VOL.2
NO. 9

STATUS 1500

OCTOBER
1984

* 12 issues £10.50 (UK) * overseas £14.50 * published monthly *

* Edited by RONALD COHEN, 62 BLENHEIM CRESCENT, LONDON W.11 *

HEAVY GOING

More than a few readers find the newsletter heavy going at the moment, and this is inevitable. Much elementary ground has been covered in the past, and now we have to delve into more specialised aspects of using our computer - and (to make matters worse) they are aspects which have to be gone into fairly thoroughly. This can be boring for anyone who does not want to investigate so deeply; but it may console you to remember that material of no immediate interest may well be valuable for reference at a future date. Indeed, we have a whole 11 pages of solid stuff to plough through this month. We promise to make up for it by a shorter issue in November.

Last year's Xmas number was a great success; and I hope we shall do almost as well this year - but (again) it is up to you! We already have a few exciting games and entertainments, but we could use a few more. Deadline: early November.

"Not enough programs!" is a popular complaint. Most of you write programs, but few of you send them in, and of these only a minority are published. Few indeed meet the necessary criteria. Some are incomprehensible, some are too long. Some require specialised knowledge, or even special equipment. I try to print only programs which are either of some interest to most subscribers, or of important interest to a few. If you have programs which might qualify, please do send them: and don't be upset if they cannot be used immediately.

The retrogradation of Mercury, dire aspects between Saturn and Uranus, pressure of work, premature hibernation - in fact anything except ordinary carelessness - have caused a number of mistakes recently. Purely literary errors we will forget about: some more weighty ones are dealt with on page 84. Proof-reading is the real problem. If one deals with material which is still fresh in one's mind, one sees what one expects to see, and not what is actually on the actual page. A few days' interval would do the trick - and then the newsletter would be later than ever.

FRANCIS AKINWUNMI notes that 256*PEEK 30819+197 is suggested for finding the start of Program Area, whereas the CE 161 instructions state that programs should run from location 256 onwards. He wishes me to explain my reasoning.

This was discussed on page 118, Vol.1. No one has yet found any ill-effects from ignoring SHARP's instructions in this respect, and therefore in order to conserve space most of us are starting at 197. If you wish to be meticulous you could say start of program area=

256*PEEK 30819+197*(PEEK 30819<>0)+256*(PEEK 30819=0)

FRANCIS AKINWUNMI also writes on the subject of inverted commas. He suggests that CHR$ 162 (i.e. 34+128) functions just like CHR$ 34 for text purposes, but does not open or close strings, and may therefore be used safely where CHR$ 34 cannot. He suggests priming a Reserve key with CHR$ 162 for text purposes.

This is indeed a very clever idea, but not without its dangers. CHR$ 162 appears as a quotation mark on the display: but on the CE 150 it does not print at all. On a full-size printer your suggestion will usually work, but some dot-matrix printers have a supplementary set of characters, entirely different, from 128 onwards. Your demand that SUPERTEXT (B) - which you have never seen anyway - should be amended in accordance with your idea might have been expressed more tactfully.

J.P.WICKENS asks about saving space when DIMensioning arrays of numbers.

I started to answer your query in SIGNALS, but the reply ran on for too long, so you will find it in this month's PEEK & POKE.

TONY WILLMAN sends some information on the display pointers, and notes that changing the contents of these does not merely change the display, but actually alters the mode etc. that is indicated. He wonders however whether this has any practical use.

Some details of these pointers are given in Vol.1, page 42. It could be useful for a running program to switch automatically to PRO mode, at a certain point, for the amendment of Data statements, or other matter which must be inserted directly into the program. I have not actually tried this, but I have found it very handy to be able to switch automatically between upper and lower-case characters, when using a specialised database where some columns had to be INPUT in u/c and others in l/c, and when working fast I kept forgetting to switch, so decided to let the PC 1500 do this for me.

M.GREENING-LEWIS has been experimenting with the CE 161, and notices that when it is resident in the PC 1500 its extra consumption of current is considerable.

Of course, since it preserves memory contents for up to 5 years when in the computer, or up to 1 year outside it, it must consume the difference from the PC 1500. Perhaps this is why the batteries in my computer do not now last as long as I expect.

T.RASMUSSEN has now switched to the Casio TX500, and wishes me to write about that machine.

Not again! See Vol.1, page 65!

A.E.L.COX comes up with a useful piece of information on the Brother EP44, which is not given in its instructions. If as a result of wrong connections, or wrong commands, the EP44 goes mad, and starts printing rubbish as soon as you turn it on, its sanity may be restored by the following procedure. Turn the EP44 off, and then turn it on again while holding down the red CLEAR key.

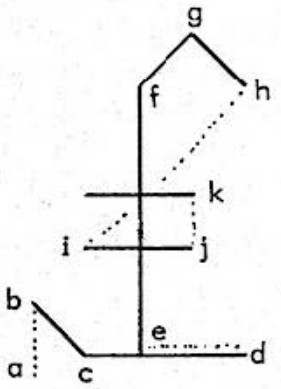
ROM CHARACTERS - 3 by David Rihoy

We have gone through the four stages in theory, so now let us put them to practical use. Let us design a pound sign. My design is shown in the figure below, and the accompanying table.

Table 5

Line	Ctrl	Drtn	Lnth	No
a	0	2	1	17
b	1	7	1	121
c	1	0	3	67
d	0	4	2	34
e	1	2	5	85
f	1	1	1	73
g	1	7	1	121
h	0	5	3	43
i	1	0	2	66
j	0	2	1	17
k	3	4	2	226

Diagram 3



If you have had the patience to follow the preceding explanation, then the construction of this table should be easy to understand. The first row moves the pen to position b, which is PEN UP (=control 0), vertically up (=direction 2), and distance (=1). So the number which gives this line is 17 (=0*64+2*8+1*1). The remaining rows are constructed similarly. For instance line h is drawn with the pen up in direction 5 a distance 3 LINSEGS and therefore the number is 43 (=0*64+5*8+3*1). The last line (line k) is calculated as 226 (=3*64+4*8+2*1).

This is included in line 20 of Program 2 below, which incorporates all the above ideas. If you plug it in, it also demonstrates how the routine can produce simple figures, in sizes of your choice without recalculation of the graphics. Just key DEF X for a demonstration. Good drawing!

[PROGRAMS on next page

MORE SIGNALS

D.E.CAMPBELL is a pilot and finds the PC 1500 helpful with navigation, particularly the descent of the aircraft. He is considering getting the TOOL-2, and enquires about address, price, etc.

It is certainly a joy to CLOAD a long program in 1 minute instead of 10 minutes, but it is only fair to say that some readers have had considerable difficulty with this module, whereas others have had absolutely no problems at all. Your suggestion that I should reprint information from vol.1 is not viable. It would be unfair to those who have taken the trouble to purchase the earlier volume, and now expect new material.

J.K.GAUTON has for sale a CE 155 (8K, non-battery) at £30.

Letters will be forwarded.

A.J.DEELEY has run into difficulties with 2 utilities from vol.1, DELETE and RENUMBER. He contacted SHARP, who advised him to take the batteries out of the computer for a few days, but this did not help. He wonders if there could be a fault in his equipment.

Nothing wrong with hardware at all. SHARP talking nonsense, as usual. The only problems were slight syntax errors: you left out a space in a prompt, so the utility was only 607 bytes long, and you were deleting 608, thus deleting the End-of-Program Marker from the subject program, which of course ran OK until you tried to edit it.

ROM CHARACTERS - 3 PROGRAMS

CHARACTER ANALYSIS

```
100 "H"CLEAR :DIM C(13,3):INPUT "Character? ";C$:S=40960:X=0:Y=0:C=ASC C$:CLS
110 CSIZE 2:LF 12:GRAPH :POKE 31220,36:SORGN :LPRINT C$:CSIZE 1
120 P=PEEK (S+C):P=256*(160+(C>58)+(C>101))+P
130 C=1:FOR I=0TO 1:V=PEEK P:C(C,0)=V
140 C(C,3)=VAND 7:Q=(V-C(C,3))/8:C(C,2)=QAND 7:C(C,1)=(Q-C(C,2))/8
150 F=(C(C,1)=2):G=C(C,2):GOSUB "d"
160 IF FRESTORE STR$ D:READ I,J:X=X+I:Y=Y+J:A=X*36:B=Y*36:GLCURSOR (A,B):
170 IF (1-F)LET D=C(C,2)                                          GOSUB "d"
180 RESTORE STR$ D:READ I,J
190 X=X+I*C(C,3):Y=Y+J*C(C,3):A=36*X+3:B=36*Y+2
200 GLCURSOR (A,B):LPRINT C(C,0)
210 C=C+1:P=P+1:I=(V>191):NEXT I:GLCURSOR (0,0):TEXT :LF 3
220 USING "####":C=1:FOR I=0TO 1:TAB 1:FOR J=0TO 3
230 LPRINT C(C,J);:NEXT J:LF 1:C=C+1:I=(C(C,0)=0):NEXT I
240 LPRINT :END
250 "0"DATA 1,0
260 "1"DATA 1,1
270 "2"DATA 0,1
280 "3"DATA -1,1
290 "4"DATA -1,0
300 "5"DATA -1,-1
310 "6"DATA 0,-1
320 "7"DATA 1,-1
330 "d"D=D+(G=0)-(G=4):D=D+8*((D<0)-(D>7)):RETURN
```

EXAMPLE

```
10 "C"C=1:INPUT "CSIZE?(MAX3)";C:CSIZE C
15 INPUT "ENTER A VALUE";V
20 POKE 30672,17,121,67,34,85,73,121,43,66,17,226,0
25 GOSUB "P":CALL P:LPRINT STR$ V:END
30 END
35 "X"CSIZE 9:RESTORE :X$=CHR$ 0:FOR I=1TO 16:READ X
40 X$=X$+CHR$ X:NEXT I:READ X:Y$=CHR$ X
45 GOSUB "P":CALL P:END
50 DATA 18,65,77,98,82,66,119,101,33,122,71,67,74,97,103,23,255
55 "P"P=STATUS 3-1000
60 POKE P,253,168,72,122,74,9,88,119,90,208,190,167,136,154:RETURN
```

CHARACTER CONSTRUCTOR

```
10 CLEAR :POKE STATUS 2+2,253,168,72,122,74,9,88,119,90,208,190,167,136,154
20 DIM X(16):F=1
40 INPUT "control ";A:INPUT "direction ";B:INPUT "length ";C:GOTO 50
45 BEEP 3,200:GOTO 40
50 BEEP 1:X(F)=64*A+8*B+C:X$=X$+CHR$ X(F)
55 IF A=3GOTO 70
60 F=F+1:GOTO 40
70 TEXT :CSIZE 2:FOR G=1TO F:LPRINT X(G);:NEXT G:LF 7
80 TEXT :FOR D=2TO 6STEP 2:LCURSOR D-1:CSIZE D:CALL STATUS 2+2:NEXT D
```

PEEK POKE & MEMORY - XIX

A DIMensioned numerical variable occupies 8 bytes, and this is wasteful of memory when large quantities of small numbers are to be dealt with. Greater economy would be valuable, particularly for the unexpanded.

There are in fact several methods of achieving this.

1) If your numbers are integers, not exceeding 5 digits, you could divide alternate ones by 10000, and pack each as a decimal appendage to the preceding one, thus getting 2 for the price of 1. To unpack, your numbers would be alternately INT A$(N), and 1E4*(A$(N)-INT A$(N)). Thus each 5-digit number will occupy only 4 bytes of memory. With care, this method could be adapted to deal with decimal numbers, provided they are all in the same range.

2) If your numbers are small integers, by DIMensioning and storing them as strings. Say you wish to deal with 250 3-digit integers. Then:

```
 5: DIM A$(250)*3
10: FOR F=1 TO 250
20: INPUT N: A$(F)=STR$ N
30: NEXT F
```

To unpack: N=VAL A$(F)

However if you are sorting these without unpacking them, in string form, you must be careful that the strings are all the same length. Since strings are sorted by the ASCII codes of the 1st and succeeding characters, "99" would be greater in sorting than "199". Obviate this by :

```
25: IF LEN A$(F)<3LET A$(F)=CHR$ 48 +A$(F): GOTO 25
```

If you wish to mix positive and negative integers, it is a little more complicated, and you would not be able to use line 25 in its present form. First you must ensure that all strings contain a sign:

```
 5: DIM A$(250)*4
25: IF LEFT$ (A$(F),1)<>"-" LET A$(F)="+"+A$(F)
26: L=LEN A$(F): IF LEN A$(F)<>4 LET J$=LEFT$ (A$(F),1):
    K$=RIGHT$ (A$(F),L-1): A$(F)=J$+CHR$ 48+K$: GOTO 26
```

With this method 250 integers ranging in value from -999 to +999 could be packed into just over 1000 bytes, instead of 2000 by normal methods.

3) If the numbers you have to deal with are all positive integers with values not greater than 255, here is the neatest and most economical method of all. Quite simply, treat the numbers as ASCII codes and store them as characters! To unpack, reverse the process.

```
 5: DIM A$(250)*1
10: FOR F=1 TO 250
20: INPUT N: A$(F)=CHR$ N
30: NEXT F
```

To unpack: N=ASC A$(F)

The only very slight snag with this system is that you cannot inspect the contents of strings directly. You must always remember to examine their ASCII codes, even if you think that you know all these by heart, because on the display 65 and 193 would both appear as A.

With the above method 250 numbers, each a positive integer not greater in value than 255, would occupy only 250 bytes, as against 2000 bytes in the ordinary way.

4) In fact this method may be adapted to store positive integers up to 65535, each in 2 bytes, in 256ary. Quite a few of our utilities have used this method for translating altering and replacing line numbers and pointers. If the PC 1500 can store numbers of this magnitude in 2 bytes, why should you not do the same?

LETS WRITE A PROGRAM - VIII
filling in a few gaps

We must try and complete the directions given by the menu at line 5300. The most important item is 'GRP', which stands for "Groups of numbers". As no doubt you know, at Roulette one can stake on groups of adjacent numbers by placing one's stake on the line or corner separating them. But how shall we know if numbers are adjacent? So we ought really to do the graphics first.

These are not hard, but do require planning, and patience. Apart from the Zero, which we will treat separately, we have a simple symmetrical arrangement. First, some measurements. The width of the paper is 216, so 3 boxes across will be 72 each. Since they should be square, a depth of 12 boxes gives a depth of 12*72=864 LINELS. We could draw 36 boxes and their numbers in sequence, but this is uneconomic. We prefer to draw 1 big box and divide it up by lines across and down.

Unfortunately the distance of 864 LINELS is too great for the paper to pull back far enough for drawing a box. So lets do it in 2 halves. 6*72=432 is less than 500, and just fits nicely.

It might be possible to calculate exactly the size and placing of the figures in the boxes, but it could be tedious. It may be easier to make an intelligent estimate, and correct it by experiment.

But before writing this routine it really is important to plan exactly what loops we need. On my first attempt I ignored my own advice, and just started writing code. The result was so tangled that I had to scrap the lot and start again.

So what do we need? 1) An outside loop, for "how many copies?" Sometimes 1 master copy will be sufficient; sometimes 1 for each player. 2) A loop for the 2 halves described above. 3) In each of these a) a loop for drawing the horizontal lines, and b) another for the vertical lines. Then to put the numbers in. To avoid changing pen color 36 times, we should be able to insert first all the red ones in the half layout we are dealing with, and then the black ones. So, still inside loop 3), we want 2 further loops c) and d) for this purpose. And, of course, we must not forget Zero!

A couple of minor problems need to be sorted out before we start. The first, to repeat the second set of boxes immediately after the first, is easily dealt with by the use of the expression SORGN. The other problem appears less simple. Having inserted the numbers 1 to 18, we certainly do not want the same set of numbers again! A little ingenuity may be used here. Instead of our loop for the 2 halves being

```
FOR F=1 TO 2
```

let us make it

```
FOR F=0 TO 1
```

Then when we put the numbers in, something like the following will take care of the full range:

```
FOR G=1 TO 18: LPRINT STR$ (G+18*F)
```

We shall need a technique for actually placing the numbers. Here something along the following lines may do the trick. Horizontal positions 1,2,3 indicated by a value P. After each number is printed, P=P+1. If P>3, then P=P-3, (back to horizontal position 1), but vertical position will be V+1. All these of course multiplied by 72, which is the side of our square.

As for printing one color only at a time, we have already made provision for this. Just GOSUB "odd/even" which leads to "color". The complete routine will be printed next month - but have a go meanwhile!

MINDBOGGLE CORNER

The one-liner competition proved popular, so here is another on the same lines. A game or entertainment in not more than 5 lines - or less if you can! Closing date: November 15th, in time for the Xmas edition.

Here are the results of the August competition. All are self-explanatory. The winner is MIKE O'REGAN, by a short head from A.E.L.COX. His program is the more ingenious, but the other is the more generally useful, and so wins.

ANGUS CRAWFORD'S contribution is dangerous: do not use it if you have any valuable program or data in memory, since they will be destroyed. Your editor did not heed the suggestion that the program should be analysed before use, since this was not quite in the spirit of the competition. In consequence 2 pages of this newsletter were lost. (The best excuse yet for late publication?).

C.P.UNDERWOOD claims that his program is a fully-developed Word-Processor. This is perhaps exaggerating slightly; the routine works fine with CE 150, but cannot be handled via CE 158 in its present form.

from Mike O'Regan

```
1: ARUN: USING "###.####": T=((TIME/100)-INT (TIME/100))*100:
     PAUSE T;" GMT": BEEP (T*(INT (T)=T)): GOTO 1
```

from A.E.L.Cox

```
1: CLEAR: C=1: B=π: FOR J=1 TO 4: B=1/(B-INT B): A=INT B*C+D:
     D=C: C=A: LPRINT  "π="; INT (π*A+.5);"  /"; A: NEXT J
```

from Angus Crawford

```
1: POKE 30976,165,120,100,8,181,0,249,67,76,1,147,6,78,1,147,10,
     186,227,63: CALL 30976
```

and from C.P.Underwood

```
1: LPRINT INKEY$;: GOTO 1
```

ERRORS

page 73

The routine for capital letter U reads:

```
10: FOR N=1 TO 5: ... : NEXT A
```

It should of course end: NEXT N

page 75

In the SELF-DELETION subroutine no provision was made for restoring the SEARCH/ EDIT pointers. Line 999 should read:

```
999: POKE 30823,Q,R,PEEK 30821,PEEK 30822: POKE P,255
```

page 75

In the IMPROVEMENT to SCREEN-DUMP, line 45 ends

```
...GOSUB 110: GOTO 40
```

It should in fact end:

```
...GOSUB 100: GOTO 45
```

series 1 amendments to Supertext (B)

Line 33000 is shown as:

```
33000: POKE 86+256*PEEK 30819,1,.....
```

The label "B" was unfortunately omitted. The line should commence:

```
33000: "B" POKE 86+256*PEEK 30819,1,...
```

SORTING ROUTINES - 3 by A.E.L.Cox
the Shell Sort

The difference between the Bubble and the Shell sort is that, whereas the Bubble compares adjoining registers, in the Shell sort the registers compared have a large gap between them, which decreases in length with each new pass.

Shell Sort: *Routine 3.1*

```
100: D=A
110: D=INT (D*.73+.5): A$(0)=CHR$ 0: FOR B=1 TO A-D
120: IF LEFT$ (A$(B),16)>LEFT$ (A$(B+D),16) LET A$(0)=A$(B):
     A$(B)=A$(B+D): A$(B+D)=A$(0)
130: NEXT B: IF D>1 THEN 110
140: IF A$(0) THEN 110
150: END
```

If the number of strings (A) is 101, then the gap length (D) is 74 for the first pass, which starts at A$(1) and A$(75), and ends at A$(27) and A$(101). Subsequent gap lengths will be 54,39,28,20,15,11,6,4,3,2 and 1 (1 being adjoining registers). When the gap length is down to 1 there is no certainty that sorting is complete, and so running continues at gap 1 until a pass has been made in which there are no exchanges. Usually two or three passes take place at gap 1, sometimes four, but I have never known there to be more than four.

The mysterious factor of 0.73 for diminishing the gap length was arrived at after many trials and gives the fastest run times irrespective of the number of items. I sorted several different random-order sets of 101 strings and the average run time was almost exactly 2 minutes, which is more than four times as fast as the best Bubble sort. The number of inner loops averaged about 1250, and the exchanges only about 250. The increase in speed is quite remarkable, and for such a small increase in program length. There is however one minor disadvantage to be overcome. The Shell has to run through its fixed routine of decreasing gap lengths whether the strings are random-ordered or already partly sorted. To get round this, two alterations are required. If D is made equal to 1 at the start, the routine converts to a Bubble; also the direction of the passes can be inverted by amending the last statement in line 110:

Shell Sort, with Inverted Bubble alternative: *Routine 3.2*

```
100: "A" D=A: GOTO 110
101: "B" D=1
110: D=INT (D*.73+.5): A$(0)=CHR$ 0: FOR B=A-D TO 1 STEP -1
120: IF LEFT$ (A$(B),16)>LEFT$ (A$(B+D),16) LET A$(0)=A$(B):
     A$(B)=A$(B+D): A$(B+D)=A(0)
130: NEXT B: IF D>1 THEN 110
140: IF A$(0) THEN 110
150: END
```

We now have the best of both sorts: DEF A for a fast main sort, and DEF B for quickly inserting a few additional entries.

The time taken by Bubble sorts is roughly proportional to the square of the number of items being sorted - i.e to sort 100 items takes not 5 times but 25 times as long as sorting 20 items. But using the Shell sort, the increase in time is only about 1.3 times the increase in quantity, so sorting 100 items would take about 8.5 times as long as sorting 20. So the greater the quantity of strings to be sorted, the greater the advantage of the Shell sort.

Next month we shall investigate various ways of dealing with the problem of sorting quantities larger than 255.

EASI-NOTE reviewed by Sydney Lenssen

Having had fun reviewing Supertext (B) recently, I was delighted to receive Easi-Note. Then the prospect of being forever typecast as a 1500 W/P reviewer suddenly hit me. But the market is surely not so big, and there cannot be many more packages to go. I have indeed enjoyed contrasting the two programs.

MINIMICRO package their software well, and go to considerable lengths to give clear instructions which are readily understood. Easi-Note is no exception, with a 13 page leaflet which tells you all you need. I found the fastest success comes from just trying the program while reading the detailed instructions, just typing in rubbish to give yourself confidence. After a while you will find a few snags which are not explicitly covered by the instructions - e.g. how do you start a new paragraph? (Returning to the menu in fact does the trick.)

The PC 1500 displays a menu for the user to choose the various modes, and I found this more comfortable than Supertext; I found I got 182 lines of text available with 16K memory, and 29 lines even with 4K. I found wordwrap and changing modes speedier with Easi-Note.

Another big difference is the editing/reading of entered text. Easi-Note scrolls continuously across the screen at a speed which can be varied. You can freeze the screen and alter single letters using the same insert/delete method as when writing a program, or replace a whole line by pressing the RCL key.

The crucial question will be "which program is easiest to use?" Of course, there is no easy answer. I found the shimmering march of words across the screen a little dazzling, and thus missed several errors. Editing is easier with Supertext; and if you wish to format your output - print in colour - space out paragraphs - leave unjustified - or if you find it difficult to keep in mind what you have just written, as some do when they dictate, then you would have to plump for Supertext. But if you are prepared to accept the choices already made for you by Easi-Note then I can see many preferring its ease of operation.

When it comes to printing, there is really no comparison. Supertext is in a different class, and offers virtually all permutations. With Easi-Note, you can only use the CE 150. When it prints out in CSIZE 1 or 2, it takes an age - well 6 or 7 seconds - between each line every time, as the justification is done from scratch. Although usually placid, I found this feature singularly irritating. I am surprised that the author of the program did not design round this. I also regretted that there was no facility for printing out a selected part of the text - it is all or none!

If you can, I should somehow buy both programs. The best comparison I feel is between a fully automatic camera and one with all the facilities which have to be set by the user. If you understand how a camera works, you would never buy the robot which does it all for you, because inevitably most of the usual conditions will defeat the pre-settings. I want as much control as possible; especially since Supertext makes adjustment so easy anyway.

But I am sure there are many others, who do just want to make notes, as they go about the day's affairs. They will find Easi-Note merits its title, and is most suitable, and they will like its handy facility which can note the time and date of each entry.

EASI-NOTE is written by S.Rickaby and marketed by MINIMICRO, £19.95

EP44 AND PC 1500 by David Bowring

The Brother EP44 is a portable electronic printer/typewriter. It uses thermal or plain paper. There is a 4K memory, 16-character display and an RS232 interface. Brother state in their literature that the machine "can be used as an I/O printer, able to print out texts ..... from your computer". They also say that it can be used to transmit via a telephone coupler as a communications terminal. In a limited way the EP44 can be used for input into the PC 1500.

1. Interface connexions

Obviously the PC 1500 must be used with its own RS232 interface, the CE 158. The CE 150 plotter can be connected at the same time.

A special interface cable must be used between the EP44 and the CE 158. This must have 25-way male DIN plugs at both ends. The following pin connexions are required:

EP44	PC 1500
2	3
3	2
20	5
7	7

2. Setting Input/Output modes

a) EP44

Switch to TERMINAL.
Using the Mode and CR keys set:

Baud	110
Bit	8
Parity	N
New Line	CR
Code	7bit
ER	Y

b) PC 1500

i. In PRO/RUN mode:

```
SETCOM 110,8,N,1  enter
CONSOLE 0,0,1  enter
```

ii. In TERMINAL mode set:

Baud	110
Word	8
Parity	N
Stop bit	1
Echo	ON
Trace	OFF

Note: The batteries in both units must be fully charged, or the a/c adaptor used. 300 Baud can also be used, but higher values may cause loss of information.

3. General Notes

For printing out of full or part programs, data, lists, etc., the EP44 performs adequately in the role I require. To use the EP44 as an input medium its main use would be for loading of data via the TERMINAL mode of the PC 1500 and then converting it for use within the program. The machine cannot be used as a "full-size" keyboard for easy typing in of programs. Perhaps future experiments with the interface cable (DSR and CTS signals) may enable loading programs to the PC 1500, if time permits. This article will be continued next month, and will deal with Transmitting Data to and from the EP44.

TWO NEW 'EASI's from minimicro

Put your thoughts on paper - neatly, dramatically with two new programs from the Minimicro library

en EASI-NOTE 1500

EASI-NOTE is a sophisticated text editor that allows you to use your PC1500/PC2 to rapidly enter text, view it, correct it and print it out in a neatly justified 80-character format in either of 2 type sizes. Features include:-

- Easy text entry with full word-wrap between lines
- View the whole document by automatic scrolling at a speed selected by you - and move to any line in the document easily and quickly
- Insert or delete characters at any point - or rewrite whole lines
- Optional "time-stamping" of text entries for data-logging, laboratory measurements etc
- Storage and recovery of text from tape
- Usable with any RAM module from 4K upwards - and use as much or as little memory as you need
- Ultra-compact coding allows documents of about 300 words to be entered with 4K RAM and over 1000 words with 8K RAM.

ec EASI-CAP 1500

EASI-CAP - previously known as CAP-SET - has been significantly improved to meet Minimicro's exacting standards. It enables you to produce outline capital letters up to the width of the CE-150 printer paper.

- Multiple lines of text can be printed in different sizes and colours, upright or sloping left or right.
- Originally designed by a professional sign-writer to ease his workload, EASI-CAP can, with scissors and paste, be used as a cheap alternative to Letraset to provide dramatic headlines for memos, sales letters, club bulletins - or in any situation where you want your written words to have an impact. Commercial artists are using EASI-CAP to provide rough visuals for their clients.

Both programs feature Minimicro's easy-to-use program structures and detailed documentation.
EASI-NOTE requires a minimum of 4K RAM pack.
EASI-CAP requires 8K RAM pack as a minimum.

Prices: - EASI-NOTE £19.95 (incl VAT, p&p)
EASI-CAP £13.95 (incl VAT, p&p)
Both programs ordered together £29.95 (incl VAT, p&p)

Minimicro software is available from:

WMM Micros for Managers
149 Gloucester Road
LONDON SW7 4TH
Tel 01-370-5125

ELKAN ELECTRONICS
11 Bury New Road
Prestwich, MANCHESTER M25 8JZ
Tel 061-798-7613

who can also supply

EASI-CALC - The Spreadsheet
EASI-FILE - The Database
EASI-TREND - The Forecaster
EASI-CASH - The Book-keeper
£19.95 each (incl VAT, p&p)

EASI-ONE! - Multi task program
- spreadsheet, mini-texthandler, statistics...etc, etc
£24.95 (incl VAT, p&p)

If it's SHARP enough to go in your pocket Minimicro brings it to life!