VOL.2
NO. 8

# STATUS 1500

SEPTEMBER
1984

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

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

## *ON TIME?*

*Miraculously, it seems that the lost weeks have been regained. Of course, a few days must elapse between writing this, and actually posting the edition, so I am keeping my fingers crossed. The reason for my obsession with the date of publication is of course the Xmas number. It would be optimistic to post it to Iceland or Ascension Island on the 23rd of December.*

* * * * *

*Apologies to any subscriber who has recently changed his address, and who has had the August number posted to the old address. A disaster to the mailing list has meant that I have been obliged to reconstruct it from earlier recordings. If I have your address wrongly recorded, please let me know.*

* * * * *

*'Lets Write a Program' takes a holiday this month. I have had no feedback from the unfinished program last month, so either it is perfectly comprehensible to everybody, or perfectly incomprehensible to everybody - or of no interest to anybody at all?*

* * * * *

*Two very clever entries have been received so far for the ONE-LINE competition (August 'Mindboggle') but there is still time for more. I look forward to receiving them.*

* * * * *

*The answers you receive to your problems in 'Signals' are often the result of logical deduction, without actual confirmation by experiment. Sometimes queries are formulated without the utmost precision, so that the question answered is not really the one you wanted to ask. It would be much appreciated if those whose queries are answered would confirm whether the proposed solution satisfactorily cures the real problem.*

*69 ON TIME?*
*70 SIGNALS*
*71 PEEK POKE & MEMORY - XVIII*
*72 ROM CHARACTERS - 2*
*74 SORT ROUTINES - 2*
*75 SELF-DELETE*
*75 IMPROVEMENTS*
*75 "CHILD'S PLAY"*
*76 "SWORD" reviewed*
*77 MARKETPLACE*

## SIGNALS

STEVE RICKABY has a modification to his Olivetti Praxis typewriter which allows him to use it as a printer. Via the RS 232 interface he is also able to use it as a keyboard for the PC 1500, but has some problems with this system. He is using the command INPUT$, and though careful to limit the strings to 80 bytes, when he uses statements such as SETDEV KI,PO: Z$(0)="": INPUT$ ""; Z$(0) the program crashes, and he gets ERROR 65.

I do not use this system myself, but the answer seems obvious. The INPUT$ command, dealing as it does with strings, must be subject to the same limitations as the response to an INPUT prompt. ERROR 65 is not one I have hitherto encountered, but it must be that the inverted commas are unacceptable to this command; or else they terminate the string and open a new one prematurely. Try using ZZ$=CHR$ 0 and INPUT$ CHR$ 0. This method should solve the problem. See this month's PEEK & POKE.

A.E.L.COX is puzzled by commands and statements such as RINKEY$, SETDEV, INSTAT, and so on, and while awaiting the cable which is to connect the PC 1500 to his EP44, would like some information as to the application of these commands.

These commands are in fact dealt with at great length in the handbook to the CE 158. From what you say, it sounds as if you are trying to connect the EP44 direct to the PC 1500 without an intervening interface. This is doomed to disappointment. You may have been misled by the fact that the EP44 contains its own RS 232 interface. But each device needs its own interface. This was clearly stated on page 61, vol.1. The best analogy is a telephone. You and I are devices for receiving, storing, and emittng information. To communicate over a distance we use an interface called a telephone. But if you telephone to me I cannot just plug the wires into my ears. I need an interface at my end too to re-interpret the signals that have travelled down the wires.

CRISTER SKOGLUND has very kindly sent me a copy of the German magazine CHIP in which the PC 1500 is well represented.

I was amused by the ALARM program for PC 1500 which occupies several pages of the magazine. It provides a fully programmable alarm system. A separate melody is sounded for each victim. In the example given, GERHARDT is woken at 6.31, whereas CORNELIA is permitted to sleep on until 6.37. And on Sundays they are allowed a whole hour extra! The program caters for up to 4 persons, each with their own tune. The program is over 5K long, and I did not feel inclined to spend a sleepless night keying it in, in order to wake early in the morning - but it is good to see that the PC 1500 is well supported somewhere, even if not in UK.

ROLAND SAAM, of MICROS FOR MANAGERS, asks me to inform readers that he is always interested in considering software for production and commercial distribution.

The most important factor is probably not the ingenuity with which your program is written, but its practical application for real use. You will find the address of MICROS FOR MANAGERS in the advertisement on the last page of this issue.

A number of readers are using the EP44 as a printer/terminal for their PC 1500. I would be grateful if ALL those who adopt this combination would send me all the information they can on their problems and successes in interfacing and running this system, in as much detail as possible, so that I can prepare one or two articles on the subject.

## PEEK POKE & MEMORY - XVIII

'"Hell!" said the duchess'

No other character on the keyboard has quite so many functions, and yet presents quite so many difficulties, as the quotation mark, or inverted commas. Consider the following. If you key Z$="" then Z$ will be CHR$ 0. i.e. nothing at all. If you key Z$=" " then Z$ will be a space, i.e. CHR$ 32. If you just key Z$=" then Z$ will still equal CHR$ 0. The only thing it never seems to equal is an inverted comma.

So why is it that when you key CHR$ 34, it never appears as such? Well of course you never did key CHR$ 34! You keyed CHR$ 18, and the system that monitors the Reserve interpreted your keystroke in the light of the adjacent ones.

Let us for the moment consider any other character. Take for instance *. (We have avoided using a character that can also be used as a variable). Now the character we have chosen can be used in a number of ways:

As part of a simple statement on the keyboard, i.e 5*7

As a program statement i.e. 10: N=5*7

As a string: A$="*" either from keyboard or as part of a program line.

In response to a prompt such as 10: INPUT "A$?";A$. When this line is RUN you will see on the screen the prompt A$? and you will reply by keying * (NOT "*").

By a statement in a program (or from the keyboard) A$=CHR$ 42 (But NOT in reply to an INPUT prompt. If you reply to the prompt A$? by A$=CHR$ 42 then A$ will read: A$=CHR$ 42.

By POKE instructions such as POKE 14539,42 (keyboard or program).

But you will have noticed that on certain occasions in the above list we used quotation marks to describe our instructions. We enclosed the contents of our string in quotation marks. So it is no use keying (or writing into a program: A$(0)=""Hell!" said the duchess"

A$(0) would equal CHR$ 0, as a result of the first pair of quotes. Leave out the first quote, and write: "Hell!" said the duchess"
Then A$(0) would just equal Hell.

So what must we do? If we use quotes from the keyboard (or in a program statement) they will ALWAYS execute their function of opening or closing a string. If this is wrong syntax in the context, then ERROR 1. But we are allowed to insert them into a string by character description. To enter our subtitle, as it stands, into a string we must write:

```
A$(0)=CHR$ 39+CHR$ 34+"Hell!"+CHR$ 34+" said the duchess"+CHR$39
```

(CHR$ 39 being the apostrophe, or single inverted comma. This is not supported by keyboard or CE 150, but is supported by most printers.. It appears on our display as the SPACE FOR INSERTION symbol).

The other limitation on the use of inverted commas is that they cannot be used in reply to an INPUT prompt. If you use them thus, you will get ERROR (normally ERROR 0). This is what has caused the problem described in 'Signals". It can often be obviated by character description i.e. A$=CHR$ 0, or A$=CHR$ 32, instead of "" or " " respectively.

Mention has already been made in previous issues of the fact that where a quotation mark terminates a program line then it may be omitted. Mention has also been made of the dangers of this practice: for instance, a line ending A$=" (meaning CHR$ 0) and a line ending A$=" (with a space after it, meaning CHR$ 32) would appear identical on the display.

Note also that if you enter what is normally a Reserved Word such as PRINT within inverted commas, it will not take on its Reserved Word condition as CHR$ 241 + CHR$ 151, but will be spelt out in full. Nor can you get round this by entering it first, and adding the inverted commas afterwards: you will just get ERROR 28.

## ROM CHARACTERS - 2 by David Rihoy

The table and diagram below will make it clearer how the central group of bits (see table 1, last month) control the directions of the lines drawn. Remember that the numbers in the table represent multiples of 8, which when added to the digits representing length, and to the 64's of the control number, will fully describe the line drawn. The numbers start with 0, in an easterly direction, and move anticlockwise.. All these angles are absolute, not relative, with one exception described below.

*Table 2*

| No. | Degrees | x | y |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 45 | 1 | 1 |
| 2 | 90 | 0 | 1 |
| 3 | 145 | -1 | 1 |
| 4 | 180 | -1 | 0 |
| 5 | 225 | -1 | -1 |
| 6 | 270 | 0 | -1 |
| 7 | 315 | 1 | -1 |

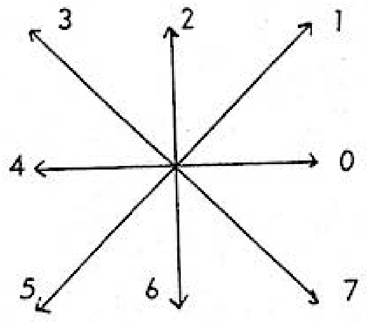

*Diagram 1*

The control number is (as is SHARP'S wont) not straightforward, and I do not pretend to fully understand all its implications. However the three main functions are fairly clear. They are to control the UP/DOWN movement of the pen and to stop the operation when the figure has been completed. The actual assignments are:

*Table 3*

| | |
|---|---|
| 0 = | PEN UP |
| 1 = | PEN DOWN |
| 2 = | ??? |
| 3 = | PEN DOWN and FINISH |

It is important to note that control 3 assumes that a normal character is being drawn i.e. one within the usual square. If this is not the case then be very careful; while experimenting I produced a yard of paper - YOU HAVE BEEN WARNED! What happens with control 2 depends upon the direction value, of which two only seem to have any effect; these are 0 and 4. The easiest way to describe in simple terms the line drawn is that it is like a boomerang with the 2nd arm longer than the 1st. To be precise: if the direction is 4 then one LINSEG is drawn at 45 degrees in a clockwise direction relative to the previous line, and then the number of LINSEGs specified by the length parameter again at 45 degrees clockwise relative to the last line drawn. Exactly the same process applies to direction value 0 except that the rotation is anti-clockwise. All this should become clearer if we work through an example. Let us take a comparatively easy one, say capital letter U.

If you peek at the 5 addresses starting at 41378 you will obtain the following numbers: 22,117,130,73,213. These are the numbers which describe the letter U. Using the rules above we can draw up the table with the diagram produced by the PC 1500 (see next page). If we take 22 and divide by 8 we get 2 remainder 6, divide 117 by 8 we get 14 remainder 5, dividing 14 by 8 we get 1 remainder 6, and so on.

Reading this table: the 1st line says "move the pen vertically 6 LINSEGs with the pen up"; the 2nd line says "pen on paper and move it vertically down 5 LINSEGs"; the 3rd line says "pen still on paper move 1 LINSEG 45 degrees anti-clockwise and a further 2 LINSEGs another 45 degrees"; the 4th line says "draw 1 LINSEG at 45 degrees to the horizontal"; the 5th line says "draw 5 LINSEGs vertically upwards and finish, by moving with pen up to the bottom righthand corner of the square".

You may like to experiment with letter O. This starts at 41342, and has 6 numbers.

continued overleaf >

You should first attach the PC 1500 to the CE 150 when conducting these experiments, since the ROM which holds the characters is actually inside the CE 150 itself. You may also like to note that the last line of any character is never a 'double-barrelled' one, because these all have control 2, whereas the last line of any character is always described by control 3.

Now here is the diagram of capital U and its analysis.

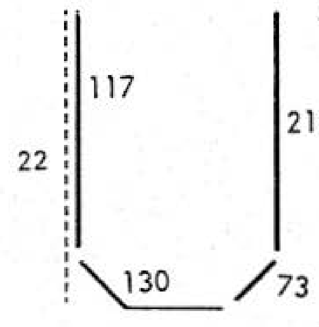

Diagram 2

Table 4

| | | Control (64's) | Direction (8's) | Length (1's) |
|---|---|---|---|---|
| 22 | = | 0 | 2 | 6 |
| 117 | = | 1 | 6 | 5 |
| 130 | = | 2 | 0 | 2 |
| 73 | = | 1 | 1 | 1 |
| 213 | = | 3 | 2 | 5 |

Where does all this get us? Well the m/c routine that LPRINTs starts by getting the addresses of each letter one by one as it LPRINTs them. This means that if we point instead to our characters and then call the routine that draws them, we can design whatever we want within the limits prescribed above. To achieve this we need to do four things: design our figure, convert this to a number sequence, store this in suitable locations, use some machine code to access them. The method used to design a figure is obviously up to the individual to choose but I would recomend using quarter-inch squared paper. If designing a character remember to draw it within a square of 6*6 units. However for other figures this need not be a limitation.

The next step is to assign numbers to the lines and this can be done as follows: find the shortest line (=1 LINSEG) and measure all others in terms of this one; draw up a table (similar to the one shown) and for each line in sequence enter the control, direction, and distance; then calculate the number required for each line by multiplying the control value by 64; the direction value by 8; add these two results, and add the distance value. We then have the required number for the line. Repeat this for each line, and the resulting sequence of numbers describes the character or figure.

Everybody will have there own likes and dislikes about where to store the number sequence, but I think the best place in in fixed memories X$, Y$ and Z$. Then various number sequences can be stored in DATA statements, and assembled into these memories via the CHR$ function. No POKEing at all is required.

A routine for this with capital letter U could be:

```
10: FOR N=1 TO 5: READ A: X$=X$+CHR$ A: NEXT A
20: DATA 22,117,130,73,213
```

The actual machinecode routine is quite short:

```
253,168,72,122,74,9,88,119,90,208,190,167,136,154
```

This routine is specific for X$, and would have to be altered for another location. Its function is to point to the address of X$, and then SJP to the standard subroutine in ROM which controls the drawing of characters.

[to be continued next month]

## SORTING ROUTINES - 2
## more 'Bubble Sorts'

by A.E.L.Cox

The very simple sort given last time can be speeded up by decreasing by one the length of each successive pass, so as to avoid testing strings which have already been moved to their final positions. The number of passes remains the same, but the number of tests is reduced.

**Bubble Sort, Diminishing Passes:**

*Routine 2.1*

```
100: FOR C=1 TO A-1
110: FOR B=1 TO A-C
120: IF LEFT$(A$(B),16) > LEFT$(A$(B+1),16) LET A$(0)=A$(B):
     A$(B)=A$(B+1): A$(B+1)=A$(0)
130: NEXT B: NEXT C: END
```

This routine sorted the same set of 101 character-strings as was sorted by the simple bubble. There was a 30% saving in time, because the number of times the inner loop was run was reduced from 10,000 to 5,050.

An alternative method is to arrange for each pass to terminate one place short of where the last exchange took place in the previous pass. This requires an extra statement in the inner loop, but gives a saving in the total number of tests.

**Bubble Sort, Last Exchange:**

*Routine 2.2*

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

Sorting is complete after a pass in which there are no exchanges. A$(0) is used as a flag. It is emptied at the beginning of the loop, but will be filled if an exchange takes place, thus indicating that sorting is not complete, and so the routine is performed again.
The same set of strings was sorted in about 98% of the time taken by the previous routine. There was a reduction of about 5% in the number of exchanges, balanced by the extra time in the inner loop.

When the character strings are already partially in order, this routine is usually much quicker, and never more than very slightly slower, than Routine (2.1) above.

It often happens in practice that a set of strings is entered and sorted and then edited. Then a few new strings are added, and so the set then needs re-sorting - but not much. This routine (2.2) would be very suitable except that the new items come in at the end: and this is the least convenient position. Because the routine works from the first data to the last, the new items must be moved against the flow, and can only get shifted by one position on each pass. But we can arrange for the passes to start at the other end and move towards the beginning of data, carrying the new items along. These new items will then need only one pass each to reach their proper positions.

**Bubble Sort, Last Exchange, Inverted**

*Routine 2.3*

```
100: D=2
110: A$(0)=CHR$ 0: FOR B=A TO D STEP -1
120: IF LEFT$(A$(B),16) < LEFT$(A$(B-1),16) LET A$(0)=A$(B):
     A$(B)=A$(B-1): A$(B-1)=A$(0): D=B
130: NEXT B: IF A$(0) THEN 110
140: END
```

The next article will deal with a less simple, but speedier system, known as the 'Shell' Sort.

## SELF-DELETION

Here is the self-deletion routine as suggested in PEEK & POKE last month. It has been pointed out to me that is not necessary to GOTO the line whose address is required. It is only necessary to RESTORE the line in question. The statement RESTORE only works in PROGRAM mode: but then from the keyboard we can always use LIST to find the address of a line.

The routine below will work equally well to delete a MERGEd program, or the last part of a continuous program.

[nnnn] represents the line number of the earliest line to be deleted.

```
998: RESTORE [nnnn]: P=PEEK 30887+256*PEEK 30886-4:
      P=P+(PEEK P=13): Q=INT (P/256): R=P-256*Q
999: POKE 30823,Q,R: POKE P,255
```

## IMPROVEMENTS

| | |
|---|---|
| **title:** | SCREEN-DUMP |
| **original:** | C.SIMPSON (vol.1, p.77) |
| **improved by:** | MIKE O'REGAN |
| **purpose:** | flexible graphic display |
| **comment:** | simpler than CAPSET |

```
10: " "CLEAR: DIM T$(0)*26
20: INPUT "SIZE 3 to 20 ",SI
25: INPUT "WIDTH 1 to 3 ",WI
30: INPUT "SLOPE (0 to 5) ",SL
35: IF (SI<3+SI>20)+(WI<1+WI>3)+(SL<1+SL>5) BEEP 5: GOTO 20
40: SP=SI+2+SL: WP=SI*.5*WI: WS=SP*.5*WI
45: INPUT "TEXT ", T$(0): CLS: PAUSE T$(0): GOSUB 110: GOTO 40
100: "A" GRAPH: GLCURSOR (200,0): SORGN
120: FOR K=0 TO 155
130: P=POINT K: IF P=0 THEN 170
140: FOR J=0 TO 7: DOT=2^J: GLCURSOR ((-J*SP+J*SL),(-K*WS+J*SL))
150: IF (P AND DOT)=DOT RLINE -(SI,WP),,,B
160: NEXT J
170: RETURN
```

## CHILD'S PLAY

This is the original program by FRANK ODDS with which his small daughter is kept amused. When the computer is switched on, a letter appears on the display, and must be matched by pressing the corresponding key. (See July edition, page 53).

```
 1: ARUN: RANDOM: WAIT 0
 2: A=RND 26: A=A+64
 3: CURSOR RND 26-1: PRINT CHR$ A
 4: IF INKEY$ = CHR$ 0 GOTO 4
 5: A$=INKEY$: IF ASC A$=A GOTO 7
 6: GOTO 4
 7: FOR K=1 TO 20
 8: L=RND 255: BEEP 1,L,(2000/L)+20
 9: NEXT K
10: CLS: GOTO 2
```

The cassette loaded without difficulty.

Having dealt with the virtues of this program, I will now examine it in greater detail. It is designed for use with RS 232 interface, or with CE 150. Text must be written into the program itself in the form of DATA statements, prefaced by inverted commas and line numbers, starting at 1000. The length of line as printed does not depend on the length of the program line, but on the insertion of certain symbols. Margins may be set, and full justification is supported.

Since text is written in program mode, the editing facilities are those normally available when writing programs. Tabulation is also supported, when in the special 'Heading' mode. However in order to tabulate, a special line of DATA instructions must be written, setting the tabs, and then a further bunch of symbols inserted in the text itself. Similarly, to insert in the text special characters, such as apostrophes or inverted commas, spaces must be left in the text for the characters, and then the DATA text line must be preceded by yet another DATA line (a separate one for each type of character) specifying which character is to go where. Being myself the author of a word-processing program, I may be suspected of prejudice. A few figures may dispel this illusion. In SWORD the insertion of a simple pair of inverted commas takes 27 extra keystrokes, as against 2 in my own program. The same applies to the tabulation. To write the heading of this article, the following lines of program would be necessary:

```
1000: DATA " $ 27 87 $
1010: DATA "$T 36 56 $
1020: DATA "$S 34 37 43 $
1030: DATA "#  SWORD WORD   PROCESSOR%
1040: DATA "## software review %%&$
```

I should add that the number of columns must be specified each time the text is printed, since this parameter is flexible, and does not depend on the length of text lines as written.

The above applies to use with RS 232 interface and daisywheel or dotmatrix printer. For use with CE 150, in CSIZE 1 you are limited to a line of exactly 31 characters. However there is an option for printing in CSIZE 2. Quite simply, LLIST the text you have written into the program. It will of course be printed complete with line numbers, DATA expressions, and control lines. The generosity with which this facility is offered is less than overwhelming.

The program is less than 2.5K in length. The suggested price is £19.00. The instructions are brief, but probably adequate. A full listing is provided, together with sample text.

The style in which the program is written combines complexity with crudity. No use is made of any of the more advanced techniques which have come to light over the past 2 years. Indeed program lines like the following (and there are many such):

```
385: IF S=1 THEN RETURN
```

will undoubtedly evoke the derision of any user who has owned his PC 1500 for more than a week, and knows that in this context the expression 'THEN' is entirely redundant.

This piece of software is shabbily presented, severely limited in application, ludicrously overpriced, and tediously clumsy to use. Of course, I may be prejudiced.

"SWORD" is written by P.Ashmole, and distributed by MICROS FOR MANAGERS.

## MARKETPLACE

Here are some of the addresses in Germany sent to me by CRISTER SKOGLUND. If any reader obtains any useful material or information from any of these, we shall all be grateful for details.

FISCHEL Gmbh
Kaiser-Friedrichstr. 54 A
D-1000 BERLIN 12

The above produces a newsletter containing much useful information, including some details of FORTH for the PC 1500.

JJ Taschencomputer Software
Jens Jurgens
Rohrteichstr. 66
D-4800 Bielefeld 1

has some very interesting machine-code programs.

Wolf-Gernot Heidel
Saarbrucker Str. 48
D-5600 WUPPERTAL 2

sells among other things a floppy-disc controller for PC 1500, and some software

Peter Littfinski
Grosser Reitweg 5 A
D-2080 PINNEBERG

has a number of interesting mathematical and scientific programs.

USER SYSTEMS
Soft & Hardware entwicklung Gmbh
Dorfweisen 2
D-7151 AFFALTERBACH

advertises a special Video-interface for PC 1500. It may however be still under development.

And finally, in Switzerland:-

Urs Ribi
im Melcher 8
Rumikon
CH-8352 Raterchen

has a machine-code program giving many extra commands, such as REPEAT...UNTIL; WHILE...WEND; GET; SWAP; SORT; FRAC; HEX$ and so on.

Christer Skoglund adds that the best programs he has used are the EASI- series from MINI-MICRO.

**********

Last month I announced the intention of keeping a register of equipment that subscribers wished to acquire, or dispose of. So far I have been offered 2 programmable calculators, and a derelict lawn-mower. My hopes of matching any of these with requirements is NIL.

May I make it clear that the offer is confined exclusively to the PC 1500 and its accessories, or at least potentially ancillary equipment!