VOL.2
NO. 5

STATUS 1500

JUNE
1984

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

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

DOOMED?

Although the PC 1500 is less than 3 years old, it seems to be on the verge of obsolence. Few shops stock it; those that have done so are selling remaining stocks off cheaply - sometimes at less than half-price. The PC 1500A is flourishing in USA. Will SHARP (UK) market it over here? I wish I knew: it is no use asking SHARP. (Indeed, if any subscriber remains on friendly terms with this weird organisation, would he try and find this out from them, and let me know?) The TANDY PC 2 is already, as we know, out of production. When the PC 1500 first appeared, it received more unqualified praise than any other computer before or since, and it is sad that it has languished in this country, mainly owing to the lack of support and competent marketing from SHARP (UK).

Meanwhile STATUS 1500 also is not without difficulties. For the first time, the steady stream of new subscribers has dwindled to a trickle. As time goes on, natural wastage will no longer be replaced, and the total number of subscribers will decrease. Fortunately, we have not reached this stage yet.

Interestingly, an early experiment with EASITREND predicted a peak about this time, but the forecast was rejected because it was not based on a sufficiently large sample of data. The newsletter started life with just 23 subscribers. It will, I hope, be a very long time until we are reduced below that level. Until that time, we shall do our best to carry on.

LAST MINUTE NEWS: A reliable source informs me that SHARP have just decided to market both the CE 161 and the PC 1500A in UK, and that these should be released here very shortly. It will be interesting to see the prices. SHARP also intend to renew their support for Pocket Computers with an intensive sales campaign. For my opinion of such a project, see the editorial in the July issue last year. I hope that this time I am wrong. Since the sales-manager in charge is more experienced, and less bombastic than his predecessors, perhaps there is a chance.

Gone, gone are the days when every post brought letters praising this newsletter. Yet it is (I think) as good as ever, if not better. Perhaps now you have all come to take its excellence for granted. However, June appears to be 'Have a Go at the Editor' month.

C.P.UNDERWOOD reproves me for not appreciating the suggestion last month from A.E.L.COX about dispensing with terminal quotes. He points out that this is a great timesaver when entering commands from the keyboard. He adds that leaving out terminal quotation marks in SUPERTEXT (B) would create space for 2 extra lines of text.

Alright! you win! But all the same, the system does give rise to the most terrible bugs sometimes. How would you distinguish between:

```
GOTO "A
```

and

```
GOTO "A
```

The former will run OK: the latter will give ERROR 11. The difference is completely invisible both on LLIST and LIST, and only by running the cursor over the line to the end of the second example can you detect the unwanted space after "A

In reply to this, A.E.L.COX asks why one should have a 2-chr. label with a space.

Why indeed should one have any bug or any error in a program which is being developed? But most of us do. Many of us are in the habit of deleting with a SPACE (1 keystroke) instead of SHIFT//DEL (2 keystrokes) while writing programs: and this is fine except during REMs or between inverted commas, when SHIFT//DEL must be used. The point is that the omission of the final quotation marks can allow an error to be invisible.

A.E.L.COX also points out an error on page 25, in the CLOCK/CALENDAR program. He says that line 70 should read:

```
70: IF F=1 IF T>13 LET T=T-12
```

Your line too is erroneous. Surely you mean:

```
IF F=1 IF T>=13 LET T=T-12
```

ALLAN THOMAS chides me for an error in the variable-changing program (see page 23 and page 35). He says that no provision is made for protecting HEX numbers from being disastrously changed.

Quite right. I never use HEX except in the preparation of m/c routines, at the beginning of a program, and would never expose this section to automatic variable changing. If you introduce the protection you suggest, the routine will run even more slowly. For some purposes it is easier, not to have a compendious standard utility, but just to 'run something up' for the occasion. The original article was intended to illustrate some of the snags in doing so.

J.K.GAUTON has had some teething troubles with the TRAMSOFT TOOL-2. He has cured them by allowing the tape to run an inch or two before FSAVE/FLOAD and again when the operation ends.

I have not found this necessary. It does sound as if you may have some trouble with the motor of your cassette-recorder. Another suggestion, that we should standardise our equipment, is idealistic rather than practical. Some readers have no trouble at all with non-standard equipment: others have trouble with standard equipment. All these consumer products suffer variously from wear-and-tear: heads go out of alignment: motors slow up: even the electronic bits can deteriorate.

PEEK POKE & MEMORY - XV

- on error carry on! -

One of the most powerful tools in your repertoire is ON ERROR GOTO..... For instance, on first using a program, it may be necessary to DIMension: on further use the DIM instruction would create ERROR, unless reDIMensioned, which would destroy the contents of the variables. Again, you may have written a routine into the program, which on further development you would wish also to use a subroutine: if you insert RETURN for subroutine purposes then you create ERROR on the initial RUN. But you can use ON ERROR GOTO to hop over the DIM statement, or over RETURN. Nevertheless the use of this expression can be tricky. It remains in force until cancelled by another ON ERROR expression. Otherwise you could find that a further error will return you to a totally irrelevant part of the program. If immediately after the ON ERROR expression you had a number of branches, then your program could become littered with cancellations. Of course you can always rewrite the program completely - if you have time! If your lines are rigidly numbered at regular intervals, perhaps something could be done like:
1: ON ERROR GOTO 100 § 100: GOTO STATUS 4+10 - but even this can create problems, since STATUS 4 will represent the line before the line where the error occurred, if the error commenced the line; whereas if a statement in the line has been accepted, then STATUS 4 will represent the error line itself. **The following routine will always just pass to the next line after the error line, however irregularly your lines are numbered.** Lines 101 and 105 are optional.

```
10: ON ERROR GOTO 100
::::::::::::::::::::::::::
100: Q=256*PEEK 30898+PEEK 30899
101: L=256*PEEK 30900+PEEK 30901
105: BEEP 1: PAUSE "ERROR";PEEK 30875;" IN";L
110: IF PEEK Q<>13 LET Q=Q+1: GOTO 110
120: R=256*PEEK (Q+1)+PEEK (Q+2): GOTO R
```

How it works: PEEK 30900 and 30901 give (in 256ary) the line number which contained the error. PEEK 30898 AND 30899 give the actual address where the error occurred. Note that normally this address is usually the last memory location occupied by the unacceptable statement. PEEK 30875 gives the type of error - ERN (wrongly given in the Technical Reference Manual as ERL, which means 'Error Line'). The routine, advised of an error, finds the error address, PEEKs along the line until End-of-Line marker 13, reads the next line number (from the two addresses following EOL), and goes to that line.

You may like to experiment with the routine below. Key it in while the computer is attached to the CE 150, then separate them, and RUN the routine. Note that ERROR 1 (syntax error) will NOT be overcome by any ON ERROR GOTO instruction. And (in case you have forgotten) an ON ERROR GOTO (n) can be cancelled by ON ERROR GOTO 0.

```
10: ON ERROR GOTO 100
17: GRAPH
19: A$(0)=1
22: DIM A$(10): DIM A$(11)
26: READ A$
27: RETURN
38: DATA 1,2,3
54: READ A$
63: GOTO 60000
74: GOTO 700000
99: BEEP 1,10,3000: PRINT "ERRORS OVERCOME!": END
```

- and then lines 100 to 120 from the routine above.

LETS WRITE A PROGRAM - V

Last month we ended by suggesting that a chart should be made of the variables used. If you did so, you would immediately have spotted an anomaly: the use of A(F) in line 3020, which appears as A(3,7) in line 1000. So alter line 3020 to A(1,F). I have been disappointed that I have not received a single letter pointing out this error: it seems that you are not taking a very active interest in this series.

The A(3,F) is so that the effect of bets, although calculated before the spin of the wheel is displayed, may nevertheless be held 'in escrow' until all bets are completed. A(1,F) for original capital. A(2,F) for the effect of staking. A(3,F) for the capital after adding or deducting A(2,F). And while we are about it, lets do the same for the bank. Otherwise a player's bets could 'break the bank' before the wheel is visibly spun: painful explanations might be called for. So in line 3040 alter variable B to A(1,0). We will keep A(n,0) for the bank.

We ought to be looking back at the scheme we drew up. But lets do a bit more coding. Have a go at RED OR BLACK. This subroutine will be used by 2 sections. It will be used with the calculation and display of the winning number. It may also be used by the graphics section, if we do a layout of the table. So we will put it outside the flow as designed. We will put it at 40000. Lets be a bit clever here. We can start at 40000 by calculating also ODD OR EVEN, and COLUMNS and DOZENS. So for betting purposes we will enter the subroutine at 40000. For the graphics we can enter with RED OR BLACK at 41000.

Now from 1 to 10 and from 19 to 28 all odd numbers are red, and even numbers black. For the remaining numbers the reverse is the case. We could deal with this by 37 IF statements. (Don't forget about ZERO). We could more economically deal with half the numbers, and let the rest have the opposite colour by default. But again we can try and be clever. Lets establish 2 conditions: a) within 1-10 or 19-28 and b) odd or even. (In fact we will already have had to establish the latter!). Surprisingly, we can also establish our colour without using a single IF statement, and compress this problem to a mere 72 bytes: it would go in a single line! But we will spread it out a little for the sake of clarity. Lets try it and see if it works.

```
40499: FOR W=1 TO 36
40500: "odd/even" E=(W/2=INT(W/2))
41000: "color" D=(W<11 OR (W>18 AND W<29))
41010: C=3*(D=E) § or is it 41010: C=3*(D<>E)  ?
41020: COLOR C
41029: LPRINT W;
41039: NEXT W
```

Test this: it could need changing a bit. When you have tested it, the lines 40499, 41029, 41039 can be deleted (though we shall need something like them when we come to do the graphics). And add:

```
41050: C$="BLACK": IF C=3 LET C$="RED": RETURN
```

We could probably even sort out this one without an IF statement: but here the process would waste space, not save it.

That's enough, I think, for this month. Those of you who are really following this series may like to have a go at working out the columns and dozens. I don't propose to go into too much detail over these. But before we forget, ZERO!

```
40000: IF W=0 LET C=2: C$="0": RETURN
```

Next we will have a go at the really tough part: "PLACE YOUR BETS!"

1) Make sure ROW I of RESERVE keys is operative.

2) To start, key DEF [SPACE]

3) When setting up a new database, answer the first prompt with "YES" written in full. This will now CLEAR all memories and set DIMensions. If a database is already in memory, press any key to continue.

4) You will now see the MENU displayed. It is used as follows:

key 1 - ENTer - used to INPUT data (UPPER CASE ONLY, please). The stars indicate the acceptable length of each ALPHA field. To return to the MENU simply press ENTER key when prompted for TITLE.

key 2 - AMD - (amend) - When pressed the prompt "Number" is displayed. Key in the Record Number you wish to deal with. If you do not know this, you can interrogate the file by PRT or SCH. You may amend any field by choosing T,C,S, or P, and then the old contents will be displayed on the left of the screen, and your new contents on the right. Return to the MENU by pressing ENTER key **without** an entry, at the 2nd prompt.

key 3 - ORDer - This key is used to re-arrange the order of records in accordance with the alphabetic order of the chosen ALPHA field.

key 4 - SCH - (search) - This key causes the prompt "Search String" You may enter a string of **any** length, (UPPER CASE, please), and the program will search **all fields** for a match. For example, if one title was "KILLER GORILLA" and another had ACORN as the source, then if "OR" was your response to "Search String" , KILLER GORILLA and ACORN would both be selected. The selected files can then be printed (Screen or Printer) showing record numbers and details from the main file.

key 5 - PRT - (print) - allows printing of the whole file on CE 150 in CSIZE 2. The PRICE fields are also totalled, and the total is printed. Finally, a date/time/group unique reference is also printed.
If the SCREEN option is selected, a prompt asks you for "speed". This is a WAIT time: 50 is about 1 second. If you answer by speed 0, you must step through the file by ENTER key. Since the size of the display is so limited, you must also select ONE field only. The Record Number, Title, and the selected additional field (Category, Source, Price) will be displayed. These limits on Screen Display also apply to displays generated by SCH (key 4 - search).

key 6 - TPE - (tape) - allows the file to be saved to or loaded from tape.

NOTES : UPPER CASE must be used throughout. ERROR trapping has been omitted to avoid lengthening the program unduly. ERRORS will be caused if Row I of the Reserve Keys is not selected. You can recover from reported ERRORs by pressing CL or BREAK key, without losing data,and re-enter the program either from the MENU, or by keying DEF [SPACE]

This program was created to catalogue a collection of software for the BBC computer, but can easily be adapted for any other small database.

```
  5 " "R=STATUS 2-STATUS 1-111
 10 POKE R,1,241,146,34,90,34,64,2,241,146,34,65,34,64,3,241,146,34,83,34,64
 20 POKE R+21,4,241,146,34,76,34,64,5,241,146,34,68,34,64,6,241,146,34,77,34,64
 25 BEEP 2:INPUT "CLEAR THE FILE-SURE???";Q$
 26 IF Q$<>"YES"THEN 40
 30 CLEAR :DIM T$(200)*14,C$(200)*2,S$(200)*5,P(200),CT$(0)*5,ST$(0)*5:K=1
 40 WAIT :PAUSE "SELECT FUNCTION KEY":PRINT " ENT AMD ORD SCH PRT TPE"
 50 "Z"INPUT " ************** TITLE      ",T$(K)
 60 IF T$(K)=""THEN 40
 70 INPUT " ** Category                 ",C$(K)
 80 INPUT " ***** Source               ",S$(K)
 90 INPUT "Price ";P(K):K=K+1
100 GOTO "Z"
110 L=0
120 FOR Z=1TO K-1
130 IF T$(Z)<T$(Z-1)GOSUB 290
140 NEXT Z
150 IF L=1THEN 110
160 BEEP 5:GOTO 40
170 L=0
180 FOR Z=1TO K-1
190 IF C$(Z)<C$(Z-1)GOSUB 290
200 NEXT Z
210 IF L=1THEN 170
220 BEEP 5:GOTO 40
230 L=0
240 FOR Z=1TO K-1
250 IF S$(Z)<S$(Z-1)GOSUB 290
260 NEXT Z
270 IF L=1THEN 230
280 BEEP 5:GOTO 40
290 CLS :PAUSE "..Wait for pips, please":TT$=T$(Z):T$(Z)=T$(Z-1):T$(Z-1)=TT$:L=1
300 CT$(0)=C$(Z):C$(Z)=C$(Z-1):C$(Z-1)=CT$(0)
310 ST$(0)=S$(Z):S$(Z)=S$(Z-1):S$(Z-1)=ST$(0)
320 PT=P(Z):P(Z)=P(Z-1):P(Z-1)=PT
330 RETURN
```

```
335 "D"INPUT "SCREEN or PRINTER-S or P",Q$
339 IF Q$="S"THEN 420
340 CSIZE 2:FOR M=1TO K-1
345 IF FF<>1THEN 351
346 FOR Y=1TO 14:IF SS$=MID$ (T$(M),Y,LEN SS$)THEN 351
347 IF SS$=MID$ (C$(M),Y,LEN SS$)THEN 351
348 IF SS$=MID$ (S$(M),Y,LEN SS$)THEN 351
349 NEXT Y
350 GOTO 360
351 USING "###":LPRINT M;TAB (4);T$(M):LF 1
360 NEXT M:LF 3
370 USING "#####.##":FOR M=1TO K-1
371 IF FF<>1THEN 380
372 FOR Y=1TO 14:IF SS$=MID$ (T$(M),Y,LEN SS$)THEN 380
373 IF SS$=MID$ (C$(M),Y,LEN SS$)THEN 380
374 IF SS$=MID$ (S$(M),Y,LEN SS$)THEN 380
375 NEXT Y
379 GOTO 390
380 LPRINT C$(M);TAB (3);S$(M);TAB (8);P(M):TQ=TQ+P(M):LF 1
390 NEXT M:LF 3
400 LPRINT "TOTAL ";TAB (8);TQ:LF 1:TQ=0
410 CSIZE 1:USING :LPRINT TIME :LF 6:FF=0:GOTO 40
420 GOSUB 700:FOR M=1TO K-1
421 IF FF<>1THEN 428
422 FOR Y=1TO 14:IF SS$=MID$ (T$(M),Y,LEN SS$)THEN 428
423 IF SS$=MID$ (C$(M),Y,LEN SS$)THEN 428
424 IF SS$=MID$ (S$(M),Y,LEN SS$)THEN 428
425 NEXT Y
426 NEXT M:FF=0:GOTO 40
427 GOTO 429
428 GOSUB 800:PRINT M;" ";T$(M);" ";@$(V)
429 NEXT M:FF=0:GOTO 40
430 "M"INPUT "SAVE or LOAD (S/L) ";Q$
440 IF Q$="L"THEN 480
450 PRINT "TAPE to RECORD-ENTER";
460 PRINT #K,T$(*),C$(*),S$(*),P(*)
470 BEEP 1:PAUSE "RECORDED":GOTO 40
480 PRINT "TAPE to PLAY-ENTER";
490 INPUT #K,T$(*),C$(*),S$(*),P(*)
500 BEEP 1:PAUSE "LOADED":GOTO 40
510 "S"INPUT "ORDER BY (T,C or S) ";Q$
520 IF Q$="T"THEN 120
530 IF Q$="C"THEN 170
540 IF Q$="S"THEN 230
550 GOTO 40
560 "A"INPUT "Number? ";NB
570 CLS :D$="":INPUT "T,C,S or P? ";D$
580 IF D$="T"PAUSE T$(NB);:INPUT " New ";T$(NB):CLS :GOTO 570
590 IF D$="C"PAUSE C$(NB);:INPUT " New ";C$(NB):CLS :GOTO 570
600 IF D$="S"PAUSE S$(NB);:INPUT " New ";S$(NB):CLS :GOTO 570
610 IF D$="P"PAUSE P(NB);:INPUT " New ";P(NB):GOTO 570
620 GOTO 40
630 "L"INPUT "Search string ";SS$
640 FF=1:TQ=0:GOTO "D"
700 INPUT "speed? ";J:WAIT J:IF J=0WAIT
710 INPUT "field?(C/S/P) ";V$
720 V=2*(V$="C")+3*(V$="S")+4*(V$="P")
730 RETURN
800 B$=C$(M):C$=S$(M):D$=STR$ P(M)
810 RETURN
```

MINDBOGGLE CORNER

Only a couple of quickies this month. No prizes.

How good are your abbreviations? You know them all? Are these valid? And what do they stand for? Try them and see.

LI. W. SI. S. LE. P. PE. SG. T. G. TE. A. GR. GRA. TR. O.

Next question: you are running a program. At a certain stage you get ERROR 1. And yet there is nothing wrong with the program! How could this happen? The answer is elsewhere in this issue.

INPUT

Notes from David Rihoy, A.E.L.Cox, Mike O'Regan, & C.P.Underwood

There are a number of valid syntaxes for INPUT statements; they may not all be well known. Below is a list of the main forms:

```
INPUT A
INPUT A,B,C
INPUT "prompt",A
INPUT "prompt";A
INPUT "prompt";A,B,C
INPUT "prompt",A,B,C
INPUT "prompt a";A,"prompt b";B,"prompt c";C
INPUT "prompt x",A,"prompt y",B,"prompt z",C
```

The forms are equally valid for A$/B$/C$. Where a direct sequence of variables is separated by commas, these cannot be replaced by semicolons. The important difference between the use of a comma after a prompt, is that the prompt disappears while an entry is written, whereas with the semicolon the prompt and entry are both displayed. It will be seen that the repetitions of INPUT are unnecessary in the form:

```
INPUT "prompt a";A: INPUT "prompt b";B: INPUT "prompt c";C
```

With the multiple entries, you must be careful not to miss an entry. If you press the ENTER key without replying to the prompt, the later prompts, and other statements on the line, will be skipped. However, a little ingenuity can turn this limitation into an asset.

```
99: INPUT "name";A$(J),"address";B$(J),"number";C(J): J=J+1: GOTO 99
```

will demand consecutive entries. To escape from the sequence, and continue the program, merely press the ENTER key without an entry. The following example permits the retrieval of information without disturbing the sequence:

```
10: INPUT "next entry";A$(N): N=N+1: GOTO 30
20: PAUSE "N=";N: GOTO 10
30: GOTO 10
```

Here the number of entries reached so far may be interpolated in the sequence, by pressing the ENTER key without an entry. Note that the previous entry will not be destroyed. In fact, on INPUT X or INPUT X$, if the ENTER key is pressed without an entry being made, then X or X$ will retain its previous value.

Another useful technique is based on the length of the prompt. If the prompt is 27 characters long, (and the first 5 characters are SPACEs or MINUS SIGNS, etc) then the entry will appear at the beginning, not the end, of the prompt. For examples see lines 50 to 80 of PICOBASE, on page 43, where the technique is used as a guide to the number of characters permitted in each type of entry. It is of course essential that the length of the prompt be exact.

It is often desirable to have changing information preceding a constant prompt, as in the following example:

```
10: DIM A$(10): WAIT 0
20: FOR N=1 TO 10
30: PRINT "NUMBER";N;: INPUT " :your entry";A$(N)
40: CLS: NEXT N
```

Finally, a curious case of ERROR 1. This normally indicates a syntax error in the program itself. However if you reply to a prompt for a numeric entry with something that cannot have a value (e.g. a sign or puctuation mark) then ERROR 1 will appear. Pressing the red CL key will remove the ERROR and restore the prompt.

IMPROVEMENTS

Here are the first of the long-promised 'improvements' to programs previously published. Not all the improvers can be identified, because of their reluctance to put their names on their programs. The routines are offered for your consideration: they are not necessarily recommended, nor have they all been exhaustively tested.

title: GOLF
original: RJC (vol.1, p.113)
improved by: ?????
purpose: to indicate definitively whether ball has been holed.
comment: may not coincide with visual, owing to 'graphic slip'.

```
 255: GOSUB 7000
 500: "A" GLCURSOR (T,-TT):C=0:H0=0
7000: H0=H0+1: XC=PEEK &79E1: IF PEEK &79E0 LET XC=XC-256
7005: YC=PEEK &79E3: IF PEEK &79E2 LET YC=YC+256
7010: IF ABS (XC-(H1+3.5))<3 AND ABS (YC-(H2+3.5))<3 BEEP 5: GLCURSOR
      (0,-TT-50*B-50): GOTO 7020
7015: RETURN
7020: WAIT 10: PRINT "HOLE IN";H0: LPRINT "HOLE IN";H0: GLCURSOR
      (0,-TT-50*B-200): END
```

title: SCREEN-DUMP
original: C.SIMPSON (vol.1, p.77)
improved by: H.H.HEINE
purpose: choose size of print-out.
execute by: DEF SPACE

```
  10: " "CLEAR: S=STATUS 2: FOR I=S TO S+155: POKE I,POINT (I-S): IF
      POINT (I-S) LET T=I
  20: NEXT I: T=T-6: WAIT 0: REM cursor suppression
  30: "SZ" CLS: INPUT "which size 2-28 ? "; SZ: IF SZ<2 OR SZ>28 GOTO "SZ"
  40: FOR I=S TO T: GPRINT PEEK I: NEXT I: GRAPH: GLCURSOR (110+SZ*3.5,0):
        SORGN: FOR I=0 TO T-S: K=0: GLCURSOR (0,-SZ): SORGN
  50: FOR Y=0 TO 6: H=2^Y: IF H AND POINT I LINE (K,-SZ)-(K-SZ,-2*SZ),,,B
  60: K=K-SZ-1: NEXT Y: NEXT I: WAIT: TEXT: LF 7: BEEP 1,100,100: END
```

subroutine 3 — PASSWORD — by Pete Eldridge

This routine is extracted from the cassette-indexing program mentioned on page 3. Unless the PASSWORD is entered correctly it is impossible to proceed with the program. C$ in line 1 should be written into the program, and may be any word up to 16 letters (upper-case only). When the password is entered correctly, the time and date are displayed. To continue, press first the red CL key, and then the RCL key.

```
 1: ARUN: CLEAR: LOCK: WAIT 0: POKE# 61453,128: C$="PASSWORD":  FOR P=1
    TO LEN C$
 2: PRINT" Password >";B$;"<"P: A$=INKEY$: IF A$=""GOTO 2
 3: BEEP 1: B$=B$+A$: NEXT P: IFB$=C$ THEN 5
 4: GOTO 1
 5: T=TIME, A=INT (T/100), B=T-(A*100), C=INT (A/100), D=A-C*100
 6: PRINT "ready";D;C;" 84";USING "#####.####";B: USING: A$=INKEY$:
    IF A$="" GOTO 5
 7: IF ASC A$=24 GOTO 10
 8: GOTO 5
 9: END
10: PRINT CHR$ 0: IF ASC INKEY$ <>25 GOTO 10
11: [proceed with main program]
```

PROBE! software review

This fascinating program is a disassembler. It is enthusiastically recommended to anyone delving into machine-code. I have encountered several disassemblers: this is certainly the most helpful and user-friendly. It is also affordably priced.

You answer the prompt with a Start Address. You are also asked to give information about flipflop settings, and which memory bank (ME0 or ME1) you wish to explore. The instructions make entirely clear how you should answer these prompts. The CE 150 prints out a record of your settings: and then PEEKs successively at memory locations from the start address onwards, translating the code in each location into the correct Assembly Language code. Cleverly, where a code is followed by data (addresses, etc.) the data is given numerically. But you must be careful to choose a valid Start Address. If you pick an invalid one, you could find data printed as code! In any case, you will have printed out a totally spurious routine. Memory locations are printed for the Start Address, and then every 16 locations. In the case of jumps, the address jumped to is also given.

Because this is the marvellous advantage of this particular dissassembler. The ROM of the PC 1500 is convoluted, and there are many vector jumps. With PROBE you may instruct the program to actually follow these jumps, and print out the ROM routine in the order in which it works! You can always disregard this option if you prefer.

It can of course also be used on user m/c routines. It will make the printing of readers' m/c routines much easier: I shall no longer have to worry so much about explaining them: you can just use PROBE on them to find out how they work.

To get the best out of PROBE the Technical Reference Manual is essential. This contains most of the Start Addresses for subroutines in ROM. I also found helpful page 61 and page 72 of STATUS 1500 (vol.1) to refresh my memory as to the meaning of certain codes. Otherwise, the 4 pages of instructions, clearly printed and laid out, which accompany the cassette, are all that is required. The program is easy to use; though to make use of its output requires intelligence and concentration. Fortunately, an appendix to the instructions explains in greater detail the difference between conditional branches (which are not followed) and conditional jumps (which may be followed optionally). It is not the duty of a program such as PROBE to give lessons in what machine-code routines are all about - but they may have been welcome all the same! Quite a lot of helpful backgound is indeed given. However I was slightly puzzled by one or two expressions, although their general meaning was apparent.

I have used the program successfully to disassemble the CE 158. I have also used it to attempt to disassemble the TRAMSOFT TOOL-2, without success. The code of this module is accessible to PEEKing: but gives ERROR messages to PROBE. I wonder why? PROBE will give such a message when a Start Address is given containing CHR$ 255. A facility which searched for the next valid address could have been helpful. But the program is already 9K long, and any elaboration would hardly leave room for any routines to be inserted for examination. Apart from a preliminary routine in m/c, PROBE is in BASIC. The program is ingenious, but quite transparent: so it would not be difficult to add minor tweaks of your own. If you should do so, avoid the use of variables P$, Q$, & R$, since their locations are occupied by PROBE's own machine-code routine.

PROBE! is by F.C.ODDS, 20 St.Philips Road, Leicester, at £7.50

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). ***