VOL.2
NO.10

STATUS 1500

NOVEMBER
1984

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

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

ALL THE BITS AND PIECES

Portability is all very well, but have you ever counted up all the bits you need in order to be truly portable? With the EP44 or some other type of electronic typewriter-printer you can be entirely self-contained. All you need is your PC 1500, and a cassette-recorder, for data and programs, and the CE 150, to allow the computer to communicate with the cassette-recorder, and of course the cable from cassette-recorder to CE 150, and the EP44, and the CE 158, and the cable from the EP44 to the CE 158, and of course the instruction book for the PC 1500, and the instruction book for the CE 158, and the instruction book for the EP 44 - (since we are travelling light, we won't bother to carry the instructions for the CE 161). However, since batteries may be in short supply, we must connect up a mains-adaptor for the CE 150, another for the CE 158, another for the EP44, and a fourth for the cassette-recorder. Add some paper for the EP44, spare ribbon, some batteries, cassettes, pens and paper-rolls for the CE 150, a few dozen extracts from this newsletter, and we are away!

Why not something really neat and portable, all in one! SHARP do make a lightweight typewriter/printer. Why not a version with its own mini-cassette-recorder, and a slot for slotting the PC 1500 into the printer, as it does into the CE 150. And I am sure that the interfaces of the CE 158 could be tucked away more neatly. I like the colour graphics on the CE 150, but would willingly sacrifice colour for a full-page printout. The dot-matrix system of the EP44 is perfectly fine enough for decent graphics.

Of course I would like a few necessary luxuries too. A display large enough for a whole line of program. A choice of fonts on the printer, either by calling them or by inserting modules. And most desirable of all:
NO user-memory in the PC 1500! ALL memory to be on 16K modules, so that I can take out or insert a WHOLE PROGRAM and the data it has generated.

Anything else? Well of course I would like the system to be reasonably priced .....

89 ALL THE BITS & PIECES
90 SIGNALS
91 PEEK POKE & MEMORY - XX
92 LETS WRITE A PROGRAM - IX
93 SORTING ROUTINES - 4
94 FROM THE KEYBOARD
95 EP44 & PC 1500 part 2
96 MINDBOGGLE CORNER
97 MARKETPLACE

F.C.ODDS points out that the price of £99 for the CE 161 (see page 59) is without VAT or carriage, and the true price is more like £115.

E.MACMILLAN suggests that CALL &D091 will CLEAR 2-chr. and DIMensioned variables, while leaving Fixed Variables undisturbed.

This is the same effect as one gets with RUN, is it not?

M.GREENING-LEWIS points out a danger with ROM character construction, and indeed with graphics in general. If the pen is left poised in the middle of the paper, instead of being returned to the side, then the consumption of current is considerable, and it is even possible that over-heating could result.

It could be helpful to note that TEXT will always return the pen to the side, even when already in TEXT mode. Similarly GRAPH will do the same, even in GRAPH mode: but this will of course reset your X,Y co-ordinates.

Your further query about ROM characters, as to why we use the expression CHR$ and not STR$ to get the numbers into their locations, shows a fundamental misunderstanding of the working of the computer's memory. Perhaps this month's PEEK & POKE may help.

SYDNEY LENSSEN wishes to be reminded of the method suggested for entering a new program while still preserving the one at present in memory. He emphasises that he will not be satisfied with a reference to the previous explanation (vol.1, p.118) but wishes the method to be described anew.

a) Note contents of system pointers 30821 to 30826
b) Key NEW STATUS 2
c) Write the new program.
d) To restore the old one, key NEW.
e) Then POKE the figures you noted back into the system pointers.

It seems to me that there are 3 ways of making use of this newsletter:
1) To study and experiment with each technique as it appears.
2) To note each for future use as occasion may arise.
3) To vaguely recollect subjects dealt with, and require them to be explained again if you need to make use of them.

Of the 3 methods, the last is the least helpful. You have complained of this column becoming too bland: I hope this reply is sufficiently savage for you. If not, please read between the lines.

If you do not wish to study how the computer's memory works, I really cannot recommend using the technique you are enquiring about. If you make a mistake, and anything goes wrong, you will not know how to put it right, and may lose both programs.

R.H.DAVIS asks whether a machine-code routine would not be the fastest method of all for Sorting.

Undoubtedly. But the object of the series was not merely to provide a few ready-made routines, but to explain the underlying methods used, so that readers could adapt them for their own specific needs. Machine-code routines must be taken on trust. They cannot easily be read, nor briefly explained, nor simply adapted. If any reader cares to send me such a routine in m/c I shall be delighed to print it.

ANGUS CRAWFORD sends some further information on the PC 1500A. The 8.5K RAM breaks down as follows: System area 1.9K // Input buffer 80 bytes // Stack 196 bytes // Machine-code Free Area 1K (&7C01 to &7FFF) // User Area 6.6K // Basic Program Area 5946 bytes // Reserve Area 188 bytes. He confirms that the CE 161 is indeed compatible with PC 1500A.

BASIC is misleading. It appears to make sense. This can lead users to assume that something that makes sense grammatically, or indeed mathematically, makes sense to the computer. This is not necessarily so. An assumption even harder to disentangle is the idea that memory locations hold BASIC statements. This is exacerbated by the fact that we frequently cut corners, and refer to them as if they do.

But the actual contents of memory locations are just numbers - and even this is a simplification. We talk about POKEing a number into a memory location, but this is really only the EFFECT of what we do. To be strictly accurate, what a memory location holds is a set of 8 switches.

OFF OFF OFF OFF OFF OFF OFF OFF would return CHR$ 0.

Whereas ON ON ON ON ON ON ON ON

would return CHR$ 255. These switches are conventionally represented by 1 for ON, and 0 for OFF, and thus 11111111 is a binary representation of decimal 255, and may be retrieved as such. When we say we POKE a number into a location, we really mean that we set the switches in that location to return the number required.

But on retrieving the contents of a location, they are not necessarily used as a number. It depends on the function to which that location is assigned, and the method of retrieval used.

There are a number of ways by which memory locations are filled. If you write a program, your first line will start with 2 locations for the line number, 1 for the quantity of locations occupied by the line, numbers representing the ASCII codes of the statements of the line, and then the last location occupied by the line will be set at 13, indicating the End-of-Line Marker. If you POKE a number into one of these locations you will alter the program, or the line number. If you respond to an INPUT prompt (i.e INPUT A$) with a string of characters, then the locations corresponding to A$ will be set at numbers corresponding to the ASCII codes of those characters. But it is quite wrong to say that A$ contains ASCII codes. It contains numbers: and if you retrieve A$ then the computer goes to the locations corresponding to A$ and interprets the contents as ASCII codes. If you retrieved the PEEKs of these locations they would be displayed as numbers. If you CALL these locations then the series will be treated as a machine-code program. The numbers in the locations allocated to A$ are only ASCII codes if you treat them as such.

This enables us to play tricks. If we have bothered to study the Instruction Manual for the PC 1500 we know that ASC will return the ASCII code of a character, whereas CHR$ will return the character corresponding to the number in any location. If A$="1345" then VAL A$ will be simply the number 12345. The process is reversed by STR$. If A=12345 then the statement A$=STR$ A will make A$ equal to "12345" and the locations of A$ will hold 49 50 51 52 53.

These tricks have been found useful in the construction of ROM, CHARACTERS. If we want Control 1, Direction 2, Length 3 this would be 01......+..010...+.....011. Put them together as: 01 010 011. This may be treated as the binary representation of decimal 83. We have used X$, so that we can get this number into a location of X$ by saying X$=X$+CHR$ 83, just as we would do if we wanted to add letter S. The computer is not interested in what use we make of this. Of course if we retrieve X$ as a string variable, it will seem nonsense. But in fact the sequential locations of X$ will be retrieved as numbers, and each of these will be broken down again into a set of switches to execute the lines which make up our ROM CHARACTER.

LETS WRITE A PROGRAM - IX

Last month I told how I had attempted to write the "graphics" routine, and how - mainly because of the number of loops and repetitions involved - I had got in a muddle with it. So instead I attacked the problem in a more patient manner, by describing in detail the problems to be tackled, and how I intended to deal wth them. I put off the chore of actually writing the routine (NOT the way to write a program!) until this evening. Because of all the loops, I did not even contemplate keying the program directly into the computer, but wrote it out on paper first. I followed the lines I had laid down. When I keyed it in, I was agreeably surprised to find that hardly any adjustments were necessary. In line 2140 I had written COLOR C, and this caused many unwanted (and unused) pen color changes. These were easily obviated by COLOR Z. Apart from one or two such minor matters, such as the values of P and Q, and saying in line 2110 W=N+F instead of W=N+18*F, it worked perfectly almost first time. I was happy also to find the routine slotted in neatly at 2000. You can try it out on its own, but you will need to add the brief "odd/even" and "color" subroutine starting at line 40500 (page 63) in order to make it work.

```
2000 "table"INPUT "how many copies? ";CC
2010 FOR CP=1TO CC:TEXT :CSIZE 2:LF 12:GRAPH
2020 LINE (0,0)-(216,72),0,2,B
2030 GLCURSOR (80,20):CSIZE 4:LPRINT "0"
2040 FOR F=0TO 1:LINE (0,0)-(216,-432),0,2,B
2050 FOR G=72TO 360STEP 72
2060 LINE (0,-G)-(216,-G):NEXT G
2070 FOR G=72TO 144STEP 72
2080 LINE (G,0)-(G,-432):NEXT G
2100 FOR Z=0TO 3STEP 3
2105 P=12:Q=-60
2110 FOR N=1TO 18:W=N+18*F
2120 GOSUB 40500:N$=STR$ W:IF C<>ZLET N$=CHR$ 32
2140 GLCURSOR (P,Q):COLOR Z
2145 LPRINT N$
2150 P=P+72:IF P>156LET P=12:Q=Q-72
2160 NEXT N:NEXT Z
2170 GLCURSOR (0,-432):SORGN :NEXT F:NEXT CP
```

We still have to tackle the problem of staking on groups of numbers. We can now see which groups are legitimate. We may need quite a few IF statements for the various sorts of group. 3 numbers, such as 1 to 3, or 6 numbers, such as 1 to 6, should present little difficulty. Pairs of numbers, such as 1 and 4, or 18 and 21, are a little harder. But you notice such pairs are always 3 apart. So we can say that where the bet is group N to (N+3) then the stake is divided half on number N and half on number (N+3). This gives a guide for dealing with stakes on groups of 4 numbers: for we can see that any such bet, on 1,2,4,5 (placed theoretically on the intersection between them) gives a difference of 4 between first and last. We may be able to specify that where a bet gives this difference, the stakes are divided between the first 2 numbers and the last 2.

We still have to write the "wheel-spinning" display: I hope to complete some more routines next month.

SORTING ROUTINES - 4
Two-dimensional sorts

by A.E.L.Cox

To sort more than 256 character strings calls for a two-dimensional array. In the following program, the principle employed is to treat the entries as a single series from 1 to A, where A is the number of items. For the INPUT routine, and for print-out, each serial number is converted into two subscripts, E and F (lines 40 and 150). But for the Shell sort this method would be too slow, and so the principle adopted is to start each pass with the correct values of the subscripts for the two variables, and then increment the second subscript after each repeat. This will result in ERROR 9 when a second subscript exceeds the limit set by the DIM statement in line 30, but then ON ERROR GOTO takes over; and both subscripts of the variable causing the ERROR are adjusted so that computing continues without interruption.

This system keeps the sorting loop, which has to be executed thousands of times, as short as possible.

While a 2-DIMensional Bubble Sort is feasible in theory, with our machine, in BASIC, it would be inordinately slow for large numbers, and so the program below is written for the Shell sort.

Approximate running times for the sort routine were: 400*40 in 10m; 800*21 in 23m; 1200*14 in 38m; 1600*10 in 52m; and 2000*8 in 67m.

Shell sort, 2-DIM array

```
 10: CLEAR: INPUT "String length? ";Y
 20: Z=INT ((STATUS 3-STATUS 2-7)/Y):
      X=INT(Z/256): Z=INT(Z/(X+1))-1
 30: DIM A$(X,Z)*Y: Q=(X+1)*(Z+1)-1:
      WAIT: PRINT "Max items=";Q
 40:"A" WAIT 0: A=A+1: E=INT(A/(Z+1)): F=A-E*(Z+1)
 50: CLS: INPUT "INPUT?";A$(E,F):
      IF A<Q THEN 40
 60: WAIT: PRINT "FULL": END
 70: "B" WAIT 0: CLS: PRINT "SHELL": D=A*1.05
 80: D=INT(D*.73+.5): A$(0,0)=CHR$ 0
 90: K=0: L=0: M=0: N=0: ON ERROR GOTO 140
100: FOR B=1 TO A-D
110: IF A$(K,B-L) > A$(M,B+D-N) LET A$(0,0)=A$(K,B-L):
    A$(K,B-L)=A$(M,B+D-N): GOTO 130
120: NEXT B: IF D>1 THEN 80
121: IF A$(0,0) THEN 80
122: ON ERROR GOTO 0: END
130: A$(M,B+D-N)=A$(0,0): GOTO 120
140: IF B-L>Z LET K=K+1: L=L+Z+1: GOTO 110
141: M=M+1: N=N+Z+1: GOTO 110
150: "C" FOR J=1 TO A: E=INT(J/(Z+1)): F=J-E*(Z+1)
160: LPRINT A$(E,F): NEXT J: END
```

Explanations:

Line 10:

The string length must be less than 41 to avoid ERROR 15, as explained in my first article. Obviously the shorter the string length specified, the greater the number of strings that can be stored and sorted.

Lines 20 to 30:

These two lines compute a suitable DIM, and the capacity, Q. STATUS 3-STATUS 2 is the number of spare bytes but 7 bytes must be allowed for the storage of a 2-dim array. Division by the string length gives the number of bytes available for allocation between the 2 subscripts.

Division by 256 gives the first subscript, X. 256 is chosen as the divisor to obtain as long a run as possible between the points where the subscripts have to be adjusted. The second DIM subscript is obtained by dividing by (X+1) and subtracting 1. To compute the capacity, Q, 1 must be added to X and Z to take account of the zero values, (n,0), before they are multiplied together; then 1 is subtracted for the working register, A$(0,0). (It may then be simplified to: Q=X*Z+X+Z.)

Lines 40 to 60:

This is the input routine. The tally, A, has been zeroed in line 10. The two working subscripts, E and F, are computed in line 40, using Z+1 as the divisor.

Lines 70 to 141:

DEF B starts the Shell sort routine. These longer sorts, with more than about 200 items, run some 4% faster if the initial gap length is increased - hence the 1.05 factor in line 70. It causes a reduction in the number of gap-1 passes. The second subscripts consist of B, plus (for the upper variable) the gap length, D, minus L or N, which is the adjustment made when the net current value tops the limit.

This program cannot easily be converted to inverted passes and there is therefore no point in providing for a Bubble sort here.

Lines 150 to 160:

The print-out routine is similar to the input. Additions can and should be made to this and other parts of this program to suit the user's personal requirements; they will not reduce the capacity by much.

External Sorts

For still longer sorts, when there are more items than the computer can store all at once, the only possible method is storage of sorted data, section by section, on tape, but this requires special techniques which are outside the scope of the present series.

However the process (which I use frequently) is by no means as slow nor as cumbersome as one might imagine, and may be the subject of an extra article in a future issue.

The Shuffle

The opposite of sorting is shuffling, and to round off this series here is a one-line shuffle routine.

[It is assumed the variables to be shuffled are stored in A$(1) to A$(A)]

```
500: FOR J=1 TO A: B=RND A: A$(0)=A$(J):
        A$(J)=A$(B): A$(B)=A$(0): NEXT J: END
```

Nothing more complicated than this is required!

FROM THE KEYBOARD

Since BEEPs of different tones must be specified at different lengths in order to sound of equal duration, guessing a suitable length can be tedious. Did you know that the length parameter need not be specified at all? Statements such as:

```
BEEP 1,50     or     BEEP 3,255
```

are perfectly legitimate, and will give sounds of reasonable duration, though of course this duration will vary somewhat according to the tone.

EP44 AND PC 1500 (part 2)

by David Bowring

4. Transmitting data from the PC 1500

a) To LLIST program or specific lines:

Set both machines as described last month, with PC 1500 in RUN mode.

```
enter SETDEV PO
      LLIST or LLIST (line no.) etc.
```

The program will now LLIST to the EP44.

b) Using CSAVE "(filename)":

```
SETCOM CO
CSAVE "(filename)"
```

The output will be @COMfilename followed by printing of PRINT statements having strings i.e. all information enclosed between "....."is printed.
This is useful if the programs are menu-driven, and a list is required for reference.

c) CSAVEa;"....."

Again with SETDEV CO printing starts at the program label enclosed between " ".

d) In TERMINAL/DTE mode:

Any information typed on the PC 1500 will be output to the PC 1500 and EP44 screens, and printed.

e) In PRO/RUN mode:

Using SETDEV PO:

Any information which is normally printed by the CE 150 in obedience obeying an LPRINT instruction in a program will now be output to the EP44; however PRINT statements will still output to the PC 1500 display.

Alternatively SETDEV DO will output PRINT statements to the EP44, while LPRINT statements are dealt with by the CE 150.

5. Transmitting data from the EP44.

a) PC 1500 in TERMINAL/DTE mode:

As information is typed onto the EP44 it is displayed and stored in a Buffer in the PC 1500. To make use of this information (other than to print it on the CE 150) you need a short program which will retrieve it from the Buffer and place it into your main program.

I have a requirement to read in data and transfer it to an array; the following routine is used:

```
41: DIM M$(200)*40, S$(0)*40
42: S$(0)=" ": F=STATUS 2
43: FOR N=0 TO 200
45: A=PEEK F: IF A=13 THEN 47
46: X$=CHR$ PEEK F: S$(0)=S$(0)+X$: F=F+1: GOTO 45
47: M$(N)=S$(0): F=F+1
48: S$(0)=" ": NEXT N
49: END
```

The information may be held in the EP44 memory and downloaded using TEXT key.

I use the above mainly for retrieving Machine Tool Computer programs and saving them on tape. Programs are dumped from the Machine Tool into the EP44 (or directly to the PC 1500). They can be modified at the EP44 before either sending them back to the Machine Tool or saving them via PC 1500.

b) PC 1500 in PRO/RUN mode:

Basic programs may be loaded or added to from the EP44. use:

```
SETDEV CI
CLOADa
```

The PC 1500 goes to BUSY

Lines of program can now be entered from the EP44 and sent to the PC 1500. A single CR will enter the line; and a further CR will end transmission.

Unfortunately the typed information is entered directly into the program and is not shown on the screen of either the EP44 or PC 1500. Nor can typing errors be directly corrected. The PC 1500 must be in PRO mode, and the editing done in the normal manner.

6. Notes

Using CLOAD"(filename)" causes ERROR 61 on the PC 1500, and the program held in memory is cleared. The filename needs to be in the correct form for the load to be successful. As yet I have not been able to load a program from EP44 memory into the PC 1500, but have succeeded using a Teletype ASR33. I have found no way of sending a line from the EP44, waiting, and then sending the next line. The information is sent in a continuous stream.

It would seem that the PC 1500 requires exact timing signals for receiving information - see the notes in the manual of the CE 158, page 32, para 19.

Further experiments will be made, and if they are successful the information will be published in this newsletter.

MINDBOGGLE CORNER

Much interest has been aroused by DAVID RIHOY's articles on ROM characters. This month's competition is for the most interesting, or useful, or imaginative character or set of characters, devised by the methods described. The figures should be sent as well as a printout of the characters. I hope to print the results in the Xmas number, so please could you rush your entries to me to arrive by November 24th. I know this does not give much time; but I know that quite a few of you have already had fun experimenting with the system.

The competition for the most exotic use of the PC 1500 was disappointing. Not one single entry! although I know that there are several readers who do make strange use of their computers in faraway places.

Nor have I yet received any entries for last month's competition for a game in 5 lines. I will extend the deadline for this also to November 24th. Please have a go! We surely don't want our Xmas number to appear with blank pages?

SUPERTEXT (B)

A very friendly Word-Processing program for PC 1500.
Designed for use with CE 161 and daisywheel or dot-matrix printer.

Limited use with CE 150 and 8K module.

Developed from original SUPERTEXT - almost all original commands retained.

CLEAR DETAILED INSTRUCTIONS - LLISTING - SUPPORT

A host of new features -
- and the old ones now streamlined -

Variable linewidth.
Automatic DIMensioning to memory.
Automatic location of RESERVE.
Automatic word-wrap.
Automatic space insertion.
Last Entry recall.
Last character deletion.
New Error-trapping.
Tabulation.
Quotation marks.
Apostrophes.
Non-keyboard characters.
Non-printing characters.
Justification.
Right justification.
Centralising.
De-justification.
Reformatting to different linewidth.
Line and paragraph deletion.
Line and paragraph insertion.
Re-arrange paragraph order.
2-stage optional editing.
Single-key space insertion.
Insert/delete/change words or characters.
Check actual linelength while editing.
'Margin' for Print Instruction Codes leaves text lines free for text.
Repeat facility for Print Instructions.
Straight or 'draft' printout.
Retains CE 150 horizontal printout.
Automatically selects TRAMSOFT TOOL-2 if attached.

103 Lines of text with CE 161 - with CE 150 and 8K from 13 to 28 lines.

£13.95 inclusive of post and packing in UK. (overseas: add £2 p. & p.)

Introductory offer: -
If you already possess original SUPERTEXT only pay the difference! Send £6 + p. & p. (UK £1 - overseas £2)

Please make cheques etc. payable to:- STATUS 1500 SOFTWARE

* STATUS 1500 is now produced entirely with SUPERTEXT (B). *