

VOL. 1  
NO. 9

# STATUS 1500

OCTOBER  
1983

\* 12 issues £10.50 (UK) \* overseas £14.50 \*  
Published Monthly

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

## TIME

Learning takes time. Learning about computers takes a lot of time. Theoretical solutions must be laboriously keyed in, debugged, amended. So often readers say "I just don't have the time...". There are no short cuts, no easy ways of learning how to make best use of your computer. This is particularly true of the PC 1500, for which so little software exists. The younger generation, who are taught the "new mathematics" right from the earliest level, find computing easy. Others find the necessary concepts harder to acquire: and as one grows up one finds other demands on one's time, such as eating, sleeping, watching television, and putting new washers on taps. Make time for your computer if you can. Time spent working with it is a good investment.

\*\*\*\*\*

On page 84 we print an interesting letter from a reader who has just returned from the Far East. Important conclusions may be deduced from it. It would seem that, despite the introduction of new models, SHARP are not in fact abandoning interest in the PC 1500 (as many of us have feared). One may reasonably expect it to remain current for some years to come. The causes of its comparative lack of popularity in this country must lie partly in the availability and popularity of many machines with big-screen access, and partly in SHARP (U.K.)'s complacently inadequate marketing and support.

\*\*\*\*\*

I was disgusted to receive, with an application from a prospective subscriber, the heading torn from the cover of the AUGUST issue. The newsletter is designed so that it may be assembled in a binder for reference. Much of the contents (depending on individual needs) will be more useful for reference, or for future use, than for immediate execution. To this end, an index will be published with the last issue of each volume. If you do not bother to preserve each issue of the newsletter as it appears, you are wasting your money and my time.

\*\*\*\*\*

## CONTENTS

\*\*\*\*\*

83 TIME  
84 Letter to the Editor  
85 SIGNALS  
86 PEEK, POKE & MEMORY - IX  
86 New INSTANT RESCUE  
87 RENUMBER (2)  
87 DATA

88 INSTANT SUPERSALVAGE  
88 MINDBOGGLE CORNER  
89 EASICASH reviewed  
90 "SCREEN INVERSION"  
92 Screen Inversion -demo program  
92 INTERFACE - [2]  
93 MARKETPLACE

LETTER  
TO THE  
EDITOR

Dear Mr. Cohen,

I'm just back from a week in Tokyo and Hong Kong, where there were many new PC 1500-related goodies on display. The CE 157 is one accessory that won't go on sale in the U.K. - it's a 4k RAM (I suspect it must be really a ROM!) that plugs into the memory extension bay and converts the machine to display the Japanese characters that are printed adjacent to the English letter keys on Japanese versions of the PC 1500.

Just out is the CE 161. This is described as a 16k "Program module". In fact it is a nonvolatile 16k memory expansion. It sold for about £150 in Tokyo, £100 in Hong Kong. (In the U.K. we seem to get Tokyo prices!) Incidentally, page 95 in the Sharp Technical Manual mentions the CE 160, as you've probably already noticed. This is clearly intended to be yet another RAM extension, but it doesn't exist in Japan - I suspect the CE 161 may be the physical realization of the originally planned CE 160.

The other interesting machine I saw on display was the PC 1501. This is

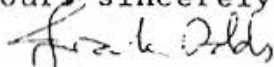
identical to the PC 1500 but it has 8.5k user RAM wired in instead of the 3.5k of the PC 1500. It accepts all the PC 1500 expansion modules, so it must be capable of legitimate (i.e. "Sharp approved") expansion to 24.5k RAM with the CE 161.

The "system briefcase" was often on display in department stores. It is very large, because it's designed to carry the CE 153 laid flat alongside the PC 1500/CE 150 combination. At a price in the region of £60 I can understand the decision of Sharp U.K. (taken some time ago, according to one of their reps) not to market the briefcase in England after all.

It was a pleasure to see our computer and its accessories so widely displayed in shops and department stores throughout Tokyo. In Britain it seems to be a very rare sight. There's no shortage of competition. NEC have a good-looking pocket micro and an excellent miniature (portable) dedicated word-processor. I saw at least 2 other new models of "briefcase" computer, though none had printers to compare with the CE 150.

Two Japanese peculiarities in the microelectronics scene: calculators from Canon with vertical LCD panels to display exclusively Japanese script, and a calculator/abacus combined unit from Sharp!

Keep up the good work with Status 1500! Yours sincerely,



Frank Odds



## SIGNALS

FRANCIS AKINWUNMI has been trying to obtain TANDY's Service Manual, so far without success.

*This manual is not normally available to the public. If you write to:-  
TANDY CORPORATION(UK)// NATIONAL PARTS // BILSTON ROAD // WEDNESBURY //  
WEST MIDLANDS // WS10 7JN they may perhaps be able to help you.*

J.K.GAUTON has acquired the DBASE database program. He wonders why it should not be executed by a DEF instruction rather than the suggested GOTO 1.

*The only effective difference is that GOTO clears the display, whereas DEF does not. See page 132 of SHARP's Instruction Manual for the different effects of various methods of initiating execution. However it is possible that even the addition of the label for DEF initiation may eat into the space required for DIMensioned variables. You should check carefully before altering the program, even as slightly as this.*

L.R.BIRD of AZTEC INTERNATIONAL writes from Thailand to say that he has developed an 8K program concerned with drilling engineering, covering Well Cementing, Bit Hydraulics, and Kick Control, which he would be willing to make available to other readers involved with this limited field. *Thank you!*

JOHN MACK suggests I should reconsider my policy of not printing programs which are only of use to one or two specialists, on the grounds that the programming techniques used may be of general interest.

*No.*

Several readers have expressed interest in SIMON COX's 96K extension.

*It is not the policy of this newsletter to publish the names and addresses of subscribers. If any of you wish to communicate with other readers, all you have to do is to write to the Editor, enclosing a letter to the subscriber in question, together with an adequately stamped envelope, which will be addressed and forwarded. If this is too much trouble, you can hardly expect the subscriber (or myself) to take the trouble to reply to you.*

G.ALLEN, who is new to computing, enquires whether there is a simple way of using "whichever is the greater" of two variables without a lot of IF statements.

$(A*(A>B)+B*(B>A))$

IAN TRAYNOR would like to know the location of the GOSUB stack, in connection with a program he is writing, which will make use of novel techniques.

*The GOSUB pointer is situated at &7891, and the stack, which is also used for FOR-NEXT loops, etc., extends from &7638 to &7AFF. We have been somewhat dilatory in printing details of the convoluted Memory Map, and will try to rectify this in forthcoming issues.*

JOHN BLAND has been investigating the possibility of speeding up the CSAVE and CLOAD facilities. The TECHNICAL REFERENCE MANUAL states that this is possible, and gives much necessary information - but, of course, does not actually say how to do it. He has found that developing a program for the purpose is by no means as simple as the manual would suggest.

*Has any other reader been working on this problem?*

If you and I had been born with 8 fingers on each hand, how much simpler computing would be! But we are biologically condemned to find a decimal system natural, and hex unnatural. One may calculate in hex - with effort - but it is decimal in which one thinks. BASIC is partly to blame, since it relates to ordinary language. I sometimes think that computing would be easier in the long run, if high level languages had never been invented, and we had been forced to learn computing the hard way. Similarly, if hex consisted entirely of alphabetic symbols, instead of a mixture of such symbols and ordinary numbers, it might be less confusing. It is hard work learning something totally new, but if instead one merely *extends* what one knows already, one tends whenever possible to revert to the familiar. Thus getting nowhere.

The snootier experts treat hex as familiar. Distasteful as it is to laymen like ourselves, it really is worthwhile coming to terms with it. For instance, take all this 256ary. In hex it becomes natural. Take a line number such as 1000. This, as we know, is held in 256ary in 2 bytes; showing up as 3 and 232. In hex this would be 03, and E8 (and this is what the bytes really contain: the decimal display is a translation).

Look again : 03, E8 . Put them together. 03E8 is simply decimal 1000 translated into hex. No multiplication and division by 256. Just put them together, or take them apart, as appropriate. This becomes particularly important when dealing in machine code, since the X,Y, and U 16-bit registers each divide into 2 parts, higher and lower, XH and XL, etc.

To some extent it is, like so much else, a matter of practice. The more you use hex, the less horrible it will become.

#### NEW "INSTANT RESCUE"

A program which has been RESCUed by the routine on page 56 can sometimes give trouble after it has been edited, particularly if the technique - NEW STATUS 2 - NEW - NEW 0 - RESCUE - has been used, as detailed on page 66. The routine below, which works on a somewhat different principle, avoids these difficulties. It also takes care of the rare difficulty caused by the first line of the subject program being an exact multiple of 256. In this event the first line will be numbered as 1, and may be renumbered by hand.

- a) `G=STATUS 2-STATUS 1-(STATUS 1=0)-189 [ENTER]`
- b) `POKE G,72,PEEK 30821,74,197,73,0,&B5,0,68,7,&89,4,79,1 [ENTER]`
- c) `POKE G+14,68,&B5,13,7,&99,6,68,&B5,255,7,&99,12 [ENTER]`
- d) `POKE G+26,&84,&AE,120,103,4,&AE,120,104,&9A [ENTER]`
- e) `CALL G [ENTER]`

Some readers may be interested to know how this routine works.

Line a) sets location for m/c routine. On NEW the 1st byte of program area was filled with 255: so 0 is placed in this byte, by line b), which also checks that the 2nd byte is not 0, (which it would be if 1st line number was originally an exact multiple of 256). If both bytes are now 0, 1 is placed in the 2nd byte, since a line number 0 will not RUN.

Line c) checks through addresses for the occurrence of 13 and 255 in successive bytes. 13 is End-of-Line marker, and 255 is End-of-Program marker. All programs terminate thus, automatically. Line d) takes the address where this occurs, and places its major and minor parts into the End-of-Program system pointers, in addresses 30823 and 30824.

RESCUE is executed by line e), and the lost program is fully rehabilitated.

## RENUMBER - (2)

This program is for the purpose of RENUMBERING part of a subject program. Care must be taken NOT to allocate new numbers which may be identical with numbers already existing. If in doubt, renumber the latter first. MERGE it with the subject program, and execute by DEF B. As usual, this label must be changed, if it exists in the subject program.

RENUMBER (2) prints out a list of old and new Line numbers. If this is not required, alter LPRINT to REM in line 60215.

```
60100:"B"CLEAR :
      WAIT 0
60110:INPUT "1st l
      ine=";A
60120:INPUT "new n
      umber=";B
60130:INPUT "step=
      ";C
60140:INPUT "last
      line=";D
60150:S=STATUS 2-
      STATUS 1:P=S
      :E=B
60160:N=256*PEEK P
      +PEEK (P+1):
      PRINT N
60170:IF N<AGOTO 6
      0230
60180:IF N>DGOTO 6
      0700
60200:Q=INT (E/256
      ):R=E-256*Q
60210:POKE P,Q,R
60215:LPRINT N;E
60220:E=E+C
60230:GOSUB 60500:
      GOTO 60160
60500:P=P+3+PEEK (
      P+2):RETURN
60700:M$="N":INPUT
      "more?(Y/N)"
      ;M$:IF M$="Y
      "GOTO 60100
60710:F=STATUS 2-3
      94
60720:G=INT (F/256
      ):H=F-256*G
60730:POKE 30825,
      PEEK 30821,
      PEEK 30822
60740:POKE 30823,G
      ,H
60750:BEEP 5:END
STATUS 1
394
```

Next Month: RESTRUCTURE1 - a program which will renumber a block, and rearrange the program accordingly.

## DATA

DATA statements are not quite so inflexible as one would think. The RESTORE (line no.) facility can be manipulated to advantage. First you could do a dummy READ of all the DATA statements, since you cannot initially RESTORE them until you have READ them all. Just READ them all, without doing anything with what you have READ. Then RESTORE N, where N is a variable subject to the workings of your program. You do have to work out with care exactly what you want to do: but the ability to GOTO X (a variable) is a valuable one which should be in everyone's vocabulary.

You may not realise that a DATA statement can itself be a variable, numeric or string, or even an expression. e.g.

```
100: DATA 25,37,X,A$, "HELP", 21*57,LEFT$(Q$,3) and so on
```

What is more, the value of a variable in a DATA statement can be varied during the running of the program. Try this experiment:

```
10: FOR F=1 TO 6:READ A: PAUSE A: NEXT F: END
20: DATA 10,20,F,40,50,100*F
```

Also try this:

```
1: DATA "FIRST "
2: DATA "MIDDLE "
3: DATA "LAST "
10: FOR F=1 TO 3:READ A$:LPRINT A$;:NEXT F:LPRINT
20: FOR G=3 TO 1 STEP-1:RESTORE G:READ A$:LPRINT A$;:NEXT G
```

## INSTANT SUPERSALVAGE

FROM THE KEYBOARD:

(operating time: 3 seconds)

- 1) K=STATUS 2-STATUS 1-1-(STATUS 1=0) [ENTER]
- 2) G=K-132 [ENTER]
- 3) POKE G,68,&FD,&5A,68,68,5,&FD,&CA,&B5,13,7,&9B,13,&FD,24,&FB,&9A [ENTER]
- 4) CALL G,K [ENTER]
- 5) POKE K,255 [ENTER]
- 6) POKE 30823,INT(K/256), K-256\*INT(K/256) [ENTER]
- 7) POKE STATUS 2-STATUS 1,0 [ENTER]

- NOTES:
- a) The illogical mixture of hex and decimal is to save space.
  - b) INSTANT SUPERSALVAGE will not work on a "protected" program.
  - c) It could possibly fail as a result of coincidences as explained in ADDRESS FINDER (page 77)
  - d) It will only work on programs which are "normally" placed in the program area.
  - e) It may be used to restore programs which have failed to CLOAD as a result of ERROR 44 or 43, up to the point of interruption. It may also be used to rehabilitate a program which has gone to pieces at the end, as a result of adventures with the system pointers.
  - f) The routine may also be used to DELETE a MERGED program.
- 

## MINDBOGGLE CORNER

This months problem is not an easy one. It is to translate into useful, meaningful English the following 3 extracts from SHARP's *TECHNICAL REFERENCE MANUAL*. Closing date October 21st. Usual splendid prize. Entries will be judged on the basis of clarity, accuracy, and helpfulness.

- a) "It is possible to comprise a memory back-up system using this BF flipflop."
- b) "If there have been the expression-3 during data save, program execution will automatically take place from the address represented by the expression-3 upon completion of program load. However, automatic execution will not take place if the expression is given."
- c) [Creation of Header - Entry Preparation]  
"Use another code to avoid confusion, as the file mode for other than PC 1500 is used."

\* \* \*

Solution to the SEPTEMBER competition. ANGUS CRAWFORD and MIKE SMITH both got very near the answer, by changing the value of F or R in line 45.  $F=\sqrt{R} \cdot \cos(X+P)$  produced a similar, but not an identical distortion. The correct amendment was the addition of & in line 50:

50: LINE (D,E)-(&F,G)-(H,I)-(D,E)

The inevitable winner is: \*\*JOHN MACK\*\*



Undoubtedly a bestseller, this new program in the "EASI---" series is perhaps the most useful yet. It is designed as a complete book-keeping program for a small business, or for domestic use. It works on the "double-entry" and analysis system.

If the thought of double-entry book-keeping is frightening, do not be daunted! The manual, as usual a model of clarity and simplicity, not only tells all you need to know about the program, but explains the mysteries of the book-keeping process in such a way as to remove previous terrors. I have of course adopted it in place of my own primitive accounting system for the newsletter, and now have available at a glance much information which previously required diligent searches.

The cassette contains 2 programs; the 1st for setting up an account-book according to your own specific requirements, the 2nd for running it. Transactions may be printed out in full, or in summary. The full print-out is arranged in such a way as to make it possible to paste up the strips into a conventional analysis account book. 2 colours are used, black and red, for credits and debits respectively. It is not necessary to have any previous acquaintance with accounting methods, since the manual tells you all you need to know.

Book-keeping is of course a continuous process. Details of transactions entered may be cleared, to create space for further ones. Depending on the complexity of analysis required, about 70 to 100 transactions, on average, may be entered before clearing; however, as well as the printouts, data entered may be saved to tape: and all balances are carried forward despite the clearance of details of transactions.

There is only 1 main and 1 secondary menu, making for ease and speed of working. However full printout is naturally slow: as is loading data: this applies of course to all PC1500 programs, and is a totally unnecessary fault in the machine. Unless you keep the program permanently in the computer, you would need some patience when loading both program and data; no doubt merely to note transactions in rough, and have a weekly or monthly book-keeping session is the answer. This is probably how most people work, anyway.

While "setting-up an account book" is as flexible as you could wish, and must always be executed first, the program itself contains extremely sophisticated coding. It is educational to see what can be done with the apparently limited BASIC provided by the PC 1500. On the other hand, it does mean that it is extremely inadvisable to tamper with the programming in order to provide facilities for unusual requirements.

Being lazy, I would have liked to see a facility which could automatically transfer any payments-in from 'cash account' to 'bank account', instead of having to physically enter the transaction twice. But of course the more facilities, the less space for transactions to be recorded; and the program strikes a balance between the two needs in a manner of which no one could reasonably complain.

An admirable facility is the way in which the calendar in the PC 1500 automatically enters the correct date: even this may be circumvented if you are working on a backlog of entries from earlier dates. If EASICASH had been available when the newsletter started, I would have been saved a great deal of trouble.

EASICASH has been developed by IAN TRAYNOR, and is obtainable from ELKAN ELECTRONICS, at £19.95

One obvious effect which is possible with the LCD display is 'screen inversion' - changing the display to white-on-black. You may have tried the GPRINT/POINT method illustrated in the manual,

```
10: WAIT 0: PRINT "ABCDEFGHJKLMNOPQRSTUVWXYZ": CURSOR 0
20: FOR N=0 TO 155: GPRINT 127-POINT N;: NEXT N
30: WAIT: PRINT
```

This takes about six seconds, making it a suitable case for machine-code treatment; in fact the machine-code version will be well over a thousand times faster.

You can think of the display as an area of memory. A location in this area will contain an eight-bit binary number, for example '10010101'. The '1's in this code correspond to display dots which are switched on, while the '0's similarly stand for blank dots. So the problem is simply to 'flip' all of these 1s and 0s; starting from '10010101', we should end up with '01101010'. If this is done for every location in the display area of memory, then the whole screen will be inverted.

The microprocessor does most of its data processing in the internal register 'A', which is just big enough to hold a single 'PEEK' value, or eight binary bits (1s and 0s). So the first step is to get the value from the LCD location into register A. Suppose that the address of this LCD location is held in the U register, ignoring for the moment the problem of how it got there. Then register U is said to 'point' to the required location, and since the 16-bit registers (X, Y and U) are often used in this way, they are sometimes called 'pointer registers'. The instruction

LDA (U)      [*load A with contents of address held in U*]

(read "load A, address in U") instructs the processor to copy a value from memory into register A; the particular memory location required is 'pointed to' by the U register, which contains its address.

Next, we have to invert (or 'complement') the value which has been fetched. The simplest way to do this is to use the exclusive-OR function. This is an operation similar to addition & subtraction; it works on two input values to produce a result. In binary, its action is to produce an output of '0' where the two inputs are the same, and '1' when they differ (i.e., when either of the inputs is '1', but not both; this is the meaning of 'exclusive-OR'). This test is performed on all eight bits individually. So in the example shown, look first at the right-hand column. Here the inputs are different, so the corresponding result bit is '1'; in the next column along, the inputs agree (both '1') so the result has '0' in this column; and so on.

Suppose now that one of the inputs is all ones; '11111111' binary, or &FF (hexadecimal), or 255 (decimal). As the second example shows, the

10010101	) inputs	result in this case is just the inversion, or
exor 11111111	) inputs	complement, of the other input. This is what we
01101010	= result	wanted, for starting with the value '10010101'
		for one of the inputs, we have result '01101010'.

So to invert the value held in register A, we can use the single assembly language instruction,

EAI &FF

(read "exclusive-OR the value in register A, with immediate data FF hexadecimal"). The phrase 'immediate data' means that the required data (&FF) 'follows immediately' in the instruction; so it doesn't have to be fetched, for example, from some other memory area.

[continued.....]



Now the inversion is accomplished, and it only remains to store the result in the memory location from which the initial value was read. This location is still indicated by the address in register U, so the instruction

STA (U)

("store A, address in U") will do that.

This sequence of three instructions, 'LDA (U); EAI &FF; STA (U)', will correctly invert a small group of display dots; those in a particular location whose address is in the U register. If U can be made to point to each of the LCD locations sequentially, then our work is done. But there is a snag. The display memory consists of two disjoint sections, between which are found the display annunciators (we certainly don't want to invert those), as well as the fixed string storage areas for E\$ through O\$. So a simple loop, as used in the original Basic program, will not suffice.

The LCD locations are at addresses &7600 - &764D and &7700 - &774D. Suppose that U contains the address of a location in the first of these areas; &76xx, say. Recall that register U consists of a high-order part 'UH', holding the &76, and a low-order part 'UL', holding &xx. If UH is increased by one, using the assembly language instruction 'INC UH', then

its value will be &77, and the complete U register will contain &77xx - which is also an LCD address, but in the second area. Now we can use our sequence of three instructions again, to invert this new location, finally restoring the pointer to the first LCD area by means of a DEC UH instruction. The new instruction sequence is shown on the left. This will invert the location which U points to, in the first LCD area, and also the corresponding location in the second area. It's enough now to let U point to all the area 1 locations in turn; area 2 will take care of itself! This trick will be useful in any routine to manipulate the LCD display at the dot level.

So register U has to take values &7600 through &764D; in other words, UH=&76, and UL takes values &00 to &4D. It so happens that UL is the best choice for a loop counter, because there is a special instruction LOP UL,i known as a 'loop primitive' (that just means a useful building block for loops). It is placed at the end of a series of instructions which are to be repeated a certain number of times. When the microprocessor encounters this instruction, it subtracts one from UL; then, if UL was not zero before the subtraction, it moves back to the start of the loop, so that the loop sequence is repeated; otherwise it just continues with the next instruction after LOP UL,i. This facility was the reason underlying the original choice of U as pointer; in all other respects, X or Y would have done as well.

LDI UH,&76	68 76
LDI UL,&4D	6A 4D
LDA (U)	25
EAI &FF	BD FF
STA (U)	2E
INC UH	FD 60
LDA (U)	25
EAI &FF	BD FF
STA (U)	2E
DEC UH	FD 62
LOP UL,&E	88 0E
RTN	9A

Here's the complete program with hex opcodes. The value of 'i' in the LOP UL,i instruction is &0E, or 14 decimal, since that's the number of 'bytes' (two-digit hex codes) between the first LDA (U) instruction and the LOP instruction itself, including both of these. This is the loop section.

The first two LDI ("load immediate") instructions set up the initial address, &764D, in the U register. The LOP instruction ensures that all LCD locations in the range &764D down to &7600 are processed, while the INC UH and DEC UH take care of the range &774D down to &7700.

As usual, the opcodes have to be POKEd into a free area of RAM (any area will do); for example, to use the fixed memory areas for A\$ and B\$, POKE the nineteen hex codes in sequence starting at &78C0. Then if you CALL &78C0 from Basic the LCD screen will be inverted.

[demonstration program overleaf

Serial (RS-232C) Interface (cont.)

MODEMS: Most users are likely to use Datal 200 compatible 300 bps modems of two possible types: acoustic coupled or hard-wired. Examples of these are the Anderson Jacobson and British Telecom Modem 13A, respectively. These modems are undemanding, usually needing only pins 2,3 and 7, while providing DCD when on-line. Most (but not the BT 13A) also provide CTS & DSR. Many 1200 bps modems are similar (and the PC-1500 will work quite happily at this speed) but faster or more complex modems will present special problems. Prestel modems will not work as they use two different data transmission rates for TD and RD (75/1200 bps).

Parallel (Centronics) Interface

I don't pretend to understand the theory of parallel interfaces, however, by following my own advice of keeping the connections to a minimum I have had successful results with a cable connected as follows:

CE-158 Pin	DESCRIPTION	PRINTER Pin	CE-158	DESCRIPTION	PRINTER
1	Strobe	1	<nc>	Acknowledge	10
2	Bit 0	2	10	Busy	11
3	Bit 1	3	<nc>	Paper Out	12
4	Bit 2	4	<nc>	Select	13
5	Bit 3	5	11	Init	<nc>
6	Bit 4	6	14	0v	14
7	Bit 5	7	<nc>	+5v	18
8	Bit 6	8	<nc>	Input Prime	31
9	Bit 7	9	<nc>	Fault	32

The BASIC commands required to drive the interface are -  
OPN<LPRT> : CONSOLE 0,0,1 : FEED n : LPRINT A\$

My CE-158 works quite successfully with GE Terminet 2030 and 2120 serial printers, Terminet 3304, which has both serial and parallel interfaces; Televideo 950 CRT; via BT 13A modem, Anderson Jacobson 300 bps and 1200 bps acoustic couplers into a time-sharing service where the PC-1500 makes a neat terminal for electronic mail; and acting as a host computer to an IBM PC running a terminal emulator program.

This text was prepared and printed using SUPERTEXT!, with a small change to remove ASCII characters 123 & 125 which are printable on the printer used (GE Terminet 3304, parallel i/f).

SCREEN INVERSION - Demonstration Program

```
1: POKE &78C0,&68,&76,&6A,&4D,&25,&BD,&FF,&2E,&FD,&60
2: POKE &78CA,&25,&BD,&FF,&2E,&FD,&62,&88,&0E,&9A
3: WAIT 0:PRINT "ABCDEFGHJKLMNOPQRSTUVWXYZ"
4: CALL &78C0: BEEP 1,10,1E3: GOTO 20
```

FOOT NOTE

The cat next door enters through my window and walks across my computer, treading delicately. I have deterred it by amending the WEIRD NOISES program (page 59) so that they are set going at the touch of a single key. Successfully. The cat no longer walks across the computer. It sits on it.



\*\*\*\*

MARKET INFORMATION

\*\*\*\*

The CE 161 memory extension is advertised in USA at the attractive price of \$139.00. To this must be added Postage. There will probably also be a liability to Customs Duty and VAT. Write to:-

COMPUTER MAIL ORDER EAST  
Dept. 901  
477 E. 3RD STREET  
WILLIAMSPORT  
PA 17701

or COMPUTER MAIL ORDER WEST  
Dept. 901  
PO Box 6689  
STATELINE  
NV 89449

---

**\*\* STATUS SOFTWARE \*\***

SUPERTEXT! has been available to subscribers to this newsletter at the 'Special Offer' price of £7.95, including postage and packing within UK.

This offer is now EXTENDED INDEFINITELY. See review on page 58 of STATUS 1500.

Write to:

STATUS SOFTWARE, 62 BLENHEIM CRESCENT, LONDON W.11

---

**\* TANDY \***

TANDY now have available their DMP 200 Daisywheel Printer at £599. including VAT. Write to TANDY CENTREPOINT COMPUTER CENTRE for further information.

---

## #CAP-SET#

THE PROGRAM THAT  
PRODUCED THIS AD.

WRITES

# BIG

SMALL, OR  
**SLOPING**  
PROGRAMMABLE  
PRINTING OR  
TYPEWRITER MODE

REQUIRES 8K RAM  
EXPANSION.

PRICE

**£9.50**

## #SIXPACK#

1 GRID...  
MATHEMATICAL  
SHAPES

2 3-D...  
DRAWS IN THREE  
DIMENSIONS

3 LURD...  
USER-FRIENDLY  
SCREEN GRAPHICS

4 SCRAMBLOID...  
MINI ARCADE  
GAME

5 NOUGHTS &  
CROSSES

6 COLOUR DEMO...  
INTERESTING  
GRAPHICS

ALL ON ONE TAPE  
PRICE

**£6.75**

# D.O.D.O. SOFTWARE



ALL TAPES ARE  
SUPPLIED WITH A  
MANUAL AND PRICE  
INCLUDES POST &  
PACKING.

D.O.D.O. SOFTWARE  
150 HIGH STREET  
TWERTON  
BATH BA2 1BY