

FEB '86

4



UK EINSTEIN USER GROUP  
NEWSLETTER

Issue Number Four February 1986

# EDITORIAL

We have lots of good news for you in the form of new sections within the Newsletter. First you will find an entry form at the bottom of this page for our competition. We hope this will become a fairly regular feature. Second we have our new Adventure column and third our usual mix of ideas and articles.

I nearly forgot, We are more organised and have a mailing list set up (saves writing all those envelopes), but we still don't have any membership cards. The label on your envelope also has your membership No on it, make a note of it you will need it for the competition.

Now it's reminder time again. Don't forget the listings are available on disk at only £6.50 (apologies for any delay with January's disks but we ran out) please make cheques payable to UKEUG (I don't mind but the Treasurer does).

Also if you send anything to us for publication please help following these simple rules.

1. Send it all on disk if possible ( we will return it A.S.A.P ).
2. Include a print out .
3. Articles should be in the form of text files ( any word processor will do, WordStar if possible ).
4. Include your name and address on everything.

The address for all of the above is.

Keith Stokes  
Hillcroft  
Codmore Hill  
Pulborough  
West Sussex  
RH20 1BQ

Right on with the show

## IT'S COMPETITION TIME.

We have a copy of FORTH which has been donated by TATUNG to give away in a simple competition. All you have to do is list in your order of preference the six features of the Einstein printed below and then fill in the breaker and post it to me at the above address to arrive by 28th March.

### UKEUG Competition No 1

- |                        |                                  |
|------------------------|----------------------------------|
| A. GOOD KEYBOARD       | B. BUILT IN DISK                 |
| C. USES MANY LANGUAGES | D. CAN BE USED FOR BUSINESS      |
| E. CP/M COMPATIBLE     | F. BOTH RS232 & CENTRONICS PORTS |



SCIENCE and SORCERY - THE ADVENTURES COLUMN  
( DAMMIT I'VE DIED AGAIN )

Welcome to the first of many (I hope) adventurers column to appear in the UK Einstein User Group Newsletter. This column will feature reviews, advice, hints and tips for specific games, news of new releases, books to read and just about anything else I can think of that may appeal to adventure players. If you have any comments on this column, tips to pass on, questions on games or adventuring in general, completed maps, games to recommend, or if you just want to have a chat about adventuring, why not drop me a line? I will try to reply as promptly as possible, but if you want a personal reply, please include an SAE. If you want to be put in touch with other adventure game players in your area, let me know what areas are close to you, and I will try to match you up to others in your area. That's enough intro, on with the column.

HITCHHIKERS GUIDE TO THE GALAXY  
-----

**DON'T PANIC!** A sound piece of advice for hitchhikers playing Infocom's latest game. Hitchhikers guide is based on the highly successful trilogy by Douglas Adams, who co-wrote the game with Infocom's Steve Meretzky.

The game, like all Infocom games is a text only adventure game (or 'interactive fiction' as Infocom like to describe it). Given the current trend towards graphics adventure games, this lack of graphics may put you off forking out '29.99 for Hitchhikers, but DON'T LET IT!! Infocom games are quite simply the best available, and this is one of their best. The location descriptions in Hitchhikers are so full of depth that graphics really are not necessary.

Hitchhiker arrives in first class packaging. Along with the game disc you also receive such vital goodies as Peril Sensitive Sunglasses which darken in the presence of danger (they are always black!), a piece of Fluff, a DON'T PANIC Badge and others. The documentation cannot be faulted. It includes an overview of what interactive fiction and Hitchhikers is all about, tips for novices, advice on communicating with the game, a quick reference guide, lists of important commands and some of the verbs the game understands. The only complaint I have is that in the package I received didn't include an order form (for sending off to Infocom for hints and a map), or a registration form to be placed on Infocom's mailing list.

You begin the game as Arthur Dent, waking up with a mighty hangover. After spending some time getting up, curing your hangover and trying to prevent your house being knocked down by a bulldozer, you eventually discover that the Earth is about to be destroyed by a Vagon fleet to make way for an interstellar bypass. If you have read the book, this will probably help you in the early stages of the game, but the game soon diverges from the plot of the book, and then the only thing you have to help you is your wits.

This is a game which should be equally difficult for experienced or beginner adventure players alike. Accepted methods of playing these games will not necessarily be of any use. To be a successful hitchhiker you must have an unusual sense of humour, and you mustn't take anything at face value. This is one area where knowledge of the book may help, if you know Adam's sense of humour you should have some idea of the strange universe you are about to enter. The game can be extremely devious as well, just wait until you try to get the Babel fish, I can guarantee you will curse the names Adam's and Meretzky! If you should get really stuck, you can always consult the guide. On your travels your friend Ford Prefect gives you a copy of 'The Hitchhikers Guide to the Galaxy', 'a sort of electronic book. It tells you everything you need to know about anything'...sometimes. By the way, if you think I have forgotten to mention what your objective in this game is, I haven't. On asking the game 'what is the object of this

game' you are told 'that is for me to know and for you to find out'. Hitchhikers Guide to the Galaxy is a game beyond all others. It will take you out into the farthest and strangest corners of the galaxy (just as well, considering the Earth is reduced to lump of smouldering ash!). You will meet all manner of unusual being, such as the two-headed President of the Galaxy - Zaphod Beeblebrox, or perhaps the Ravenous Bugblatter Beast of Traal. You may even find you become some of these beings, and then you really will get confused. It's the game that sets the standard by which all other adventure games will be measured, and it's one you will spend many weeks, possibly months playing. Buy it, you won't regret it.

## BIGSUMS

By Peter Heffernan

```

10 REM BIGSUMS
25 I=1:SHAPE63,"0000000000000000"
40 PRINT@3,12;"WILL WE DO HARD SUMS"
60 PRINT@3,16;"TYPE IN'HARD'OR'EASY'";
80 CLS
100 IFA$="EASY"THENGOTO600
120 IFA$="EASY"THENGOTO600
130 FORL=16TO22STEP1
136 PRINT@37,I;" "
140 PRINT@23,L;" "
160 PRINT@1,2;"WHAT DOES "
180 PRINT@1,6;"UP TO ";
200 IFO=QTHENX=Q
220 IFO<>QTHENBEEP
230 IFO<>QTHENI=I+1
245 TCOL15
260 LETB=B+8
280 PRINT@23,L;" "
300 GOSUB930
320 IFF=>10THEN PRINT@22,14;"1"
340 IFW=>1THENGOTO360
360 LETA=A-8
380 PRINT@16,L;" "
400 PRINT@1,2;"WHAT DOES "
420 PRINT@1,6;"UP TO ";
440 IFO=WTHENGOTO470
451 TCOL1
453 IFO<>QTHENPRINT@37,I;" "
460 IFO<>WGOTO400
480 PRINT@16,L;" "
500 X=W
520 FORL=1TO7
540 NEXTL
560 PRINT@1,2;"WELL DONE"
580 SPEED255
600 FORL=0TO14
620 NEXTL
640 TCOL6
660 TCOL15
680 PRINT@16,L;" ";
700 LETA=24
720 LETB=0
740 PRINT@16,15;" "
750 TCOL6
770 PRINT@12,L;" "
790 PRINT@10,11;" "
810 LETE=RND(9)

20 RST:BCOL4:TCOL15
30 PRINT@3,10;"HELLO"
50 PRINT@3,14;"OR EASY SUMS"
70 INPUTA$
90 IFA$="HARD"THENGOTO1850
110 IFA$="HARD"THENGOTO1850
125 GOTO60
135 TCOL1
137 TCOL15
150 NEXTL
170 PRINT@1,4;"THIS ADD "
190 INPUT O
210 IFO=QTHENGOTO260
225 TCOL1
240 IFO<>QTHENPRINT@37,I;" "
250 IFO<>QTHENGOTO160
270 FORL=16TO22
290 NEXTL
310 IFF=>10THENTCOL11
330 TCOL15
350 IFW< 1THENGOTO520
370 FORL=16TO22
390 NEXTL
410 PRINT@1,4;"THIS ADD "
430 INPUTO
450 IFO<>WTHENBEEP
452 IFO<>QTHENI=I+1
454 TCOL15
470 FORL=16TO22
490 NEXTL
510 GOSUB930
530 PRINT@1,L;" "
550 SPEED150
570 PRINT@1,4;"TRY AGAIN"
590 GOTO110
610 PRINT@16,L;" "
630 C=C+1
650 PRINT@35,G;" "
670 FORL=16TO22
690 NEXTL
710 LETW=0
730 TCOL10
745 PRINT@16,23;" "
760 FORL=9TO13STEP1
780 NEXTL
800 LETD=RND(9)
820 LETF=E+D

```



```

830 IFF>=10THENW=J+1
850 IFF<=9THENQ=F
870 LETX=D:
890 LETX=E
910 GOSUB930
930 TCOL15
950 IFX=2THENGOSUB1130
970 IFX=4THENGOSUB1290
990 IFX=6THENGOSUB1450
1010IFX=8THENGOSUB1610
1030IFX=0THENGOSUB1770
1050 PRINT@A,B;"  x"
1070 PRINT@A,(B+2);"  x"
1090 PRINT@A,(B+4);"  x"
1110 PRINT@A,(B+6);"  xxx"
1130 PRINT@A,B;"  xxx "
1150 PRINT@A,(B+2);"  x"
1170 PRINT@A,(B+4);"  x "
1190 PRINT@A,(B+6);"xxxxx"
1210 PRINT@A,B;"  xxx"
1230 PRINT@A,(B+2);"  x"
1250 PRINT@A,(B+4);"  x"
1270 PRINT@A,(B+6);"  xxx"
1290 PRINT@A,B;"x  "
1310 PRINT@A,(B+2);"x  "
1330 PRINT@A,(B+4);"x x"
1350 PRINT@A,(B+6);"  x"
1370 PRINT@A,(B );"xxxxx"
1390 PRINT@A,(B+2);"x  "
1410 PRINT@A,(B+4);"  x"
1430 PRINT@A,(B+6);"  xxx"
1450 PRINT@A,(B+0);"  xxx"
1470 PRINT@A,(B+2);"x"
1490 PRINT@A,(B+4);"x  x"
1510 PRINT@A,(B+6);"  xxx"
1530 PRINT@A,(B+0);"xxxxxx"
1550 PRINT@A,(B+2);"  x"
1570 PRINT@A,(B+4);"  x"
1590 PRINT@A,(B+6);"x"
1610 PRINT@A,(B+0);"  xxx"
1630 PRINT@A,(B+2);"x  x"
1650 PRINT@A,(B+4);"x  x"
1670 PRINT@A,(B+6);"  xxx"
1690 PRINT@A,(B+0);"  xxx "
1710 PRINT@A,(B+2);"x  x"
1730 PRINT@A,(B+4);"  x"
1750 PRINT@A,(B+6);"  x"
1770 PRINT@A,(B+0);"  xxx"
1790 PRINT@A,(B+2);"x  x"
1810 PRINT@A,(B+4);"x  x"
1830 PRINT@A,(B+6);"  xxx "
1850 FORL=0TO14
1870 NEXTL
1890 TCOL6
1910 TCOL15
1930 PRINT@16,L;"
1950 LETJ=0
1970 LETA=16
1990 LETX=RND(3)
2010 IFX=0GOTO2030
2030 LETA=16
2050 LETX=RND(3)

```

```

840 IFF>=10THENQ=F-10
860 IFF<=9THENW=J
880 GOSUB930
900 LETB=B+8
920 GOTO130
940 IFX=1THENGOSUB1050
960 IFX=3THENGOSUB1210
980 IFX=5THENGOSUB1370
1000 IFX=7THENGOSUB1530
1020 IFX=9THENGOSUB1690
1040 RETURN
1060 PRINT@A,(B+1);"  xx"
1080 PRINT@A,(B+3);"  x"
1100 PRINT@A,(B+5);"  x"
1120 RETURN
1140 PRINT@A,(B+1);"x  x"
1160 PRINT@A,(B+3);"  x "
1180 PRINT@A,(B+5);"  x "
1200 RETURN
1220 PRINT@A,(B+1);"x  x"
1240 PRINT@A,(B+3);"  x"
1260 PRINT@A,(B+5);"x  x"
1280 RETURN
1300 PRINT@A,(B+1);"x  "
1320 PRINT@A,(B+3);"x  "
1340 PRINT@A,(B+5);"xxxxxx"
1360 RETURN
1380 PRINT@A,(B+1);"x"
1400 PRINT@A,(B+3);"  xxx "
1420 PRINT@A,(B+5);"x  x"
1440 RETURN
1460 PRINT@A,(B+1);"x  "
1480 PRINT@A,(B+3);"xxxxx"
1500 PRINT@A,(B+5);"x  x"
1520 RETURN
1540 PRINT@A,(B+1);"  x"
1560 PRINT@A,(B+3);"  x"
1580 PRINT@A,(B+5);"x"
1600 RETURN
1620 PRINT@A,(B+1);"x  x"
1640 PRINT@A,(B+3);"  xxx"
1660 PRINT@A,(B+5);"x  x"
1680 RETURN
1700 PRINT@A,(B+1);"x  x"
1720 PRINT@A,(B+3);"  xxxx"
1740 PRINT@A,(B+5);"  x"
1760 RETURN
1780 PRINT@A,(B+1);"x  x"
1800 PRINT@A,(B+3);"x  x"
1820 PRINT@A,(B+5);"x  x"
1840 RETURN
1860 PRINT@16,L;"
1880 G=G+1
1900 PRINT@35,G;"  "
1920 FORL=16TO22
1940 NEXTL
1960 LETW=0
1980 LETB=0
2000 K=X
2020 IFX>0THEN GOSUB930
2040 LETB=B+8
2060 J=X+K

```

2070 W=W+J  
2090 IFX>0THENGOSUB930

2080 IFX=0GOTO700  
2100 GOTO700

# SETXXBD - Setting the Printer Baud Rate Automatically. Paul Burgess.

It's often convenient to drive a printer via the serial port, perhaps because you already have a serial printer, or, like me, have the parallel port tied up with something else. Very often, the printer will require a data rate which is not the default (9600 for the Einstein) and it can be annoying to have to enter MOS after every 'Ctrl-Break'. To deal with this, I wrote SETXXBD, and set up the DOS to execute it automatically after every cold boot. The program shows the use of the Einstein's MOS calls inside a program, too. This version was assembled using HiSoft's excellent 'Devpac' assembly language system, but it can be entered by hand or with the Kuma 'Zen' assembler. Any of the standard baud rates may be set by changing the value loaded into the L register according to the table in the Einstein Mos/Dos manual. You should modify the name to indicate what baudrate is used: the example here is SET48BD for 4800 baud.

Firstly, EQU statements are used to define labels for the MOS calls and for useful things like carriage return, and so on. This is just to make the program easier to read after I have forgotten what it was for! The ORG statement starts the program at address 100 Hex., this is the standard start address for CP/M and of course XTAL DOS programs. It is usually wise to save the incoming stack pointer from the DOS, and establish a local stack for the program: this is done by the next two statements and the DEFS (Define Storage) statement at the end of the program.

Now we get round to the proper job. We'll use the BAUD Mos call ("Mcal") which is the equivalent of the "b" command in the MOS. We need to supply the code for receive and transmit baud rate in the L register of the Z80 and set the DE register to 0 to ensure the data format is not changed. Then, we call the routine by executing a RST 8 (Restart 8) followed by the code for "b", which happens to be 81 Hex. That's all that is needed to do the job, but we should print a short message so the user knows what is going on. The "ZPRM" Mcal is used for this: again we do a 'RST 8' followed by the code, which is CF Hex. We then supply the prompt message, and terminate it with a carriage return and a line feed. The 80H is added to the line feed in order to tell the ZPRM routine to stop printing. Finally we restore the stack pointer to its old value, and do a RETURN to the DOS.

Having assembled the program, or entered it in hexadecimal using the 'M' command in MOS and then 'SAVE 1 SET48BD.COM' all that's needed is to load the name of the program into the command buffer in the DOS. This is done by reading the system into memory, altering the appropriate locations and re-writing the modified system to disc. ALWAYS try this on a blank disc first. Also, ensure that SETXXBD.COM is present on the system disc and any other you intend to use in drive 0.

Enter Mos and modify the system with

>R 1000 3800	(System in memory from 1000H)
>M 1208	(Modify command buffer)
1208 00	(Current contents shown)
1208 07	(You enter this)
1209 53 45 54 34 38 42 44	(Enter 'SET48BD' in Ascii)
1210 00	(Now enter full stop)
>W 1000 3800	(Re-write the system to disc)
>Y	(Back to the DOS)



- If all is well the system will boot and execute SET48BD (or whatever your version is called). Note that the comments in brackets above are not supposed to be entered!

You may also be wondering about other MOS calls. Some of these were dealt with in the last edition of the Tatung Einstein User magazine. I hope to expand on the theme in future issues of EUGUK, and also take a look at the DOS calls, which are in the main compatible with CP/M.

HiSoft GEN80 Assembler 27 Mar 85 Page: 1

Pass 1 errors: 00

```

0100      1 ;*****
0100      2 ; SETXXBD - SET EINSTEIN BAUDRATE FB 860103 *
0100      3 ; Can put this on system disc for autoboot. *
0100      4 ;*****
0100      5 ;
0100      6 ;
0081      7 BAUD      EQU    81H      ; SET BAUDRATE CALL
00CF      8 ZPRM      EQU    0CFH    ; PRINT MESSAGE CALL
000D      9 CR        EQU    0DH     ; CARRIAGE RETURN
000A     10 LF        EQU    0AH     ; LINE FEED
0100     11 ;
0100     12 ;
0100     13          ORG    100H      ; CP/M & DOS PROGRAMS START.
0100     14 ;
0100 ED733001 15      LD     (STAK),SP ; SAVE INCOMING STACK POINTER
0104 314401   16      LD     SP,STAK+20 ; AND SET UP OUR OWN
0107         17 ;
0107 217700   18 SETBD    LD     HL,0077H ; 4800BD TRANSMIT/RECEIVE RATE
010A 110000   19          LD     DE,0      ; NO COMMAND CHANGE
010D CF       20          RST     8        ; CALL THE MOS
010E 81       21          DEFB   BAUD     ; BAUD RATE FUNCTION
010F         22 ;
010F CF       23          RST     8        ; CALL THE MOS
0110 CF       24          DEFB   ZPRM     ; OUTPUT A MESSAGE
0111 42617564 25          DEFM   "Baud Rate Reset to 4800."
0129 0D8A     26          DEFB   CR,LF+80H ; END MESSAGE
012B ED7B3001 27          LD     SP,(STAK) ; RESTORE DOS STACK
012F C9       28          RET          ; AND RETURN TO DOS
0130         29 ;
0130         30 STAK     DEFS    20      ; RESERVE SOME STACK MEMORY
0144         31          END          ; ALL DONE.

```

Pass 2 errors: 00

# PATTERNS OF CHAOS (CONTINUED)

by Peter Moon.

## JULIA SETS

The difference between Julia sets (of which there are an infinite number) and the Mandelbrot set is rather subtle. The programmes for generating these two kinds of sets are very similar. The difference lies in the fact that to generate the Mandelbrot set we chose a fixed point in the complex plane and followed its fate for different values of the complex number C, whereas for any one Julia set we must take just one fixed constant C (=RC+IC) and explore the behaviour of all other points in the plane. Hence we can generate as many different Julia sets as we choose by specifying different values for C.

# THE MECHANISM

Under the iterations of the Julia treatment a given point has three possible fates; it can increase towards infinity, shrink towards some interior point or points, or remain at some stable position. If we choose  $C=0$  the process simply generates a circle. All points outside the circle flee to infinity, those inside go towards zero and those on the circumference stay put. For non-zero values of  $C$  (for example  $C=0.31+0.04i$ ) the boundary is no longer a smooth circle but a jagged crumpled closed curve resembling a circular saw that has been in a bad accident (Fig. 2). Like the Mandelbrot set, the boundary is self-similar; that is, a magnified section has very much the same jagged features as the whole curve.

Depending on the value of  $C$ , many types of pattern emerge. Figures 3-5 are examples. To show the interior structure of a Julia set - which can be very intricate - requires much more elaborate programming than I have supplied, and extends execution time considerably.

There is a remarkable connection between the Mandelbrot set and Julia sets. The general form of a Julia set can be predicted from where its complex constant  $C$  would lie if plotted on the Mandelbrot mapping. If  $C$  is within the main body of the Mandelbrot we obtain patterns like Figure 2. Inside a wart we get another kind of pattern, while at the root of excrescence we obtain yet another type.

Each Julia set is connected; that is, every point is joined to every other point. The limiting case is reached with  $C=0+i$ , when the closed curve shrinks to a branched thread, and beyond this stage it breaks up into isolated blobs (Figure 4) which strictly are not Julia sets at all.

## PROGRAMMING

In plotting Julia sets we can make use of the symmetry of the patterns, and for every calculated point we can plot the mirror image point without doing the calculation, thus halving execution time.

A general programme for Julia sets in monochrome is given below. As with the Mandelbrot, the availability of colours adds to the fascination, but as I said in the previous article, you must be able to colour every pixel individually without affecting the colour of its neighbours.

```

10 REM *****
20 REM *
30 REM *
40 REM *          PROGRAMME JULIA4
50 REM *
60 REM *          P.W.H. Moon. 11 January 1986
70 REM *
80 REM * Version 4.2 for UK Einstein User Group
90 REM *
100 REM *          BBCBASIC
110 REM *
120 REM *****
130 :
140 REM INPUT WINDOW CO-ORDINATES
150 INPUT "XMIN, XMAX, YMIN, YMAX";XMIN,XMAX,YMIN,YMAX
160 REM INPUT REAL AND IMAGINARY PARTS OF COMPLEX CONSTANT C
170 INPUT "RC, IC";RC,IC
180 CLS
190 TIME=0
200 REM SCREEN ADDRESSES 0-767 x 0-767, BUT ONLY 192 x 192 PIXELS
    AVAILABLE.
210 DELTAX=(XMAX-XMIN)/767
220 DELTAY=(YMAX-YMIN)/767
230 FOR X=0 TO 384 STEP 4
240 FOR Y=0 TO 767 STEP 4
250 XK=XMIN+X*DELTAX
260 YK=YMIN+Y*DELTAY

```

```

270 XCOUNT=0
280 XKNEW=XK*XK-YK*YK+RC
290 YKNEW=2*XK*YK+IC
300 XCOUNT=XCOUNT+1
310 SIZ=XKNEW*XKNEW+YKNEW*YKNEW
320 XK=XKNEW;YK=YKNEW
330 IF SIZ>100 THEN 360
340 IF XCOUNT<30 THEN 280
350 PLOT 69,X,Y:PLOT 69,767-X,767-Y
360 NEXT Y
370 NEXT X
380 PRINT TAB(76,2);"JULIA4";TAB(76,4);"X";TAB(76,5);XMIN;TAB(76,6);XMAX
390 PRINT TAB(76,8);"Y";TAB(76,9);YMIN;TAB(76,10);YMAX
400 PRINT TAB(76,12);"C";TAB(76,13);RC;TAB(76,14);IC
410 PRINT TAB(76,16);"TIME";TAB(76,17);TIME$;TAB(76,18);INT(TIME/6000);
    TAB(76,19);"MINS."
420 END

```

WHERE DO WE GO FROM HERE ?

There is enough material here for a lifetime study. Exploration of the coastline of the Mandelbrot figure, especially in colour and at high magnifications, is a pursuit which has almost no end. The many varieties of Julia sets offers a constructional challenge which is hard to resist.

Clearly, users will not be content with programmes taking hours or days to run. My family are used to me getting up in the small hours to see whether a plot has finished, but I shall be echoing the feelings of a number of users who have got their feet wet in this fascinating puddle and who want to see results more rapidly.

The need, as I see it, is for double precision arithmetic for the high magnification pictures, and for machine code for the frequently executed steps of the iteration procedures.

UK Einstein users, it's up to you! We want to hear the results of your forays into the complex plane, and share your programming improvements.

#### RECOMMENDED READING

Dewdney, A.K.; Scientific American, August 1985 pp 8-12.  
 Oakley, H.; Personal Computer World, January 1986 pp 216-219.  
 Holden, A.V.; Research Page, University of Leeds Reporter, No.243, 4 October 1985.  
 Eitgen, H.O. and Richter, P.H.; "Schonheit im Chaos", Forschungsgruppe Komplexe Dynamik, University of Bremen, 1985.  
 Holden, A.V.; "Chaos". Manchester University Press.  
 (To be published February 1986).

### The WORLD (part two)

Now to put in the utility bit. By adding the lines shown below to your listing of the WORLD what is produced is a general purpose drawing routine. The way it works is to Poke the data from the data statements into the machine code routine at the end and then save the machine code program complete with its newly acquired data for use with later programs. In response to the prompt "ENTER NAME OF PROGRAM TO SAVE AS AN OBJECT FILE" reply with "TRY1.OBJ". This is the name used in the second listing.

```

81 CLEAR&8000
85 LOAD"SCRPLT.OBJ"
20000 RESTORE
20010 FKLOC=&804A
20020 FORX=1TO100
20030 READ N
20040 IF N=0THENX=100;GOTO 20090
20045 N=N*2;DOKE FKLOC,N;FKLOC=FKLOC+2

```



```

20050 FORJ=1 TO N+2
20060 READ A:POKE PKLOC,A
20070 PKLOC=PKLOC+1
20080 NEXT J
20090 NEXTX
20095 POKE PKLOC,0:PKLOC=PKLOC+1:POKE PKLOC,0:PKLOC=PKLOC+1
20100 PRINT"ENTER NAME OF PROGRAM TO SAVE AS AN OBJECT FILE"
20110 INPUTFILE$
20120 FILE$=FILE$+".OBJ"
20130 SAVEFILE$,8000,PKLOC+1
    
```

The machine code program is used as follows:-

```

10 CLEAR 8000
20 LOAD "TRY1.OBJ"
30 CALL 8000
40 GOTO 40
    
```

To get the best effect from the WORLD program delete lines 10 to 150 and replace them with lines 10 to 30 as above.

Ultimatly, of course, any shape can be defined within the data statements held in the program and because it can be developed in BASIC and then transferred to the machine code version it should make the designing of complicated pictures much more rewarding.

Crystal Research "System 5" - a new DOS package for the Einstein.  
Paul Burgess.

'System 5' is Crystal's latest offering for the Einstein. It comprises a single 3" disc holding a new operating system (DOS 2.01 at the time of writing); a new XBAS with extra facilities; a full-screen text editor for both 40 and 80 column screens, an assembler and a couple of small utility programs in addition to updated versions of BACKUP and COPY. There is also some documentation in the form of twenty-four pages of A4, aparently photocopied.

The DOS itself offers one obvious improvement: the ability to fully support both single- and double-sided drives. The default condition is with drives 0 and 1 set to single, and 2 and 3 to double sided, but this may be altered to any other combination by the user. It is stated that the main improvement is in disc access speed, which I did indeed find to be true. I suspect this is achieved by using sector skewing, hence the need for a new 'Backup' utility. The new DOS is of course fully compatible with discs written under older versions, but to make maximum use of it you would need to reformat your discs with the new utility.

XBAS offers several enhancements including REPEAT/UNTIL; WHILE/WEND; a graphics dump to the printer, enhanced floating-point precision with a command to set the number of significant figures, and, of great interest to many users, I think: 40/32/80 column screen handling and editing.

XSM is a fairly standard Z80 - mnemonics assembler offering conditional assembly but not supporting macros. It is disc-based and operates in two-pass mode upon a file written with the screen editor XED. It produces output in HEX-file format which is then loaded with the utility 'HLOAD' (supplied.) Most of the directives and facilities for printout, etc., that one expects are available: the INCLUDE directive which allows the access of one or more 'library' files from a main program is particularly useful. The error messages I did not find too helpful, particularly compared to something like HiSoft's 'Gen80' assembler (which also supports macros).



XED, the screen editor, I found straightforward. I found its ability to select 40 or 80 column mode automatically to be convenient. Its basic operation was fairly close to the 'WordStar' standard pattern, but the Einstein cursor keys are employed and, of course, it does not attempt to perform text formatting operations as it is not a word processor. It does not appear to support block operations (unlike some other editors) but there is a 'find-and-replace' facility. (more on XED below -ED)

Finally, the utilities: BACKUP operates as normal. COPY also shows no obvious changes, although there is now a sign-on message after it is activated. TIME allows the inbuilt real-time clock to be set and reset, and AUTOGEN is a convenient way of inserting a command line into the DOS for auto-execution on bootup.

CONCLUSIONS: 'System 5' will be available in the very near future at just under £70. I think it offers some welcome improvements, particularly the bundled system software: such should be provided with all serious computers as part of the package anyway. I regret that the DOS still does not cater for batch programs (i.e. CP/M's 'Submit' utility). It seems likely that serious users who have not yet purchased CP/M itself and a set of programming tools like 'Devpac' or the Digital Research standards will find it of most use. I suspect though that many of our members may already have done this and will find the extended XBAS of most interest. In short, it's not bad but a pity it comes so late!

#### Important Note:

At the time of writing, there were several bugs manifesting themselves in the new DOS. One of these in fact was sufficiently serious as to prevent certain programs including Einstein Forth from running correctly. This has been brought to the attention of Crystal Research, and I feel sure it will have been corrected by the time it becomes available.

#### XED

O.K. so what is XED. XED is the editor from CRYSTAL RESEARCH and it comes with the new SYSTEM 5. I have been using it to write the articles for this month's issue and to be quite frank I am surprised at its usefulness. It has a very good type ahead facility and only misses a few letters when it is creating a new line. The documentation on the TAB function is slightly less than explicit, like there is none to say how it works. Setting the TAB positions is easy enough but there is nothing on its operation. All in all if you are looking for an editor of this sort it and don't already have one then XED is well worth getting for the Einstein because it will work in both 40 and 80 columns and has all the facilities required to do the job.

#### REVIEW

SUBJECT-The Brain

ITEM-Magazine

PUBLISHED BY-Glenfield Computer Consultants

WHAT YOU GET-A 30 page magazine with articles on all aspects of computing with the Einstein.

PRICE- 1.35 (See Comments later)

FREQUENCY-6 times a year

COMMENTS-Where do I start? I'll start at the back and work forwards. The reason I used frequency as a title is because our esteemed Editor past (Robbie to his friends) confused us all by saying that BI-ANNUAL is twice a year so BI-MONTHLY should be twice a month. Twice a month would be good value for 16.00 annual subscription (yes I know that doesn't look right I'll come to it in a moment) and it would be great for serialised articles. The truth of it is that The Brain will be published 6 times a year. As for the price, I rang Syntaxsoft to find

out exactly how much the magazine is and found out that the 16.00 that has been put about as the annual subscription is in fact membership of their club which entitles the member to a discount on the normal cost of software and hardware from themselves and also the magazine. If the magazine only is required then that can be obtained at a price of 1.35, although I didn't find out how to go about arranging that. The content of the magazine is thorough in the articles although I did feel that it was aimed at the Experienced user with a capital E. Looking at the Ads it did strike me that Syntaxsoft took up nearly all the advertising space and the balance was shared equally between Screens and Glenfield Computer Consultants.

# TRIALS AND TRIBULATIONS OF CONVERTING PROGRAMS FROM ONE BASIC LANGUAGE TO ANOTHER VERSION.

By V.J.Day.

Most of my spare time since I wrote the article on FRAME for the last issue of the Newsletter has, in many ways, been wasted.

After I wrote the article on FRAME I had a long discussion with the originator, Mr F.R.Pettit (Frank). He reminded me of a companion program he wrote some while after he developed FRAME. Although I had heard of this program I had never seen it working. He therefore promised he would send me a listing. As he is no longer at Oxford he rewrote the program on an Amstrad 8266.

Perhaps a brief resume would be of help. In my article I said that the program was a framework on which to "hang" other programs, in fact I should have said to "build" other programs on. Frank's program is in fact called BUILD. The new programs or suite of programs are written in text form without line numbers but using standard Basic statements and syntax.

The text file however needs (a) a Title and (b) Purpose. These are used in the BUILD program to modify automatically the Menu, the Macro and to give a title to the particular module. After (a) and (b) the text follows the normal requirements of a Basic program but without line numbers. The end of each program in the suit of programs must be marked with the word END and the last program in the suite must also be marked with the word FINISH. These are purely control words and do not appear in the final program.

The final resultant program is similar after running BUILD and FRAME to my example FRAME2, each separate program having been allocated a block of line numbers in the 1000, 2000, 3000 series etc. And a module map appears at 18000 etc.

Naturally I was anxious to convert the Amstrad program to either XBAS or BBCBASIC but there was a snag. The Amstrad has a statement "LINE INPUT" which reads a complete line from a file up to a line feed and then skips to the next line. XBAS has no such facility but BBCBASIC has an "INPUT LINE". This however does'nt work with disc files and when the program came to that line it stopped awaiting a keyboard input, and was not suitable for working with disc files.

After many hours of patient attempts I finally got it to read a file and at the same time read the essential "END" and "FINISH" lines on which the program has to take special action. But, and here is a snag which appears to have no solution, the BBC Merge program, which is essential to the whole program in that it merges the new program on top of the original FRAME does'nt really merge. According to the BBCBASIC Manual it just

interleaves the two programs and where line numbers are duplicated it reproduces both line numbers.

I wrote two short test programs to see if this was in fact so and sure enough it does list both sets of line numbers but only works on the first. The only way to get rid of the unwanted lines is to renumber and then delete the unwanted lines. At this point I gave up! as the retention of the line numbers in the FRAME program are essential.

There was a possibility in that the Xitan XBASIC, which comes with the new Tatung CP/M disc has a proper merge, will be more suitable so perhaps I may eventually get it working....but that's another sad tale. But meanwhile I have BUILD working under Microsoft MBASIC on a KAYPRO.

#### CP/M PLUS TITAN XBASIC

Like new cars you should never order on the day they are announced you should never buy a new program, let someone else find all the bugs. However I did order a copy, mainly to take advantage of the still existing offer of 30% off on all software ordered before the end of '85. So on the 5th December I was the proud owner of CP/M and Xitan's XBASIC.

Looking through the Xitan Manual I was impressed by the facilities offered it has a complete range of MAT functions which must be a first. I hastily wrote a short test program for the MAT functions and found I could solve linear simultaneous equations in the time it took for my finger to come off the ENTER key. But strange as it may seem I can't find an every-day reason for solving such equations.

I then tried a few routine tests only to find that such functions as SIN(x), COS(x), SQR(x) etc brought up a fault condition and nothing I could do would make them work. In fact the XBASIC language as supplied on the disc was useless.

A hasty 'phone call to Tatung got the response that I should contact Xitan direct. Xitan responded that they do not normally deal with the end-user direct, but took my telephone number. I then wrote a letter to Tatung giving details of the faults and passed the matter to them, but at this time Xitan 'phoned asking for details of the fault so I agreed to send them a copy of the letter I wrote to Tatung.

After this a deadly silence lasting several weeks, and after a few telephone calls and a further letter Tatung said the matter had been passed to Xitan to be resolved. It took a letter to the MD of Tatung before I got a telephone call to say that Xitan had found the bug and that I should return the disc to Tatung for updating. This I did and I now have an XBASIC language disc which so far, appears to work.

Tatung have, throughout been less than helpful, in spite of the fact that I said my out-of-pocket expenses, eg 'phone calls, numerous letters and registered post to return the disc are in the order of some £15 no offer has been made to refund this nor, apart from a short letter, have I received any acknowledgement of any of my letters or that the fault that I complained of did in fact exist.

On the basis that troubles all come in threes it was no surprise to me when the Tatung colour monitor packed up. It is now back with Dixon's who promise a 2 to 3 week turn around. So that puts paid to two XBAS programs I was intending to include for this month's newsletter. The program may be of interest to those who may have, at Technical College, plotted a mixture of odd harmonic sine and cosine waves to prove (demonstrate?) that triangular and square waves are made up of such mixtures of third harmonics. Perhaps by the next newsletter I will have my monitor back.



## SCROLLING VRAM

The 80-column screen is divided into 8 pages, numbered 0 - 7 and accessed through ports 40 - 47 (hex.) respectively. The characters are held in a one-dimensional array in VRAM starting at the top left of the screen and moving across the row, then across the next row down, and so on. The screen offset is given by

$$\text{screen offset} = \text{row} \times (\text{chars per row}) + \text{column}$$

noting that the first row and column are numbered zero.

The first character on the screen can be at any address in VRAM, although in normal scrolling only addresses with a zero last hexadecimal digit are encountered. The offset of a character in VRAM is therefore given by

$$\text{VRAM offset} = \text{screen offset} + \text{start offset} [- 800\text{H}].$$

The VRAM is 2 Kbytes ( = 800H ) in size so if the result is negative 800H must be added, because the VRAM array is circular. When the screen start address is in the middle of VRAM, the end of the array wraps round to the start of VRAM. Thus at most times the bottom portion of the screen is at low VRAM addresses and the top portion at high addresses. This means that the whole screen can be scrolled instantaneously by changing the start address (as Paul Burgess showed) but it also means that it is not possible to scroll parts of the screen in this way unless MOS is altered.

Finally the VRAM address is obtained by

$$\text{VRAM address} = \text{VRAM offset} + 4000\text{H}.$$

The high byte is placed in the C register and the low byte in the B register: that is the bytes are switched before passing to the machine code port instructions.

It will be apparent that the VRAM size of 2048 bytes is not an exact multiple of the line length and in fact the screen effectively wraps round after 780H ( = 24 x 80 ) bytes. This leaves 80H bytes for a menu line (has anyone managed to display the last 30H bytes?). The menu line appears at the bottom of the screen and stays there when the screen is scrolled, because it is actually scrolled in VRAM and held just before the start address of the screen.

The parameters of the screen are stored as follows:

address (hex.)	bytes	
FB4A	1	cursor column
FB4B	1	cursor row
FB4C	2	screen start offset in VRAM
FB4E	1	used by 40-column screen
FB4F	1	characters per row

The values read will be the values at the time of access and are liable to change in the process of display on the screen (alternatively MCAL D1H,ZSCURS can be used). For further information on screen manipulation try the BBC Advanced User Guide.



One useful item which Tatung omitted from the MCAL descriptions is direct cursor addressing. This moves the cursor to any specified position using ASCII code 29 ( = 1DH = CTRL-J ) and is essential for customising CP/M programs. The control code must be followed by the column first and the row next. Try it in MOS: column 2 is of course input as CTRL-B from the keyboard. For column 0 try one of the (unset) function keys.

The listing shows a demonstration program to illustrate these points written in Nevada Pascal and machine code. This combination shows the interface better than most. The only unusual Pascal instructions are MAP which assigns an absolute address to a pointer and '+' which concatenates arrays. No optimisation has been carried out since this is liable to make the code harder to follow.

The basic loop is quite simple and is illustrated in procedure time which outputs the elapsed time to a portion of the menu line. Writing backwards facilitates the coding. Procedure moveline copies a line in a window to the next line up or down. The same loop is used but additional code is required to check for page changes and wrapping. However the main complication is that the sequence of instructions carried out by INI and OUTI is different, in contradiction to the Z80 manual: can anyone explain?

It is helpful to follow the changes in the registers when looking at machine code: compile the program with the first byte of code replaced by OFFH. When the program stops replace the original value using M (at the PC address). Use T to inspect the code and E to move to the next instruction. The screen display is of course destroyed. If the program seems a bit too complicated, you can always cheat by using CTRL-L ( 012H ) first. After this control code, the screen start address is zero and no wrapping occurs (until you produce line feed on the bottom line!).

The information given here is mainly based on deduction plus trial and error. If I have missed any points, further elaboration would be welcome.

C. P. WALLIS.  
23/1/86.

```
PROGRAM window_dressing;
    { demonstration of screen addressing and window scrolling }
CONST
    up = TRUE;
    down = FALSE; { refer to movement of text NOT apparent movement of window }
TYPE
    frame = RECORD
        toprow, bottomrow, leftcol, rightcol: INTEGER END;
VAR
    code: ARRAY[1..66] OF CHAR;
    movecode: ARRAY[1..52] OF CHAR;
    upcode, downcode: ARRAY[1..14] OF CHAR;
    picture: frame;
    i, j: INTEGER;
    screen_start: ^INTEGER;
    chars_per_row: ^CHAR;
```

# PROCEDURE load\_code;

*( sets up machine code in ARRAYs movecode, upcode & downcode )*

BEGIN *( PROCEDURE load\_code )*

```

movecode[1] := CHR(043H);      ( get parameters into correct registers )
movecode[2] := CHR(04AH);      ( LD B,E last address )
movecode[3] := CHR(05FH);      ( LD C,D note byte switch )
movecode[4] := CHR(016H);      ( LD E,A length of line )
movecode[5] := CHR(0);         ( LD D,0 )
movecode[6] := CHR(0E5H);      ( PUSH HL keep for write )
movecode[7] := CHR(0D5H);      ( PUSH DE )
movecode[8] := CHR(0C5H);      ( PUSH BC )

movecode[9] := CHR(03EH);      ( read line into buffer backwards )
movecode[10] := CHR(0FFH);     ( rloop: )
movecode[11] := CHR(0EDH);     ( LD A,0FFH address stop )
movecode[12] := CHR(0A2H);     ( read: )
movecode[13] := CHR(01DH);     ( INI get char. )
movecode[14] := CHR(028H);     ( DEC E )
movecode[15] := CHR(020H);     ( JR Z,dest ? end of line )
movecode[16] := CHR(0B8H);     ( CP B stop at B=0FFH )
movecode[17] := CHR(020H);     ( JR NZ,read next char )
movecode[18] := CHR(0F8H);     ( DEC C end of page )
movecode[19] := CHR(0DH);      ( LD A,C next port )
movecode[20] := CHR(079H);
movecode[21] := CHR(0FEH);
movecode[22] := CHR(040H);     ( CP 40H ? hit start of VRAM )
movecode[23] := CHR(030H);
movecode[24] := CHR(0F0H);
movecode[25] := CHR(0EH);
movecode[26] := CHR(047H);     ( JR NC,rloop )
movecode[27] := CHR(018H);     ( LD C,47H go to end of VRAM )
movecode[28] := CHR(0ECH);     ( JR rloop )

movecode[29] := CHR(0D1H);     ( write out from buffer backwards )
movecode[30] := CHR(0E1H);     ( load: POP DE get parameters back )
movecode[31] := CHR(04H);      ( POP HL )
movecode[32] := CHR(0EDH);     ( INC E !!!!! but essential )
movecode[33] := CHR(0A3H);     ( write: )
movecode[34] := CHR(020H);     ( OUTI write char. )
movecode[35] := CHR(08H);
movecode[36] := CHR(0DH);
movecode[37] := CHR(079H);
movecode[38] := CHR(0FEH);
movecode[39] := CHR(040H);
movecode[40] := CHR(030H);
movecode[41] := CHR(02H);
movecode[42] := CHR(0EH);
movecode[43] := CHR(047H);
movecode[44] := CHR(01DH);
movecode[45] := CHR(020H);
movecode[46] := CHR(0F1H);
movecode[47] := CHR(0C9H);

movecode[48] := CHR(021H);
movecode[49] := CHR(04FH);
movecode[50] := CHR(0FBH);
movecode[51] := CHR(0C1H);
movecode[52] := CHR(078H);

( write out from buffer backwards )
( load: POP DE get parameters back )
( POP HL )
( INC E !!!!! but essential )
( write: )
( OUTI write char. )
( JR NZ,wloop stop at B=0 )
( DEC C new port )
( LD A,C )
( CP 40H ? start of VRAM )
( JR NC,wloop )
( LD C,47H go to end of VRAM )
( wloop: DEC E )
( JR NZ,write ? any chars left )
( RET all written )
( set up address for writing )
( dest: )
( LD HL,0FB4FH chars_per_row addr. )
( POP BC )
( LD A,B source line addr. )

( subtract chars_per_row from address of source line end )

```

```

upcode[1] := CHR(096H);      { SUB (HL)      addr. for next line }
upcode[2] := CHR(047H);      { LD B,A      replace addr. }
upcode[3] := CHR(030H);
upcode[4] := CHR(0E4H);      { JR NC,load   use same port }
upcode[5] := CHR(0DH);       { DEC C      next port }
upcode[6] := CHR(079H);      { LD A,C }
upcode[7] := CHR(0FEH);
upcode[8] := CHR(040H);      { CP 40H      ? start of VRAM }
upcode[9] := CHR(030H);
upcode[10] := CHR(0DEH);     { JR NC,load }
upcode[11] := CHR(0EH);
upcode[12] := CHR(047H);     { LD C,47H    go to end of VRAM }
upcode[13] := CHR(018H);
upcode[14] := CHR(0DAH);     { JR load }

```

```

      { add chars_per_row to address of source line end }
downcode[1] := CHR(086H);    { ADD (HL)    addr. for next line }
downcode[2] := CHR(047H);    { LD B,A      replace addr. }
downcode[3] := CHR(030H);
downcode[4] := CHR(0E4H);    { JR NC,load   use same port }
downcode[5] := CHR(0CH);     { INC C      next port }
downcode[6] := CHR(079H);    { LD A,C }
downcode[7] := CHR(0FEH);
downcode[8] := CHR(048H);    { CP 48H      ? end of VRAM }
downcode[9] := CHR(038H);
downcode[10] := CHR(0DEH);   { JR C,load }
downcode[11] := CHR(0EH);
downcode[12] := CHR(040H);   { LD C,40H    go to start of VRAM }
downcode[13] := CHR(018H);
downcode[14] := CHR(0DAH);   { JR load }
END; { PROCEDURE load_code }

```

#### PROCEDURE time;

{ WARNING - if de or hl parameters are altered, and any character after the first has a zero last digit in the hex. address, a start of VRAM check is needed (as in moveline). Use 5 separate blocks of menu for simplicity }

#### VAR

```

reg16,reg_dump: RECORD
    af,bc,de,hl: INTEGER END;
timecode: ARRAY[1..6] OF CHAR;

```

#### BEGIN

```

timecode := 'm+] [I';      { some compilers accept code in this form }
timecode[1] := CHR(0EDH);   { write code to reserved ARRAY }
timecode[2] := CHR(0ABH);   { OUTD      write backwards }
timecode[3] := CHR(01DH);   { DEC E      adjust char count }
timecode[4] := CHR(020H);
timecode[5] := CHR(0FBH);   { JR NZ,start ? end of string }
timecode[6] := CHR(0C9H);   { RET      finished }
WITH reg16 DO BEGIN      { initialise registers }
    { screen offset = 24 * 80 + 79 = 1999 = 7CFH }
    bc := screen_start^ + 07CFH; { locate end of menu line }
    IF bc >= 0800H THEN bc := bc - 0800H; { VRAM offset }
    bc := bc + 4000H;        { last menu address }
    bc := (bc MOD 256)*256 + bc DIV 256; { inefficient byte switch }
    bc := bc + 256;         { yes, it does write to the next byte }
                           { no, you can't add before the switch }

```



```

de := 6;                { time string length }
hl := 0FB91H            { time string end }
END; { WITH reg16 }
CALL (ADDR (timecode),reg16,reg_dump) { returned values not required }
END; { PROCEDURE time }

```

```

PROCEDURE moveline( row: INTEGER; window: frame );
{ sets up parameters for machine code in STRING code and calls code }
{ row specifies screen row containing line in window to be copied }
{ code copies line in window to next line up or down }

```

```

VAR
  buffer: ARRAY[1..80] OF CHAR;
  register: RECORD CASE INTEGER OF
    1: ( f,a,c,b,e,d,l,h: CHAR );
    2: ( af,bc,de,hl: INTEGER ) END;

BEGIN { PROCEDURE moveline }
  WITH register DO BEGIN
    WITH window DO BEGIN
      a := CHR (rightcol-leftcol+1); { no. of chars }
      de := (row*ORD (chars_per_row^)) + rightcol; { screen offset }
      de := de + screen_start^ { VRAM offset }
      END; { WITH window }
      IF de>=0800H THEN de := de - 0800H;
      de := de + 4000H; { initial address }
      hl := ADDR (buffer) END; { WITH register }
      CALL (ADDR (code),register,register) { enter machine code }
      END; { PROCEDURE moveline }

```

```

PROCEDURE scroll_window( window: frame; up: BOOLEAN );
{ sets up lines to be moved and calls PROCEDURE moveline }

```

```

VAR
  i: INTEGER;

BEGIN { PROCEDURE scroll_window }
  WITH window DO
    IF up THEN BEGIN
      code := movecode + upcode;
      FOR i := toprow+1 TO bottomrow DO
        moveline (i,window) END { IF up }
      ELSE BEGIN { move down }
        code := movecode + downcode;
        FOR i := bottomrow-1 DOWNT0 toprow DO
          moveline (i,window) END { ELSE down }
      END; { PROCEDURE scroll_window }

```

```

BEGIN { PROGRAM window_dressing }
  MAP (screen_start,0FB4CH);
  MAP (chars_per_row,0FB4FH);
  FOR i := 1 TO 24 DO BEGIN
    WRITELN;
    FOR j := 1 TO 18 DO BEGIN { fill screen }
      WRITE (CHR (ORD ('@')+i),':');
      WRITE (CHR (ORD ('@')+i),'.') END { FOR j }
    END; { FOR i }

```



```

WITH picture DO BEGIN
    toprow := 5;
    bottomrow := 15;
    leftcol := 40;
    rightcol := 65;
    load_code;
    time;
    scroll_window (picture,down);
    WRITE (CHR (29),CHR (leftcol),CHR (toprow));    { move cursor }
    FOR j := leftcol TO rightcol DO
        WRITE ('c');
    scroll_window (picture,down);
    WRITE (CHR (29),CHR (leftcol),CHR (toprow));
    FOR j := leftcol TO rightcol DO
        WRITE ('d');
    scroll_window (picture,up);
    WRITE (CHR (29),CHR (leftcol),CHR (bottomrow));
    FOR j := leftcol to rightcol DO
        WRITE ('e') END; { WITH picture }
WRITE (CHR (30));    { home cursor }
time
END. { PROGRAM window_dressing }
    
```

#### FOR SALE

Einstein computer with twin drive, colour monitor, package software and Brother printer £500.00

Details available from Chris Giles on Brighton 419100.

No I'm not selling mine but an acquaintance finds he no longer has the time to devote to his Einstein and complete his studies.

#### HINTS & TIPS

##### APLHA LOCK TOGGLE

We have been asked how to toggle the Alpha Lock from basic in one of our letters (see letter page). In the scratch pad area which starts at Hex FB00 and continues to Hex FBFF the value in location FB3E determines whether an upper case or lower case character is registered. This can be accessed from basic by Peeking and Pokeing. The values found there are 8 for lower case and 136 for upper case. Therefore if the instruction PRINT PEEK(&FB3E) is issued then 8 will be returned if it is set to lower case and 136 will be returned for upper case. Likewise to set the Alpha Lock this location can be poked. POKE (&FB3E),8 will give lower case and POKE (&FB3E),136 will give upper case. An INP(&22) will toggle the Alpha Lock Light. ie if it is ON then it will go OFF and if it is OFF then it will go ON. So combining the two instructions within a program will duplicate the APLHA LOCK key. Provided the toggle and poke are both done together then they will stay in step and upper or lower case can be selected from within the program rather than relying on the operator.

34 (DEC)

## LETTERS OVER MY HEAD

Thank you for the newsletter. Very interesting, even if some of it went right over the top of my head! I'm actually writing to put a few questions to the club.

1. Are there any members who live in Cirencester or Swindon area? I am interested in everything except games.

2. When working in basic, if I open four random access files simultaneously I find the memory areas allocated to each file corrupt one another. (I am reading and writing randomly between the four files). Have I done something wrong?

3. How do I program the Alpha Lock key. I notice that some programs such as WDPRO are able to set the alphalock without intervention.

4. Do you know of a good book on Machine Code Programming for the Einstein that a simpleton could understand? All the ones I've seen so far leave me confused.

Thanks for the good works - I am still trying to think of a use for the maps of England and Ireland - will come up with something eventually.

Frank Skinner  
Cirencester.

### ANSWER

1. Keep reading the newsletter. We will be asking for volunteers to start up their own local user groups soon and we hope to be able to supply lists of people who are in your area who have Einsteins.

2. I DON'T KNOW. Any info readers??

3. See elsewhere in this issue under HINT & TIPS  
ALPHA LOCK TOGGLE

4. The short answer to this is NO. The best way into machine code is to look at as many Z80 machine code programs as possible and eventually it will click. But watch this space. We will be running articles from time to time on learning machine code. (If I get time the first will be in this issue)

A use for the MAPS?? We were hoping some enterprising person would pick up on it and write some Educational software or a weather routine??

### LONELY

I am interested in learning more about the National Einstein User Group. I read the piece in the Einstein News, Issue 1.

We seem to be a minority compared to the Sinclair or Commodore users and we would be better informed about the computer we have chosen if we club together and pool our information, agreed?

In the article it stated you were looking for groups around the country to join but I, as yet, don't know anyone at all who has had "Great Sense" and bought an Einstein. This letter was composed on WDPRO, I even like to use that.

## PAGE

I would appreciate it if you could put me in touch with my nearest User Group or maybe give me details how to join yours.

I also have a small, silly sounding, problem, I haven't sussed out how to load and link machine code programs to Crystal BASIC, daft isn't it but I am having a hard time understanding a couple of points from the user manual even after all this time.

Steve Carter (Now joined UKEUG)

### ANSWER

We are actually looking for volunteers to organise User Groups in their own area and we are getting to the stage whereby the list of contacts we have warrants a group in a particular area. If you want to volunteer for your area write to us and we will let you know if we have sufficient for your area.

### ALIVE AND KICKING

With reference to your letter in last weeks PCW you mentioned a rumour that the Einstein is being used as a development machine. While I was working for Ocean I co-wrote Kong Strikes Back with Mike Webb for the Amstrad 464. The source code was written on two Einsteins and transferred to the 464 using a simple interface.

After finishing Kong, we then wrote Street Hawk for the Spectrum, again on the Einsteins.

Mike Webb is now software development manager at another company and all the Amstrad and Spectrum software is developed on the Einstein and then transferred to the target machine using a parallel interface or RS232.

I am now a freelance programmer writing software on a contract basis. My first contract was with Elite in Birmingham to write a game that has since topped the charts - Commando on the Spectrum. Commando was written by myself and another freelance programmer, Keith Birkhill, with the help of one of Elites in house programmers. We all wrote the source on Einsteins and transferred the software to the Spectrum for testing.

So, as you see, the Einstein is far from dead up here in Manchester, at least as far as software development goes.

I hope this letter helps you to keep your enthusiasm for the Einstein as I believe it is a superb machine. Good luck with your user group.

Yours Sincerely  
NIGEL ALDERTON

### ANSWER

Well there isn't really any answer to that except "didn't we all do well to get one?", and I hope we hear a lot more from Nigel Alderton who knows what's what, being a professional.





