POCKET COMPUTER NEWSLETTER

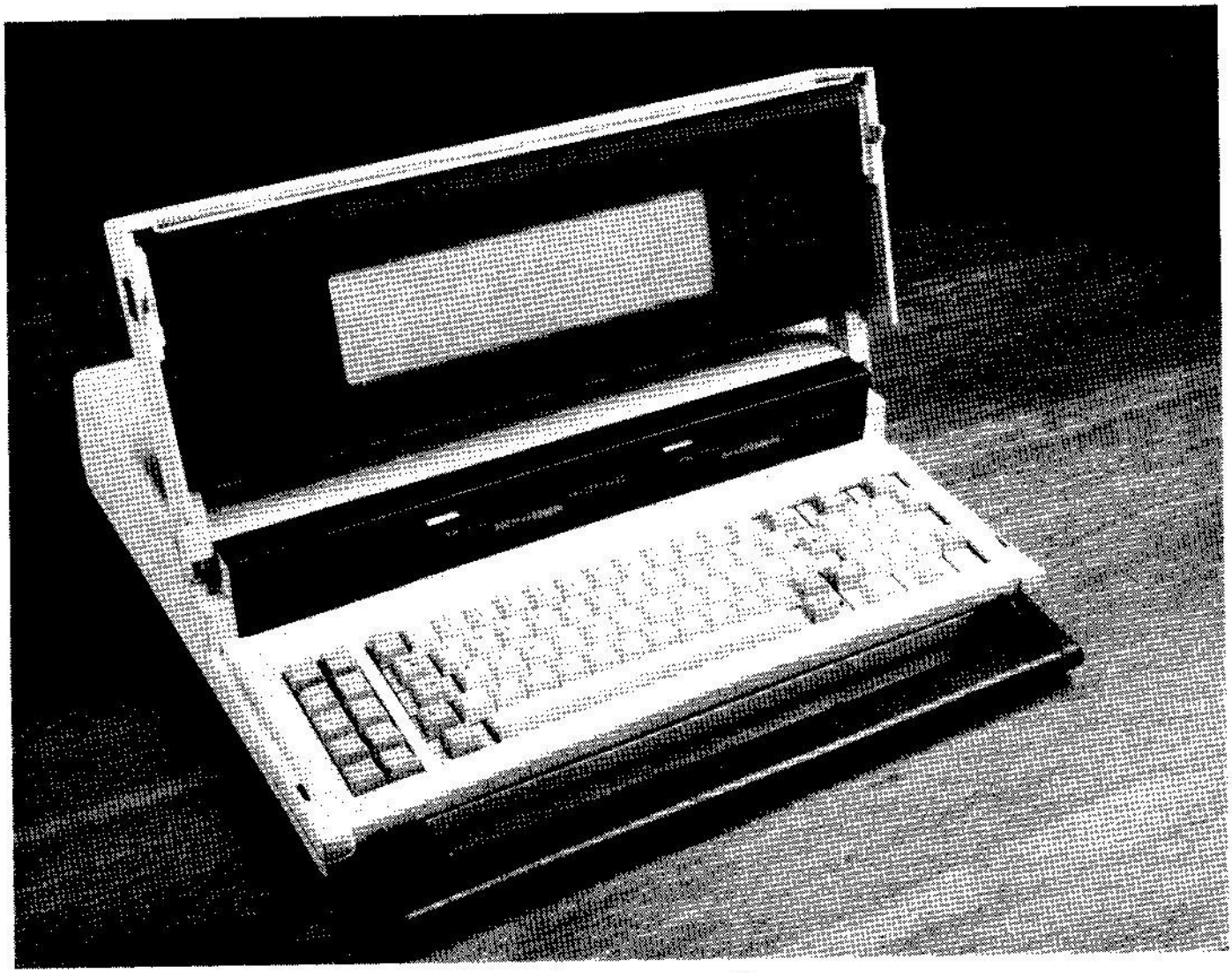
Issue 31 - January/February

NEW PORTABLE DUBBED "THE COMMUTER"

Visual Computer Incorporated of Marlboro, Massachusetts, has announced the design of a new IBM-compatible personal computer that may provide direct competition to the recently released Sharp PC-5000.

The new briefcase-sized portable sports a 16-bit 8088 microprocessor, 128K of memory with parity-checking (expandable to 512K), a 5-1/4 inch floppy disk drive (expandable to two drives), an 83 key low profile keyboard which includes function keys, a numeric pad and layout identical to IBM.

The machine measures 18 by 15 by 3 inches and weighs 16 pounds. It is designed to operate from 110- or 220-volt lines.



The basic unit is supplied without a display. It contains support logic for an 80 x 25 or 40 x 25 display and an RF connector for a monochrome or color TV monitor. It is claimed to be compatible with IBM bit map color graphics.

As an option, The Commuter may be supplied with an 80 column by 16 line LCD display at a cost of $495.

The unit includes a parallel port for connecting to a letter quality or dot-matrix printer and a serial asynchronous RS-232C port. A connector is also provided for expanding the computer using an IBM expansion chassis.

The Commuter will use MS-DOS (2.1) as an operating system. Software for the IBM PC is said to be compatible with the machine.

For additional information contact *Visual Computer Inc., 135 Maple Street, Marlboro, MA 01752.*

THE SHARP PC-5000 PORTABLE COMPUTER

The new Sharp PC-5000 Portable Computer has finally arrived. Sharp has certainly packed a lot into this relatively small package. The machine really has the power of a conventional desktop-sized computer. However, it is all compressed into a portable unit the measures close to a 13 by 12 by 3-1/2 inches. When you read about all that can be fit within these short dimensions, I think you too will be impressed.

The basic unit without options weighs just 12 pounds. However, there is room within the case of the computer to install a printer and a modem. The addition of these options can add another 4 to 5 pounds.

The front cover of the PC-5000 lifts to reveal a full typewriter-style keyboard containing 72 keys. Eight of these keys are user definable "softkeys."

The underside of this cover contains an 80-column by 8-line liquid-crystal display (LCD). The 640 by 80 pixels on this LCD can be used to display text or graphics. There are 128 text characters/symbols and and equal number of graphic elements available for use. The angle of the cover can be adjusted to one of several possible positions to facilitate comfortable viewing. In addition, an LCD contrast control allows the user to customize the contrast of the display to compensate for actual lighting conditions.

Memory Everywhere

The PC-5000 is loaded with memory. However, because of its sophisticated modes of operation, actual user RAM appears to be used up rapidly!

There are 192 kilobytes of internal ROM installed. This apparently stores the MS-DOS operating system, various system utility routines and part of the highly sophisticated Microsoft BASIC interpreter. Somewhat surprisingly, there is another separate 64 kilobyte ROM containing much the the Microsoft BASIC interpreter. It resides in an expansion slot and usurpes 64 kilobytes of RAM address space. Furthermore, in order to use the BASIC package, you need to have a bubble memory unit plugged in for use as a mass storage device. Now this seems like an awful lot of memory just to run BASIC. However, you may have never seen the likes of this kind of BASIC before. It is undoubtably one of the most powerful dialects of the language available on anything short of a large scale mainframe computer! More about it later.

Perhaps the most intriguing aspect of the PC-5000 is its use of 128 kilobyte bubble memory cartridges. These devices measure about 3 by 2-1/2 by 3/8 inches. They plug directly into a slot on the top of the unit. (The slot is equipped with a flip-up/down cover so the cartridge is invisible when not being inserted or removed.) These bubble memory cartridges are treated exactly as though they were a combination of a floppy disk and its disk drive! In fact, when using the MS-DOS operating system, the bubble memory is treated as drive A (or, by swapping bubble memory cartridges, as drive B). The bubble memory modules are small and slim. They fit easily, for instance, into a shirt pocket. Thus, you can carry several with you in lieu of floppy disks and their associated drives. (But, if you want to use floppies, you can purchase a 5-1/4 inch dual-drive unit as an external option.)

Now the list price of these 128 kilobyte bubble memories is currently rather steep: $269.00 U.S. list. However, you receive one with the PC-5000. It is supplied as part of a software package that includes a text editor program called SuperWriter, published by Sorcim. (Sorcim is a firm that has been producing commercial software for desktop computer. The firm is now planning to provide software support for a number of portable computer systems.)

The module containing Super- Writer has about 24K left over on which you can store your own programs and/or data. However, since 24K can be used up rapidly on a system such as this, you may want to consider obtaining one or two more bubble cartridges to use for storage of your programs and data. (While this may seem expensive when compared to purchasing a couple of floppy disks, remember their use saves the initial costs of the disk drives!)

While bubble memory cartridges cost quite a

bit at the moment, their increasing use should drop the price dramatically in coming years. It would not be surprising to see 128 kilobyte bubble cartridges selling in the range of $50 - $75 within a few years.

When you add it all up, you find that the little PC-5000 comes equipped with an astounding 512 kilobtyes of combined memory, distributed as follows: 192K system ROM, 64K BASIC ROM, 128K user RAM and 128K bubble memory. That is a lot of memory to pack into a portable unit.

Where Does It All Go?

At first, this amount of memory sounds like a staggering sum. However, it turns out that much of this memory is already dedicated to system uses. Thus, the 192K of system ROM and 64K ROM module devoted to BASIC consumes half of the 512K total. Now it turns out that the version of BASIC utilized can only address a maximum of 64K of memory. Close to half of this address range is apparently devoted to the BASIC operating system. The net result of all this is that you are limited to a maximum of approximately 35 kilobytes of user program/data storage when running BASIC. This is nothing to sneeze at (especially if you are accustomed to squeezing programs into a pocket computer!), but it seems a far cry from making use of that nice 128 kilobytes of RAM that is sitting right there in the machine!

On the other hand, the extremely powerful version of BASIC provided, along with the operating system, allows virtually unlimited program "chaining" and "overlay" capability. Since this is accomplished with the bubble memory unit, complex program overlays take but a second or two. Thus, you can construct huge programs as long as program and/or data blocks are held under about 35K at a time!

There are similar types of limitations when using other software packages. Out of the 128K of system RAM provided in the PC-5000, you can count on a significant portion of this being used by the operating system and/or applications package. For instance, when the word processor Super-Writer is loaded into memory (from the bubble cartridge), you are informed that the text buffer has space for about 24,000 characters. That is the equivalent of about 12 pages of ordinary text. Not an awful lot if you are writing a book. In fact, it seems like a rather chintzy portion of that 128K of RAM. It turns out, however, that the amount of memory available for a text file is limited by the amount of file storage (bubble memory or disk space). About 24K is what is left on the bubble memory that is used to supply the SuperWriter program. If you plug in another bubble memory or use the disk version, you can have much larger text files.

In summary, don't make the assumption that the amount of memory supplied with the unit is a lot more than you will ever need. In fact, its rapid consumption by application programs can be downright disconcerting to beginning users. Chances are you will eventually want to add at least a few more bubble memory units for storage of files or expand to the floppy disk system.

What A BASIC

A customized version of Microsoft BASIC is provided with the PC-5000. It is one of the most powerful, flexible implementations of the BASIC language that you are likely to see short of a large mainframe installation. Just about the only short-coming is the poor instruction manual that covers this part of the machine's operation.

To give you an idea of the extent of this BASIC's power, here is an alphabeticized list of the reserved words associated with it: ABS, AND, ASC, ATN, AUTO, BEEP, BLOAD, BSAVE, CALL, CDBL, CHAIN, CHR$, CINT, CIRCLE, CLEAR, CLOSE, CLS, COLOR, COM, COMMON, CONT, COS, CSNG, CSRLIN, CVD, CVI, CVS, DATA, DATE$, DEF, DEFDBL, DEFINT, DEFSNG, DEFSTR, DELETE, DIM, DRAW, EDIT, ELSE, END, EOF, EQV, ERASE, ERL, ERR, ERROR, EXP, FIELD, FILES, FIX, FN, FOR, FRE, GET, GOSUB, GOTO, HEX$, IF, IMP, INKEY$, INP, INPUT, INPUT#, INPUT$, INSTR, INT, KEY, KILL, LEFT$, LEN, LET, LINE, LIST, LLIST, LOAD, LOC, LOCATE, LOF, LOG, LPOS, LPRINT, LSET, MERGE, MID$, MKD$, MKI$, MKS$, MOD, MOTOR, NAME, NEW, NEXT, NOT, OCT$, OFF, ON, OPEN, OPTION, OR, OUT, PAINT, PEEK, PLAY, POINT, POKE, POS, PRESET, PRINT, PRINT#, PSET, PUT, RANDOMIZE, READ, REM, RENUM, RESET, RESTORE, RESUME, RETURN, RIGHT$, RND, RSET, RUN, SAVE, SCREEN, SGN, SIN, SOUND, SPACE$, SPC, SQR, STEP, STOP, STR$, STRING$, SWAP, SYSTEM, TAB, TAN, THEN, TIME$, TO, TROFF, TRON, USING, USR, VAL, VARPTR, VARPTR$, WAIT, WEND, WHILE, WIDTH, WRITE, WRITE#, and XOR!

That, my friends, is a powerful implementation that offers something for almost everyone. Yes, you can access serial and random files, do almost anything you want in terms of controlling the LCD screen, call machine language routines, write structured programs, etc. Many of the commonly used keywords are available at the stroke of a single key by using the ALTernate button. (This feature can save a lot of time if you are not a skilled typist.) Furthermore, installed in the PC-5000, this version of BASIC is *fast!*

After perhaps 30 - 40 hours of programming

with this version of BASIC, I have the following complaints: 1. The instruction manual provided with the PC-5000 for this BASIC was obviously done on a "crash" basis. While there is about 150 pages of information describing the instruction set, it looks as though it was literally thrown together with not so much as a single proof-reading. There are countless typographical errors, sections are rather incomplete and difficult to understand, and sometimes missing altogether. For instance, the PRINT# statement, which is crucial to writing to external files, is referenced in parts of the manual (so that you know it is available), but is never presented. Hopefully, Sharp will institute a "crash" program to get this portion of their documentation upgraded to the level of the quality of the machine itself. (It should be noted that some of the other documentation supplied with the unit, especially the SuperWriter material, is excellent.) 2. The RENUMber command seems to lack a facility normally provided - that of being able to renumber just a specific section of a program. This seems to be an oversight also, as you are permitted to start renumbering anywhere in a program. You just cannot specify an ending line number. In other words, once it starts renumbering, it does so until it reaches the end of the program.

In summary, however, this is a powerful, extended version of BASIC that is sure to please most users, especially if future manuals are upgraded so that all of its many features are clearly explained.

SuperWriter

The PC-5000 is supplied with an applications overlay menu that is designed to support several programs designed by Sorcim. You receive one of these programs (supplied on a bubble memory cartridge) with the unit. The program is a text editor or word processor named *SuperWriter*. It is good. And, the manual for it is also good.

I am not going to go into a lot of detail on the operation of SuperWriter, other than to say that it is a powerful, flexible, menu-driven, text editor. It permits formatting of the output (including full justification, pagination, footers, etc.), movement of text, merging operations, and so forth. The program is supplied with a comprehensive manual plus a short "ten-minute tutorial" booklet and a handy pocket reference card. If you write letters, reports, articles, even books, you will find a lot of use for this program. It, as all other application programs do, will undoubtably have idiosyncracies that may annoy you slightly. But, all-in-all, it will get the job done for most people.

The Applications System

The SuperWriter program is just one in an optional integrated set of programs that may be obtained for the PC-5000. An understanding of how these programs can interrelate is important to gaining a full appreciation of the true power provided in this portable machine.

Several other application packages may be purchased as options. Currently, these include: SuperCalc, SuperPlanner and SuperComm. As you might surmise, SuperCalc provides spreadsheet capabilities. SuperPlanner provides appointment and database functions. SuperComm is for managing communications when using a modem.

Now having all of these capabilities available in a portable unit might be enough in itself. But wait until you hear the rest.

Access to each of these programs is provided through a menu system. For instance, if you want to use SuperCalc, you make sure a bubble memory containing the program is installed and select the desired menu option. (If the correct memory module is not installed, you will be so-informed by the menu program.) Once the SuperCalc program is on-line, new menus enable you to select the operations you currently desire. One of the options provided is the ability to copy data from a spreadsheet (or other Super application programs) into what Sharp calls "scratch-pad" memory.

You guessed it! When you switch from one Super applications program to another, the scratch-pad memory is not altered. Thus, you can transfer information from a spreadsheet into a SuperWriter text file! Or, from a SuperComm file into SuperWriter. And, vice-versa. Do you get the picture? The system allows integration of data between application programs! Now that is Super!

MS-DOS

Files under Super applications programs, BASIC, and presumably other language options in the future, all run under the built-in Microsoft-DOS system. This is a powerful file-handling and directory system that is accessible to the user. You can create files (on external memory devices such as bubble memories of floppy disks), examine their contents, duplicate, merge, delete, rename, etc. You can also create hierarchal directories and these can be nested.

Once again, the system is powerful. My only complaint in this area is the poor documentation. They provide a manual, but it looks as though it was thrown together in the greatest haste. If Sharp doesn't rewrite this portion soon, someone else may make a fortune producing supplemental manuals for the PC-5000.

Other Features

The tape cassette interface is relatively fast. You

can load an 8K BASIC program in about one minute. (It would take about 10 minutes to load the same amount into a PC-1500.) This means that the tape interface is a practical way to archive programs and data. You can keep your working copies on a bubble, knowing that should something ever happen, you won't have to take half-a-day to restore the information from audio tape cassettes. It may also be a good way for people to distribute programs. By the way, at least when using BASIC, the PC-5000 provides an option of writing programs to external devices, such as via the tape cassette interface, that cannot be listed or edited, only executed.

The optional printer ($395.00 extra) is amazing. In the first place, it is truly quite. It is the first printer I have ever seen on a portable computer that I would not be embarrased to operate in, say, a public library. However, it has its limitations and is not for everyone. It is relatively slow. It is designed to be sheet-fed. A long report means many stops to insert fresh sheets of paper. It requires special thermal paper or the use of special (expensive) ribbons when used with regular paper. If you need to obtain hardcopies of information when you are on the road, the printer is a must. However, if you can wait until you get back to your base of operations in order to get hardcopies, you might be better-advised to select another alternative: purchase a separate external printer and drive it via the PC-5000's built-in RS-232 serial interface!

The PC-5000 does not use nickel-cadmium batteries. It uses a special, sealed, lead-acid package that is surprisingly small. This will power the unit for about 8 hours of operation if the optional printer is not used. Using the printer considerably reduces the battery life. A warning light comes on when the battery needs recharging. A recharger, supplied with the PC-5000, may be used to continuously power the unit, thereby keeping the battery continuously recharged until needed for field-portability.

Looks Like A Winner

All-in-all, the PC-5000 looks like a lot of value, at this time, for its $1995.00 list price. While it will surely face stiff competition in the future, the fact that it is one of the first out of the gate will surely be an important factor in the race to gain popular acceptance. It contains a lot of memory, a powerful version of BASIC, the popular Microsoft operating system, and a menu-driven means of integrating a line of optional application packages. Right now it appears to be the lowest-priced unit with such capabilities. Put all that together and it sure looks like a potential winner!

LITTLE EDITOR

H. David Jackson, 126 Smithfield Drive, Endicott, NY 13760, provides this version of a text editor for PC users.

Introduction

LED is the abbreviation I have assigned to a program called the Little EDitor. This program may be used to input data sets and later edit, alter or delete that data as desired. Although it is small, it still contains many of the functions found in editors that run on large mainframe computers. It is somewhat slow because it is written in BASIC. But, what it lacks in speed it makes up for in function. The use of BASIC also allows the user to easily add to, delete or modify its operation.

LED is considered user friendly as commands are entered under function key control rather than having to type in a directive. Only data and certain parametric information, such as data set names, have to be typed in.

This program was developed on a Radio Shack PC-2 with an 8K memory module. A 4K module can be used, but the maximum size of the data set that could be handled will be considerably reduced. The program requires a printer/cassette interface for saving, restoring and printing the data sets.

Data Set Size

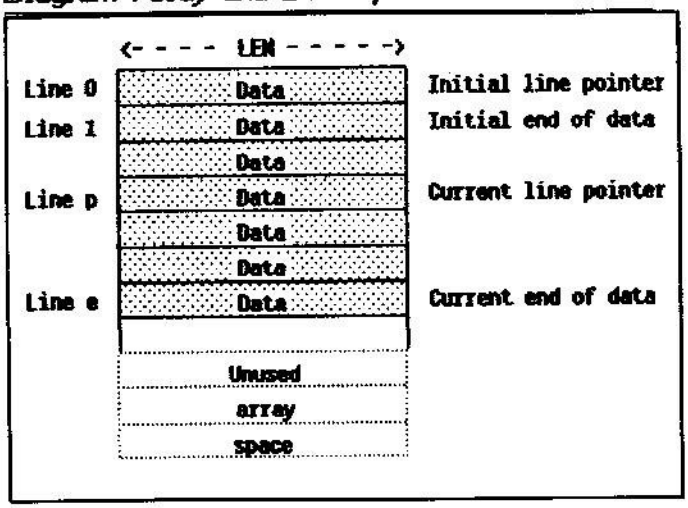
The data set size is set by the user immediately upon entry to the program. A variable called NUM will be requested. This value is the maximum number of lines that will be entered. The largest value that may be used is 255. LEN will then be requested. This is the maximum length of the lines that will be used. The largest allowable value here is 80. These two variables are used to dimension an array that stores the data. Plan your data set size carefully as the array cannot be re-dimensioned without losing data already stored. Also, the entire array is saved on tape when requested, no matter how much or little data has actually been entered. This takes some time if maximum limits have been chosen and wastes time if the space is not needed. Of course, it is not necessary to fill all lines reserved. The array can contain unused space.

The size of the array (NUM times LEN) cannot exceed the amount of storage available. If it does, an error message will be displayed. Use the MEM command prior to running the program to see how much memory is available. With an 8K module there will be approximately 7500 bytes, 3400 bytes with a 4K module, provided that memory is not

allocated for other purposes.

Regardless of the array size, internal pointers to the current line being entered or edited, as well as the last line in the data set, are maintained by the LED program. These are initially set to zero and one respectively. Using the end of data (EOD) allows the line pointer to be more easily set to selected locations without having to traverse large amounts of unused lines. Thus, the initialized data set is two lines long. Additional lines may be added as needed up to the extent of the array size (NUM) that was originally specified.

Diagram *Array and Data Space Allocations*



Data Display and Modes

The amount of data that is seen at one time is limited to the 26 characters of the display screen. The length of a line, however, can be up to the LEN orginally specified. In order to accomodate this, two types of displays are used. The first is the input display. This is used when the program has been placed in the input mode. It is denoted by the presence of a question mark (?) at the left of the screen. In this case, the normal responses and editing associated with the use of the INPUT statement in BASIC apply. The display scrolls left one character at a time when the right end of the screen is reached.

The second type of display is associated with the edit mode. It is denoted by a flashing cursor. While in this mode, all of the function keys that will be described later are operational. Key usage in this mode is a little different. Various modes are used to tell the program what to do. The current mode(s) are indicated by the cursor. These modes are summarized as follows:

SHIFT MODE - This mode is similar to the regular shift key function used in the INPUT section. It, however, also modifies the commands associated with some of the cursor movement keys. This mode is entered by pressing the SHIFT key. If the current line is null (contains no data), then the program will enter the input mode (a ? appears on the left of the screen). This allows data to be entered faster than when in the edit mode. The program will remain in this mode for each line entered until a non-null line is found or the ENT key is pressed without entering any data. In either case, the program will revert back to the edit mode.

ALTERNATE MODE - This mode also changes the command that is associated with function and cursor movement keys as described later. This mode is set by pressing the Reserve mode change (up and down solid triangle) key.

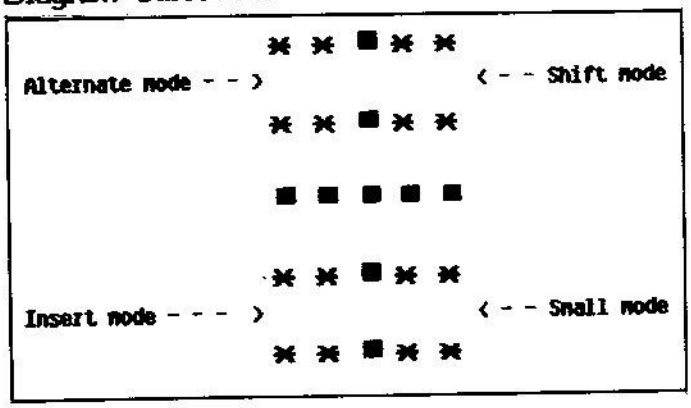
(The above two modes are automatically reset after another key is pressed. That is, they are only in effect for one key stroke.)

INSERT MODE - This mode indicates that characters are to be inserted at the cursor position instead of overlaying those that are already there. This mode is toggled by pressing the MODE key.

SMALL MODE - As the name implies, characters that are overlaid or inserted will be in lower case. This mode is toggled by pressing the SML key.

The cursor is used to indicate the current mode status. This is done by placing small blocks in each corner of the cursor. If just a flashing character (or lone underline when the position is blank) is observed, then none of the above modes are set. The accompanying diagram shows which corner blocks are active for the various modes. Note that more than one mode may be set at a time.

Diagram *Cursor Mode Indicators*



In addition, when in the edit mode, the cursor and display positioning is slightly different than that used by the input mode. When the cursor leaves the right side of the display in the edit mode, it will be positioned to the middle of the display instead of just scrolling left one. This gives the user a more complete view of the line being edited and allows one to see data on both sides of the

cursor. The same occurs, when moving to the left, unless the cursor is in the first position.

Normal Function Keys

The following functions or commands are executed by pressing the specified key when editing in the normal mode. A beep indicates that the depression of a function key (F1 - F6) has been recognized by the program.

F1 - LOAD: This command will load a data set that was previously stored on tape. It is necessary to position the tape to the proper spot. Then enter the name of the data set that is to be recovered. The program restores the array parameters that were saved along with the data. Pressing the ENT key by itself causes the command to be ignored.

F2 - ADD: This command will add a single null line immediately after the line on which the cursor currently resides. If you listen carefully, an audio click may be discerned, indicating that one line has been added.

F3 - CLEAR MARKS: This command will clear any marks that may have been placed on selected line(s) of data.

F4 - MARK DATA: The first press of this key will *mark* the line at the cursor position for later reference. This line will be highlighted by reverse video. A second press will mark a group of lines, all of which will be denoted by reverse video. The group comprises all lines between and including the marked lines. A marked line or group of lines may be moved, deleted or printed. Marks may also be used to limit changes to a selected line or group.

F5 - MOVE: Pressing this key causes the marked area to be moved after the line on which the cursor is currently located. The marked data is not erased. Thus, this directive may be used to duplicate information at various locations. (To erase data see alternate F2.)

F6 - FIND DATA: This key is used to indicate that a word or phrase is to be searched for in the data beyond the current cursor or in a marked area. If located, that line will become the current line in the display. If the search is unsuccessful, the line pointer remains the same.

Alternate Function Keys

The following functions will be executed upon pressing the specified key while editing in the alternate mode.

AF1 - SAVE: This command will save the data set and associated parameters. You will have to position the tape, make sure you have set the tape unit to the record mode, and enter a file name for the data set. Pressing the ENT key without other input will cause the command to be ignored.

AF2 - DELETE: This command causes the line that the cursor is on or a marked group of lines to be removed from the data set. A low tone is issued for each line that is deleted.

AF3 - DUPLICATE: This key causes the line that the cursor is on to be duplicated immediately after the line.

AF4 - Not used.

AF5 - PRINT: The entire data set or a marked group of lines is printed. Prior to outputting of the data, the program will prompt for the CSIZE desired by the operator.

AF6 - CHANGE DATA: A global change function. The system prompts for *from* data and *to* data. All cases of the former following the cursor to the end of the data set or within a marked group are modified to the latter. A null *from* entry causes the *to* data to be inserted at the start of each line. A null *to* entry will cause the *from* data to be removed.

Shifted Function Keys

If the shift mode is on, the function keys F1 - F6 will react as though in the input mode. That is, the characters !, ", #, $, % and & will be entered. The shift key is also used to enter @, >, <, ?, ;, : and comma as indicated on the keyboard.

The shift key is additionally used to activate the CLR key. This combination causes the data array to be re-initialized.

If the OFF key is pressed while in the shift mode, the program is terminated.

Cursor Control Keys

The cursor control keys perform in the following manner:

The right arrow causes the cursor to move one character position to the right.

Shift right arrow causes the character beneath the cursor to be deleted. The cursor remains at the same position.

The left arrow causes the cursor to move one character position to the left.

Shift left arrow causes the character to the left of the cursor to be eliminated. The cursor also moves left one position.

The up arrow causes the line pointer to move one line closer to the start of the data set (-1).

Alternate up arrow causes the line pointer to be moved five lines closer to the start of the data set (-5).

Shift up arrow causes the line pointer to be set to the start of the data set. (Note that this cannot be performed on a null line as the shift will cause the program to go into the input mode.)

The down arrow moves the line pointer one line closer to the end of the data set (+1).

Alternate down arrow causes the line pointer to

move five lines closer to the end of data (+5).

Shift down arrow sets the line pointer to the end of data. (Again, this operation cannot be used while on a null line.)

Additional Key Capabilities

The following keys also assist during editing operations:

The RCL key adds five lines after the current line.

Alternate RCL adds 10 lines.

Shift RCL adds 20 lines.

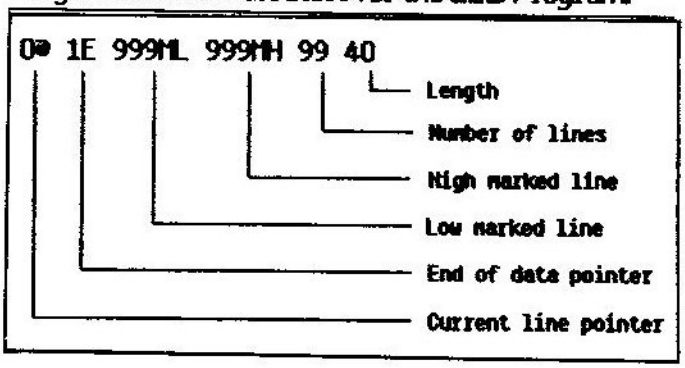
A fast audio click announces the addition of each line.

The DEF key displays the status of all the important pointers. This display uses the format illustrated in the accompanying diagram. (The diagram assumes that the data set initially had an array size whereby NUM=99 and LEN=40. The value 999 in the marked lines positions indicates that no lines are marked.)

Program *Little Editor.*

```
  4:CLEAR :INPUT "
    NUM-";N:INPUT
    "LEN-";O
  5:DIM M$(N)*O:A=
    1
 10:L=999:M=999:
    WAIT 0:C=1:P=1
 20:POKE 18409,72,
    118,74,0,5,189
    ,255,65,78,78,
    153,8
 30:POKE 18421,76,
    119,139,6,72,1
    19,74,0,158,18
    ,154
 50:D=0:E=0
 55:IF B<=0LET B=0
 60:IF B>=ALET B=A
 70:Y=LEN M$(B):IF
    Q>25LET P=P+13
    :Q=Q-13
 80:IF Q<0AND P>1
    LET P=P-13:Q=Q
    +13
 90:IF Q<0LET Q=0
 95:IF (P+Q)>YLET
    Q=Y-P+1
100:C=P+Q:PRINT
    MID$ (M$(B),P,
    26)
110:IF B>=LAND B<=
    MCALL 18409
120:GCURSOR 6*Q:IF
    D+E+F+G+((MID$
    (M$(B),C,1)="
    ")OR C>Y)=1
    THEN GPRINT "4
    040404040":
    GOTO 140
130:GPRINT E+G;E+G
    ;0;D+F;D+F
134:IF (Y=0)AND (D
    >0)GOTO 4000
140:K=ASC INKEY$ :
    IF K=0GOTO 100
145:IF K<>32AND K<
    40GOTO 200
150:IF D=0GOTO 170
155:IF K=61LET K=6
    4
160:IF K=32LET K=9
    4
165:IF K>39AND K<4
    8LET K=ASC (
    MID$ ("<>:;,,.
    ?",K-39,1))
170:IF K>64AND K<9
    1AND F>0LET K=
    K+32
175:IF C>YLET M$(B
    )=M$(B)+CHR$ K
    :GOTO 185
180:M$(B)=LEFT$ (M
    $(B),C-1)+CHR$
    K+RIGHT$ (M$(B
    ),Y-C+(G>0))
185:Q=Q+1:GOTO 50
200:IF K=12AND D=0
    LET Q=Q+1:GOTO
    50
210:IF K=8AND D=0
    LET Q=Q-1:GOTO
    50
220:IF K=31LET G=1
    12*(G=0)
225:IF K=2LET F=11
    2*(F=0)
230:IF K=1LET D=7*
    (D=0):GOTO 55
235:IF K=9LET E=7*
    (E=0):GOTO 55
240:IF K=13LET B=B
    +1:P=1:Q=0
245:IF K=10LET B=B
    +1+(4*(E>0)):B
    =B+((A-B)*(D>0
    ))
250:IF K=11LET B=(
    B-1-(4*(E>0)))
    *(D=0)
255:IF K=2SLET Y=5
    +(5*(E>2))+(15
    *(D>0)):GOSUB
    3000
260:IF K>16AND K<2
    3AND D>0LET K=
    K+16:GOTO 175
275:IF K=12LET M$(
    B)=LEFT$ (M$(B
    ),C-1)+RIGHT$
    (M$(B),Y-C+(C>
    Y))
280:IF K=8LET M$(B
    )=LEFT$ (M$(B)
    ,C-1-(C<>1))+
    RIGHT$ (M$(B),
    Y-C+1):Q=C-1
290:IF K=24AND D>0
    GOTO 4
300:IF K=15AND D>0
    END
315:IF K=27THEN
    PAUSE B;"@";A;
    "E";L;"ML";M;"
    MH";N;O
317:IF K<17OR K>22
    GOTO 50
320:BEEP 1
321:ON K-16+6*(E>0
    )GOTO 1000,120
    0,1300,1400,15
    00,1600,2000,2
    100,1800,50,22
    00,2300
1000:INPUT "LOAD-
    ";Z$
1005:IF Z$=""GOTO
    50
1010:CLEAR :INPUT
    #Z$;N,O,A:
    DIM M$(N)*O:
    INPUT #"";M$
```

Diagram *Status Indicators for the LED Program.*



Error Messages

Normal system errors, such as exceeding storage capabilities, are announced by regular system error messages.

Errors associated with the editor itself cause five beeps to be emitted. This occurs if an attempt is made to add more lines than are defined for the array or move an unmarked data item. (Note that moving data may cause the array limits to be exceeded.)

```
      (*):GOTO 10
1200:GOSUB 1850:
      GOTO 50
1300:GOSUB 2500:
      GOTO 50
1400:M=B:IF L=999
      LET L=B
1410:IF M<LLET Z=
      M:M=L:L=Z
1420:GOTO 50
1500:IF L=999THEN
      BEEP 5:GOTO
      50
1505:Y=M-L+1:
      GOSUB 3000:
      IF A+Y>NTHEN
      RETURN
1510:BEEP 2,100,3
      00:IF L>9LET
      L=L+Y:M=M+Y
1520:FOR Z=1TO Y:
      M$(Z+B)=M$(Z
      +L-1):NEXT Z
      :GOTO 50
1600:X=B:Y=A:
      GOSUB 1900:T
      $="FIND-":
      GOSUB 1950:U
      =0:GOSUB 196
      0:IF TLET B=
      Z:GOTO 50
1610:PAUSE "NOT F
      OUND":GOTO 5
      0
1800:GOSUB 1850:M
      $(B+1)=M$(B)
      :GOTO 50
1850:Y=1:GOSUB 30
      00:RETURN
1900:IF L<999LET
      X=L:Y=M:L=99
      9:M=L
1910:RETURN
1950:W$="":PRINT
      T$;:INPUT ""
      ;W$
1955:U=LEN W$:
      RETURN
1960:T=0:FOR Z=X
      TO Y
1970:U=U+1:IF
      MID$ (M$(Z),
      U,U)=W$LET T
      =1:RETURN
1980:IF U<=(LEN M
      $(Z)-U+1)
      GOTO 1970
1990:U=0:NEXT Z:
      RETURN
2000:INPUT "SAVE-
      ";Z$
2005:IF Z$=""GOTO
      50
2010:PRINT #Z$;N,
      0,A:PRINT #"
      ";M$(*):GOTO
      50
2100:X=B:Y=B:
      GOSUB 1900:
      FOR Z=YTO A:
      M$(Z-Y+X)=M$
      (Z+1):NEXT Z
      :A=A-Y+X-1
2105:FOR Z=1TO Y-
      X+1:BEEP 1,2
      55,100:NEXT
      Z:GOTO 50
2110:FOR Z=YTO A-
      1:M$(Z-Y+X)=
      M$(Z+1):NEXT
      Z:GOTO 50
2200:X=0:Y=A:
      GOSUB 1900
2210:INPUT "SIZE-
      ";T:CSIZE T:
      FOR Z=XTO Y:
      LPRINT M$(Z)
      :NEXT Z:GOTO
      50
2300:X=B:Y=A:
      GOSUB 1900:T
      $="FROM-":
      GOSUB 1950
2310:T$="":PRINT
      "TO-";:INPUT
      "";T$
2320:IF U=0THEN
      FOR Z=XTO Y:
      M$(Z)=T$+M$(
      Z):NEXT Z:
      GOTO 50
2330:U=0:GOSUB 19
      60
2340:IF TLET X=Z:
      M$(Z)=LEFT$
      (M$(Z),U-1)+
      T$+RIGHT$ (M
      $(Z),LEN M$(
      Z)-U-U+1):U=
      U+LEN T$+1:
      GOTO 2330
2350:GOTO 50
2500:L=999:M=L:
      RETURN
3000:IF A+Y>NBEEP
      5:RETURN
3010:FOR Z=ATO B+
      1STEP -1:M$(
      Z+Y)=M$(Z):
      NEXT Z:FOR Z
      =B+1TO B+Y:M
      $(Z)="":BEEP
      1,1,1
3020:NEXT Z:A=A+Y
      :RETURN
4000:INPUT M$(B)
4010:D=D*(M$(B)<>
      ""):B=B+(D>0
      ):GOTO 55
```

Notes for Programmers

The following information is provided for those who may wish to add to, modify or remove functions.

Lines	Description
4 - 10	Initialize variables, array.
20 - 30	Set up inverse display routine.
50	Reset shift/alternate switches.
55 - 60	Adjust current line number if outside array bounds.
70 - 90	Determine length of current line, adjust display partition and cursor position if cursor is off the screen.
95	Adjust cursor position to end of line if higher than line length.
100	Determine cursor position, display current partition of current line.
110	If current line is within marked limited, display partition in reverse video.
120 - 130	Display editor cursor at screen cursor location.
134	If null line and shift mode go to input routine.
140	Get key data. Repeat display if no key pressed.
145	Determine if a function key was pressed.
150 - 165	If in shift mode, adjust key data to shifted value.
170	Adjust key data if small mode and alpha key.
175	If cursor is at end of line, add character to end of line.
180	Otherwise, overlay or insert character at cursor position depending on status of mode switch.
185	Increment screen cursor position and go display again.
200	Adjust cursor if unshifted right arrow.
210	Adjust cursor if unshifted left arrow.
220	Adjust insert switch if MODE pressed.
225	Adjust small switch if SML pressed.
230	Adjust shift switch if SHIFT pressed.
235	Adjust alternate switch if up/down triangle pressed.
240	Adjust line number, partition number and screen cursor if ENT is pressed.
245	Adjust line pointer if down arrow pressed.
250	Adjust line pointer if up arrow pressed.
255	Add multiple lines to data set if RCL is pressed
260	If shifted function key, adjust key data and go insert or overlay data in the current line.
275	Delete the character at the cursor if shifted right arrow key.
280	Delete character to left of cursor if shifted left arrow key.
290	Re-initialize if shift and CLR.
300	End if shift and OFF.
317 - 321	Go to appropriate function key routine.
1000 - 1010	Load data routine.
1200	Add a line routine.
1300	Clear marked data routine.
1400 - 1420	Mark a line routine.
1500 - 1520	Move data routine.
1600 - 1610	Find data routine.
1800	Duplicate a line routine.
1850	Add a single line subroutine.
1900 - 1910	Set limits subroutine.
1950 - 1955	Input parameters subroutine.
1960 - 1990	Find data subroutine.
2000 - 2010	Save data routine.
2100 - 2110	Delete a line or group routine.
2200 - 2210	Print data routine.
2300 - 2350	Change data routine.
2500	Reset marks subroutine.
3000 - 3020	Add lines subroutines.
4000 - 4010	Input data subroutine.

Variable	Description
A	End of data pointer.
B	Current line pointer.
C	Line cursor position.
D	Shift switch.
E	Alternate switch.
F	Small switch.
G	Insert switch.
K	Key data.
L	Low mark.
M	High mark.
N	NUM.
O	LEN.
P	Screen partition.
Q	Screen cursor position.
V	Line length.
M$	Data array.

INSIDE THE PC-1500

Norlin Rober, 407 North 1st Ave., Marshalltown, IA 50158 is back with lots of "inside" information for users of the Sharp PC-1500. (Virtually all of this information also applies to the Radio Shack PC-2.)

In this issue Norlin provides a recap of information relating to the use of RAM. In coming issues he will provide detailed information on the ROM. If you are a long time subscriber to *PCN* you are well acquainted with the type of valuable information Norlin is able to glean. If you are a new subscriber, we are sure you will marvel at the wealth of information served up by our long time PC *Super Sleuth!*

PC-1500 SYSTEM RAM

Addresses 7600-7C00 contain the DisPlay Buffer, InPut Buffer, fixed variables, and various other RAM required by the system.

Because of incomPlete decodin9, the memory located 7600-77FF may also be addressed as 7000-71FF, 7200-73FF, or 7400-75FF.

In the PC-1500, the memory located 7800-7BFF is also addressable as 7C00-7FFF. (In some versions of the PC-1500, this will be recall-only.)

In the PC-1500A, the area 7C00-7FFFF contains RAM, intended for storage of machine-language Pro9rams.

DISPLAY BUFFER

Each dot of the LCD is turned on by the Presence of a '1' bit in a corresPondin9 location in the DisPlay Buffer. The uPPer four dots (U) in a column of the LCD are determined by one nybble, and the lower three dots (L) by another, as tabulated below.

Memory Address:	Columns controlled Lo nybble:	Hi nybble:	Memory Address:	Columns controlled Lo nybble:	Hi nybble:
7600	0 U	78 U	7700	39 U	117 U
7601	0 L	78 L	7701	39 L	117 L
7602	1 U	79 U	7702	40 U	118 U
7603	1 L	79 L	7703	40 L	118 L
7604	2 U	80 U	7704	41 U	119 U
7605	2 L	80 L	7705	41 L	119 L
. . .			. . .		
764C	38 U	116 U	774C	77 U	155 U
764D	38 L	116 L	774D	77 L	155 L

DisPlay annunciators are set by bits as follows:

764E Misc: 01, BUSY; 02, SHIFT; 04, Katakana characters; 08, SMALL; 10, III; 20, II; 40, I; 80, DEF
764F Modes: 01, DE; 02, G; 04, RAD; 10, RESERVE; 20, PRO; 40, RUN

774E Unused (By the PC-1500 or CE-150).
774F Unused

FIXED STRING VARIABLES

7650-5F E$	76B0-BF K$	7750-5F P$	77B0-BF V$
7660-6F F$	76C0-CF L$	7760-6F Q$	77C0-CF W$
7670-7F G$	76D0-DF M$	7770-7F R$	77D0-DF X$
7680-8F H$	76E0-EF N$	7780-8F S$	77E0-EF Y$
7690-9F I$	76F0-FF O$	7790-9F T$	77F0-FF Z$
76A0-AF J$		77A0-AF U$	

SYSTEM RAM

7800-784F comprise the stack used by the CPU. Storage into this stack begins at 784F, continuing in reverse direction through memory.

7850 Unused
7851 Unused
7852 Unused
7853 Unused
7854 Unused
7855 Unused
7856 Unused
7857 Unused
7858 Unused
used By CE158
7859 Unused
785A Unused
785B High byte, address of external character input routine
785C Low byte, external character input routine
785D If 80, Katakana displayed; 00, displayed and printed; FF, neither
785E High byte, location of table of Katakana character codes
785F Unused
7860 High byte, START OF ROM in Module; FF if no Module present.
7861 High byte, START OF BASIC Program in ROM Module; FF if no Module.
7862 Low byte, START OF BASIC Program in Module; FF if no Module.
7863 High byte, START OF RAM
7864 High byte, TOP OF RAM
7865 High byte, START OF BASIC Program in RAM
7866 Low byte, START OF BASIC Program in RAM
7867 High byte, END OF BASIC Program (stop byte address)
7868 Low byte, END OF BASIC Program
7869 High byte, START OF EDIT
786A Low byte, START OF EDIT
786B Flags: 01, BEEP OFF; 80, RMT ON
786C Unused
786D Unused
786E Unused
used By Rom
786F Unused
7870 Unused
7871 WAIT setting: 01, WAIT; 02, WAIT nnnn; 03, WAIT 0
7872 High byte, WAIT time counter
7873 Low byte, WAIT time counter
7874 Flags: 01, Cursor enabled; 80, display currently saved 7B10-7BAB
7875 CURSOR POINTER (current display column number)
7876 Character Position number in display, with INPUT statement
7877 Complement of number of display positions left for INPUT Prompt
7878 Specification for BEEP frequency
7879 Cassette Parameter: 1, MERGE; 2, CHAIN; 4, Data file; 8, M.L.; 10, RMT 1; 40, CLOAD?; 80, load operation
787A Unused

787B Position of blink character in display, plus 8
787C Flags: 01, blink cursor enabled: 80, a character is now blinked
787D Code of character blinked
787E High byte, location (in Display Buffer) of Blink Cursor
787F Low byte, location of Blink Cursor
7880 Display Parameter; determines display at READY, depending on bits set. 20, display contents of R0; otherwise, contents of Input Buffer. 10, display as BASIC line; 04, include colon after line number. 40, cursor activated in display. 80, ERROR message displayed. 08, RESERVE keyphrase displayed. 01, RESERVE template displayed, with Previous display saved in 7B10-7BAB.
7881 Parsing Parameter; specifies type of instruction that will conform to rules of syntax for evaluating BASIC expression.
7882 Subpointer, used with GOSUB and FOR stacks; also, 01 indicates a USING statements that is not part of a PRINT or PAUSE statement.
7883 High byte, LET Pointer; also, 80 indicates no data pass with CALL.
7884 Low byte, LET Pointer; also used to store code of RESERVE key.
7885 With LET Pointer: length if string, 80 if numeric variable
7886 High byte, INPUT Pointer (starting address of variable)
7887 Low byte,INPUT Pointer
7888 With INPUT Pointer: length if string, 80 if numeric variable
7889 Flag: 01, variable subscript(s) being Parsed
788A Program Halt Parameter: 10, waiting for input; 20, halt for PRINT; 40, INPUT statement Pending; 80, Break.
788B Low byte, Input Buffer Pointer
788C Number of function input arguments (used with Parsing routine).
788D TRACE: 00, OFF; other, ON. (Contents of 79D1 stored here by TRON)
788E TRACE Parameter; 00, starting execution of Previous line; 01, starting execution of a New Line; 02, expression in Input Buffer is to be evaluated; 04, expression has been evaluated (immediate mode)
788F Low byte, Output Buffer Pointer
7890 Low byte, FOR stack Pointer
7891 Low byte, GOSUB stack Pointer
7892 Low byte, BASIC DATA STACK Pointer, used by Parsing routine
7893 Low byte, BASIC PENDING OP STACK Pointer, used by Parsing routine
7894 Low byte, String Buffer Pointer
7895 USING editing character: 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific. (01 used to check syntax in interpretation of USING statement)
7896 USING number of characters,including sign, before decimal Point
7897 USING number of characters in string; 00, unspecified
7898 USING number of characters including and following decimal Point
7899 High byte, START OF VARIABLES in Main memory
789A Low byte, START OF VARIABLES in Main memory
789B Error Code
789C High byte, CURRENT line number; 00 if no Program in Progress
789D Low byte, CURRENT line number; 00 if no Program in Progress
789E High byte, beginning address of CURRENT Program
789F Low byte, beginning address of CURRENT Program
78A0 High byte, PREVIOUS address
78A1 Low byte, PREVIOUS address
78A2 High byte, PREVIOUS line number
78A3 Low byte, PREVIOUS line number
78A4 High byte, beginning of Program containing PREVIOUS line
78A5 Low byte, beginning of Program containing PREVIOUS line
78A6 High byte, SEARCH address
78A7 Low byte, SEARCH address

78A8 High byte, SEARCH line number
78A9 Low byte, SEARCH line number
78AA High byte, beginning of Program containing SEARCH line
78AB Low byte, beginning of Program containing SEARCH line
78AC High byte, BREAK address
78AD Low byte, BREAK address
78AE High byte, BREAK line number
78AF Low byte, BREAK line number
78B0 High byte, beginning of Program containing BREAK line
78B1 Low byte, beginning of Program containing BREAK line
78B2 High byte, ERROR address
78B3 Low byte, ERROR address
78B4 High byte, ERROR line number
78B5 Low byte, ERROR line number
78B6 High byte, beginning of Program containing ERROR line
78B7 Low byte, beginning of Program containing ERROR line
78B8 High byte, ON ERROR address
78B9 Low byte, ON ERROR address
78BA High byte, ON ERROR line number
78BB Low byte, ON ERROR line number
78BC High byte, beginning of Program containing ON ERROR line
78BD Low byte, beginning of Program containing ON ERROR line
78BE High byte, DATA Pointer; 80 indicates no DATA line yet located.
78BF Low byte, DATA Pointer

FIXED VARIABLES

78C0-CF A$	7920-27 E	7858-5F L	7998-9F T
78D0-DF B$	7928-2F F	7960-67 M	79A0-A7 U
78E0-EF C$	7930-37 G	7968-6F N	79A8-AF V
78F0-FF D$	7938-3F H	7970-77 O	79B0-B7 W
7900-07 A	7940-47 I	7978-7F P	79B8-BF X
7908-0F B	7948-4F J	7980-87 Q	79C0-C7 Y
7910-17 C	7950-57 K	7988-8F R	79C8-CF Z
7918-1F D		7990-97 S	

SYSTEM RAM (FOR OPTIONS)

79D0 ROM Bank: 00, ROM 1; 01, ROM 2
79D1 OPN device code: 60, LCD; 5C, CMT; 58, MGP; C4, LPRT; C0, COM
79D2 Unused
79D3 If 55, bypass setting of Modulation clock frequency, serial output
79D4 If 55, bypass keyboard scan; obtain input from external device
79D5 Unused
79D6 Unused
79D7 Unused
79D8 Unused
79D9 Used (by option)
79DA If 55, interrupt routine address in option is obtained from 79DB-DC
79DB High byte, interrupt routine address in option
79DC Low byte, interrupt routine address in option
79DD Unused
79DE Unused
79DF Unused

79E0 High byte of X-coordinate, in signed binary (GRAPH mode); also used for high byte of address of first line to be LLISTed
79E1 Low byte of X-coordinate, in signed binary (GRAPH mode); also used for low byte of address of first line to be LLISTed
79E2 High byte of Y-coordinate, in signed binary (GRAPH mode); also, high byte of address of last line to be LLISTed
79E3 Low byte of Y-coordinate, in signed binary (GRAPH mode); also, low byte of address of last line to be LLISTed
79E4 High byte, paper reverse feed count (counts from 0001 to 01FF)
79E5 Low byte, paper reverse feed count
79E6 Location of pen (00 to D8)
79E7 Low byte, extension of X-coordinate beyond available space
79E8 High byte, extension of X-coordinate beyond available space
79E9 Pen Parameter; specifies whether pen is to be raised or lowered
79EA LINE TYPE (0 to 9), GRAPH mode
79EB Dotted-line counter
79EC Current pen position: 00, up; 01, down
79ED X-motor hold counter
79EE Motor Phase; stored into Port C
79EF Y-motor hold counter
79F0 Print Mode: 00, TEXT; FF, GRAPH
79F1 Printer disable: 0F, pen change mode; FF, low btry or head blocked
79F2 ROTATE setting (0 to 3)
79F3 COLOR setting (0 to 3)
79F4 CSIZE setting (1 to 9)
79F5 LPRINT parameter: 00, single-item; 04, comma LPRINT; 08, semi-colon LPRINT; 01, last item reached. Also used with LLIST for maximum permissable line length.
79F6 With LINE, used as direction parameter; with LLIST, to determine line feed; with COLOR, to save pen location.
79F7 Type of data LPRINTed: 00, numeric; FF, character string
79F8 Temporary storage of pen location during paper feed
79F9 Flag: flag indicating powerup or interrupt in progress
79FA Unused
79FB Unused
79FC Unused
79FD Unused
79FE Unused
} used By CE158
79FF LOCK Mode: 00, LOCK; FF, UNLOCK

BASIC REGISTERS

Seven registers, used in BASIC computation, are located as follows:

7A00-07 R0 (Floating-Point accumulator)
7A08-0F R1 (Scratch register)
7A10-17 R2 (Second operand)
7A18-1F R3 (Scratch register)
7A20-27 R4 (Scratch register)
7A28-2F R5 (Scratch register)
7A30-37 R6 (Temporary storage register)

Each of these registers may contain a two-byte signed-binary number; a floating-point decimal number; or a pointer to a character string. The eight bytes in a register are utilized as follows:

BYTE	SIGNED BINARY	FLOATING-POINT	STRING POINTER
0	Unused	Exponent (binary)	Unused
1	Unused	Sign: 80 specifies -	Unused
2	Unused	1st & 2nd digits	Unused
3	Unused	3rd & 4th digits	Unused
4	B2 (to identify)	5th & 6th digits	D0 (to identify)
5	1st byte	7th & 8th digits	High byte, address
6	2nd byte	9th & 10th digits	Low byte, address
7	Unused	11th & 12th digits	Number of characters

Mantissas of calculated floating-point numbers are rounded to ten digits prior to display, printing, or storage. A string pointer formed by execution of CHR$ uses C1 instead of D0 as byte 4.

The area 7A00-7A37 is also used for other purposes, as follows:

The powerup routine uses addresses 7A10-14 (7A30-34 in earlier ROM versions) to temporarily save the addresses to be stored into the memory allocation pointers (RAM addresses 7860-64). Other pertinent information is stored by the powerup routine as follows:

7A20 If 00, powerdown was by 'OFF' key; 01, timed powerdown; 02, other type powerdown; 04, memory allocation pointers have been changed.
7A21 Set to other than zero by option with low battery power, indicating need for display of 'CHECK' message.
7A22 Set by option to indicate need for 'CHECK' message

A powerdown resulting from 7 minutes of non-use is identified by storage of A0, A1, ..., AF into addresses 7A10-7AFF; the stack pointer is saved in 7A30-31. (The powerdown initiated by use of the OFF key stores 50, 51, ..., 5F into 7A10-7AFF.)

The following are used in calculation of transcendental functions:

7A18 Flag: 00, SIN, COS, or ASN; 01, LOG, LN, or EXP; 20, ACS; 40, TAN or ATN; 80, result must be subtracted from 90 (inverse trig fctns)
7A20 Flag: 00, SIN; 01, COS

The PRINT and PAUSE routines use the area 7A10-7A34 to form a string of characters that represent numeric data formatted for display. The same area is used to form an ERROR or BREAK message.

The area beginning 7A00 is used as a scratch area in the interpretation of USING statements, and in execution of certain printer routines. In some of the printing routines, the contents of CPU register Y are saved in 7A26-7A27.

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

Issue 32 - March/April

NEW POCKET COMPUTER FROM HP

Hewlett-Packard Company has announced a new handheld computer dubbed the HP-71B. With measurements of just 3-7/8 by 7-1/2 by 1/2 inches and a weight of only 12 ounces, the unit fits easily into a jacket pocket. It was designed especially for technical professionals and has been optimized for numeric computations and calculations.

HP claims that it has the "most powerful 64-kilobyte ROM-based operating system available in a handheld computer." The machine combines an extended version of BASIC language

Photo *Hewlett-Packard HP-71B Shown With Optional HP-IL Module and Magnetic Strip-Card Reader.*

Another *Personal Information* (TM) product.

and a calculation mode to provide people with a highly portable tool capable of serious number crunching. Repetitive calculations may be performed by writing programs in BASIC or buying application packages that solve specific types of problems. A special CALC mode makes it easy to perform single calculations.

The 64 kilobytes of ROM is accompanied by 17.5 kilobytes of RAM in the basic unit. The unit is equipped with four slots that can accept additional RAM or ROM modules. Up to 16K of extra RAM or 256K of extra ROM can be added to the unit using these slots.

Space is also provided in the machine to install an optional magnetic card reader and an HP Interface Loop (HP-IL). The card reader reads (and writes) 10-inch magnetic strips that can hold up to 1.3 kilobytes of information. The HP-IL modules permits the HP-71B to be connected to a variety of peripheral devices such as a magnetic tape drive, printers and test equipment.

The pocketable unit is powered by four "AAA" alkaline batteries or by an optional alternating current adaptor.

The 71B has a block QWERTY-style keyboard with typing aids for facilitating use of BASIC. A separate 10-digit pad is used for numeric inputs. The entire keyboard is redefinable. Mylar overlays are available so that users can customize the computer for specific purposes. The display is an 8 by 132-element, single-line dot matrix LCD with a large font. It is supported by status annunciators. Twenty-two characters within a 96-character line may be displayed at a time. However, since each element in the display is individually controllable, users may create their own set(s) of graphics or customized characters.

The version of HP extended BASIC used in the machine provides over 240 instructions. In addition to compliance with IEEE floating-point math standards, it includes enhancements such as trigonometric functions and statistics operations. The language also provides: complete control of display pixels, ability to maintain multiple programs in memory, dynamically declared variables, and the ability to define multi-line user functions. Furthermore, additional keywords can be added to extend the language.

A built-in timer allows events to be controlled in real time. Time and date information is retained even when the HP-71B is turned off.

The handheld computer also has a calculation mode that enables it to function as an advanced calculator. Even though this mode is not programmable, it does provide error-checking and results can be tracked as a calculation is performed. Variables assigned values in the BASIC mode can be used in the CALC mode (as well as the reverse).

HP plans to support the new handheld with a series of application programs including math, curve fitting, AC circuit analysis, surveying, finance and text-editing. The firm also plans to provide a combination Assembly/FORTH System for developing special application programs. There will also be a user's library made available for the new model. Programs submitted by users will be made available through the library.

Hewlett-Packard Company has also indicated that the HP-71B will be a completely open machine. That is, it is encouraging third-party vendors to develop software, hardware and interfaces for the computer. It is making extensive documentation and a choice of development languages available as well as offering support of several media on which software may be delivered.

For instance, in addition to the usual owner's documentation, HP is making available three volumes of internal design specifications. These manuals contain detailed information on the unit's internal hardware and software. Three types of software development languages will be available: HP-71B Assembly, FORTH and BASIC. Software vendors will be able to distribute their wares on ROMs, cassette tapes and magnetic strip cards.

The HP-71B is available from HP authorized dealers. It is said to be priced at less than $550.00. For the location of the nearest dealer phone, toll free, 1-800-FOR-HPPC.

SCHEDULE PLOTTER PROGRAM

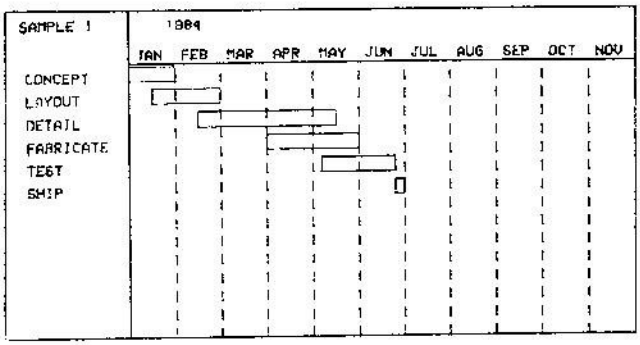
Richard H. Chrystie, 14824 E. Walbrook, Hacienda Heights, CA 91745, submitted this program. You can use it to plot schedules over a period of weeks or months.

To use it, just load the program and execute a RUN command or press the DEF key and and then the S key. Respond to the queries and watch your schedule be plotted right before your eyes!

You can customize the program to your own liking. For instance, if, when using it in the monthly mode, you want it to start with a month other than January, change the start date in line 620. You can then change the column month labels that begin at line 780.

Program *Schedule Plotter.*

```
 10:"S"REM   SCHED
    PLOT
 11:TEXT :COLOR 0:
    CLEAR
 15:DIM A$(12),B(1
    2),C(12)
 20:INPUT "PROGRAM
     TITLE?",B$
100:GOSUB 700
110:GLCURSOR (0,-4
    00)
120:IF Z$="M"THEN
    600
400:N=1
410:INPUT "TASK TI
    TLE?",A$(N)
420:INPUT "START (
    DAYS FROM NOW)
    ?",B(N):B(N)=B
    (N)*6
430:INPUT "TASK DU
    RATION (DAYS)?
    ",C(N):C(N)=C(
    N)*6
435:IF (B(N)+C(N))
    >330THEN LET C
    (N)=330-B(N)
440:L=183-(15*N)
449:COLOR 2:K=-80
450:GLCURSOR (L,-1
    3):LPRINT A$(N
    )
455:LINE (L,K-B(N)
    )-(L+10,K-B(N)
    -C(N)),,,B
460:GLCURSOR (0,-3
    80)
470:IF N=12THEN 49
    8
480:INPUT "WANT AN
    OTHER TASK?",D
    $
481:IF D$="N"THEN
    498
485:N=N+1
487:GOTO 410
498:GLCURSOR (0,-5
    00)
499:INPUT "WANT AN
    OTHER PLOT?";Z
    $
500:IF Z$="Y"THEN
    502
501:GLCURSOR (0,-5
    00):END
502:GOTO 11
600:N=1
610:INPUT "TASK TI
    TLE?",A$(N)
620:INPUT "START (
    MOS AFTER 1 JA
    N)?",B(N):B(N)
    =B(N)*30
630:INPUT "TASK DU
    RATION (MOS)?"
    ,C(N):C(N)=C(N
    )*30
635:IF (B(N)+C(N))
    >330THEN LET C
    (N)=330-B(N)
640:L=183-(15*N)
649:COLOR 2:K=-80
650:GLCURSOR (L,-1
    3):LPRINT A$(N
    )
655:LINE (L,K-B(N)
    )-(L+10,K-B(N)
    -C(N)),,,B
660:GLCURSOR (0,-3
    80)
670:IF N=12THEN 69
    8
680:INPUT "WANT AN
    OTHER TASK?",D
    $
681:IF D$="N"THEN
    698
685:N=N+1
687:GOTO 610
698:GLCURSOR (0,-5
    00)
699:GOTO 499
700:GRAPH
710:COLOR 0
715:LINE (0,0)-(21
    5,-410),,,B
720:LINE (0,-80)-(
    215,-80)
730:LINE (180,-80)
    -(180,-410)
740:FOR X=110TO 37
    0STEP 30
745:LINE (0,-X)-(1
    80,-X),5
750:NEXT X
755:CSIZE 1:ROTATE
    1:COLOR 1
760:GLCURSOR (203,
    -10)
775:LPRINT B$
776:INPUT "WEEKS O
    R MONTHS (W/M)
    ?",Z$
777:IF Z$="W"THEN
    800
780:GLCURSOR (183,
    -85):LPRINT "J
    AN"
781:GLCURSOR (183,
    -115):LPRINT "
    FEB"
782:GLCURSOR (183,
    -145):LPRINT "
    MAR"
783:GLCURSOR (183,
    -175):LPRINT "
    APR"
784:GLCURSOR (183,
    -205):LPRINT "
    MAY"
785:GLCURSOR (183,
    -235):LPRINT "
    JUN"
786:GLCURSOR (183,
    -265):LPRINT "
    JUL"
787:GLCURSOR (183,
    -295):LPRINT "
    AUG"
788:GLCURSOR (183,
    -325):LPRINT "
    SEP"
789:GLCURSOR (183,
    -355):LPRINT "
    OCT"
790:GLCURSOR (183,
    -385):LPRINT "
    NOV"
791:GLCURSOR (203,
    -105):LPRINT "
    1984"
792:RETURN
800:GLCURSOR (183,
    -90):LPRINT "1
    "
801:GLCURSOR (183,
    -120):LPRINT "
    2"
802:GLCURSOR (183,
    -150):LPRINT "
    3"
803:GLCURSOR (183,
    -180):LPRINT "
    4"
804:GLCURSOR (183,
    -210):LPRINT "
    5"
805:GLCURSOR (183,
    -240):LPRINT "
    6"
806:GLCURSOR (183,
    -270):LPRINT "
    7"
807:GLCURSOR (183,
    -300):LPRINT "
    8"
808:GLCURSOR (183,
    -330):LPRINT "
    9"
809:GLCURSOR (183,
    -360):LPRINT "
    10"
810:GLCURSOR (183,
    -390):LPRINT "
    11"
811:GLCURSOR (203,
    -105):LPRINT "
       WEEKS"
812:RETURN
STATUS 1        1722
```



APPOINTMENT CALENDAR

This program can be of use to almost everybody. It is particularly valuable because it gives you the capability of searching for a key word within a field to help you locate a specific appointment.

For example, suppose you had been invited to a golf game on Friday, September 23rd. But, you do not remember the exact time. Just run the program and select item 2 (Find Appointment) from the menu. Respond to the prompt for a date. If you make a mistake while entering it, error-trapping will advise you and prompt you again. Once you have entered the date, the PC will list all your committments on that date. The listing will be in the order in which you entered them. (In keeping with a goal of having enough memory available for a practical calendar, a rather lengthy sorting routine was omitted.) When the appointments for the given date have been shown, the program returns to its main memory.

Here is another example. Suppose you are running out of a favorite product that is sold only by Amway at Amway parties. You cannot remember when the next party is scheduled. Just run the program and select item 3 (Find Date) from the menu. In a few seconds the computer will begin displaying all of your Amway parties scheduled for the entire year! You could even spell Amway wrong and, unless you really mutilate it, the program will still find it for you!

If you like, you can also review appointments by the month (choose item 4 from the main menu). Other capabilities included in the program are: deleting the appointment file, saving data, loading data and adding appointments.

Operating Tips

There are a few things you need to remember about the PC-2 and PC-1500 Pocket Computers when

Program *Appointment Calendar.*

```
  5:DIM M$(60)*25,
    N$(60)*25,J$(2
    )*2,K$(2)*5
 10:"A":CLS :WAIT
    99:CURSOR 4:
    PRINT "APPOINT
    MENT CALENDAR"
 12:CLS :BEEP 1,10
    0
 20:CLS :CURSOR 8:
    PRINT "***MENU
    ***"
 22:CLS
 30:PAUSE "1.ENTER
     DATES/APPOINT
    MENTS"
 40:PAUSE "2.FIND
    APPOINTMENT"
 50:PAUSE "3.FIND
    DATE"
 60:PAUSE "4.REVIE
    W APPTS BY MON
    TH
 70:PAUSE "5.I/O O
    PTIONS      "
 75:PAUSE "6 DELET
    E APPOINTMENT"
 80:PAUSE "7.SEE M
    ENU AGAIN"
 90:INPUT M
100:ON MGOSUB 1000
    ,2000,3000,400
    0,5000,6000,70
    00
110:GOTO 12
 1000:BEEP 1,150:
      PRINT "IS TH
      IS AN ORIGIN
      AL ENTRY?"
1002:INPUT "
           (Y/N)?
      ";J$
1003:IF J$="N"
      THEN GOTO 80
      00
1004:FOR L=1TO 60
1005:INPUT "ENTER
      DATE (MO/DA
      ): ";M$(L)
1015:IF M$(L)="ZZ
      "THEN GOTO 1
      050
1016:IF LEN (M$(L
      ))<>5THEN
      GOTO 9000
1020:INPUT "ENTER
      APPOINTMENT
      :";N$(L)
1030:NEXT L
1040:GOTO 20
1050:RETURN
2000:BEEP 1,150
2010:PAUSE "FIND
      APPOINTMENT"
2015:INPUT "ENTER
      DATE (MO/DA
      ): ";K$
2020:IF LEN (K$)<
      >5THEN BEEP
      5,150,25:
      PRINT "FORMA
      T IS 4 DIGIT
      S--REDO":
      GOTO 2015
2021:X=1:Y=0:GOTO
      2025
2024:X=X+1
2025:FOR X=XTO L-
      1
2030:IF K$=M$(X)
      THEN 2070
2040:NEXT X
2045:IF Y>=1THEN
      2110
2050:PRINT "NO AP
      POINTMENT ON
      : ";K$
2055:K$=""
2060:RETURN
2070:PRINT "YOUR
      APPOINTMENT
      ON: ";M$(X);
      "IS:"
2075:PRINT "IS: "
      ;N$(X)
2077:Y=Y+1
2080:FOR C=1TO 75
      :NEXT C
2093:GOTO 2024
2110:PRINT "NO MO
      RE APPTS ON
      THIS DATE"
2120:K$=""
2130:RETURN
3000:BEEP 1,150
3005:CLS :CURSOR
      8:PRINT "FIN
      D DATE"
3010:INPUT "ENTER
      APPOINTMENT
      ";U$
3012:X=0:Y=0
3014:X=X+1
3020:FOR X=XTO L-
      1
3025:IF U$=N$(X)
      THEN 3050
3030:IF LEFT$ (N$
      (X),4)=LEFT$
      (U$,4)THEN 3
      050
3035:IF RIGHT$ (N
      $(X),4)=
      RIGHT$ (U$,4
      )THEN 3050
3040:NEXT X
3042:IF Y>=1THEN
      3090
3045:GOTO 3080
3050:PRINT M$(X);
      " ";N$(X)
3052:Y=Y+1
3055:FOR R=1TO 10
      0:NEXT R
3057:GOTO 3014
3060:U$=""
3070:RETURN
3080:PRINT "THAT
      APPT IS NOT
      ON FILE"
3090:U$=""
3100:RETURN
4000:CLS :BEEP 1,
      150
```

using this program.

First, remember that when you give a RUN command, all variables stored in user memory will be lost. Thus, you should only use the RUN command when you use the program for the first time or after loading the *program* from tape. It is necessary to do so under these circumstances in order to properly dimension the string arrays used by the program.

How do you start the program without typing RUN? Just press the DEF key followed by the letter A (DEF/A).

In order to start building up an appointments list you will need to choose item 1 from the main menu. When you do this you will queried as to whether this is an original entry. If it is an original entry, (the first time you have entered information into an appointment file) respond with a Y for yes. A yes response causes the data file to be initialized to the empty condition. If you are adding data to a file that already has appointments stored, then be sure to answer N for no!

You can customize the display timings to your own liking. There are three basic methods of controlling the length of time a display stays on the screen of a PC-2 or PC-1500. The PAUSE and PRINT commands are used in this program. PAUSE gives a display time that is set at about one second. Using the PRINT command coupled with WAIT (followed by a number), you can keep the display on for whatever period is desired. A WAIT value of 99 is used in this program. It is specified in line 10. If you would like the displays to last a little longer, increase the value in line 10. Remember that altering this value will change all PRINT statements used in the program but has no effect on the PAUSE statements.

This program has been designed to fit in a 4K

```
4005:INPUT "ENTER
     MONTH TO RE
     VIEW ";H$
4008:IF LEN (H$)<
     >2THEN BEEP
     10,150,25:
     PRINT "FORMA
     T IS 2 DIGIT
     S---REDO":
     GOTO 4005
4020:X=0:Y=0
4030:X=X+1
4035:FOR X=XTO L-
     1
4040:IF H$=LEFT$
     (M$(X),2)
     THEN 4070
4050:NEXT X
4055:IF Y>=1THEN
     4110
4060:PRINT "NO AP
     PTS LISTED T
     HIS MONTH"
4065:H$="":RETURN
4070:PRINT RIGHT$
     (M$(X),2);"
     ";N$(X)
4080:Y=Y+1
4090:FOR C=1TO 75
     :NEXT C
4100:GOTO 4030
4110:PRINT "NO MO
     RE APPTS THI
     S MONTH"
4120:H$="":RETURN
5000:CLS :BEEP 1,
     150
5005:CLS :CURSOR
     4:PRINT "MEN
     U--I/O OPTIO
     NS"
5010:PAUSE "1.SAV
     E TO TAPE"
5020:PAUSE "2.LOA
     D FROM TAPE
     "
5030:PAUSE "3.RET
     URN TO MAIN
     MENU"
5033:PAUSE "4.SEE
      I/O MENU AG
     AIN"
5035:INPUT N
5040:ON NGOSUB 10
     000,11000,12
     000
5060:GOTO 5000
6000:CLS :BEEP 1,
     150:J=1
6005:CURSOR 5:
     PRINT "DELET
     E APPOINTMEN
     T"
6010:INPUT "APPOI
     NTMENT DATE
     (MO/DA) ";G$
6015:IF LEN (G$)<
     >5THEN BEEP
     5,150,25:
     PRINT "FORMA
     T IS 5 DIGIT
     S---REDO":
     GOTO 6010
6030:FOR X=1TO L-
     1
6040:IF G$=M$(X)
     THEN PRINT "
     ITEM #";X;"
      ";N$(X):J=J
     +1
6050:NEXT X
6051:IF J<=1THEN
     PRINT "THERE
     S NO APPT ON
      ";G$:GOTO 6
     100
6052:PRINT "SEE A
     PPTS ON ";G$
     ;" AGAIN?
     "
6053:CLS :CURSOR
     9:PRINT "(Y/
     N)?"
6054:INPUT L$
6055:IF L$="Y"
     THEN 6030
6060:INPUT "DELET
     E ITEM NUMBE
     R: ";F
6070:N$(F)="":M$(
     F)=""
6075:L$=""
6080:RETURN
6100:G$="":RETURN
7000:GOTO 20
8000:CLS :BEEP 1,
     150
8005:PRINT "MAKE
     ADDITIONS TO
      CALENDAR
8010:FOR L=LTO 60
8030:GOTO 1005
9000:CLS :BEEP 10
     ,100,25
9005:PRINT "  FOR
     MAT IS (MO/D
     A) REDO
9010:GOTO 1005
10000:PRINT "SAVE
     TO TAPE"
10020:PRINT "PREPA
     RE TAPE RECO
     RDER"
10030:FOR B=1TO 20
     0:NEXT B
10040:PRINT #L,M$(
     *),N$(*)
10050:CLS :BEEP 1,
     150
10060:RETURN
11000:PRINT "LOAD
     FROM TAPE"
11010:PRINT "PREPA
     RE TAPE RECO
     RDER"
11020:FOR D=1TO 20
     0:NEXT D
11030:INPUT #L,M$(
     *),N$(*)
11040:CLS :BEEP 1,
     150
11050:RETURN
12000:GOTO 20
STATUS 1    2689
```

machine. If you have an 8K memory module installed (which nets you 10K), you can change the number of potential records by the changing the values in the loops at lines 1004 and 8010, to something in the order of 100. If you do this you must also change the numbers within the parenthesis of the dimension declarations (in line 5) to 100 instead of 60. If you assign too many records in the dimension statements, you will simply get an OUT OF MEMORY error when you try to use the program. Reduce the values used if this occurs.

Finally, if you have never saved a data file before using your PC, then you may be in for a bit of a surprise. Did you realize that even if you only have two appointment entries in your appointment file, a tape data save command will save all of the records that have been dimensioned, even though most of them are empty? Yes, the PC will save every bit of the reserved array space, whether or not it contains information. The result is that saving the appointment file on tape will take a fixed amount of time, regardless of how many entries you have made. Have patience, please.

This program was submitted by: *Steve Eichman, 5809 Northland Road, Manteca, CA 95336.*

FORMATTED MORTGAGE PROGRAM

Here is a program that will calculate monthly payments and amortization information on a mortgage or loan and print out a neatly formatted schedule.

The program asks for the mortgage/loan amount, annual interest percentage and amount of monthly payments. If the monthly payment is specified, the amortization will be calculated. Optionally, if the monthly payment is not given (simply press ENTER at the prompt), the program will ask the user for amortization and calculate the appropriate monthly payments. Next, the starting date and number of payments to be printed in the schedule must be provided. Finally, the program gives the option of specifying a number of extra payments. If only regular payments will be made, press the ENTER key at this prompt. When executed the program lists a summary of the input data, followed by a payment schedule that shows the payment number, date, interest and principal portions of the payment, and the account balance as of that date.

NOTICE: This program was written to calculate mortgage schedules according to Canadian laws. However, with a few simple changes it can be adapted to other types of loans.

In Canada, the law states that interest on mortgages may not be charged in advance. This means that a lender is not entitled to interest more often than the mortgage's compounding frequency which is a maximum of semi-annually. Thus, if interest payments are in fact being made more often than the law permits (such as monthly), then interest is being charged in advance, unless an appropriate discount is made. This may be done by reinvesting the interest portion of the monthly payment at the mortgage rate until the end of the current compounding period. This effectively reduces the actual interest rate. The effective

Program *Formatted Mortgage.*

```
  5:PAUSE "... MOR
    TGAGE SCHEDULE
    ...":LOCK
 10:REM     1983 -
    Peter V. HART
    MANN
 20:CLEAR :CSIZE 2
    :B=10:U=1:DIM
    PN(B),EP(B)
 30:A$="Jan":B$="F
    eb":C$="Mar":D
    $="Apr":E$="Ma
    y":F$="Jun"
 40:G$="Jul":H$="A
    ug":I$="Sep":J
    $="Oct":K$="No
    v":L$="Dec"
 50:INPUT "AMOUNT
    = $ ";A
 60:INPUT "INTERES
    T (%) = ";I
 70:C=2
 80:INPUT "MONTHLY
     PAYMENTS = $"
    ;P
 90:IF P=0INPUT "A
    MORTIZED OVER
    (yrs) = ";Z:N=
    Z*12
100:INPUT "STARTIN
    G DATE- DAY: "
    ;D,"MONTH: ";M
    ,"YEAR: ";Y
110:Y=Y-100*INT (Y
    /100)
120:INPUT "NO. OF
    PAYMENTS = ";U
130:INPUT "EXTRA P
    AYMENTS ? ";Z$
140:IF Z$<>"Y"GOTO
    170
150:FOR J=0TO B:
    INPUT "PAYMENT
     # = ";PN(J)
155:IF PN(J)=0GOTO
    170
160:INPUT "PAYMENT
     AMOUNT = $";E
    P(J):NEXT J
170:R=(1+(I/100)/C
    )^(C/12)-1
180:IF P>0GOTO 200
190:X=(R+1)^N:P=A*
    R*X/(X-1):GOTO
    210
200:N=LOG (P/(P-A*
    R))/LOG (1+R):
    Z=N/12
210:COLOR 2:LPRINT
    "-------------
    -----"
220:COLOR 3:LPRINT
    "MORTGAGE SCHE
    DULE "
230:COLOR 2:LPRINT
    "-------------
    -----"
240:COLOR 0:LF 1:
    USING "########
    "
250:LPRINT "AMOUNT
      = $";A
260:LPRINT "INTERE
    ST=";USING "##
    ###.##";I;"%"
270:LPRINT "YEARS
      =";USING "##
    ####.##";Z
```

monthly interest rate is given by the equation:

$$r = ((1 + i/c)^{(c/12)}) - 1$$

In this formula, r = the effective monthly interest rate, i = the annual interest rate and c = the number of compounding periods per year.

Program Options

The maximum number of extra payments that can be handled by the program is a function of the amount of memory available. The value of variable B in line 20 of the listing sets the allowable limit. In an unexpanded PC-1500 the number of such payments is limited to 11. (B is set to 10 and is then used to DIMension an array having 11 elements.) If you have additional memory in your PC you can increase this value accordingly.

```
280:COLOR 1:LF 1:
    LPRINT "EFF.MO
    NTHLY RATE:"
290:LPRINT USING "
    ##.############
    ";R;"%"
300:COLOR 3:LF 1:
    LPRINT "MONTHL
    Y PAYMENTS:"
310:LPRINT "    $"
    ;USING "#####.
    ##";P
320:COLOR 2:LPRINT
    "-------------
    -----"
330:LF 2:CSIZE 1:
    COLOR 0:IF U=0
    GOTO 510
340:LPRINT "  #
    DATE       INT.
      PRINC. BALAN
    CE"
350:LPRINT "------
    --------------
    --------------
    --"
360:LPRINT TAB 3;
    USING "###";D;
    @$(M);Y;TAB 29
    ;USING "######
    #";A
370:K=M+1:IF K>12
    LET Y=Y+1:IF K
    >12LET K=1
380:S=0:T=0
390:FOR J=KTO 12:Q
    =P
400:FOR F=1TO B:IF
    PN(F)<>UGOTO 4
    20
410:Q=EP(F):GOTO 4
    30
420:NEXT F
430:G=A*R:H=Q-G:A=
    A-H:IF A>0.01
    GOTO 450
440:H=H+A:Q=H+G:A=
    0
450:LPRINT USING "
    ###";U;D;@$(J)
    ;Y;USING "####
    ##.##";G;USING
    "######.##";H;
    USING "#######
    ";A
460:S=S+G:T=T+H:U=
    U+1:IF A=0GOTO
    480
470:NEXT J
480:LPRINT "------
    --------------
    --------------
    --":X=Y+1900
490:LPRINT " TOTAL
    ";USING "#####
    ";X;USING "###
    ####.##";S;
    USING "######.#
    #";T:LF 1
500:K=1:Y=Y+1:IF U
    <UAND A>0.01
    GOTO 380
510:UNLOCK :END
STATUS 1        1649
```

To use the program to calculate other types of mortgages or simple loans, change the title in line 5 to suit your purposes. Also change line 70 to:

70 INPUT COMPOUNDED(* A YEAR)";C

and alter line 270 to read:

270 R=I

This program was submitted by: *Peter V. Hartmann, P.O. Box 7003, Station A, Toronto, Ontario, Canada M5W 1X7.*

Sample *Listing from Formatted Mortage Program.*

```
--------------------
MORTGAGE SCHEDULE
--------------------

AMOUNT   = $   60000
INTEREST=     12.75%
YEARS    =     25.00

EFF.MONTHLY RATE:
 0.01035329508%

MONTHLY PAYMENTS:
     $   650.81
--------------------
```

#	DATE	INT.	PRINC.	BALANCE
	17Aug 83			60000
1	17Sep 83	621.19	29.61	59970
2	17Oct 83	620.89	29.91	59940
3	17Nov 83	620.58	30.22	59910
4	17Dec 83	620.26	30.54	59879
TOTAL 1983		2482.93	120.38	
5	17Jan 84	619.95	30.85	59848
6	17Feb 84	619.63	31.17	59817
7	17Mar 84	619.30	31.50	59786
8	17Apr 84	618.98	31.82	59754
9	17May 84	618.65	32.15	59722
10	17Jun 84	618.32	32.48	59689
11	17Jul 84	617.98	32.82	59656
12	17Aug 84	617.64	33.16	59623
13	17Sep 84	617.30	33.50	59590
14	17Oct 84	616.95	33.85	59556
15	17Nov 84	616.60	34.20	59522
16	17Dec 84	616.25	34.56	59487
TOTAL 1984		2417.59	392.12	
17	17Jan 85	615.89	34.91	59452
18	17Feb 85	615.53	35.27	59417
19	17Mar 85	615.16	35.64	59381
20	17Apr 85	614.79	36.01	59345
21	17May 85	614.42	36.38	59309
22	17Jun 85	614.04	36.76	59272
23	17Jul 85	613.66	37.14	59235
24	17Aug 85	613.28	37.52	59197
25	17Sep 85	612.89	37.91	59159
26	17Oct 85	612.50	38.30	59121
27	17Nov 85	612.10	38.70	59082
28	17Dec 85	611.70	39.10	59043
TOTAL 1985		7366.00	443.71	
29	17Jan 86	611.29	39.51	59004
30	17Feb 86	610.88	39.92	58964
31	17Mar 86	610.47	40.33	58924
32	17Apr 86	610.05	40.75	58883
33	17May 86	609.63	41.17	58842
34	17Jun 86	609.21	41.60	58800
35	17Jul 86	608.77	42.03	58758
36	17Aug 86	608.34	42.46	58716
37	17Sep 86	607.90	42.90	58673
38	17Oct 86	607.45	43.34	58629
39	17Nov 86	607.01	43.79	58586
40	17Dec 86	606.55	44.25	58541
TOTAL 1986		7307.62	502.09	

FLIGHT COMPUTER USES PC-1500

A new flight computer dubbed the PETREL has been announced. The unit is based on a Sharp PC-1500 Pocket Computer. A custom made memory expansion module about the size of a matchbook expands the memory of the PC-1500 by 32K of ROM and 6K of RAM. This additional memory allows the unit to hold a data base containing the location of all 1000 VOR stations in the United States. This data base coupled with proprietory operating programs enables the unit to compute the latitude and longitude of any Victor airways intersection after entry of just the VOR identifiers and radials. The system can also compute as many RNAV waypoints as desired for great circle routes up to 5000 miles in length. Up to 104 intersections or waypoints may be stored by the user and recalled by identifier alone. A user can also add up to 127 new entries or changes to the VOR data base. Furthermore, a variety of flight planning, preflight, navigation and inflight functions are included in the computing package.

For instance, the preflight capabilities allow the pilot to enter the empty weight and moments for fuel, oil, seats and baggage just once. This information is retained by the system for future use. Before each flight the operator adds the actual fuel and weight at each station. The PETREL system then computes the total weight and center of gravity figures as well as maximum gross weight and C.G. limits. In fact, the package will store the data needed for two separate aircraft so that one craft can be flown without losing the data used by the other. Additionally, the unit handles all typical conversions used by pilots: miles-nautical to miles-kilometers-feet, farenheit-celsius, liters-gallons-imperial gallons, and kilograms-pounds. It also handles conversions of true airspeed from calibrated airspeed or Mach numbers. The system provides 10 auxiliary memory scratch pads so that converted numbers can be entered directly into operating programs.

While navigating the PETREL can compute the total distance, flight time and fuel requirements without map reference. Three separate flight plans can be stored and run independently. Winds and climb requirements are computed automatically. The distance and bearing between any two airway points or geographic points can be determined in seconds without map reference. ETAs and fuel requirements can then be computed. Using its self-contained VOR data base, LORAN-C and other users can compute and display in a matter of seconds, the latitude and longitude of any VOR intersection. Only the VOR identifier and radial is entered to obtain this information. For a DME fix, just add the distance. Up to 104 intersections or other fixes can be stored and later recalled using just the identifier.

While inflight the system aids pilots through the maintenance of two 15-item general purpose check lists. A holding pattern routine displays the proper pattern entry procedure and provides a countdown timer for the outbound leg. If you input the final approach course, distance and wind, the PETREL will compute the time to missed approach and display a countdown/countup timer with an alarm. The programs are designed to keep inflight key punching to a minimum. For instance, the Distance-Time-Fuel program allows the operator to enter just the distance, leaving the speed and fuel consumption as they had been previously defined, and quickly obtain a current read out of the flight time, fuel used and ETA.

The PETREL Flight Computer is the result of several years of programming and engineering development work. It represents one of the most powerful applications of a pocket computer ever announced. It uses a combination of machine language and BASIC routines plus a comprehensive VOR data base stored on ROM. The PETREL currently is priced at $695.00. For more information contact: *Somerset Engineering Company, P.O. Box 14, Concord, MA 01742.* The phone number is (617) 369-1090.

GRAPHICS PACKAGE FOR THE PC-2

Radio Shack has announced the availability of a program called called the PC-2 Graphics Pak. It is said that the software package makes it easy to create charts using the Radio Shack TRS-80 Pocket Computer Model PC-2 combined with the PC-2 Printer/Cassette Interface.

The package of programs can be used to draw six different types of charts: horizontal and vertical bar charts, bar-segment charts, point, line and pie charts.

Data may be entered from the keyboard of the PC. It is turned into chart format using combinations of labeling and shading. A data file used in a chart can be stored on cassette tape and used to create another chart.

The package retails for $19.95 in the U.S. at Radio Shack Computer Centers and other Radio Shack outlets.

INSIDE THE PC-1500

This is a continuation of the series by *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* We pick up in this issue from where we left off in *PCN* Issue 31, with a recap of information relating to the use of RAM in the PC-1500. (This information also applies to the Radio Shack PC-2.)

Then Norlin goes into the subject of ROM routines used in the PC-1500 (which are again generally applicable to the PC-2). There are many routines within the BASIC ROM that may be of use to machine language programmers. Norlin has done an amazing job of uncovering these routines and providing a synopsis of their capabilities. Space does not permit the publication of all the ROM information he has dug up in this issue, so look for more in future editions of *PCN*.

THE BASIC STACK

The BASIC stack, located 7A38-7AFF, includes a FOR stack, a GOSUB stack, and an arithmetic stack.

The FOR loop stack begins at 7A38. UP to 16 loops may be Pending at a given time (if the BASIC stack is not required for other PurPoses). Each FOR register contains 12 bytes, used as follows:

0-1 Memory address of control variable
2-3 Test value (in signed binary)
4-5 Step size (in signed binary)
6-7 Address of next statement following FOR; Plus 80 if not the first statement in a line.
8-9 Number of line containing the statement following FOR
A-B Address of beginning of Program in Progress

GOSUB return addresses are stacked beginning at 7AFF, Proceeding in reverse direction through memory. UP to 33 return addresses may be Pending (if the BASIC stack is not required for other PurPoses). Each GOSUB register contains 6 bytes, as follows:

0-1 Address of statement following GOSUB; Plus 80 if not the first statement in a line.
2-3 Number of line containing the statement following GOSUB
4-5 Address of beginning of Program in Progress

The space between the FOR and GOSUB stacks comPrises the arithmetic stack, which is used by the Parsing routine in evaluating exPressions. Pending numeric data (or string Pointers) will be stacked (in 8-byte blocks) starting at 7A3C, or at the location four bytes after the last address used for storing a FOR register. Codes rePresenting Pending oPerations are stacked in reverse direction through memory, starting at 7AFF, or at the location just Preceding the stacked GOSUB registers.

Each Pending oPeration code contains two bytes, of which the first indicates the Priority of the oPeration. Those rePresenting functions consist of the token codes. (The tokens for SQR and PI rePresent the corresPonding symbols.) The other Pending oPeration codes follow:

84	5E	↑	70	51	OR
83	21	sign change	70	50	AND
82	2F	/	60	2C	comma seParating arguments
82	2A	*			
81	2D	-	5A	xx	variable name

```
81 2B  +                        59 xx  variable name
80 06  >=                       ...
80 05  <=                       41 xx  variable name
80 04  =
80 02  >                        40 80  @(
80 01  <                        20 28  left-Parenthesis
80 00  <>
```

SYSTEM RAM

7B00-7B07 contain the current value of the random number.

7B08 Unused
7B09 Counter for time that a key is held down
7B0A High byte, timed Powerdown counter (counts from FE1D1D to FFFFFF)
7B0B Second byte, timed Powerdown counter
7B0C Low byte, timed Powerdown counter
7B0D Cursor Blink counter (counts from 80 to FF)
7B0E Cursor Control Parameter: 80 indicates down arrow key required to continue Program execution. Other bits control cursor repeat.
7B0F Key matrix code for depressed key

STRING BUFFER

The 80-byte String Buffer, located 7B10-7B5F, contains the codes for character strings formed by the Parsing of string expressions.

The String Buffer and Output Buffer together form a temporary storage area into which the Previous contents of the Display Buffer are saved, during display of a Program line at an execution halt, or during the display of a BREAK message or RESERVE template.

Certain Printing routines use portions of the String Buffer as a scratch area. Address 7B1F is used as a flag, containing 00 for LLIST and FF for auto Print of the Input Buffer contents in immediate mode.

OUTPUT BUFFER

The 80-byte Output Buffer is located 7B60-7BAF.

The codes for the characters to be displayed in response to a PRINT, PAUSE, or immediate calculation, are stored, in the required format, in addresses 7B60-7B79. (Codes for numeric output are first Placed into the 7A10-34 area, then transferred here to construct the display; codes for string output are transferred from the string buffer.)

With LPRINT, the characters representing numeric data to be Printed are stored, in the required format, in the area 7B80-7B9E. Another Portion of the Output Buffer (ending 7B7F) serves as a scratch area used to form Printed characters. Other locations are used as follows:

7BA9 A line feed (following Printing) is specified by 04
7BAA USING editing character: 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific
7BAB USING number of characters Preceding decimal Point
7BAC USING number of characters in string; 00 if unspecified
7BAD USING number of characters including and following decimal Point
7BAE High byte, saved value of register Y
7BAF Low byte, saved value of register Y

Graphic Printing routines use the area 7B92-7BAD to store coordinates of points specified by LINE or RLINE statements, as well as of the present location. The following addresses are also used:

7B83 Flag: 00, Line; FF, Box
7B84 Count of number of line segments to be drawn
7B85 Flag: 00, LINE; nonzero, RLINE

Prior to recording (or locating) a cassete file, a header is formed in the Output Buffer, located as follows:

7B60-67	Lead-in; consists of the bytes 10, 11, ..., 19
7B68	File type: 00, ML; 01, BASIC; 02, RESERVE; 04, data
7B69-78	File name; null bytes if unspecified
7B79-81	Unused
7B82-83	Beginning memory address of file; not used for data files
7B84-85	Number of bytes in file, less 1; zero for data file
7B86-87	Starting address for automatic execution, ML files only; if not specified, FFFF.

During a load operation, header information read from the recorded file is stored into the Output buffer as follows:

7B91-A0	File name
7BA1-A9	Unused
7BAA-AB	Beginning memory address (used when CLOAD M statement does not specify an address)
7BAC-AD	Number of bytes in file, less 1; zero for data file
7BAE-AF	Starting address for automatic execution of ML Program

INPUT BUFFER

The 80-byte Input Buffer, located 7BB0-7BFF, holds the codes for characters as they are entered from the keyboard. It is also used for storing codes that form input prompts, displayed Program lines, expressions to be evaluated in immediate mode, and displayed RESERVE key phrases.

USABLE ROM SUBROUTINES

The list which follows contains information related to a number of subroutines in ROM that are usable in user machine-language Programs.

A subroutine that is addressable as a Vector Call will be listed as 'Call nn'; the address of the subroutine will follow, in Parentheses.

CLEARING AND MOVING

BLOCKS OF MEMORY

Call BA (F763) clears the contents of memory from addresses X to X+UL. Register Y is not affected.

Subroutine D3C5 clears memory from addresses X to X+U; Register Y is not affected.

Subroutine D3C7 fills memory, from addresses X to X+U, with the byte contained in Register A. Register Y is not affected.

CLEARING OF BUFFERS

Call F2 (EE71) clears the Display Buffer (7600-764D and 7700-774D). Register Y is not affected.

Subroutine EF81 clears the Output Buffer (7B60-7BAF). Register Y is not affected.

Subroutine D02B fills the Input Buffer with 0D codes. Both Register Y and the Input Buffer Pointer (788B) are set to point to 7BB0, the beginning address of the Input Buffer.

DATA TRANSFERS TO BUFFERS

Call 94 (EC5C) copies the contents of addresses X to X+UL-1 into the Output Buffer, starting at the position specified by the Output Buffer Pointer (788F). If insufficient room is available, flag C will be set.

Subroutine EDC1 copies the contents of the Display Buffer into the area containing the String and Output Buffers. Subroutine EDD8 copies these codes back into the Display Buffer.

Subroutine FBCB, with C clear, copies the contents of the String Buffer into the Output Buffer.

Subroutine FBCB, with C set, copies the contents of the Output Buffer into the String Buffer.

FLOATING-POINT ARITHMETIC

The floating-point routines use the BASIC registers located at the following addresses:

R0: 7A00-7A07
R2: 7A10-7A17
R6: 7A30-7A37

Register R0 act as a floating-point accumulator, with R2 containing the second operand. Register R6 is used for temporary storage; except as noted, it is unaffected by the floating-point arithmetic routines.

The 8 bytes in a floating-point register are used as follows:

Byte 0	Exponent (in signed binary)
Byte 1	Sign: 00 = +, 80 = -
Byte 3	1st and 2nd mantissa digits (binary-coded decimal)
Byte 4	3rd and 4th mantissa digits
Byte 5	5th and 6th mantissa digits
Byte 6	7th and 8th mantissa digits
Byte 7	9th and 10th mantissa digits
Byte 8	11th and 12th mantissa digits

PRELOADING OF XH AND YH

In the case of certain indicated routines, XH and YH must contain 7A prior to execution of the routine.

Call 54 (F7B0) will load XH and YH with 7A.

Unless otherwise noted, both XH and YH will contain 7A following execution of any of the floating-point arithmetic routines listed.

DATA TRANSFER BETWEEN R0 AND FIXED VARIABLES

Variable into R0: subroutine DC20. Precede by loading X with initial address of variable. (Following execution, XH and YH will not necessarily contain 7A.)
R0 into variable: subroutine DC0C. Precede by loading X with initial address of variable. Register Y will be unchanged by this routine.

The initial addresses of fixed numeric variables are as follows:

A 7900	E 7920	I 7940	N 7968	R 7988	V 79A8
B 7908	F 7928	J 7948	O 7970	S 7990	W 79B0
C 7910	G 7930	K 7950	P 7978	T 7998	X 79B8
D 7918	H 7938	L 7958	Q 7980	U 79A0	Y 79C0
		M 7960			Z 79C8

CLEARING OF REGISTERS

Clear R0: Call EC (F757). Register Y is unchanged by this routine.
Clear R2: Subroutine F753. (XH and YH must contain 7A.)

DATA TRANSFER BETWEEN REGISTERS

R0 into R2: Call E6 (F70D).
R0 into R6: Call 80 (F707).

R2 into R0: Call 56 (F73D). (XH and YH must contain 7A.)
R2 into R6: Subroutine F701. (XH and YH must contain 7A.)

R6 into R0: Subroutine F737.
R6 into R2: Call 68 (F715). (XH and YH must contain 7A.)

Exchange R0, R2: Call 66 (F7B9). (XH and YH must contain 7A.)
Exchange R0, R6: Call 64 (F7B5). (XH and YH must contain 7A.)

DATA TRANSFER BETWEEN R0 AND BASIC STACK

The BASIC stack, located 7A3C-7AFF, permits up to 24 PUSHes of R0.

PUSH R0 onto BASIC stack: Subroutine DBF5.
POP R0 from BASIC stack: Call 30 (DC16)

ARITHMETIC OPERATIONS

All calculated results are normalized, with underflow to zero; each contains a 12-digit mantissa. Calculation of trigonometric functions and their inverses is based on the current trigonometric mode setting.

If an overflow results or an illegal operation is attempted, each of these routines will terminate with flag C set, and an error code in UH. Included are ERROR 37 (overflow), ERROR 38 (division by zero), and ERROR 39 (illegal function input argument).

R0+R2 into R0: Call F0 (EFBA)
R0-R2 into R0: Subroutine EFB6
R0*R2 into R0: Call 7E (F01A)
R0/R2 into R0: Call 58 (F084)
R0↑R2 into R0: Subroutine F89C (Register R6 used)

R0*R0 into R0: Subroutine F019
1/R0 into R0: Call 6E (F080). (XH and YH must contain 7A.)
SQR(R0) into R0: Subroutine F0E9

ABS(R0) into R0: Subroutine F597 (X, Y, and R2 will be unchanged.)
INT(R0)into R0: Subroutine F5BE
SGN(R0) into R0: Subroutine F59D (Y and R2 will be unchanged.)

Round R0 to ten digits: Subroutine F932 (Y and R2 will be unchanged.)

3.14159265359 (PI) into R0: Subroutine F5B5. (PI also stored into R2)

LN(R0) into R0: Subroutine F161
LOG(R0) into R0: Subroutine F165
EXP(R0) into R0: Subroutine F1CB
10↑R0 into R0: Subroutine F1D4

SIN(R0) into R0: Subroutine F3A2 (Register R6 used)
COS(R0) into R0: Subroutine F391 (Register R6 used)
TAN(R0) into R0: Subroutine F39E
ASN(R0) into R0: Subroutine F49A (Register R6 used)
ACS(R0) into R0: Subroutine F492 (Register R6 used)
ATN(R0) into R0: Subroutine F496 (Precede by clearing flag C)

DEG(R0) into R0: Subroutine F531 (Register R6 used)
DMS(R0) into R0: Subroutine F564 (Register R6 used)

COMPARISONS

Subroutine D0D2 performs a comparison of R0 and R2. The comparison performed will depend on the contents of A, as follows:

A=00:	R0<>R2	A=04:	R0=R2
A=01:	R0<R2	A=05:	R0<=R2
A=02:	R0>R2	A=06:	R0>=R2

If the test condition is met, R0 will contain 1, with flag Z clear; otherwise, R0 will contain zero, with flag Z set. Following execution, YH will not necessarily contain 7A.

CONSTANTS (FROM ROM TABLE) INTO R2

In each routine, XH and YH must contain 7A prior to execution.

1 into R2: Call 6A (F08F)
3.14159265359 (PI) into R2: Subroutine F875

57.2957795131 (180/PI) into R2: Subroutine F866
0.434294481903 (1/LN 10) into R2: Subroutine F87B
0.6 into R2: Call 62 (F88B)
0.9 into R2: Subroutine F87F
90 into R2: Subroutine F883
180 into R2: Subroutine F887

SEPARATION OF INTEGER AND FRACTION

Call 60 (F6B4) will separate the integral and fractional parts of R0, placing the integral part into R0 and the fractional part into R2. Neither result will be normalized. The given value of R0 is taken to be positive. XH and YH must contain 7A prior to execution.

REMOVAL OF SIGN

Call 6C (F6FB) loads the sign byte of R0 into A, and makes R0 positive. XH and YH must contain 7A prior to execution.

NORMALIZATION OF R0

Either of the subroutines Call E8 (F661) or Call 52 (F663) will normalize the contents of R0. The former applies a positive sign to the result; the latter takes the contents of A as the sign byte. In both routines, underflow results in zero; overflow sets flag C, with the code for ERROR 37 in Register UH.

Register XH must contain 7A prior to execution. Register Y is not changed by either routine.

BINARY-TO-DECIMAL CONVERSION

A two-byte binary number contained in Register U is converted to its decimal equivalent by execution of Call 10 (DD2D)\ The instruction calling this subroutine must be followed by a data byte, which specifies the effect produced. Two options exist: the converted value may be stored into R0; or, character codes for its display may be stored into memory, beginning at the address contained in Register Y.

Data byte	Interpretation of U	Disposition of result
00	unsigned binary	stored into R0
40	unsigned binary	characters stored, no sign
60	unsigned binary	characters stored, with sign
80	signed binary	stored into R0
C0	signed binary	characters stored, no sign
E0	signed binary	characters stored, with sign

Register YH will not necessarily contain 7A following execution.

DECIMAL-TO-BINARY CONVERSION

The binary equivalent of the contents of R0 (ignoring fractions) is calculated and stored into Register U by Call D0 (D9F9). The result is also stored into R0 (in binary format). A will equal UL.

Two data bytes must follow the instruction calling this subroutine.

The first of these specifies the range of values permitted; the second is added to the return address when the contents of R0 are not within that range, in which case UH will contain the code for ERROR 19.

Tabulated below are the permitted ranges (in decimal) for given values of the first passed byte. (When negative values are permitted, results are in signed binary.)

00:	0 to 65535	0A:	0 to 155
02:	0 to 65279	0C:	0 to 80
04:	-32768 to 32767	0E:	0 to 26
06:	0 to 32767	10:	0 to 25
08:	0 to 255		

If any of the above passed bytes is increased by 1, zero will be eliminated as a permitted value.

BINARY ROUTINES

STORAGE AND RECALL OF TWO-BYTE DATA

Call F6 (DDB5), followed by data bytes nn nn, stores UH into address nnnn and UL into nnnn+1. Following execution, A will contain UL; X will contain nnnn+1; Y and U are unchanged.

Call F4 (DBBC), followed by data bytes nn nn, loads UH with the contents of nnnn and UL with the contents of nnnn+1. A will contain the same byte as UH; X and Y are unchanged.

Call CA (C001), followed by data byte nn, stores XH into address 78nn and XL into 78nn+1. Following execution, A will contain XL; U will contain 78nn+1; X and Y are unchanged.

Call CC (DDC8), followed by data byte nn, loads XH with the contents of 78nn and XL with the contents of 78nn+1. A will contain the same byte as XH; Y and U are unchanged.

OPERATIONS IN BINARY

Subroutine DAA8 will replace U by its complement. (This is equivalent to replacing U by &10000-U.) X and Y are unchanged.

Subroutine DFE2 replaces U by U-X. If the result is negative, flag C is cleared, with UH containing 16. X and Y are unchanged.

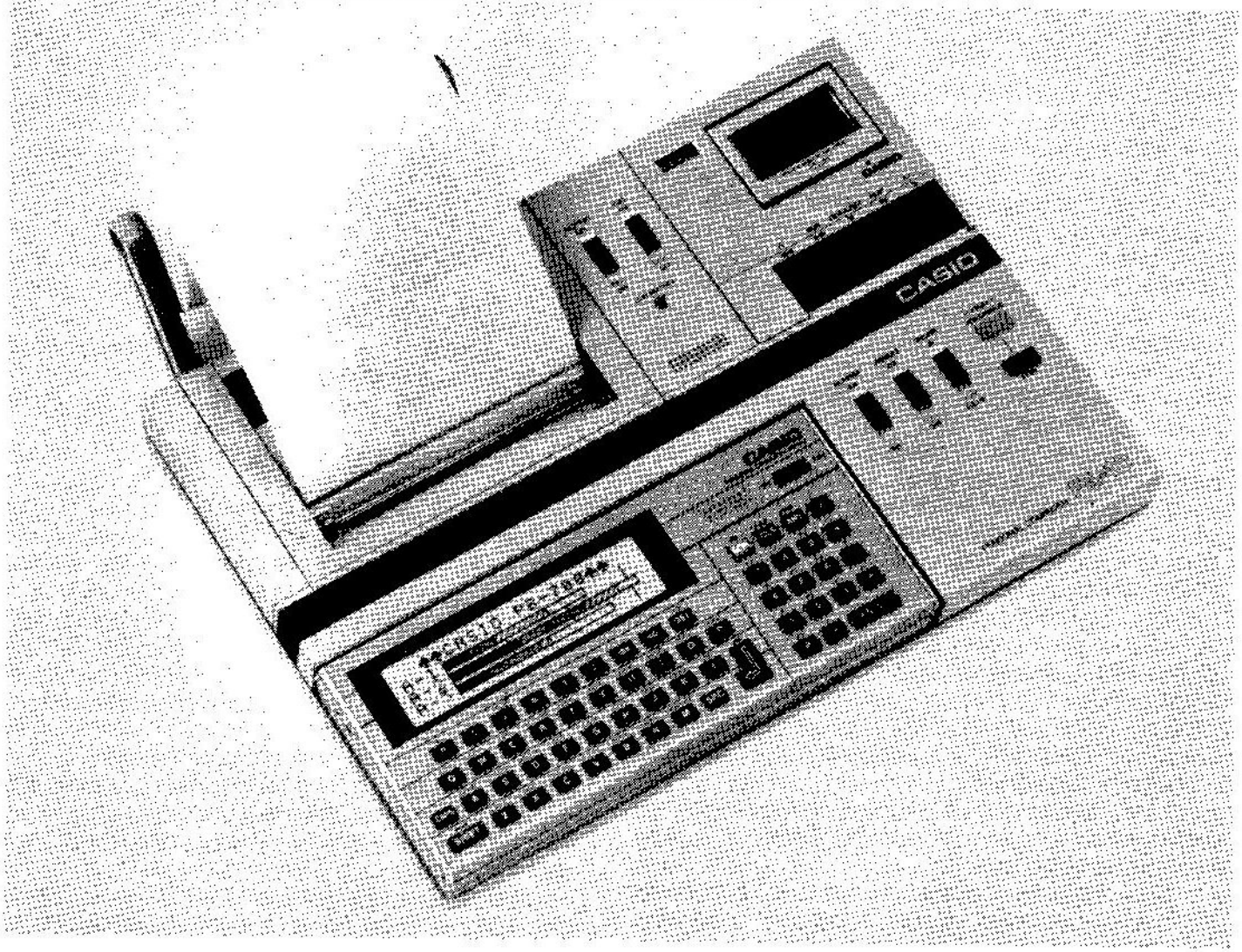
Call 50 (DA71) replaces Y by U*Y; the result is also placed into X. If the result exceeds FFFF, flag C is set. Contents of U are lost.

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

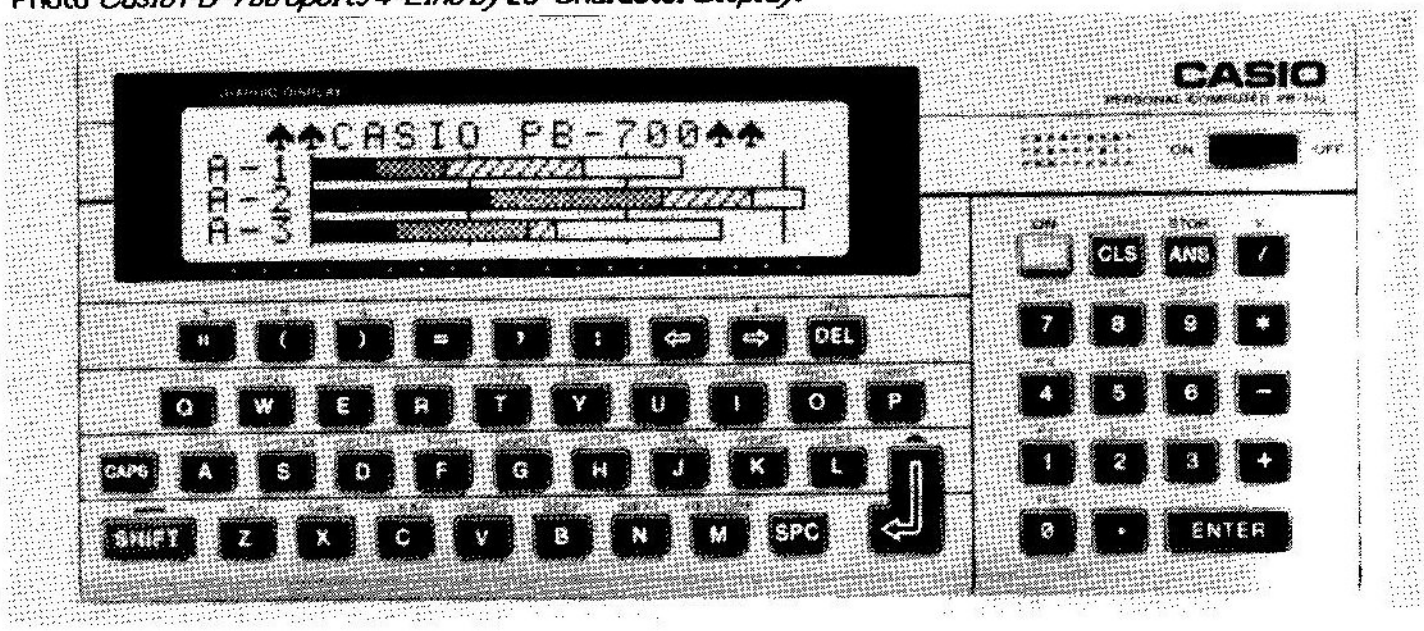
Issue 33 - May/June

Photo *Casio PB-700 Mounted on FA-10 Equipped with CM-1 Micro Cassette Module.*



Another *Personal Information*™ product.

Photo *Casio PB-700 Sports 4-Line by 20-Character Display.*



CASIO INTRODUCES PB-700 HAND-HELD

Casio, Inc., recently introduced a hand-held computer that sports 4K of user memory. Memory is expandable in 4K increments to 16K. It also features a 4-line by 20-character display capable of printing both upper- and lower-case letters as well as graphic symbols. Alternately, the display can be addressed on a pixel-by-pixel basis over its 160 by 32 dot range, making it suitable for the presentation of graphs, charts, and simple pictorials.

The new unit has a 58 keys including a numeric keypad. Commonly used BASIC keywords are accessible through a shifted key feature. The PB-700 has full editing features that facilitate the keying in of a BASIC program.

The unit includes an audio annunciator.

It runs for approximately 100 hours (actual display time) on a set of 4 AA cells. User RAM memory is backed-up by a separate lithium battery. Main battery power is conserved by an automatic shutdown feature that turns off the display after approximately 8 minutes of computer inactivity.

The BASIC interpreter and general system operating software is contained in 25K of ROM installed in the PB-700.

The hand-held unit measure 7/8 by 7-7/8 by 3-1/2 inches and weighs approximately 11 ounces with batteries installed.

The machine includes a number of built-in mathematical functions such as trigonometric and reverse-trigonometric, logarithmic, powers, exponentials, square roots, rounding, random numbers and others. Its command set includes: CONT, DELETE, EDIT, LIST, LLIST, LOAD, NEW, PASS, PROG, RUN, SAVE, SYSTEM TROFF, TRON and VERIFY. BASIC keywords include: ANGLE, BEEP, CHAIN, CLEAR, CLS, DATA, DIM, DRAW, DRAWC, END, ERASE, FOR-TO-STEP, NEXT, GET, GOSUB, GOTO, IF-THEN-ELSE, INPUT, LET, LOCATE, LPRINT, PRINT, PUT, READ, REM, RESTORE, RETURN and STOP. String functions include: ASC, CHR$, LEFT$, LEN, MID$, RIGHT$, STR$ and VAL. Other functions provided are: INKEY$, POINT, TAB and USING.

Up to 10 separate programs or routines may be stored in memory simultaneously.

The list price of the PB-700 in the U.S. is said to be $199.95.

The PB-700 can be expanded into a larger system by using the Casio FA-10 plotter/cassette interface unit. The FA-10 has a 4-1/2 inch wide four color printer/plotter. Up to 80 characters can be printed on a single line. It can hold a small supply of paper or utilize a large external roll of paper. A cassette interface allows data transfers to take place between the PB-700 and a standard external cassette recorder. The FA-10 is powered by internal rechargeable batteries. The list price in the U.S. is stated as $249.95.

An optional micro cassette module can be mounted on the FA-10 to provide integral data storage/recovery. This module is referred to as the CM-1 and has a suggested retail price of $89.95 in the United States.

An alternative to the FA-10 is the FA-4 interface. It contains a cassette interface and a Centronics port enabling the unit to send data to a variety of external printers. The FA-4 interface lists at a U.S. price of $69.95.

HP-71B PRODUCT REVIEW

Hewlett-Packard has finally produced a computer that will fit in a pocket. Well, let's say a good-sized pocket, such as a jacket pocket. The unit is close in size and weight to a Sharp PC-1500 or Radio Shack PC-2. Press releases incorrectly reported that the unit was just one-half an inch thick. That is simply not true. It is about one inch in thickness at the back, sloping down to 3/4-inch at the front.

The unit is powered by four AAA cells which will supposedly operate the unit for approximately 60 hours of "on" time or several months of normal usage. Alternately, the machine may be powered by an a.c. adaptor (sold separately).

In essence, the HP-71B is a stripped down and compacted HP-75. In its unadorned state it has just about the same capabilities of the 75 sans the magnetic strip card reader, the HP-IL interface, and a few software facilities, such as the 75's appointment/alarm scheduler. The strip card reader and HP-IL interface can be added as extra cost options. With its built-in clock and calendar functions, you can program your own appointment and alarm utilities.

However, it appears that HP learned some things from experience with the earlier HP-75. For instance, a rather disconcerting feature of the 75 was the fact that you had to hold down its shift key while you punched the desired "shifted" key. On the HP-71B this nuisance has been removed. Once tapped, a shift key determines the mode of the next key to be pressed. There are two shift keys on the 71B to facilitate packing the keyboard with a lot of easily-accessed features.

The keyboard is arranged in QWERTY format. It is too small for touch-typing, hence the keys are in the traditional HP calculator style, with sloping fronts. Some of the keys have a secondary legend on the sloping front portion. A third set of legends is shown directly above the keys. The second and third legends are accessed via the two shift keys (colored blue and yellow to correspond with the key legends). The keyboard feel is virtually identical to that of a HP-41 series calculator.

The display is about four inches in length. The central portion of the display, consisting of a dot matrix, is about 3 inches in length and 1/4-inch high. It is flanked at each end by a number of special annunciators. The dot matrix can present up to 22 characters (with scrolling across a line of up to 96 characters). Alternately, it may be operated in a graphics mode. The display intensity (reflectivity) may be adjusted through a software command.

Software Powerhouse

The HP-71B actually packs more software power than the considerably larger 75. It comes equipped with 64K of ROM (versus the 75's 48K) and 17.5K of RAM. About 1K of RAM is used by the operating system, but this still leaves a healthy 16+ kilobytes for the user in a stock model. (Currently, RAM can be expanded to 32K by adding 4K modules which are available at extra cost.)

Within the 64K of ROM is a powerful version of HP BASIC containing some 240 statements, commands and functions. Perhaps the most useful feature of the 71B is its powerful file-handling capabilities. All programs and data bases are accessed as independent, memory-resident files. You can have countless files of various types in memory all at one time. A series of programs can all access a common data file. Programs can be chained together. Even better, one program can be "called" (and recursively at that) by another. There is complete control over variables (including the deletion of individual variables or arrays). These are powerful capabilities and a real joy to have available on such a physically small unit. (How many of us wish such refinements were available on desktop-sized computers?)

To give you an idea of the power of the HP-71B, here is a list of the keywords available to the user along with brief comments on some of the more unusual or particularly powerful ones: ABS, ACOS, ADD (used by the built-in statistics package), ADDR$ (returns hexadecimal address of a file), ADJABS, ADJUST and AF (these three keywords are used to adjust the system's clock), AND, ANGLE (polar), ASIN, ASSIGN #, ATAN, AUTO, BEEP and BEEP OFF/ON, BYE (power-down), CALL (true call to another subprogram where variables can be local or global), CAT, CAT$, CEIL (ceiling), CFLAG (clears user or system flags), CHAIN, CHARSET (used to define alternate character set), CHARSET$, CHR$, CLAIM PORT (used to reclaim RAM memory that had been assigned to particular RAM modules), CLASS, CLSTAT (clears statistical array), CONT, CONTRAST (display intensity), COPY (files), CORR (correlation function used in statistics package), COS, CREATE, DATA, DATE (current calendar date), DATE$, DEF FN (define single- or multiple-line user-defined functions), DEF KEY, DEFAULT EXTEND/ON/OFF (sets math exception traps), DEG, DEGREES, DELAY (sets line and character scroll rates), DELETE (lines), DESTROY (delete variable(s) and arrays from memory), DIM, DISP (display on the LCD screen), DISP USING, DISP$, DIV (integer quotient division),

DROP (a pair of coordinates in a data set used by the statistical package), DTH$ (convert decimal to hexadecimal), DVZ (status of divide-by-zero flag), EDIT, END, END DEF/SUB (terminate function definition or subprogram), ENDLINE, ENG (set engineering display format), EPS (epsilon), ERRL, ERRM$, ERRN, EXACT (calibrate system clock), EXOR, EXP, EXPM-1 (natural antilog minus 1), EXPONENT, FACT (factorial), FETCH (a line within current file), FETCH KEY, FIX (fixed format), FLAG (returns condition of a specified flag), FLOOR, FN (executes user-defined function), FOR...NEXT, FP (fractional part), FREE PORT (transfers RAM in a module from main memory to independent use), GDISP (graphic display), GDISP$, GOSUB, GOTO, HTD (hexadecimal to decimal), IF...THEN...ELSE, IMAGE, INF, INPUT, INT, INTEGER, INX (inexact flag status), IP (integer part), IVL (invalid flag status), KEY$, KEYDEF$, KEYDOWN (indicates whether a key is being pressed), LC (upper/lower case toggle), LEN, LET, LINPUT (input an entire line at a time), LIST, LOCK, LOG (natural logarithm), LOGP1 (logarithm of argument plus 1), LOG10, LR (linear regression), MAX, MAXREAL, MEAN, MEM, MERGE (part or all of a file), MIN, MINREAL, MOD (modulo), NAME (a file), NAN (not-a-number function), NOT, NUM, OFF (turns off the unit), OFF ERROR/TIMER (deactivates previous ON ERROR or ON TIMER # statement), ON ERROR GOSUB/GOTO (error trap), ON...GOSUB/GOTO/RESTORE, ON TIMER # (used to enable any of three separate timers), OPTION ANGLE/BASE/ROUND, OR, OVF (overflow flag), PAUSE (suspends program execution), PEEK$, PI, PLIST, POKE, POP (subroutine stack to cancel next return instruction), POS (substring position), PREDV (predicted value, used as a statistical function), PRINT, PRINT USING, PRINT #, PRIVATE (provides file protection), PROTECT (a magnetic card track), PURGE (files), PUT (character input), PWIDTH (defines line length), RAD, RADIANS, RANDOMIZE, READ, READ #, REAL, RED (reduction to return a remainder), REM, RENAME (a file), RENUMBER (any part or all of a program file), RES (result), RESET/RESET CLOCK (resets system flags and clock adjustment factor), RESTORE, RESTORE #, RETURN, RMD (returns the remainder), RND (generates a random number), RUN, SCI (scientific notation), SDEV (standard deviation), SECURE (protects a file from alteration or purging), SETDATE, SETTIME, SFLAG (set flag), SGN, SHORT (defines 5-digit precision), SHOW PORT (displays type and size of all plug-in memory devices), SIN, SQR, STARTUP (command string that is executed when unit is turned on), STAT, STD, STOP, STR$, SUB (denotes beginning of a subprogram), TAN, TIME, TIME$, TOTAL (of a particular variable in a statistical array), TRACE FLOW/VARS/OFF (debugging aid), TRANSFORM (text files to program files and vice versa), TRAP, UNF, UNPROTECT (remove write-protection from a magnetic card track), UNSECURE (a file), UPRC$ (convert lower case letters to upper case), USER (activate/deactivate user-defined keys), VAL, VER$, WAIT, WIDTH, and WINDOW.

As you may infer from the instruction set, this is a rich and powerful implementation with many subtleties. For example, while the PEEK$ function will allow you to examine memory (by nibbles, since the HP-71B is a nibble-oriented machine), it does not allow you to examine a file that has been designated a "private" file. The built-in statistical functions will be of value to many mathematicians and analysists and save a great deal of work in that area. People with special needs can create their own symbol sets and redefine the keyboard to reflect their requirements. The file handling capabilities are superb. In summary, the 71B sports an instruction set that will likely please even hardened programming professionals.

In fact, the instruction set is so rich that you have to be careful not to trip yourself up. For instance, you can control the length of time a message is displayed on the unit's screen through use of the DELAY statement. This delay can range from fractions of a second to infinity. Of course, when you use a delay of infinity, you are really opting to have the message stay on the screen until the user presses the END LINE key. Now the HP-71B has another feature called a "type-ahead" keyboard buffer that allows you to enter information before an input prompt even appears on the screen. Guess what happens if you use a DISPlay statement with a DELAY of infinity and later use an input statement? It appears that the END LINE input used to terminate the DISPlay gets placed in the "type-ahead" keyboard buffer. When the following input statement is processed, it finds that same END LINE character already in the keyboard buffer and assumes that it means the end of the input. Ooops! One way to get around this situation is to execute a statement such as setting a dummy variable equal to the function KEY$ just before executing the INPUT directive. The KEY$ function clears out the keyboard buffer, thus knocking out the troublesome END LINE character.

Add-Ons

Two optional devices that fit within the case of the HP-71B provide it with the capability to rival the HP-75. The first such add-on is for transferring information to and from magnetic strip cards. The

second is an HP-IL (Hewlett-Packard Interface Loop) adaptor.

The magnetic strip card device allows the unit to use magnetic strip-cards. These cards are about 10 inches long and 3/8-inches in width. Each card contains two 650-byte tracks. Processing both sides of a card takes in the order of 15 seconds. It is a convenient way of storing relatively small programs and blocks of data.

Adding the HP-IL adaptor allows the unit to communicate with a variety of peripheral devices ranging from video drivers, to several types of printers and plotters, a mass storage digital tape drive, and other devices such as measuring and test instruments. It is a relatively simple matter, using such peripherals, to have the 71 serve as a data logger, etc. The HP-IL driver software is provided as part of the system software.

The HP-71B is definitely a first-class product. If you have professional requirements and can afford its top-of-the-line price (about $500) you will likely find it an instrument that provides a great deal of a special kind of satisfaction: the satisfaction that comes from owning a product that exhibits far above average quality and capability.

REVIEW OF ROM SOFTWARE MODULES

The following reviews and commentaries were provided by: *Robert Sutliff, 55 North Avenue, New Rochelle, NY 10805.* Readers are advised that the opinions stated herein are those of Mr. Sutliff and do not necessarily reflect the views of *PCN* or its editorial staff.

Introduction

At the end of 1983, Sharp Corporation released its long awaited line of Software Libraries for the PC-1500 and/or PC-1500A. Each module is the size of a CE-155 RAM module and occupies the expansion port in the back of the computer. Only one module can be installed at a time. Denoted the CE-500 series, each module adds 16K of ROM memory, at hexadecimal addresses 0000H through 3FFFh. Although the ending address of actual content for each module is different, the remaining space cannot be used since it is ROM. The Reserve memory space is controlled by the module, so that in the PC-1500 the usual Reserve area is left free for RAM, thereby giving a total of 2047 bytes of user memory in addresses 4000H through 47FFH.

Each module contains programs and subroutines, written in BASIC by American Micro Products, Inc., which are addressed by softkeys. Each module comes with a detailed manual providing operating instructions and examples. Modules can be found through mail order houses and dealers ranging at anywhere from $25 to $60 each, so a buyer should shop around! The main advantage of a module is that 16K of programs are loaded by a quick installation, as opposed to the lengthy cassette loading procedure. Access time to any program in the module is instantaneous. The programs in any module (except SHARPCALC) can be (L)LISTed in the usual fashion, although no lines can be altered. The ROM programs act as though they have been MERGEd into current memory. That is, new lines in RAM appear after the ROM programs when LISTed. Programs in RAM must be labeled and run by using the label and DEF or GOTO. The CE-150 printer must be used for certain modules; for others, it is an optional device and printer related prompts will not appear if it is unattached. The modules use the Reserve keys for program selection and provide extensive prompts, default values, and recovery-from-error traps. In particular, default values often flash and make it easy to rerun a problem by changing only certain parameters. Some of the modules are excellent in covering their subject matter, while others appear deficient and contain programming errors. The following discussion reviews several of the modules.

Graphics Development (CE-501A)

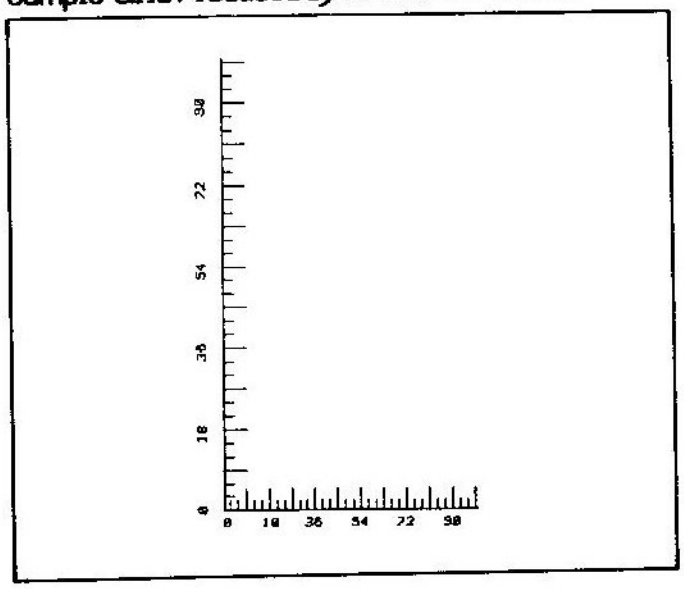
The Graphics Development module contains many subroutines which aid the user in generating graphs in X-Y coordinates, as well as two stand-alone plotting programs. The subroutines are so extensive that they cannot all be listed here. However, they can be grouped to show their general purpose as follows:

1. Graphics Initiation Subroutines. These three routines build a physical description of the graph to be drawn by later routines. How much of the plotter paper is to be used, the scale of the axes, and fixed decimal places of numeric labels are all selected in this section.

2. Plotting Initiation Section. Seven routines allow the user to describe the graph paper and draw it. The user specifies the intercept values, extreme values, axes colors, major and minor tic mark spacing, tic mark lengths, numeric label intervals and colors, label direction, grid line specifications, and framing limits. This provides almost limitless variations in producing a graph paper background for a plot. By writing a program in RAM one can call these subroutines to draw the graph paper and

then plot a scaled function on the graph. The routines are arranged in such a way that LAXIS and LGRID call all the others in this section. In LAXIS, note that the labeling as described in the manual does not work unless the axes have been drawn along the lower and left boundaries; that is, unless y(int)=min(y) and x(int)=min(x). If not, only the labels at the ends of the axes are printed. Also, the axes are drawn wherever x(int) and y(int) specify; they do not necessarily establish the origin. The main use for these routines would seem to be to aid in producing a graphing area on which to plot functions or curves. Unfortunately, regaining your coordinates (through "CURSOR") and scaling a curve to your graph is tedious work. The authors have made it very simple to produce a neat graphing area but difficult to graph on it! It is

Sample *Grid Produced by CE-501A Module.*



almost tempting to tear off the graph paper area and plot on it by hand, a somewhat self-defeating idea. Some this is saved, however, by MPLOT.

3. Plotting Programs. These are two very useful function plotting programs called MPLOT and PLOT. MPLOT is a macroplotter which offers most (but not all) of the options described in section 2 above, except that a grid pattern is always used. An error in line 20290 ignores the input color for the graph area and always selects COLOR 0. The user puts almost any number of functions to be simultaneously plotted into program memory. Unfortunately, the form of a function is "FX=" followed by an expression where the independent variable is the letter I. Thus, to plot f(x)=x*sin(x) the function is entered as FX= I * SIN(I). Confusing! But MPLOT is still impressive. A macroplotter allows boundaries for the plot to extend beyond the limits of the printer page. The graph is then produced in labeled sections which the user detaches, cuts apart, and then pastes together to form a plot of almost any size! When someone complains that the CE-150 plotter is too small to be of any use, an output of MPLOT is impressive to wave in front of them. Alternate colors are used for each function, so be sure that all pens are loaded. Plotting can be done in a continuous or point-by-point mode. The number of points plotted depends on the INCREMENT value. Note that this value should be in relatives values of x based on min(x) and max(x).

PLOT is an altogether different plotter. Several functions can be entered, as well as a few parameters, as in MPLOT. Most of the labeling and graph area is selected automatically, drawn, and then the functions are plotted in this single area. (This program is the quickest way to plot a function in this library.) Once the plot is completed, the prompt 'WINDOW?" appears. By entering the value defining a rectangular area (the window) one can selectively enlarge areas of the plot! The plot is redone, using the window as the new graph area. Again, a smaller window can be selected, enlarged, and the process continued. Since the axes are continually relabeled, one can study the behavior of a function in any neighborhood, conduct root searches, study function intersections, etc. All pens must be loaded since all are used.

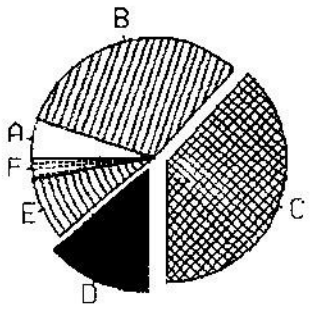
The Graphics Development Library is accompanies by of of the best manuals in the module series. The documentation of subroutines, labels, variables and parameters is extensive. Many examples are given. This module offers extensive exercises in graphics development, though its chief assets are MPLOT and PLOT.

Business Graphics (CE-501B)

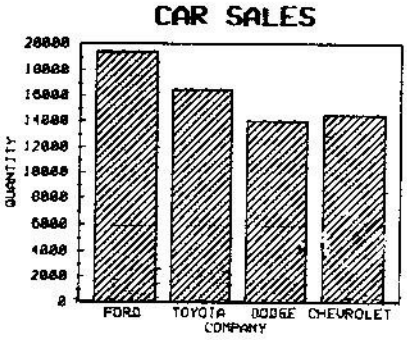
This module produces excellent business charts which use the full capabilities of the CE-150 plotter. It also includes a curve fitting program! Though the graphs produced are indeed small, they provide vual interpretations usually provided only to those with access to a graphics department. These little graphs can also be enlarged by a reproduction center. The main attribute of the Business Graphics Library is its eas of use. The programs include:

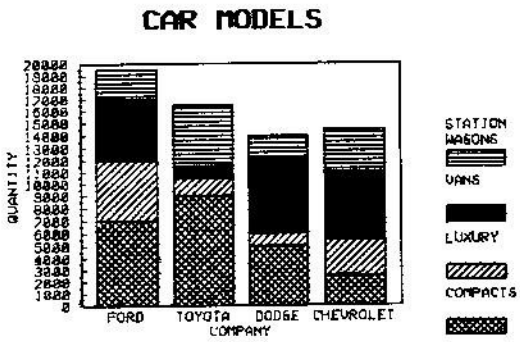
1. Pie Chart. The charts can be selected as plain (black outline, blank slices) or multicolor (allowing for several colors and shadings). The user selects the title and data for each slice, and whether or not each slice is to be "exploded"

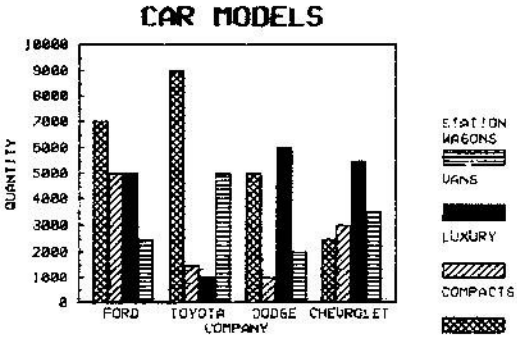
Sample *Outputs from Business Graphics (CE-5018) Module.*

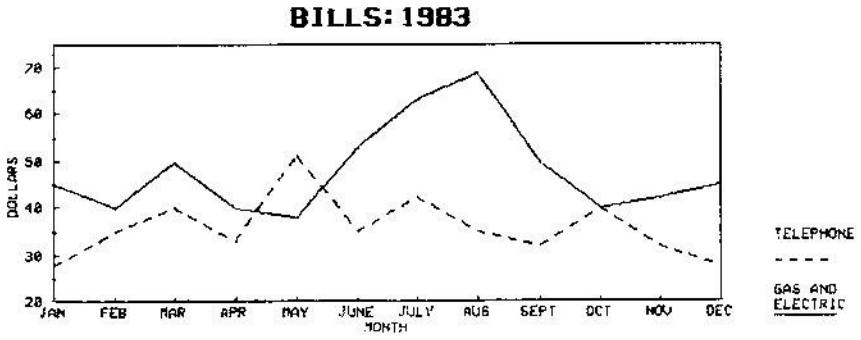


A: ZEBRAS	20	5.27%
B: MONKEYS	120	31.66%
C: SNAKES	145	38.25%
D: DEER	52	13.72%
E: BEARS	30	7.31%
F: LIONS	12	3.16%
TOTAL:	379	100.00%







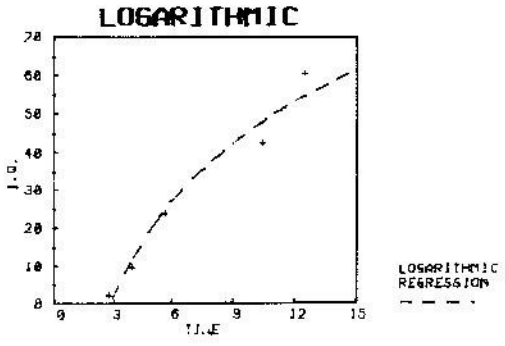


LOGARITHMIC REGRESSION

	DF	SS	MS	F
TOTAL	4	2304.2		
REG	1	2217.9	2217.9	
RESID	3	86.2	28.7	7.7E 01

R^2 = 0.9625

$\hat{Y}$ =(-3.9194E 01) +(3.7247E 01) LN(X)



(slightly offset). In multicolor, each slice can have one of four colors and one of six types of shadings. Once the choices are made, the chart is automatically drawn proportionally to the data, and a table of the data and percentages is listed.

2. Bar Chart. As in the pie charts, any bar can have any selected color and shade. The user has complete control over the titles which appear for the total graph, two axes, and each bar. Bar charts can be chosen to be normal, stacked, or grouped. Usually, selection is made based on the type of comparison of data to be made. The maximum number of data sets is 56. The maximum character length for each title in each set of x-data is 8 to insure no title overlapping. Longer titles may be used if adjacent titles are shorter. It is very impressive to simply input the titles and data for each set and to see such a precise graph be drawn so quickly!

3. Line Chart. Up to 12 lines may be plotted simultaneously in 9 different dash patterns. Up to 53 data points may be plotted per line (depending on the number of lines drawn).

4. Point Chart. The user can create a scatter chart by inputting up to 81 points using (X,Y) coordinates. Each point is plotted by an isolated dot. It is useful to make a point chart to visualize which type of curve fit best suits the data. The data may also be read from or save to a cassette file.

5. Curve Fitting. The user inputs data and can select any or all of the following analyses: linear, power, exponential or logarithmic. Associated statistics are printed and a graph of the data and fitted curve is plotted, along with the equation of the curve. Note that the x values *must* be entered in ascending order. Any x-value that is not in the proper order will be omitted. Again, cassette files may be used.

The Business Graphics Library is an excellent module for its range of offerings and its ease of use. In fact, Sharp now has on the market a "calculator/plotter" which does the exact same plots as this library, at almost the same cost as the PC-1500!

Math (CE-505A)

This module tackles some computational problems in higher mathematics and yet contains more programming errors than any other module. Numerical input allows for the entry of pi, scientifically notated, arithmetic, or function values. The programs include:

1. Prime Factors.

2. Gamma Function. Note that the input *n* must have a value less than 71. This program can be used to compute factorials.

3. Polynomial Multiplication.

4. Hyperbolic Functions. Note that the input must be less than 231. Inverse functions are also computed.

5. Relative Minimum of a Function. There is often compounded rounding error. Repeating minimal values are not detected. The endpoints of a given interval are not considered in detecting a minimal value.

6. Matrix Arithmetic including Matrix Addition, Multiplication, Inversion, Determinant, and Scalar Multiplication. The maximum allowable matrix dimension in the Determinant and Inverse programs is ten. There are also programming errors in these two programs that cause the "SINGULAR MATRIX" error to appear even when there should be no error. In the Determinant program, line 5410 announces this error if there is a zero on the diagonal from row 2 to row n of the input matrix. In the Inverse program, line 5527 detects an error if any element on the diagonal is zero in the inverse matrix as it is being constructed; this certainly occurs if the input matrix element A(1,1) is zero. These are serious flaws, as the input matrix might not necessarily be singular. A possible solution is to remove all zeros from the diagonal by row and column operations (which do not affect the determinant) in the original matrix before it is input.

7. Ordinary Differential Equations (First and Second Order). The documentation is vague in this section.

8. Roots of an Equation (f(x)=0). Newton's Method is used to approximate roots with user defined precision. The user must input f(x) and, optionally, f'(x).

9. Linear Systems of Equations. Annoyingly, the order of entry of elements of the coefficient matrix is reversed from the input of the Matrix programs. Also, if a program yields a "NOT SOLVABLE" message, one cannot retry the problem using the "EXIT(N)" option because the FOR/NEXT stack is not cleared, thereby causing an ERROR 2 in line 20045. Instead, one must exit the program and restart it. But more importantly, this program suffers from the same problems found in the Matrices programs. Upon input of a system to which a solution *does* exist, the program may signal a "NOT SOLVABLE" message! However, when the order of equations is switched, or a single equation is multiplied by a constant, the correct solution may appear. This is due, once again, to an error in line 21280 which causes the program to halt whenever a zero appears on the diagonal of the reduced augmented matrix as it is being constructed. The problem is the programmers'

algorithm for producing the reduced matrix which does not guarantee the correct ordering of the rows. The error trap is therefore incorrect. Also, their matrix reduction is halted when n-1 of the n variables are discovered; this leads to values for X(n) which do not necessarily satisfy the n'th equation! These serious flaws render this program almost useless. If the program signals "NOT SOLVABLE", this may not be the case. If it does yield a solution, this solution should be checked in all the equations.

10. Non-Linear Systems of Equations. Two non-linear systems in two unknowns are solved by Newton's approximation method. The user enters the functions, their partial derivatives, and a degree of precision. Enter the functions as the instructions on page 85 of the manual show, not as in the examples.

11. Integration. To avoid a common problem of numerical integration, investigate the asymptotic nature of the function being integrated. For example, y = 1/(x-1)^2 integrated from 0 to 2 will give an error n=5, but a numerical value when n=10, while the true value is infinite.

12. Cubic Spline Interpolation. The interpolation point x must be intermediate between x(1) and x(n). Also, the entry of the value y" at x(1) and x(n) are crucial. They should not simply be taken to zero, as in the examples.

It should be noted that the 2047 bytes of RAM are used extensively by a number of the programs. Functions are entered as program lines with confusing labels. Why couldn't a simple parsing routine have been used for function input, as Radio Shack did in its Plotter program? The manual lacks documentation as to variable usage and uses conflicting notation. The Mathematics Library often misses its mark as a useful tool.

ADDRESSING PIXELS ON THE PC-1500

This program was submitted by: *Patrick L. Pollet, University of Petroleum and Minerals, Department of Chemistry, UPM Box 298, Dhahran International Airport, P.O. Box 144, Saudi Arabia.*

one of the nice features of the PC-1500 is the possibility of producing graphics using the GPRINT instruction. Almost unlimited applications could be thought of, such as graphics games, sophisticated messages or, more seriously, redefining the keyboard by adding new characters such as French accentuated vowels or Greek or Russian alphabets.

Other than "try-and-hit" techniques, the only way to get the GPRINT hexadecimal codes for a desired pattern is to draw it on a 7 by 156 grid and then translate into hexadecimal using the "bit pattern" of every column. That process is quite long and prone to errors.

The GPRINTAID ROUTINE provided here gives you the possibility of addressing every individual dot of the screen with a flashing cursor that can be moved in any of 4 directions by pressing the following numeric keys:

"4"	cursor left by one row
"2"	cursor down by one line
"6"	cursor right by one row
"8"	cursor up by one line

Program: *GPRINTAID*

```
3A90 58 7A 5A 00   3ADC 2C 93 29 B7   3B2C B5 40 AE 79   3B78 B7 3A 81 02
3A94 6A 9B B5 00   3AE0 0D 8B 6F B7   3B30 00 BD FF AE   3B7C B3 06 28 24
3A98 51 88 03 E9   3AE4 36 89 0F A5   3B34 79 01 91 8F   3B80 B9 0F BB 30
3A9C 78 75 00 B5   3AE8 78 75 F9 B3   3B38 9E 40 B7 35   3B84 B7 3A 81 02
3AA0 01 AC 79 00   3AEC 01 B7 9C 93   3B3C 89 07 15 AB   3B88 B3 06 2A 48
3AA4 BD FF AE 79   3AF0 3B AE 78 75   3B40 79 00 1E 9E   3B8C 7B 4A 7F B5
3AA8 01 48 74 4A   3AF4 9E 4D B7 34   3B44 3A B7 01 89   3B90 00 43 A4 0E
3AAC 00 5A 00 6A   3AF8 89 0D A5 78   3B48 07 15 A9 79   3B94 BE A7 81 B5
3AB0 9B 55 CD 88   3AFC 75 FB B1 01   3B4C 01 1E 9E 45   3B98 00 43 24 0E
3AB4 88 05 CD 8C   3B00 91 4C AE 78   3B50 9E 9C 5A 9B   3B9C BE A7 81 14
3AB8 A5 78 75 1A   3B04 75 9E 5E B7   3B54 15 89 03 56   3BA0 A7 79 00 8B
3ABC 15 AB 79 00   3B08 32 89 16 A5   3B58 9E 06 54 14   3BA4 12 A5 79 01
3AC0 FD 88 CD 88   3B0C 79 00 D9 B7   3B5C AE 79 00 BE   3BA8 B3 02 AE 79
3AC4 6A 20 BE AB   3B10 80 81 02 B5   3B60 B0 EB BE A9   3BAC 01 A7 79 F5
3AC8 2A 88 05 15   3B14 01 AE 79 00   3B64 D5 BE AB EF   3BB0 99 41 BE A9
3ACC A9 79 01 FD   3B18 BD FF AE 79   3B68 BE B7 3F 5A   3BB4 F1 9E 4A BE
3AD0 0A CD 88 6A   3B1C 01 91 76 9E   3B6C 00 E9 79 01   3BB8 B1 16 BE A9
3AD4 20 BE AB 2A   3B20 3A B7 38 89   3B70 00 55 2A F1   3BBC E4 BE AB E6
3AD8 88 05 BE E4   3B24 15 A5 79 00   3B74 B9 0F BB 30   3BC0 BA A7 69 38
                   3B28 F9 D5 81 02
```

Once the desired dot has been reached it can be turned on by pressing the number "5" key or turned off by pressing the "shift" key (for correction). In these latter two cases, the cursor will then move automatically to the next dot below or to the top of the display if the screen's limits have been crossed. This feature allows the drawing of solid black area or the erasing of an entire screen area simply by keeping the keys down. Once the desired graphic pattern has been composed, pressing the "ENTER" key will cause the printer to produce the corresponding GPRINT hexadecimal codes for archiving or other uses. (The use of any other keys except "break" is ignored.)

The routine is written entirely in machine code. This code is fully relocatable. After loading the 306 bytes in a convenient block of memory it is suggested that you protect it using the command NEW+START ADDRESS+307. You can then access the routine by using CALL ADDRESS or, even better, by using the following BASIC line:

```
1 CALL ADDRESS:WAIT:GPRINT:END
```

This line allows the pattern to stay on the screen after the displaying of the codes until you press the "ENTER" key again.

By the way, you can change the sensitivity of the keys (and the flashing rate) by modifying the value in register UL at the addresses &3AC5 and &3AD4 in the accompanying listing. (The value 20 is used at these points in the listing.)

Program *Shell Game*.

```
100:REM Shell Game
    , 22 March 198
    4
110:REM Copyright
    1984 by John R
    . Gibson
120:"S"CLEAR :DIM
    X$(0)*8,Z$(0)*
    36:RANDOM
130:X$="00000000":
    REM 4 Graphic
    Spaces
140:S$="707870707C
    7C7870":REM Sh
    ell
150:L$="0E0F0F0F0F
    0F0F0E":REM Li
    fted Shell
160:P$="0E0F0F6F6F
    0F0F0E":REM Li
    fted With Pea
170:WAIT 120
180:PRINT "
     Shell Game"
190:PRINT " Cprt 1
    984, John R. G
    ibson"
200:PRINT " The PC
    -2 is quicker
    than"
210:PRINT " the ey
    e!  To move th
    e pea"
220:PRINT "  to th
    e left, use le
    ft"
230:PRINT "arrow.
     To move right
    , use"
240:PRINT "  the r
    ight arrow.  L
    ift"
250:PRINT "shells
    with up arrow.
     Set"
260:PRINT "   shel
    ls down with d
    own"
270:PRINT " arrow.
     Press R and
    the"
280:PRINT "compute
    r will hide th
    e pea"
290:PRINT "at rand
    om.  Can you g
    uess "
300:PRINT " the lo
    cation of the
    pea?":WAIT 0
310:"random"LET I=
    (RND 3)-1
320:"down"IF I=0
    THEN LET Z$(0)
    ="781414140800
    38545454180038
    44443C4000"
330:IF I=1THEN LET
    Z$(0)="7C14141
    4080038545454
    8003844443C400
    0"
340:IF I=2THEN LET
    Z$(0)="7C14141
    4080038545454
    8003844443840
    0"
350:GPRINT X$;S$;X
    $;S$;X$;S$;
360:PRINT "    Fin
    d the ";
370.GPRINT Z$(0);
380:PRINT "!"
390:"XX"IF INKEY$
    =""THEN "XX"
400:LET N$=INKEY$
410:IF N$="R"THEN
    "random"
420:IF ASC N$=8
    THEN LET I=I-1
430:IF I<0THEN LET
    I=2
440:IF ASC N$=12
    THEN LET I=I+1
450:IF I>2THEN LET
    I=0
460:IF ASC N$=11
    THEN GOTO "up"
470:GOTO "down"
480:"up"IF I<>0
    THEN "test1"
490:A$=P$:B$=L$:C$
    =L$
500:"test1"IF I<>1
    THEN "test2"
510:A$=L$:B$=P$:C$
    =L$
520:"test2"IF I<>2
    THEN "test3"
530:A$=L$:B$=L$:C$
    =P$
540:"test3"GPRINT
    X$;A$;X$;B$;X$
    ;C$;
550:PRINT "    Fin
    d the pea!"
560:"ZZ"IF INKEY$
    =""THEN "ZZ"
570:LET N$=INKEY$
580:IF N$="R"THEN
    "random"
590:IF ASC N$=8
    THEN LET I=I-1
600:IF I<0THEN LET
    I=2
610:IF ASC N$=12
    THEN LET I=I+1
620:IF I>2THEN LET
    I=0
630:IF ASC N$=10
    THEN GOTO "dow
    n"
640.GOTO "up"
STATUS 1
                1474
```

SHELL GAME

This program was submitted by *John R. Gibson, 924 Chapman Drive #9, Colorado Springs, CO 80916.* He explains it in the following manner.

Here is a simulation of the old shell game. The arrows on the PC-2 are used to move the pea and shells. Can you keep up? If you are in on the secret you will always know the location of the pea!

The Effect

You can always find the hidden pea whether it is hidden by a spectator or at random by the computer.

The Presentation

There are two ways to present the effect. First, the pea can be hidden manually so that the magician must "guess" its location. (The other players can have you leave the room or turn away while they shuffle the pea around.) Alternately, press R to have the computer randomly select a hiding place. Either way, you as the magician will not be fooled.

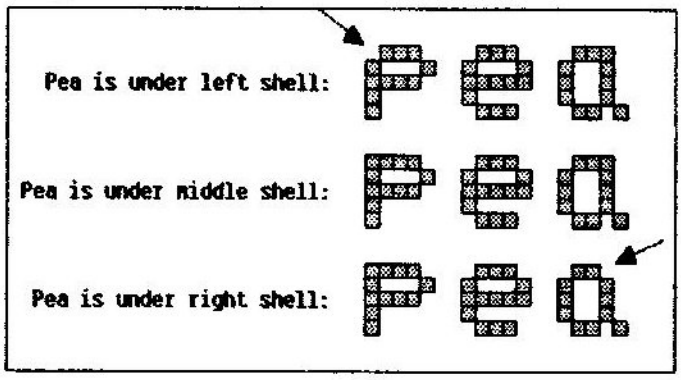
Press DEF/S to execute the program after it has been loaded. A short set of instructions is followed by a display of three shells. To lift the shells and view the pea, push the up arrow. To move the pea right use the right arrow. Use the left arrow to move the pea in that direction. After you have positioned the pea where you want it, use the down arrow to cover the pea.

Note: The program is running continuously in order to read these control keys. Thus, to stop you *must* use the break key.

The Secret

In this trick, the computer is your accomplice. It signals the location of the pea using the word "pea" in the display. If the upper left corner of the letter "p" is missing, then the pea is on the left. If the pea is in the center, no signal is given. When the pea is on the right, the upper right corner of the a is gone.

Diagram *Secret of the Shell Game!*



BINARY MAGIC

This program was submitted by *John R. Gibson, 924 Chapman Drive #9, Colorado Springs, CO 80916.* John has the following to say about his program.

This program can be enjoyed on three levels. To the computer newcomer, this program can be presented as a magic trick. It is impressive both visually and intellectually. On the second level the student will find it a practical demonstration of the binary numbering system. Play around with it for awhile and you will have a much better understanding of how this counting system works. Even advanced computerists will be amused by the program as it is unsurpassed as an example of PC graphics.

The Effect

A spectator selects a graphics object from those displayed (16 total). The person then answers four questions (truthfully) with a yes or no. The psychic computer then reveals the object the spectator had in mind.

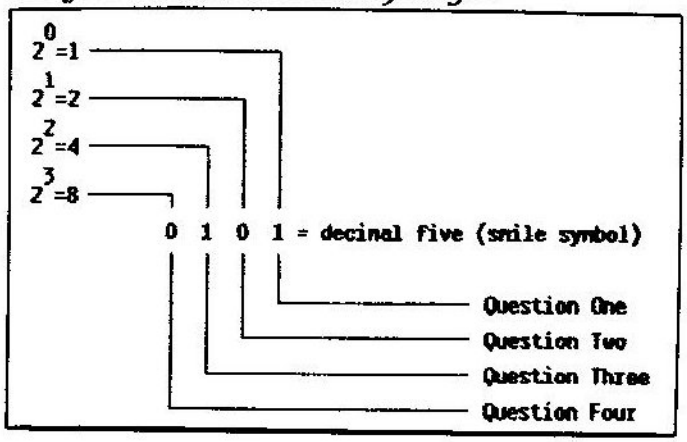
The Presentation

Load the program and hit DEF/B to execute. Some instructions will be displayed, followed by the graphics symbols. Choose one. Press the ENTER button. The computer will ask whether or not your symbol is among the ones displayed. If it is, press Y and the ENTER key. If not, then press N and the ENTER key or alternately just press ENTER. This process is repeated three more times after which the computer reveals the object that was selected by the spectator! To do the trick again, press ENTER.

The Secret

The trick is self-working. Each object is represented by a binary number from 0000 (male symbol) to 1111 (female symbol) where 0 = no and 1 = yes. See the accompanying diagram.

Diagram *The Secret of Binary Magic!*



Program *Binary Magic*

```
100:REM Binary Mag
    ic, 12 March 1
    984
110:REM Copyright
    1984 by John R
    . Gibson
120:"B"CLEAR :DIM
    A$(15)*20:
    RESTORE
130:FOR I=0TO 15
140:READ A$(I):
    NEXT I
150:WAIT 130
160:PRINT "     Bi
    nary Magic"
170:PRINT " Cprt 1
    984, John R. G
    ibson"
180:PRINT "  The P
    C-2 will guess
     the"
190:PRINT " object
     you are think
    ing"
200:PRINT " of.  C
    hoose a symbol
     from"
210:PRINT " those
    displayed.  Pr
    ess"
220:PRINT "   ente
    r key.  Answer
    "
230:PRINT "questio
    ns with a yes(
    Y) or"
240:PRINT "no (N)
    plus ENTER and
     your"
250:PRINT " symbol
     will be revea
    led!"
260:PRINT " Press
    enter to conti
    nue."
270:"G"WAIT 0:N=0
280:FOR I=0TO 14:
    GPRINT A$(I);
290:BEEP 1,I,4:
    NEXT I:WAIT
300:GPRINT A$(15):
    WAIT 0
310:GPRINT A$(1);A
    $(3);A$(5);A$(
    7);A$(9);A$(11
    );A$(13);A$(15
    );
320:INPUT "  Here(
    Y,N)?";X$
330:IF X$="Y"THEN
    LET N=N+1
340:CLS :X$="  "
350:GPRINT A$(2);A
    $(3);A$(6);A$(
    7);A$(10);A$(1
    1);A$(14);A$(1
    5);
360:INPUT "  Here(
    Y,N)?";X$
370:IF X$="Y"THEN
    LET N=N+2
380:CLS :X$="  "
390:GPRINT A$(4);A
    $(5);A$(6);A$(
    7);A$(12);A$(1
    3);A$(14);A$(1
    5);
400:INPUT "  Here(
    Y,N)?";X$
410:IF X$="Y"THEN
    LET N=N+4
420:CLS :X$="  "
430:GPRINT A$(8);A
    $(9);A$(10);A$
    (11);A$(12);A$
    (13);A$(14);A$
    (15);
440:INPUT "  Here(
    Y,N)?";X$
450:IF X$="Y"THEN
    LET N=N+8
460:CLS :X$="  "
470:FOR I=1TO 30
480:BEEP 2,250,2:
    GCURSOR 60
490:GPRINT "1C1C1C
    1C7F3E1C0800";
500:GPRINT A$(RND
    (16)-1);
510:GPRINT "00081C
    3E7F1C1C1C1C"
520:NEXT I
530:PRINT "You are
     thinking of t
    he ";:WAIT
540:GPRINT A$(N):
    CLS
550:GOTO "G"
560:DATA "00304848
    48350307000",
    "000E112142211
    10E0000","0004
    324A1429261000
    00"
570:REM Male, Hear
    t, Swirl
580:DATA "00364949
    36494936000",
    "0003474F7F4F4
    7030000","001C
    2E5B5F5B2E1C00
    00"
590:REM Clover, Go
    blet, Smiley
600:DATA "0063594F
    454F59630000",
    "00003E2222223
    E000000","0030
    28242224283000
    00"
610:REM Phone, Squ
    are, Triangle
620:DATA "007F5D7F
    777F5D7F0000",
    "001C22414141412
    21C0000","0008
    1C2A7F6B7F2A1C
    08"
630:REM Die, Circl
    e, UFO
640:DATA "00003E7E
    7E7E3E24120C",
    "0040783E2B2F2
    B3E7840","0008
    142241221408000
    00"
650:REM Cup, Wumpu
    s, Diamond
660:DATA "00062979
    29060000000";
    REM Female
STATUS 1        1724
```

INSIDE THE PC-1500

This is the third part of the current series on the Sharp PC-1500. The information is provided through the efforts of *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* We pick up in this issue from where we stopped in *PCN* Issue 32. That is with a continuation of synopsis of a number of the routines contained in the ROM of the PC-1500. (Of course, virtually all of this information is applicable to the Radio Shack PC-2). These routines are capable of performing many useful functions when called as machine language subroutines, etc. Norlin has spent a great deal of time gleaning this information and providing an easy to read summary of their uses. The next page picks up where the previous article left off -- in the midst of presenting information on binary routines.

BINARY FORMAT IN R0

The following format is used for data in R0:

```
7A04  B2, identifying R0 as containing binary
7A05  First byte of data
7A06  Second byte of data
```

Data in this format may not be used as input for floating-point arithmetic routines. It may, however, constitute the contents of a variable; BASIC routines will correctly interpret it.

Subroutine D9E7 stores the contents of U into R0, in binary format. Following execution, A will contain UL. Y and U are unchanged.

Subroutine D9E3 stores the byte pointed to by X into R0, in binary format. Following execution, UL will contain that same byte; UH will contain 00. Y is unchanged.

CHARACTER STRING ROUTINES

STRING POINTER

A string pointer in R0 has the following format:

```
7A04  D0, identifying R0 as a string pointer
7A05  High byte, beginning address of string
7A06  Low byte, beginning address of string
7A07  Number of characters in string
```

Call 24 (DEAF) stores a string pointer into R0, taking X as the beginning address, and A as the length of the string. Y is unchanged.

Call DC (DEBC) loads X with the address, and UL and A with the length of the string pointed to by R0. Register Y is unchanged.

Subroutine D048 will copy the string beginning at X, containing UL characters, into memory beginning at address Y. After execution, Y will contain the address following that of the last character stored.

EQUIVALENTS OF BASIC FUNCTIONS

(1) String input, numeric output. Prior to execution, R0 must contain a pointer to the string; the numeric result will be placed into R0.

(a) ASC: Subroutine D9DD. Precede by loading YL with 60. Result will be in binary format.
(b) LEN: Subroutine D9DD. Precede by loading YL with 64. Result will be in binary format.
(c) VAL: Subroutine D9D7. Result will be in decimal.

(2) Numeric input, string output. Prior to execution, R0 must contain numeric data (either binary or decimal format), and 7894 must contain 10. Following execution, R0 will contain a pointer to the resulting string, and UH will contain either 00 or an error code.

(a) CHR$: Subroutine D9B1. (C1 is placed into 7A04, instead of D0)
(b) STR$: Subroutine D9CF.

STRING COMPARISONS

Subroutine D0F9 will compare the two strings indicated by string pointers in R0 and R2 (here designated as R0$ and R2$). The comparison performed will depend on the contents of A, as follows:

A=00:	R0$<>R2$	A=04:	R0$=R2$
A=01:	R0$<R2$	A=05:	R0$<=R2$
A=02:	R0$>R2$	A=06:	R0$>=R2$

If the test condition is met, R0 will contain 1 with flag Z clear; otherwise, R1 will contain 0, with flag Z set. R2 will be unchanged. (NOTE: Comparisons using A=5 or 6 will not work with ROM version A01.)

VARIABLES USED BY BASIC

LOCATING A NAMED VARIABLE

A variable specified by its name may be located by Call 0E (D461). Preceding execution, the variable sought must be specified, according to the following rules:

(1) UH must contain the ASCII code for the first character (A to Z) of the name of the variable. (@( ) and @$( ) may not be used here.)
(2) For one-character names, UL must contain zero. For two-character names, UL must contain the ASCII code for the second character--except that when the second character is 0 to 9, UL should contain 10 to 19.
(3) If a string variable is specified, UL must be increased by 20; if a dimensioned variable, by 80.
(4) When a single subscript is used, 01 must be stored into 788C, and R0 must contain the value of the subscript (in decimal).
(5) When a double subscript is used, 02 must be stored into 788C. The first subscript (in decimal) must be pushed into the BASIC data stack; the second subscript (in decimal) must be contained in R0.

Two data bytes must follow the instruction calling this subroutine. The first of these two passed bytes should be 5A. Following execution of Call 0E, U will contain the beginning address of the variable. Also, R0 will contain a variable pointer, having the following format:

7A04	D0
7A05	High byte of variable address
7A06	Low byte of variable address
7A07	Length of string variable; 88 if numeric

If a subscript is too large, or a subscripted variable was not previously dimensioned, the return address is incremented by the second passed byte, with UH containing an error code. If an non-subscripted previously-undefined variable was sought, 7A05-7A06 will contain the codes for the variable name, with 80 added to the first code.

LOADING R0 WITH VARIABLE CONTENTS

Call 0E (D461) may be used to load R0 with the contents of a variable specified by name. Preceding execution, the variable sought must be specified according to the same rules used above.

The first of two passed bytes must be 52. Following execution, R0 will contain numeric data or a pointer to a string, as determined by the variable specified.

If a subscript is too large, or a subscripted variable was not previously dimensioned, the return address is incremented by the second passed byte, with UH containing an error code. If an non-subscripted previously-undefined variable was sought, R0 will contain zero.

CLEARING VARIABLES

Subroutine D080 clears all BASIC variables. No change in Y.

Subroutine D091 clears Main-memory variables only, by resetting the Start of Variables Pointer (7899-789A). No change in X, Y, or U.

TIME and RANDOM NUMBER

TIME (FORMAT USED BY BASIC)

Subroutine DE82 copies TIME into R0, in the same format used by BASIC. Registers XH and YH will contain 7A following execution.

Subroutine DE1D stores R0 into TIME, with R0 given in the same format used by BASIC. One data byte is passed to this subroutine; it is added to the return address if R0 specifies TIME incorrectly.

TIME (ALTERNATE FORMAT)

Data may be transferred between R0 and the timer with the following format used in R0:

```
7A00  Not used
7A01  Not used
7A02  1st nybble=month number (hex); 2nd nybble=week day (0 to 6)
7A03  Date of month (decimal)
7A04  Hour (0 to 23, decimal)
7A05  Minute (0 to 59, decimal)
7A06  Second (0 to 59, decimal)
7A07  Not used
```

Subroutine E59A copies R0 into TIME; X and Y are unchanged.

Subroutine E5B4 copies TIME into R0; X and Y are unchanged.

RANDOM NUMBER

Subroutine F5DD places RND (R0), interpreted as in BASIC, into R0. The random number is replaced by its next value. Both R2 and R6 are

used: XH and YH will contain 7A following execution.

Call 5C (F61B) replaces the random number (located 7B00-7B07, in decimal format) by its next value. Precede by loading XH and YH with 7A. Following execution, XH and YH will contain 7A. The contents of both R0 and R2 will be changed.

READING CODES FROM BASIC

EVALUATION OF EXPRESSION IN BASIC

Call DE (D6DF) evaluates an expression specified by BASIC code, beginning at the address contained in Y. The result is placed into R0. The BASIC expression is terminated by any of the codes 00 (null), 0D (ENTER), 2C (comma), 3A (colon), or 3B (semicolon).

The instruction calling this subroutine must be followed by a data byte, which will be added to the return address (with UH containing an appropriate error code) if an error occurs.

Numeric results obtained from AND, OR, NOT, ASC, LEN, POINT, PEEK, or PEEK# are in signed binary; other numeric results are in decimal. String results are expressed as string pointers in R0, with the string itself located in the String Buffer.

CONVERSION OF HEX CHARACTERS TO BYTE

Subroutine ED95 loads A with a byte determined by a pair of codes for hex digits, located starting at address X. Codes for characters other than 0-9 or A-E will clear flag C, signaling an error. After execution, X will point to the next address; Y and U are unchanged.

SEARCH FOR BASIC PROGRAM LINE

Subroutine D2EA searches for the address in memory of the BASIC line whose number (in hexadecimal) is contained in U. The search will begin at the address indicated by the START OF BASIC (or START OF BASIC in ROM) pointer. It will end if a stop byte is encountered.

The same subroutine will search for the line with a specified label. In this case, UH must contain FF, and R0 must contain a string pointer pointing to the string forming the label sought. If necessary, a search for a label will bypass stop bytes, continuing until END OF BASIC is reached.

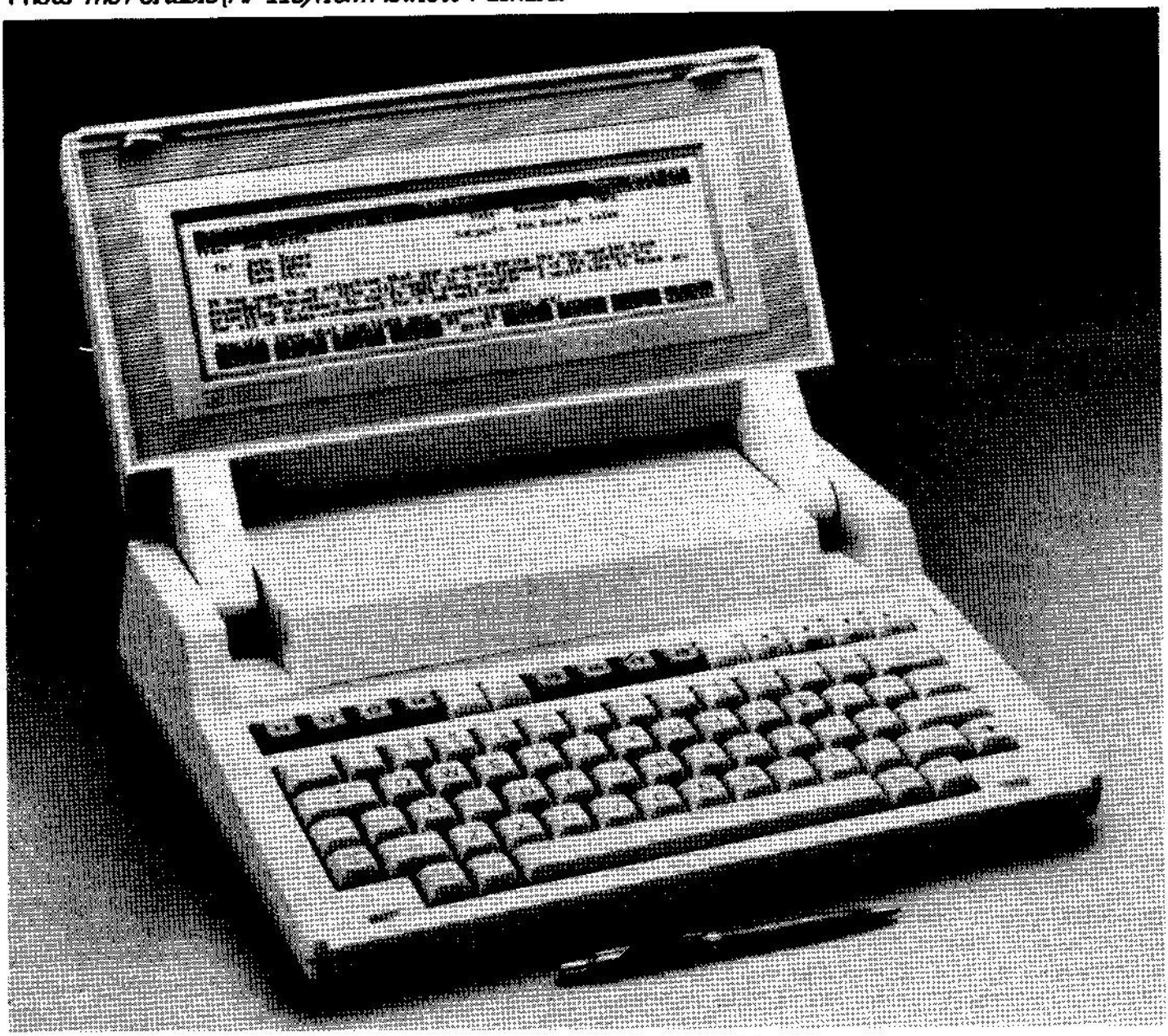
In either case, if the line sought is located, the address of the first statement in that line is placed into 78A6-A7. (The address at which the line's number begins is less than that address by 3.)

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

Issue 34 - July/August

Photo *The Portable (HP 110) from Hewlett-Packard.*



Another *Personal Information* ™ product.

"THE PORTABLE" ANNOUNCED BY HP

Hewlett-Packard Corporation, a company that has been becoming increasingly aggressive in the personal computer market, has now announced availability of a new, nine-pound, notebook-sized computer that the company refers to as "The Portable". Model-wise, it is known as the HP-110.

The new computer features a flip-up, 16-line by 80-column liquid-crystal display. The display is bit-mapped, having 480 by 128 pixels, and can thus support graphics. The flip-up display is adjustable to any desired viewing angle. Display contrast is controlled from the keyboard. The full-sized keyboard has 75 keys. Eight of these are designated as special-function keys.

Perhaps most notable about the new portable machine is the massive amount (384K!) of software built-in on ROMs. The Portable includes the popular "1-2-3" software package from Lotus Corporation. This program provides integrated spreadsheet, graphing and database management capabilities. Other software included in the unit serves up fundamental word processing capability and communicating functions. There is also a "Personal Applications Manager" executive-type program designed to make it easy to set up and manage programmed operation of The Portable.

A similarly large amount of user RAM memory (272K) is provided in the unit. A user-selectable amount of this RAM can be designated as an "electronic-disk," with the balance serving as the user's active memory area. Since the standard operating programs are resident in ROM, virtually all of the unit's RAM is available to the user for manipulation of spreadsheets, memo files, and other kinds of data.

The Portable also contains provisions for running industry-standard MS-DOS software (Version 2.11) and can share data with HP, IBM and other compatible personal computers.

The portable computer weighs less than nine pounds, including a rechargeable battery that can supply power for 16 to 20 hours of continuous operation. The special, sealed, lead-acid battery (which is not replaceable by the user) is said to have a service life in excess of five years. A unique indicator on the unit's display informs the user of the amount of charge remaining in the battery at any time. When remaining energy falls below 20%, special precautions are taken to notify the user that recharging is required. If battery energy falls to a very low level (less than 5%) the display cannot be activated. All available energy is then used to conserve the contents of the unit's CMOS memory. A fully charged battery can maintain the system's memory intact for over a year if the computer is kept turned off.

When the flip-up display is closed down over the keyboard, The Portable measures a scant 13 by 10 by 3 inches. A padded carrying case with shoulder strap and handle is provided for ease and comfort when taking the unit on a journey.

A built-in 300-baud rate, direct-connect modem enables the unit to serve as a terminal in a large computer system. The standard communications software supplied in ROM permits files to be downloaded from a remote site and vice versa.

The Portable is extremely fast in operation thanks to the use of a CMOS 8086 CPU. This is a true 16-bit microprocessor and HP has it running at a 5.33 Megahertz clock rate in The Portable. Speed of operation is also enhanced by the fact that the operating software is built into ROM. There is no waiting for disk loads, memory swapping, etc. Switching between applications, such as running Lotus 1-2-3 and then going to the word processing package, takes just a matter of seconds.

The new traveling computer will be supported by several fully-portable peripheral devices. One is HP's new ThinkJet Personal Printer, a lightweight, battery-operated printer that can be connected to The Portable. The other is the HP-9114A battery-powered 3-1/2 disk drive with 710 kilobytes-per-disk capacity. The Portable, coupled to these two peripherals, can form a complete desktop system or be used anywhere in the field, operating entirely on built-in batteries!

Other features of The Portable include a real-time clock and calendar system. This system can keep date/time events anywhere between January 1, 1980 and December 31, 2039. Built-in alarm and appointment-keeping functions can be coupled with the clock/calendar to track appointments as well as organize and find files by name, time and date. It is even possible to have The Portable turn itself on and execute a program at a specific time and date.

Hewlett-Packard reports a number ofsoftware packages, including languages such as PASCAL and BASIC, will be available on 3-1/2 inch disks fo use with The Portable.

The Portable has a list price of $2995.00 in the U.S. The list price of the ThinkJet printer is said to be $495.00. The HP-9114A disk drive has a reported price of $795.00. Contact your local HP personal-computer dealer or phone toll-free 800-FOR-HPPC in the U.S. for more information.

REVIEW OF ROM SOFTWARE MODULES

This is a continuation of the reviews and commentaries on the Sharp software modules that originated in Issue 33 of *PCN*. The views are those of *Robert Sutliff, 55 North Avenue, New Rochelle, NY 10805,* and do not necessarily reflect the opinions of *PCN* or its editorial staff.

General Statistics (CE-502A)

This library offers a wide variety of programs in statistics and is primarily intended for the user who needs to analyze raw data. Norlin Rober gave a review of this module in Issue 26 of *PCN*. Given paired or unpaired data these programs compute many of the statistics which, with the aid of standard tables, allow decisions in hypothesis testing to be made. Statistical tables (or the Statistical Distributions module) are a must when using this library. Unfortunately, only numerical input is allowed so that arithmetic, function or coded values cannot be input (except in the ANOVA programs). The programs include:

1. Means and Moments. These statistics can be used to compute standard deviation, skewness and kurtosis. As Mr. Rober pointed out, there are often compounded rounding errors. This can be reduced by coding the data.
2. t-Test: Unpaired.
3. t-Test: Paired. Note that there is a typo on page 61 of the manual which should read: "If the statistic calculated falls *outside* this region of acceptance...", etc. Also, the correct region in this problem should be (-2.262, 2.262).
4. Analysis of Variance: One-Way.
5. Analysis of Variance: Two-Way.
6. Contingency Table. Just over 200 elements can be entered in one run.
7. Mann-Whitney Rank Sum Test. This test is comparable to the Wilcoxon test, although the formulas are slightly altered. As Mr. Rober noted, a Shell sort subroutine labeled RS1 is contained within this program and can be used by a user program residing in RAM.

A routine for computing normal distribution values from values of z exists in lines 28205 to 28340. This routine may also be utilized by a user program residing in RAM. Additionally, the first five programs allow unlimited amounts of data to be entered in a single run by using data bases. Data is entered in blocks which fill the memory. Each block may then be edited, summed, recorded and finally discarded to make room for the next block. Use of a printer is optional. If the optional printer/cassette unit is not attached, then prompts for its use will not appear.

The manual has many good points to recommend it. It offers essays on the applications of each program that both the novice and expert will appreciate. There are extensive comments on variable use, subroutines, cassette files, statistical formulas, and suggested reading. The General Statistics library will appeal to users who are knowledgeable in the subject matter and who can work from samples.

Statistical Distribution (CE-502B)

This library is intended as a companion to the General Statistics Module (CE-502A) just described. The values generated by the previous module can be compared to standard test values and evaluated by this module. This library contains the most commonly used lookup tables as well as interesting histogram and regression programs. The module includes the following:

1. Histogram. This program classifies raw or grouped data into up to 243 cells of any specified width. Data bases are used to allow unlimited amounts of data to be entered. Data can be loaded from or saved to cassette files. This program does not actually draw the histogram, but outputs the cells with frequencies, means, and moments. Once these statistics have been computed the user has the option of comparing the histogram to any of these classical theoretical distributions: binomial, normal, Poisson, Weibull or uniform. One can input parameter values to be used in the theoretical comparison or default to values calculated from the input data. The expected frequencies of the theoretical distribution which are output can be compared to the observed frequencies, and a chi-square statistic can be used to test goodness of fit. If the null hypothesis is that the input data come from a population having the selected theoretical distribution, then an output chi-square area gives the probability that this hypothesis is correct. The higher this value, the better the indication that the data does indeed fit the theoretical distribution. There is a programming error which fails to reinitialize certain values whenever the error indicator "EXP.FREQ.=0" occurs. Though the program seems to go to a recovery point after this, make sure to break the program and reenter the data in order to assure correct results. There are also several instances where a "recovery from error" leaves the user at a different point than where the error was

encountered. Annoyingly the program must be redone from start. Finally, depending on the selected theoretical distribution, there is sometimes considerable rounding error in the chi-square values.

2. Binomial Distribution. Given a number of trials and the probability of success in each trial, this program computes the probability of any exact (or fewer) number of successes.

3. Poisson Distribution. Given the average number of successes in the run, this program offers the same probability as the binomial distribution.

4. Normal Distribution. This program relieves the tedium of using the standard z table. One can input z-value and obtain left- and right-tail probabilities. Alternately, one can input a right-tail probability and obtain the corresponding z-value.

5. Weibull Distribution. Upon input of the shape and scale parameters, one can enter a point and obtain the right-tail probability or vice versa.

6. Exponential Distribution. The previous program is used to compute this variation of the Weibull distribution.

7. Chi-Square Distribution. In measuring variance, the degrees of freedom and chi-square value are input and the right-tail area is output.

8. Student t Distribution. The degrees of freedom and test statistics determine the left-tail probability, not the two-tail probability as the manual states.

9. F Distribution. Given the two degrees of freedom and the F statistic, the right-tail probability is computed.

In programs (7) to (9), it is the inverse procedure that is needed in most applications, namely, inputting a probability value and obtaining a test statistic. There is no provision for this operation. However, inverse algorithms are not very accurate and sometimes are not available. It is easy to get used to using the programs as they exist in order to get results. The user who is determined to use the inverse procedure can easily add a RAM program to approximate inverse values using the programs in this library.

10. Multiple Linear Regression. If a variable is thought to be a linear combination of several independent variables, then this program will approximate the coefficients in the functional expression. The user enters observed data points and the program computes the linear expression as well as correlation coefficients and other associated statistics. There must be more data points than the number of independent variables. This program can be used for a single linear regression by using only one independent variable. This program uses a routine called INV to compute the inverse of a matrix. This routine may be used by a program in RAM. However, in computing the inverse of the correlation coefficient matrix, the same algorithm at that in the Math Library Matrices programs is used. This procedure contains an error and so the same kind of problem may occur here. An "INSOL.MATRIX" prompt may appear even when a solution does exist. Switching the order of entry of data points may help.

In all of the mentioned programs of this library, function and arithmetic values can be directly entered as input. The printer is optional and printer/cassette related prompts will not appear if it is not attached. The good quality of the manual repeats from the General Statistics Library manual. The programs are straightforward to use, but the user will have to bear in mind the errors in the Histogram and Multiple Linear Regression programs. Although not technically a glitch, one other thing is annoying. Titles for each distribution are set up so that each program can announce itself when chosen, but they are utilized only when the printer is attached!

Finance Module (CE-504A)

This module contains a wide (and relatively unassociated) collection of programs meant to aid in financial decision making. The programs will appeal to many different users, although any particular person might find only one or two of the programs of interest. There are annuity, days-between-dates, and cash flow analysis programs for bankers, bond analysis for investment analysts, a rent-buy program for real estate investors, and machine replacement and depreciation programs for business managers. In all these programs, only a limited number of key quantitative factors can be taken into account. The output should only be considered an aid, and not authority, in decision making. As the adage goes, there is no substitute for experience. The seven programs are:

1. Annuity/Annuities Due. This extremely versatile program handles the five variables used in ordinary annuity computations: the principle value PV, future value FV, interest I, number of payment periods N, and payment per period PMT. Upon input of four variable values, the unknown fifth is computed. An option allows payments to be computed at the beginning or end of the

compounding period. This handy program is built into business calculators, such as the TI Business Analyst. The PC-1500/A goes one step further by printing out an amortization schedule on the plotter! For example, consider a $6,000 car loan made by a firm offering 10% (annual interest) financing. If the loan is to be repaid in semi- annual installments over a period of three years, the computer calculates an installment payment of $1,182.10 per half year. The installment schedule is printed and the total interest of $1,092.62 can be added to the loan to find the total repayment amount. This program can be used for mortgages, loans, savings accounts, investments, etc. Compound interest problems are handled by letting PMT equal zero. All variables can be printed by pressing P. The payment schedule can be printed for any interval of periods specified by the user. One item not mentioned in the manual: you must press E to exit the program.

2. Depreciation. This program gives the figures for a depreciation table on an item over a period of time, based on its book value, salvage value and expected useful life. The user can select the straight line, declining or sum-of-years method of depreciation, making this program great for tax purposes. If the declining method is chosen, the user can elect to use a crossover to the straight line method so that the depreciation will be optimized. The crossover is calculated and done automatically! A few other inputs yield the output which consists of four (unexplained by the manual) quantities each year: ACU.D = the accumulated depreciation to date, DEP = depreciation this year, RBV = revised book value (of item at this date) and RDV = revised depreciation value (= RBV - salvage value). A six year depreciation table using the declining method with crossover is shown for illustrative purposes. The example assumes an item with a book value of $10,000, salvage value of $1,000, a 5-year life and declining balance rate of 120%, with the purchase occurring in June.

3. Machinery Replacement Analysis. Enter the useful life, interest rate, price, salvage value and operating costs of a new machine as well as the same data for a machine already in use, and the program compares the expenditures on each for the remaining life of the existing machine and declares which option is better, i.e., whether or not to replace the existing machine. This program actually states the decision for you, but the user should simply take this as advice. Any item that wears out can be considered as a "machine" in this program.

4. Days-Between-Dates. Used by bankers (and computer show-offs), this program will accept any date and output the day of the week on which the date falls. Alternately, it will accept any two dates and output the number of days between the dates. Certain restrictions exist on allowable dates due to calendar changes made throughout history. The user can select 360- or 365-day years (366 on leap year).

5. Rent/Buy Decision. Consider the decision of buying a house or renting a dwelling. Thirteen inputs are made about the house and rent alternative. Ten output values give the yield (rate of return) gain or loss over the rent alternative as well as the break-even values of each alternative over the other.

6. Bond Analysis. The user inputs data on a bond, such as purchase and redemption dates, compounding period, type of bond, interest rate, call price and accounting period. Based on a $100 bond, the program then computes the percent yield when given the purchase price or vice versa. The user even has the option of asking that these computations be made before or after the tax yield. When this choice is taken, then the user must enter personal income and capital gains tax brackets! This handy program quickly computes the purchase price necessary to give a $100 bond value at a given percent yield. This can be used to classify types of bonds according to yield value. The inverse process provided by the program is more useful. The yield is computed based on a purchase price for a $100 bond. If the purchase price exceeds $100, then the bond is being bought "at a premium". There is a typo in example 2 in the program's manual (the yield is being computed, not the purchase price).

7. Cash Flow Analysis. This program outputs the net present value (NPV) or financial management rate of return (FMRR) for a grouped, variable or graduated cash flow analysis. Up to 100 cash flows can be input for a single analysis. The inputs vary with the type of analysis. The NPV or FMRR can be used to judge the profitability of a project based on an initial investment. The FMRR is used when one needs to take into account cash flows generated by the investment, reinvestment amounts, and short- as well as long-term investment rates.

In all of these financial programs, arithmetic and function expressions can be used for input. Output is always rounded to two decimal places. Sometimes more decimal places are needed (for

example when dealing with interest rates) and can be obtained by recalling the variable values from locations A to Z. Although no documentation is given for any program except Bond Analysis, it is easy to find where the variables are stored by recalling A to Z. Use of the printer is optional in all programs. Due to an oversight, whenever the printer is selected it is left in the GRAPH mode. The manual provides good examples and summaries, though it could use more explanations on the use of the output in decision-making. The programs overlap in different fields of financial management. In all, the Finance Library offers an interesting potpourri of financial applications.

SharpCalc Module (CE-507A)

A spreadsheet for your pocket computer? Here it is! Although SharpCalc can't match the speed or versatility of a Multiplan, it certainly shows off the capabilities of the PC-1500/A and its printer. Many people choose a pocket computer for their first experience with a micro, so SharpCalc can act as an introduction to spreadsheet analysis. For people who use their pocket computer extensively in the field, Sharpcalc can be an impressive tool for on-the-spot analysis. (Even though any spreadsheet takes time to set up, the great advantage of CMOS memory is that the spreadsheet is not lost when the computer is turned off!) The entire program is loaded by a quick module installation. The printer is optional since any cell (and even a mock-up of a row) can be displayed on the LCD screen. However, use of the printer enables one to elaborately print out the entire spreadsheet, complete with titles.

The program is not available for listing in the PRO mode or by use of the arrow keys. The main body, however, is written in BASIC. When the module is installed and the computer is in Reserve Group II, the six softkeys become direction keys for moving throughout the spread- sheet. Fifteen remaining keys, when used with DEF, activate program sections. A plastic template is provided to fit over the keyboard and show the operation of each defined key. User RAM is used up quickly by this program so do not plan on any resident RAM programs. Unlike other modules, there are no flashing default values provided for inputs. Thus, every question put forth by SharpCalc needs a typed response. The program contains extensive prompts, cues and responses in a very friendly manner. SharpCalc, through its well written manual, is a snap to learn!

A spreadsheet consists of a table with rows and columns, each location of which is called a cell. SharpCalc allows up to 225 cells to be set up with a maximum of 26 rows (labeled 1 to 26) and 26 columns (labeled A to Z). For example, the cell in the fith row and sixth column would be denoted as 05F. The home location is 01A. If many relations are defined (see below) the size of the spreadsheet might have to be reduced.

The first task that must be done in order to use the program is to "set up" a spreadsheet. Once set up (sized) it cannot be altered by adding or deleting rows or columns. The display "windows in" on one cell at a time. The user moves the window using the GO UP, GO DOWN, GO LEFT and GO RIGHT keys. GO HOME sets the window to 01A, while GO LOC may be used to jump to any specified cell. WINDOW shows an entire row of the spreadsheet (in groups of 7 cells) using clever LCD graphics to display the row number and indicate the contents of each cell using abbreviations. This display also denotes which cell the window FOCUS command is on.

The next step is to use the FILL IN mode to define the contents of each cell. A cell may contain a numeric value, a string up to four characters in length, or an arithmetic or functional expression involving other cells, called a relation. The latter concept is what makes electronic spreadsheets so popular.

SharpCalc allows the following relations:

1. The operations +, -, *, / and ^ (exponential) on two cells.

2. Operations involving a constant. A constant must be assigned a value. That value may be used repeatedly throughout a spreadsheet.

3. Functions on a single cell, such as SIN, COS, TAN, LOG, LN, INT, ABS, EXP, CHS, FRA, PCT(%), %CH, COPY, SQR and CUB (cube). For example, cell 07B could contain @FRA02C which would direct that the program find the decimal fractional part of the numeric value in cell 02C.

4. Row/column operations or functions on multiple elements. The following kinds of operations may be performed on any consecutive elements in a row or column: +, -, *, /, ^, AVG, ANB, MAX, or MIN. For example, cell 04B could contain 02A@ANB10A meaning find the average of the non-zero numbers in the column elements 02A through 10A.

Once the cells in a spreadsheet have been filled in, the user may verify the contents using the VIEW IN mode or the PRT INPUT mode, if a hardcopy is desired. All parameters such as the number of rows,

columns, relations, etc., may be checked using the VIEW CONFIG or PRT CONFIG directives.

The spreadsheet may be evaluated by either row or column precedence, using the COMPUTE command. The computed values may be viewed on the display in the VIEWOUT mode. Alternately the entire spreadsheet may be printed via the PRT OUTPUT instruction. Additionally, titles of up to 4 characters may be filled in for each row and column. Any errors may be located by the LOC CHECK operation. Any section may be ended using EXIT.

The spreadsheet may be stored on cassette tape using SHT TFR to activate sheet transfer. Likewise, a stored sheet may be loaded into the computer. When the printer is used, the entire spreadsheet is printed along with an inputted title. Either the inputted data or computed cells can be printed. Any or all rows may be accented with a line feed, underline (solid or dashed) and color. Negative amounts may be specified to be output in red. A two decimal place business format may be selected. Finally, the entire spreadsheet may be printed in any of three sizes! (The sheet is printed vertically in sections.)

Some of the best spreadsheet abilities are provided by SharpCalc. The LCD is effectively utilized to permit studying of the spreadsheet with abbreviations on the side of the display always showing which mode the user is in. If you own a PC-1500/A and would like to experience spreadsheet techniques, SharpCalc is a simple way to do so.

Example *Outputs from the Finance Module (CE-501A)*

```
                AMORTIZATION SCHEDULES

        PRINCIPAL              -6000.00
        INTEREST RATE              5.00
        PERIODS                    6.00
        REGULAR PAYMENT         1182.10

 NO.    AMORTIZED     INTEREST      BALANCE
  1        882.10        300.00      5117.90
  2        926.21        255.89      4191.69
  3        972.53        209.58      3219.16
  4       1021.14        160.96      2198.02
  5       1072.21        109.90      1125.81
  6       1125.81         56.29         0.00

ACCUM INT. FROM 1 TO 6 IS      1092.62
ACCUM AMO. FROM 1 TO 6 IS      6000.00
```

```
YEAR:  1
DEP          1200.00
ACU.D        1200.00
RDV          7800.00
RBV          8800.00

YEAR:  2
DEP          2112.00
ACU.D        3312.00
RDV          5688.00
RBV          6688.00

YEAR:  3
DEP          1625.14
ACU.D        4937.14
RDV          4062.86
RBV          5062.86

YEAR:  4
DEP          1625.14
ACU.D        6562.29
RDV          2437.71
RBV          3437.71

YEAR:  5
DEP          1625.14
ACU.D        8187.43
RDV           812.57
RBV          1812.57

YEAR:  6
DEP           812.57
ACU.D        9000.00
RDV             0.00
RBV          1000.00

CROSSOVER:3
```

IMPROVED VERTICAL LISTER PROGRAM

Here is an improved version of the Vertical Lister previously described in Issue 27 of *PCN*. This one is written entirely in machine language.

It can be left residing in memory and used at any time without the need for MERGEing a BASIC portion. It will print token words used by programs utilizing the CE-158 (provided that the CE-158 is connected).

Listings may be made in either CSIZE 1 or CSIZE 2. In size 2, each "page" printed will contain 12 lines of up to 42 characters-per-line. In size 1, 24 lines having up to 84 characters each may be printed. Long lines will wrap-around to the following line.

If memory contains MERGEd programs, they will all be listed.

Entering the Program

The listing supplied with this article places the Vertical Lister program in the Reserve memory area of a computer having the CE-155 8K memory module installed. However, this program is completely relocatable. You can put it wherever you want it. If you have a PC-1500A, it is suggested that you put it into memory starting at the address &7D00. If you have a 16K RAM module and do not want to lose the Reserve area, you may wish to reallocate the computer's memory by executing NEW &01A0 and then putting the program into the area beginning at &0D00. If you do not have a Monitor program, you will have to enter the program using POKE statements.

When the program has been loaded into memory, it may be saved on tape by execution of the command string CSAVE M "VERTICAL LISTER";&3800,&38C4. (Naturally, if you locate your copy of the program elsewhere, you will have to change the starting and ending addresses used in your CSAVE M statement accordingly.) The program may be reloaded from tape using the CLOAD M command into the same addresses from which it was originally CSAVEd. Alternately, load it into some other location by specifying the starting address in the CLOAD M command.

Using the New Vertical Lister Program

To produce a vertical listing, simply execute CALL &3800 (or whatever address you have used as the beginning address of the machine language program). If you interrupt the vertical listing process with BREAK, execute TEXT following the break in order to reset the printer.

The program is not totally idiot-proof. If you execute it when there is no BASIC program in memory, the result may be having some garbage printed!

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

Program *Improved Vertical Lister.*

```
3864 B7 3A ED 79  3800 E9 78 50 00
3868 F4 01 8B 02  3804 EB 79 F0 FF
386C B7 63 83 24  3808 BE D2 6F BE
3870 44 F5 88 03  380C AC 97 DD AE
3874 FD 0A 84 5F  3810 79 F2 B5 08
3878 3A ED 79 F4  3814 FB B1 09 ED
387C 01 8B 02 5E  3818 79 F4 01 83
3880 63 83 14 0E  381C 02 B1 09 FD
3884 36 4A 10 14  3820 C8 48 7B 4A
3888 00 2A BE B4  3824 7F 43 B5 00
388C F4 BE D2 B3  3828 43 43 0E BF
3890 83 10 8E 12  382C AC 07 BE A7
3894 FD 0A 46 46  3830 6C 5A 1C CC
3898 CA 50 4A 10  3834 50 6B 08 E9
389C 14 00 2A BE  3838 78 50 00 91
38A0 B4 F4 FD 80  383C 11 6E 1C 4B
38A4 93 92 FD C8  3840 7F 43 B0 FE
38A8 42 7B 4A 7F  3844 A5 B2 3A 33
38AC B5 00 43 43  3848 04 B1 3C 33
38B0 B5 C0 43 B5  384C 0A B5 3A 51
38B4 FD 0E BE AC  3850 45 B7 0D 8B
38B8 07 BE A7 69  3854 30 B7 E0 81
38BC FD 3A 3B B5  3858 1E 28 45 2A
38C0 BE AC BB CD  385C FD 88 CD 1C
38C4 42           3860 04 0F 2A 12
```

LINEAR SYSTEMS

This program for the PC-1500/PC-2 will find the solutions of a system of linear equations. The size of the system that it can handle depends on the amount of memory present. When used with the PC-1500A and a CE-161 RAM module, it can solve a system of 50 equations. Using the PC-1500 and an 8K RAM module, a system may contain up to 32 equations.

An Example

This example illustrates how the program may be used. Assume we wish to solve the system of equations:

$$\begin{aligned} 3W + 2X + 6Y + 2Z &= 2 \\ W + 4X + 3Y + 2Z &= 6 \\ 4W + 2X + 5Y + Z &= 3 \\ W + 5X + 3Y + 3Z &= 4 \end{aligned}$$

Execute RUN after loading the program. Respond to the prompt "NO OF EQ?" with 4. The display will show prompts for inputting the coefficients, using A1, A2, etc., with the constant term designated C. Thus, in this example, when asked for "EQ 1: A1?", the first coefficient in the first equation should be inputted, which is 3. Similarly, respond to "EQ 1: A2?" with 2, to the next prompt with 6 and then with 2. When the prompt "EQ 1: C?" appears, the constant term on the right side of the equation (which is 2 here) should be inputted.

After inputting the values, the prompt "EDIT?" will be displayed. If you do not wish to edit or review the inputs you have entered, key ENTER. If editing is desired, input "Y" followed by ENTER. If the latter option is taken, the equation number that you wish to review will be requested. After this information has been given, each coefficient in that equation will be presented for review, followed by a question mark. If you wish to alter the coefficient, enter the new value. Otherwise, just press ENTER.

After editing, the prompt "PRINT?" will be displayed. Respond with a "Y" to print a list of the coefficients or key ENTER to skip printing.

The correct solutions for the equations used in this example are W=-2, X=3, Y=2 and Z=-5. The display should show these as X1, X2, X3 and X4. Due to limitations of floating-point arithmetic, the first solution will be given as -1.999999999 rather than as -2 and the fourth as -4.999999999 rather than -5.

The query "PRINT?" will be shown again, offering the option of printing the calculated results.

Execution Times

A system of five equations will take about 6 seconds. Ten equations: 42 seconds. Twenty equations: 5 minutes and 20 seconds. Thirty equations: just under 18 minutes.

The message "NO SOL" will be displayed when there is no unique solution to the system. This occurs when the system is either inconsistent or dependent.

Norlin Rober, 407 North 1st Avenue, Marshalltown, IA, 50158, submitted this routine.

Program *Linear Systems.*

```
10:INPUT "NO OF E
   Q? ";N:DIM A(N
   ,N+1):GOTO 12
11:GOTO 10
12:WAIT 0:FOR I=1
   TO N:FOR J=1TO
   N
13:CLS :PRINT "EQ
   ";I;": A";STR$
   J;:INPUT "? ";
   A(I,J):NEXT J:
   GOTO 15
14:GOTO 13
15:CLS :PRINT "EQ
   ";I;": C";:
   INPUT "? ";A(I
   ,N+1):NEXT I:
   GOTO 24
16:GOTO 15
20:INPUT "EQ NO?
   ";I:IF I>0AND
   I<=NTHEN 22
21:GOTO 20
22:FOR J=1TO N:
   CLS :PRINT "EQ
   ";I;": A";STR$
   J;"=";A(I,J);:
   INPUT "? ";A(I
   ,J)
23:NEXT J:CLS :
   PRINT "EQ";I;"
   : C=";A(I,N+1)
   ;:INPUT "? ";A
   (I,N+1)
24:CLS :INPUT "ED
   IT? ";A$:GOTO
   20
30:WAIT :INPUT "P
   RINT? ";A$:
   GOTO 32
31:GOTO 40
32:FOR I=1TO N:
   LPRINT "EQUATI
   ON";I:FOR J=1
   TO N:LPRINT
   STR$ J+" ";A(I
   ,J):NEXT J:
   LPRINT "C ";A(
   I,J):LPRINT :
   NEXT I
40:CLS :FOR J=1TO
   N-1:IF A(I,I)<
   >0THEN 50
41:FOR J=I+1TO N:
   IF A(J,I)=0
   NEXT J:PRINT "
   NO SOL":END
42:FOR K=1TO N+1:
   M=A(J,K),A(J,K
   )=A(I,K),A(I,K
   )=M:NEXT K
50:FOR J=I+1TO N:
   FOR K=I+1TO N+
   1:A(J,K)=A(J,K
   )-A(I,K)*A(J,I
   )/A(I,I):NEXT
   K:NEXT J:NEXT
   I
51:IF A(N,N)=0
   PRINT "NO SOL"
   :END
52:M=N+1:FOR I=N
   TO 2STEP -1:
   FOR J=1TO I-1:
   A(J,M)=A(J,M)-
   A(J,I)*A(I,M)/
   A(I,I):NEXT J:
   NEXT I
60:FOR I=1TO N:
   PRINT "X"+STR$
   I+"=";A(I,M)/A
   (I,I):NEXT I
70:INPUT "PRINT?
   ";A$:LPRINT "S
   OLUTION:":FOR
   I=1TO N:LPRINT
   STR$ I+" ";A(I
   ,M)/A(I,I):
   NEXT J:LPRINT

STATUS 1 = 890
```

ISOMETRIC DRAWINGS

Richard H. Chrystie, 14824 E. Walbrook, Hacienda Heights, CA 91745, provides this program that is designed to make isometric drawings. The number of components per drawing is dependent on the amount of memory in your PC as shown in the accompanying table.

Components are rectangular or cylindrical solids. The axis of rotation for the cylinders is either the X, Y or Z axis. A scale factor is inputted by the user. If a scale of 1 is selected, all dimensions are in inches. The color of each item may be specified. After an item is drawn, the program asks if you want another item, thus you may make drawings of single or multiple items.

The program is written in BASIC so it is slow. Drawings are limited in size to the distance that the printer can back up.

After an initial view has been drawn you can alter the scale or viewing angle. A rotation of 30 degrees and a tilt of 10 degrees is nice. A rotation of 0 degrees and a tilt of 0 degrees produces a side view. Rotating to 90 degrees results in an end view.

Program *Isometric Drawings.*

```
 5 "3VIEW2"
11 CLEAR
15 TEXT :COLOR 0
20 C=-1
50 DIM X(15,26),Y
   (15,26),Z(15,2
   6),C(15)
70 DIM R(26),T(26
   ),B$(15),G(15)
72 GOSUB 300
75 INPUT "SCALE?"
   ,S:S=126.5823*
   S
80 CSIZE 1
81 LPRINT "SCALE=
   ",S/126.58:LF
   1
82 LPRINT "    NO
       UP     RT
       BK"
83 LF 1
100 INPUT "WANT AN
   OTHER ITEM?",A
   $
101 IF A$="N"THEN
   430
105 INPUT "ITEM NU
   MBER?",N
106 INPUT "COLOR?"
   ,C(N)
110 INPUT "BOX or
   CYLINDER (B/C)
   ?",B$(N)
111 IF B$(N)="C"
   THEN 700
115 INPUT "DIST UP
   ?",W
116 INPUT "DIST RI
   GHT?",U
117 INPUT "DIST BA
   CK?",V:V=-V
120 INPUT "BOX HEI
   GHT?",Z
125 INPUT "BOX WID
   TH?",X
130 INPUT "BOX DEP
   TH?",Y:Y=-Y
133 GOSUB "E"
134 G(N)=16
135 X(N,1)=U:Y(N,1
   )=V:Z(N,1)=W
140 X(N,2)=U+X:Y(N
   ,2)=V:Z(N,2)=W
145 X(N,3)=U+X:Y(N
   ,3)=V+Y:Z(N,3)
   =W
150 X(N,4)=U:Y(N,4
   )=V+Y:Z(N,4)=W
155 X(N,5)=U:Y(N,5
   )=V:Z(N,5)=W
160 X(N,6)=U:Y(N,6
   )=V:Z(N,6)=W+Z
165 X(N,7)=U+X:Y(N
   ,7)=V:Z(N,7)=W
   +Z
170 X(N,8)=U:Y(N,8
   )=V:Z(N,8)=W+Z
175 X(N,9)=U:Y(N,9
   )=V+Y:Z(N,9)=W
   +Z
180 X(N,10)=U:Y(N,
   10)=V+Y:Z(N,10
   )=W
185 X(N,11)=U:Y(N,
   11)=V+Y:Z(N,11
   )=W+Z
190 X(N,12)=U+X:Y(
   N,12)=V+Y:Z(N,
   12)=W+Z
195 X(N,13)=U+X:Y(
   N,13)=V+Y:Z(N,
   13)=W
200 X(N,14)=U+X:Y(
   N,14)=V+Y:Z(N,
   14)=W+Z
205 X(N,15)=U+X:Y(
   N,15)=V:Z(N,15
   )=W+Z
210 X(N,16)=U+X:Y(
   N,16)=V:Z(N,16
   )=W
220 GOTO 100
250 INPUT "TAPE RE
   ADY?",B$
252 IF B$="Y" THEN
   256
254 GOTO 250
256 RETURN
300 INPUT "LOAD FR
   OM TAPE?",B$
305 IF B$="Y"THEN
   315
310 RETURN
315 INPUT "WHAT TI
   TLE?",I$
320 GOSUB 250
330 INPUT #I$;X(*)
   ,Y(*),Z(*),N,S
   ,C(*),G(*),B$(
   *)
335 RETURN
350 INPUT "SAVE ON
   TAPE",B$
355 IF B$="Y"THEN
   365
360 RETURN
365 INPUT "WHAT TI
   TLE?",I$
370 GOSUB 250
375 PRINT #I$;X(*)
   ,Y(*),Z(*),N,S
   ,C(*),G(*),B$(
   *)
380 RETURN
430 GRAPH
450 GLCURSOR (30,-
   250):SORGN
455 T=25:F=35.26
460 FOR M=1TO N
465 GOSUB "DRAW"
470 NEXT M
475 GLCURSOR (0, -3
   00)
480 INPUT "NEED AN
   OTHER STRIP?",
   D$
485 IF D$="N"THEN
   491
486 GLCURSOR (-215
   ,-500):SORGN
488 K=K+215
490 GOTO 460
491 INPUT "WANT TO
   REVISE DATA?"
   ,A$
492 IF A$="N"THEN
   500
493 GOTO 100
500 INPUT "WANT AN
   OTHER VIEW?",A
   $
510 IF A$="N"THEN
```

Tilting to 90 digrees gives a top view. Large drawings are produced by pasting together strips side-by-side.

Drawings may also be stored on tape for long term storage.

Caution: Be sure to change the DIMension statements in lines 50 and 70 in accordance with the accompanying table! Failure to properly size the arrays to match the memory available in your system can result in improper operation and loss of data.

Table *Memory Requirements.*

Items	RAM Req'd	Equipment	Data Storage
5	7.3K	8K Module	About 90 in.
10	10.7K		
15	14.1K	16K Module	About 270 in.
20	17.5K	16K Module	About 360 in.
25	20.1K	26K System	

```
    990
511 INPUT "WANT AN
    OTHER SCALE?",
    C$
512 IF C$="N"THEN
    520
514 INPUT "SCALE?"
    ,S:S=126.5823*
    S
520 INPUT "HORIZ.
    ROTATION?",T
530 INPUT "VERTICA
    L TILT?",F
536 GLCURSOR (K,-2
    00):SORGN
537 K=0
540 GOTO 460
700 INPUT "DIST UP
    ?",W
701 INPUT "DIST RI
    GHT?",U
702 INPUT "DIST BA
    CK?",V
703 INPUT "RADIUS?
    ",F
704 INPUT "HEIGHT?
    ",H
705 G(N)=26
706 TEXT :CSIZE 1:
    USING "###.###
    ":LPRINT N;W;U
    ;V
707 TAB 7:LPRINT "
    R= ";F;"  H= "
    ;H:LF 1
708 INPUT "MAJOR A
    XIS (X,Y,Z)?",
    B$
709 IF B$="X"THEN
    730
710 IF B$="Z"THEN
    750
714 FOR A=15TO 375
    STEP 30
715 K=1+A/30
716 X(N,K)=U+F*SIN
    A:Y(N,K)=-V-F*
    COS A:Z(N,K)=W
717 NEXT A
718 FOR K=14 TO 26
719 X(N,K)=X(N,K-1
    3):Y(N,K)=Y(N,
    K-13):Z(N,K)=H
    +W
720 NEXT K
721 GOTO 100
730 FOR A=15TO 375
    STEP 30
731 K=1+A/30
732 Z(N,K)=W+F*COS
    A:Y(N,K)=-V-F*
    SIN A:X(N,K)=U
733 NEXT A
734 FOR K=14TO 26
735 Z(N,K)=Z(N,K-1
    3):Y(N,K)=Y(N,
    K-13):X(N,K)=H
    +U
736 NEXT K
737 GOTO 100
750 FOR A=15TO 375
    STEP 30
751 K=1+A/30
752 X(N,K)=U+F*COS
    A:Z(N,K)=W+F*
    SIN A:Y(N,K)=V
753 NEXT A
754 FOR K=14TO 26
755 X(N,K)=X(N,K-1
    3):Z(N,K)=Z(N,
    K-13):Y(N,K)=V
    -H
756 NEXT K
757 GOTO 100
760 FOR K=1TO 13
761 LINE (R(K),T(K
    ))-(R(K+13),T(
    K+13))
762 NEXT K
763 RETURN
800 "DRAW"
810 FOR I=1TO G(M)
820 R(I)=X(M,I)*
    COS T-Y(M,I)*
    SIN T
830 T(I)=-X(M,I)*
    SIN T*SIN F-Y(
    M,I)*COS T*SIN
    F+2(M,I)*COS F
840 R(I)=R(I)*S:T(
    I)=T(I)*S
842 NEXT I
844 C=C+1
845 IF C>3THEN LET
    C=0
847 GLCURSOR (R(1)
    ,T(1))
848 FOR I=2TO G(M)
850 LINE (R(I-1),T
    (I-1))-(R(I),T
    (I)),0,C(M)
860 NEXT I
865 IF B$(M)="C"
    THEN GOSUB 760
870 GLCURSOR (0,0)
880 RETURN
940 "E":TEXT :
    CSIZE 1
952 USING "###.###
    ":LPRINT N;W;U
    ;V
953 TAB 7:LPRINT Z
    ;X;Y
954 LF 1
955 RETURN
990 GOSUB 350
995 PAUSE "BY-BY!"
999 END
STATUS 1       2961
```

INSIDE THE PC-1500

This is the fourth part of the current series on the Sharp PC-1500. (Most of this information is also applicable to the Radio Shack PC-2.) This valuable data is supplied through the dedicated efforts of *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* This issue picks up from where we left off in Issue 33 of *PCN.* That is, with a continuation of a brief summary of a number of the routines contained in the ROM of the PC-1500. Specifically, Norlin was discussing the subroutine at address &D2EA that is capable of searching for a BASIC program line

A data byte must follow the instruction calling this subroutine. If the line sought is not found, this data byte is added to the return address, with UH containing 0B.

As an alternative, a search for a specified line number (but not for a label) may be begun at the address pointed to by START OF EDIT. This is done by subroutine D2E0, which is identical in all other respects to the above subroutine.

KEYBOARD

POLL OF KEYBOARD

Subroutine E42C loads A with the code for the key (other than BREAK) that is currently depressed. If no key is depressed, A will be loaded with zero. Register Y is unchanged by the subroutine.

The codes determined by the keys are as follows:

01 SHIFT	0F OFF	19 RCL	2D -	35 5	43 C	4B K	53 S
02 SML	11 F1	1B DEF	2E .	36 6	44 D	4C L	54 T
08 left	12 F2	1F MODE	2F /	37 7	45 E	4D M	55 U
09 template	13 F3	20 SPACE	30 0	38 8	46 F	4E N	56 V
0A down	14 F4	28 (	31 1	39 9	47 G	4F O	57 W
0B up	15 F5	29)	32 2	3D =	48 H	50 P	58 X
0C right	16 F6	2A *	33 3	41 A	49 I	51 Q	59 Y
0D ENTER	18 CL	2B +	34 4	42 B	4A J	52 R	5A Z

WAIT FOR INPUT FROM KEYBOARD

Subroutine E243 polls the keyboard until a key code is obtained. Except for the SHIFT, SML, and DEF keys (which act as prefixes), the use of a key produces a return, with a character code in A. The 7-minute powerdown timer operates during the wait. Register Y is unchanged.

The codes obtained for display characters are their ASCII codes.

The codes for control and editing keys requiring the SHIFT prefix are as follows:

1A CA 1C INS 1D DEL 1E SHIFT/MODE

The codes for alphabet keys prefixed by DEF are their ASCII codes plus 40. The other keys accepting the DEF prefix use these codes:

80 DEF SPACE 9D DEF =

The code for BREAK is 0E. If BREAK has been keyed, subroutine E243 will continue to return with this code in A each time it is called, until bit 1 of address F00B in the I/O Port is cleared.

DETECTION OF BREAK KEY

Call A6 (E451) will clear flag Z if BREAK has been keyed. There is no effect on the CPU registers or on other flags.

If BREAK has been keyed, this subroutine will continue to return with flag Z cleared each time it is called, until bit 1 of address F00B in the I/O Port is cleared.

DISPLAY

CURSOR POINTER

The contents of the Cursor Pointer (7875) specify a column of dots in the LCD display; columns are numbered from 00-9B (0-155 decimal).

TO CLEAR DISPLAY

Call F2 (EE71) clears the display. Registers X and Y are unchanged.

TO DISPLAY DATA IN R0 (OR STRING POINTED TO BY R0)

Subroutine E8CA may be used to display numeric data contained in R0 (given in decimal or binary), or a string pointed to by R0. Display will be in default format. Strings are displayed left-justified, and numeric data right-justified. Precede by storing 20 as the value of the Display Parameter (7880).

TO DISPLAY NUMERIC CONTENTS OF R0, FORMATTED

The display will be formed beginning at the location pointed to by the Cursor Pointer (7875). Contents of R0 must be in decimal.

(1) Store format specifications as follows:

7895: Editing characters. 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific format

7896: Number of characters preceding decimal point, including spaces for sign (and commas, when specified).
7898: Number of characters including and following decimal point

(2) Load A with 00, 01, or 02, to specify the position of the characters to displayed, according to the following:

A=00: data displayed right-justified in 13-character block
A=01: data displayed left-justified in block of whatever size is required (up to a maximum of 16 characters)
A=02: data displayed right-justified in 26-character block

Then execute Call 96 (EA78), to form a string of characters (in the area 7A10-7A34) to represent the data to be displayed.

(3) Execute the instruction STX U; then execute Call 92 (ED00).

The cursor pointer will be updated to point to the next available display position. Flag C will be set if the display has been filled.

TO DISPLAY A STRING POINTED TO BY R0

Execute Call DC (DEBC), followed by Call 92 (ED00). The cursor pointer will be updated to point to the next available display position. Flag C will be set if the display has been filled.

TO DISPLAY CONTENTS OF OUTPUT BUFFER

Subroutine ECFA displays the contents of the Output Buffer starting at 7B60, ending when the display is filled. Characters are entered into the display beginning at the position specified by the Cursor Pointer.

TO DISPLAY A SEQUENCE OF CHARACTERS

Subroutine ED3B will display a sequence of characters whose codes are located anywhere in memory. Precede by loading U with the beginning address of the stored codes, and XL with the number of codes. The characters will be entered into the display beginning at its left end.

TO DISPLAY A PROGRAM LINE

The line to be displayed must be put into the Input Buffer, starting with its two-byte line number, omitting the link byte, and continuing with the codes for the BASIC statements in the line. A line may be placed into the Input Buffer in that form in any of the following ways:

(a) Subroutine D26F may be used to place the first BASIC line into the Input Buffer; the SEARCH pointer will be set to point to that line.
(b) Subroutine D2EA may be used to locate the line whose number has been loaded into U. (A data byte, which will be added to the return address if the specified line does not exist, must follow the instruction calling this subroutine.) After execution of subroutine D2EA, load XH and XL with the contents of 78A6 and 78A7; decrement X twice; and execute subroutine D2D0. The specified line will be in the Input

Buffer, and the SEARCH Pointer will Point to that line.
(c) The Program line that follows the one currently pointed to by SEARCH may be Placed into the Input Buffer, with automatic updating of the SEARCH Pointer, by execution of subroutine D2B3.

After the BASIC line has been Placed into the Input Buffer, it is displayed by execution of subroutine E8CA, Provided that an appropriate value has been stored as the Display Parameter (7880). With 10 as that value, the line is displayed from its start, without a colon following the line number in the display; with 14, the colon will be included.

The cursor will be included in the display if the Display Parameter contains 50 (no colon) or 54 (colon included). Register Y must be loaded with the address (in the Input Buffer) at which the cursor is to appear.

TO DISPLAY CONTENTS OF INPUT BUFFER, WITH TOKENS EXPANDED

Subroutine E8CA, with the Display Parameter (7880) containing zero, displays the contents of the Input Buffer, starting at 7BB0, ending when a 0D code is reached. (The Input Buffer may be filled with 0D codes Preceding this, by execution of subroutine D02B.)

The cursor will be included in the display if 40 is used as the Display Parameter value; Register Y must be loaded with the address (in the Input Buffer) at which the cursor is to appear.

TO DISPLAY A SINGLE CHARACTER

Subroutine ED4D enters the character whose code is in A into the display, at the Position Pointed to by the Cursor Pointer (7875). The Cursor Pointer will be incremented by 6 each time a character is added to the display, Pointing to the next character Position. If the right end of the display is reached, additional characters wrap around to the left end of the display.

The character whose code is in A may also be Placed into the display by subroutine ED57. In this case, there will be no automatic increment of the Cursor Pointer.

GRAPHIC DISPLAY

Subroutine EDEF displays the dots specified by the contents of A, in the column Pointed to by the Cursor Pointer. The Cursor Pointer is not incremented automatically; Call 8E (EDB1) may be used to do so.

The dots displayed are determined by the bits contained in A:

01	Top dot	10	5th
02	2nd	20	6th
04	3rd	40	Bottom
08	4th		

TIMED DELAY

Call AC (E88C) produces a delay, equal in duration to the product of 1/64 second by the contents of Register U. If the BREAK key is pressed during execution of this routine, the timing cycle ends, and there is a return with flag C set. Register Y is not affected.

This subroutine may be used to produce the equivalent of a PAUSE by effecting a temporary delay while information is being displayed.

BEEP

Subroutine E669 is equivalent to BEEP 1. It ignores BEEP ON/OFF.

Subroutine E66F produces a tone with pitch determined by UL, and duration determined by X. BEEP ON/OFF is ignored. The frequency of the tone is approximately 59230/(7.47+UL) cps (calculated in decimal). The duration (number of cycles) is &100 less than the contents of X.

POWERDOWN

Subroutine CD71 is equivalent to OFF.

Subroutine E33F is equivalent to the automatic 7-minute powerdown.

TERMINATION AT 'READY'

A machine-language program may be terminated by setting the computer to 'Ready' mode, in which it is awaiting further instructions from the keyboard.

TERMINATE WITH DISPLAY

Call 46 (CA8D) sets 'Ready' mode, retaining the display contents.

If subroutine E8CA has been used to produce a display with the cursor contained in it, the cursor left and cursor right keys will be enabled. (In other cases, however, use of the cursor control keys may result in a meaningless display.)

TERMINATE WITH PROMPT MARK

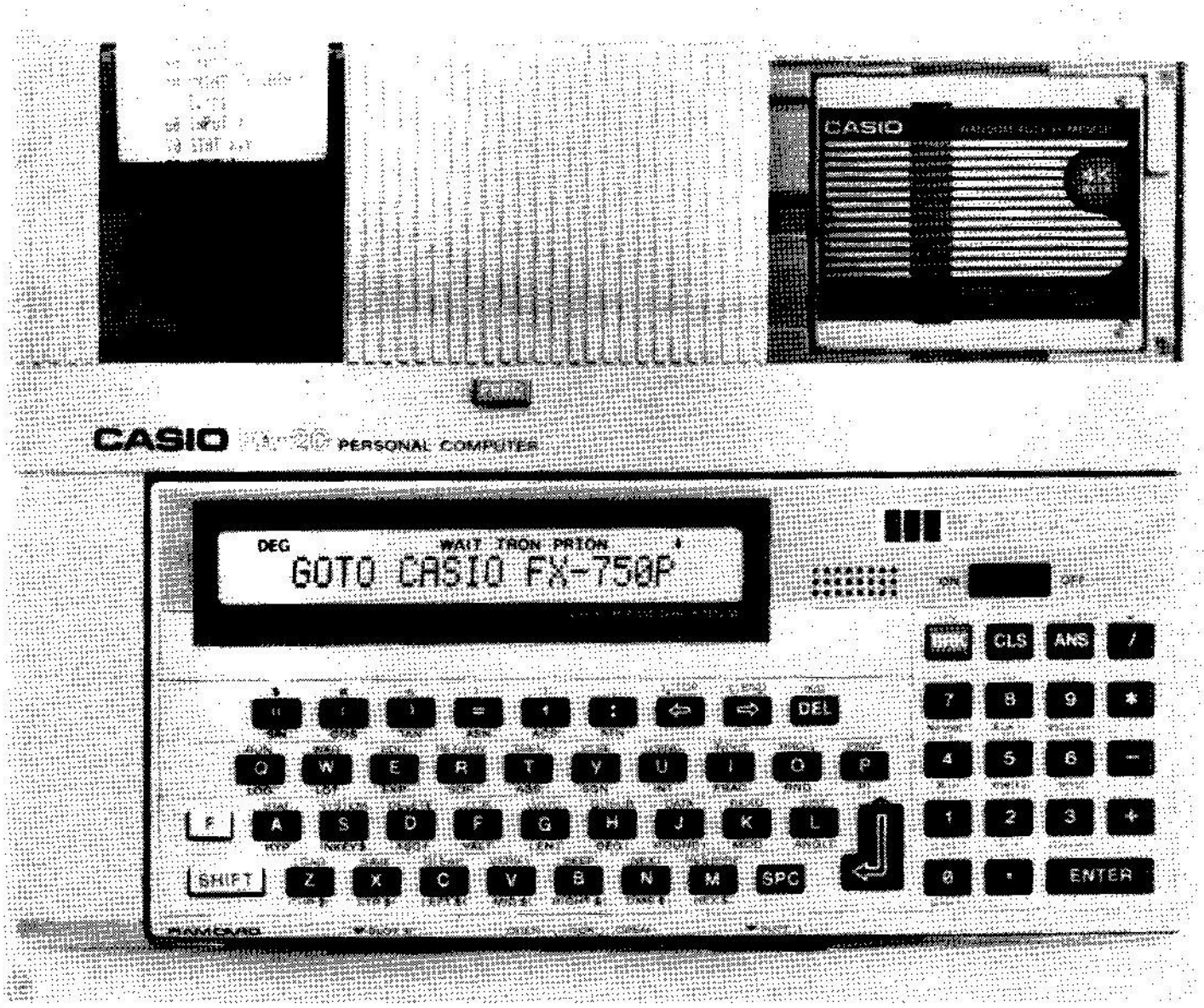
Call 42 (CA58) will place the computer in 'Ready' mode, with the prompt mark displayed.

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

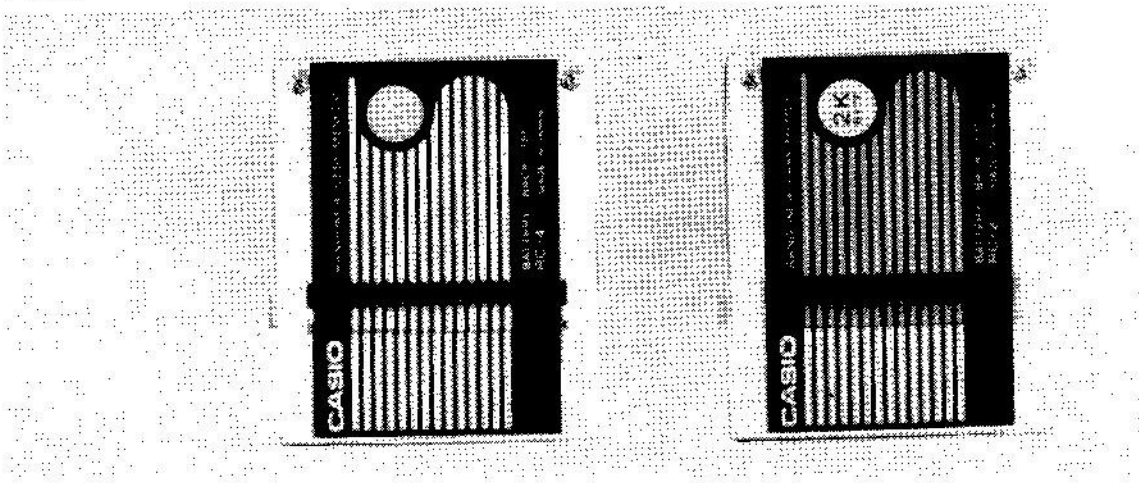
Issue 35 - September/October

Photo *Casio FX-750P Shown Mounted on Companion FA-20 Printer/Cassette Interface.*



Another *Personal Information* ™ product.

Photo *Casio RC-2 and RC-4 RAM Modules Can Expand Memory in the Casio FX-750P.*



CASIO INTRODUCES MODEL FX-750P

Casio has announced the availability of a compact hand-held scientific computer named the model FX-750P.

Designed primarily for engineers and those that work in the sciences, it includes scientific functions such as hyperbolics, stastical functions such as standard deviation, and powerful analytical capabilities, such as performing linear regression. The unit is also programmable in the BASIC language. It contains 4K of user RAM in its standard configuration. However, memory may be expanded to 6 or 8 kilobytes. This is accomplished by sliding in unique battery-backed "RAM cards" into the front of the unit. Two types of these cards are currently available. The model RC-2 contains 2 kilobytes of memory. The model RC-4 holds 4 kilobytes of memory. The battery-backed memory cards may be used to hold frequently utilized programs. These programs can then be quickly accessed by plugging in the appropriate module containing the desired program.

The FX-750P has a 24-character, single-line liquid-crystal alphanumeric display. The display also contains a number of status annunciators.

There are 58 keys on the unit. Almost all of these may be used in two or three modes through the use of the "F" (function) and "Shift" keys. When in the special modes these keys can invoke BASIC statements or mathematical functions.

The FX-750P can be connected to the Casio FA-20 printer/cassette interface. This permits the storing of programs on an audio cassette through the use of a separate tape recorder unit, or the printing of programs and/or data on a thermal printer that is part of the FA-20.

Casio states that over 30 software titles are available for the FX-750P. These include programs dealing with financial analysis, games, business, electronics, education, statistics, math and scientific/engineering disciplines.

The Casio FX-750P is said to have a U.S. list price of $129.95 and to be available from Casio sales outlets.

REVIEW OF THE SHARP PC-1261

Sharp Electronics has released a pair of new pocket computers dubbed the models PC-1260 and PC-1261. The models are equivalent except that the PC-1260 contains 4.5 kilobytes of user RAM while the PC-1261 contains 10.5 kilobytes of RAM. This review specifically covers the PC-1261.

The PC-1261 is identical in size to the earlier models PC-1250/1251. In fact, it may be used with the CE-125 printer/casette interface that was originally provided for those models. Despite its small size, the 1261 has a number of exciting additional features. For starters, its display is capable of showing two lines of 24 characters each. While the characters are slightly smaller in size than those of the earlier 1250, they are still highly readable. Of course, the 1261 includes the same kind of variable contrast control as that provided in its predecessor. Thus, the intensity of the display can be easily changed to compensate for actual lighting conditions. Most of the annunciators on the display of the 1261 are

indicated by the presence or absence of tiny cursor bars at the bottom of the screen. These correspond directly to legends that are provided on the case of the unit just below the LCD. The two-line display is an exciting advance for pocket computer users. It makes it much easier to input information as the query can be placed on one line while the answer appears on the second. Additionally, when outputting information, one line can serve for labels or identifiers while the second line reflects corresponding values.

The keyboard on the PC-1261 is identical in size and number of keys to the earlier PC-1250. However, Sharp, has made several significant improvements. For example, the right and left parentheses have been positioned over the digit 1 and digit 2 keys. They used to be over the scroll-up and scroll-down keys where they could cause a lot of trouble if you forgot to hit the shift key when you wanted to invoke a paren sign. Another change to the keyboard is the removal of the exponent character that was activated using the shifted + sign on the 1250. Now you just use the letter E. There are also two new functions that may be invoked from the keyboard that were not on the earlier 1250. These are the HELP function and SMALL function. The former is invoked by using shift-7. The latter by using shift-8. The SMALL function enables one to use upper and lower case characters. The HELP function brings up a compact reference that lists all of the possible key words available in the unit. This function can be expanded to provide examples of the use of each key word. The HELP function can even be used to bring up a character code table. This table lists the hexadecimal code for each displayable character.

The new PC-1261 is equipped with an impressive amount of ROM: 40 kilobytes! This amount of ROM enables the unit to contain an enhanced BASIC interpreter, the HELP function, and other features not found in previous models having such small physical size.

The PC-1261 has a powerful new feature that Sharp refers to as "Easy Simulation Programming." Perhaps a more appropriate description is "automated calculation." This feature allows you to put a number of formulas in memory, such as : A = B + C - D. The lefthand side of the equation must be preceded by a # sign and may only contain one variable. The righthand side of the equation may contain up to 9 variables. These formulas are saved in a special section of user RAM. The amount of memory set aside for this purpose may be specified by the user in 128-byte increments. Memory used for this purpose (over and above an initial 128-byte block) is taken away from BASIC program space.

In order to automatically calculate the value of an expression, you merely enter a # sign followed by the identifying variable (when the PC is in the RUN mode) and then the ENTER key. Doing this causes the variable names that are on the right side of the expression to appear on the top line of the display. The numerical value desired for each variable may then be entered on the second line of the display underneath the corresponding variable names. When the last value for an expression has been entered, the PC-1261 will automatically calculate the value for the expression and display it under the variable name assigned to the lefthand part of the equation. Once an initial set of values has been entered for an expression, you can quickly obtain a whole series of answers by varying parameter values at will. This is accomplished by moving the cursor to the variable values that you want to change and entering new data. Each time new data is entered, the expression is recalculated and displayed for the user.

The active engineer, scientist, mathematician, or businessman will like this automated formula-solving capability. A number of your favorite formulas can be stored in memory for instant recall and calculation. Since the PC-1261 allows variables to have multiple character names, you can even set up formulas using plain English statements such as NET = SALES - COST. If you do a lot of repetitive formula solving, you will definitely appreciate this somewhat novel feature.

The PC-1261 is a somewhat unique creature in that, along the lines of the 1250/1251, it tries to maintain programming compatability with the original PC-1211 while offering an expanded instruction set and advanced capabilities. Where it succeeds, of course, it means that PC-1211, PC-1250, PC-1250A, PC-1251, PC-1260 and PC-1261 programs are "upwards" compatible. (That "upwards" means that a program created on an earlier model should run on a later version. You may not, however, be able to go the other way.)

The best way to gain an overview of the 1261's operational and programming capabilities is to show a synopsis of its commands and keywords: ABS, ACS, AND, AREAD, ASC, ASN, ATN, BEEP, CHAIN, CHR$, CLEAR, CLOAD, CLOAD?, CLS, CONT, COS, CSAVE, CURSOR, DATA, DEG, DEGREE, DIM, DMS, END, EQU #, EXP, FOR/TO/STEP, GOSUB, GOTO, GRAD, IF/THEN,

INKEY$, INPUT, INPUT #, INT, LEFT$, LEN, LET, LIST, LIST #, LLIST, LLIST $, LN, LOG, LPRINT, MEM, MEM #, MERGE, MID$, NEW, NEW #, NEXT, NOT, ON...GOSUB, ON...GOTO, OR, PASS, PAUSE, PI, PRINT, PRINT #, RADIAN, RANDOM, READ, REM, RESTORE, RETURN, RIGHT$, RND, RUN, SGN, SIN, SQR, STOP, STR$, TAN, TROFF, TRON, USING, VAL and WAIT.

Many of these will be familiar to even early PC-1211 users, but note that there are many new ones, such as those ending in $ (for string operations) and some of those ending in the # sign (such as EQU #) which are associated with the "Easy Simulation Programming" mode.

PC-1500 users as well as 1250 afficionados may also note that there are no PEEK or POKE directives. This may be a disappointment to those people who like to delve into machine language programming. It could be a real challenge - perhaps impossible - to "crack" into this PC and utilize it at the machine level. On the other hand, the lack of PEEK and POKE coupled with the fact that there is a PASS (password) statement, could be encouraging to software developers. The PASS command allows you to effectively protect a program so that it cannot be listed, modified or copied. In earlier models, such as the 1250, the PASS directive could be circumvented by an astute programmer using the PEEK and POKE directives. Since there are lots of astute programmers around these days (especially amongst readers of *PCN*), trying to protect a program using the PASS command has not been a very secure method in the past. The lack of PEEK and POKE in the 1261 may make the password option more effective as a security device.

PC-1500 programmers might note that statements associated with printing operations, such as LCURSOR, are absent. The 1261 appears to be more difficult to program for use with a printer such as the CE-125 or CE-126P. This appears to be due primarily to the fact that in normal use only two fields can be printed on a 24-character line. Alphanumeric (string) data is printed left-justified within a 12-character field. Numeric data is printed right-justified in a similar 12-character field. Two 12-character fields make up a 24-character printer line. This is somewhat restrictive. You might think this could be circumvented by the use of LPRINT USING statements. While the instruction manual suggests that numeric and alphanumeric data can be mixed within a printed line through LPRINT USING formats (such as ##.##&&&&&####&&&&&), tests indicate you can only switch from string data to numeric data (or vice-versa) once within a printed line! This makes it tough to print fancy stuff (do "pretty-printing").

[There is a way around this limitation. It requires that you convert numeric data into string format. You can then build up, say, a 24-character string-variable that combines several alphanumeric segments (including those segments that were converted from numeric format) under one variable name. This entire string can then be printed out by declaring a USING format that consists of, say, 24 alphanumeric characters.]

Considering that the PC-1261 was really designed to fit into a shirt pocket sans printer, it is not surprising that its printer-driving software is somewhat limited compared to, for instance, that of the PC-1500.

All-in-all, the PC-1261 seems to be a nice advance in the state-of-the-art. Having 10K of user memory available in a truly pocketable unit is nice. Being able to maintain your library of programs from the earlier PC-1211/PC-1250 models is definitely a plus factor. And, having advanced features such as 2-dimensional arrays and the "automatic calculation", (coupled with the extra memory) is a nice kicker. If you have been thinking about upgrading your pocket computing power, take a good look at the PC-1261. It may be just the solution to your advancing needs!

POCKET WORDPRO

This machine language program converts the Sharp PC-1500/PC-1500A or Radio Shack PC-2 into a simple word processor. Admittedly, it is stretching things a bit to call it a word processor. It will not relocate paragraphs, search for words, do automatic page numbering or print headers and footers. But, it is easy to learn to use and its capability may surprise you.

When the program is in use, all but about 500 bytes of RAM is used as a text storage area. Using a PC-1500A equipped with a CE-161 (16K) memory module, this means that you have over 22,000 bytes of character storage! You simply type in the text you want printed and then tell the computer to print it. Printing is done in vertical format, separated into pages of twelve 42-character lines each. The computer will determine where to end a line to avoid splitting a word and it will left-justify each printed line. You may indent paragraphs and center headings. However, centering of lines is not automatic.

Editing is simplified by the fact that you use the cursor controls the same way you are accustomed to using them -- with a few improvements! Insertion and deletion are auto-repeat. The computer will continue inserting or deleting as long as you keep the key held down.

Text may be saved, loaded or merged from cassettes, in the same simple way it is done for BASIC programs!

Loading the Program

You will need to load the ML codes in the listing, using either POKE statements or a Monitor program. It is 503 bytes long, but needs to be done only once. You can save the program on cassette. You must have either a CE-155, CE-159 or CE-161 memory module (8K or 16K expansion). Before testing the program, save it on tape using:

CSAVE M "POCKET WORDPRO";&3800,&39F6

Then, if you have made any errors in entering the program, at least you won't have to re-enter the whole listing.

After you have a correctly-entered program, you can relocate it by specifying a different CLOAD M address when loading. The best choice is made by loading with CLOAD M 256 * PEEK &7863. This will insure that you are using all the memory your computer has available. (This program will *not* work if loaded into the 7C00 - 7FFF area of the PC-1500A.)

Since PRO and REServe modes have no use with this program, the PC should be LOCKed in the RUN mode as a precaution against disaster.

Executing the Program

There are two addresses used in CALL statements. One is used when starting up the program and the other when continuing operations following a BREAK. The first of these is the beginning address at which the program is located. This would be &3800 when the program is located as shown in the accompanying listing. If you have relocated the program as suggested above, the address would be 256*PEEK &7863. When you begin execution with CALL &3800 (or CALL 256*PEEK &7863), you will clear the entire text area. Hence, you will want to be careful not to execute this call later on, unless you want to deliberately clear whatever text you have entered.

The second of the addresses to be used in a CALL is &381D (or &1D+256*PEEK &7863). To make things convenient, you can put this address into variable C. Then, if you need to continue the program after a BREAK (either intentional or unintentional), you can simply execute CALL C.

Entering and Editing Text

After you have started up the program will CALL &3800 (or, if relocated, CALL 256*PEEK &7863), you should see an underscore indicating the cursor position at the left end of the display. Start typing. If you want to indent your initial paragraph by 3 spaces, type 3 space before beginning your first sentence. All characters, including the quotation mark, are usable. If you want to enter any of the characters labeled above the REServe keys, you will need to SHIFT for them. Note that you may use the SML key in the usual way.

The cursor-left/right keys operate in the normal fashion, with auto repeat when held.

The up-arrow and down-arrow keys will change the displayed portion of text by 26 characters. They also automatically repeate when held down. If up-arrow is held, you can reach the beginning of text fairly quickly. Down-arrow may be used to locate the end of text. Note that the operation of these keys is somewhat similar to their operation when scanning back and forth through the lines of a BASIC program.

Insert and Delete operate about the same as they normally do. However, for multiple insertions or deletions, you do not have to repeat SHIFT with each use. Furthermore, the insertion and deletion continue rapidly when the corres- ponding key is held down. Thus you may delete sections of text rather quickly.

Keying ENTER will display a solid rectangle that is used to mark the end of a paragraph. This makes it easy to locate paragraph endings as you scan rapidly through the text using the up-arrow or down-arrow keys. Remember to type three spaces following the ENTER if you are indenting paragraphs. Note also that the ENTER key should be used immediately following the concluding punctuation mark of the last sentence in a paragraph.

To see how many bytes of unused memory are remaining, just hold down the RCL key. When you release this key, you will be back to where you were before.

To print the entered text, key DEF P. A blank line is printed between paragraphs, except at the top of a page.

Note that a sequence of more than 42 characters that are not separated by spaces will not fit on a 42-character line. In the unlikely event that such a sequence is encountered, the printer will stop. You will have to do appropriate editing to

Program *Word Processor ("WORDPRO") for the PC-1500/PC-2.*

```
3800 FD 58 FD 40
3804 B5 F5 FD CA
3808 CA 65 CA 50
380C CC 64 4A 00
3810 46 CA 52 FD
3814 6A CC 65 BE
3818 DF E2 BE D3
381C C5 BE D0 2B
3820 B5 40 AE 78
3824 80 CC 50 CA
3828 50 FD 98 5A
382C B0 6A 19 F5
3830 88 03 FD 1A
3834 BE E8 CA BE
3838 E2 43 28 CC
383C 50 81 09 45
3840 99 03 46 46
3844 CA 67 CD 42
3848 EB 7B 0E 40
384C 6C 08 89 0C
3850 5E B0 56 90
3854 21 54 46 05
3858 99 33 9E 28
385C 6C 0C 80 0C
3860 15 9B 2F 54
3864 5E CA 91 41
3868 44 56 9E 45
386C 6C 0A 89 0D
3870 B5 1A FD CA
3874 47 9B 03 DD
3878 9B 06 44 8E
387C 0A 6C 0B 89
3880 14 B5 E6 FD
3884 CA FD 42 F4
3888 78 65 BE DF
388C E2 81 02 CC
3890 65 5A B0 9E
3894 6E 6C 1C 89
3898 04 B5 0C 8E
389C 06 6C 1D 89
38A0 48 B5 08 28
38A4 FD C8 FD 98
38A8 15 8B 28 14
38AC 5A B0 10 FD
38B0 CA 6C 08 8B
38B4 10 68 20 05
38B8 2A A4 41 24
38BC 28 99 08 CC
38C0 52 43 00 8E
38C4 07 FD 6A 44
38C8 45 61 93 04
38CC CC 50 6A 19
38D0 F5 88 03 FD
38D4 1A BE E8 CA
38D8 BE E2 43 EB
38DC 7B 0E 40 28
38E0 CC 50 FD 8A
38E4 A6 9B 43 9E
38E8 9D E3 7B 0E
38EC BF 6C 0D 89
38F0 02 68 7F 6C
38F4 80 83 19 6C
38F8 20 81 15 14
38FC B1 B0 FD CA
3900 84 A7 78 52
3904 81 04 4E FF
3908 93 D6 A4 0E
390C CC 50 9E AD
3910 6C 19 89 17
3914 45 99 03 F4
3918 78 52 64 BE
391C DF E2 CD 10
3920 00 BE EE EF
3924 BE E4 2C 93
3928 05 9E F7 6C
392C 90 99 FB A5
3930 B0 00 DD 9B
3934 0C FD 98 45
3938 99 03 DD AE
393C 77 4E AE 79
3940 F2 DD AE 79
3944 F4 46 46 CA
3948 67 CC 65 CA
394C 54 EB 79 F0
3950 FF BE AC 97
3954 B5 D8 FB B1
3958 12 FD C8 48
395C 7B 4A 7F 43
3960 B5 00 43 43
3964 0E BE AC 07
3968 BE A7 69 CC
396C 54 ED 77 4E
3970 F0 8B 0C E9
3974 77 4E 0F FD
3978 8A FD C8 B7
397C C6 89 52 ED
3980 77 4E 0F 8B
3984 09 E9 77 4E
3988 F0 05 8B 05
398C 8E 16 45 89
3990 0E FD 8A DD
3994 B7 C7 89 3D
3998 FD C8 BE A9
399C D5 8E 49 B7
39A0 20 9B 15 46
39A4 CA 56 6A 00
39A8 05 8B 1B 44
39AC B7 7F 89 05
39B0 AE 77 4E 8E
39B4 11 60 6E 2A
39B8 99 12 47 B7
39BC 20 8B 05 62
39C0 33 08 3E 33
39C4 44 44 FD A8
39C8 CA 54 FD 2A
39CC CC 56 BE B4
39D0 F4 FD 8A 99
39D4 7F FD C8 48
39D8 7B 4A 7F B5
39DC 00 43 43 B5
39E0 C0 43 B5 FD
39E4 0E BE AC 07
39E8 BE A7 69 FD
39EC 8A 9B 9E BE
39F0 AC BB FD 1A
39F4 9E CD 00
```

correct this condition before trying to print again.

The BREAK key (actually, the ON key) may be used to discontinue execution of the program. There are two reasons for wanting to do this: (1) you want to save or load text on cassette, or (2) you want to quit and use your computer for some other task. It is also possible to unintentionally press this key. To get back into the program, you can execute CALL C -- provided that you have previously stored &1D + the starting address into variable C, as suggested earlier. Your text will then reappear in the display, right where you left off.

If you should use BREAK to interrupt the printer while it is busy printing your text, you should execute TEXT to reset the printer.

The MODE, CL, OFF and Reserve-Group-Selector keys are inoperative while the program is running.

Using Cassette Tape

Following a BREAK, the CSAVE directive may be executed (in immediate mode) to record the text that is stored in memory. The recording may also be verified using CLOAD? (As usual, the use of a specific file name is optional.)

A previously-recorded file of text may be loaded into memory using CLOAD (again, in immediate mode) following a BREAK. Text already in memory (if any) may first be cleared by the initialization routine (executed by CALL starting address).

Executing MERGE will append recorded text to that currently in memory. (It does *not* prevent you from editing earlier portions, as is the case with BASIC programs.)

Centering

A heading may be centered by treating it as a paragraph and indenting an appropriate number of spaces. For example, to center a 12-character heading, note that (with a 42-character line) there will be 30 spaces left over. Therefore, indent 15 spaces, then type in the heading and follow it by ENTER. (It will look a little bit funny if this centered heading happens to be located at the bottom of a page -- you just have to take your chances!)

Note that these centered heading may occur only: (1) at the very beginning of text, or (2) immediately following the end of a paragraph that has been terminated using the ENTER key.

To Quit

When you are finished using the program and want to do something else with your computer, key BREAK and then execute NEW 0.

Summary of Operation

1. Load program with CLOAD M 256*PEEK &7863.
2. Execute C=&1D + 256 * PEEK &7863.
3. Initialize with CALL 256 * PEEK &7863.
4. Start typing. (Or, key BREAK and load in previously-recorded text file using CLOAD, then CALL C.)
5. Cursor controls operate with auto-repeat. Key ENTER to end a paragraph. Hold the RCL key to see how much memory is available.
6. To print, key DEF P.
7. To save on tape, key BREAK, execute CSAVE, return to program via CALL C.
8. To quit press BREAK then key NEW 0.

Do you know of any other word processor whose instructions are that simple? Perhaps we ought to try keeping things in perspective, however. This isn't really a *serious* word processor, let's admit it. (How *could* it be, when there isn't even an apostrophe?) In spite of that, many users may find it useful.

Thanks for this clever machine language program should be directed to: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158*

PERSPECTIVE PLOT

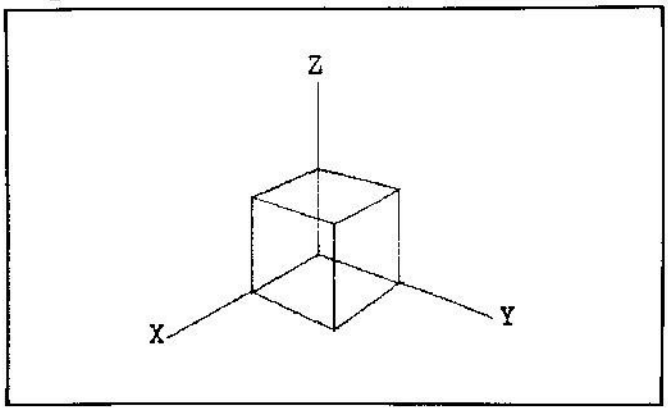
You can use this program on a Sharp PC-1500 or Radio Shack PC-2 to plot line segments and form a perspective drawing.

The line segments to be projected onto a plane of projection lie within a cube. The size of this cube is specified by the user in response to an INPUT prompt. The X, Y and Z coordinates of the end points of these segments must be specified (as positive numbers) in DATA statements starting at line 100. The letter S as a DATA item specifies the start of a new sequence of connected line segments. The final DATA item should be the letter E, which is interpreted as the end of the list. Here is an example illustrating the use of DATA statements to produce a cube:

```
100 DATA 0,0,1,1,0
    ,1,1,0,0,0,0,0
    ,0,0,1,0,1,1,1
    ,1,1,1,1,0
101 DATA 0,1,0,0,1
    ,1,S,1,1,1,1,0
    ,1,S,0,0,0,0,1
    ,0,S,1,1,0,1,0
    ,0,E
```

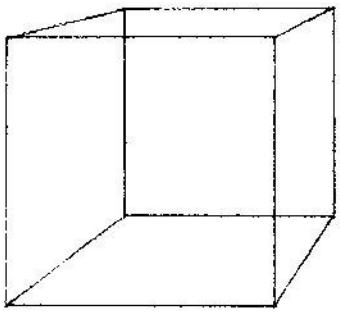
The relationship between the X, Y and Z coordinates is illustrated in the accompanying diagram.

Diagram *X, Y and Z Coordinates for Simple Cube.*

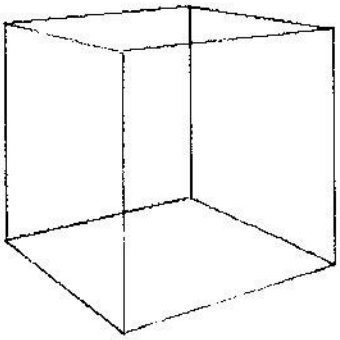


The point from which the cube (or object within a cube) is to be viewed is specified by the user in response to prompts. Its location, *relative to the center of the cube,* is L units in the X- direction, M in the Y-direction and N in the Z- direction. Note that L, M and N are measured in the same units as the X, Y and Z coordinates. The view point may be chosen in any direction from the center of the cube containing the object projected. However, it may

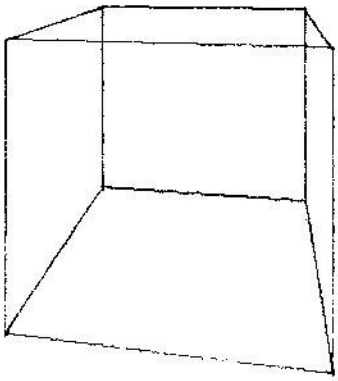
Diagram *Data and Several Sample Perspectives.*



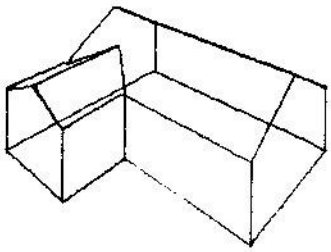
1-PT PERSP, FROM
4, 1.5, 1



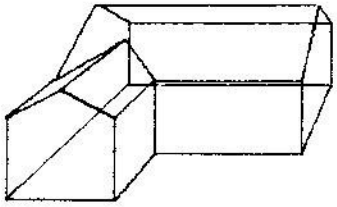
2-PT PERSP, FROM
3, -2, 1



2-PT PERSP, FROM
2, 0.3, 0.8



2-PT PERSP, FROM
75, 65, 50



1-PT PERSP, FROM
150, 60, 55

Diagram *Several Perspectives of a Cube.*

```
100:DATA 5,50,0,5,
    50,15,20,50,25
    ,35,50,15,35,5
    0,0,5,50,0
101:DATA 5,35,50,0
    ,35,20,0,50,20
    ,0,50,0,0,50,0
    ,15,50,10,20,5
    0,20,15
102:DATA 35,20,15,
    35,50,15,5,35,
    20,0,35,20,15,
    27.5,10,20,35,
    0,15,20,0,25,2
    0,50,25
103:DATA 5,50,0,0,
    5,0,0,5,50,0,5
    ,5,0,0,5,0,15,
    5,5,50,15,5,0,
    15,20,0,25
104:DATA 5,50,0,15
    ,35,0,15,5,27.
    5,10,20,50,10,
    20,50,20,15,50
    ,20,0,E
```

Program *Perspective Plot.*

```
10:INPUT "VIEW PO
   INT: L? ";L,"M
   ? ";M,"N? ";N
11:INPUT "1 OR 2-
   POINT? ";T
12:INPUT "CUBE SI
   ZE? ";D:D=D/2
13:CLS :IF T=2
   THEN 30
20:IF ABS L<=D
   THEN 60
21:A=1/(1-D/ABS L
   ),B=1/(1+D/ABS
   L),C=1/(ABS L+
   ABS M)
22:K=100*C/D,HX=K
   *M,HY=K*L
23:VX=K*N*SGN L,V
   Y=0,VZ=K*ABS L
24:P=A*B:IF ABS M
   <DLET P=A*C*
   ABS L
25:PX=P/L,PY=0,PK
   =P+P*D/L
26:H0=100*A/P:IF
   L*M<0LET H0=21
   6-H0
27:K=-100*C/P:IF
   N>DLET V0=K*B*
   (ABS L+N),VB=2
   *V0*A:GOTO 40
28:V0=K*A*(ABS L-
   N):IF N>-DLET
   VB=2*K*A*ABS L
   :GOTO 40
29:VB=2*V0*B:GOTO
   40
30:IF ABS L<=DAND
   ABS M<=DTHEN 6
   0
31:A=D*(ABS L+ABS
   M),B=D*ABS (
   ABS L-ABS M),C
   =L*L+M*M
32:K=100/A,HX=K*M
   ,HY=K*L
33:K=K/SQR(C+N*N),V
   X=K*N*L,VY=K*N
   *M,VZ=K*C
34:IF ABS L<DOR
   ABS M<DLET P=(
   B/A/(C-A)+1/(C
   -B))/2:GOTO 36
35:P=(1/(C+B)+1/(
   C-B))/2
36:PX=P*L,PY=P*M,
   PK=P*(C+(L+M)*
   D)
37:H0=100/P/(C-B)
   :IF M*SGN L>L*
   SGN MLET H0=21
   6-H0
38:S=A*SGN (N-D):
   V0=-K/P*(C*D+N
   *S)/(C+S):IF
   ABS N>DLET VB=
   2*C*V0/(C-S):
   GOTO 40
39:VB=2*C*V0/(C-A
   *N/D)
40:HK=(HX-HY)*D,V
   K=(VX+VY-VZ)*D
   ,V0=V0-20,S=1E
   99,E=2E99,F=0:
   RESTORE 100:
   GRAPH :SORGN
41:READ X:IF X>=S
   THEN 45
42:READ Y,Z:P=PK-
   PX*X-PY*Y,H=H0
   +(HK-HX*X+HY*Y
   )/P,V=V0+(VK-V
   X*X-VY*Y+VZ*Z)
   /P
43:IF FLINE -(H,V
   ):GOTO 41
44:GLCURSOR (H,V)
   :F=1:GOTO 41
45:F=0:IF X=STHEN
   41
50:GLCURSOR (0,VB
   -60):TEXT :
   PRINT "KEY ENT
   ER TO PRINT"
51:LPRINT T;"-PT
   PERSP, FROM";
   LPRINT L;",";M
   ;",";N:END
60:PRINT "NO PLOT
    POSSIBLE":END

 STATUS 1     1131
```

not be located *within* the cube. If the view point is chosen close to the cube, the resulting drawing will appear distorted. This is particularly true when N is large compared to L and M. In extreme cases the vertical range of the printer/plotter may be exceeded. For 1-point perspectives, the most pleasing views are obtained when L is at least twice as large as either M or N.

The plotted result will be scaled by the program to automatically produce a projection that spreads the cube over the full width of the plotting area.

The accompanying diagram shows examples of perspective views of a cube, plotted using the DATA statements listed earlier. The user should input 1 in response to the CUBE SIZE? prompt. Another figure illustrates a second example. In it, 50 should be input for the cube size. (For a rear view of the house, try -75, 65, 50 as L, M and N in a 2-point perspective. For an end view, try 0, 1000, 0.)

In a 1-point perspective projection, the plane of projection is parallel to one vertical face of the cube. In a 2-point perspective, the plane is perpendicular to a horizontal line connecting the center of the cube with the vertical line through the view point. In either case, the resulting projection is what a photograph would show, provided that the camera's film plane is parallel to the plane of projection.

Although entering DATA statements specifying coordinates can be a tedious process, once it has been done a wide variety of views of the object may be drawn. After you have determined the best view point, you wish to delete the DATA statements that involve hidden lines.

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

ALTERNATE CHARACTER SETS

This program, written entirely in machine language, provides for a convenient way of creating characters for display.

When you use this program there is no need to calculate binary codes. You simply use cursor controls and the decimal-point key to create the desired character in the display. Once characters have been defined they may be assigned to shifted keys for later recall.

Generation of Alternate ASCII Codes

In normal use, the keyboard input routine in ROM obtains an ASCII code from tables in ROM. The table for unshifted keys is located in addresses FE80 - FEBF. Shifted keys occupy FEC0 - FEFF.

When bit 2 of address 764E is set (as indicated by the Japanese Katakana annunciator), codes are obtained from alternate Key-to-ASCII tables. These tables are located at nn80 - nnBF and nnC0 through nnFF, where nn is one less than the byte stored in RAM address 785E. Keying SML twice will cancel the alternate keycode mode.

Alternate Display Characters

A ROM table containing codes for generating the display characters assigned to ASCII codes is located in FCA0 - FE7F. Each character is specified using 5 bytes. Normally, codes 80 - FF produce (by default) the same characters as the ASCII codes do. However, if RAM location 785D contains hexadecimal 80, an alternate table is used to obtain display codes for characters represented by codes greater than &80. Since codes above &80 are not ASCII codes, this table forms an extension of, rather than a replacement for, the set of available characters.

This alternate table begins at address nn00, where nn is the byte contained in address 785E.

Whenever the computer is powered up following a power down using the OFF key, the contents of 785D will be set to FF, thereby causing the alternate display mode to be cancelled.

An Alternate Character Set Program

Execution of CALL &7C72, when this program is in memory in a PC-1500A, will set both the alternate key code and alternate display modes.

Shifted alphabet and numeral keys will generate the codes A1 to C4 (rather than the usual ASCII codes for lower-case characters). These codes are displayed as characters that have been defined by the user. The ASC and CHR$ functions also apply to these codes.

After the alternate key code and alternate display modes have been set by CALL &7C72, execution of CALL &7D00 initiates a routine for formation of user-defined characters. The display will blank. Depress the key (alphabet or numeral) to which a character is to be defined. The symbol for that character will appear, followed by a 5 by 7 grid. The position of a "dot cursor" within this grid is indicated by an unlit dot.

The right and left cursor keys may be used to move the "dot cursor" through this grid. Keying the decimal-point key will enter a dot in the position indicated by the dot cursor, into the character being formed. Keying SPACE will clear a dot. The character being formed will be displayed in the position to the right of the dot cursor grid.

When the desired character has been completed, key ENTER to store it into memory. It will be displayed whenever the assigned key (following SHIFT) is depressed.

If BREAK is used to exit the character-creation mode, the character will not be assigned.

Using the Program with the PC-1500

The preceding instructions and listing of the machine language program apply to the Sharp PC-1500A. Since the earlier Sharp PC-1500 and the Radio Shack PC-2 do not contain RAM in the address range 7C00 - 7FFF, a different location is required. The program itself is relocatable without modification. However, its possible locations are restricted by the necessity to locate the Key-to-ASCII table and the Character Table at locations compatible with the PC's operating system routines.

The following locations are suggested for various memory configurations:

1. PC-1500 with CE-155 memory module. First assign the BASIC area with NEW &3B59. Enter the machine language program beginning at &3972, with it ending at &3AA3. (The character table will use the area &3AA5 - 3B58.) To execute the mode-setting routine, use CALL &3972. For the character-setting routine: CALL &3A00.

2. PC-1500 with CE-159 memory module. Begin with NEW &2359. Load the machine language program in the area &2172 - 22A3. The character table will occupy &22A5 - 2358. To execute the mode-setting routine use CALL &2172. For character creation, use CALL &2200.

3. PC-1500 with CE-161 memory module. Begin with NEW &0359. Load the machine language

program into the &0172 -02A3. The character table will occupy &02A5 - 0358. Set the mode using CALL &0172. For character creation use CALL &0200.

4. PC-1500 unexpanded. Use NEW &4359. Load the machine language program into &4172 - 42A3. The character table will occupy &42A5 - 4358. Set the mode using CALL &4172. For the character-setting routine: CALL &4200.

Location of Stored Character Codes

In the PC-1500A version, the codes that produce the characters defined by the user are stored in the area &7DA5 - 7E58. They may be viewed by using PEEK. These are the codes that can be used with

Table *Character Codes.*

ADDR	KEY	ASCII
7DA5	A	A1
7DAA	B	A2
7DAF	C	A3
7DB4	D	A4
7DB9	E	A5
7DBE	F	A6
7DC3	G	A7
7DC8	H	A8
7DCD	I	A9
7DD2	J	AA
7DD7	K	AB
7DDC	L	AC
7DE1	M	AD
7DE6	N	AE
7DEB	O	AF
7DF0	P	B0
7DF5	Q	B1
7DFA	R	B2
7DFF	S	B3
7E04	T	B4
7E09	U	B5
7E0E	V	B6
7E13	W	B7
7E18	X	B8
7E1D	Y	B9
7E22	Z	BA
7E27	0	BB
7E2C	1	BC
7E31	2	BD
7E36	3	BE
7E3B	4	BF
7E40	5	C0
7E45	6	C1
7E4A	7	C2
7E4F	8	C3
7E54	9	C4

Program *Alternate Character Sets.*

```
7C72 FD 58 84 DD  7D0E FD C8 BE ED
7C76 0A 48 80 CA  7D12 57 EC 41 41
7C7A 5D EB 76 4E  7D16 41 4A 04 DF
7C7E 04 9A 0B 4E  7D1A 43 43 43 43
7C82 59 01 48 38  7D1E DF 0E B5 0A
7C86 35 32 09 58  7D22 AE 78 75 2A
7C8A 57 11 53 0F  7D26 58 7A 5A 00
7C8E 2D 2E 30 4D  7D2A 55 BE ED EF
7C92 55 15 4A 37  7D2E CD 8E 88 03
7C96 34 31 0D 28  7D32 BE E2 43 C3
7C9A 49 16 4B 4F  7D36 42 CD 54 4A
7C9E 4C 29 19 43  7D3A 00 5A 06 6A
7CA2 45 12 44 2F  7D3E 04 B7 2E 8B
7CA6 2A 2B 20 56  7D42 25 B7 20 8B
7CAA 52 13 46 50  7D46 2A B7 0D 8B
7CAE 08 3D 02 5A  7D4A 42 B7 0C 8B
7CB2 51 1B 41 18  7D4E 29 B7 08 99
7CB6 1F 0C 0A 42  7D52 21 45 B7 FF
7CBA 54 14 47 39  7D56 9B 05 46 4B
7CBE 36 33 5B AE  7D5A FF 42 93 3F
7CC2 B9 01 A8 C3  7D5E 4A 04 FB D1
7CC6 C0 BD 09 B8  7D62 93 45 B5 BF
7CCA B7 21 B3 0F  7D66 9E 49 45 BD
7CCE 2C 2E B8 AD  7D6A FF 1B 51 88
7CD2 B5 25 AA C2  7D6E 07 8E 05 45
7CD6 BF BC 0D 3C  7D72 19 51 88 05
7CDA A9 26 AB AF  7D76 4A 00 45 B7
7CDE AC 3E 19 A3  7D7A FF 9B 05 46
7CE2 A5 22 A4 3F  7D7E 4B FF 44 4E
7CE6 3A 3B 5E B6  7D82 05 91 66 4A
7CEA B2 23 A6 B0  7D86 00 DB 9D 6B
7CEE 1D 40 02 BA  7D8A D9 9E 6E FD
7CF2 B1 1B A1 1A  7D8E 8A B1 41 83
7CF6 1E 1C 5D A2  7D92 02 B3 2B FD
7CFA B4 24 A7 C4  7D96 58 4A A5 FD
7CFE C1 BE BE EC  7D9A CA D9 D9 FD
7D02 A2 BE E4 2C  7D9E CA 55 41 88
7D06 B7 30 91 07  7DA2 04 9A
7D0A B7 3D 9B 0B
```

GPRINT statements to produce the same characters.

The accompanying table provides the following information: 1) the address of the first of the five bytes that are used to generate the character, 2) the key to which that character is assigned, and 3) the code for the character, acting like an ASCII code.

SAVEing the Program on Tape

The program may be saved on tape using the CSAVE M statement. For the PC-1500A execute: CSAVE M "ALT CHAR SET";&7C72,&7DA3. (If you want to save the characters you have created,

along with the program, then use the command: CSAVE M"ALT CHAR SET",&7C72,&7E58.)

If you are using the PC-1500 or PC-2, the addresses in the CSAVE M statement will need to be modified to coincide with the area of memory used. For example, when using the CE-155 memory module, save the programing using: CSAVE M"ALT CHAR SET",&3972,&3B58.

To reload the program, simply execute: CLOAD M.

A Suggestion

You can save a lot of typing by assigning the two CALL statements to a couple of REServe keys.

Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158, provided this program.

INSIDE THE PC-1500

This is the fifth part of the current series on the Sharp PC-1500. (Most of this information is also applicable to the Radio Shack PC-2.) This valuable information is supplied through the efforts of *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* This issue continues from Issue 34 of *PCN.* Norlin has been describing a number of the ROM routines. Many of the functions they perform are of value to machine language programmers when called as machine language subroutines. Norlin has spent a great deal of effort deriving this information and taking the time to write about it. If you use it, try and find time to let him know how much you appreciate it!

TERMINATE WITH ERROR MESSAGE

The following routines will terminate a machine-language program with an ERROR message displayed in 'Ready' mode:

(a) Call E0 (CD88) displays ERROR n, where n is contained in UH

(b) Call E4 (CD89) displays ERROR 1

When the ERROR message is displayed, the up arrow key may be used to display a BASIC line at the address pointed to by Y. A meaningless display will result unless an effort has been made to load Y to point to the desired location.

Register Y can be made to point to the location in BASIC from which the machine-language program was called by either of the following:

(a) Provided that the stack has been kept balanced, POP Y twice.

(b) Load register S with 784A; then execute POP Y.

PRINTER ROUTINES

MOTOR OFF

Subroutine A769 is required following certain printer subroutines.

EQUIVALENTS OF CERTAIN BASIC STATEMENTS

TEXT	Subroutines ACBB, A9D5, A769
GRAPH	Subroutines A9D5, A769, ABEF; store FF into 79F0
CSIZE n	Store n (1 to 9) into 79F4
COLOR n	Subroutine A519. Precede by loading UL with n (0 to 3). Follow by subroutine A769 (Motor off)
LF n	Subroutine AA04. Precede by storing n (in signed binary) into 7B7F-7B80, and loading XH with 7B, XL with 7F.

Follow by subroutine A769 (Motor off)
LCURSOR n Subroutine AC52. Precede by loading XH with FF, and XL with n. Follow by subroutine A769 (Motor off)
SORGN Subroutine AC97
ROTATE n Store n (0 to 3) into 79F2

PRINTING IN TEXT MODE

(1) To print contents of R0 (or string pointed to by R0), using default format:

Store 04 into 788E; execute subroutine B3EA.

Numeric data is printed right-justified; strings, left-justified. Printer advances to next line if necessary. If DSIZE set is larger than 2, it will be changed to 2. TEXT mode will be set. A carriage return (with line feed) will follow printing.

(2) To print contents of R0 (or string pointed to), formatted:

(a) Store format specifications into memory, as follows:
7BAA: Editing characters. 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific format
7BAB: Number of characters preceding decimal point, including spaces for sign and commas (when specified)
7BAC: Maximum number of characters to be printed (in printing of character strings); if no limitation, zero
7BAD: Number of characters including and following decimal point
(b) Store zero into 79EA and 79F5
(c) Execute subroutine B49A (Print); optionally, subroutine A9F1 (CR with line feed); subroutine A769 (Motor off)

(3) To print contents of R0 (or string pointed to by R0), formatted, starting at current pen location, with line advance when necessary:

(a) Store format specifications into 7BAA-AD, as described above
(b) Store 00 into 79EA, 08 into 79F5
(c) Execute subroutine B49A. Follow by subroutine A769 (Motor off)

(4) To print a single character (no carriage return or line feed):

(a) Store the ASCII code of the character to be printed into 7B7F
(b) Load XH with 7B, XL with 7F. Store zero into 79EA
(c) Execute subroutine A781. Follow by subroutine A769 (Motor off)

(5) To execute carriage return with line feed:

Load XH with 7B, XL with 7F. Execute subroutines A9F1, A769

PLOTTING (GRAPH MODE)

(1) To move pen, using given x and y increments:

(a) Specifying increments in two-byte signed binary, store the x-increment into 7B7E-7B7F, and the y-increment into 7B7C-7B7D

(b) Load XH with 7B, XL with 7C
(c) Execute subroutine AD78. Follow by subroutine A769 (Motor off)

(2) To move Pen to location having coordinates (x,y):

(a) Specifying x and y in two-byte signed binary, store x into 7B7E-7B7F, and y into 7B7C-7B7D
(b) Load XH with 7B, XL with 7C
(c) Execute subroutine AC07. Follow by subroutine A769 (Motor off)

(3) To draw a line, starting from current Pen location, using given increments in x and y:

(a) Specifying increments in two-byte signed binary, store the x-increment into 7B7E-7B7F, and the y-increment into 7B7C-7B7D
(b) Load XH with 7B, XL with 7C. Store FF into 79E9: store line type (0 to 9) into 79EA; store 01 into 7B84
(c) Execute subroutine AD07 (Draw line) and A769 (Motor off)

(4) To draw a square, taking current Pen location as lower left corner:

(a) Load A with length of a side of the square
(b) Load XH with 7B, XL with 7F. Store line type (0-9) into 79EA
(c) Execute subroutine AC2F (Draw square) and A769 (Motor off)

(5) To draw a rectangle, beginning at current Pen location:

(a) Specifying dimensions in two-byte signed binary, store the x-dimension into 7B7E-7B7F, and the y-dimension into 7B7C-7B7D
(b) Load XH with 7B, XL with 7C; store FF into 79E9; store line type (0 to 9) into 79EA
(c) Execute subroutine AD64 (Draw rectangle) and A769 (Motor off)

PRINTING IN GRAPH MODE

(1) To Print contents of R0 (or string Pointed to by R0), formatted, starting from Present Pen location, in direction specified by ROTATE:

(a) Store format specifications into 7BAA-AD, as in TEXT mode
(b) Store zero into 79EA and 79F5
(c) Execute subroutine B433. Follow by subroutine A769 (Motor off)

(2) To Print a single character, in direction specified by ROTATE:

(a) Store the ASCII code of the character to be Printed into 7B7F
(b) Load XH with 7B, XL with 7F. Store zero into 79EA
(c) Execute subroutine A781. Follow by subroutine A769 (Motor off)

PRINTING PROGRAM LINES (TEXT MODE)

(1) To Place a Program line into the InPut Buffer Prior to Printing:

The line to be Printed must be Put into the InPut Buffer, starting with its two-byte line number, omitting the link byte, and continuing with the codes for the BASIC statements in the line. A line may be

Put into the Input Buffer in that form in any of the following ways:

(a) Subroutine D26F may be used to place the first BASIC line into the Input Buffer; the SEARCH pointer is set to point to that line.

FROM THE WATCH POCKET

In a recent column in the magazine InfoWorld, John Gantz lamented the demise of the pocket and notebook computer field. Much of his pessimism apparently evolved from a personal experience wherein he tried to dispose of some 200 Texas Instrument Compact 40 handheld computers. We don't blame him for possibly being a little disappointed with the CC-40, but we do not think that the pocket and portable computer field is about to fade away.

Indeed, the signs are evident that all is well in the PC business despite recent withdrawals from the market by some suppliers. It does seem to currently appear that we are in a consolidation phase. Suppliers who can't match consumer's tastes are dropping out. Products are improving. New genre are being identified. The recently announced Sharp PC-1260 and 1261 models are good examples of this forward progress. The 1261, with its 10K of user RAM and 40K of built-in ROM software, points to the future. Here is a pocket computer that fits easily in a shirt pocket, yet contains enough power to really ease one's daily task. 10K of user memory is more than enough to hold several practical programs for use on a daily basis. The automatic calculation capability of the 1260 and 1261 are examples of the types of features that will be provided as standard in pocket computers of the future.

Pocket computers, now entering their fifth year of existence, are still in the pioneer stages. Manufacturers are beginning to learn just what it is that users want from their PC's. Models recently or about to be announced will provide more and more of what users seek. For example, the recently announced Casio FX-750P model and the upcoming Sharp PC-1350 allow for wafer-thin memory modules, complete with battery backup, to be easily plugged into the pocket computer. These new modules will make it easier for users to access a large variety of programs. They represent a significant step in the development of the PC towards becoming a practical tool for users from all walks of life.

Opportunities abound for visionaries and entrepreneurs who understand the vistas that are opening. While the society as a whole is still struggling with the legal concepts that need to be in place to protect both the consumer and producer in areas such as software copyright and protection, new technologies will make it practical to distribute information processing tools. For instance, the Sharp PC-1260 or 1261 can be password protected. This means that those with specialized knowledge or techniques can program this information into PC's and sell the entire package as a ready-to-use application tool. This can be accomplished with relatively low risk of having the source code and methodology comprised unless authorized by the program author. In the case of newer models, such as the PC-1350, application programs can be distributed installed directly in the battery backed RAM modules. Once again, various levels of source code protection can be provided at the author's discretion.

It should be noted that each generation of pocket computer products provides benefits in several areas. Not only is the hardware itself more powerful, but deficiencies in operating modes and capabilities are being corrected as a result of feedback from earlier users. And these early users have been speaking out. Hence, multiple line displays, which provide for the easier comprehension of data, will become the norm. Feedback from users indicates they would rather have several lines of information, even though the characters may be smaller, than having a single line that forces prompts to be remembered while data is being entered or leaves the user guessing about the interpretation of an output value.

In case you haven't heard, Sharp now produces a low cost combination thermal printer and cassette interface called the Model CE-126P. The list price of this unit is $94.95. The machine contains a 24 digit thermal printer as well as a standard audio-cassette interface. Originally designed for the EL-5500 and EL-5510 calculator units, this

machine will also work with the Sharp PC-1250/51 and PC-1260/61 models. Note that this unit is designed to operate with your own audio cassette unit and does not contain a micro-cassette recorder as does the more expensive CE-125. If you already have your own tape unit, the CE-126P appears to be good buy.

The new Sharp PC-1350 is due out in the middle of October. It is said to sport a four-line display, contain approximately five kilobytes of memory, and will retail for $195. This is the unit mentioned earlier in which you can install 8 or 16 kilobyte battery-backed RAM modules. It is reported to be approximately the size of an EL-5500. It looks for sure as though Sharp intends to remain a serious competitor in the pocket computer field.

As most of you know, Radio-Shack recently discontinued its PC-2 model. There has been no indication whether or not the giant chain intends to continue to support the pocket computer market. There closeout price on the PC-2 was $99.95. At about $80 cheaper than the Sharp PC-1500A, most stores quickly sold out. That was a good deal for those who were able to get in on it.

Radio-Shack apparently intends to remain active in the notebook computer field. In particular, it has recently announced a new software package for the Model 100 computer. Called Business Finance, the package is said to be a tool for both business and personal financial analysis. The package is reported to consist of 12 programs, including those used to calculate net present value, internal rate of return, financial management rate of return, and depreciation. The package also contains programs that assist one in making rent or buy decisions, perform lease/purchase analysis, machinery replacement analysis and bond analysis. Several other programs enable one to perform annuity computations, as well as calculate compound interest, leases, loans, and perform pension and portfolio analysis. The list price of the Business Finance Program is $39.95. It runs on a Model 100, equipped with at least 16 kilobytes of user memory.

A few months ago *PCM* ran a brief blurb about an outfit producing an A/D board for the PC-1500. The company, ENERTECH, *558 Highland Avenue, Upper Montclair, NJ 07043,* has announced that their new A/D 1500 I/O Interface is now in stock and available for immediate delivery. The price is $299.00 plus shipping charges. The interface allows one to use programs in either BASIC or machine language to allow collection of data in the form of a bit count between 0-255 (to represent analog information) or 0-1 to represent digital information. Information collected from the interface can be displayed, printed, or saved to a cassette tape. The interface can also be used to control relays, solenoids, light-emitting diodes, etc., at currents of up to 100 milliamperes (and potentials of as much as 12 volts). The A/D interface is powered by a small 12 volt battery. This battery can also supply a CE-150 printer/cassette interface. Write directly to the company or phone (201) 744-4962 for additional information.

If you have a Casio Model PB-700, you may be interested in checking out a new software package released by KASTER Corporation, *P.O. Box 117, Alpine, NJ 07620,* telephone (201) 784-9430. The package is called DeskMate. It is said to combine a database manager, spreadsheet, and graphics package into one integrated program. Up to 70 personal records or a 200 cell spreadsheet can be retained in memory. The graphics program can plot four different graphs from data generated by the spreadsheet. KASTER Corporation also states that it has other packages for the PB-700 available.

Hewlett-Packard Corporation has announced the availability of dBASE-II for The Portable (HP-110) computer. This is a powerful information management software package that has been available for several years on desktop computers. The package is supplied on diskette. It retails for $500. Contact your nearest Hewlett-Packard sales representative for additional information. Incidentally, the HP-110 continues to be a hot seller for Hewlett-Packard Corporation.

If you are into using personal computers at work, then you may be interested in a bi-monthly newletter that has just been announced. The newsletter claims to review a major best-selling software package from a general business category in each issue. The objective of the newsletter is said to be helping users to understand how to select from an incredible array of constantly updated software. Each report includes a summary, illustrations, real-world usage examples, and a "clip-and-save" tear sheet index of commands, as well as the reviewer's personal evaluation of the program's capabilities. For subscription information, contact Delicious Publishing, 314 East Holly #106, Bellingham, WA 98225.

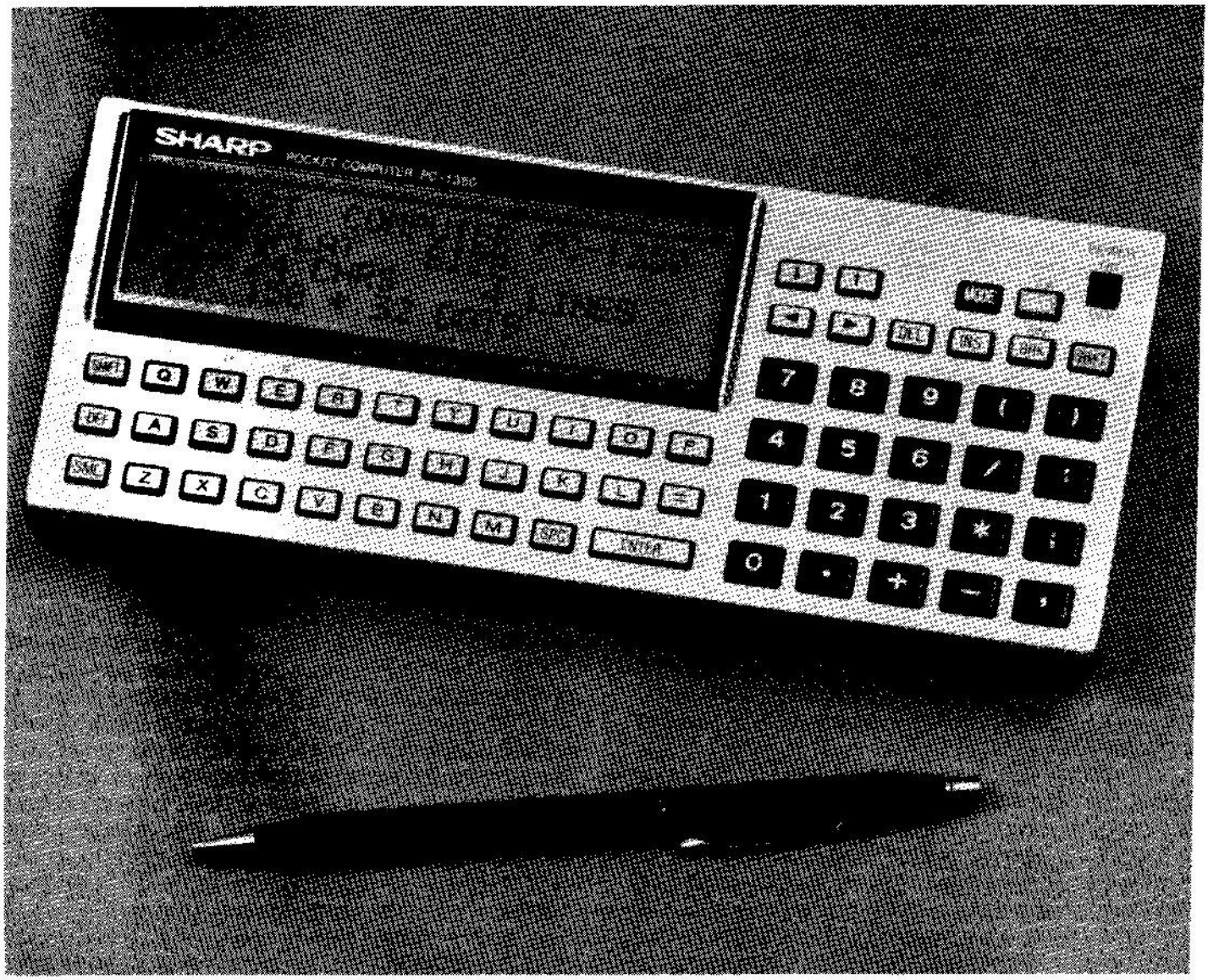
POCKET COMPUTER NEWSLETTER

P.O. Box 252, Seymour, CT 06483

POCKET COMPUTER NEWSLETTER

Issue 36 - November/December

Photo *Sharp PC-1350 Pocket Computer Sports Multiple-Line Liquid-Crystal Display.*



Another *Personal Information* ™ product.

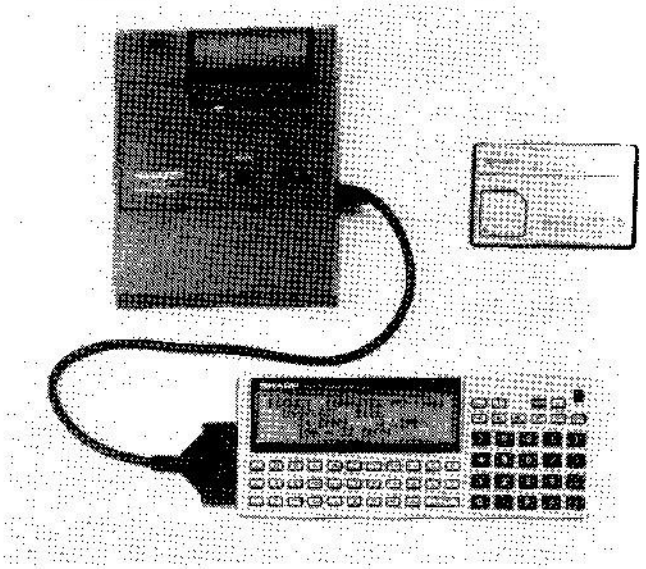
SHARP PRODUCING NEW MODEL PC-1350

Sharp Electronics Corporation is now delivering a new, slim pocket computer dubbed the model PC-1350. It is believed that the model may eventually replace the popular PC-1500 produced by the company. The unit weighs less than seven ounces. It is just slightly over 1/2 inch in thickness, with a length of approximately 7-1/4 inches and a width under 3 inches. The PC-1350 sports a large 4 line by 24 characters display. The U.S. list price is said to be $195.00 at the present time.

The unit's keyboard contains 62 keys. Thirty-two of these are arranged in QWERTY fashion. Twenty are grouped in a numeric/operand keypad. Ten more serve as cursor, scrolling, mode and editing aids. There is a separate power on/off switch. A contrast control, accessed on the right-hand side of the unit, provides for adjustment of the liquid-crystal display intensity.

The new pocket machine has a power-packed software package residing in 40 kilobytes of ROM. This includes a BASIC interpreter and user operating system. The PC also contains 5 kilobytes of user RAM provided as standard. This can be expanded by an additional 8 or 16 kilobytes. The RAM expansion units are somewhat unusual. They are thin, card-like modules containing battery-backed CMOS memory. Each card can hold programs and/or data for up to several years. There is only one major drawback to using these cards as mass storage devices: their relative cost. The 8K RAM card is said to retail at approximately $100 while the 16K version will be about $170.00. That is equivalent to what a RAM module such as the CE-161 costs on the PC-1500. Such a price will tend to restrict the use of the cards as general purpose storage devices.

Photo *PC-1350 With CE-126P Printer/Cassette Unit and CE-202M (16K) Memory Card.*



The PC-1350 is powered by a pair of CR-2032 lithium cells. It is equipped with an interface enabling it to be coupled to a CE-126P printer/cassette interface. A second serial interface provides a means of interchanging information with Sharp's PC-5000 Portable Computer.

Will the new PC-1350 become the volume pocket computer leader? Will Radio Shack soon begin promoting a PC-5? Only time will tell.

To obtain more information on the PC-1350, contact **Sharp Electronics Corporation,** *10 Sharp Plaza, Paramus, NJ 07652.* The telephone number is (201) 265-5600.

REVIEW OF THE CASIO PB-700

The Casio PB-700 is the same size as the Radio Shack PC-2 Pocket Computer. It has a staggered QWERTY keyboard. The keyboard includes a key marked "CAPS". This key may be used to provide lower case characters.

The most striking feature of the PB-700 upon initial examination is its display. It is the largest of any current pocket computer. It measures 4 inches in length by almost 1 inch high. It is capable of displaying 4 lines with 20 characters per line. You can position the display cursor using the LOCATE X,Y statement. When used in the graphics mode, the display provides 160 by 32 dots (compared to the PC-2's 160 by 7 dots). It is easy to draw pictures on the display. Just use the statement DRAW X,Y. To erase, use DRAWC X,Y. The POINT statement can be used to have a program automatically determine the state of a pixel on the screen.

There are no annunciators on the LCD screen. To ascertain such things as which angle mode is in use, the amount of memory remaining, or which of the 10 possible program areas have been used, you utilize a SYSTEM command. You execute the program in a particular area by holding down the SHIFT key and pressing the digit (0 - 9) that corresponds to the program storage area.

The PB-700 does not have a mode switch. It does, however, have a connector that may be used to attach a printer or cassette interface, both of

which are available as options. It also has a memory bay on the back that can hold up to 3 RAM modules. Each RAM module can contain 4K of memory. The RAM modules currently retail for $39.95 each. Since the basic PB-700 includes 4K of user memory, a fully-loaded unit can have up to 16 kilobytes of RAM. However, about 2.8K of RAM is consumed as overhead.

The unit is powered by 4 type AA batteries. They are accessed via a door in the back of the machine.

The BASIC instruction set of the PB-700 includes the following directives: POINT, NEW, LIST, CONT, RUN, BEEP, CHAIN, CLEAR, CLS, DATA, DIM, END, GOSUB, RETURN, GOTO, FOR, TO, STEP, NEXT, INPUT, LET, PRINT, LPRINT, READ, REM, RESTORE, STOP, TRON and TROFF. It provides a number of string functions as well as trigonometric and inverse-trig functions. It has EXP, SQR, LOG (natural) and LGT (base 10 log), ABS, INT, SGN, RND and ROUND functions as well as INKEY$, USING and TAB. The ROUND(X,Y) function will round X to the decimal position indicated by Y. Thus, for instance, if Y is 2, the value X will be rounded to the nearest hundredths place!

When compared with the Radio Shack PC-2, the Casio PB-700 lacks the following statements and commands: TIME, PAUSE, CALL, PEEK, POKE, ARUN, AREAD, BEEP ON, ON ERROR GOTO, NOT, STATUS, WAIT, LOCK and UNLOCK.

However, it does have features that are not found on the PC-2, such as:

1. The ability to use non-integer values in FOR, NEXT and STEP statements.

2. More flexibility within IF-THEN-ELSE statements.

3. The use of half-precision array variables (5-digit mantissa with a 2- digit exponent) instead of full-precision variables (12-digit mantissa and 2- digit exponent). This can save a lot of memory space when manipulating arrays.

4. A key that makes it easy to delete a lot of characters. Once the cursor has been positioned, one press of this key will delete everything to the right on a particular line.

The PB-700 permits the use of 1- and 2-dimensional arrays as well as 2- character array variable names.

The unit is supplied with a carrying case, a manual, a quick reference booklet, 4 AA batteries and a RAM backup battery. The manual contains demonstration programs including one that provides impressive graphics.

The PB-700 has a U.S. list price of $199.95. It is available through Casio distributors.

This review provided by *Jeffrey Woodhead, P.O. Box 770, Davis, CA 95617.*

FOR PC-1250/51/60/61 USERS

ALARM CLOCK PROGRAM

This program converts a Sharp PC into a "smart" alarm clock. Several versions of the program are provided: one for PC-1250/1251s (or the equivalent Radio Shack PC-3) and another for PC-1260/1261 owners. Furthermore, a slight modification to the version for the 1250 will enable it to be used on the original Sharp PC-1211 (Radio Shack PC-1)! The program can be "calibrated" so that it is accurate to within a few seconds per day. It is "smart" in that it allows the SET and ALARM time to occur at any time within a 24 hour period. Thus, for example, you can set it at 8:00AM for an alarm time of 7:00AM and it will "know" that you mean the next morning. It won't bother you at 7:00PM as a 12-hour clock would!

The input times are entered in the format (HH.MMSS) followed by AM or PM (when using the PC-1250/1251 version). When using the 1260/1261 version, you should follow the time by the notation *A*M or *P*M. (The asterisk meaning multiplication. The reason that the multiplication symbol must be used in the 1260/1261 version is because it does not recognize "implied" multiplication. The symbols A, P, and M are actually used in the program as variable names. They assume values of plus or minus one. These types of differences amongst supposedly compatible PCs stems from the fact that the manufacturer assumes that you follow rigid programming conventions. The use of "implied" multiplication as done in the 1250/1251 version is not sanctioned by Sharp.)

In any event, if you do not specify AM (*A*M) or PM (*P*M) when inputting a time value, the program will assume you mean AM. Also, the program assumes that noon is 12.00AM and midnite is 12.00PM. The program is sufficiently accurate that you may wish to

Program *Alarm Clock for PC-1250/51 (Or PC-3).*

```
10:BEEP 1: PAUSE " 24 H
   R ALARM CLOCK"
12:REM "NOON = 12.00AM"
14:REM "MIDNITE = 12.00
   PM"
20:Y=3.225597096E-4:A=1
   :P=-1:M=1: USING "##
   #.####"
30:INPUT "ALARM TIME(HH
   .MM):";U
40:INPUT "TIME NOW(HH.M
   M):";S:Z= DEG ABS S
50:IF U>0 IF ABS U>12
   LET U=U-12
55:IF S>0 IF ABS S>12
   LET S=S-12
60:IF U<0 LET U= ABS U+
   12: IF U>24 LET U=U-
   12
65:IF S<0 LET S= ABS S+
   12: IF S>24 LET S=S-
   12
70:V= DEG U- DEG S: IF
   V<0 LET V=V+24
80:V=V+Z:Z=Z+0.7250Y:X=
   Z
90:IF X>=V THEN 130
100:Z=Z+Y: IF Z>=13 LET
   Z=Z-12
110:X=X+Y:T= DMS Z
120:PAUSE "TIME:";T:
   GOTO 90
130:USING "###.##"
140:BEEP 3: PAUSE "TIME:
   ";T
150:Z=Z+1.55782Y: IF Z>=
   13 LET Z=Z-12
160:T= DMS Z: GOTO 140
200:"C": PAUSE "CALIBRAT
   ION -->": USING
210:INPUT "TRUE CLOCK(H.
   MMSS):";W
220:IF W>0 IF ABS W>12
   LET W=W-12
230:IF W<0 LET W= ABS W+
   12: IF W>24 LET W=W-
   12
240:J= DEG U- DEG S: IF
   J<0 LET J=J+24
250:K= DEG W- DEG S: IF
   K<0 LET K=K+24
260:L=KY/J: PRINT "NEW Y
    = ";L: END
```

Program *Alarm Clock for PC-1260/1261.*

```
10:BEEP 1: PAUSE " 24 H
   R ALARM CLOCK"
12:REM "NOON = 12.00AM"
14:REM "MIDNITE = 12.00
   PM"
20:Y=3.251197073E-4:A=1
   :P=-1:M=1: USING "##
   #.####"
30:INPUT "ALARM TIME(HH
   .MM):";U
40:INPUT "TIME NOW(HH.M
   M):";S:Z= DEG ABS S
50:IF U>0 IF ABS U>12
   LET U=U-12
55:IF S>0 IF ABS S>12
   LET S=S-12
60:IF U<0 LET U= ABS U+
   12: IF U>24 LET U=U-
   12
65:IF S<0 LET S= ABS S+
   12: IF S>24 LET S=S-
   12
70:V= DEG U- DEG S: IF
   V<0 LET V=V+24
80:V=V+Z:Z=Z+0.7250*Y:X
   =Z
90:IF X>=V THEN 130
100:Z=Z+Y: IF Z>=13 LET
   Z=Z-12
110:X=X+Y:T= DMS Z
120:PAUSE "TIME:";T:
   GOTO 90
130:USING "###.##"
140:BEEP 3: PAUSE "TIME:
   ";T
150:Z=Z+1.55782*Y: IF Z>
   =13 LET Z=Z-12
160:T= DMS Z: GOTO 140
200:"C": PAUSE "CALIBRAT
   ION -->": USING
210:INPUT "TRUE CLOCK(H.
   MMSS):";W
220:IF W>0 IF ABS W>12
   LET W=W-12
230:IF W<0 LET W= ABS W+
   12: IF W>24 LET W=W-
   12
240:J= DEG U- DEG S: IF
   J<0 LET J=J+24
250:K= DEG W- DEG S: IF
   K<0 LET K=K+24
260:L=K*Y/J: PRINT "NEW
   Y = ";L: END
```

enter times to the nearest second. This is accomplished by appending the seconds value after minutes, i.e., entering a time as 12.5315PM (12.5315*P*M on a 1260/1261).

Suppose you want to have the PC tell you when it is 7:15 in the morning? Assume that the current time is 10:35PM. You would run the program as follows:

```
>RUN
24 HOUR ALARM CLOCK
ALARM TIME(HH.MM):  7.15*A*M
TIME  NOW (HH.MM)  10.35*P*M
.... at 7.15AM the next morning ....
BEEP  -  BEEP  -  BEEP
TIME:  7.15
```

Suppose you found the alarm clock program to be slightly off as far as timing was concerned? You can calibrate it against a chronometer (er, perhaps your trusty wrist watch will suffice) having a second hand, if you desire. Let's assume that you ran the program as in the example above, but the alarm went off at an actual time (according to your chronometer) of exactly 7:17:30 in the AM. To enhance the accuracy of the program for your specific PC, you do the following:

```
>RUN 200   or  DEF/C
CALIBRATION -->
TRUE CLOCK(H.MMSS): 7.1730*A*M
NEW Y = WHATEVER IT SAYS
```

You complete the adjustment by placing the PC into the PROGRAM mode and changing the calibration constant Y in line 20 to the NEW Y value given!

To use the program on an original PC-1211 (Radio Shack PC-1) you can use the program listing given for the PC-1250/1251 but start with a value for Y in line 20 of Y=5.561374309E-04.

It is a pretty nifty little program. Thanks for it go to: *Emerich Auersbacher, 41 King Street #2, Belleville, NJ 07109.*

FOR HEWLETT-PACKARD HP-71B USERS

WEATHER PROGRAM FOR HP-71B

Ever since the introductory announcement in Issue 32 of *PCN* and the subsequent product review article in Issue 33, we have been getting more and more requests for additional information and programs specifically for the HP-71B. OK, OK, we get the message!

It sure looks like HP has cooked up a winner with the 71B. We haven't had as many requests to cover a particular model since Sharp Electronics Company introduced the PC-1500.

So, to kick things off in what we plan to be continuing coverage of this model, we offer up a HP-71B-tailored weather forecasting program. Long-time *PCN* readers may recognize this as a customized version of *E. Auersbacher's* program originally published in Issue 20 for the Radio Shack PC-2 and Sharp PC-1500. (By the way, line 500 of the original version contained a typo - the IF Z=3 statement should have been IF X=3!) Operation of the program is straightforward. So, just load and RUN. We will have more for the 71B next issue!

Program *Weather Forecasting on the HP-71B.* *(Listing continued on the next page.)*

```
10 DELAY 1,.5 @ WIDTH 96 @ DISP "WEATHER FORECASTER @ DESTROY ALL
20 A$="S. SW W. NW N. NE E. SE "
30 INPUT "BAROMETER (IN): ";P @ IF P>=30.2 THEN S=0
40 IF P<30.2 THEN S=1
50 IF P<30.1 THEN S=2
60 IF P<30 THEN S=3
70 IF P<=29.8 THEN S=4
80 INPUT "RISE, FALL, STEADY? ";T$ @ X=0
90 IF FNA$(T$)#"R" THEN 130
```

Program *Weather Forecasting on the HP-71B. (Listing starts on previous page.)*

```
100 INPUT "RISING FAST, SLOW? ";R$ @ IF FNA$(R$)="F" THEN X=1 @ GOTO 190
110 IF FNA$(R$)#"S" THEN BEEP @ GOTO 100
120 X=2 @ GOTO 190
130 IF FNA$(T$)="S" THEN 190
140 IF FNA$(T$)#"F" THEN BEEP @ GOTO 80
150 INPUT "FALLING FAST, SLOW? ";R$
160 IF FNA$(R$)="F" THEN X=3 @ GOTO 190
170 IF FNA$(R$)#"S" THEN BEEP @ GOTO 140
180 X=4
190 INPUT "WIND FROM THE: ";W$ @ IF LEN(W$)#2 THEN BEEP @ GOTO 190
200 W=POS(A$,UPRC$(W$)) @ IF W=0 THEN BEEP @ GOTO 190
210 W=INT(W/3)+1
300 Y=ABS(W-3)<2
310 IF Y*(S<3) AND X=0 THEN Z=1 @ GOTO 600
320 IF Y*(S=1) AND X=1 THEN Z=6 @ GOTO 600
330 IF Y=(S=0) AND X=4 THEN Z=3 @ GOTO 600
340 Y=W=1 OR W=8
350 IF Y*(S=1) AND X=4 THEN Z=4 @ GOTO 600
360 IF Y*(S=1) AND X=3 THEN Z=5 @ GOTO 600
370 Y=ABS(W-7)<2
380 IF Y*(S=1) AND X=4 THEN Z=4 @ GOTO 600
390 IF Y*(S=1) AND X=3 THEN Z=5 @ GOTO 600
400 IF Y*(S>2) AND X=4 THEN Z=6 @ GOTO 600
410 IF Y*(S>2) AND X=3 THEN Z=7 @ GOTO 600
420 Y=W=6 OR W=7
430 IF Y*(S<2) AND X=4 THEN Z=8 @ GOTO 600
440 IF Y*(S<2) AND X=3 THEN 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 @ 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 DISP "FORECAST -->" @ K$=""
610 ON Z GOSUB 710,720,730,740,750,760,770,780,790,800,810,820,830
620 K$=KEY$ @ IF K$="" THEN 610 ELSE 30
710 DISP "FAIR, LITTLE CHANGE." @ RETURN
720 DISP "FAIR, LITTLE COOLER." @ RETURN
730 DISP "FAIR, LITTLE WARMER." @ RETURN
740 DISP "RAIN WITHIN 24 HRS." @ RETURN
750 DISP "WINDY, RAIN BY 12 HRS." @ RETURN
760 DISP "CLOUDING WITH RAIN." @ RETURN
770 DISP "RAIN AND HIGH WINDS." @ RETURN
780 DISP "UNSETTLED, MAYBE RAIN." @ RETURN
790 DISP "RAIN/SNOW, WINDY." @ RETURN
800 DISP "CLEARING TO FAIR." @ RETURN
810 DISP "SEVERE STORM IMMINENT." @ RETURN
820 DISP "HEAVY RAIN/SNOW, WIND." @ RETURN
830 DISP "CLEARING AND COOLER." @ RETURN
900 DEF FNA$(R$)=UPRC$(R$[1,1])
```

FOR PC-1500 & PC-2 USERS

MACHINE LANGUAGE PROGRAMMING

Readers of the series *Machine Language Programming the Sharp PC-1500 and Radio Shack PC-2* (produced in 1983 as a separate publication by *PCW*) will especially appreciate the following article. As such readers know, the series was abruptly terminated due to serious illness on the part of the author.

What follows is a sample of how the author had planned to continue the series by building upon the instruction set foundation that had been laid in the early installments. Enjoy!

MACHINE LANGUAGE PROGRAMMING
THE SHARP PC-1500
AND RADIO SHACK PC-2
POCKET COMPUTERS

Once a truly interested ML fan acquires an understanding of the types of instructions that are available on a machine, he or she soon develops an itch to do some real ML programming. Let's satisfy that urge right now.

Clearing Memory

When batteries are installed in a PC, the individual bits in memory will "come up" in random states. Some set to the logic 1 state, some cleared to the 0 state. Sometimes it is not desirable to have areas of RAM in such a chaotic condition. One way for a ML programmer to set an area in memory to a known conditon is to fill it with zero bytes. That is, load bytes set to the value zero into a specific range of memory addresses. Sounds like a pretty simple procedure, right? It is! And it can be done in a whole lot of different ways.

Suppose, for example, that you wanted to clear out the section of memory normally used to store the fixed string variables A$ through D$. These are stored in memory addresses &78C0 - &78FF.

One way to accomplish this job would be to load CPU register A with the value zero. Next, the starting address of the area to be cleared might be set up in CPU register X. Once this had been done, STAI (X) directives could be used to stuff the contents of the accumulator (register A) into successive locations in memory as pointed to by the contents of CPU register X. Remember, the STAI (X) instruction automatically advances the value in X each time it is executed. There is just one more parameter to consider. How many times must the STAI (X) directive be repeated in order to clear out the desired block of memory?

Since the size of the memory block is known in this example, a counter could be established, say in register UL. Each time the STAI (X) command was performed, the count in UL could be decremented. When this count reached zero, it would be time to stop stuffing zeros into memory. Thus, one could use the sequence of directives (after the STAI (X) instruction) consisting of: DEUL and RBNZ #nn. That is, decrease the count in register UL and if it is still non-zero, then loop back to repeat the STAI (X) directive. If this method was used, in this example, then register UL would initially have to be set to the decimal count of 64 (hexadecimal 40), representing the number of bytes in memory (from &78C0 through &78FF) that were to be cleared. See the accompanying listing for a detailed example of this method.

If you were wide awake when you read the previous section of this series, then you might remember that this type of situation is ideal for the use of the BNZD #nn instruction. Since, however, the BNZD directive tests for zero *before* it decrements the contents in UL, then the count initially placed in UL must be one less than the number of loops (locations to be cleared). Thus, in this specific example, if BNZD was used, register UL would need to start out with a count of decimal 63 (which is hexadecimal &3F). An accompanying listing illustrates this method, too.

Yet another way of determining when to stop looping would be to check for an appropriate address value in CPU register X. In this case, when the value in X exceeded &78FF (i.e., reached &7900), then it would be time to discontinue the clearing operation. There are several ways this type of procedure might be implemented.

One way would be to set the ending value (say &78FF in this case) into another CPU register. CPU register Y would be available for such use in this example. A comparison could then be made between the contents of X and Y each time that X was advanced by the STAI directive. When the value in X exceeded that in Y, then it would be appropriate to stop the clearing operation. See the

Program *Routine for Clearing Memory Using RBNZ Instruction*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	B5	00				CLRMM1	LDA #00	Load register A with zero.
00	02	48	78					LDXH #78	Set up register X to point to 16-bit address 678C0.
00	04	4A	C0					LDXL #C0	Set up register X to point to 16-bit address 678C0.
00	06	6A	40					LDUL #40	Set up counter (64 decimal here) in register UL.
00	08	41					CLRM1	STAI (X)	Stuff contents of A into memory, then increment pointer.
00	09	62						DEUL	Decrement the counter value in UL (UL=UL-1).
00	0A	99	04					RBNZ CLRM1	If counter is not zero, jump back to stuff another byte.

Program *Routine for Clearing Memory Using BNZD Instruction*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	B5	00				CLRMM2	LDA #00	Load register A with zero.
00	02	48	78					LDXH #78	Set up register X to point to 16-bit address 678C0.
00	04	4A	C0					LDXL #C0	Set up register X to point to 16-bit address 678C0.
00	06	6A	3F					LDUL #3F	Set up counter (63 decimal here) to count-1 (64-1=63).
00	08	41					CLRM2	STAI (X)	Stuff zero byte into memory, advance memory pointer in X.
00	09	88	03					BNZD CLRM2	Check UL for zero, loop back if non-zero, (UL=UL-1).

Program *Routine for Clearing Memory Using Compare Operation to Terminate Loop*

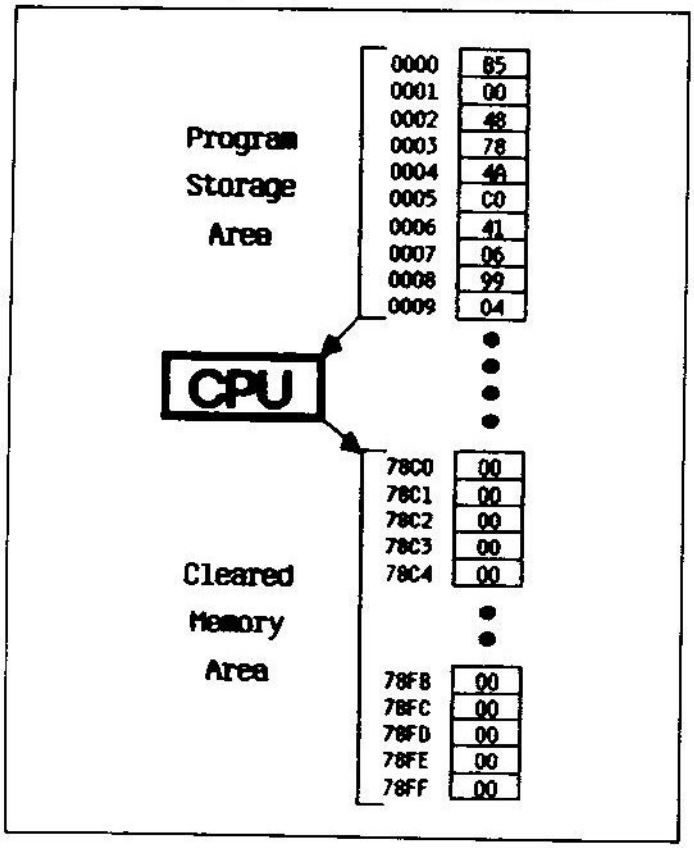
PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	B5	00				CLRMM3	LDA #00	Load register A with zero.
00	02	48	78					LDXH #78	Set up register X to point to 16-bit address 678C0.
00	04	4A	C0					LDXL #C0	Set up register X to point to 16-bit address 678C0.
00	06	41					CLRM3	STAI (X)	Stuff zero byte into memory, advance memory pointer in X.
00	07	06						CPA XL	Check XL for zero (indicating address 67900 reached).
00	08	99	04					RBNZ CLRM3	Loop back to stuff next location if XL is not zero.

example program listing.

An even easier way, in this particular case, would merely be to test for the value in XL going to zero. That is because, when XL exceeds FF (because X reaches 78FF), it will go to the value 00 as X advances to the value 7900. That just happens to be the point at which we want to stop clearing memory in this particular example! A listing of this method is also provided for examination.

By now you should be convinced that ML programming is not an exact science. There are usually countless ways to approach a particular objective. Sometimes you can customize your approach depending on specific parameters. For instance, if you are trying to conserve your use of memory, you might try to devise a sequence of directives that uses a minimum amount of space. Or, you might be interested in maximizing speed of program execution. In that case, you could work towards selecting directives that could be performed in the minimum amount of time. (Do *not* make the assumption that the fastest program is the one with the fewest instructions! Various classes of instructions require different times to complete their execution. It may require a very detailed study in order to find a sequence of instructions that accomplishes an objective in a minimum amount of time!) In most practical applications, however, the actual sequence of instructions selected is not at all critical. Select or create a method you like and try it out!

Diagram *Memory Map of Clearing Operation.*



A Practical Memory Clearing Application

Have you ever developed a BASIC program that used a temporary variable array? That is, an array that you needed to continuously clear out in between operations with other arrays? If so, you know that you had to create a special program loop that would initialize all the elements in that specific temporary array. You could not use a BASIC command such as CLEAR as that would wipe out *all* of the other variables and array elements used in the program, something that we assume for this example, would not be desired.

Of course, there is nothing wrong with creating a BASIC program loop to clear out the elements in an array. It is just that, if the number of elements in the array is large, the process can take some time. If the clearing operation has to be done frequently, the amount of time devoted to this one aspect can become quite exasperating.

However, as a ML programmer, it is possible to devise a scheme to clear out the elements of an array in the proverbial "blink of an eye." Let us see how this could be done.

First, it is necessary to know a few facts about the operation of the BASIC interpreter provided in the PC-1500 (and Radio Shack PC-2). As you may be aware, the interpreter program stored on ROM organizes RAM memory in a specific fashion to serve its purposes. Thus, the lower range of available user memory is assigned for use by the PC's "softkeys" as the REServe memory area. Immediately above this area (in terms of memory addresses) is where user program statements for a BASIC program (or multiple programs) are stored. Finally, the "top" or highest address value locations in RAM are used for the storage of user-defined arrays and variables.

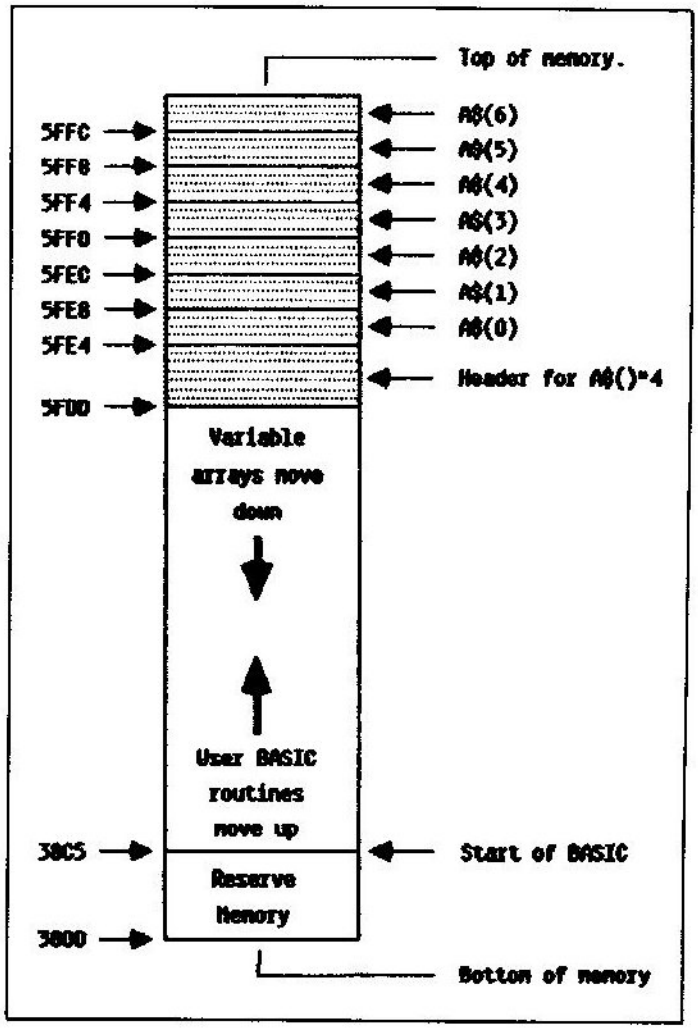
When the pocket computer is first turned on, the ROM program determines the bottom and top addresses of user RAM. The "page" (high order 8 bits) value of these locations are stored in the system RAM at addresses &7863 and &7864 respectively. This procedure is necessary as the range of user RAM can vary depending on which RAM expansion module (such as the CE-151, CE-155 or CE-161), if any, is installed in the PC. Thus, for example, if an 8K module was installed, location &7863 would contain &38 (which is the page portion of the address &3800 where RAM memory would begin). Location &7864 would contain &60. This is the page portion of the address &6000 which is one more than the top user RAM address (&5FFF) that is available in such a system.

(Knowing this information makes it possible to design a procedure that will automatically take account of the amount of memory in a user's system when it is used.)

When a BASIC programmer wants to create a variable array, it is necessary to issue a DIMension statement. This statement essentially blocks out space in user RAM for storage of the array elements. For purposes of illustration let us assume that the DIMension statement is the first statement contained in the user's BASIC program. Let us further assume that it is expressed as follows: DIM A$(6)*4. This means that a string array named A$() is being specified, that there are a total of 7 array elements (numbered 0 - 6) being reserved and that each element is to have room for 4 characters.

When the BASIC interpreter encounters this DIMension statement it will do the following: determine the top of memory (by examining system RAM location &7864), reserve 28 bytes of memory immediately below this location for storage of the array elements (7 elements with 4 characters reserved for each one), immediately below this it will record the array "header" information, (using 7 bytes of storage for this purpose). A summary of this process is shown in the accompanying diagram.

Diagram *Memory Map for Clearing an Array.*



The reason why it is necessary to assume that the DIMension statement is the first statement in the program in this description, is because any other arrays (or user-created variables) will be stored "beneath" this initial array. Thus, locating other arrays can become complicated and such complications are not needed at this point in the ML programming educational process. Right?

It will be worth knowing, so that you might customize such a routine to your own specific purposes, the following additional information about array elements: (1) The number of characters reserved for each element of a string array is precisely the number specified after the asterisk in the DIMension statement. If an asterisk is not used to specify such a value (which is limited to the range 1 to 80), then a value of 16 is assumed by the BASIC interpreter. (2) The number of characters reserved for each element of a numeric array is *always* eight! This is because all numeric values stored in such array elements are assumed to be in floating-point BCD format.

An Array-Clearing ML Routine

If we assume an array size of seven elements (numbered 0 through 6), with an element length of 4 characters, then we can calculate that the space needed by the array elements is exactly 28 bytes. (Note that this excludes the 7 bytes needed by the array "header." This is just as well, however, as we do not want to erase the contents of the array header!)

By examining the contents of memory location &7864, we can locate the start of memory. Actually, this location yields the first page address that lacks RAM, so user memory really begins on the highest possible address (&FF) on the next

Program *Routine for Clearing Memory that Terminates When Address Value Reached.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	48	78				CLRM4	LDXH #78	Set up register X to point to 16-bit address 678C0.
00	02	4A	C0					LDXL #C0	Set up register X to point to 16-bit address 678C0.
00	04	58	79					LDYH #79	Set up register Y to point to 16-bit address 67900.
00	06	5A	00					LDYL #00	Set up register Y to point to 16-bit address 67900.
00	08	85	00				CLRM4	LDA #00	Load register A with zero.
00	0A	41						STAI (X)	Stuff zero into memory, advance memory pointer.
00	0B	94						LDA YH	Put contents of YH into the accumulator.
00	0C	86						CPA XH	Compare (YH-XH) to see if pointer page values are same.
00	0D	99	07					RBNZ CLRM4	Loop back if page values are not the same.
00	0F	14						LDA YL	Put contents of YL into the accumulator.
00	10	06						CPA XL	Compare (YL-XL) to see if pointer values are same.
00	11	99	0B					RBNZ CLRM4	Loop back if page values are not the same.

Program *Routine for Clearing Array Elements.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
71	50	58	78				CARRAY	LDYH #78	Set up register Y to point to 16-bit address &7864.
71	52	5A	64					LDYL #64	This is where top of memory value stored by ROM routines.
71	54	15						LDA (Y)	Fetch top of memory value into accumulator.
71	55	DF						DEA	Decrement page value by one to point to next lower page.
71	56	08						STA XH	Store top of memory (adjusted) page value in register XH.
71	57	4A	FF					LDXL #FF	Store low portion of top of memory address in XL.
71	59	6A	1C				CARRA1	LDUL #1C	Set up counter (28 decimal here) in register UL.
71	5B	B5	00					LDA #00	Load the accumulator with zero.
71	5D	43					CARRA2	STAD (X)	Stuff accumulator into memory, then decrement pointer.
71	5E	62						DEUL	Decrement the counter value in UL (UL=UL-1).
71	5F	99	04					RBNZ CARRA2	If counter not zero, jump back to stuff another byte.
71	61	9A						RTS	Else, exit back to caller when counter equals zero.

lower page. Thus, if location &7864 contains the value &60 (indicating page &60 in the address &6000), the last location in RAM is at location &FF in the next lower page (&5F) or at address &5FFF. (This is precisely what would be encountered if a PC-1500 had an 8K RAM module such as the CE-155 installed.)

From there it is a simple matter to set up pointers and a byte counter within the CPU in order to erase the desired block of memory. However, now it will be appropriate to decrement the memory pointer as locations are cleared (instead of incrementing it as was done in previous routines). See the accompanying listing for the actual series of instructions that can accomplish this objective.

Check It Out With A Hybrid Program

You can test the operation of the array clearing routine by combining it with a BASIC program. The BASIC portion of the program can be used to DIMension the array, load the machine language routine into memory using POKE directives (since it is fairly short), and initialize the array with a set of known values.

Next, the initial values placed into the array can be displayed for checking purposes. The array-clearing ML routine can then be "called" using the CALL statement provided in BASIC. The contents of the array can then be displayed again to verify that the elements were properly cleared. This process may be repeated as long as desired for observational purposes. Here is the listing for such a *hybrid* program:

```
1000 "AA"DIM A$(6
     )*4
1010 GOSUB "CC"
1020 "BB"GOSUB "D
     D"
1030 GOSUB "EE"
1040 CALL &7150
1050 GOSUB "EE"
1060 GOTO "BB"
1090 END
1100 "CC"POKE &71
     50,&58,&78,&
     5A,&64,&15,&
     DF,8,&4A
1110 POKE &7158,&
     FF,&6A,&1C,&
     B5,0,&43,&62
     ,&99
1120 POKE &7160,4
     ,&9A
1130 RETURN
1200 "DD"FOR A=0
     TO 6
1210 A$(A)=STR$ (
     A)+STR$ (A)+
     STR$ (A)+
     STR$ (A)
1220 NEXT A
1230 RETURN
1500 "EE"WAIT 20
1510 FOR A=0TO 6
1520 PRINT A;"
      !";A$(A);"!
     ":PRINT " "
1530 NEXT A
1540 "FF"RETURN
```

(The nomenclature *hybrid* in this text refers to programs that combine BASIC and machine language methods within one general program, such as illustrated by this array-clearing example.)

Note that the ML routine is tucked into memory locations that are normally used for storing the string variables P$ and O$. Keep this in mind if you intend to meld this array-clearing capability into some other program!

INSIDE THE PC-1500

This is the sixth and concluding portion of the current series on the Sharp PC-1500. (Most of this information is also applicable to the Radio Shack PC-2.) *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158,* has been the provider of this extremely valuable information. We at *PCN*, on behalf of all our readers, would like to extend our thanks to him for a most informative treatise.

This final section picks up from where we left off in Issue 35, with a continuation of information regarding printing program lines....

(b) Subroutine D2EA may be used to locate the line whose number has been loaded into Register U. (A data byte, which will be added to the return address if the specified line does not exist, must follow the instruction calling this subroutine.) After execution of subroutine D2EA, load XH and XL with the contents of 78A6 and 78A7; decrement X twice; and execute subroutine D2D0. The specified line will be in the Input Buffer, and the SEARCH Pointer will point to that line.

(c) The Program line that follows the one currently pointed to by SEARCH may be Placed into the Input Buffer, with automatic update of the SEARCH Pointer, by execution of subroutine D2B3.

(2) To Print the line stored in the Input Buffer:

(a) Execute subroutine B73F to store the line size into 79F5. (If the CSIZE set is larger than 2, it will be set to 2.)

(b) Subroutine AE3A will Perform a line feed with carriage return, then Print the line contained in the Input Buffer. Follow by subroutine A769 (Motor off)

MISCELLANEOUS (MECHANICAL)

(1) Paper Feed Key Disable/Enable

Disable: Subroutine A306
Enable: Subroutine A30A

(2) To move Paper (no update of coordinates, GRAPH mode):

With n a two-byte signed binary number, subroutine AA0E feeds the paper by n graphic units. (Negative n will feed in reverse.) Precede by loading XH with 7B, XL with 7F, Y with n, and UL with 01.

(3) Pen UP/Down Control:

Pen UP: Subroutine AB08
Pen Down: Subroutine AAFA
Pen UP/Down: Subroutine AAE3 sets Pen up if 79E9 contains zero, down if 79E9 contains FF

(4) To move Pen (no update of coordinates, GRAPH mode):

(a) Move right, n graphic units, with subroutine A28E. Precede by loading XL with n. Follow by subroutine A769 (Motor off)

(b) Move left, n graphic units, with subroutine A28B. Precede by loading XL with n. Follow by subroutine A769 (Motor off)

(c) Return Pen to left end, without line feed, with subroutine A9D5. Follow by subroutine A769 (Motor off).

(5) To cycle Pen to next color:

Execute subroutine A629. Follow by subroutine A769 (Motor off)

CASSETTE ROUTINES

TO SAVE A FILE ON CASSETTE

(1) Construct Header in Output Buffer area (7B60-7BAF)

(a) Execute subroutine BBD6 to form Lead-in, including file type code. Precede by loading A with file type code: 00=ML; 01=BASIC; 02=Reserve memory; 04=data file; others definable by user
(b) File name is optional; if to be included, its character codes must be stored into addresses 7B69-7B78 (maximum, 16 characters)
(c) Store beginning address of file being saved, into 7B82-7B83; store 1 less than the number of bytes in the file, into 7B84-7B85
(d) If a ML Program (file type 00) is being recorded, store the beginning execution address (or default value FFFF) into 7B86-87

(2) Record Header onto cassette

(a) Store value of Cassette Parameter into 7879: 00=RMT 0, 10=RMT 1
(b) Execute Call B0 (BCE8)

(3) Record file

(a) Load X with beginning address (obtainable from 7B82-83); load U with 1 less than number of bytes (obtainable from 7B84-85)
(b) Execute Call AA (BD3C); terminate operation with Call B4 (BBF5)

TO LOAD A FILE FROM CASSETTE

(1) Construct Header in Output Buffer area, 7B60-7BAF

(a) Load A with code for file type sought
(b) Execute subroutine BBD6
(c) Store (optional) file name into 7B69-78

(2) Find file on cassette

(a) Store value of Cassette Parameter into 7879: 80=RMT 0, 90=RMT 1
(b) Execute CALL B0 (BCE8). (A search will begin for a Previously-recorded Header having the file type specified. If found, its file name is displayed and stored into 7B91-A0. If this name differs from the one specified, the search continues)
(c) If flag C is set, there was a checksum error; end with Call B4

(3) Load file into computer

(a) Load X with the address into which the file is to be loaded, and load U with 1 less than the number of bytes to be loaded. (The values for these parameters that were originally recorded are now obtainable, respectively, from 7BAA-AB and 7BAC-AD)
(b) Execute Call AA (BD3C). Following loading, Register X will contain the address immediately following the last address loaded
(c) If flag C is set, there was a checksum error; end with Call B4

(4) Terminate operation of the recorder with Call B4 (BBF5)

CONTROL OF RELAYS

Subroutine BF28 may be used to operate the remote control relays. The effect of this subroutine is determined by the contents of A.

A=03: Close relay, REM 0
A=05: Open relay, REM 0

A=09: Close relay, REM 1
A=11: Open relay, REM 1

RECORD, LOAD, OR VERIFY BLOCK OF DATA

Register X must be loaded with the beginning address of the block of memory involved, and Register U must contain 1 less than the number of bytes in that block. Then Call AA (BD3C) will perform the operation specified by the contents of the Cassette Parameter (7879), as follows:

When 7879 contains 00, the block of data is recorded onto tape.

When 7879 contains 80, the data read from tape is transferred to memory. If a checksum error occurs, there will be a return, with flag C set.

When 7879 contains 40, data read from tape is compared to the data in the block of memory, without any transfer of data. If a discrepancy is found, there is a return, with flags C and V set; if a checksum error occurs, a return with flags C and H set.

In each of the above cases, Call B4 (BBF5) may be used to terminate the operation; it will open the relay, shutting off the recorder.

RECORD OR LOAD A SINGLE BYTE

Subroutine BDCC will record the byte contained in the accumulator.

Call A4 (BDF0) will read one byte from tape, and load it into the accumulator. This routine also checks whether the BREAK key has been used; if it has, flag C is set.

1984 Index - Ordered By Type of Article

Article	Author	Issue	Page
Announcement: "The Commuter" Portable	PCN, Staff	31	1
Announcement: Casio FX-750P	PCN, Staff	35	1
Announcement: Casio PB-700	PCN, Staff	33	1
Announcement: Flight Computer Uses PC-1500	PCN, Staff	32	8
Announcement: Graphics Package for PC-2	PCN, Staff	32	8
Announcement: Hewlett-Packard HP-110 Portable	PCN, Staff	34	1
Announcement: Hewlett-Packard HP-71B	PCN, Staff	32	1
Announcement: Sharp PC-1350	PCN, Staff	36	1
Commentary: Watch Pocket	PCN, Staff	35	15
Commentary: Watch Pocket	PCN, Staff	36	16
Program: Addressing Pixels (PC-1500/PC-2)	Pollet, Patrick L.	33	9
Program: Alarm Clock (PC-1250/51/60/61 & PC-1211/PC-1)	Auersbacher, Enerich	36	3
Program: Alternate Character Sets (PC-1500/PC-2)	Rober, Norlin	35	10
Program: Appointment Calendar (PC-1500/PC-2)	Eichman, Steve	32	4
Program: Binary Magic (PC-1500/PC-2)	Gibson, John R.	33	11
Program: Formatted Mortgage (PC-1500/PC-2)	Hartmann, Peter V.	32	6
Program: Improved Vertical Lister (PC-1500/PC-2)	Rober, Norlin	34	8
Program: Isometric Drawings (PC-1500/PC-2)	Chrystie, Richard H.	34	10
Program: Linear Systems (PC-1500/PC-2)	Rober, Norlin	34	8
Program: Little Editor (PC-1500/PC-2)	Jackson, H. David	31	5
Program: Perspective Plot (PC-1500/PC-2)	Rober, Norlin	35	7
Program: Pocket Wordpro (PC-1500/PC-2)	Rober, Norlin	35	4
Program: Schedule Plotter (PC-1500/PC-2)	Chrystie, Richard H.	32	2
Program: Shell Game (PC-1500/PC-2)	Gibson, John R.	33	10
Program: Weather Forecaster (HP-71B)	PCN, Staff (Conversion)	36	5
Review: Casio PB-700	Woodhead, Jeffrey	36	2
Review: CE-501A, CE-501B, CE-505A Sharp ROM Modules	Sutliff, Robert	33	5
Review: Ce-502A, -502B, -504A, -507A Sharp ROM Modules	Sutliff, Robert	34	3
Review: Sharp PC-1261	PCN, Staff	35	2
Review: The Hewlett-Packard HP-71B	PCN, Staff	33	3
Review: The Sharp PC-5000 Portable	PCN, Staff	31	2
Technical Info: Clearing Memory and Arrays (PC-1500/PC-2)	PCN, Staff	36	7
Technical Info: Inside the PC-1500 (Part 1)	Rober, Norlin	31	11
Technical Info: Inside the PC-1500 (Part 2)	Rober, Norlin	32	9
Technical Info: Inside the PC-1500 (Part 3)	Rober, Norlin	33	12
Technical Info: Inside the PC-1500 (Part 4)	Rober, Norlin	34	12
Technical Info: Inside the PC-1500 (Part 5)	Rober, Norlin	35	12
Technical Info: Inside the PC-1500 (Part 6)	Rober, Norlin	36	12
Technical Info: Updates on Inside the PC-1500	Rober, Norlin	36	15

NORLIN UPDATES

Norlin Rober has noted a few errors in the ongoing *Inside the PC-1500* series that has been appearing in 1984 issues of *PCN*. Please make the following corrections:

Issue 31 - page 12 - line 785D should read: If 80, Katakana displayed; 00, displayed and... etc.

Addresses 786C to 786F *are* used by ROM, to temporarily save the contents of 787C-7F while the Display Buffer is temporarily saved in the String and Output Buffers.

Issue 31 - page 13 - 788E should read: TRACE parameter: 00, **ending** execution of **previous** line; 01, starting execution of **new line**;... etc.

Issue 31 - page 14 - location 79DA: . . . is obtained from 79DB - DC.

Issue 32 - page 11: In the middle of the page, the Output Buffer addresses for storage of tape header information are incorrectly given as beginning with 7A... the 7A should be replaced by **7B**.

Issue 32 - page 15: Under DECIMAL-TO-BINARY CONVERSION, the memory address of CALL D0 is incorrectly stated as D0F9; it should be **D5F9**.

Note too, that system RAM addresses that are "unused" by the PC-1500 actually are used by the CE-158. These are **7850 to 7858 and 79FA to 79FE**.

Norlin also notes that there is a mistake in his article in Issue 34 on page 8. The example given is not correct and its use can cause readers to think that the program is not operating correctly. The example printed should be changed so that the

constant term in the third equation should be 3 (not 1) and the constant in the fourth equation should be 4 (not 3). Additionally, when used with an *old* PC-1500 (having early versions of ROM), the list of coefficients will not be LPRINTed correctly. To remedy this situation, change the last part of line 32 to read:

```
LPRINT "C  ";A(I,N+1): . . .
```

instead of:

```
LPRINT "C  ";A(I,J): . . .
```

Finally, Norlin has a few general tips to pass on the PC-1500 users:

1. A BASIC program inadvertently cleared by NEW can be recovered as follows:

(1) POKE &7750, 204, 101, 202, 105, 73, 0, 69, 221, 153, 4, 70, 202, 103, 154

(2) CALL &7750

Note: If the number of the first line was larger than 255, it will be changed by this procedure. It may be corrected with the usual editing procedures. You can use the above routine, in some cases, to recover a program that has been lost by a crash. Key ALL RESET and execute NEW 0 *before* using this routine for crash recovery!

2. It is possible to make a BASIC program halt with automatic display of the RESERVE template (for whichever of groups I, II or III is currently set). This technique adds a nice touch to a menu-driven program. The program instructions for doing it are simply: POKE &7880,0:CALL &CB8B

3. Here is a way to clear RAM completely. Every byte of RAM will be cleared to zero, including system RAM, after which the power-up routine will automatically be executed.

(1) POKE &4000, 40, 36, 8, 10, 170, 255, 253, 186, 211, 197

(2) CALL &4000

(3) Key CL and execute NEW 0

Finally, although the instruction manual does not mention it, the BASIC statement ON ERROR GOTO 0 will cancel an ON ERROR directive!

FROM THE WATCH POCKET

As we close-out our fourth year of publishing this unique newsletter, it is interesting to note that Sharp Electronics continues to spearhead the effort to produce good quality, low cost PCs. Of course, many of the other pocket computer pioneers are still hanging in there. Casio continues to put up a good fight. HP is doing OK. Even Panasonic continues to push its HHC, though primarily to specialized markets.

There are probably well over a million PCs in use in the United States today. That is only about one for every 250 persons. It took about 20 years for calculators to reach their present ubiquitous status. Chances are it won't take as long for PCs to reach the same level of distribution. Based on that projection, the growth of the industry should be steep during the coming years. *PCN* looks forward to covering this advancing, exciting field.

By the way, during 1985 we will increase the frequency of publication to eight times (instead of the current six issues-per-year). However, rising postal and publishing costs dictate that we revert back to a smaller type style and our previous 8-page format. The combination of steps being taken will yield continued quality coverage of the pocket computing field in a timely manner. We all thank you for your continued support and extend our best wishes for pocket computing happiness during 1985, our fifth straight year of publication!