $100.00 per module for higher quantities.

For further information contact: **Carleton Micro Systems,** *1801 Commerce Drive, South Bend, IN 46624.* The phone number is (219) 236-4630.

HEWLETT-PACKARD ANNOUNCES HP-41CX

Hewlett-Packard Corporation keeps striving to bring pocket computer capabilities to its line of Series 40 advanced programmable calculators. Now the firm has added clock and calendar functions, plus text-file editing and extended memory to its HP-41 line in its new HP-41CX.

The new model has all the features of its predecessor, the HP-41CV, plus those mentioned in the preceding paragraph, 20 new commands, and over 3,100 bytes of memory.

The Time Module in the HP-41CX enables the operator to use the highly portable unit as a time-based controller, an alarm clock, an appointment reminder, a timer or a stopwatch.

The Extended Functions/Memory Module in the HP-41CX (which was an option in the 41CV) provides 868 bytes of extended memory, memory-management functions, programmable versions of several HP-41 functions, plus several register and flag manipulation functions.

The 24K bytes of ROM in the HP-41CX features an RPN operating system. This sytem permits users to view intermediate results as well as recover easily from error conditions. The RAM in the HP-41CX retains its contents when the unit is turned off. An alphanumeric LCD screen permits viewing of both numeric and alphabetical characters plus special symbols and operators. The alphanumeric keyboard is redefinable. Thus, users may assign frequently used programs or functions to a single key to customize operation. Keyboard overlays are available for users who want to create their own labeled keyboards.

The HP-41CX can be expanded, via the HP-IL (Hewlett-Packard Interface Loop) . Peripherals include printers, plotters and instruments. All of these can be maintained as a battery-operated system allowing complete portability.

For more information on the $325.00 HP-41CX, contact your local HP sales office.

Photo ***HP-41CX Programmable Calculator.***

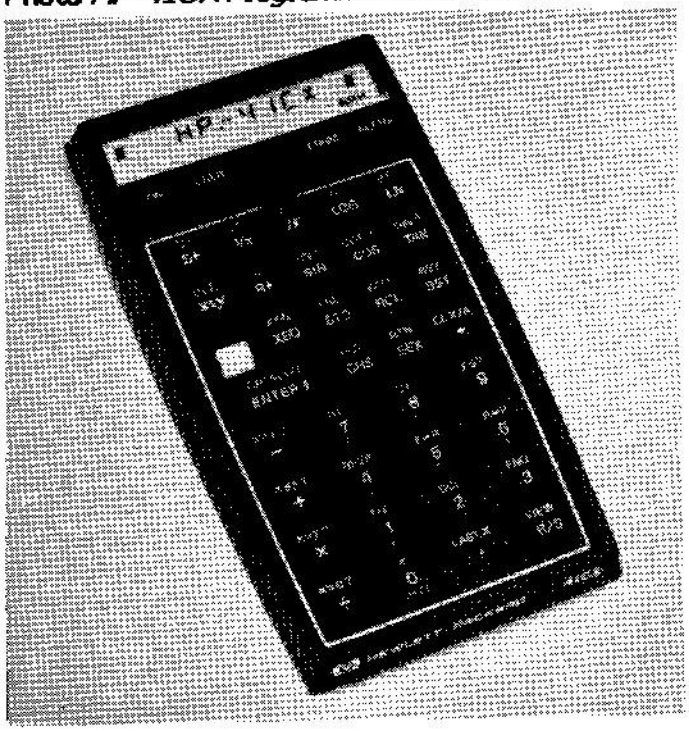
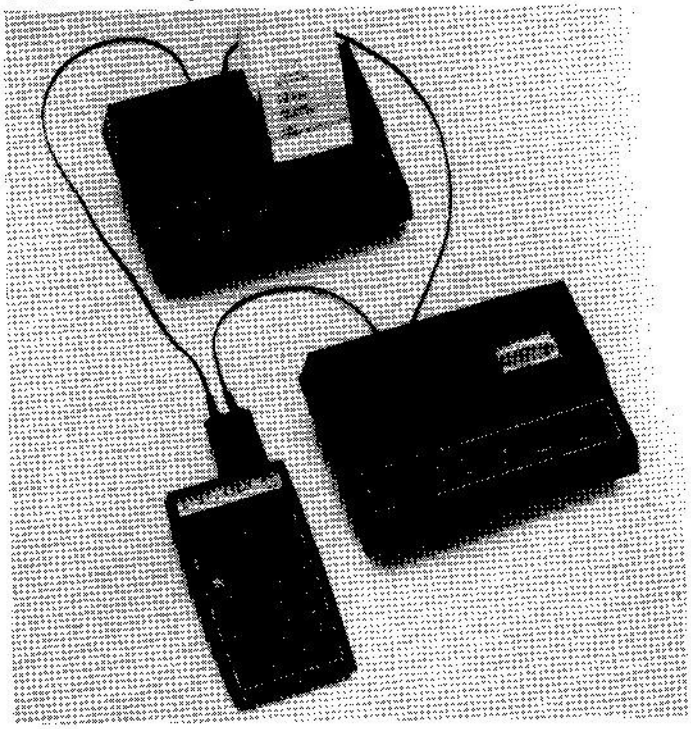


Photo ***HP Peripherals Connected Via HP-IL.***



ENHANCED FILE MAINTENANCE PROGRAM

One of the satisfying aspects of being the Editor of *PCN* is seeing the improvements that come about over time as a consequence of publishing hard to obtain information. As you know, during the past year (and more), *PCN* has published a great deal of information on machine language programming for the Sharp PC-1500 and Radio Shack PC-2. We are now seeing the results of having made this type of information widely available.

The program presented here is an excellent example of the types of improvements that can be obtained through the judicious application of ML techniques. This is an enhanced version of the File Maintenance program that originally appeared in Issue 24 of *PCN*. It was developed by the originator of the program, *Stephen Tomback, 19 Maplewood Way, Pleasantville, NY 10570*. Through the use of machine language routines, Mr. Tomback has vastly improved the overall operating speed of the program. Additionally, he has removed the former barrier (imposed by BASIC) that limited record lengths to 80 characters. If you found the original version useful, you will be ecstatic over this one!

Overview

The File Maintenance program will allow the set up of a filing system to a user's specific needs. Think of a file box filled with 3 x 5 cards. The box is called a *file*, the cards are called *records*, and the lines on the cards are called *fields*.

This programs allows the user to define the fields on the cards, store information in those fields, and have the ability to retrieve cards by searching fields for specific information. Thus, one could utilize this file maintenance program as a filing system for such things as birthdays, anniversaries, magazine subscriptions, or investments. A periodic search (such as monthly) would serve as a reminder as to which items were coming due.

Equipment Required

This program operates in a Radio Shack PC-2 or a Sharp PC-1500 that is equipped with an 8K memory expansion module. In order to make use of the program you will also need a printer/cassette interface such as the Sharp CE-150.

About the File

Here is important information for you to know about the structure of the file:

Number of Records: The program utilizes just under 3K of memory. The remainder is used for storage of the file. Using an 8K memory expansion module gives the PC a total memory capacity of about 10 kilobytes. Thus, about 7 kilobytes would be available for data storage. The number of records that can be stored will depend on the total number of bytes assigned to a record. For instance, a file utilizing 100 bytes per record would allow for the storing of approximately 70 records.

File Name: When a file is created, it must be assigned a name. This name may not exceed 15 characters. The assigned name is saved with the file. (Using a file name that is shorter than 15 bytes will not increase the amount of data that can be stored.)

Field Name: A record may contain up to 12 fields. (A field is simply an item within a record. A name, an address, and a telephone number might each constitute a field.) When a file is created, the user must assign a name to each field. That name must not exceed 15 characters. Again, assigning less than 15 characters to a field name will not make additional room for the storage of data.

Field Length: The length of a field may be set at from 1 to 40 bytes.

Record Length: The record length is a function of the number of field(s) defined and their length(s). Since a field may have a maximum of 40 bytes and a record may contain up to 12 fields, the maximum record length is 480 bytes. (A record this size would not be practical in most situations because it would only allow approximately 14 records in a file.)

Fields Per Record: A record may have from 1 to 12 fields.

Field Search: A record may be searched by 1 to 4 fields. If you set up a file of birthdays as well as anniversaries, it would be beneficial to establish a field called TYPE. This field would be used to distinguish between the two categories (say B and A, respectively). Another field might be named MONTH DUE. Finding all the birthdays that occurred in a particular month could then be accomplished by a two field search.

Before Starting

The first time the RUN command is issued, the program POKES several machine language routines into memory. It then erases those POKE commands from the BASIC program in order to save memory. Attempting to copy the program after it has been started should thus be avoided.

Also, using the RUN command will cause any file currently in memory to be erased. Do *not* use RUN to restart the program once you have any part of a file stored in memory. Always use the directive GOTO 15 to restart the program without loss of data. (You might want to place such a statement under control of a softkey.)

The program uses the black pen of the printer. Make sure you have a good pen in that position.

When selecting from the menu, remember that

Program *File Maintenance*

```
10:GOSUB 995:LOCK
   :TEXT :CSIZE 1
   :COLOR 0
15:CLS :WAIT 0:D=
   9:H=0:RESTORE
   :FOR I=1TO D:
   READ I$(0):
   PRINT I$(0)
20:FOR J=1TO 30:I
   $=INKEY$ :IF I
   $<>""LET J=60
25:NEXT J
30:IF I$=""THEN 5
   0
35:FOR J=1TO D:IF
   MID$ ("NOI1AVP
   FC", J, 1)=I$LET
   H=J:J=D
40:NEXT J
45:IF HCLS :ON H
   GOSUB 100, 200,
   300, 400, 500, 70
   0, 700, 800, 900:
   GOTO 15
50:NEXT I:GOTO 15
55:DATA "N - NEW
   FILE","O - OUT
   PUT/TAPE","I -
    INPUT/TAPE"
60:DATA "1 - FILE
    PARAMETERS","
   A - ADD RECORD
   ","P - PRINT R
   ECORD"
65:DATA "V - VIEW
    RECORD","F -
   FIND RECORD(S)
   ","C - CHANGE
   RECORD"
100:IF LEN N$=0
   THEN 115
105:BEEP 3:INPUT "
   SURE? ";I$
110:IF I$<>"Y"
   RETURN
115:GOSUB 995
120:INPUT "NEW FIL
   E NAME: ";N$
125:IF LEN N$<1OR
   LEN N$>15THEN
   120
130:INPUT "FIELDS
   PER RECORD: ";
   K
135:IF K<1OR K>12
   THEN 130
140:FOR I=1TO K
145:CLS :PRINT "FI
   ELD";I;" NAME:
   ":CURSOR 14:
   INPUT "";@$(I+
   14)
150:IF LEN @$(I+14
   )<1OR LEN @$(I
   +14)>15THEN 14
   5
155:CLS :INPUT "HO
   W MANY BYTES:
   ";@(I+14)
160:IF @(I+14)<1OR
   @(I+14)>40THEN
   155
165:L=L+@(I+14):
   NEXT I
170:GOSUB 190
175:PRINT "MAXIMUM
    RECORDS";N:
   BEEP 15:RETURN
190:N=INT ((STATUS
   3-STATUS 2-63)
   /L):RETURN
200:IF LEN N$=0
   GOSUB 960:
   RETURN
205:INPUT "TAPE RE
   ADY? ";I$
210:IF I$<>"Y"
   RETURN
215:PRINT #N$;@$(*
   ),@(*)
220:I$=N$+"*":
   CSAVE MI$;
   STATUS 2,
   STATUS 2+(M*L)
   :RETURN
300:IF LEN N$=0
   THEN 315
305:BEEP 3:INPUT "
   SURE? ";I$
310:IF I$<>"Y"
   RETURN
315:GOSUB 995:
   INPUT "FILE NA
   ME: ";N$
320:INPUT #N$;@$(*
   ),@(*)
325:GOSUB 190
330:IF N<MPRINT "F
   ILE SPACE EXCE
   EDED!":BEEP 15
   :GOSUB 995:
   RETURN
335:I$=N$+"*":
   CLOAD MI$;
   STATUS 2:
   RETURN
400:IF LEN N$=0
   GOSUB 960:
   RETURN
405:LF 5:LPRINT "F
   ILENAME: ";N$
410:LPRINT "MAXIMU
   M RECORDS:";N
415:LPRINT "RECORD
   S USED:";M
420:LPRINT "BYTES
   PER RECORD:";L
   :LF 1
425:FOR I=1TO K:
   LPRINT "FIELD"
   ;I;" ";@$(I+14
   );TAB 28;USING
   "###";"BYTES";
   @(I+14):USING
430:NEXT I:LF 6:
   RETURN
500:IF LEN N$=0
   GOSUB 960:
   RETURN
505:IF M=NPRINT "O
   UT OF RECORDS!
   ":BEEP 15:
   RETURN
510:F=STATUS 2+(M*
   L):FOR I=1TO K
   :G=I+14
515:CLS :I$(0)="".
   PRINT @$(G),".
   ":CURSOR LEN @
   $(G)+2:INPUT "
   ";I$(0)
520:IF LEN I$(0)>@
   (G)GOSUB 985.
   GOTO 515
525:GOSUB 955:F=F+
   @(G):NEXT I:
   CLS :M=M+1:
   PRINT "ADD REC
   ORD";M:BEEP 15
   :RETURN
600:LF 1:LPRINT "R
   ECORD:";I.LF 1
605:G=STATUS 2+((I
   -1)*L):FOR F=1
   TO K:E=F+14
610:CALL STATUS 3-
   5,@(E):CALL
   STATUS 3-14,G:
   J=STATUS 3+7:
   CALL STATUS 3-
   35,J:G=G+@(E)
615:IF I$<>"V"
   LPRINT STR$ (F
   );" ";I$(0):
   NEXT F:RETURN
620:J=1
625:PRINT STR$ (F)
   ;" ";MID$ (I$(
   0), J, 24)
630:IF ASC INKEY$
   =8LET J=J+1:
   GOTO 650
635:IF ASC INKEY$
   =12LET J=J-1:
   GOTO 650
640:IF ASC INKEY$
   <>13THEN 630
645:CLS :NEXT F:
   RETURN
650:IF J=0LET J=1
655:IF J>LEN I$(0)
   LET J=LEN I$(0
   )
660:GOTO 625
700:IF M=0GOSUB 96
   0:RETURN
705:I=0:INPUT "REC
   ORD NUMBER. ";
   I
710:IF I<1OR I>M
   THEN RETURN
715:IF I$="V"GOSUB
   605.RETURN
720:GOSUB 600.LF 5
   :RETURN
800:IF M=0GOSUB 96
   0:RETURN
805:CLS .H=0:INPUT
   "HOW MANY FIEL
   D SEARCH? ";H
810:IF H<1OR H>4OR
   H>KTHEN RETURN
815:RESTORE 885:
   FOR I=1TO H:
   READ I$:J$=
   STR$ I
820:CLS :@(I)=0:
   CURSOR :PRINT
   J$;I$:CURSOR 4
   :INPUT "FIELD
   NUMBER: ";@(I)
825:IF @(I)<1OR @(
   I)>KTHEN 820
830:CLS :I$(0)="".
   CURSOR :PRINT
   "SEARCH";@(I);
   " FOR: ";
   CURSOR 14:
```

```
    INPUT "";I$(0)
835:IF LEN I$(0)>1
    6OR LEN I$(0)>
    @(@(I)+14)
    GOSUB 985:GOTO
    830
840:IF LEN I$(0)<1
    THEN 830
845:@$(I)=I$(0):
    NEXT I
850:FOR I=1TO M:
    FOR J=1TO H
855:CLS :PRINT "RE
    CORD";I
860:G=@(J):GOSUB 9
    90
865:I$="".FOR E=1
    TO LEN @$(J).I
    $=I$+CHR$ PEEK
    F:F=F+1:NEXT E
870:IF MID$ (I$,1,
    E-1)<>@$(J)
    NEXT I.RETURN
875:NEXT J
880:GOSUB 600.NEXT
    I.RETURN
885:DATA "st","nd"
    ,"rd","th"
900:IF M=0GOSUB 96
    0:RETURN
905:I=0:INPUT "REC
    ORD NUMBER. ",
    I
910:IF I<1OR I>M
    THEN RETURN
915:G=0:INPUT "FIE
    LD NUMBER: ";G
920:IF G<1OR G>K
    THEN 915
925:I$(0)="":INPUT
    "CHANGE TO: ",
    I$(0)
930:IF LEN I$(0)>@
    (G+14)GOSUB 98
    5:GOTO 925
935:GOSUB 990:G=G+
    14:GOSUB 955:
    RETURN
955.CALL STATUS 3-
    5,@(G):CALL
    STATUS 3-14,F:
    J=STATUS 3+7:
    CALL STATUS 3-
    63,J:RETURN
960:PRINT "FILE EM
    PTY!".BEEP 15:
    RETURN
985.CLS .PRINT "TO
    O LONG!":BEEP
    15.RETURN
990.F=STATUS 2+((I
    -1)*L)
992.IF G=1RETURN
994:FOR E=1TO G-1.
    F=F+@(E+14):
    NEXT E:RETURN
995:CLEAR .DIM I$(
    0)*41
996:POKE STATUS 3-
    35,&A5,&76,&D0
    ,&DF,&2A,&A5,&
    76,&D1,&18,&A5
    ,&76,&D2,&1A,&
    55,&41,&88
997:POKE STATUS 3-
    19,4,&B5,0,&41
    ,&9A,&84,&AE,&
    76,&D1,4,&AE,&
    76,&D2,&9A,4,&
    AE,&76,&D0,&9A
998:POKE STATUS 3-
    63,&A5,&76,&D0
    ,&DF,&2A,&A5,&
    76,&D1,&18,&A5
    ,&76,&D2,&1A,&
    45
999:POKE STATUS 3-
    49,&B7,0,&8B,4
    ,&51,&88,8,&9A
    ,&B5,32,&51,&8
    8,3,&9A
1000:POKE STATUS
    2,&84,&28,4,
    &2A,&BE,&D2,
    &EA,0,&A5,&7
    8,&A6,8,&A5
1005:POKE STATUS
    2+13,&78,&A7
    ,&A,&46,&46,
    &46,&FB,&A4,
    &AE,&76,&D0,
    &9A
1010:POKE STATUS
    2+25,&84,&AE
    ,&78,&67,4,&
    AE,&78,&68,&
    9A
1015:I=996.CALL
    STATUS 2,I:
    IF PEEK &76D
    0=&BSTOP
1020:X=STATUS 2-I
    -27
1025:A=STATUS 2-1
    -X
1030:CALL STATUS
    2+25,A
1035:POKE I-1,&3A
    ,&F1,&99,&D,
    3,&E4,&14,&F
    1,&AB,&53,&2
    0,&54,&4F
1040:POKE I+12,&4
    D,&42,&41,&4
    3,&4B,&20,&4
    6,&4D,&20,&3
    8,&2F,&38,&3
    3,&D,&FF
1045:POKE I-15,&1
    1
1050:CLEAR :DIM I
    $(0)*41:
    RETURN
STATUS 1      3566
```

only a single keystroke is needed. All other entries, such as data, must be terminated by pressing the ENTER key. For instance, to establish a new file just depress the N key when the menu is cycling. To enter a new file name, type in the desired name followed by the ENTER key.

At the Beginning

Load the program into the computer, place the PC into the RUN mode and type RUN. A continuously cycling menu will be displayed containing the following items:

N - NEW FILE
O - OUTPUT/TAPE
I - INPUT/TAPE
1 - FILE PARAMETERS
A - ADD RECORD
V - VIEW RECORD
P - PRINT RECORD
F - FIND RECORD(S)
C - CHANGE RECORD

An Illustration

To show how the program may be applied, a file for holding birthdays and anniversaries will be created. It will be named AUTOCALENDAR. The seven fields will be titled and sized as follows:

	Name	Length (Bytes)
Field 1:	TYPE	1
Field 2:	MONTH DUE	2
Field 3:	NAME	20
Field 4:	ADDRESS	20
Field 5:	CTY ST ZP	23
Field 6:	PHONE	12
Field 7:	NOTE	16

Setting Up A New File

A new file is created using the NEW FILE option. Depress the N key while the menu is cycling and NEW FILE NAME will be displayed. If a file already exists the programs asks SURE? Any response other than Y at this point will return the program to the main menu. Responding Y clears the memory of any previous file. The program will then prompt for

a NEW FILE NAME. This procedure permits a user who has inadvertently selected the NEW FILE function to exit without erasing the existing file.

If you desire to follow along with this example, enter the name AUTOCALENDAR for the new file name. The program will then ask FIELDS PER RECORD. Enter the number 7 for this illustration. (Remember, the maximum permitted is 12, a value greater than that will not be accepted.)

The PC will then request the name of each field and its length in characters. Respond as outlined previously if you want to follow the example. After the file has been set up, the program will issue a beep, then display the maximum number of records that may be stored (74 in this example). After this the program will go back to displaying the menu.

Printing File Parameters

This option causes the current filename, maximum number of records that can be stored in the file, the number of records used, the number of bytes in a record, and each field name and its length to be printed. After outputting this information the program reverts to displaying the menu.

If you are following the example procedure, it is a good idea to select this option and verify that you have set up the file correctly. Depress the number 1 digit key while the menu is cycling to select this function. If you find that the file is not set up as you desire, then stop the program with the BREAK key. You would then issue a RUN command to restart the program.

It is important to realize that once defined, you cannot alter the file parameters without losing all the data currently in memory. Make sure you have things set up properly before you start building up a file of valuable data!

Adding A Record

Now that a file has been created and its proper format verified, try adding a birthday to it. Depress the A key while the menu is cycling. The first field description (TYPE:) will appear. Respond B to this field for birthday. (Remember, field 1 in this application is used used to differentiate between anniversaries [A] and birthdays [B].)

The second field description, MONTH DUE, will then be displayed. Respond with a two-digit answer. Thus, 01 should be used for January. The value 12 would be used for December. (If a single digit was used for January, then a single character search of the MONTH DUE field for the digit 1 would find both the 1 and the 12 entries. Using two characters means that a properly specified search will yield exactly the desired match.)

Continue entering the rest of the information for the record as prompted by the field names. (You might want to use the NOTE field for the actual birthdate or some other relevant information.)

When all the fields in the record have been inputted, the computer will beep. It will then show the record number assigned to that entry before going back to display the menu.

Viewing A Record

To display the contents of a record on the LCD, use this option. It is selected by pressing the V key when in the menu mode. Respond to the prompt for RECORD NUMBER by entering the desired value. (To view the example record just entered, respond with a 1, assuming it is the first entry in your file.)

The number that appears on the lefthand side of the screen while viewing is the field number. This is followed by the name of the field and then its contents. Press ENTER to display each consecutive field within a record.

The LCD can display 24 characters of the fields numbered 1 through 9 and 23 characters of the fields numbered 10 through 12. However, the forward and back arrow keys may be used to scroll the screen right and left when fields exceed these lengths.

When the last field in a record has been displayed, pressing the ENTER key will return the program to the cycling menu.

Printing A Record

Use the PRINT RECORD option, selected by pressing the P key, to list a record on the printer. (To list the example record, respond to the prompt for RECORD NUMBER with a 1.) When the record has been printed, control returns to the menu.

Changing A Record

Depress the C key when in the menu mode to bring up this option. It permits the altering of data within a specified field in a designated record. Respond appropriately to the prompts for RECORD NUMBER and FIELD NUMBER. When the prompt CHANGE TO is displayed, enter the desired data.

If the new information is different in length than the original field entry, the remainder of the field is filled with spaces. To delete the contents of a field, respond to the CHANGE TO prompt by pressing just the ENTER key. This causes the entire field to be filled with spaces.

After a field has been changed, the program returns to the menu mode.

Finding Record(s)

Records in a file may be located using this option. From one to four fields may be searched at a time. After you have inputted several birthdays and anniversaries into the example file, try finding a

particular birthday by month and name as follows:

Depress the F key while in the menu mode to bring up the prompt HOW MANY FIELD SEARCH? For purposes of illustration, assume a three-field search on the fields named TYPE, MONTH DUE and NAME.

Now respond to the 1st FIELD NUMBER prompt with a 1 to indicate that the first field (TYPE) is to be checked. Respond to SEARCH 1 FOR with a B for birthday in this example.

Respond to the 2nd FIELD NUMBER prompt with a 2 as the second field (MONTH DUE) is to be examined. Respond to the prompt SEARCH 2 FOR with the month of the birthday you are seeking.

Respond to 3rd FIELD NUMBER with a 3 to indicate that the third field (NAME) is to be inspected. Respond to SEARCH 3 FOR with the name of the person you are seeking.

After this entry the program will begin looking for the record that matches all the parameters requested. As it searches it will display the record number that it is examining. When it finds a record matching all the specifications, it will output it to the printer. When the search has been completed, the program returns to the menu mode.

It should be noted that a field search may be performed in any order. While field numbers 1, 2 and 3 were specified in the example, they could have been requested in the order 3, 1, and then 2.

Saving A File On Tape

Files may be saved on tape. Make sure the tape recorder is properly connected and set to the record mode. Check that the cassette has been advanced beyond the tape leader. Press the O key to bring up the TAPE READY? query. Any response other than Y will terminate the procedure. A response of Y will cause the file to be recorded on tape using the name of the current file. (For instance, AUTOCALENDAR, in the case of the current example.) When the file has been written, program operation reverts to the menu.

Since data files can *not* be verified, it is a good idea to make a second copy of important files on a second tape cassette. Just load in another tape and repeat the tape save process.

Restoring A File From Tape

Files that have been previously stored on tape may, of course, be read back for use by this program. Make sure that the tape unit is properly connected and in the playback mode. Press the I key to bring up the prompt SURE? An answer of Y at this point clears the memory of any previous file. The program will then prompt for the FILE NAME. This procedure provides an exit from inadvertent selection of the INPUT/TAPE option without erasing an existing file.

Inputting the name of a file (such as the example AUTOCALENDAR) causes the program to begin reading the tape. The file names it finds are displayed on the LCD.

A data file is actually saved in two parts by this program. Hence, when it is recovered, it will be processed as one file under the original user-assigned name (AUTOCALENDAR for example) and another file (AUTOCALENDAR*) that has the same name suffixed by an asterisk. The program will automatically process both files. When the procedure has been completed, the display will return to the cycling menu.

Deleting Records

In order to save memory, a delete option was not included in the program. However, if a record that has been created is no longer needed, then alter one of its fields to signify that it has been deleted. For instance, place the letter D in field 1. When you need a new record, use the FIND RECORD(S) option to locate such a deleted record. Then use the CHANGE RECORD option to edit in your new data.

RABBIT CHASE

David Hergert, 4714 Chickering Ave., Cincinnati, OH 45232 submitted this clever and challenging game. It is designed to run on the Sharp PC-1500 or Radio Shack PC-2 equipped with at least 4K of RAM. It uses the printer to draw graphics at the beginning of the program. Additionally, if you successfully complete a game by concluding the fourth level, another set of graphics is drawn as a reward.

The game itself is played on the LCD. In the game, the player is pictured as a rabbit that is trying to eat all the vegetables in Farmer Brown's garden. Farmer Brown is trying to catch the rabbit with his tractor. The objective is for the rabbit to eat all of the vegetables on all four levels of play, before Farmer Brown catches it. Each level of play is harder than the previous one. (Level four is really tough!)

Pressing the number 4 key causes the rabbit to move to the right. The number 6 key changes its direction to the left. The rabbit has wrap-around capabilities, but the tractor does not. Don't let the tractor catch the rabbit!

If you become totally frustrated and unable to win the game at level four, you can see the final graphics by entering RUN 950. If you want to skip the opening printer graphics, use RUN 20. The normal start is to simply issue a RUN command.

Listing *Rabbit Chase*

```
 5:FOR X=25TO 40:
   BEEP 1,X,X*2:
   NEXT X
10:TEXT :COLOR 0
15:LPRINT TAB 3;"
   RABBIT CHASE":
   LF 4:GOSUB 700
20:POKE 20509,72,
   118,74,0,5,189
   ,68,64,65,78,7
   8,153,9
30:POKE 20522,76,
   119,139,6,72,1
   19,74,0,158,18
   ,154
40:WAIT 0:Z=1:DIM
   Y(5):Y(1)=75:Y
   (2)=120:Y(3)=7
   5:Y(4)=120
50:O=1:N=11:X=0:Y
   =Y(Z):A2=147:A
   1=10:IF Z=5
   THEN 950
60:IF Z>2LET N=12
80:CLS
90:CALL 20509:
   POKE &7701,&44
95:GCURSOR X:
   GPRINT "000000
   00002070707C30
   "
100:IF INKEY$ ="4"
   GCURSOR X:
   GPRINT "307C70
   702000000000000
   ":X=X-4:IF X<=
   0THEN LET X=14
   7:GOTO 300
110:IF INKEY$ ="4"
   AND A1>=Y+10
   AND A2>=Y
   GCURSOR Y:
   GPRINT "3C783B
   3F780000000000
   0000":Y=Y-2:
   GOTO 100
120:IF INKEY$ ="4"
   AND A1<=YAND A
   2<=Y+10GCURSOR
   Y:GPRINT "3C78
   3B3F7800000000
   00";Y=Y-2:GOTO
   100
130:IF A1>Y-1AND A
   2>YGCURSOR Y:
   GPRINT "3C783B
   3F780000000000
   00":GOTO 160
140:IF A1<YAND A2<
   Y+11GCURSOR Y:
   GPRINT "3C783B
   3F780000000000
   ":GOTO 160
150:GCURSOR Y:
   GPRINT "3C783B
   7F384040404040
   "
160:Y=Y-2
170:IF A1>=A2-NLET
   Z=Z+1:GOTO 50
180:IF Y<=X+4
   LPRINT "GOTCHA
   ":CLS :INPUT "
   NEW GAME?(Y/N)
    ";BB$:IF BB$=
   "Y"CLEAR :GOTO
   40
190:IF BB$="N"END
200:IF INKEY$ ="4"
   AND Y<=XLPRINT
   "GOTCHA":CLS :
   INPUT "NEW GAM
   E?(Y/N) ";BB$:
   IF BB$="Y"
   CLEAR :GOTO 40
210:X=X+2:IF X>A1
   LET A1=X
220:IF INKEY$ ="4"
   THEN 100
230:GOTO 95
300:GCURSOR 0:
   GPRINT 0;0;0;0
   ;0;0;0;0;0;0;0
310:IF X<=A2-10LET
   A2=X
320:GCURSOR X:
   GPRINT "307C70
   702000000000000
   "
330:IF INKEY$ ="6"
   GCURSOR X:
   GPRINT "000000
   00002070707C30
   ":X=X+4:IF X<1
   47THEN 370
340:IF INKEY$ ="6"
   AND X>146LET X
   =0:GCURSOR 147
   :GPRINT 0;0;0;
   0;0;0;0;0;0;0:
   GOTO 95
350:IF A2>101LET O
   =1
360:IF A2<=101AND
   Z>2LET O=3
370:IF A1<=YAND A2
   <=YGCURSOR Y-2
   :GPRINT "00000
   0000000783F3B7
   83C":GOTO 400
380:IF A1>=Y-9AND
   A2>=YGCURSOR Y
   :GPRINT "00000
   00000783F3B783
   C":GOTO 400
390:GCURSOR Y:
   GPRINT "404040
   4040783F3B783C
   "
400:IF A1>=A2-NLET
   Z=Z+1:GOTO 50
410:Y=Y+O
420:IF Y>=X-4
   LPRINT "GOTCHA
   ":CLS :INPUT "
   NEW GAME?(Y/N)
    ";BB$:IF BB$=
   "Y"CLEAR :GOTO
   40
430:IF BB$="N"END
435:IF INKEY$ ="6"
   THEN 330
440:X=X-2:IF X<A2
   LET A2=X
450:GOTO 300
700:GRAPH :LINE (0
   ,0)-(155,40),2
   ,3,B
710:LINE (0,0)-(15
   5,-40),2,3,B
720:LINE (0,-40)-(
   155,-80),2,3,B
730:LINE (20,-80)-
   (15,-80)-(15,-
   78)-(13,-78)-(
   13,-72),0,2
731:LINE (13,-72)-
   (17,-68)-(19,-
   68)-(22,-70)
740:LINE (22,-70)-
   (22,-65)-(24,-
   63)-(25,-62)-(
   23,-70)-(26,-7
   4)
741:LINE (26,-74)-
   (23,-76)-(22,-
   76)-(20,-78)-(
   15,-78)
750:LINE (22,-76)-
   (20,-78)-(15,-
   78)
800:LINE (20,-40)-
   (15,-40)-(15,-
   38)-(13,-38)-(
   13,-32)
801:LINE (13,-32)-
   (17,-28)-(19,-
   28)-(22,-30)
805:LINE (22,-30)-
   (22,-25)-(24,-
   23)-(25,-22)-(
   23,-30)-(26,-3
   4)
806:LINE (26,-34)-
   (23,-36)-(22,-
   36)-(20,-38)-(
   15,-38)
810:LINE (20,0)-(1
   5,0)-(15,2)-(1
   3,2)-(13,8)
811:LINE (13,8)-(1
   7,12)-(19,12)-
   (22,10)
815:LINE (22,10)-(
   22,15)-(24,17)
   -(25,18)-(23,1
   0)-(26,6)
816:LINE (26,6)-(2
   3,4)-(22,4)-(2
   0,2)-(15,2)
820:LINE (20,40)-(
   15,40)-(15,42)
   -(13,42)-(13,4
   8)
821:LINE (13,48)-(
   17,52)-(19,52)
   -(22,50)
825:LINE (22,50)-(
   22,55)-(24,57)
   -(25,58)-(23,5
   0)-(26,46)
826:LINE (26,46)-(
   23,44)-(22,44)
   -(20,42)-(15,4
   2)
830:LINE (0,40)-(0
   ,-80),0,1
835:LINE (155,40)-
   (155,-80)
840:LINE (100,45)-
   (100,62)-(114,
   62)-(114,55)-(
   132,55)-(132,4
   5)
841:LINE (132,45)-
   (100,45):LINE
   (105,40)-(110,
   45),,,B:LINE (
   120,40)-(125,4
   5),,,B
842:LINE (120,72)-
   (125,67),,,B:
```

```
    LINE (125,70)-
    (125,55)
850:LINE (50,5)-(5
    0,22)-(64,22)-
    (64,15)-(82,15
    )-(82,5)
851:LINE (82,5)-(5
    0,5):LINE (55,
    0)-(60,5),,,B:
    LINE (70,0)-(7
    5,5),,,B
852:LINE (70,32)-(
    75,27),,,B:
    LINE (75,30)-(
    75,15)
860:LINE (100,-35)
    -(100,-18)-(11
    4,-18)-(114,-2
    5)-(132,-25)-(
    132,-35)
861:LINE (132,-35)
    -(100,-35):
    LINE (105,-40)
    -(110,-35),,,B
    :LINE (120,-40
    )-(125,-35),,,
    B
862:LINE (120,-8)-
    (125,-13),,,B:
    LINE (125,-10)
    -(125,-25)
870:LINE (50,-75)-
    (50,-58)-(64,-
    58)-(64,-65)-(
    82,-65)-(82,-7
    5)
871:LINE (82,-75)-
    (50,-75):LINE
    (55,-80)-(60,-
    75),,,B:LINE (
    70,-80)-(75,-7
    5),,,B
872:LINE (70,-48)-
    (75,-53),,,B:
    LINE (75,-50)-
    (75,-65)
900:COLOR 1
910:TEXT :LF -6:
    TAB 13:LPRINT
    "LEVEL"
915:TAB 14:LPRINT
    "4"
920:LF 1:TAB 14:
    LPRINT "3"
925:LF 1:TAB 14:
    LPRINT "2"
930:LF 1:TAB 14:
    LPRINT "1"
935:LF 2:LPRINT "P
    RESS 4 TO MO
    VE LEFT":
    LPRINT "PRESS
     6 TO MOVE RI
    GHT":LF 2
940:IF INKEY$ ="S"
    LPRINT "YOU AR
    E ON":RETURN
945:WAIT 0:PRINT "
    PRESS S TO S
    TART":GOTO 940
950:FOR X=40TO 1
    STEP -1:BEEP 1
    ,X,X*2:NEXT X
960:TEXT :LF 4:
    GRAPH
970:LINE (150,0)-(
    160,10)-(180,1
    0)-(185,20)-(2
    00,20),0,3
971:LINE (200,20)-
    (190,0)-(150,0
    ):LINE (155,0)
    -(160,-5),,,B
972:LINE (175,0)-(
    180,-5),,,B:
    LINE (167,10)-
    (180,30),0,2:
    LINE (180,30)-
    (190,40),,,B
973:LINE (173,17)-
    (183,17)
975:LINE (20,0)-(2
    4,35)-(35,46)-
    (40,45)-(55,42
    )-(60,42),0,1
976:LINE (60,42)-(
    70,42)-(75,62)
    -(69,70)-(62,6
    0)-(60,42)-(70
    ,42)
977:LINE (70,42)-(
    80,35)-(78,20)
    -(69,15)-(52,0
    )-(20,0)
978:LINE (50,-10)-
    (25,-10)-(25,0
    ):LINE (66,31)
    -(69,34),,,B:
    LINE (24,33)-(
    18,39),,,B
980:COLOR 0
990:TEXT :LF 5:
    LPRINT "SUPER
    RABBIT!":END
STATUS 1        3847
```

ALTERNATE CONTROL OF THE PC-1500 LCD

This is the third and concluding part of this series. The challenge I put forth at the end of the previous article (in Issue 29 of *PCN*) was for you to work on converting the LINE routine from BASIC to ML. That routine was presented in Issue 28 of *PCN* and had been assigned BASIC line numbers 5000 - 5140. It is used to plot the pixels that must be turned on in order to draw a smooth straight line between two pairs of X,Y coordinates on the LCD screen. Converting the LINE routine to machine language will greatly increase the speed at which lines can be drawn.

There are, of course, almost an infinite number of ways in which one could approach such a project. Machine language offers such a wide range of instructions and approaches that virtually no two programmers would produce the exact same series of code for such a project. Thus, the sequence of instructions you might have used to approach this problem could be completely different than those that I chose.

A Parallel Course

In order to provide a comparative approach for readers, I elected to essentially duplicate the BASIC routine using ML methods. In particular, I chose to manipulate the same BASIC variables (M, N, R, S, U and V). I used this method primarily so that those of you who may not have a lot of ML experience could observe how a ML routine can closely pattern a BASIC routine. This technique does not provide for the most efficient or fastest running ML program. However, even this relatively crude approach offers vastly increased speed over that of BASIC. For, while the same variables are being manipulated, the "interpretive" process that slows down BASIC has been removed.

The ML listing on the accompanying pages shows the ML code that replaces lines 5000 through 5060 of the original BASIC program (refer to *PCN* Issue 28). This represents roughly half of the LINE routine. In order to conserve space (since the commented ML listing is rather lengthy), the second half of the converted LINE routine (representing lines 5070 - 5140) is provided simply as an object code listing. It is similar in format to the first half. (Those with a deep interest can use a disassembler to study this part of the code in detail.)

The entire machine code for the LINE routine fits within exactly 512 bytes of memory. I assigned this code to the address range &0000 - &01FF since I have a 16K RAM module in the PC I used for development of this package. However, the entire code in this 512-byte block is relocatable. Thus, if

Listing *First Half of LINE Routine Converted To Machine Language.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	58	7A				LINE	LDYH #7A	Value required by ROM FP routine.
00	02	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	04	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	06	BE	F7	3F				JSR F73F	ROM subroutine to transfer S into FP register R0.
00	09	E6						CALL E6	ROM subroutine to transfer FP register R0 into R2.
00	0A	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	0C	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	0E	BE	F7	3F				JSR F73F	ROM subroutine to transfer R into FP register R0.
00	11	BE	EF	B6				JSR EFB6	ROM subroutine to subtract FP registers R0-R2 (R-S).
00	14	A5	7A	02				LDA 7A02	Fetch first byte of R0 mantissa into the accumulator.
00	17	8B	E7					FBZ 5070	Jump to work 2nd dimension when S=R (mantissa is zero).
00	19	A5	7A	01				LDA 7A01	Fetch sign byte of R0 into the accumulator.
00	1C	8B	06					FBZ S<R	If sign byte is zero, then S less than R.
00	1E	68	00					LDUH #00	When sign byte negative, S>R, so set register U=60001.
00	20	6A	01					LDUL #01	Finish setting register U=60001.
00	22	8E	04					FB UINR0	Jump to place contents of register U into FP register R0.
00	24	68	FF				S<R	LDUH #FF	When sign byte positive, S<R, so set register U=-1.
00	26	6A	FF					LDUL #FF	Finish setting register U=-1 (6FFFF).
00	28	CD	10				UINR0	CALL 10	ROM subroutine to place CPU register U into R0.
00	2A	80						80	Byte that needs to be passed for ROM subroutine.
00	2B	48	7A					LDXH #7A	Value required by ROM FP routine.
00	2D	58	79					LDYH #79	Establish pointer to BASIC variable P.
00	2F	5A	78					LDYL #78	Establish pointer to BASIC variable P.
00	31	BE	F7	11				JSR F711	ROM subroutine to transfer R0 into BASIC variable P.
00	34	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	36	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	38	58	79					LDYH #79	Establish pointer to BASIC variable M.
00	3A	5A	60					LDYL #60	Establish pointer to BASIC variable M.
00	3C	BE	F7	33				JSR F733	ROM subroutine to transfer BASIC variable R into M.
00	3F	38						NOP	Save a byte for alterations??? (Safety valve.)
00	40	58	7A				5040	LDYH #7A	Value required by ROM FP routine.
00	42	48	79					LDXH #79	Establish pointer to BASIC variable U.
00	44	4A	A0					LDXL #A0	Establish pointer to BASIC variable U.
00	46	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable U to R0.
00	49	E6						CALL E6	Ron subroutine to transfer FP register R0 to R2.
00	4A	48	79					LDXH #79	Establish pointer to BASIC variable V.
00	4C	4A	A8					LDXL #A8	Establish pointer to BASIC variable V.
00	4E	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable V to R0.
00	51	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (V-U).
00	54	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 (V-U) onto FP stack.
00	57	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	59	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	5B	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable R to R0.
00	5E	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	5F	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	61	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	63	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	66	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (S-R).
00	69	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	6A	BE	DC	16				JSR DC16	ROM subroutine to pop FP stack into R0 (V-U).
00	6D	CD	58					CALL 58	ROM subroutine to divide R0 by R2 ((V-U)/(S-R)).
00	6F	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 ((V-U)/(S-R)) onto FP stack.
00	72	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	74	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	76	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable R to R0.
00	79	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	7A	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	7C	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	7E	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	81	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (M-R).
00	84	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	85	BE	DC	16				JSR DC16	ROM subroutine to pop FP stack into R0 ((V-U)/((S-R)).
00	88	CD	7E					CALL 7E	ROM subroutine to multiply R0 by R2 ((V-U)/(S-R))*(M-R).
00	8A	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	8B	48	79					LDXH #79	Establish pointer to BASIC variable T. Note: T=0.5!
00	8D	4A	98					LDXL #98	Establish pointer to BASIC variable T.
00	8F	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable T to R0.
00	92	F0						CALL F0	ROM subroutine to add R2 to R0 ((V-U)/(S-R))*(M-R)+T.
00	93	BE	F5	BE				JSR F5BE	ROM subroutine to form integer of quantity in R0.
00	96	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	97	48	79					LDXH #79	Establish pointer to BASIC variable U.
00	99	4A	A0					LDXL #A0	Establish pointer to BASIC variable U.
00	9B	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable U to R0.
00	9E	F0						CALL F0	Add R2 to R0 to form (INT((V-U)/(S-R))*(M-R)+0.5)+U.
00	9F	D0						CALL D0	ROM subroutine to convert R0 to CPU register U.
00	A0	04	00					04 00	Bytes that need to be passed for ROM subroutine.
00	A2	FD	A8					PSHU	Save contents of register U on processor stack.
00	A4	58	7A					LDYH #7A	Value required by ROM FP routine.
00	A6	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	A8	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	AA	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	AD	D0						CALL D0	ROM subroutine to convert R0 to CPU register U.
00	AE	04	00					A0 00	Bytes that need to be passed for ROM subroutine.
00	B0	FD	28					LDX U	Load CPU register X with contents of CPU register U.
00	B2	BE	70	50				JSR 7050	Pass X-coordinate value to X-coordinate display routine.
00	B5	FD	0A					POPX	Pop former contents of CPU register U into CPU register X.
00	B7	BE	70	A0				JSR 70A0	Pass Y-coordinate value to Y-coordinate display routine.
00	BA	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BB	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BC	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BD	58	7A					LDYH #7A	Value required by ROM FP routine.
00	BF	48	79					LDXH #79	Establish pointer to BASIC variable P.
00	C1	4A	78					LDXL #78	Establish pointer to BASIC variable P.
00	C3	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable P to R0.
00	C6	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	C7	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	C9	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	CB	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	CE	F0						CALL F0	ADD M to P to get new value of M (loop variable).
00	CF	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 (M) onto FP stack.
00	D2	48	7A					LDXH #7A	Value required by ROM FP routine.
00	D4	58	79					LDYH #79	Establish pointer to BASIC variable M.
00	D6	5A	60					LDYL #60	Establish pointer to BASIC variable M.
00	D8	BE	F7	11				JSR F711	ROM subroutine to transfer R0 to BASIC variable M.
00	DB	A5	79	79				LDA 7979	Fetch sign byte of BASIC variable P into accumulator.
00	DE	8B	0E					FBZ SUBM	If P is positive, go calculate S-M.
00	E0	58	7A					LDYH #7A	Value required by ROM FP routine. (Prepare for M-S.)
00	E2	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	E4	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	E6	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	E9	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	EA	CD	30					CALL 30	ROM subroutine to pop M from FP stack to FP register R0.
00	EC	8E	0A					FB SUBTCT	P is negative, go calculate M-S.
00	EE	CD	30				SUBM	CALL 30	ROM subroutine to pop variable M from FP stack into R0.
00	F0	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	F1	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	F3	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	F5	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	F8	BE	EF	B6			SUBTCT	JSR EFB6	Subtract R2 from R0 (S-M when P+, M-S when P-).
00	FB	A5	7A	01				LDA 7A01	Fetch sign byte of R0 into the accumulator.
00	FE	9B	C0					RBZ 5040	Loop back to process next display position on positive R0.
01	00	58	7A				5070	LDYH #7A	Begin second part of the LINE subroutine. . . .

Listing ***Second Half of LINE Routine.***

```
0100 58 7A 48 79   0180 3F BE EF B6
0104 4A A8 BE F7   0184 E6 BE DC 16
0108 3F E6 48 79   0188 CD 58 E6 48
010C 4A A0 BE F7   018C 79 4A 98 BE
0110 3F BE EF B6   0190 F7 3F F0 BE
0114 A5 7A 02 8B   0194 F5 BE E6 48
0118 E6 A5 7A 01   0198 79 4A 88 BE
011C 8B 06 68 00   019C F7 3F F0 D0
0120 6A 01 8E 04   01A0 04 00 FD 28
0124 68 FF 6A FF   01A4 BE 70 50 58
0128 CD 10 80 48   01A8 7A 48 79 4A
012C 7A 58 79 5A   01AC 68 BE F7 3F
0130 70 BE F7 11   01B0 D0 04 00 FD
0134 48 79 4A A0   01B4 28 BE 70 A0
0138 58 79 5A 68   01B8 38 38 38 38
013C BE F7 33 38   01BC 58 7A 48 79
0140 58 7A 48 79   01C0 4A 70 BE F7
0144 4A A0 BE F7   01C4 3F E6 48 79
0148 3F E6 48 79   01C8 4A 68 BE F7
014C 4A 68 BE F7   01CC 3F F0 BE DB
0150 3F BE EF B6   01D0 F5 48 7A 58
0154 BE DB F5 48   01D4 79 5A 68 BE
0158 79 4A 88 BE   01D8 F7 11 A5 79
015C F7 3F E6 48   01DC 71 8B 0E 58
0160 79 4A 90 BE   01E0 7A 48 79 4A
0164 F7 3F BE EF   01E4 A8 BE F7 3F
0168 B6 E6 BE DC   01E8 E6 CD 30 8E
016C 16 CD 7E BE   01EC 0A CD 30 E6
0170 DB F5 48 79   01F0 48 79 4A A8
0174 4A A0 BE F7   01F4 BE F7 3F BE
0178 3F E6 48 79   01F8 EF B6 A5 7A
017C 4A A8 BE F7   01FC 01 9B BF 9A
```

you are using, say, a CE-155 (8K) module, then you could place this block of code in the range &3800 - &39FF. (Just remember to use a NEW &XXXX command before loading the code to protect the ML area. On my 16K system I use NEW &0200. If you had a CE-155 you would use NEW &4000. And, if you relocate the code, then you must be sure and change the address of the BASIC CALL statement that is used to access this code!)

Before attempting to use the code presented here, you must also install the two ML routines provided in the previous issue (29) of *PCN*. These start at addresses &7050 and &70A0. This is vital as the LINE routine calls upon these subroutines. (They are called by jump to subroutine instructions in the ML code for the LINE routine.)

The BASIC Connection

With the ML routines installed in memory (which replace the original BASIC LINE and ON routines), all one needs to do to draw a line on the LCD is to do the following: Define the starting coordinates of the line by setting the variables R,U (for X- and Y-coordinate positions). Define the ending points by setting the variables S,V. Call the new LINE routine in this manner:

```
5000:"LINE"T=.5:CALL &0000:RETURN
```

(Note that the BASIC variable T must be defined as the constant 0.5 prior to calling the machine code. This value is used by the ML routines. While it could have been set by the ML routines, doing so would have caused them to consume more than 512 bytes.)

Want to see your PC quickly draw a box on the LCD? Just use the following BASIC statements that call on the ML line-drawing routines:

```
1000 CLS
1010 R=0:U=0:S=6:V=0:GOSUB "LINE"
1020 R=6:U=0:S=6:V=6:GOSUB "LINE"
1030 R=6:U=6:S=0:V=6:GOSUB "LINE"
1040 R=0:U=6:S=0:V=0:GOSUB "LINE"
1050 END
```

A Few Words of Caution

ML programming gives you greater control over the operation of the PC. However, using it requires greater responsibility of the programmer. For instance, if the values of R, S, V and U are not within valid ranges (0-155 for R and S, 0-6 for U and V), then the ML routines can end up stuffing garbage into RAM locations, ultimately leading to a "crash" of the PC.

Thus, it was with grief that I found out that I had made an error in the BASIC line doodler routine that I provided in Issue 28 of *PCN*. Line 310 of that routine should read as follows:

```
310 R=RND (156)-1:
    S=RND (156)-1:
    U=RND (7)-1:V=
    RND (7)-1
```

The original statements in that line could lead to out of range values being generated (where the X-coordinate exceeded 155 or Y exceeded 6). Make sure you correct this line if you want to use that random line generator routine to test the ML routines!

Now You Are One Your Own

You now have the capability of dealing with the LCD screen as an X,Y plane. You can rapidly draw lines between selected coordinates. As illustrated earlier, you can connect lines to form boxes. Of course, this concept can be extended to create elaborate drawings. You have a new tool in your software library. How you use and apply this new capability is up to you.

SEASONS GREETINGS

FROM THE WATCH POCKET

Here it is, the end of our third year of publication. As is the custom in the year-end edition, I will take a little extra space for this column. This enables me to review progress in the past year as well as speculate on what may occur in the year ahead.

Not Much Happened

The past year was certainly not a stellar one in terms of PC progress. Hardly anything that might be considered revolutionary was introduced on the hardware side. Most of what might be called real progress boils down to just a few minor model changes.

Perhaps the most important advance came when Sharp was convinced that 2K of RAM was not enough for today's sophisticated PC users. The Sharp PC-1500A is an exact replica of the PC-1500 (right down to the ROM chips) with some more internal RAM. In the new PC-1500A there is 6K of user RAM in the address range &4000 through &57FF. There is another 1K of RAM, apparently intended for ML routines, tucked in the address range &7C00 through &7FFF. Just don't use the first byte of that area as it occassionally gets stuffed with a &0D when the BASIC input buffer overflows. (A software quirk?) Add in the area dedicated to storage of string and numeric variables and you have the 8K of memory Sharp is entitled to advertise. For all practical purposes, however, your BASIC programs must fit in 6K of memory, not the 8K you might assume.

Despite the hopes and prayers of many users, there was very little in the way of peripherals added by the PC manufacturers. Most notably absent was the introduction of any kind of practical external mass storage device. Of course, Sharp may view their 16K RAM module, backed up by batteries, as a viable alternative. Personally, I do not. At about $150 per clip, it is a long way from what I consider low cost mass storage. Of course, you can always spend a half-hour loading 16K of RAM from tape cassette, provided you don't catch a glitch.

Hurray For The Little Guys

Perhaps the most exciting news of the year is being generated by a small firm in Pennsylvania. *Robert Undi* of **Usonics**, *7901 Oak Hill Drive, Cheltenham, PA 19012*, reports that they have successfully mated a "stringy floppy" data drive to the PC-1500. Priced at about $400.00, this unit gives users the capability of storing and recovering data at a rate that allows 16K to be transferred in about 30 seconds. Supplies of the special drive are said to be somewhat limited. But, if you are looking for something about 50 times faster than audio cassette, you might look into this item.

Usonics also sells the new Brother EP-22 portable typewriter. This is the advanced version of the earlier EP-20. The beauty of this model is that it has a built-in RS-232C interface. Just connect it to your CE-158 RS-232C interface and you have a highly portable external printer. The EP-22 is priced at about $250.00. When not being used as a printer, it can serve as an electronic portable typewriter, complete with 2K of internal, editable, memory! A big cut above the earlier model, and at just a $50.00 premium, this seems like a good buy.

Put Your Program(s) On ROMs?

A growing trend among PC users in businesses is to develop packages of programs that can be used by many employees. Groups of these programs are then "burned" into ROMs for general distribution.

Several firms are now engaged in providing ROM programming services. Out in Indiana, a firm by the name **Carleton Micro Systems** is providing a complete modular package that can hold 8K or 16K of EPROM. The module is intended primarily for businesses, institutions or software vendors that need 25 or more packages of the same program(s). The firm takes software supplied on cassette or a battery-backed RAM module and converts it to their EPROM units. The modules disable the editing, program listing and store functions of the PC-1500 so that the installed software cannot be copied. In lots of 25, the 16K modules sell for approximately $150.00 each plus a software setup fee.

If you don't need 25 or more copies of a program and you can get by with just 8K of program storage, you might be interested in the module duplication service offered by **Atlantic N.E. Marketing**, *P.O. Box 921, Marblehead, MA 01945*. Duplication of your own 8K (battery-backed) module starts at about $100.00 for a single unit and declines as the quantity increases.

OEM Action Increasing

The pocket computer market currently seems to be flat to declining at the consumer level, but increasing in the OEM market. In other words, the PC is being repackaged by others to end up with a product that performs a specific function or serves a particular purpose.

Thus, there is now a company that sells a PC-1500 dressed up as a Bond Analyzer. For about $700.00 you can get a system, completely pre-programmed, that calculates just about everything a businessperson would want to know about bond prices, yields, etc. Another firm makes a specialty of providing PC-1500s to banks, all set up to

1983 Index -- Ordered By Author

Last Name	First Name	Title	Issue
Auersbacher	Enerich	Wind Chill Equivalent Temp.	21
Auersbacher	Enerich	Count-Down Timer Program	27
Bean	Norman	Renumbering Utility	29
Beckman	Mel	Sidelister Program	22
Campbell	Diane	Cross Reference for the PC-2	27
Carter	Don	26K of RAM in the PC-1500/PC-2	23
Cloutier	David	Star Targets	29
Delinger	Willian	Controlling Lamp Using CE-158	28
Delinger	Willian	Hexadecimal Loader	28
Doughty	Russell	Histogram on PC-2/PC-1500	21
Etheridge	Steven	Fractional Base Conversion	28
Fieux	Robert	Using the CE-158 Interface	28
Frey	Michael	RPN Calculator for Programmers	26
Hergert	David	Meteors Game for PC-2/PC-1500	28
Hergert	David	Rabbit Chase	30
Jackson	David	Expanded Display Routine	23
Motto	David	Epson HX-20 Review	21
Motto	David	Epson HX-20 Follow-Up	23
Norton	John	Graphing on PC-2/PC-1500	24
Norton	John	BASIC Line Deletion	24
Organ	R.	Fuel Consumption Analyst	29
PCN	Staff	Radio Shack Announces the PC-3	27
PCN	Staff	Sharp Steps Up In Size	26
PCN	Staff	The Sharp CE-125 Printer/Microcassette Interface	23
PCN	Staff	Radio Shack Offers PC-4	23
PCN	Staff	The Sharp PC-1251	22
PCN	Staff	Epson HX-20 Weather Forecaster	22
PCN	Staff	Texas Instrument's CC-40	22
PCN	Staff	Sharp Introduces Wallet-Size PC	21
PCN	Staff	Radio Shack Micro Color Computer	25
PCN	Staff	TRS-80 Model 100 Sets Trend	25
PCN	Staff	The CE-158 Interface	24
PCN	Staff	PC-1250/51 Memory Dump Program	24
PCN	Staff	Radio Shack TRS-80 Model 100	24
PCN	Staff	Cassette Volume Monitor	27
PCN	Staff	Paper Holder Kit for PC-1500 & PC-2	27
PCN	Staff	Memory Maps	23
PCN	Staff	Casio Software Modules	28
PCN	Staff	Casio FX-802P Includes Printer	28
PCN	Staff	Gavilan Readying Mobile Computer	28
PCN	Staff	Alternate Control of the PC-1500 LCD	28
PCN	Staff	Hewlett-Packard HP-41CX	30
PCN	Staff	Carlton 16K EPROM Modules	30
PCN	Staff	Alternate Control of the PC-1500 LCD	29
PCN	Staff	Casio FX-700P Product Review	29
PCN	Staff	RoadRunner Portable Computer	29
PCN	Staff	Alternate Control of the PC-1500 LCD	30
Pollett	Patrick	Redefining the PC-1500 Keyboard	29
Rober	Norlin	The CE-502A General Statistics Module	26
Rober	Norlin	Inside the Sharp PC-1250	26
Rober	Norlin	Machine-Language Monitor for the PC-1500/PC-2	22
Rober	Norlin	Normal Probability Using Machine Language	25
Rober	Norlin	PC-1500/PC-2 Cassette Storage	25
Rober	Norlin	Understanding PC-1500 Part 4	21
Rober	Norlin	Three-Dimensional Plotting	21
Rober	Norlin	Three-Dimensional Projections	27
Rober	Norlin	Vertical Lister	27
Stock	Robert	RS-232C Interface Print Formatter	28
Thomas	Arthur	Date & Time on the PC-2/PC-1500	28
Tonback	Stephen	File Maintenance Program	24
Tonback	Stephen	Enhanced File Maintenance	30

1983 Index -- Ordered By Article

Title	Last Name	First Name	Issue
26K of RAM in the PC-1500/PC-2	Carter	Don	23
Alternate Control of the PC-1500 LCD	PCN	Staff	28
Alternate Control of the PC-1500 LCD	PCN	Staff	29
Alternate Control of the PC-1500 LCD	PCN	Staff	30
BASIC Line Deletion	Norton	John	24
Carlton 16K EPROM Modules	PCN	Staff	30
Casio FX-700P Product Review	PCN	Staff	29
Casio FX-802P Includes Printer	PCN	Staff	28
Casio Software Modules	PCN	Staff	28
Cassette Volume Monitor	PCN	Staff	27
Controlling Lamp Using CE-158	Delinger	Willian	28
Count-Down Timer Program	Auersbacher	Emerich	27
Cross Reference for the PC-2	Campbell	Diane	27
Date & Time on the PC-2/PC-1500	Thomas	Arthur	28
Enhanced File Maintenance	Tonback	Stephen	30
Epson HX-20 Follow-Up	Motto	David	23
Epson HX-20 Review	Motto	David	21
Epson HX-20 Weather Forecaster	PCN	Staff	22
Expanded Display Routine	Jackson	David	23
File Maintenance Program	Tonback	Stephen	24
Fractional Base Conversion	Etheridge	Steven	28
Fuel Consumption Analyst	Organ	R.	29
Gavilan Readying Mobile Computer	PCN	Staff	28
Graphing on PC-2/PC-1500	Norton	John	24 25
Hewlett-Packard HP-41CX	PCN	Staff	30
Hexadecimal Loader	Delinger	Willian	28
Histogram on PC-2/PC-1500	Doughty	Russell	21
Inside the Sharp PC-1250	Rober	Norlin	26
Machine-Language Monitor for the PC-1500/PC-2	Rober	Norlin	22
Memory Maps	PCN	Staff	23
Meteors Game for PC-2/PC-1500	Hergert	David	28
Normal Probability Using Machine Language	Rober	Norlin	25
Paper Holder Kit for PC-1500 & PC-2	PCN	Staff	27
PC-1250/51 Memory Dump Program	PCN	Staff	24
PC-1500/PC-2 Cassette Storage	Rober	Norlin	25
Rabbit Chase	Hergert	David	30
Radio Shack Announces the PC-3	PCN	Staff	27
Radio Shack Micro Color Computer	PCN	Staff	25
Radio Shack Offers PC-4	PCN	Staff	23
Radio Shack TRS-80 Model 100	PCN	Staff	24
Redefining the PC-1500 Keyboard	Pollett	Patrick	29
Renumbering Utility	Bean	Norman	29
RoadRunner Portable Computer	PCN	Staff	29
RPN Calculator for Programmers	Frey	Michael	26
RS-232C Interface Print Formatter	Stock	Robert	28
Sharp Introduces Wallet-Size PC	PCN	Staff	21
Sharp Steps Up In Size	PCN	Staff	26
Sidelister Program	Beckman	Mel	22
Star Targets	Cloutier	David	29
Texas Instrument's CC-40	PCN	Staff	22
The CE-158 Interface	PCN	Staff	24
The CE-502A General Statistics Module	Rober	Norlin	26
The Sharp CE-125 Printer/Microcassette Interface	PCN	Staff	23
The Sharp PC-1251	PCN	Staff	22
Three-Dimensional Plotting	Rober	Norlin	21
Three-Dimensional Projections	Rober	Norlin	27
TRS-80 Model 100 Sets Trend	PCN	Staff	25
Understanding PC-1500 Part 4	Rober	Norlin	21
Using the CE-158 Interface	Fieux	Robert	28
Vertical Lister	Rober	Norlin	27
Wind Chill Equivalent Temp.	Auersbacher	Emerich	21

determine mortgage rates or calculate consumer loans. Still another organization supplies heating contractors with a PC-1500 system programmed to solve heating and air conditioning problems, as well as deliver a sales pitch to prospective customers. The sales pitch generates charts and graphs illustrating how much money a new energy-efficient heating/cooling system can save over its estimated lifespan. I have even heard about an ambitious project that will help airline pilots keep on course as they fly all over the country. A PC-1500, programmed with all kinds of technical data, can calculate the best flight plans, advise on mid-flight course corrections, and serve as a general navigation aid. There are people working on the same type of capability for mariners, too.

Many of these systems will appear on the market packaged so that the actual users will hardly even know that what they are really using is just a specially programmed PC. In other words, the PC is becoming an OEM instrument. It is being used as the vehicle in which special-purpose vendors simply package their product. After all, their real product is simply the software that gives the PC the ability to perform a particular type of task.

The Trend Is Definitely Towards Bigger!

There is not doubt that people want portability. But, pocketable units are having a hard time keeping up with the expectations and demands of the public. People are willing to pay for full size keyboards and larger displays. The appetite for more memory is probably insatiable. Witness the success of notebook-sized units such as the Radio Shack Model 100 and Epson HX-20. There will be lots more new entries of this approximate size during the coming year.

Why, even Sharp may forget what a PC is in its rush to push its new PC-5000. (By the way, I hear the unit is now finding its way into stores in the United States. Check with your local Sharp dealer if you have been anxiously waiting to examine an actual unit. Price of a standard 128K unit is reportedly $1995.00. Add about $400.00 if you want the optional integrated printer.)

Looking For A/D?

A firm by the name of **Enertec**, *558 Highland Ave., Upper Montclair, NJ 07043*, has developed an interface for the PC-1500 & PC-2 that includes: 8 digital inputs, 8 digital outputs, and an A to D converter with 8 analog inputs. The company developed the interface for a special project. However, it is considering manufacturing the unit for general distribution. Write to *Paul Schwartz* at the company address or phone (201) 744-4962 and speak to him if you would like to see such a product on the market.

PCN To Stay With PCs.

During 1984 we plan to continue devoting the bulk of our space to the most popular "pocketable" units, primarily the Sharp PC-1500/1500A and Radio Shack PC-2. Of course, we will follow new entries and keep an eye on the smaller notebook portables.

If you have been wondering what happened to our popular feature PC-1500 writer and internationally respected PC whiz **Norlin Rober**, never fear, he has been hard at work compiling a massive report on the PC-1500's ROM. We plan to provide an ongoing comprehensive series revealing all its secrets during the coming year, thanks to Norlin's monumental efforts.

There are lots of other goodies planned too, such as the powerful Little EDitor (LED) program that will be featured in the next issue. This is really a line oriented word processor program that is sure to thrill and amaze many PC-1500/PC-2 owners.

We Have Come A Long Way

Current editions of *PCN* are now four times larger than when we first started publishing. We are most definitely rather large for a "newsletter." I wish to extend my personal thanks to all our faithful subscribers who have helped make this growth possible. Best wishes for a healthy and prosperous New Year to everyone!