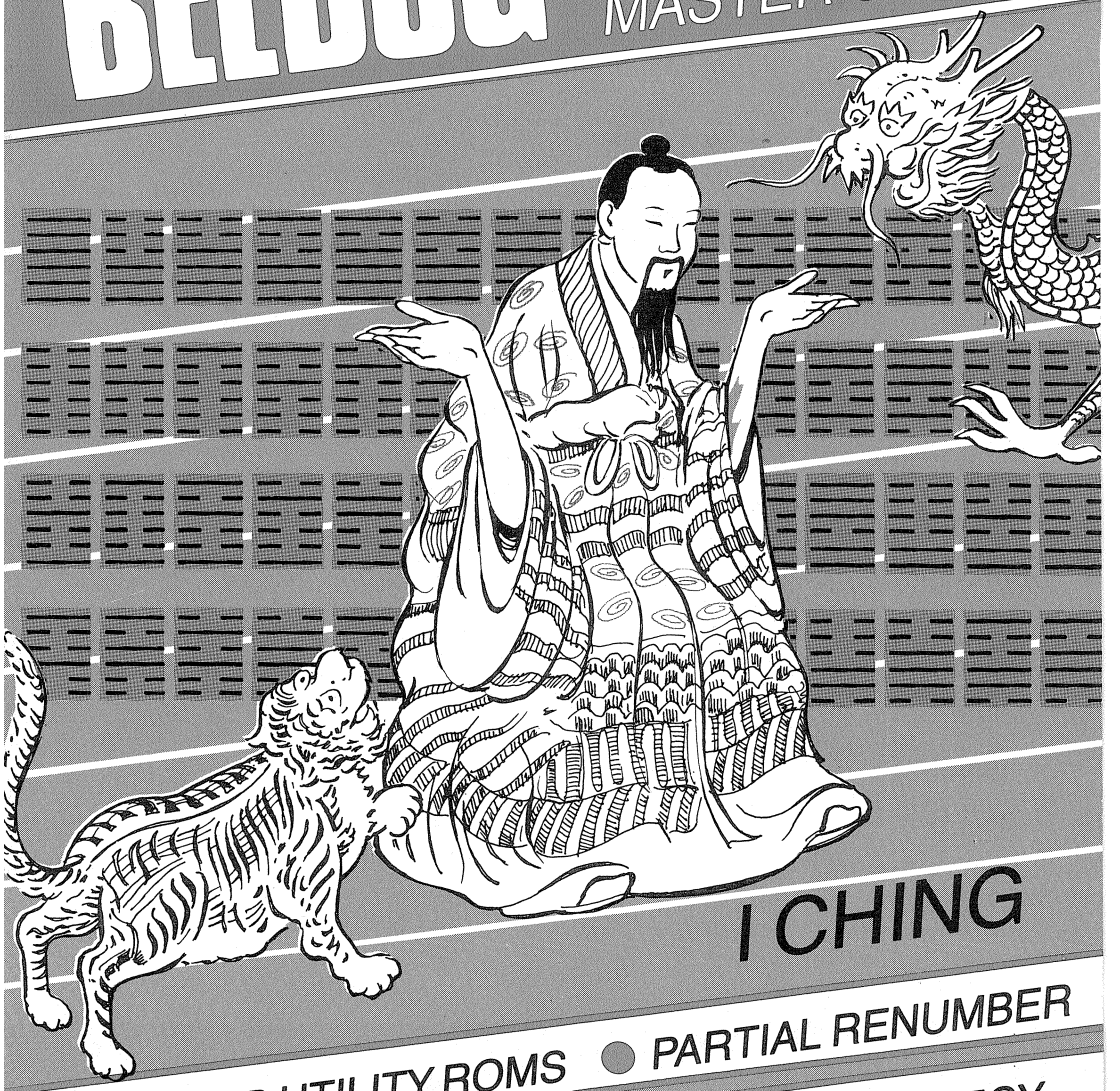
Vol.7 No.7 December 1988

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



I CHING

- PRINTER UTILITY ROMS
- PARTIAL RENUMBER
- A MONTHLY DESK DIARY
- GAME STRATEGY

BEEBUG Vol.7 No.7 December 1988

FEATURES

REVIEWS

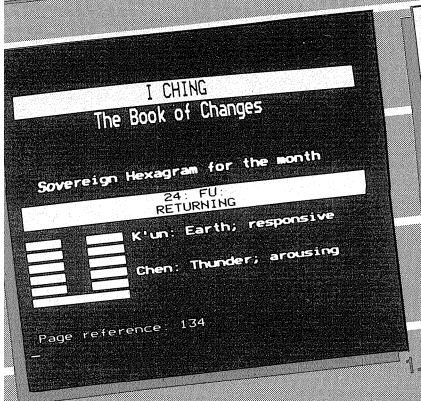
REGULAR ITEMS

HINTS & TIPS

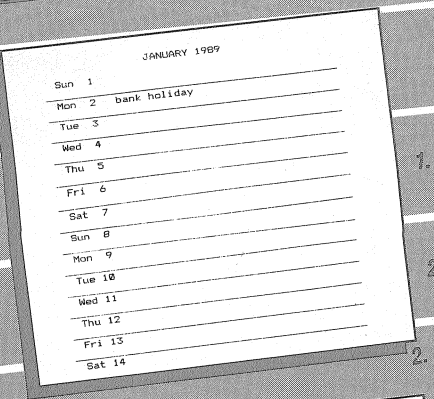
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

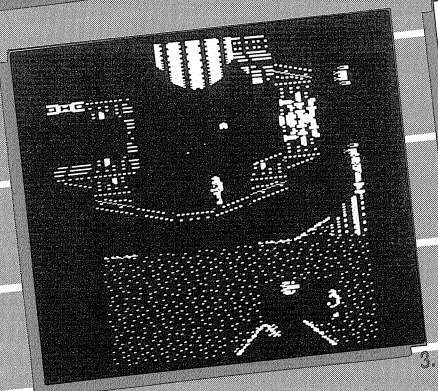
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



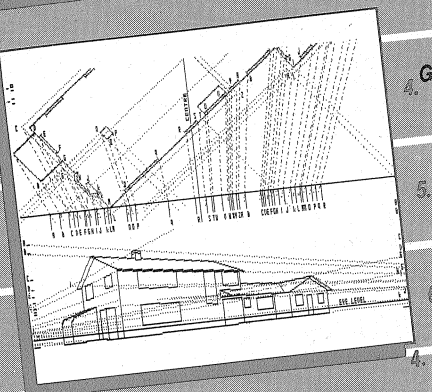
1.



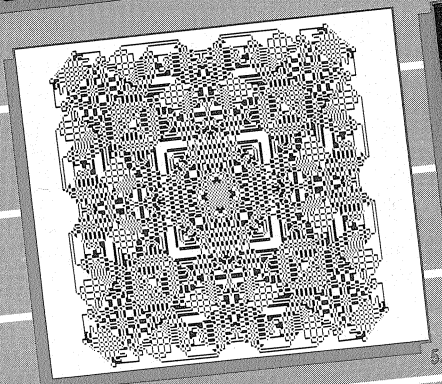
2.



3.



4.



5.

6.

1. I Ching
2. A Monthly Desk Diary
3. Exile
4. Graphic Design with ASTAAD
5. Symmetrical Patterns
6. Printer Utility ROMs

vailable on receipt of an A5 SAE), and are strongly dvised to upgrade to Basic II. Any second processor itted to the computer should be turned off before the rograms are run.

Where a program requires a certain configuration, his is indicated by symbols at the beginning of the rticle (as shown opposite). Any other requirements re referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

THE CHANGING WORLD OF BEEBUG

This issue of BEEBUG sees the conclusion of our highly informative series on *Using Assembler*, itself a follow-up to the previous set of articles entitled *Introducing Assembler*. Although assembler programming is not to everyone's taste, we hope that this series has gone some way to removing the mystique that can surround this subject, and that it has proved useful for those wanting a good introduction to assembler programming. If you have any particular comments, or ideas for further coverage on assembler, then why not let us know.

Commencing with the next issue of BEEBUG, we propose to introduce a new theme to the magazine, by publishing a number of educational programs. BBC micros have always played a prominent part in education, particularly in schools, and that role is as strong as ever. As a result, we feel that it would be appropriate, from time to time, to publish programs which clearly have an educational background or purpose, and the first such program is scheduled to appear next month.

If any readers have educational programs which they believe could be published in BEEBUG, and which would make a good feature, then do let us know. Likewise, if you have a good idea for an educational program, then we may be able to get this written. All original material published in BEEBUG will be paid for. We are not aiming for programs of too specialised a nature, and indeed, we are hoping that many of our readers will find these programs instructive and even entertaining.

BEST OF BEEBUG

Our two compilations of previously published programs have proved very successful (stocks are still available if you haven't yet ordered - see supplement for details). As a result we are planning more enterprises of a similar nature, and expect to announce the next of these in our January/February issue. Note that Magscan, our computerised bibliography to everything in BEEBUG, has now been updated to work on all BBC micros, and is complete up to the end of volume 6.

A reminder - the next BEEBUG is a two month issue covering both January and February.

News News News

BRIDGE OF CARDS

Colossus 4.0 Bridge, CDS Software's implementation of the traditional card game which is already well known on other micros, has been released for the Beeb. For the dedicated bridge player who knows about such things, *Colossus 4.0 Bridge* plays the Acol system and incorporates the Blackwood, Stayman and Baron conventions. If this means nothing to you then you may well benefit from the book *Begin Bridge* by G.C.H. Fox that is supplied free with each copy of the game. As well as a fast response time, *Colossus 4.0 Bridge* offers the option of saving and loading partially played games, and the replay option helps you learn from your mistakes. There is also an option to allow a particular set of hands to be entered. This is useful for solving bridge brain-teasers. *Colossus 4.0 Bridge* will work on the model B, Master 128 and Electron. The cassette version costs £11.99 inc. VAT, and the 5.25" disc version £14.99 inc. VAT. Further details are available from CDS Software, CDS House, Beckett Road, Doncaster DN2 4AD, tel. (0302) 21134.

CC SUPPORT FOR RISC OS

Computer Concepts Ltd., which is developing *Impulse*, its own alternative to Acorn's Archimedes' operating systems, has given RISC OS its seal of approval. Initially, CC claimed that the Arthur operating system was not sufficient to support the company's advanced applications software, and planned to release cut-down packages for use with Arthur, with the complete systems needing *Impulse*. However, Charles Moir of CC has said that he feels RISC OS is a great improvement over Arthur 1.20, and it should be possible to produce full versions of all the packages for running under RISC OS.

READ ALL ABOUT IT

Mc-Graw Hill publishers has just released a book for people wanting to know about the theory of computer systems, as well as the practical aspects of programming. The book, called *A Programmer's Introduction to Computer Systems Hardware and Software*, is written by two computer science lecturers from Reading university, and costs £10.95 from book shops. The 224 pages cover all aspects of computer

systems including microprocessors and peripherals. Further details can be obtained from Mc-Graw Hill Book Company, Shoppenhangers Road, Maidenhead, Berks SL6 2QL, tel. (0628) 23431/2.

EVENING ALL

As part of their crime prevention initiative, West Mercia Constabulary have released *Cop Shop*, a disc-based viewdata system. The 186 viewdata frames, which run under a cut down version of the Communitel display software, cover all aspects of crime prevention, and are aimed at children in particular. The subjects covered are grouped under six headings: Stranger Danger, Personal Safety, Self Defence, Car Watch, Home Watch, and Car and House Security. Some of these sections include interactive exercises that allow the user to decide how to respond to a situation, and to find out the effects of their actions.

Cop Shop will work on any model B, Master or Compact, and can be obtained free of charge by sending a blank formatted disc (5.25 or 3.5", DFS or ADFS) and an SAE to Inspector David Wornham, West Mercia Constabulary, Police Station, Rubery, Birmingham B45 9JA. People wanting to modify the viewdata pages, or to include them within a larger database, will need the complete Communitel host package.

ADVENTURING AROUND

Topologika, the company that took over all the Acornsoft adventure games, has released two packages aimed at children. The first is called *Whale Adventure*, and is a graphical adventure loosely based on the story of Moby Dick. As well as being a game, *Whale Adventure* aims to teach children in the 8-13 age group about the life of whales. *Whale Adventure* costs £18.40 inc. VAT on a 5.25" disc, or £20.70 inc. VAT on 3.5".

The second new package, called *Curves*, is aimed at children in the 9 to GCSE age group. As its name suggests, *Curves* teaches about the drawing of mathematical curves and loci. *Curves* is supplied on two discs with a tutorial audio cassette, an 80 page resource guide, and a 50 page activities book. The packages costs £33.44 inc. VAT on a 5.25" disc, and £36.74 inc. VAT on 3.5" disc.

Topologika are also working on a number of other educational packages, as well as a number of full-blown adventure games. Further details from Topologika, PO Box 39, Stilton, Peterborough PE7 3BR, tel. (0733) 244682.

CHEAP MICRONET

To boost the number of new subscribers, the Telemap Group are offering a special half price subscription to Micronet. From 1st January 1989, new subscribers will, for a limited period of three months, only have to pay £9.95 for their first quarter's use. This compares to the standard rate of £20 a quarter. The offer, which only applies if the subscription is paid for by direct debit, is in addition to the offer of a free modem to annual subscribers. Further details can be obtained by ringing 01-278 3143. If you already have a modem then you can log on to Prestel by dialling up 021-618 1111, and entering a user ID of 4444444444 and a password of 4444, to see a demonstration of Micronet.

ASK SID

Still on the subject of comms, Acorn has made its SID (Support Information Database) system more widely available. SID is a standard viewdata system that contains details of all Acorn products, along with various News and Reviews. Technical notes and other documents produced by Acorn can also be downloaded directly from SID.

Another very useful feature is the ability to send mailboxes to Acorn's customer support team. Using SID costs 8p (+VAT) a minute, in addition to normal 'phone charges, and there is a minimum charge of £11.50 inc. VAT per quarter. As a trial offer, the minimum charge is not applied for the first quarter after you join. Further details, and instructions on joining, can be obtained by logging on to the demonstration version of SID on (0223) 243642, or by writing to Customer Support, Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.

B

I Ching: The Book of Changes

Why not face the new year ahead by consulting the ancient Chinese oracle I Ching? Barry Thorpe explains.

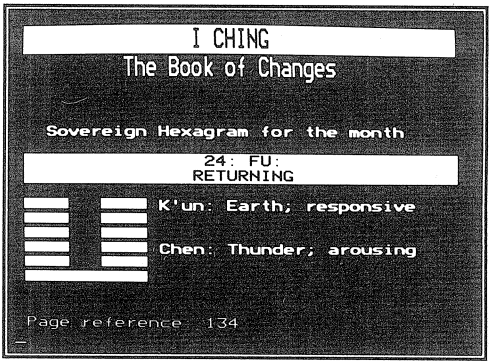
There is a wide interest these days in the art of divination by various means, whether by astrology, the Tarot or whatever. Software already exists for drawing up natal charts, and it is fiendishly complex.

The ancient Chinese method of divination, the *I Ching* (pronounced "yee jing"), is finding favour, and there are many attractively illustrated books on the subject. It is based on the principle of Yin and Yang, and originally one consulted it by casting yarrow stalks in a complex fashion to build up a hexagram, or six lines representing combinations of Yin and Yang. Yin lines are broken or yielding, while the Yang lines are solid and firm. It is also possible to use a simpler and quicker method of divination involving three coins, associated with professional fortune tellers more concerned about their fee than the accuracy of their interpretation.

Once a hexagram is arrived at, it becomes necessary to consult the associated *text* and *commentary upon the text*. For this you will need to refer to a suitable copy of *I Ching* (see reference at end).

Furthermore, under certain circumstances both Yin and Yang lines may become *moving* lines. A hexagram with moving lines leads to a second hexagram, and the text and commentary for this hexagram should also be studied in connection with that for the first hexagram.

According to published books on the subject, consulting the *I Ching* is not something to be undertaken lightly, and the whole process is imbued with ceremony designed to instil the appropriate state of well-being in the mind of the enquirer. This should be borne in mind when using the program listed here to simulate the method of divination.



USING THE PROGRAM

The program offers you an easier way of consulting the oracle, and furthermore the option of hardcopy. The program should be typed in and saved before running. The first prompt asks you for the number of the month, and then the date. You are then asked if you want hardcopy, though the screen display is the same in either case. The program checks whether the printer is on line or not.

Next you are asked whether you want a simulation of the yarrow stalks method, or the three coins method. The first method takes noticeably longer in the calculation.

The first hexagram is the sovereign hexagram for the month (in the Chinese calendar; the program makes the conversion). The next hexagram is the one more specific to the purpose of your consultation. If there are moving lines (shown as white) in this hexagram, a third hexagram is generated. When each hexagram has been displayed, press Return for the next one.

Once the hexagrams have been generated you should refer to the text and commentary as your *I Ching* text will tell you. As with all fortune telling, the responses require a certain

amount of interpretation to relate them to your original enquiry.

NOTES

1. The page references are to the book, "*I CHING: The Book of Change" by John Blofeld,* published by Unwin Paperbacks (1986) at £4.95. Ideally one would like to store the text on disc, and see the interpretation displayed, but there are copyright problems.

2. The consultation is supposed to be a calm, deliberate affair; the computer's instant generation of the hexagram is tempered by a delay loop in line 2050, which slows the display of each line. The value can be changed or the line omitted if desired. Incidentally, the lines are generated from bottom up according to the ancient methods, but are displayed top down on the screen for convenience.

3. The printout uses two of the graphics characters available on the Epson RX80; if your printer cannot cope, substitute, say, the hash (ASCII 35) for 140, and the asterisk (ASCII 42) for 139, in lines 1060 and 1070.

```
  10 REM Program ICHING
  20 REM Version B1.2
  30 REM Author  Barry Thorpe 1988
  40 REM BEEBUG  Dec 1988
  50 REM Program subject to copyright
  60 :
 100 MODE7:ON ERROR GOTO250
 110 PROCinit:PROCdisplay
 120 IF hardcopy PROCtestbuffer
 130 IF hardcopy PROChardcopytitle
 140 PROCinsertritnums
 150 PROCfindhexagram:PROCprintlines(-1
,"Sovereign Hexagram for the month")
 160 REPEAT UNTIL GET=13
 170 IF stalks THEN source$="yarrow sta
lks":PROCstalks ELSE source$="coins":PRO
Ccoins
 180 PROCfindhexagram:PROCprintlines(0,
"Hexagram from the "+source$)
 190 REPEAT UNTIL GET=13
 200 IF oldy PROCmove:PROCfindhexagram:
PROCprintlines(-1,"Hexagram from moving
lines")
 210 REPEAT UNTIL GET=13
 220 MODE7
 230 END
 240 :
 250 MODE7:IF ERR<>17 REPORT:PRINT" at
line ";ERL
 260 END
 270 :
1000 DEF PROCinit
1010 width%=39:dot$=STRING$(60,".")
1020 DIM trinum%(2),trinam$(7),hexptr%(
7,7),ritn%(5),month%(12),inhand%(3)
1030 ritn$="6789":inhand$="25211713"
1040 sovhex$="888777777888"
1050 bar$=STRING$(11,CHR$47)
1060 ypbar$=STRING$(11,CHR$140)
1070 opbar$=STRING$(11,CHR$139)
1080 yang$=bar$:opyang$=opbar$
1090 ypyang$=ypbar$
1100 opyin$=LEFT$(opbar$,4)+"   "+LEFT$
(opbar$,4):ypyin$=LEFT$(ypbar$,4)+"   "+
LEFT$(ypbar$,4)
1110 yin$=LEFT$(bar$,4)+"   "+LEFT$(bar
$,4)
1120 hexorder$="74210356"
1130 FOR t=0 TO 7:READ trinam$(t):NEXT
1140 DATA K'un: Earth; responsive,Ken:
Mountain; immovable,K'an: Water; dangero
us,Sun: Wind; gentle,Chen: Thunder; arou
sing,Li: Fire; clinging,Tui: Lake; joyfu
l,Ch'ien: Heaven; active
1150 FOR row=0 TO 7
1160 r=VAL(MID$(hexorder$,row+1,1))
1170 FOR col= 0 TO 7
1180 c=VAL(MID$(hexorder$,col+1,1))
1190 READ hexptr%(r,c)
1200 NEXT col,row
1210 FOR m%=0 TO 12:READmonth%(m%):NEXT
1220 dm$="312931303130313130313031"
1230 mn$="JanFebMarAprMayJunJulAugSepOc
tNovDec"
1240 ENDPROC
1250 :
1260 DEF PROCdisplay
1270 C1$=CHR$134+CHR$157+CHR$132+CHR$14
1
1280 C2$=CHR$132+CHR$157+CHR$134+CHR$14
1
1290 PRINT C1$TAB(16)"I CHING"
1300 PRINT C1$TAB(16)"I CHING"
1310 PRINT C2$TAB(10)"The Book of Chang
es"
1320 PRINT C2$TAB(10)"The Book of Chang
```

```
es"
 1330 VDU28,0,24,39,7
 1340 PROCdate
 1350 PRINT''CHR$131"Would you like to c
ast the stalks or"'CHR$131"use the coins
?"
 1360 PRINT'CHR$134"Type S or C";
 1370 REPEAT:A$=GET$:UNTIL INSTR("SCsc",
A$)
 1380 stalks=INSTR("Ss",A$)>0
 1390 ENDPROC
 1400 :
 1410 DATA 1,34,5,26,11,9,14,43
 1420 DATA 25,51,3,27,24,42,21,17
 1430 DATA 6,40,29,4,7,59,64,47
 1440 DATA 33,62,39,52,15,53,56,31
 1450 DATA 12,16,8,23,2,20,35,45
 1460 DATA 44,32,48,18,46,57,50,28
 1470 DATA 13,55,63,22,36,37,30,49
 1480 DATA 10,54,60,41,19,61,38,58
 1490 :
 1500 DATA 106,205,307,406,506,607,708,8
08,908,1009,1108,1207,1306
 1510 :
 1520 DEF PROCdate:CLS
 1530 PRINT CHR$131"Enter the month numb
er: ";
 1540 REPEAT
 1550 INPUTTAB(25,0)"  "TAB(25,0) m%
 1560 UNTIL m%>0 AND m%<13:mn%=m%
 1570 dm%=VAL(MID$(dm$,(m%-1)*2+1,2))
 1580 PRINT CHR$131"and the date: ";
 1590 REPEAT
 1600 INPUTTAB(15,1)"  "TAB(15,1) d%
 1610 UNTIL d%>0 AND d%<=dm%
 1620 date%=m%*100+d%
 1630 m%=0:found=FALSE
 1640 REPEAT
 1650 IF date%>=month%(m%) AND date%<mon
th%(m%+1) THEN found=TRUE
 1660 m%=m%+1
 1670 UNTIL found OR m%>11
 1680 m%=m%-1:IF m%=0 m%=12
 1690 PRINT'CHR$131"Hardcopy Y/N?";:REPE
AT:A$=GET$:UNTIL INSTR("YyNn",A$)
 1700 hardcopy=INSTR("Yy",A$)>0
 1710 ENDPROC
 1720 :
 1730 DEF PROCcoins
 1740 oldy=FALSE
 1750 FOR ln=5 TO 0 STEP-1
 1760 ritnum=0
 1770 FOR coin=1 TO 3
 1780 IF RND(100)>50 THEN val=3 ELSE val
=2
 1790 ritnum=ritnum+val
 1800 NEXT
 1810 IF ritnum=6 OR ritnum=9 THEN oldy=
TRUE
 1820 ritn%(ln)=ritnum
 1830 NEXT ln
 1840 ENDPROC
 1850 :
 1860 DEF PROCfindhexagram
 1870 t=2:p%=4:trinum%(1)=0:trinum%(2)=0
 1880 FOR ln=5 TO 0 STEP -1
 1890 IF ritn%(ln)=7 OR ritn%(ln)=9 trin
um%(t)=trinum%(t)+p%
 1900 p%=p%DIV2:IF p%=0 p%=4:t=t-1
 1910 NEXT
 1920 h%=hexptr%(trinum%(2),trinum%(1))
 1930 RESTORE(9000+h%):READhexagram$,pn%
 1940 colon=INSTR(hexagram$,":"):head1$=
STR$(h%)+": "+LEFT$(hexagram$,colon):L1%
=LENhead1$
 1950 head2$=MID$(hexagram$,colon+2):L2%
=LENhead2$
 1960 ENDPROC
 1970 :
 1980 DEF PROCprintlines(moving,title$)
 1990 CLS:L%=LEN(title$):PRINTTAB((width
%-L%)DIV2)title$'
 2000 VDU134,157,132
 2010 PRINTTAB((width%-L1%-4)DIV2)head1$
 2020 VDU134,157,132:PRINTTAB((width%-L2
%-4)DIV2)head2$'
 2030 t=1
 2040 FOR ln=0 TO 5
 2050 TIME=0:REPEAT UNTIL TIME>100
 2060 IF ritn%(ln)=6 OR ritn%(ln)=9 THEN
 col=151 ELSE col=146
 2070 IF NOT moving THEN PRINTCHR$(col);
 ELSE PRINTCHR$(147);
 2080 IF ritn%(ln)=7 OR ritn%(ln)=9  PRI
NTyang$;ELSE PRINTyin$;
 2090 IF ln=0 OR ln=3 PRINTTAB(12)CHR$(1
35)trinam$(trinum%(t)):t=t+1 ELSE PRINT
 2100 NEXT
 2110 PRINT''CHR$130"Page reference: ";p
n%
 2120 IF hardcopy PROChardcopyhex
 2130 ENDPROC
 2140 :
 2150 DEF PROCmove
 2160 FOR ln=0 TO 5
 2170 IF ritn%(ln)=9 ritn%(ln)=8
```

```
 2180 IF ritn%(ln)=6 ritn%(ln)=7
 2190 NEXT
 2200 ENDPROC
 2210 :
 2220 DEF PROCinsertritnums
 2230 FOR ln=0 TO 5
 2240 ritn%(ln)=VAL(MID$(sovhex$,m%,1))
 2250 m%=m%+1:IF m%>12 m%=1
 2260 NEXT
 2270 ENDPROC
 2280 :
 2290 DEF PROChardcopyhex
 2300 VDU2,21:PRINT'dot$'
 2310 PRINTTAB((1.5*width%-L%)DIV2)title
$'
 2320 PRINTTAB((1.5*width%-L1%)DIV2)head
1$ 'TAB((1.5*width%-L2%)DIV2)head2$'
 2330 t=1
 2340 FOR ln=0 TO 5
 2350 IF ritn%(ln)=6 PRINT opyin$;
 2360 IF ritn%(ln)=7 PRINT ypyang$;
 2370 IF ritn%(ln)=8 PRINT ypyin$;
 2380 IF ritn%(ln)=9 PRINT opyang$;
 2390 IF ln=0OR ln=3 PRINTSPC(5) trinam$
(trinum%(t)):t=t+1 ELSE PRINT
 2400 NEXT
 2410 PRINT'''"Page reference: ";pn%'':VD
U6,3
 2420 VDU3
 2430 ENDPROC
 2440
 2450 DEF PROChardcopytitle
 2460 VDU2,1,27,1,109,1,4,1,27,1,108,1,1
0,1,27,1,69
 2470 VDU21
 2480 VDU1,14:PRINTTAB(2)"I CHING: the b
ook of changes"''
 2490 PRINT"Date: ";d% " "MID$(mn$,(mn%-
1)*3+1,3)
 2500 VDU6,3
 2510 ENDPROC
 2520 :
 2530 DEF PROCstalks
 2540 oldy=FALSE
 2550 FOR ln=5 TO 0 STEP -1
 2560 nstalks=49
 2570 FOR cast=1 TO 3
 2580 lfheap=nstalks DIV 3+RND(nstalks D
IV3):rtheap=nstalks-lfheap
 2590 inhand%(cast)=1:rtheap=rtheap-1:ns
talks=nstalks-1
 2600 leftpick=FNreduce(lfheap):rightpic
k=FNreduce(rtheap)
 2610 inhand%(cast)=inhand%(cast)+leftpi
ck +rightpick
 2620 NEXT cast
 2630 hful$=STR$(inhand%(1)+inhand%(2)+i
nhand%(3))
 2640 ritn%(ln)=VAL(MID$(ritn$,(1+INSTR(
inhand$,hful$))DIV2,1))
 2650 IF ritn%(ln)=6 OR ritn%(ln)=9 THEN
 oldy=TRUE
 2660 NEXT ln
 2670 ENDPROC
 2680 :
 2690 DEF FNreduce(leftover)
 2700 IF leftover<=4 =leftover
 2710 leftover=leftover MOD 4
 2720 nstalks=nstalks-leftover
 2730 =leftover
 2740 :
 2750 DEF PROCtestbuffer
 2760 REPEAT
 2770 *FX21,3
 2780 probe1=ADVAL(-4):VDU2,1,0,1,0,3:pr
obe2=ADVAL(-4)
 2790 printerconnected=probe1=probe2
 2800 IF NOT printerconnected THEN CLS:P
RINTTAB(8,10)CHR$129 CHR$136"please conn
ect printer"''TAB(8)"Press Return to con
tinue":REPEAT UNTIL GET=13
 2810 UNTIL printerconnected
 2820 ENDPROC
 2830 :
 9000 REM names of hexagrams
 9001 DATA CH'IEN: CREATIVITY,85
 9002 DATA K'UN: QUIESCENCE,90
 9003 DATA CHUN: BIRTH PANGS,94
 9004 DATA MENG: INEXPERIENCE,96
 9005 DATA HSU: BIDING ONE'S TIME,98
 9006 DATA SUNG: STRIFE,100
 9007 DATA SHIH: THE ARMY,102
 9008 DATA PI: UNITY,104
 9009 DATA HSIAO CH'U: RESTRAINT BY THE
WEAK,106
 9010 DATA LU: TREADING,108
 9011 DATA T'AI: PEACE,110
 9012 DATA P'I: STAGNATION,112
 9013 DATA T'UNG JEN: FELLOWSHIP,114
 9014 DATA TA YU: WEALTH,116
 9015 DATA CH'IEN: MODESTY,118
 9016 DATA YU: CALM CONFIDENCE,119
 9017 DATA SUI: FOLLOWING,121
 9018 DATA KU: DECAY,123
 9019 DATA LIN: GETTING AHEAD,125
```

Continued on page 20

Form Designs From Mewsoft

David Somers describes the latest enhancements to his personal organiser, and an A4 form designer, all from Mewsoft.

In BEEBUG Vol.6 No.8 I reviewed Fax*File from Mewsoft, a program intended to fill your personal organiser with all manner of specially designed forms and layouts printed on pre-punched and correctly-sized paper. Just over a year on we look at the latest extensions to Fax*File, and its burgeoning offshoot, a full A4 forms design package.

Product	**Fax*File Extension Disc**
Supplier	**MEWsoft, 11 Cressey Road, London NW3 2NB. Tel. 01-267 2642.**
Price	**£9.95 inclusive**

The Extension disc for Fax*File comes as a DFS disc containing four programs: Label Printer, Short Form Printer, Database, and Calendar. For ADFS users, there is also now an ADFS version containing the original Fax*File and these four extension programs (price £22.09; upgrade from DFS to ADFS £9.96).

```
ADDRESS BOOK
-------------------------------------------
Name     A B Designs
Address  81 Sutton Common Road
         Sutton              ☎ 01-
         Surrey                644 6643
-------------------------------------------
Name     Acorn Computers
Address  645 Newmarket Road
         Cambridge           ☎ 0223-
         CB5 8PD               214411
-------------------------------------------
Name     BBC Telesoftware
Address  BBC TV Centre
         Wood Lane           ☎ 01-
         London W12 7RJ        576 7481
-------------------------------------------
Name     BEEBUG
Address  Dolphin Place, Holywell Hill
         St. Albans          ☎ 0727-
         Herts. AL1 1EX        40303
-------------------------------------------
```

Address Book

LABEL PRINTER

The Label Printer enables details of names and addresses stored in the Address Book to be printed out, ready formatted for address labels. Support is provided for multiple column labels.

SHORT FORM PRINTER

This utility, like the above, uses information previously stored in the Address Book. It enables selected information from the entries to be printed, to produce, for example, a printout of just the names and telephone numbers. The fields for printing are user-selectable, and can be saved to disc for future use. A preview facility shows the form as it will be printed, which can save a lot of wasted paper!

```
                CALENDAR FOR 1989

        JANUARY                              JULY
Su Mo Tu We Th Fr Sa                 Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7                                    1
 8  9 10 11 12 13 14                  2  3  4  5  6  7  8
15 16 17 18 19 20 21                  9 10 11 12 13 14 15
22 23 24 25 26 27 28                 16 17 18 19 20 21 22
29 30 31                             23 24 25 26 27 28 29
                                     30 31

       FEBRUARY                             AUGUST
Su Mo Tu We Th Fr Sa                 Su Mo Tu We Th Fr Sa
          1  2  3  4                                 1  2  3  4  5
 5  6  7  8  9 10 11                  6  7  8  9 10 11 12
12 13 14 15 16 17 18                 13 14 15 16 17 18 19
19 20 21 22 23 24 25                 20 21 22 23 24 25 26
26 27 28                             27 28 29 30 31

        MARCH                             SEPTEMBER
Su Mo Tu We Th Fr Sa                 Su Mo Tu We Th Fr Sa
          1  2  3  4                                       1  2
 5  6  7  8  9 10 11                  3  4  5  6  7  8  9
12 13 14 15 16 17 18                 10 11 12 13 14 15 16
19 20 21 22 23 24 25                 17 18 19 20 21 22 23
26 27 28 29 30 31                    24 25 26 27 28 29 30
```

Calendar

DATABASE

As personal organisers are used, often as not, to carry vast amounts of information, it seems only right that a Database program should be included. This is not a fully-fledged database, with lots of wonderful and esoteric facilities, but instead one that is simple and easy to use. Up to four *fields* are allowed, with a maximum of sixty records. A sort, either alphabetic or

numeric, can be applied using one or two fields. Again, a preview facility is available before printing out.

CALENDAR

The Calendar program enables a calendar to be printed on a single page (to fit your organiser), as two columns of six months.

Product	**A4 Forms Designer**
Supplier	**MEWsoft**
Price	**£9.95 inclusive.**

A Forms Designer was always supplied with the Fax*File, enabling customised forms to be created and printed to fit your organiser. Taking the concept further, this has now evolved into the A4 Forms Designer, which as its name implies, allows forms of up to A4 size (vertically) to be produced.

To use this program it is necessary to have either a Master 128, Master Compact, Model B with Shadow RAM, or an Archimedes.

Like the Fax*File suite of programs, the A4 Forms Designer uses a window environment, enabling forms to be designed and edited. The screen cannot show the whole form at once, and so it scrolls to reveal the hidden parts.

EDITING A FORM

Operation is fairly simple, with the cursor being moved over the page using the cursor control keys, and text simply entered as and where required. A line drawing mode enables lines to be draw on the form wherever the cursor travels. When lines cross, it is, however, necessary to press a function key to *join* them lines together properly. When creating complex tables, this can become rather tedious and time consuming.

BLOCK OPERATIONS

A block cut-and-paste facility can help reduce the time to create regular forms, while data can be imported from text files onto a form at marked places. This leads to the ability to produce a *template* form, and import data for formatted printing.

SIDEWAYS MOTION

An extended version of the A4 Forms Designer is available (for £19.90) offering the ability to produce forms in landscape mode (A4 viewed horizontally). It only works under the ADFS on a Master 128, Compact, or Archimedes. The data import facility is slightly enhanced, with the ability to print multiple forms with different data imported for each one, and is an improvement over the original version of the Fax*File Form Designer.

CALCULATING FORMS

An addition to the A4 Form Designer is the A4 Calculating Form (price £9.95). As the name implies, it offers the ability for mathematical operations to be applied to information on a form. This provides a *spreadsheet* type of facility, which could prove quite useful.

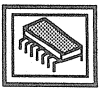
FINAL COMMENTS

Both packages are straightforward, if a little esoteric, in operation, but perform their jobs well, and the printout as you can see from the accompanying samples is excellent. It is alright to have a package offering lots of facilities, but how many of them are used? With these programs, the few facilities provided perform their tasks competently. I suspect it is their lack of numerous functions that makes them appealing: just simple, straightforward, and easy to use.

Although the packages are user-friendly, there is one niggling point. At various times it is necessary to enter the name of a data file, etc. If this is wrong, a list of possible files is displayed, then one has to be entered. Given the windowing environment used, it is odd that selection by simply placing the cursor over the list of files is not allowed. Instead you need to memorise the name and then enter it.

All the above-mentioned programs are supplied with an A5 instruction manual, and software on 5.25" disc. For 3.5" discs, add £1.00 to the appropriate price. B

Partial Renumber



Save hours of tedious work with this utility by David Spencer.

There are many occasions when it would be nice to be able to renumber just part of a program. For example, to achieve a particular numbering convention (as in BEEBUG), or to open up a gap to insert additional lines. The *BEEBUG Partial Renumber* utility allows just this.

The program is in the form of a ROM image that must be loaded into sideways RAM. Start by entering the listing given below and saving it. Running the program will assemble the image and save it with the name 'RENUMobj'. The method for loading the image depends on the sideways RAM in use, but for a Master or Compact the command:

```
*SRLOAD RENUMobj 8000 WQ
```

will suffice. Once the image is loaded press Ctrl-Break to initialise it. Otherwise refer to the instructions supplied with your sideways RAM.

USING THE BEEBUG PARTIAL RENUMBER

The renumber utility offers two new star commands. These are:

```
*RENUMBER [<first>],[<last>],[<start>],
[<inc>],[F]
```

and

```
*CHECKORDER [<first>],[<last>]
```

The first of these, *RENUMBER, can take up to five optional parameters. The first two are the numbers of the first and last lines to be renumbered. If the first is missed out, then the start of the program is assumed, and if the second is omitted then the end of the program is used. The third parameter is the new line number to start at, with 10 being assumed if this parameter is omitted. The fourth number is the increment between line numbers, again with a default of 10. The final parameter is an optional 'F', the function of which is described below. If any value other than the last is omitted, then commas must be used to indicate this. For example:

```
*RENUMBER 100,190,,25
```

would renumber the lines between 100 and 190 starting at 10 and going up in increments of 25. Similarly,

```
*RENUMBER ,1000,10000
```

will renumber all the lines up to 1000, to start from 10000, in increments of 10.

LINE ORDER

Obviously, with a utility like this it is very easy to end up with lines out of order. In other words, the line numbers do not necessarily increase from one line to the next. If the parameters given to *RENUMBER would result in the line numbers being out of order, then the command prompts for confirmation before continuing. This can be overridden by putting an 'F' on the end of the command, in which case a warning will still be issued, but no confirmation is sought. For example:

```
*RENUMBER 100,200,1000,10 F
```

The *CHECKORDER command can be used to check the line numbers in a program. It will list any points at which the line number does not increase from one line to the next. The two offending numbers are displayed, separated by a dash. For example, if line 990 came immediately after line 1000, the output would be:

```
1000 - 990
```

*CHECKORDER can take optional start and end line numbers to restrict the checking to only part of the program.

If programs lines are out of order, then *RENUMBER should be used to rectify this before any changes are made to the program.

```
 10 REM Program Partial Renumber
 20 REM Author  David Spencer
 30 REM Version B1.0
 40 REM BEEBUG  December 1988
 50 REM Program subject to copyright
 60 :
100 page=&18:lin=&70:min=&72
110 lptr=&74:temp=&76:start=&78
120 after=&7A:curl=&7C:ytemp=&7E
130 inc=&7F:flag=&80:bstart=&81
140 bend=&83:doflag=&85:line=&86
150 sflag=&88
160 osrdch=&FFE0:osnewl=&FFE7
170 oswrch=&FFEE:osbyte=&FFF4
180 DIM code 3000
190 FORpass=4 TO 7 STEP3
200 P%=&8000:O%=code
210 [OPT pass
220 EQUB 0:EQUB 0:EQUB 0:JMP service
230 EQUB &82:EQUB cw AND &FF:EQUB 1
240 EQUS "BEEBUG Partial Renumber"
```

```
 250 .cw EQUB 0:EQUS "(C) BEEBUG 1988"
 260 EQUB 0:.service CMP #4:BEQ comm
 270 RTS:.comm TYA:PHA:LDX #0
 280 .comm2 LDA (&F2),Y:AND #&DF
 290 CMP ctab,X:BNE comm4:INY:INX
 300 LDA ctab,X:BPL comm2:LDA (&F2),Y
 310 CMP #ASC"A":BCS nextc
 320 .comm3 JSR sskp:LDA ctab,X:PHA
 330 LDA ctab+1,X:PHA:RTS
 340 .comm4 LDA (&F2),Y:INY:CMP #ASC"."
 350 BNE nextc:.comm5 INX:LDA ctab,X
 360 BPL comm5:BMI comm3:.nextc DEX
 370 .nextc2 INX:LDA ctab,X:BPL nextc2
 380 INX:INX:LDA ctab,X:BNE nextc3
 390 PLA:TAY:LDX &F4:LDA #4:RTS
 400 .nextc3 PLA:PHA:TAY:JMP comm2
 410 .cout PLA:TAY:LDX &F4:LDA #0:RTS
 420 .ctab EQUS "RENUMBER"
 430 EQUB (rnum-1) DIV &100
 440 EQUB (rnum-1) MOD &100
 450 EQUS "CHECKORDER"
 460 EQUB (chkord-1) DIV &100
 470 EQUB (chkord-1) MOD &100:EQUB 0
 480 .lookhigh SEC:JSR look:LDA lptr
 490 STA bend:LDA lptr+1:STA bend+1:RTS
 500 .looklow CLC:LDA #&FF:STA after
 510 STA after+1:JSR look:LDA lptr
 520 STA bstart:LDA lptr+1:STA bstart+1
 530 RTS
 540 .look PHP:STY ytemp:JSR setptr
 550 .llow LDY #0:STY temp:INY
 560 LDA (lptr),Y:CMP lin:BNE llow1
 570 DEC temp:.llow1 DEY:LDA (lptr),Y
 580 SBC lin+1:BCC llow2:BNE llow15
 590 BIT temp:BPL llow15:PLP:PHP
 600 BCS llow2:.llow15 LDY ytemp:PLP
 610 RTS:.llow2 PLP:PHP:BCS llow3
 620 LDA (lptr),Y:STA after+1:INY
 630 LDA (lptr),Y:STA after
 640 .llow3 JSR nline:JMP llow
 650 .nline LDY #2:LDA (lptr),Y:CLC
 660 ADC lptr:STA lptr:BCC nline2
 670 INC lptr+1:.nline2 LDY #0:RTS
 680 .sskp DEY:.sskp2 INY:LDA (&F2),Y
 690 CMP #ASC" ":BEQ sskp2:RTS
 700 .nget LDA #0:STA lin:STA lin+1
 710 .nget2 JSR sskp:CMP #ASC","
 720 BNE nget3:INY:.ngeto LDA #0:RTS
 730 .nget3 JSR digchk:BCC ngeto
 740 STA lin:.nget4 INY:LDA (&F2),Y
 750 JSR digchk:BCS nget5:JSR sskp
 760 CMP #ASC",":BNE nget45:INY
 770 .nget45 LDA #&FF:SEC:RTS
 780 .nget5 PHA:LDA lin+1:PHA:LDA lin
 790 PHA:ASL lin:ROL lin+1:BCS toobig
 800 ASL lin:ROL lin+1:BCS toobig:PLA
 810 ADC lin:STA lin:PLA:ADC lin+1
 820 STA lin+1:BCS toobig:ASL lin
 830 ROL lin+1:BCS toobig:PLA:ADC lin
 840 STA lin:BCC nget6:INC lin+1
 850 .nget6 LDA lin+1:BPL nget4
 860 .toobig LDX #0
 870 .error LDA #0:STA &100:LDA err,X
 880 STA &101:INX:LDY #0
 890 .error2 LDA err,X:STA &102,Y:INX
 900 INY:CMP #0:BNE error2:JMP &100
 910 .err EQUB 100
 920 EQUS "Line number too large"
 930 EQUB 0
 940 .berr EQUB &FE:EQUS "Bad command"
 950 EQUB 0
 960 .binc EQUB 101
 970 EQUS "Bad increment":EQUB 0
 980 .bqual:EQUB 102
 990 EQUS "Bad qualifier":EQUB 0
1000 .borr EQUB 103:EQUS "Bad range"
1010 EQUB 0
1020 .digchk CMP #&30:BCC digno
1030 CMP #&3A:BCS digno:SEC:AND #&F:RTS
1040 .digno CLC:RTS
1050 .rnum JSR cprog:LDA #10:STA start
1060 STA inc:LDA #0:STA start+1
1070 STA flag:JSR getsf:JSR nget
1080 BEQ nost:LDA lin:STA start
1090 LDA lin+1:STA start+1
1100 .nost JSR nget:BEQ noi:LDA lin
1110 STA inc:LDA lin+1:BEQ noi
1120 LDX #binc-err:JMP error
1130 .noi JSR sskp:BCC pgot2:AND #&DF
1140 CMP #ASC"F":BEQ pgot1
1150 LDX #bqual-err:JMP error
1160 .pgot1 DEC flag:INY
1170 .pgot2 JSR chkend:LDA start
1180 STA line:LDA start+1:STA line+1
1190 LDA #0:STA doflag:JSR renum
1200 LDA after:AND after+1:CMP #&FF
1210 BEQ noaf:LDA after:CMP start
1220 LDA after+1:SBC start+1:BCS ohdear
1230 .noaf LDY #1:LDA line:CMP (lptr),Y
1240 DEY:LDA line+1:SBC (lptr),Y
1250 BCC rnum2:.ohdear LDX #0:BIT flag
1260 BMI noque:LDX #check-warn
1270 .noque JSR mprt:BIT flag
1280 BMI rnum2:LDA #15:LDX #1
1290 JSR osbyte:JSR osrdch:BCC noesc
1300 JMP cout
1310 .noesc AND #&DF:CMP #ASC"Y"
1320 BEQ yes:LDA #ASC"N"
1330 .yes JSR oswrch:PHA:JSR osnewl:PLA
1340 CMP #ASC"Y":BEQ rnum2:JMP cout
1350 .rnum2 LDA start:STA line
1360 LDA start+1:STA line+1:DEC doflag
1370 JSR renum:JMP cout
1380 .mprt LDA warn,X:BEQ mprt2
1390 JSR oswrch:INX:BNE mprt:.mprt2 RTS
1400 .warn EQUS "Program lines are out
of sequence":EQUW &D0A:EQUB 0
1410 .check EQUS "This would result in
program lines being out of sequence. Con
tinue (Y or N)? ":EQUB 0
```

```
1420 .chkord JSR cprog:JSR getsf
1430 JSR chkend:LDA bstart:STA lptr
1440 LDA bstart+1:STA lptr+1
1450 .chkord2 LDY #0:LDA (lptr),Y
1460 STA temp+1:INY:LDA (lptr),Y
1470 STA temp:JSR nline:LDA (lptr),Y
1480 STA lin+1:INY:LDA (lptr),Y
1490 STA lin:CMP temp:BNE chkord3
1500 LDA lin+1:CMP temp+1:BEQ outord
1510 .chkord3 LDA lin:CMP temp
1520 LDA lin+1:SBC temp+1:BCS chkord4
1530 .outord LDA #&FF:STA sflag
1540 JSR decprt:JSR sprt:LDA #ASC"-"
1550 JSR oswrch:JSR sprt:LDA lin
1560 STA temp:LDA lin+1:STA temp+1
1570 INC sflag:JSR decprt:JSR osnewl
1580 .chkord4 LDA lptr+1:BMI chkordo
1590 CMP bend+1:BNE chkord2:LDA lptr
1600 CMP bend:BNE chkord2
1610 .chkordo JMP cout
1620 .renum JSR setptr
1630 LDA #progend MOD &100:STA min
1640 LDA #progend DIV &100:STA min+1
1650 .renum2 LDY #1:LDA (lptr),Y
1660 STA (min),Y:DEY:LDA (lptr),Y
1670 STA (min),Y:BMI renumd:LDA min
1680 CLC:ADC #2:STA min:BCC renum3
1690 INC min+1:.renum3 JSR nline
1700 JMP renum2:.renumd LDA bstart
1710 STA lptr:LDA bstart+1:STA lptr+1
1720 .renumd2 LDY #0:LDA (lptr),Y
1730 BMI renumd4:BIT doflag:BPL renumd3
1740 LDA line+1:STA (lptr),Y:INY
1750 LDA line:STA (lptr),Y
1760 .renumd3 JSR nline:LDA lptr
1770 CMP bend:BNE renumd35:LDA lptr+1
1780 CMP bend+1:BEQ renumd4
1790 .renumd35 LDA line:CLC:ADC inc
1800 STA line:BCC renumd2:INC line+1
1810 BCS renumd2:.renumd4 BIT doflag
1820 BPL rloopo:JSR setptr
1830 .rloop LDY #0:LDA (lptr),Y
1840 BPL rloop2:.rloopo RTS
1850 .rloop2 STA curl+1:INY
1860 LDA (lptr),Y:STA curl:INY
1870 .rloop3 INY:LDA (lptr),Y:CMP #13
1880 BEQ rloop4:CMP #&8D:BNE rloop3
1890 JSR change:JMP rloop3
1900 .rloop4 JSR nline:JMP rloop
1910 .change INY:LDA (lptr),Y:ASL A
1920 ASL A:TAX:AND #&C0:INY
1930 EOR (lptr),Y:STA lin:INY:TXA
1940 ASL A:ASL A:EOR (lptr),Y
1950 STA lin+1:STY ytemp:LDA lptr:PHA
1960 LDA lptr+1:PHA:JSR setptr
1970 LDA #progend MOD &100:STA min
1980 LDA #progend DIV &100:STA min+1
1990 .change2 LDY #0:LDA (min),Y
2000 BMI nof:CMP lin+1:BNE change3:INY
2010 LDA (min),Y:CMP lin:BEQ change5
2020 .change3 LDA min:CLC:ADC #2
2030 STA min:BCC change4:INC min+1
2040 .change4 JSR nline:JMP change2
2050 .change5 LDA (lptr),Y:STA lin:DEY
2060 LDA (lptr),Y:STA lin+1:PLA
2070 STA lptr+1:PLA:STA lptr:LDY ytemp
2080 LDA lin+1:AND #&3F:ORA #&40
2090 STA (lptr),Y:LDA lin:AND #&3F
2100 ORA #&40:DEY:STA (lptr),Y
2110 LDA lin+1:AND #&C0:LSR A:LSR A
2120 LSR A:LSR A:STA temp:LDA lin
2130 AND #&C0:LSR A:LSR A:ORA temp
2140 EOR #&54:DEY:STA (lptr),Y
2150 INY:INY:RTS
2160 .nof LDX #9:.nof2:LDA fmess,X
2170 JSR oswrch:DEX:BPL nof2:LDA curl
2180 STA temp:LDA curl+1:STA temp+1
2190 LDA #0:STA sflag:JSR decprt
2200 JSR osnewl:LDY ytemp:INY:INY:INY
2210 PLA:STA lptr+1:PLA:STA lptr:RTS
2220 .decprt CLC:PHP:LDY #4
2230 .dp LDX #&30:.dp2 SEC:LDA temp
2240 SBC tenlo,Y:PHA:LDA temp+1
2250 SBC tenhi,Y:BCC dp3:STA temp+1
2260 PLA:STA temp:INX:BCS dp2
2270 .dp3 PLA:TXA:CMP #&30:BEQ dp4
2280 PLP:SEC:BCS dp5:.dp4 PLP:BCS dp5
2290 PHP:BIT sflag:BPL dp6:JSR sprt
2300 JMP dp6:.dp5 PHP:JSR oswrch
2310 .dp6 DEY:BPL dp:PLP:RTS
2320 .setptr LDA #1:STA lptr:LDA page
2330 STA lptr+1:RTS
2340 .sprt LDA #32:JMP oswrch
2350 .chkend JSR sskp:BCC chkend2
2360 LDX #berr-err:JMP error
2370 .chkend2 RTS
2380 .getsf JSR nget:JSR looklow
2390 JSR nget:BNE getsf2:LDA #&FF
2400 STA lin:LSR A:STA lin+1
2410 .getsf2 JSR lookhigh:LDA bend+1
2420 CMP bstart+1:BCC getsfe:BNE getsf3
2430 LDA bend:CMP bstart:BCC getsfe
2440 BEQ getsfe:.getsf3 RTS
2450 .getsfe:LDX #borr-err:JMP error
2460 .cprog STY ytemp:LDA page
2470 STA temp+1:LDY #0:STY temp
2480 LDA (temp),Y:CMP #13:BNE np:INY
2490 LDA (temp),Y:BMI np:LDY ytemp:RTS
2500 .np PLA:PLA:JMP cout
2510 .fmess EQUS " ta deliaF"
2520 .tenhi EQUB 1 DIV 256:EQUB 10 DIV
256:EQUB 100 DIV 256:EQUB 1000 DIV 256:E
QUB 10000 DIV 256
2530 .tenlo EQUB 1 MOD 256:EQUB 10 MOD
256:EQUB 100 MOD 256:EQUB 1000 MOD 256:E
QUB 10000 MOD 256
2540 .progend:]NEXT
2550 OSCLI "SAVE RENUMobj "+STR$~code+"
"+STR$~O%
```

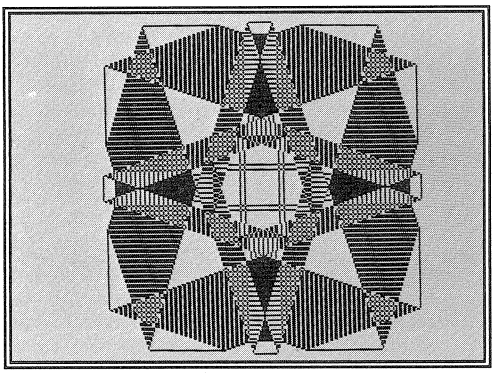
B

Symmetrical Patterns

We make no apologies for this highly mobile program by Matthew Draper, which is sure to fascinate young and old alike.

If you are fed up with typing in lengthy programs to generate good graphics, here's a real quickie to produce interesting patterns that move and change in endless detail.

Once you have entered the program and saved it away, typing RUN will produce two rectangles that move symmetrically about the centre of the screen, leaving trails and lines as they bounce around. The rectangles are displayed in suitable random colours that show up well on the cyan background. Pressing the space bar as the program runs will clear the screen and start a new pattern. Pressing 'P' will pause the screen, until 'C' is pressed to continue.



PROGRAM NOTES

The program works by literally *bouncing* a point around the co-ordinates of the screen, and reflecting that point vertically and horizontally. This creates four points which are joined up to make a rectangle. The X and Y co-ordinates of the points of the rectangle are reversed, creating a second rectangle. The lines are drawn using PLOT6, which inverts the pixels on the screen along the specified line. This creates the pattern effect as the rectangles pass over old lines, deleting some and producing others.

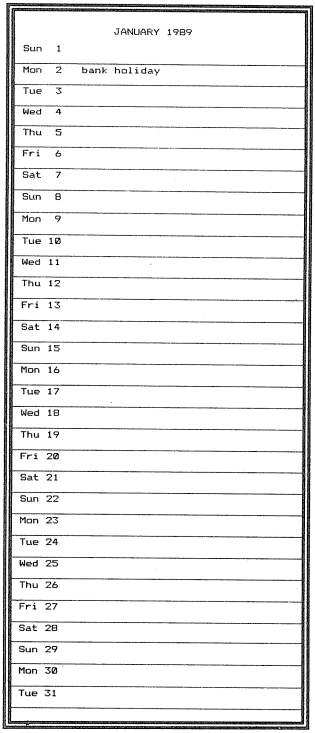
This program displays some of the fast and colourful graphics capabilities of the Beeb, in only a few lines of Basic. I find it is easy to gaze for ages at its soporific patterns as they shift and change endlessly.

```
  10 REM Program PATTERNS
  20 REM Version B2.7
  30 REM Author  M.J.Draper
  40 REM BEEBUG  December 1988
  50 REM Program subject to copyright
  60 :
 100 MODE4:VDU23,1,0;0;0;0;
 110 ON ERROR GOTO 390
 120 REPEAT
 130 IF RND(2)=1 col%=RND(2)-1 ELSE col
%=RND(2)+3
 140 VDU19,0,6,0,0,0
 150 VDU19,3,col%,0,0,0:CLS
 160 max%=800:step%=4:res%=4:xp%=200:yp
%=100
 170 x%=RND(400/res%)*res%
 180 y%=RND(400/res%)*res%
 190 xst%=RND(step%)*res%
 200 yst%=RND(step%)*res%
 210 REPEAT
 220 MOVEx%+xp%,y%+yp%
 230 PLOT6,x%+xp%,(max%-y%)+yp%
 240 PLOT6,(max%-x%)+xp%,(max%-y%)+yp%
 250 PLOT6,(max%-x%)+xp%,y%+yp%
 260 PLOT6,x%+xp%,y%+yp%
 270 MOVEy%+xp%,x%+yp%
 280 PLOT6,y%+xp%,(max%-x%)+yp%
 290 PLOT6,(max%-y%)+xp%,(max%-x%)+yp%
 300 PLOT6,(max%-y%)+xp%,x%+yp%
 310 PLOT6,y%+xp%,x%+yp%
 320 x%=x%+xst%:IF x%>max% OR x%<0 xst%
=-xst%
 330 y%=y%+yst%:IF y%>max% OR y%<0 yst%
=-yst%
 340 IF INKEY(-56)=-1 REPEAT UNTIL INKE
Y(-83)=-1
 350 UNTIL INKEY(-99)
 360 UNTIL FALSE
 370 END
 380 :
 390 MODE7:IF ERR<>17 THEN REPORT:PRINT
" at line ";ERL
 400 END                                B
```

A Monthly Desk Diary

Barry Thorpe shows how to combine computer and printer to provide a month-to-view diary, and you can use your favourite word processor to provide the text.

BEEBUG has published computerised diaries in the past, but that presented here is different, and much more practical. For a start it is not really a computerised diary at all. The purpose of this program is to print a complete month-to-view diary or calendar on a standard sheet of printout paper, with any entries taken from a separate text file created with your word processor.



```
                JANUARY 1989
Sun  1
Mon  2   bank holiday
Tue  3
Wed  4
Thu  5
Fri  6
Sat  7
Sun  8
Mon  9
Tue 10
Wed 11
Thu 12
Fri 13
Sat 14
Sun 15
Mon 16
Tue 17
Wed 18
Thu 19
Fri 20
Sat 21
Sun 22
Mon 23
Tue 24
Wed 25
Thu 26
Fri 27
Sat 28
Sun 29
Mon 30
Tue 31
```

Keep the printout handy and write in any new entries. When you choose, update the text file and reprint the desk calendar for any month. In addition, the days of the week and all bank holidays are included automatically. To print a monthly calendar you just need to enter the year, first month and number of months required.

Enter the program as listed and save away before using. You will also need an Epson compatible printer in order to use this program. To make full use of the calendar you will access to a word processor like View or Interword, or some other means (such as the Master's Edit) that allows you to create a simple ASCII text file of calendar entries.

When you run the program you are presented with a menu of three choices, Desk Calendar, Check data file, and Quit. The first two options are described below.

THE DESK CALENDAR

You will need to enter the year, followed by the first month and number of months for which printout is required. There are a number of routines to trap errors, including one to test for the presence of the printer, and one to check that the yearis within the range of the Gregorian calendar. Suitable error messages are displayed.

When the desk calendar is to be printed, the print head should be aligned very carefully at the very top of the page - in fact, just below the perforation - in order to fit the printout onto a single sheet. The printout is relatively slow, because emphasised and/or double-strike type is selected to ensure a good result, suitable for photo-copying as well if you wish. The printer commands can be changed if desired: they are located at lines 2120 and 2420.

The text file containing details of any events is not essential - you can print just a blank calendar ready to write in entries if you wish. However, to set up the engagement file on a word processor, simply enter edit mode and in date order type each engagement on a separate line using the format:

```
day/month/details of engagement
```

The details given should not exceed 70 characters total for each entry. For example:

```
5/10/Dog's birthday
```

The slashes are important, but leading zeros for day and month are not. The last character entered in the engagements file must be a '|' (Shift-Backslash). If you wish to change this terminating value, alter line 2310.

Of course, no formatting commands are required. Save the file under the name

W.DATES (ADFS users should create a directory 'W' first), and ensure that the disc with this file is present in drive 0 when the program is running.

When creating or editing a text file use whichever SAVE option produces a pure ASCII file. In general, aim for unjustified text, with no left margin and no embedded formatting codes, rulers or similar. With Interword then use the spool option; with View Wordwise just save as normal.

When the data file is used in printing a calendar, each engagement record is printed in condensed type immediately after the date (and bank holiday details if they coincide).

CHECK DATA FILE

The file-check option also uses the printer. It checks that the day or month number in the data file is feasible, and that the sequence of engagements is correct. Any line of text with errors is sent to the printer, with the number of the line at fault. NOTE: If your entries bridge a new year, the month number starts again at 1, which will generate an error. Provided that there is nothing else wrong, this can be ignored.

```
   10 REM Program Calendar
   20 REM Version B1.0
   30 REM Author  Barry Thorpe
   40 REM BEEBUG  December 1988
   50 REM Program subject to copyright
   60 :
  100 MODE7:PROCinit:ON ERROR GOTO210
  110 REPEAT
  120 RESTORE 190
  130 opt%=FNmenu("CALENDAR PRINTER",nit
ems%,maxlen%)
  140 IF opt%=1 THEN PROCdeskcal
  150 IF opt%=2 THEN PROCcheckfile
  160 UNTIL opt%=3
  170 MODE7:*FX4
  180 END
  190 DATA Desk calendar,Check date file
,Quit
  200 :
  210 CLOSE#0:VDU1,27,64,3:CLS:*FX15
  220 IF ERR=17 THEN GOTO110 ELSE REPORT
:PRINT" in line ";ERL:END
  230 :
```

```
 1000 DEF PROCinit
 1010 dy$="  ":mn$=dy$:rec$=STRING$(100,
" "):line$=rec$:rec$="":line$=""
 1020 DIM month$(12),daysinmonth%(12),da
y$(6),mstart%(12),daynum%(12)
 1030 pagelen%=66:width%=132:margin%=6
 1040 S$="    ":U$=S$+STRING$(70,"_"):do
t$=S$+STRING$(70,".")
 1050 DIM insert$(10),errmess$(5),dayabb
r$(31)
 1060 errmess$(1)="day out of range":err
mess$(2)="month out of range":errmess$(3
)="year out of range":errmess$(4)="print
er not connected":errmess$(5)="File not
found"
 1070 RESTORE 1130
 1080 FOR N%=1 TO 12
 1090 READ month$(N%),daysinmonth%(N%)
 1100 daynum%(N%)=daynum%(N%-1)+daysinmo
nth%(N%)
 1110 NEXT
 1120 FOR N%=0TO 6:READ day$(N%):NEXT
 1130 DATA JANUARY,31,FEBRUARY,28,MARCH,
31,APRIL,30,MAY,31,JUNE,30
 1140 DATA JULY,31,AUGUST,31,SEPTEMBER,3
0,OCTOBER,31,NOVEMBER,30,DECEMBER,31
 1150 DATA Saturday,Sunday,Monday,Tuesda
y,Wednesday,Thursday,Friday
 1160 filechecked=FALSE:nitems%=3:maxlen
%=20
 1170 ENDPROC
 1180 :
 1190 DEF FNzeller(D%,M%,Y%)
 1200 K%=(60+(100/M%))DIV100:X%=365
 1210 F%=X%*Y%+D%+31*(M%-1)-INT(.4*M%+2.
3)*(1-K%)
 1220 F%=F%+(Y%-K%)DIV4-INT(.75*(Y%-K%)D
IV100+1)+700
 1230 =F%MOD 7
 1240 :
 1250 DEF FNleap(y%)
 1260 IF y% MOD 400=0 THEN =TRUE
 1270 IF y% MOD 4=0 AND y% MOD 100<>0 TH
EN =TRUE
 1280 =FALSE
 1290 :
 1300 DEF FNyday(m%,d%)
 1310 IF m%=1 THEN =d%
 1320 LOCAL C%:tdays%=0
 1330 FOR C%=1 TO m%-1
 1340 tdays%=tdays%+daysinmonth%(C%)
 1350 NEXT
 1360 =tdays%+d%
 1370 :
 1380 DEF PROCtestbuffer
```

```
 1390 REPEAT:*FX21,3
 1400 probe1%=ADVAL(-4):VDU2,1,0,1,0,3:p
robe2%=ADVAL(-4)
 1410 printerconnected=probe1%=probe2%
 1420 IF NOT printerconnected THEN CLS:P
RINT TAB(8,10)CHR$129 CHR$136"Please con
nect printer"''TAB(8)"Press Return to go
 on":REPEAT UNTIL GET=13
 1430 UNTIL printerconnected
 1440 VDU2,1,27,1,64,3
 1450 ENDPROC
 1460 :
 1470 DEF PROCcentre(S$)
 1480 PRINTTAB((39-LENS$)DIV2)S$
 1490 ENDPROC
 1500 :
 1510 DEF FNmenu(T$,N%,L%)
 1520 LOCAL Y%,B$,S%:S%=3:VDU23,1,0;0;0;
0;:VDU26:CLS
 1530 FOR Y%=0 TO 1:PROCcentre(CHR$134+C
HR$157+CHR$132+CHR$141+T$+"   "+CHR$156)
:NEXT:PRINT
 1540 FOR Y%=1 TO N%:PRINTTAB(0,S%*Y%+1)
 1550 READ B$:PROCcentre(CHR$156+CHR$156
+CHR$134+LEFT$(B$+STRING$(L%," "),L%)+CH
R$156):NEXT
 1560 PRINTTAB(0,24)CHR$132CHR$157CHR$13
5"Use up/down cursors: then Return"SPC2C
HR$156;:*FX4 1
 1570 Y%=1:L%=(34-L%)/2:PRINTTAB(L%,S%*Y
%+2)CHR$129CHR$157
 1580 REPEAT:REPEAT UNTIL GET:PRINTTAB(L
%,S%*Y%+2)CHR$156CHR$156:Y%=(Y%-INKEY-42
+INKEY-58+N%-1)MODN%+1:PRINTTAB(L%,S%*Y%
+2)CHR$129CHR$157:UNTIL INKEY-74:*FX4
 1590 VDU23,1,1;0;0;0;:=Y%
 1600 :
 1610 DEF PROCerror(n%)
 1620 L%=LEN(errmess$(n%))
 1630 PRINT'TAB((39-L%)DIV2)CHR$129CHR$1
36;errmess$(n%)
 1640 TIME=0:REPEAT UNTIL TIME>200
 1650 PRINTTAB((39-L%)DIV2,VPOS-1)SPC(L%
+2)
 1660 mistake=TRUE
 1670 ENDPROC
 1680 :
 1690 DEF PROCrepeat
 1700 CLS:PRINTTAB(5,10)CHR$135CHR$157CH
R$129"Repeat this routine Y/N  "CHR$156;
 1710 REPEAT:A$=GET$
 1720 UNTIL INSTR("YNyn",A$)>0
 1730 norepeat=(INSTR("Nn",A$)>0)
 1740 ENDPROC
 1750 :
 1760 DEF PROCdeskcal
 1770 PROCtestbuffer
 1780 REPEAT
 1790 CLS:FORY%=0TO1:PROCcentre(CHR$134+
CHR$157+CHR$132+CHR$141+"DESK CALENDAR
 "+CHR$156):NEXT
 1800 VDU28,0,24,39,3
 1810 PRINT''CHR$134;"Enter year require
d: ";
 1820 REPEAT:mistake=FALSE
 1830 INPUTTAB(22,2)SPC6;TAB(22,2) year%
 1840 IF year%<1752 OR year%>4000 THEN P
ROCerror(3)
 1850 UNTIL NOT mistake
 1860 PRINT''CHR$131;"Enter number of fi
rst month: ";
 1870 REPEAT:mistake=FALSE
 1880 INPUTTAB(30,5)"  "TAB(30,5) startm
onth%
 1890 IF startmonth%<1 OR  startmonth%>1
2 THEN PROCerror(2)
 1900 UNTIL NOT mistake
 1910 PRINT''CHR$134;"Enter number of mo
nths: ";
 1920 REPEAT:mistake=FALSE
 1930 INPUTTAB(25,8)"   "TAB(25,8) nm%
 1940 IF nm%>12 THEN PROCerror(2)
 1950 UNTIL NOT mistake
 1960 month%=startmonth%:IF FNleap(year%
) THEN daysinmonth%(2)=29
 1970 day%=FNzeller(1,startmonth%,year%)
 1980 PROConepersheet
 1990 VDU26,12:PROCrepeat
 2000 daysinmonth%(2)=28
 2010 UNTIL norepeat
 2020 ENDPROC
 2030 :
 2040 DEF PROConepersheet
 2050 PROCcalc_hols(year%)
 2060 PRINT''CHR$131;"Is the text file W
.DATES available?";
 2070 REPEAT:A$=GET$:UNTIL INSTR("YyNn",
A$):inserts=INSTR("Yy",A$)>0
 2080 IF inserts AND NOT filechecked THE
N PRINT'CHR$129 "File not checked: OK? "
;:REPEAT:A$=GET$:UNTIL INSTR("YNyn",A$):
IF INSTR("Nn",A$) THEN ENDPROC
 2090 IF inserts THEN X=OPENUP"W.DATES":
PROCgetdaterec
 2100 IF month%=1 THEN I%=0 ELSE PROCfin
dstartofhols
 2110 PROCnextbankholiday
 2120 VDU2,1,27,1,69
 2130 FOR m%=1 TO nm%
 2140 T%=39-LEN(month$(month%))DIV2:PRIN
```

```
T SPC(T%);month$(month%)" ";year%'
 2150 FOR dy%=1 TO daysinmonth%(month%)
 2160 PRINT SPC(margin%);LEFT$(day$(day%
),3);SPC(2+(dy%>9));dy%;SPC(2);
 2170 IF bhd%=dy% AND bhm%=month% THEN P
RINT bh$;:PROCnextbankholiday
 2180 IF inserts THEN PROCinsertrec ELSE
 PRINT
 2190 PRINT U$
 2200 day%=day%+1:IF day%=7 THEN day%=0
 2210 NEXT dy%
 2220 L1%=pagelen%-(daysinmonth%(month%)
*2+2)
 2230 FOR K%=1 TO L1%:PRINT:NEXT
 2240 month%=month%+1:IF month%=13 THEN
month%=1:year%=year%+1:PROCcalc_hols(yea
r%):I%=0:PROCnextbankholiday
 2250 NEXT m%
 2260 CLOSE#X:VDU 1,27,1,64,3
 2270 daysinmonth%(2)=28
 2280 ENDPROC
 2290 :
 2300 DEF PROCgetdaterec
 2310 rec$="":A%=BGET#X:IF A%=ASC"|" THE
N CLOSE#X:inserts=FALSE:ENDPROC
 2320 REPEAT
 2330 rec$=rec$+CHR$(A%):A%=BGET#X
 2340 UNTIL A%=13
 2350 line$=rec$
 2360 slash%=INSTR(rec$,"/"):IF slash%=0
 THEN rec$="":ENDPROC
 2370 rday%=VAL(LEFT$(rec$,slash%-1)):ms
%=slash%+1:slash%=INSTR(rec$,"/",ms%):IF
 slash%=0 THEN rec$="":ENDPROC
 2380 rmonth%=VAL(MID$(rec$,ms%,slash%-m
s%)):rec$=MID$(rec$,slash%+1)
 2390 ENDPROC
 2400 :
 2410 DEF PROCinsertrec
 2420 IF dy%=rday% AND month%=rmonth% TH
EN VDU1,27,1,70,1,15:PRINT " ";LEFT$(rec
$,70):VDU1,18,1,27,1,69:PROCgetdaterec:E
NDPROC
 2430 PRINT
 2440 ENDPROC
 2450 :
 2460 DEF PROCfindeasterin(X)
 2470 A=X-19*INT(X/19):B=INT(X/100)
 2480 C=X-100*B:D=INT(B/4)
 2490 E=B-4*D:G=INT((8*B+13)/25)
 2500 F=19*A+B-D-G+15:Z1=INT(F/30)
 2510 H=F-30*Z1:M=INT((A+11*H)/319)
 2520 I=INT(C/4):K=C-4*I
 2530 Q=2*E+2*I-K-H+M+32
 2540 Z2=INT(Q/7):L=Q-7*Z2
 2550 R=H-M+L+90:M%=R/25
 2560 Z3=INT((H-M+L+M%+19)/32)
 2570 D%=H-M+L+M%+19-32*Z3
 2580 ENDPROC
 2590 :
 2600 DEF PROCcalc_hols(Y%)
 2610 daysinmonth%(2)=28-FNleap(Y%)
 2620 LOCAL m%,M%,d%,D%
 2630 d%=FNzeller(1,1,Y%)
 2640 IF d%>1 THEN dy$="1" ELSE dy$=STR$
(3-d%)
 2650 insert$(1)="0"+dy$+"01Bank Holiday
"
 2660 PROCfindeasterin(Y%)
 2670 insert$(3)=RIGHT$("0"+STR$(D%),2)+
"0"+STR$(M%)+"Easter Day"
 2680 m%=M%:d%=D%-2:IF d%<1 THEN d%=d%+3
1:m%=M%-1
 2690 insert$(2)=RIGHT$("0"+STR$(d%),2)+
"0"+STR$(m%)+"Good Friday - b.h."
 2700 m%=M%:d%=D%+1:IF d%>31 THEN d%=1:m
%=m%+1
 2710 insert$(4)=RIGHT$("0"+STR$(d%),2)+
"0"+STR$(m%)+"Bank Holiday"
 2720 d%=FNzeller(1,5,Y%):C%=1
 2730 IF d%<>2 THEN REPEAT:d%=d%+1:C%=C%
+1:d%=ABS(d%<7)*d%:UNTIL d%=2
 2740 insert$(5)="0"+STR$(C%)+"05Bank Ho
liday"
 2750 d%=FNzeller(31,5,Y%):C%=31
 2760 IF d%<>2 THEN REPEAT:d%=d%-1:C%=C%
-1:d%=ABS(d%<0)*7+d%:UNTIL d%=2
 2770 insert$(6)=STR$(C%)+"05Bank Holida
y"
 2780 d%=FNzeller(31,8,Y%):C%=31
 2790 IF d%<>2 THEN REPEAT:d%=d%-1:C%=C%
-1:d%=ABS(d%<0)*7+d%:UNTIL d%=2
 2800 insert$(7)=STR$(C%)+"08Bank Holida
y"
 2810 d%=FNzeller(25,12,Y%):C%=1
 2820 IF d%>1 THEN insert$(8)="2512Bank
Holiday" ELSE insert$(8)="2512Christmas"
 2830 d%=FNzeller(26,12,Y%)
 2840 IF d%>2 THEN insert$(9)="2612Bank
Holiday" ELSE insert$(9)=STR$(26+2-d%)+"
12Bank Holiday"
 2850 ENDPROC
 2860 :
 2870 DEF PROCnextbankholiday
 2880 I%=I%+1
 2890 bhd%=VAL(LEFT$(insert$(I%),2))
 2900 bhm%=VAL(MID$(insert$(I%),3,2))
 2910 bh$=" "+MID$(insert$(I%),5)
 2920 ENDPROC
 2930 :
```

```
 2940 DEF PROCfindstartofhols
 2950 I%=0:REPEAT:I%=I%+1:UNTIL VAL(MID$
(insert$(I%),3,2))>=month%
 2960 I%=I%-1
 2970 ENDPROC
 2980 :
 2990 DEF PROCcheckfile
 3000 CLS:inserts=TRUE:pyd%=0
 3010 errcount%=0:numentries%=0
 3020 X=OPENUP"W.DATES"
 3030 IF X=0 THEN PROCerror(5):ENDPROC
 3040 VDU2:PRINT"ERRORS IN W.DATES"'
 3050 PROCgetdaterec
 3060 REPEAT
 3070 numentries%=numentries%+1
 3080 IF rec$="" THEN errcount%=errcount
%+1:PRINT;numentries% SPC(3) line$ ELSE
PROCchecksequence
 3090 IF error THEN errcount%=errcount%+
1:PRINT;numentries% SPC(3) line$ " >> ou
```

```
t of sequence"
 3100 pyd%=yd%
 3110 PROCgetdaterec
 3120 UNTIL NOT inserts
 3130 PRINT';errcount%;" errors detected
"
 3140 VDU3
 3150 ENDPROC
 3160 :
 3170 DEF PROCchecksequence
 3180 error=FALSE
 3190 IF rmonth%<1 OR rmonth%>12 THEN er
ror=TRUE:ENDPROC
 3200 IF rday%<1 OR rday%>daysinmonth%(r
month%) THEN error=TRUE:ENDPROC
 3210 yd%=FNyday(rmonth%,rday%)
 3220 IF yd%<=pyd% THEN error=TRUE
 3230 REM unless, of course, the calenda
r crosses the new year
 3240 ENDPROC
```

I Ching: The Book of Changes (continued from page 9)

```
 9020 DATA KUAN: CONTEMPLATION,126
 9021 DATA SHIH HO: BITING THROUGH,128
 9022 DATA P'I: ADORNMENT,130
 9023 DATA PO: SHEDDING,132
 9024 DATA FU: RETURNING,134
 9025 DATA WU WUNG: SIMPLE INTEGRITY,136
 9026 DATA TA CH'U: ACCUMULATION THROUGH
 RESTRAINT,138
 9027 DATA I: NOURISHMENT,139
 9028 DATA TA KUO: EXCESS,141
 9029 DATA K'AN: THE PERILOUS CHASM,143
 9030 DATA LI: BRILLIANT BEAUTY,145
 9031 DATA HSIEN: MUTUAL ATTRACTION,148
 9032 DATA HENG: LONG DURATION,150
 9033 DATA TUN: WITHDRAWAL,152
 9034 DATA TA CHUANG: VIGOROUS STRENGTH,
153
 9035 DATA CHIN: PROGRESS,155
 9036 DATA MING I: HIDING OF THE LIGHT,1
57
 9037 DATA CHIA JEN: THE FAMILY,159
 9038 DATA K'UEI: OPPOSITES,161
 9039 DATA CHIEN: OBSTRUCTIONS,163
 9040 DATA HSIEH: ESCAPE,165
 9041 DATA SUN: DECREASE,167
 9042 DATA I: INCREASE,168
 9043 DATA KUAI: RENEWED ADVANCE,170
 9044 DATA KOU: SUDDEN ENCOUNTERS,172
```

```
 9045 DATA TS'UI: COLLECTING TOGETHER,17
4
 9046 DATA SHENG: ASCENDING,176
 9047 DATA K'UN: EXHAUSTING RESTRICTION,
177
 9048 DATA CHING: THE WELL,179
 9049 DATA KO: REVOLUTION,181
 9050 DATA TING: THE CAULDRON,183
 9051 DATA CHEN: THUNDER,185
 9052 DATA KEN: STILLNESS,187
 9053 DATA CHIEN: GRADUAL ADVANCE,189
 9054 DATA KUEI MEI: THE MARRYING MAIDEN
,191
 9055 DATA FENG: ABUNDANT PROSPERITY,193
 9056 DATA LU: THE TRAVELLING STRANGER,1
95
 9057 DATA SUN: GENTLE PENETRATION,197
 9058 DATA TUI: JOY,199
 9059 DATA HUAN: DISPERSION,200
 9060 DATA CHIEH: REGULATION,202
 9061 DATA CHUNG FU: INMOST SINCERITY,20
4
 9062 DATA HSIAO KUO: SMALL SUCCESSES,20
6
 9063 DATA CHI CHI: COMPLETION ACHIEVED,
208
 9064 DATA WEI CHI: BEFORE COMPLETION,21
0
```

Adventure Games

by Mitch

With the cold hand of winter knocking at the computer room door, perhaps now is the time to take a long sea voyage to the warm waters of the Caribbean.

If you decide to take my advice and book a passage under the black flag of Captain Blizzard, I can promise you calm seas, soft tropical nights and bloody murder!

Product	**Blood of the Mutineers**
Supplier	**Robico Software** **3 Fairland Close, Llantrisant,** **Mid Glamorgan CF7 8QH.** **Tel. (0443-227354)**
Price	**£12.95 inc VAT (cassette)** **£14.95 inc VAT (disc)** **£29.95 inc VAT (Archimedes disc with graphics)**

From the keyboard of the Welsh wizard comes a game which heralds the first of the Captain Blizzard adventures. The story-line follows the trials and tribulations of this pirate king as he attempts to the survive the worst that his scurvy crew can throw at him. There is of course, the obligatory treasure map which, if you care to follow it, will ensure you are given a guided tour of the most dangerous parts of the nearby island. Bears, scurvy and impassable vegetation threaten your every move as Robico seems to have shoe-horned in every possible hazard.

Your adventure as Captain Blizzard begins in the captain's cabin on board the pirate galleon, and the first inkling that all is not well is the sound of a dagger whistling past your ear lobe. In a matter of moments, the passageway to the cabin is awash with the howling of the mutinous crew baying for your blood. Bolting the cabin door will give you a few extra moments, but time is short and the window to the small ledge below the boat-deck seems the only likely avenue of escape. There are over 53 sea-faring objects such as cutlasses and spyglasses to be found skulking around in the bilges of this game, but as many of the objects are not immediately visible you must remember to SEARCH every one of the 117 locations. Robico has warned me that this is possibly the toughest game which they have produced - so faint hearts need not apply.

All the usual commands are available, including the option to give complex sentences. Should you decide your last command was not such a good idea after all, you may type "OG" - which 'takes back' your last move. Over 30K of text has been compressed into the game, which ensures more than adequate descriptions of your surroundings. The text has been written in the third person, which is to say that it is poor old Blizzard who suffers for your mistakes rather than you. This also means that you will receive replies such as "Blizzard bends carefully and picks up the knife", rather than a simple "OK".

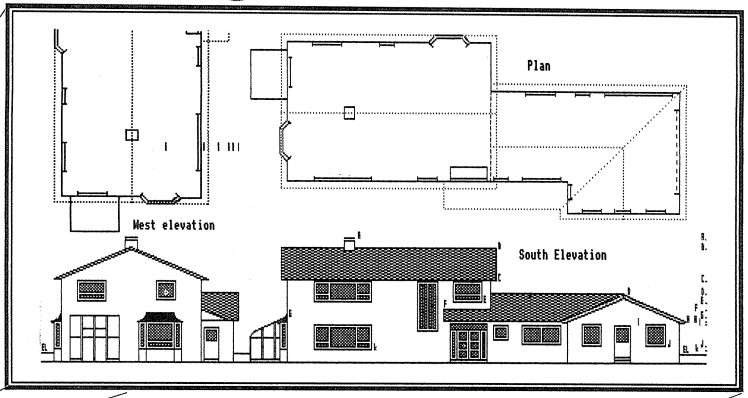
As with all Robico games the puzzles are generally speaking quite logical, and in many cases you have a rough idea of the likely solution quite early on. As usual it is the implementation of your idea which is the tricky bit. It doesn't take a genius to guess the likely use to which you could put a 6 foot cushion, a red coat and hat. However, conjuring up Guy Fawkes is one thing, finding the correct use for him while listening to the mob drawing near, is something else.

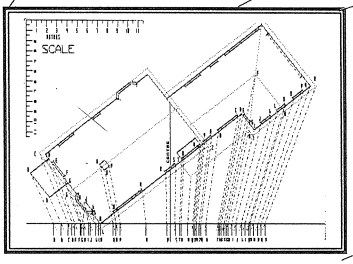
Every conceivable corner of this game has been jam-packed with devious puzzles, and some of them are quite sneaky. So far it has taken the combined efforts of three adventurers to help me escape from the ship and arrive at the island. The effort has been fun, and we all agree that there is no way that we will stop until we get Blizzard safely home to Blighty. If you have played a Robico adventure before, you will know that you are sure of a winner with this one.

I note that Robico has finally released *Enthar Seven* for the Archimedes - complete with graphic screens, while *Blazing Star* for the BBC Micro is also imminent. B

Graphic Design With ASTAAD 3

We present the final part of our powerful graphics program ASTAAD, by David Demaine.



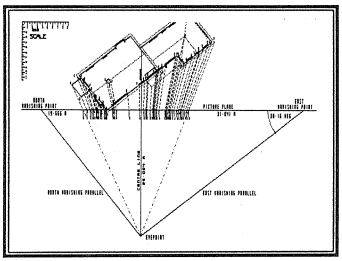


Rotate and Project

This month's listing will provide all of the remaining functions, namely the ASTAAD (Any Size Text Any Angle Display) function itself, and two more graphics functions, 'Mirror image' and 'Set ECF pattern'. Again, take care to follow line numbers when typing in the listing.

MIRROR IMAGE

Ctrl-f4 will allow you to choose between inverting or reversing the complete screen image. As this is a point-by-point function, and there are about 164,000 points to test, it is quite slow.

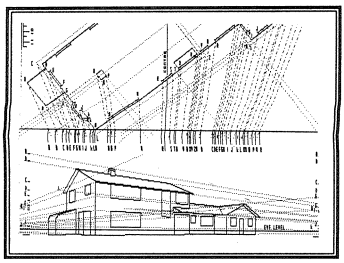


Shrink

VARIOUS FILL PATTERNS

To flood fill, move the cursor inside the area you wish to fill, and press Shift-Ctrl-Copy. The default fill is solid.

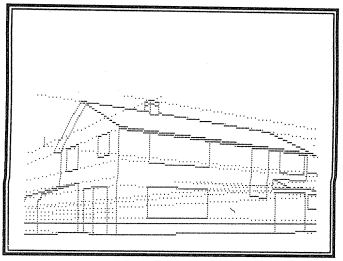
Ctrl-f5 is a multi-toggle which selects the pattern to be used for flood fill. The eight patterns provided are defined in the DATA statements from line 6080 to line 6150. These may be altered as required, but do not remove line 6160 or use the label 'S' for any new ones. This final pattern defines the solid fill. Unfortunately there is a limitation to the Extended Colour Fill (ECF) facility available in the Master's Operating System. If the pattern is comparatively open, it can potentially fill itself, and the graphics software quickly gives up, faced with too many leads to keep referenced. Naturally, it does not hang up, but just stops plotting, so that for some open shapes you will have to restart the fill a number of times to complete it. I have not found this a serious limitation. You had better make sure that you have an unbroken foreground line round the shape you wish to fill first. If you have any 'leaks' you are on your own!



Project to 3D

It is worth noting that whilst writing ASTAAD3 I have encountered three bugs in the Master graphics package. In

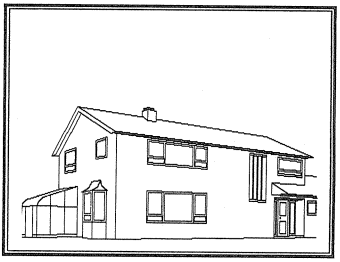
some circumstances, relative move commands give a one pixel plotting error vertically. The ellipse function has a flat top and bottom with knobs on. A many faceted polygon gives a better ellipse. I have also found one shape which causes the flood fill VDU command to hang up. If you are unlucky enough to encounter this, you will have to give up and change the shape of the area to be flooded. The moral is save the screen regularly as you develop it.



Enlarge

ASTAAD ITSELF

The method by which the ASTAAD function for displaying text works is unchanged from the original version of the program. The internal character definition is scanned with the use of an 8*8 bitwise AND algorithm (2^Q% AND A%?P%). If TRUE, this leads to the plotting of a character point on the screen. The shape of the point plotted was either a dot or a square. This has been changed to a dot, or an octagon which is distorted by the 'aspect ratio' of the character. The new function allows some flexibility in the definition of the character font. The header asks you to enter the text you require. Up to 160 characters may be entered. If you enter no text the program will present you with the last defined text. You are then given the option of redefining the font.



Line Perspective

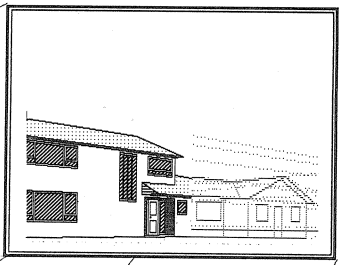
If you select 'Y' to do this, you are asked for the following:

Character width: Enter the width in standard graphics dimensions, that you wish the eight character pixels to occupy.

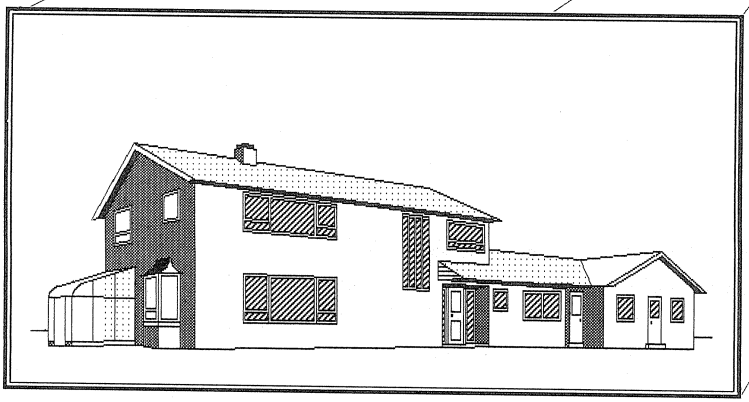
Character height: Enter the height of the eight character pixels.

Character spacing: The separation between the start of one character and the next.

'Pen' width: This defines the horizontal dimension across the flanks of the octagon. Less than 1 plots a point.



Mix Screens



Dot interpolation: 1 plots one octagon or dot for every point defined in the character. 2 plots an additional octagon or dot between every defined point. 3 plots two additional points, and so on. High numbers become slow.

The plotted octagons may be filled or open, depending on the condition of the Shift-f5 toggle. Whether you decide to redefine the font or not, you will be asked for the angle at which to print the text.

SOFT ASTAAD

An important feature of the old ASTAAD was its ability to use 'soft' characters instead of the standard Master character design. The standard characters are designed to look best when displayed on a 16*32 graphics screen layout, but they do look rather lumpy when displayed on a larger scale, and do not retain clarity well when they are reduced in size. The program listing defines a 'soft font' which is particularly useful for ASTAAD diagram or drawing notation. The upper case alphabet gives more open characters, which retain clarity better at smaller scale, or when tilted at an angle. The numerals and the lower case alphabet provide special characters which retain clarity when used at the smallest possible size, horizontally or vertically. They are designed to be used specifically with the following fonts:

	Minimum Width	Height	Space	'Pen'	Interpolation
Horizontal text:	16	32	8	0.5	1
Vertical text:	32	16	16	0.5	1

The first gives the remarkable horizontal character density of 160 characters per screen width at a spacing allowing 40 lines per screen. The second, used when the lines are written vertically, gives 60 characters per screen height with 106 lines per screen. Only a good monitor will permit the text to be read easily on screen, but it is clear on the screen dump, and is much better for ASTAAD drawing annotation than the standard characters, which are too big.

The coding which sets the soft font is contained in lines 6310 to 6980. It probably makes sense to incorporate this font-loading procedure into a preliminary program which calls ASTAAD. Then this will also leave more room for any further development of ASTAAD.

PROGRAM ORGANISATION

A major advantage of ASTAAD over commercially available packages for small micros, is the ease with which new functions can be added.

The format for a procedure to provide a new function is as follows:

First decide which function key you are going to use. You can use any function key, at the expense of the function already occupied by it, but perhaps the easiest way is to use the 'fourth row' of function keys operated by the combination Shift-Ctrl-f0 to f9, which are all unused (Shift-Ctrl-Copy is the only fourth row combination used by the published program, for flood fill with ECF). Shift-Ctrl-f0 gives the code J%=49, Shift-Ctrl-f1 gives J%=50, etc.

Next change line 2126 to alter the appropriate PROCnone to the function name you wish to use.

Your new procedure (activated by Shift-Ctrl-f0 for example) should now be arranged as follows:

```
DEF PROCname IF J%<>49 ENDPROC
```
(Strictly not necessary, but a good safeguard)
```
fn$="Functn"
```
(Six and only six characters which go in the header)
```
PROCmessage ("any text")
```
(The text, up to 40 characters, goes into the header)
```
param=FNinput("Any text")
```
(As above, but the function returns a numeric keyboard input)
```
param$=FNcheck("Any text")
```
(As PROCmessage, but the function returns a string keyboard input.)
```
. . . . . .
```
(The workings of your function.)
```
. . . . . .
PROCrehead
```
(Resets the header to the standard condition)
```
ENDPROC
```

Use PROCdelay for a 1.5 second pause to read messages if you wish.

If you want to use any of the header string varables to show additional information, as ASTAAD does when it displays font information, do not omit to save and then restore the standard header strings. Make sure that you obey the string length restrictions, mostly 4 characters, to avoid header formatting problems.

COMPLETION OF ASTAAD3

The new version of ASTAAD is now complete, but one of its delights is that it is so flexible and adaptable. The feature illustration this month shows a process of projecting a perspective view of a house, from a scale plan and elevation. The process used follows classical perspective projection methods.

```
10 REM Program  Astaad (Part 3)
   20 REM Version  B3.03
   30 REM Author   David Demaine
   40 REM Based On Original By Tim Tonge
   50 REM BEEBUG   December 1988
   60 REM Program  subject to copyright
   70 :
  105 PROCsoft
  265 IF key%=27 AND INKEY-1 AND INKEY-2
 PROCquit: UNTIL FALSE
 1680 @%=&00020206:PRINTTAB(0,1)coma$ SP
C1 "Scale:" scalep;
 1690 @%=&00020108:PRINT scl$ scale$ vec
tor$ vector*scale;
 2030 ON J% PROCtext,PROCareamove,PROCli
ne,PROCcircle,PROCpoly,PROCrepeat,PROCmo
ve,PROCdraw,PROCrubout,PROCdelete,PROCno
ne,PROCcopy ELSE ENDPROC
 2070 ON J%-16 PROCfirm,PROCcopymove,PRO
Clinearrow,PROCcircarc,PROCpolylipse,PRO
Cfill,PROCcolour,PROClinetog,PROCspeed,P
ROCorigin ELSE ENDPROC
 2110 ON J%-32 PROCdump,PROCvector,PROCs
olidot,PROCmargins,PROCmirror,PROCecf,PR
OCscale,PROClink,PROCsave,PROCload ELSE
ENDPROC
 2122 :
 2124 DEF PROCshiftctrlfnkey
 2126 ON J%-48 PROCnone,PROCnone,PROCnon
e,PROCnone,PROCnone,PROCnone,PROCnone,PR
OCnone,PROCnone,PROCnone,PROCnone,PROCfl
ood ELSE ENDPROC
 2128 ENDPROC
 4900 :
 4910 DEF PROCtext:IF J%<>1 ENDPROC
 4920 sfn$=fn$:scurs$=curs$:scopy$=copy$
:sdot$=dot$:smarg$=marg$:sfix$=fix$:sacc
el$=accel$:sorig$=orig$:orig$=b$
 4930 fn$="ASTAAD":curs$="FONT":copy$=FN
fthd(b$,asize):dot$=FNfthd(b$,aheight):m
arg$=FNfthd(b$,aspace):fix$=FNfthd(b$,ad
otr):accel$=FNfthd(" Int",ainterp%)
 4940 PROCheading:text$=FNread
 4950 IF text$="" VDU 4:PRINT TAB(0,1)at
ext$;:VDU 5:PROCdelay:text$=atext$ ELSE
atext$=text$
 4960 ans$=FNcheck("Re-define font (Y/N)
")
 4970 IF ans$="Y" OR ans$="y" PROCfont
 4980 fn$="ASTAAD":curs$=scurs$:copy$=sc
opy$:dot$=sdot$:marg$=smarg$:fix$=sfix$:
accel$=saccel$:orig$=sorig$
 4990 theta=FNinput("Angle (deg)?")
 5000 PROCrehead:VDU 4:PRINT TAB(0,1)tex
t$;:VDU 5
 5010 IF ABS(theta)>360.9 PROCrehead:END
PROC
 5020 size=asize/8:ratio=aheight/asize:s
pace=aspace*8/asize:dotr=adotr*0.5412:in
terp%=ainterp%:theta=RAD(theta):e=1+10*f
ill%
 5030 costheta=COS(theta):sintheta=SIN(t
heta):PROCocta
 5040 cos=size*costheta:sin=size*sinthet
a
 5050 chrw=7*size+dotr:chrh=(7*size+dotr
)*ratio
 5060 xx=x+chrw*costheta-chrh*sintheta:y
y=y+chrw*sintheta+chrh*costheta
 5070 FOR text%=1 TO LEN(text$)
 5080 xs=(text%-1)*space:X=FNx(0,0):Y=FN
y(0,0)
 5090 diag=SQR(asize*asize+aheight+aheig
ht)+dotr*ratio
 5100 IF X>1280+diag OR X<-diag OR Y>960
+diag OR Y<-diag GOTO 5240
 5110 PROCread
 5120 FOR P%=0 TO 7:FOR Q%=0 TO 7
 5130 IF (2^Q% AND A%?P%)=0 GOTO 5220
 5140 X=FNx(P%,Q%):Y=FNy(P%,Q%):PROCoct(
X,Y)
 5150 IF interp%=1 GOTO 5220
 5160 FOR R%=1 TO interp%-1
 5170 IF P%<>7 PROCinterp(P%+1,Q%)
 5180 IF Q%<>7 PROCinterp(P%,Q%+1)
 5190 IF P%<>7 OR Q%<>7 PROCinterp(P%+1,
Q%+1)
```

```
 5200 IF P%<>7 OR Q%<>0 PROCinterp(P%+1,
Q%-1)
 5210 NEXT R%
 5220 NEXT Q%
 5230 NEXT P%
 5240 NEXT text%
 5250 PROCrehead:ENDPROC
 5260 :
 5270 DEF PROCocta
 5280 FOR I%=0 TO 7
 5290 alpha=PI/8+I%*PI/4:cosalpha=COS(al
pha):asinalpha=ratio*SIN(alpha)
 5300 ad(I%)=dotr*(cosalpha*costheta-asi
nalpha*sintheta)
 5310 bd(I%)=dotr*(asinalpha*costheta+co
salpha*sintheta)
 5320 NEXT I%
 5330 FOR I%=0 TO 6:cd(I%)=ad(I%+1)-ad(I
%):dd(I%)=bd(I%+1)-bd(I%):NEXT I%
 5340 cd(7)=ad(0)-ad(7):dd(7)=bd(0)-bd(7
)
 5350 ENDPROC
 5360 :
 5370 DEF PROCoct(X,Y):IF adotr<1.0 PLOT
 69,X,Y:ENDPROC
 5380 FOR I%=0 TO 7:MOVE X,Y:PLOT 0,ad(I
%),bd(I%):PLOT e,cd(I%),dd(I%):NEXT I%
 5390 ENDPROC
 5400 :
 5410 DEF PROCfont
 5420 p1=FNinput("Character width? (8 to
 1280)")
 5430 IF p1<>0 asize=p1:copy$=FNfthd(b$,
p1):PROCheading
 5440 p1=FNinput("Character height (16 t
o 960)")
 5450 IF p1<>0 aheight=p1:dot$=FNfthd(b$
,p1):PROCheading
 5460 p1=FNinput("Character spacing? (8
to 1200)")
 5470 IF p1<>0 aspace=p1:marg$=FNfthd(b$
,p1):PROCheading
 5480 p1=FNinput("'Pen' width? (0.5 to c
hr width/2)")
 5490 IF p1<>0 adotr=p1:fix$=FNfthd(b$,p
1):PROCheading
 5500 p1%=FNinput("Dot interpolation (1
to 8)")
 5510 IF p1%<>0 ainterp%=p1%:accel$=FNft
hd(" Int",p1%):PROCheading
 5520 ENDPROC
 5530 :
 5540 DEF FNfthd(filler$,param):=RIGHT$(
filler$+STR$(param),4)
 5550 :
 5560 DEF FNx(P%,Q%):=xx+ratio*P%*sin-(Q
%-xs)*cos
 5570 :
 5580 DEF FNy(P%,Q%):=yy-ratio*P%*cos-(Q
%-xs)*sin
 5590 :
 5600 DEF PROCinterp(P%,Q%)
 5610 IF (2^Q% AND A%?P%)<>0 X1=FNx(P%,Q
%):Y1=FNy(P%,Q%):PROCoct(X+(X1-X)*R%/int
erp%,Y+(Y1-Y)*R%/interp%)
 5620 ENDPROC
 5630 :
 5640 DEF PROCread
 5650 X%=OS%:Y%=X% DIV 256:?OS%=ASC(MID$
(text$,text%,1))
 5660 A%=10:CALL &FFF1:A%=OS%+1:ENDPROC
 5670 :
 5680 DEF FNread
 5690 com$="Text?":PROCheading:VDU 4:PRI
NT TAB(LEN("Text?")+1,1);:text$=""
 5700 REPEAT:ascii%=GET
 5710 IF ascii%=21 PRINT STRING$(LEN(tex
t$),CHR$127);:text$="":ascii%=0
 5720 IF ascii%=127 AND LEN(text$) VDU a
scii%:ascii%=0:text$=LEFT$(text$,LEN(tex
t$)-1) ELSE IF ascii%=127 VDU 7:ascii%=0
 5730  IF ascii% AND ascii%<>13 text$=te
xt$+CHR$(ascii%+soft%):VDU ascii%+soft%
 5740 UNTIL ascii%=13:VDU 5:=text$
 5750 :
 5760 DEF PROCfirm:IF J%<>17 ENDPROC
 5770 IF soft% soft%=0:soft$="Text" ELSE
 soft%=128:soft$="Soft"
 5780 PROCheading:ENDPROC
 5790 :
 5800 DEF PROCmirror:IF J%<>37 ENDPROC
 5810 fn$="Mirror":ans$=FNcheck("Reverse
 or Invert (R/I)?")
 5820 IF ans$="R" OR ans$="r" PROCmessag
e("Reversing screen."):PROCturnover(638,
2,956,4,TRUE)
 5830 IF ans$="I" OR ans$="i" PROCmessag
e("Inverting screen."):PROCturnover(476,
4,1278,2,FALSE)
 5840 PROCrehead:ENDPROC
 5850 :
 5860 DEF PROCturnover(halfway%,step1%,o
therway%,step2%,whichway%)
 5870 wholeway%=2*halfway%+step1%
 5880 FOR p1%=0 TO halfway% STEP step1%
 5890 p2%=wholeway%-p1%
 5900 FOR p3%=0 TO otherway% STEP step2%
 5910 IF whichway% PROCswap(p1%,p2%,p3%,
p3%)
 5920 IF NOT whichway% PROCswap(p3%,p3%,
p1%,p2%)
 5930 NEXT p3%:NEXT p1%:ENDPROC
 5940 :
 5950 DEF PROCswap(lx%,rx%,ly%,ry%)
```

```
 5960 l%=POINT(lx%,ly%):r%=POINT(rx%,ry%
)
 5970 IF l%=r% ENDPROC
 5980 IF r% PLOT69,lx%,ly% ELSE PLOT71,l
x%,ly%
 5990 IF l% PLOT69,rx%,ry% ELSE PLOT71,r
x%,ry%
 6000 ENDPROC
 6010 :
 6020 DEF PROCecf:IF J%<>38 ENDPROC
 6030 IF TIME>t% RESTORE 6080
 6040 READ ecf$,hexa%,hexb%,hexc%,hexd%
 6050 IF ecf$="S" RESTORE 6080:TIME=t%+1
 ELSE TIME=0
 6060 VDU 4,23,2,hexa%;hexb%;hexc%;hexd%
;5
 6070 PROCheading:ENDPROC
 6080 DATA"1",&F00F,&F00F,&F00F,&F00F
 6090 DATA"2",&FF80,&FF08,&FF80,&FF08
 6100 DATA"3",&33CC,&33CC,&33CC,&33CC
 6110 DATA"4",&FF00,&FF00,&FF00,&FF00
 6120 DATA"5",&830E,&38E0,&830E,&38E0
 6130 DATA"6",&E038,&0E83,&E038,&0E83
 6140 DATA"7",&CCCC,&CCCC,&CCCC,&CCCC
 6150 DATA"8",&8000,&0000,&8000,&0000
 6160 DATA"S",&FFFF,&FFFF,&FFFF,&FFFF
 6170 :
 6180 DEF PROClink:IF J%<>40 ENDPROC
 6190 REM PROCnoproc:ENDPROC
 6200 fn$="Link  ":PROCmessage("Chaining
 to STEAMS")
 6210 *FX 229,0
 6220 VDU 4:J%=&7FFFFF:CHAIN "ISSUE4":EN
DPROC
 6230 :
 6240 DEF PROCquit:PROCmessage("Quitting
"):quit%=TRUE:CRASH:END
 6250 :
 6260 DEF PROCflood:IF J%<>60 ENDPROC
 6270 PROCmessage("Flood fill with ECF "
+ecf$)
 6280 VDU4,18,16,1,5:PLOT133,x,y:VDU4,18
,0,1,5
 6290 PROCrehead:ENDPROC
 6300 :
 6310 DEF PROCsoft
 6320 VDU23,171,0;0;&E040;&0040;
 6330 VDU23,173,0;0;&E000;0;
 6340 VDU23,174,0;0;&4000;0;
 6350 VDU23,176,0;&E000;&A0A0;&E0A0;
 6360 VDU23,177,0;&4000;&4040;&4040;
 6370 VDU23,178,0;&E000;&E020;&E080;
 6380 VDU23,179,0;&E000;&E020;&E020;
 6390 VDU23,180,0;&A000;&E0A0;&2020;
 6400 VDU23,181,0;&E000;&E080;&E020;
 6410 VDU23,182,0;&E000;&E080;&E0A0;
 6420 VDU23,183,0;&E000;&2020;&2020;
 6430 VDU23,184,0;&E000;&E0A0;&E0A0;
 6440 VDU23,185,0;&E000;&E0A0;&E020;
 6450 VDU23,193,&2418;&8142;&81FF;&8181;
 6460 VDU23,194,&81FE;&FC82;&8182;&FE81;
 6470 VDU23,195,&413E;&8080;&8080;&3E41;
 6480 VDU23,196,&82FC;&8181;&8181;&FC82;
 6490 VDU23,197,&80FF;&F880;&8080;&FF80;
 6500 VDU23,198,&80FF;&8080;&80F8;&8080;
 6510 VDU23,199,&413E;&8080;&8780;&3F41;
 6520 VDU23,200,&8181;&FF81;&8181;&8181;
 6530 VDU23,201,&0808;&0808;&0808;&0808;
 6540 VDU23,202,&0101;&0101;&8101;&3C42;
 6550 VDU23,203,&8281;&F884;&8488;&8182;
 6560 VDU23,204,&8080;&8080;&8080;&FF80;
 6570 VDU23,205,&C381;&99A5;&8181;&8181;
 6580 VDU23,206,&C181;&91A1;&8589;&8183;
 6590 VDU23,207,&423C;&8181;&8181;&3C42;
 6600 VDU23,208,&81FE;&8181;&80FE;&8080;
 6610 VDU23,209,&423C;&8181;&8589;&3D42;
 6620 VDU23,210,&81FE;&8181;&84FE;&8182;
 6630 VDU23,211,&807F;&7080;&010E;&FE01;
 6640 VDU23,212,&087F;&0808;&0808;&0808;
 6650 VDU23,213,&8181;&8181;&8181;&7E81;
 6660 VDU23,214,&4141;&4141;&2241;&0814;
 6670 VDU23,215,&8181;&8181;&A599;&81C3;
 6680 VDU23,216,&4281;&1824;&2418;&8142;
 6690 VDU23,217,&4141;&1422;&0808;&0808;
 6700 VDU23,218,&02FF;&0804;&2010;&FF40;
 6710 VDU23,160|
 6720 VDU23,225,0;&E000;&E0A0;&A0A0;
 6730 VDU23,226,0;&C000;&E0A0;&E0A0;
 6740 VDU23,227,0;&E000;&8080;&E080;
 6750 VDU23,228,0;&C000;&A0A0;&C0A0;
 6760 VDU23,229,0;&E000;&E080;&E080;
 6770 VDU23,230,0;&E000;&E080;&8080;
 6780 VDU23,231,0;&E000;&A080;&E0A0;
 6790 VDU23,232,0;&A000;&E0A0;&A0A0;
 6800 VDU23,233,0;&4000;&4040;&4040;
 6810 VDU23,234,0;&2000;&2020;&E0A0;
 6820 VDU23,235,0;&8000;&A080;&A0C0;
 6830 VDU23,236,0;&8000;&8080;&E080;
 6840 VDU23,237,0;&E000;&E0E0;&A0A0;
 6850 VDU23,238,0;&E000;&A0A0;&A0A0;
 6860 VDU23,239,0;&E000;&A0A0;&E0A0;
 6870 VDU23,240,0;&E000;&E0A0;&8080;
 6880 VDU23,241,0;&E000;&A0A0;&20E0;
 6890 VDU23,242,0;&E000;&E0A0;&A0C0;
 6900 VDU23,243,0;&E000;&4080;&E020;
 6910 VDU23,244,0;&E000;&4040;&4040;
 6920 VDU23,245,0;&A000;&A0A0;&E0A0;
 6930 VDU23,246,0;&A000;&A0A0;&40A0;
 6940 VDU23,247,0;&A000;&E0A0;&E0E0;
 6950 VDU23,248,0;&A000;&40E0;&A0E0;
 6960 VDU23,249,0;&A000;&E0A0;&4040;
 6970 VDU23,250,0;&E000;&4020;&E080;
 6980 ENDPROC                        B
```

Hacker ROM Review

David Spencer looks at the latest version of this utility ROM.

Product:	**Hacker ROM**
Supplier:	**CPH Software, 208 Bolehill Road, Sheffield S6 5DE. Tel. (0742) 345991**
Price:	**£19.95 inc. VAT.**

CPH Software's *Hacker ROM* is a collection of twenty-three general purpose star commands for the model B. Such products were in great abundance three or four years ago, but I thought that all but the best had died a death. It therefore comes as a bit of a surprise to see this new release, even if it is an upgraded version of an existing product.

The name *Hacker ROM* may seem a little inappropriate, especially because the term Hacker is normally associated with the world of comms. But all becomes clear when you realise that a number of the commands provided are for the implementation and cracking of protection on cassette based programs. I thought this sort of thing had been laid to rest in years past as well, but more anon.

Before explaining the new commands in any detail, I must say that the *Hacker ROM* is only for use with the model B. If you install it in a Master or Compact, then not only will the *Hacker ROM's* commands not work, but neither will any of the machine's other star commands.

NEW COMMANDS

The twenty-three star commands offered by the *Hacker ROM* can be split into three main classes:

1. Commands controlling the *Hacker ROM* itself
2. General utility commands
3. Commands connected with the protection of cassette based programs

Included in the first set of commands is the Help information. The *Hacker ROM* is one of those annoying ROMs that announces its presence in response to all *HELP commands, regardless of any parameters given. If you type:

*HELP HACKER

then all the available commands are listed using a colourful mode 7 display. This means that if you are in any mode other than 7, the mode is changed before printing the Help text. Furthermore, the Help text gives no indication of the parameters required by commands, and is therefore not particularly helpful anyway. Incidentally, any trailing spaces accidentally inserted after the command name will prevent it from working.

Two commands, *MESSAG and *NOMESS, enable and disable the display of messages by the *Hacker ROM*. By default, messages are given wherever necessary to show the progress of certain commands. The final command in this category is *OFFHAK which disables the *Hacker ROM* until the next hard reset. An addendum slip supplied with the manual states that the *Hacker ROM* clashes with certain other commercial software, including the View wordprocessor. It is therefore quite important to be able to disable the *Hacker ROM* easily.

The majority of *Hacker ROM* commands fall into the category of general utility commands. These range from the trivial up to a complete memory editor. The commands in this group are:

*CLKEYS to clear all the function key definitions. *FX18 will do exactly the same.

*CLMEM clears all memory locations from &400 up to &7FFF. As the effect of this is irreversible, you are asked for confirmation before the memory is actually cleared.

*CURSOFF turns the flashing cursor off using VDU 23,1,0, and *CURSON turns it back on again.

*DOWNLO <rom> copies the contents of the given ROM into memory between &3000 and &6FFF, and then gives the option of saving

the downloaded image using the ROM title as a filename. The ROM number must be given as a single hex digit. To avoid the screen being overwritten, mode 7 is selected if necessary. *DOWNLO will not download an empty slot, or (to prevent piracy) the *Hacker ROM* itself.

*ENVREC which must be issued from within Basic, lists the definitions of the first four sound envelopes. Again mode 7 is used, and all values are treated as positive numbers, so for example, -1 will be listed as 255.

*KEYLIST lists the current function key definitions in mode 7. The amount of space left in the key definition buffer is also given.

*KEYLOAD <file> and *KEYSAVE <file> load and save the current function key definitions.

*MEMORY <address> invokes a full-screen mode 7 memory editor. This has a constantly updated display, rather like the original editor in Computer Concepts' *Disc Doctor*. The cursor keys are used to move around memory, and locations can be altered in either hex or ASCII form. The start address given with the command must be four hex digits, for example 007F and not just 7F.

*RELOC <start> <end> <new> moves the block of memory between locations <start> and <end> to the position starting at <new>.

*RETURN is a utility that will recover as much of a Basic program as is possible.

*ROMLIST gives a mode 7 display of all the ROMs installed. This includes the title, the ROM type, and whether or not the ROM has a second processor relocation address.

*STATUS lists various information about the *Hacker ROM* itself, and other items, such as the value of PAGE in Basic.

All of these commands can be useful, but some are so trivial that it is debatable whether a special command for the task is needed.

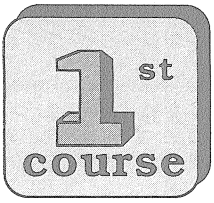
The five commands *BASLOCK, *LOCK, *LONORM, *SANORM and *UNLOCK are all concerned with cassette-based program protection. *BASLOCK transforms an already saved Basic program into a locked file that can only be *RUN. *LOCK causes all subsequent programs that are saved to be locked. *SANORM reverses the effect of *LOCK. *UNLOCK allows locked programs to be loaded normally, with *LONORM reversing the effect.

DOCUMENTATION

The *Hacker ROM* comes with a fifty page A5 manual, which is a combination of daisy-wheel and dot-matrix printed text. There are sections on the ideas behind ROM software, installing ROMs, and the *Hacker* ROM*'s error messages, as well as the* descriptions of each new command. Each command is given its own page, and this includes details of the command's usage, examples, errors that the command can produce, and any necessary notes. Screen dumps are used in places to show the results of commands, although the reproduction of these is rather poor. Two sections towards the back of the manual explain how to protect Basic programs from hackers, and how to relocate Basic programs to run on a disc-based machine. Appendices show the positioning of ROMs within the model B, the model B's memory map, and summaries of the *Hacker ROMs* commands and error messages.

CONCLUSION

In my opinion, releasing a software package at this time that doesn't work with a Master, and that concentrates on tape-based systems, is not the wisest decision. Ignoring this, I still feel that the standard of the *Hacker ROM* is far from the minimum acceptable standard. In particular, the need to specify numeric parameters as an exact number of digits is something that even simple ROM images published in BEEBUG and other magazines progressed beyond a long time ago. A budget price of £7 or £8 would serve to partly compensate for this, but for £19.95 I personally expect a professional product. CPH Software claim to be working on a version 3 of the *Hacker ROM* which will be over twice the length of the current implementation. I hope that this offers a substantial improvement. In conclusion, I have to say that unless the *Hacker ROM* is vastly improved, it really isn't worth considering. B



The Ins and Outs of Basic (Part1)

Mike Williams considers the methods of input and output available to Basic programmers.

BBC Basic provides a number of ways in which the program-mer can control keyboard input and screen (or printer) output. In this and next month's First Course, I propose to take a comprehensive look at what Basic has to offer in this department, and explain how the context helps to determine the best choice for any given application.

PRINT AND INPUT

Most people, when learning to program in Basic, find that the use of PRINT and INPUT comes very early in that process. This is hardly surprising, as these two instructions provide the simplest way to obtain input from the keyboard, or to display (or print) information. In both cases, Basic more or less takes care of the formatting for you, and very little effort is required by the programmer.

To use INPUT, just follow this keyword by a list of the variable names to hold the data typed in when the program is run. Specifying a variable as an integer (with a '%' sign), as a real number or as a string (with a '$' sign) determines how the input is stored. Be careful to match what you enter to the variable type. Thus if your program contains:

```
INPUT data
```

and you type, in response to the question mark:

```
123.456
```

the '123.456' will be stored as a number. But writing your INPUT statement as:

```
INPUT data$
```

will store 123.456 as a string of characters. Similarly:

```
INPUT data%
```

will store 123.456 as 123 because integer variables can only store whole numbers.

If you specify more than one variable in an INPUT statement, then they should be separated with commas, thus:

```
INPUT value,frequency
```

Using PRINT can be a little more complex. The items following PRINT may be either variables or constants, and these items may be separated by commas or semi-colons. Thus, writing:

```
PRINT "Press any key to continue"
```

or:

```
msg$="Press any key to continue"
PRINT msg$
```

would both have the same result. Likewise:

```
PRINT 123
```

and:

```
answer=123
PRINT answer
```

would also be equivalent. Often programs need to display messages on the screen, and PRINT followed by the string in quotes is the obvious way to do this. Assigning the string to a variable first is only worthwhile if the same message is to be printed from more than one place in the program. On the other hand, there is seldom any point in printing a constant, and numbers are more frequently displayed as the result of some calculation.

Numbers, whether as variables or constants are normally separated by commas, thus:

```
PRINT x,y,total
```

controls the way in which information will be positioned on the screen. To understand this further we need to introduce the concept of a *field*. Each line of the display is divided into consecutive fields, with each field (by default) capable of displaying a maximum of ten digits. In addition, a number is always right-justified within its field. That is, it is displayed with the right-most digit in the right-most position of the field. As a result, columns of integer numbers have all their digits correctly lined up.

When the first number has thus been displayed, Basic moves to the start of the next whole field in order to display the second number and so on. If a number is too long to fit within a field it is allowed to extend into the next field, but the second number then starts in the *third* field. For example (and a trivial example at that), type each of the following lines in turn:

```
PRINT 123,456,789
PRINT 112233445566,998877665544
```

The display should look like this:

field1	field2	field3	field4
123	456	789	
1.12233446E11		9.98877666E11	

In the first line of output the numbers are correctly aligned to the right of three 10 digit fields. In the second line, the numbers specified are too large for Basic to handle as integers, so they are automatically converted into real number format (1.12233446E11 means 1.12233446 multiplied by 10 to the power of 11). Approximations have also been made to the original numbers. Even so, the two values contain more than ten digits, so the first number fills the whole of field one and overflows into field two, while the second number starts in field three but then overflows into field four.

This detail may seem mundane, even boring, but without fully understanding what Basic is doing, displaying information on the screen can be a haphazard business.

Strings of characters (like names for example) are treated differently to numbers. By default, they are displayed left justified within each field, so that:

```
PRINT "ABC","DEF","GHI"
```

would produce the output:

ABC	DEF	GHI	

Again, if any string is too long for one field it is allowed to overflow into the next field (and beyond), but the next string output in the same PRINT statement starts at the next whole field. Thus:

```
PRINT "AABBCCDDEEFF",IIHHGGFFEEDD"
```

would produce:

AABBCCDDEEFF		IIHHGGFFEEDD	

The alternative to the use of the comma in PRINT statements is the semi-colon. Effectively, this causes any normal formatting to be ignored for the following item. Thus:

```
PRINT 123;456;789
```

will produce output:

```
123456789
```

The first number (123) is formatted normally, but the second and third numbers following semi-colons follow on immediately. If you want to treat the first number in the same way, precede it with a semi-colon, viz:

```
PRINT ;123;456;789
```

The other use for a semi-colon results from placing it at the end of a PRINT statement. Normally, a PRINT instruction causes the last item printed to be followed by a Carriage Return, Line Feed combination (CR/LF). This is inhibited by the semi-colon, and any further printing (from another PRINT) continues on the same line. This is often useful in a loop, thus:

```
100 FOR X%=1 TO 100
110 PRINT data(X%);
120 NEXT X%
```

Each value from the array data(), assuming that this has been dimensioned and values assigned to it, will be printed in successive fields. The number is formatted by being printed right justified in the next whole field; the semi-colon inhibits the CR/LF otherwise produced by PRINT. Fortunately for the display, when printout reaches the right-hand edge of the screen, it simply *wraps round* to continue from the beginning of the next line down.

The semi-colon as a separator is often used when printing strings, as we are not usually seeking column formats, rather continuous prose, but be careful to include any spaces that may be required:

```
PRINT "Total: ";sum
```

The semi-colon as a separator can be replaced with a space or simply omitted altogether with the same results. Including the semi-colon will often make things easier to read.

USING TAB

An alternative means of controlling the placing of output on the screen is provided by the use of the TAB function. This has two forms, the simplest TAB(X), where the value of X determines where on the current line the following item will b printed. If that position on the line has already been passed then the value of the TAB is ignored. Thus:

```
PRINT TAB(10)"This is a title"
```

would cause the text to start in position 10. TAB is particularly useful in exactly this kind of context.

The other version of TAB takes two arguments which specify both a position on a row and which row on the screen. The format is TAB(X,Y) where X is the position along the row, and Y is the position vertically. For example:

```
1000 DEF PROCdouble(x,y,msg$)
1010 LOCAL Y%
1020 FOR Y%=0 TO 1
1030 PRINTTAB(x,y+Y%)CHR$141msg$
1040 NEXT Y%
1050 ENDPROC
```

is a useful little procedure for mode 7 screens which displays a text message in double height characters starting in position x,y. An example of the use of this procedure would be:

```
PROCdouble(10,2,"MAIN MENU")
```

Such a procedure could easily be extended to included foreground and background colours as additional parameters.

Row and column numbers both start at zero. With a mode 7 screen for example, the values of X and Y would range from 0 to 39 and 0 to 24 respectively. Other limits apply for other modes.

Using TAB in this form has some advantages. Regardless of the existing position of the cursor, it will be moved to the position specified before printing what follows. It is also most effective when you want to change information on an existing display without scrolling the screen. For this reason you will find that many of the more sophisticated programs we publish adopt this method (look at the Desk Calendar program in this issue).

PROMPTED INPUT

There are two particular points which I want to deal with here. Used normally, the INPUT statement automatically generates a '?' to prompt for input. This is not always desirable, and can be omitted as we shall see. Secondly, many programs need to display some form of prompting message before input is typed in. The prompt can, in fact, be included as part of the INPUT statement, though with some limitations, rather than using a PRINT statement followed by INPUT.

Any INPUT statement can be written in the form:

```
INPUT <text><variable>
```

or:

```
INPUT <text>,<variable>
```

In both cases the text specified will be displayed on the screen, and Basic will then wait for appropriate input to be entered. In the latter case (with a comma *or* a semi-colon separating the two elements), a question mark will automatically be displayed. In the former case where there is no separator (though a space can be included for readability), the question mark no longer appears.

For example:

```
INPUT "Item1: "data1
```

would display the text string "Item1: " as a prompt, the space included as the last character in the string separating the text on the screen from any data then typed in. A variant of this which proves useful on occasion takes the form:

```
INPUT "" data
```

No question mark prompt would appear because of the string "" preceding the variable *data*, but because this is a null (or empty) string no prompt would appear.

There are limits as to what can be included as part of the text incorporated in an INPUT statement. The text must be an explicit string, not a string variable, and may not include any string function, although these are all quite permissible in a PRINT statement. For example, suppose we want to prompt for eight data items. If we write a FOR-NEXT loop controlled by the variable I%, then it might be thought that the prompt "Item1: " used previously could be replaced by "Item"+STR$(I%)+": " so that each item is numbered separately, but this is not possible because of the string function. In that case we must still use a combination of PRINT and INPUT:

```
100 FOR I%=1 TO 8
110 PRINT"Item"+STR$(I%)+": ";
120 INPUT "" data(I%)
130 NEXT I%
```

but note the use of the null string to avoid the otherwise automatic question mark. Note too,

Continued on page 64

UK BULLETIN BOARDS

Continued from Vol.7 No.5..

Communitel 24 Hours	01 968 7402 (London) 1200/75 Viewdata
Communitree 21.00-08.00	0874 711147 (Wales) 300/300
Compost Heap 24 Hours	0622 46036 (Maidstone) 300/300 1200/75
Cyberzone 24 Hours	01 638 2034 (London) 300/300
Cymrutel 24 Hours	0492 49194 (Colwyn Bay) 1200/75 Viewdata
Dark Crystal Fido 24 Hours	01 207 2989 (London) 1200/75 & 300/300
DCT Database 24 Hours	0384 239944 (Dudley) Prestel compatible
Distel 24 Hours	01 679 1888 (London) 300/300
Distel 24 Hours	01 679 6183 (London) 300/300 & 1200/75
EBBS No.1 24 Hours	0274 541156 (Bradford) 1200/75 300/300
E.D.D.I.E.S. 24 Hours	0635 71324 300/300
Electronic Tourist Board 24 Hours	0874 730172 (Powys) 300/300 1200/75
Enterprise HQ 24 Hours	0482 872294 (Hull) 1200/75 & 300/300
FBBS Jersey 18.00-06.00	0534 39389 (Jersey) 300/300
FBBS Swansea 24 Hours	0792 203953 (Swansea) 300/300
Forum 80 Hull 19.30-23.30	0482 859169 (Hull) 300/300
Gnome at Home 24 Hours	01 888 8894 (London) 1200/75 Viewdata
Goobtel 24 Hours	0602 706307 300/300 * 1200/1200
Gosport Apricot 18.00-00.00	0705 524805 (Gosport) 300/300
Hackney BBS 24 Hours	01 985 3322 (London) 1200/75 Viewdata
Hamnet 24 Hours Sat-Sun 18.00-8.00 Mon-Fri	0482 465150 (Hull) 300/300
Haunting Thunder 21.00-08.00	0752 364059 (Plymouth) 300/300
Health Data 24 Hours	01 986 4360 (London) 1200/75 Viewdata
HBBS Aberdeen 24 Hours	0224 632570 (Aberdeen) 1200/75 300/300

To be continued on page 36

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX. The normal copy date for receipt of all ads will be the 15th of each month.

Mega 128 ROM £55. Disc Doc £17. Paintmaster £17. Romit £17. Exmon II £17. Spellcheck III £20. Wordease £15. Mult-Forth £35. Toolkit £15. T-D ROM £15. 32K Shadow RAM board £50. ATPL Board with LP RAM fitted £30. Hyperdriver £18. Master Emulation ROM £13.Goulds switched mode power supply £60. 2764 ROMS £2.50. Manuals for all items. Tel. (0254) 76127.

Sanyo 14" colour data display monitor for use with BBC computer £125 ono. Spellmaster 128K ROM based spelling checker from Computer Concepts £38.
Tel. (0234) 67067.

BBC/Acorn Archimedes 310. As new in original packaging, can be used as an IBM clone, only £945. Tel. (0234) 67067.

ATPL ROM Board for BBC B with battery backed 16k CMOS RAM £32. GXR ROM with instruction manual & software, Toolkit ROM. Offers invited. Tel. (0384) 373928.

Colour Monitor original Acorn Archimedes RGB Monitor as new, sale due to up-grade to multi-sync. Accept offers around £120. Tel. (0532) 842780.

Acorn Electron, Plus 1, Plus 3, ROM box E2P, T2P3, View ViewSheet, Acorn Database on disc, MCP40 4 colour pen printer inc pens and paper, printer lead, hand books etc. £390 or reasonable offer. Tel. (0536) 722818 eves.

Archimedes 310 colour system, reference manuals, Archimedes assembly language book and 4 games discs. £850 ono. Tel. (0707) 371323.

BEEBUG Magazines vols 1 to 5 in binders and complete vol 6 £12. Tel. (07356) 3455.

BBC Master with 512 upgrade. Technomatic2 x 5.25" disc drives. Microvitec 653 monitor. Large amount disc and ROM software including all the inter range. Complete documentation (might split) offers Tel. 01-658 5602.

Wanted: Penpal with BBC Master, disc drive and possibly a Pagemaker and Super Art user. Write to J. Elburn, East Rudham, Kings Lynn, Norfolk PE31 8QX.

Instant Mini Office for BBC B £20. Exile 5" disc and competion form £8. Time crossword 5" disc £7. Tel. (08043) 2580.

Watford Mouse £16. Plonker Box £1. 2x 50's Antistatic disc boxes (lockable) with approx 90 5.25" discs all formatted D/S D.F.S. 80T £70. Pagemaker £17. Watford ROM Manager £16. InterBase £42. InterWord 1.02 £32. WordAid £16. Hi-Wordwise disc 3.25" £2. Also 3 manuals £9 each. Reference manual part 1 and 2. Dabhand Guide - Master Operating System. Tel. (09277) 63689.

Aries B32 RAM expansion board £40. Aries B12 ROM Sideways expansion board including 2x 8k RAMS £20. WordWise Plus £20. Basic Editor £12. All boxed and complete with instructions. Tel. (0202) 892904.

Acorn User magazines Nos 1 to 20 (except no 3) in official binders. Also individual copies of Micro User 1984 to 1988. Offers (to cover postage - £3.30 Acorn Users, 52p individual magazines). Tel. (0273) 770628 eves.

Philips 7502 Hi-Res Monitor £60. Grundig/Newark 14" remote control colour portable. TV/Med Res Monitor £195. Viglen Master cartridge system + 2 extra cartridges £12. Tel. (0268) 693770.

Sega Master System with 2 joysticks and five games (worth £95) including Space Harrier! Still under guarantee perfect condition. £95.
Tel. (0245) 268990.

Aries B32 £50. View & ViewSheet £25 each. BEEBUG's Toolkit £10. Care Electronics ROM module system with 6 modules £8. Books: Advanced User Guide £8. Advanced Basic ROM User Guide £5. The BBC Microcomputer Disc Companion £3. Tel. 01-882 3552.

HYBRID Music 5000 with keyboard and discs of music £200. Watford Lemodem £40. Acorn Teletext with ATS £45. Master Turbo board £65. Other hardware and software for 'B' and Master. Open to offers. Tel. (045383) 2897.

Solidisk Sideways RAM: Utility disc, Cartridge Base, Cartridge, Manual £15. Tel. (0823) 289378.

AMX Stop Press, latest version complete with additional fonts as new with manual £25. Tel. (0865) 271300.

Fishertechnik 30554 computing kit for Acorn/model B £100 ono. MUROM £11, ROMIT £13, BEEBFONT £12, ViewSheet £25, Repton 3, Elixir, Konami Coin-Op Hits, Play it again Sam 2 £3.50 each tape all above complete with manuals. Tel. (0837) 840718.

Colour monitor RGB Comp Video and Sound ok for 80 column print £100. Smith Corona D100 Dot Matrix printer excellent condition £90.
Tel. (04243) 4500.

InterWord new unused, unregistered £30 or swap for ADI ROM + software. BEEBUG Master ROM £20 or swap for software or EXMON II (new version). Tel. 01-698 3772.

Wanted: Acorn +3 disc drive for Electron. Micropower games cartridges for the Electron. Tel. (0236) 723615.

Teletext adaptor with a utility disc £60 ono. Tel. (0642) 474042.

Digimouse £10. Ref Manual (part 1) £5. Dabhand Master Op. Sys (book) £5. Master 128 For High Flyers (book) £3. Mini Office II (Disc/Guide) £8. BEEBUG Paintmaster/ Paintbox II (disc) £5. both softwares original. Sold separately or £30 the lot. Tel. (0933) 57811

Dual D/D, double sided 40T powered from BBC (QUME from Microware) £125 including documentation, Utilities disc, cover. Tel. 01-444 0521.

Master series reference manuals part 1 and 2 £12 each. New advanced use guide £15 plus
p&p. All new unused and unsoiled. Tel. (0424) 813794.

BEEBUG magazines vols 2 etc, up to issue 5 of vol 7. one issue only missing in vol 5. £55 worth of magazines for £20. Tel. (02216) 2965.

BBC B software cassette based incl. Flight simulator offers around £5 each. Tel. (02216) 2965.

6502 Second processor with DNFS and BASIC II ROMS, boxed £99. InterWord complete £29.
Watford ROM/RAM Board £20. Tel. (03727) 43887 eves.

Wanted: Micropower games cartridges for Acorn Electron. Also wanted Acorn Plus 3 for Acorn Electron. Write to: 13 Gartcarron Hill, Balloch, Eastfield, Cumbernauld, Nr Glasgow G68 9HR.

Wanted: SpellMaster ROM with documentation for BBC B. Tel. (0926) 882408.

Master 128: Cumana 40/80T double disc drive, Philips amber monitor plus manuals and BEEBUG box of 10 double sided discs hardly used. Computer in original box £400. Tel. 01-693 0082.

Master 128. Microvitec colour monitor. 80T D/S disc drive. Manuals 1&2 BEEBUG magazines and tapes EXMON II, Help etc £500. Tel. (0727) 24513.

Shadow RAM Watfords 32K hardly used original packaging £25 ono. Tel. (0225) 310083.

BBC Master 512K Mouse and Gem software £300. Microvitec colour 14" monitor £100. Dual floppy D/D 5.25" switchable 40/80T £150. NEC hard D/D 28 Mega Bytes £300. Epson SX80 dot matrix printer with NLQ ROM £120. AMX Mouse £20. Joystick £10. Tel. (073) 522 2898 eves.

Nidd Valley Digimouse with Grafik £25. Tel. (0698) 458137 after 5pm.

Sanyo 12" Green Screen Monitor £30. Tel. 01-942 1766.

Wanted: Status as corresponding member of a UK computer club. Energetic retired engineer. Special interests: Computer graphics. Write to: IB Heide, 24 Bueager, DK2950 Vedbak, Denmark.

MasterTurbo, Overview, AMX Mouse + software,colourmonitor.Ideal professional use £695. Logo + discs £44. Master ROM £16. Dumpmaster £16. Toolkit £14. Super Art£19.Various design and graphics software. Complete sets AU/BEEBUG mags offers. Tel. (0926) 335952.

Sayno CRT 70 colour monitor 14" nearly new £135. Tel. 01-239 5104.

Digimouse,Grafik, Chauffeur £15. PMS B2P 6502 2nd proc. plus HiBasic ROM£40. Tubelink Advanced Basic plus HiBasic Disc £12. Tubelink EXROM £5. View 2.1 £15. All manuals included. BBC B User Guide £4. New Advanced User Guide £8. Watford ROM board + batt. backup £18. 4 x 6264LP15 8K RAM £3 each. Tel. 041 634 1938.

Wanted: Function Key Strips for Acorn Z80 second processor software. Tel. 021 453 1203 eves.

ViewStore ROM £25. DabhandGuideto ViewStore and ViewSheet £5. Advanced 1770 DFS (ACP) £17. Dumpmaster ROM £15. 3.25" ADFS discs-The Superior Collection Vol 2 £5; White Knight ChessMk2£5; BAKsoft Acorn/MSDOS MSDOS/Acornfile transfer utility £5; Sidewriter £5; Logo Extension disc £5. Viglencompact cartridge system and three cartridges £10. Compact printer leadparallel £5. Compct second external 5.25" drive adaptor £7. Tel. 01-986 4442.

AMX Design £18. Mini Office II £7. Tel. (0494) 726203.

Wanted: HiWord version of ProWord wordprocessor ROM for use with Acorn 2nd processor. Tel. (0983) 864857.

Wanted: Vieglen console for BBC B. Tel. (0332) 559334.

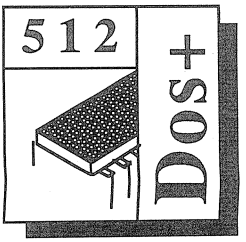
UK BULLETIN BOARDS

Continued from page 33

HBBS No.1 24 Hours	0274 42957 (Bradford) 1200/75 300/300
Hull BBS PC System 24 Hours	0482 20909 (Hull) 1200/75 300/300
IBBS Nottingham Sat 10am-4pm Sun 10am-4pm	0602 830231 (Nottingham) 1200/75 300/300 all ASCII
Infocom 24 Hours	021 476 9881 (Birmingham) 1200/75 300/300
Infotel ROS 24 Hours	01-581 3376 (London) 300/300 & 1200/1200
Intaview 24 Hours	021 622 5010 (Birmingham) 1200/75
ITCU Exchange & Mart 24 Hours	01 960 4742 (London) 1200/75 Viewdata
Keyboard 20.00-00.00	0908 668398 (Milton Keynes) 300/300
LABBS 24 Hours	01 373 6337 (London) 300/300
Livingstone BBS 24 Hours	0506 38526 300/300
Linelight 24 Hours	*0580 212043* 1200/75
London W Tech Centre 24 Hours	0895 52685 300/300 & 1200/75
London Underground 24 Hours	01 863 0198 (London) 1200/75 & 300/300
Madhouse 24 Hours	061 477 8405 (Manchester) 1200/75 300/300
Mailbox 80 24 Hours	051 428 8924 (Liverpool) 300/300
Mailbox 83 20.00-08.30 WE 24hr	0384 635336 (West Midlands) 300/300
Maptel 24 Hour	0702 552941 (Essex) 300/300
Marctel 24 Hours	01 346 7150 (London) 300/300
MBBS Leconfield 24 Hours	0401 50745 (Beverley) 300/300
MBBS Mitcham 24 Hours	01 648 0018 1200/75 & 300/300
Metrotel 24 Hours	01 941 4285 (London) 1200/75 Viewdata
MG-NET Sun 17.00-22.00	01 399 2136 300/300
MirrorWorld 24hrs	0483-844044 (Guildford) 1200
Mouse's Realm 7am-11pm Ring back	01-941 4333 (London) 300/300 & 1200/75

To be continued

Please notify us of any changes or omissions to the information published here.



512 Forum

Robin Burton answers letters on memory upgrades and documentation, and tells his own story about buying software.

Thanks for the letters, it's helpful to know what you want to know, let's have more. We may not be able to cover every point individually, but we'll start this month with some of the items raised.

G.A. Mangham from Worksop in Nottingham is thinking of acquiring a 512. The best advice I can give is to get on with it. According to the grapevine supplies of the 512 won't last much longer, if you're not already too late! People have quickly realised that at around £100 the chips alone cost twice as much as the asking price.

Now to some of the other points. Mr. Mangham also asks about memory expansion, which of course means the Solidisk PC+. This has not been advertised recently, but it is still available. Solidisk tell me that a new batch of boards should be available in January.

You have to send your 512 board to Solidisk for them to fit the additional 512K of RAM, and must also sign a disclaimer of the Acorn warranty to permit them to alter the hardware. You also need DOS+ 2.1 (or Solidisk will amend your 1.2 disc) to benefit from the extra RAM. The cost is £109.00 including VAT, plus £4.00 for return post and packing. There is no longer an annual 'maintenance charge' which, I suspect, put many potential customers off in the past. I hope to get my hands on a PC+ soon, and if so I'll include a detailed report in a future 512 Forum.

Mr. Mangham also enquires about comms software, and it's the usual problem. Most DOS comms packages directly access the serial port instead of using legal calls, hence they fail on the 512. However, a package called *Comm Plus* from Margolis Systems reportedly uses only legal calls and does work on the 512, supporting both scrolling ASCII and Viewdata. I have no price, no address and no other details: I will supply them when I find out. If anyone knows about this package, any information would be appreciated.

Another question concerns model B expansion boards. Are there any problems? For example, will the 512 board fitted to a Universal Co-processor work properly with all sideways RAM boards. Well, no problems are known by either Acorn or myself, and there should not be any with recent boards. The answer therefore is no, there should be no problems, unless anyone knows differently.

Finally Mr. Mangham mentions MS-DOS and asks if anyone has implemented it on the 512, or if there is a possibility of it, or of further DOS+ updates from Acorn. No is the simple answer to all of these, although there is nothing technically to stop somebody out there porting MS-DOS. Having said that, a little birdie has told me about some exciting new developments along these lines, although I can't say any more at the moment. Just remember where you heard it first!

DOCUMENTATION

Barry Tattersall writes from Victoria in Australia saying that there is not much support for the 512 over there. This is not confined to Australia, I am sure that most users would claim there is little support in this country either.

Barry's main point is the entirely inadequate, sometimes inaccurate (my words, he's more polite) so-called 'User Guide' supplied with the 512. He wonders where to obtain the DOS+ guides mentioned in the Acorn 512 manual. Not only is the manual inadequate, it's misleading in suggesting that these books are published by Digital Research.

The Programmers Reference Guide is published in the U.K. by Glentop Press Ltd. so asking about a book published by Digital Research will get you nowhere. The ISBN for this book is 1 85181147 8. Take note though that if you want a 'user guide' this isn't for you, this is a programmer's guide to the primitives in DOS+. Unless you write your own code it will not be

much use, and it certainly is not aimed at the average user.

As to the *DOS Plus User's Guide*, the difficulty with this is that it is not commercially published, and therefore does not have an ISBN. This means that book suppliers will not be able to help, as their sources will contain no reference to it.

The *DOS Plus User's Guide* is in fact a four-ring binder of about 700 loose A4 reference sheets. It's really a Digital Research manual and as they're not in the publishing business it is expensive, (around £40, I recall). That is if they are prepared to supply it at all. I feel that Digital Research would not be overjoyed to receive hundreds of requests for it, as each copy looks like it is put together by hand.

Do not lose all hope though. At least one publisher is actively supporting the 512 (and about time too). This is Dabs Press, and I must confess to not being totally impartial on this subject, because I have contributed a great deal to their books on the subject.

Due at the end of January is the *512 User Guide* at £9.95, or £14.95 with the programs disc. In short this is the manual that Acorn should have supplied with the 512. All the 512 DOS+ commands are documented, with examples and version differences. There is a section on disc formats and compatibility (including hard disc partitioning), a section on DOS's filing system, plus chapters on re-direction, command piping, the memory disc, batch files and so on.

The disc contains numerous utilities, including a disc editor for ALL the 512 formats, especially useful since none of the DOS editors can handle 640 or 800K discs. There's also a program to catalogue any DFS or ADFS disc from DOS. This too is particularly helpful. You don't need to exit DOS then re-boot the system when you mix discs up because (like most of us, let's be honest) you have discs lying around without up to date labels, or need to identify the contents to use MOVE, PUTFILE or GETFILE.

The second book planned by Dabs Press for the 512 is a more technical guide, which will be available around February or March. I will give you more details nearer the time.

CHEAPER SOFTWARE?

It's all very well to pay a £100 for the 512 board itself, but what happens when the first piece of software you buy costs more than the hardware did. Well, it seems that the price of some imported software suffers from the highly inflated exchange rate syndrome. I recite here the story of how, with a bit of work, I obtained a particular package at a great discount. I will not name the package involved, or the company producing it, because that would be making an example of someone when the problem exists right across the board.

The package in question is sold and supported in Britain in a 'U.K.' version, although not surprisingly it is originally American. Our version is the same except for spelling changes (eg colour for color etc.) in both the software and the manual, and of course the possibility of local backup and support, though I have never needed this.

Over the past year or two the price of this product has consistently been about £100, give or take £10, and support costs £30 per year. Nothing unusual so far.

I investigated half a dozen of the most popular alternatives before purchase, so I am satisfied this package is as good as any other, and in my opinion better, so I'm very happy with it. Like many well developed packages, its roots are in CP/M, and in consequence there's a maximum file size of one segment, or 64K. (See 512 Forum, Vol.7 No.4 for an explanation of segments).

Allowing for workspace this means a practical maximum file size of about 60K. The snag is that I use the package for admittedly an unusual purpose, but it's one which can easily produce files two or three times bigger than 60K.

Recently I learned that a new version was being released in the U.S. which removes this one segment file size limit. One evening I phoned the American supplier in Seattle to check the

facts. Lo and behold, the new version had been released five days before.

I next contacted the U.K. supplier to enquire about availability and upgrading. The answer was less than definite. It left me feeling that the new version might be released here sometime, if the development team were actually looking at it, (which could not be confirmed).

Next, I phoned Seattle again to see if I could acquire the software directly. Yes, I could, no there is no problem, an International Money Order, Visa, Mastercard, Diner's Club or American Express would all do nicely they said.

How much? (Prepare for enlightenment.) "If you want just the software the registration fee is $16 for the three discs." said the very pleasant lady on the phone, "If you want a soft-cover manual that's $35, or $45 for the hard-cover. If you also want support, the cost is $20 or $35 for support and the next two updates free. Shipping to the U.K. is $20."

What a difference! Rather than buying the software you are given it and pay to become a registered user. You can then decide how much support you actually need, and pay for this accordingly. In my opinion this is much better than the system over here where you are offered a complete package and the only choice you have is to take or leave it.

Compare prices too! I'm no international financier, but at a current rate of $1.85 to the pound the software is £8.64, a soft-cover manual just under £19 and support about £11 a year. (Where does the U.K.'s other £20 come from?)

Allowing for phone calls, exchange commissions and shipping I can obtain the latest version of this package for about £45, even if I was a first time buyer of that particular package.

Knowing the package well (it's full of 'Help' information anyway), I could get away with just the software and shipping, total about £20. In fact at this price I'll have the lot, even though I've already paid once before in this country.

Being realistic, retail prices cannot really bob up and down wildly like a cork on the sea of exchange rates, but surely my first loyalty is to myself. Provided that I acquire software legally and pay the full price there is no justification for parting with more cash than necessary.

I can now get 50% more dollars to the pound than a couple of years ago. Have you noticed the U.K. price of U.S. software reflecting this? I haven't!

If you're buying software that you know originates in the U.S. (that's most of it, because of the size of their market), and if it doesn't have important differences between international versions, do yourself a favour, invest in a phone call. You might get a pleasant surprise.

Remember time differences though, I phoned at 9.30pm. That's early afternoon in Washington State on Pacific Time, but would be closing time in New York, on Eastern Standard Time.

Finally, remember to keep those letters coming, because it is easier for me to answer specific queries than to try and guess what you want to know.

Details of the PC+ memory upgrade can be obtained from:

Solidisk Technology Ltd.
17 Sweyne Avenue,
Southend-on-Sea,
Essex SS2 6JQ.
Tel. (0702) 354674

A catalogue of Dabs Press products for the 512 can be obtained free of charge from:

Dabs Press
5 Victoria Lane,
Whitefield,
Manchester M25 6AL.
Tel. (061) 766 8423

B

Printer Utility ROMs

Surveyed by Geoff Bains

Although all printers are these days capable of a lot more than simply placing letters on paper, it can be difficult in the extreme to use them to the full. Many software producers have realised the difficulties experienced by printer owners when it comes to making use of the printing effects offered by their machines, and have provided assistance in the form of additional software.

The problem to be overcome is that printers expect commands in a very obtuse form - a series of obscure codes which really have very little in common with other computer applications.

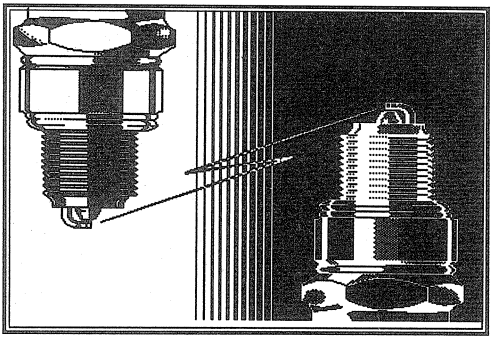
Utility software falls into two overlapping categories. First there is the type of program which looks after the difficult codes for you. The printer is controlled with a series of easy-to-remember star commands, which output the correct codes to the printer - *ITALIC, for example, would output 27,52. Such printer *driver* software often performs other utility functions such as the correct printing of pound signs.

It is important to make sure that potential driver software is suitable for your printer before purchase. Although most printers are Epson compatible (or IBM compatible, which in this case, for the most part, means the same thing), there can still be differences of codes. The Taxan KP-810 (a popular model amongst Beeb owners) is 'Epson compatible', but uses a different code to select NLQ printing from most other machines (including Epsons).

The other category of printer utility software is in the provision of graphics dumps. These are sometimes complete packages in themselves (such as Beebugsoft's Dumpmaster), and offer a wide variety of facilities. Other dump software is simpler, and is often just part of a wider driver package.

DRIVER SOFTWARE

There are four driver type software packages on the market, and these offer almost as many different approaches to the problem.



Screenprint dump

PRINTMASTER AND PRINTROM

Computer Concepts' **Printmaster** was probably the earliest printer utility ROM available for the BBC micro. PrintROM from Windmill Soft is one of the latest. Printmaster suffers a little from its speed off the mark. It has no star command to select NLQ printing, for example, because it predates NLQ printers.

However, it does have many extra features - enlarged characters (including user-defined ones) can be printed for banners, a printer spooler for printing one file while working on another and a graphics dump.

PrintROM is similar in operation, but it has no extra facilities except for an unnecessary menu selection system, a graphics dump and a *LPRINT command which emulates the Basic LPRINT found in other computers, circumventing the VDU2:PRINT:VDU3 structure normally needed.

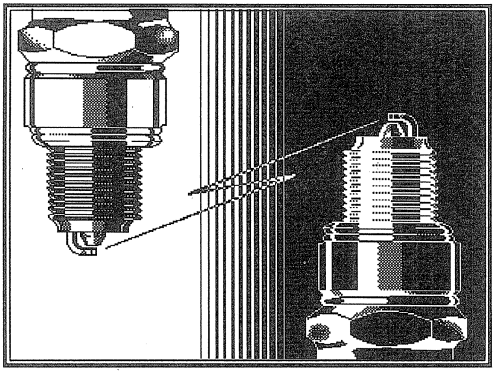
PrintROM and Printmaster are effective as far as they go, but these are really yesterday's software. Things have now moved on. They

operate efficiently enough but cost a lot for what is really just the avoidance of learning a few codes. Unless you consider buying these for their graphics dumps (see below), you would be better off with the more advanced competition.

WYSIWYG PLUS AND HYPERDRIVER

Technomatic's WYSIWYG is really designed as an extension to Wordwise Plus and View. WYSIWYG (What You See Is What You Get) not only outputs the required selection codes to the printer, but also alters the screen display to show the printed text in italics, enlarged, condensed, subscript or whatever.

Some of the on-screen effects are quite amazing. If the *CONDENSED command is issued, not only is the necessary code sequence output to the printer, but the entire character set is effectively altered to give 136 characters across the screen. This can be a little difficult to read, but there is no denying its use for visualising the printed results.



Dumpout3 print sample

As the program is really meant for use with word processors, the exact form of your printed document can be seen on the screen. However, it also serves very well in all other printer applications, and there is very little that your printer can do which is not controllable from WYSIWYG.

To round it off, WYSIWYG includes a logo generator and printer. A small piece of graphics for a letterhead or signature can be designed, stored on disc and printed whenever required.

Unlike Printmaster and PrintROM which are available in a limited number of different versions for different printer types, WYSIWYG uses a disc based printer driver which can be tailored to your printer. This is simply loaded in (if the in-built Epson driver is not sufficient) at the start of a session.

Hyperdriver from Dabs Press is a similar package. It too provides star commands to substitute control code sequences, although the command names are a little obscure - they are all two-letter mnemonics but follow a logical pattern once you've learnt it.

Commands can be strung together into *macros* to be called up with a single command. Macros help to get around the problem that Hyperdriver is not available in non-Epson form, nor does it use a disc-based driver.

Hyperdriver also provides on-screen effects like WYSIWYG. Rather than produce these all the time, they can be switched on, and printer output directed to the printer *sink*. A document can then be 'printed' to the screen with all the printing effects shown as though going to the printer.

Hyperdriver also offers two new fonts for your printer - an excellent NLQ in case your printer does not support this already - and a Beeb screen font, exactly emulating the characters on the screen, even down to the size they appear in different screen modes.

Both Hyperdriver and WYSIWYG offer an outstanding range of facilities to the Beeb printer user. They have all the simple code substitution needed, and other features (especially the on-screen effects) which give them the edge over the simpler ROMs.

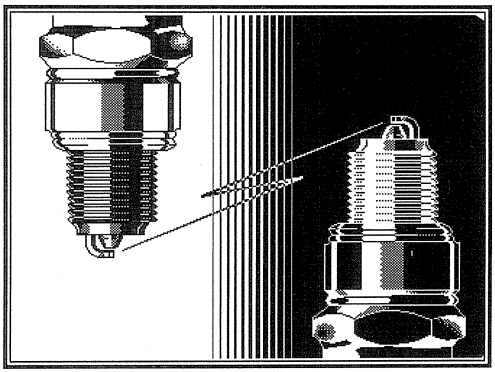
GRAPHICS DUMP SOFTWARE

A good dump program is a boon to any graphics-orientated Beeb user. Although many on-screen pictures look very pretty, they can

hardly be permanently displayed there. A hard copy printout is the best option, and a good dump routine is essential for that.

The quality of a printer dump is largely dependent on the printer itself - the print head, ribbon and so forth all contribute. However, there are still differences between the results from different dump software, and also differences between the facilities offered. For this review a laser printer with Epson emulation was used to ensure all the packages got a 'fair hearing'.

Rather than go through the packages one by one (see table for details), it is more useful to look at them together, with a discussion of the features they (should) offer.



Dumpmaster printout

It is most important to be able to dump any Beeb screen. All these packages will cope with any graphics mode (0,1,2,4 and 5), but how they deal with mode 7 and the text modes (3 and 6) is another matter. PrintROM just ignores them with a *Bad mode* error message. Printmaster copes with mode 7 but uses a separate dump routine for the text modes which just reprints the screen text on paper as with normal printing.

The other dumps **(Dumpmaster, Dumpout 3, Screenprint and Userdump)** all do proper graphics dumps of all modes. Userdump also offers an alternative dump for text modes along the same lines as Printmaster.

Mode 7 screens are first converted to mode 1 screens and then dumped. This is not completely successful for all screens (none of the programs attempt a real interpretation of the mode 7 characters, direct to paper), but it is a good compromise. All the programs which cope with mode 7 screens achieve similar results.

Screenprint and PrintROM have to dump the entire screen, whereas the other packages will dump just the area of the graphics window if required. These also have commands to adjust the window (and therefore the area dumped) with an interactive rubber-banded rectangle on the screen controlled from the keyboard.

Perhaps the most useful addition to a basic dump program is a *snapshot* facility. Once enabled, this allows a dump to be started by pressing a key combination while another piece of software is running, so you could, say, dump a scene from your favourite game without having to break into it.

Screenprint is really entirely based around this principle, and it is intended mainly for dumping pictures from commercial software in the classroom. This single minded approach explains its lack of other features.

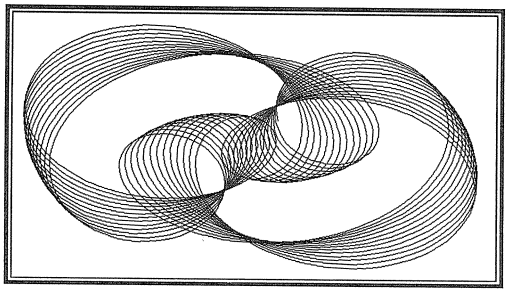
Dumpmaster, Dumpout 3 and Userdump also offer a snapshot facility, and in the case of the first two, the interactive window can also be used to select just part of the screen, once the game or whatever has been frozen, by triggering the snapshot.

None of the packages with a snapshot facility can cope with absolutely every program. There will always be some software which uses that bit of memory required by the snapshot, and so defeats it (it's usually just the screen you really wanted to dump which fails).

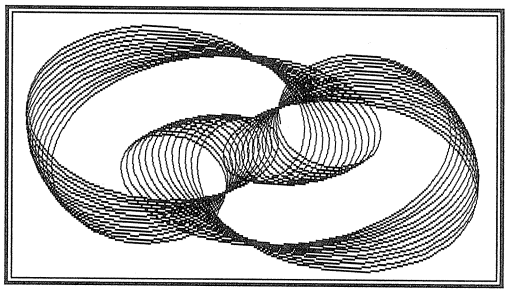
Of course, most screens on the BBC micro are in colour. If you want to get a colour printout you will have to use a colour printer. Dumpmaster and Userdump can accommodate these and both offer the Epson JX-80 standard of colour

codes. Dumpmaster also supports the Integrex ink jet colour printer.

However, most users have a printer capable of only black and white printing. Colour screen pictures printed on these are translated by the dump software into a range of grey scales representing the colours. Each grey level is printed by means of different density stippled patterns of black dots.



The Advantage of a Vector Screen Dump (above) over a Normal Screen Dump (below)



The effect is not all that convincing with any of the dump programs because all rely on the relatively low resolution graphics available on Epson FX-compatible printers. Most printers these days have higher resolutions available, so it would have been nice to have seen some software which made use of this. The higher the resolution, the finer the dots, and so the better the quality of the picture.

However, all the dump packages produced similar results. The big difference came in altering the colour patterns printed. It is sometimes useful to be able to print what is, say, red on the screen in the pattern normally used for, say, green, to achieve the best possible contrast and clarity for a particular picture.

Some of the packages allow the 'colours' printed to be different from those in the screen memory either by printing the physical rather than the logical colours (which can be altered to taste with the VDU19 command), or with a separate palette selection built into the dump command.

Userdump can print the physical colours but only in mode 2. In the other modes the logical colours are taken. Dumpout 3 allows either physical or logical colours to be printed. Dumpmaster instigates its own palette command to give you freedom of choice.

Control over the size and orientation of the dump is also useful. Dumpmaster, Dumpout 3 and Printmaster allow the dump to be vertical or horizontal on the paper and allow the margin on the paper to be adjusted to place the dump exactly where it is wanted.

Most important to the final result is the size of the printout. However, only Dumpout 3 allows any control of this. The vertical and horizontal dimension of the dump can be varied independently with this program to give not only whatever size is required but a choice of aspect ratio (the relative scale of the vertical and horizontal) as well.

However, with the 'wrong' choice of size, Dumpout 3 produces a rather peculiar dump because of the requirement to use a whole number of printer pixels for each screen pixel. Variable dump size gives greater control but also presents an easy pitfall.

Aspect ratio is important. Some other packages give a very poor aspect ratio which cannot be altered. Printmaster and Screenprint, in particular give very distorted printouts, and rather small too.

Choosing a printer dump package is not as simple as it first appears. To get the best results you really want as much control over all aspects of the process as possible. There is no doubt that Dumpmaster and Dumpout 3 provide the most variables. These two programs also use a sensible system of defaults which means that for a quick 'any-thing goes' dump you don't have to define every-thing.

A screen dump from Acorn User's gallery disc

Screenprint is simple to use, which is great for its classroom aim, but it also loses out on quality because of this. Dumpmaster and Dumpout 3 prove that good, versatile dump software needn't be difficult to use.

The dump routine in PrintROM is also too simple, although the ROM does have other features. The Printmaster dump offers a bit more versatility, and other printer utilities. Userdump gets close to the mark, but is still easily beaten to the post by Dumpmaster and Dumpout 3 - easily the best around.

VECTOR DUMPS

Quite different to these screen dump programs are two further items of software - Design Dynamics' **Mode-00 Dump** (£12.95) and Silicon Vision's Super-Dump (£15.95). Rather than produce printouts from the Beeb's screen data, these packages use the programs which produce the screens in the first place.

They convert the VDU commands which the operating system uses to draw lines and areas on the screen, directly into hard copy - rather like the commands used to drive a plotter - without using the screen display.

Extracting the VDU commands from a drawing program is simple enough. You simply add a *SPOOL <filename> before the drawing starts and a *SPOOL after the picture is drawn. The result-ing file contains all the VDU commands and it is used by the programs to drive the printer.

The advantage of this method is that much greater detail in the printed picture can be produced. The BBC micro uses a screen co-ordinate system of 1280x1024, whereas the highest resolution mode can only display 640x256. These two programs can get the Beeb's full resolution onto paper. Super-Dump will even produce a 1920x1024 resolution picture.

Both programs are similar. They are supplied on disc and use the spooled file directly. Both offer normal screen dumps to check what is going onto the paper and then a slower high resolution dump. **Super-Dump** is much slower than Mode-00 but it does produce a higher resolution picture (if you can wait anything up to an hour for it).

Mode-00 is only a little slower than a normal dump, but offer vastly superior output. Mode-00 also offers a program to produce graphs directly from input functions and axes ranges. Although this is impressive as a display, it is unlikely that there will be much use for it.

Both companies produce CAD software of one kind or another, and these programs are particularly suited to printouts of CAD drawings which are stored as line co-ordinates.

Either of these programs provides a useful top quality printout for really accurate work. They are no use at all for artful 'pictures ', but for graphs, CAD or even diagrammatic work they cannot be beaten.

PRINTER DRIVERS

	HYPERDRIVER	PRINTMASTER	PRINTROM	WYSIWYG PLUS
Price (inc. VAT)	£29.95	£33.35	£15.95	£24.15
Printer type versions		●	●	
Printer type disc driver				●
Colour printers supported	●			●
Pound sign printing	●			●
Screen effects	●			●
Graphics dump		●	●	
Logo printing				●
Extra fonts	●			
Macros	●			
LPRINT			●	
UDCs		●		
NLQ selection	●		●	●
Foreign character set	●	●	●	●
Proportional spacing	●	●	●	●
Inter-character spacing				●
Tabs	●			●
Daisywheel change pause				●
Send misc codes		●		●

All packages have command selection of a core of effects such as underline, emphasised, character size, page format, etc.

Suppliers:

Mode-00 Dump
Design Dynamics
8 Meadow Way,
Ampthill,
Bedford MK45 2QX.
Tel. (0525) 402447

Screenprint
ESM
Duke Street, Wisbech,
Cambridgeshire PE13 2AE.
Tel. (0945) 63441

Hyperdriver
Dabs Press
5 Victoria Lane,
Whitefield,
Manchester M25 6AL.
Tel. 061-766 8423

Printmaster
Computer Concepts
Gaddesden place,
Hemel Hempstead,
Herts HP2 6EX.
Tel. (0442) 63933

Super-Dump
Silicon Vision
Signal House,
Lyon Road, Harrow,
Middx HA1 2AG.
Tel. 01-422 2274

Dumpmaster
Beebugsoft
Dolphin Place,
Holywell Hill, St Albans,
Herts AL1 1EX.
Tel. (0727) 40303

PrintROM
Windmill Soft
Smock House, Hull Lane,
Terling, Chelmsford,
Essex CM3 2QD.
Tel. (024533) 371

SCREEN DUMPS

	DUMPMASTER	DUMPOUT 3	PRINTMASTER	PRINTROM	SCREENPRINT	USERDUMP
Price (incl. VAT)	£31.00*	£28.75	£33.35	£15.95	£28.75	£19.95
Printer type selection	●	●				
Colour printers supported	●					●
Graphics modes dumped	●	●	●	●	●	●
Text modes dumped	●	●			●	●
Mode 7 dumped	●	●	●		●	●
Snapshot	●	●			●	●
Dump window only	●	●	●			●
Interactive window	●	●	●			●
Colour inversion	●	●	●		●	●
Colour control	●	●				
Margins	●	●	●			
Horiz/Vert dump	●	●	●			
Variable size dump		●				

(* 25% Discount to members)

Userdump
Acorn User
20-26 Brunswick Place,
London N1 5DJ.
Tel. 01-490 1444

Dumpout 3
Watford Electronics
250 High Street,
Watford WD1 2AN.
Tel. (0923) 37774

WYSIWYG Plus
Technomatic
17 Burnley Road,
London NW10 1ED.
Tel. 01-205 9558

B

File Handling For All (Part 7)

By Mike Williams and David Spencer

Now that we have introduced all the functions provided by Basic for file handling we can begin to investigate more sophisticated file structures than the purely serial files dealt with so far.

There are a number of drawbacks to the use of simple serial files. There is no implied order to the records - the only order is the order in which the records are placed in the file. Of course, by writing a suitable sort program, we could re-arrange the records how we wished using one or more fields in each record as the key. Sorting files can be time-consuming, particularly if a file is too large to fit into memory, and frequent sorting of files is therefore best avoided. We shall be looking at this subject in due course, but if you want to find out more now, refer to the articles in BEEBUG Vol.5 Nos.7 & 8.

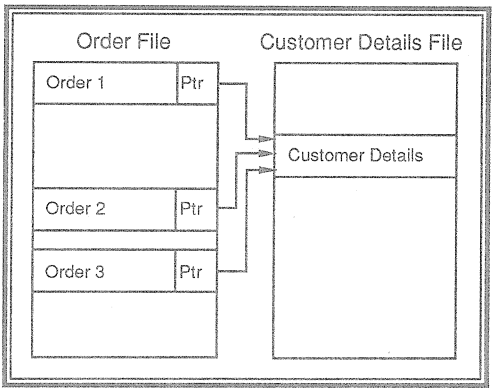


Figure 1. A Hierarchical File Structure

Another disadvantage of a serial file is that of locating any one particular record. In the worst situation the only recourse is to start at the beginning of the file and read through each record until the one required is found. This also takes time. If a file can be ordered in some way, then various techniques do exist which enable an individual record to be found much more quickly (binary search, for example).

Finally, a serial fixed-length record format, while straightforward to process, is inherently inflexible, and can be very inefficient in its use of disc storage (and maybe memory). Suppose, as an example, we wanted a file containing the names of cricketers, and that we wanted to record for each their batting scores during a season and the names of the teams against which these were made. To repeat the name of each cricketer in a record for each match would clearly seem to be unnecessary, but how to avoid this? Likewise, the number of scores would not be known in advance. How can we cater for this?

USING POINTERS

The answer to these questions, and the key to more flexible, if sometimes more complex, file handling lies in the use of pointer fields. A pointer field is simply a field like any other that is part of a record. But rather than supply a piece of data itself, it acts as a pointer to another record containing the required data.

To add a further dimension to this, a pointer field may relate to another record within the same file, or a record within a completely separate file. Furthermore, there is no reason why a record should be restricted to a single pointer field. Thus any record may be linked to several other records, any or all of which may be in the same or different files.

If you can visualise such a system with many such links and pointers then we are beginning to see the true nature of a database system. This is much more complex than the many single file handling packages available for the Beeb, though these often claim to be 'database' systems.

Sophisticated database systems using pointers are generally referred to as hierarchical databases. The use of a pointer system such as we have described has to be defined for any one application, and once set up can be difficult to modify to reflect changing needs. There is an

alternative approach which can achieve the same results but which is less rigidly designed. This is the so-called relational database.

Suppose, in a business environment, we have a computerised order processing system with a file of orders. Each record will refer to a particular customer, but to avoid duplication we will keep the details of our customers, such as name, address, credit rating etc. in a separate file. We have to establish a link or relationship between the two files. We could do this by including in each record in the order file a pointer to the corresponding customer in the customer details file. This file structure is illustrated in figure 1.

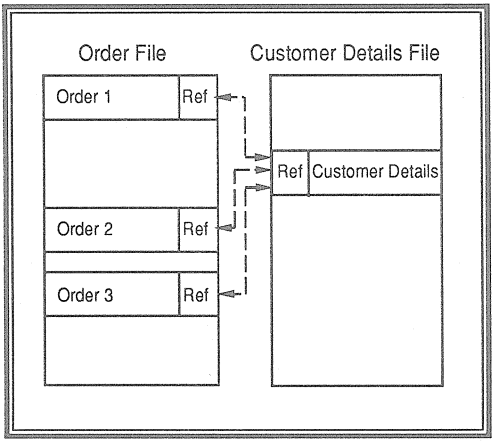


Figure 2. A Relational File Structure

Alternatively, we could assign a reference number to each customer. Each record in the order file would contain a customer reference number. Each record in the customer details file would also contain the assigned reference number as part of the record. There is no direct pointer from one file to the other, but from the order record we could look up the corresponding customer details by locating the record with a corresponding customer reference. What is more, for each customer in the customer details file we could look up all the related orders in the order file (see figure 2). We could only achieve this using a pointer system if a record in the customer details file also had a pointer back to the order file, and we would still have the problem of how to link together a set of orders relating to any one customer.

INDEXING FILES

One further point needs to be made before we look at the use of pointer fields in practice. You may feel from the description above that relational databases have all the advantages, but this is not so. Since the relationship between records depends upon locating a record containing a specific piece of information, the record *key*, there has to be some way of locating the record key in the relevant file. If we have to search sequentially through the file for the required record then processing will be very slow indeed. Pointers, on the other hand, point directly to the required record. The task of locating a particular record is clearly an important one in file handling, and more sophisticated database systems often rely upon some kind of *index* system.

Given a data file, we can select any field within a record and construct an index for that field. In order to locate a particular record, we then just look up the key for that record in the index. That in turn demands that the index be organised so that looking up a key in the index is as fast as possible. Once an index mechanism exists, then it should be possible to construct more than one index to a given file, each index based on the use of a different key field. Thus the need to sort a file or maintain a file in a particular order begins to recede as each index can be organised to allow us to access the file in whatever order we choose. We expect to deal with the subject of indexed files later in this series.

POINTER FIELDS IN PRACTICE

So much for theory and ideas. We need now to look at how we can store and use a pointer field simply, and then go on to look at applications of this. As we saw last month, the method of pointing to any position within a file is controlled with PTR#. Thus what is stored in a pointer field must represent a value which can be assigned to PTR#.

At this point we have a choice. We may want our pointer field to contain precisely the value to be assigned to PTR#, or we may want it to store the corresponding record number. In that case, multiplying the record number (less 1) by the size of a record would give the corresponding PTR# value (providing all records are of the same fixed length). A further small adjustment would arise if our data file contained a header record of any kind.

This was the method we used for our previous serial file handling in parts 3 to 5 of this series, and we shall be making further reference to the ideas and related procedures developed then as we proceed with pointers. We will use a pointer field containing a record number, and use this to generate a PTR# value when needed.

Referring again to last month's article, this showed how a two-byte integer could be stored. Using that method would give us a range of values from 0 to 65535, or a maximum of 65535 records. That should be quite enough for most applications. Given a record number *record%* this would be stored in a pointer field as:

```
BPUT#F%,record% MOD 256
BPUT#F%,record% DIV 256
```

assuming that PTR# has already been used, if necessary, to move the file pointer to the start of the appropriate pointer field. To read the contents of a pointer field, we could write:

```
record%=BGET#F%+256*BGET#F%
```

and then, to position the pointer at the record being pointed to:

```
PTR#F%=offset%+record%*record_size%
```

where *offset%* is the size of any file header, and *record_size%* is the size of a record. In fact, all we are doing to store and read a pointer field is to store and read a two-byte number, but then use it to position the file pointer.

INTERNAL POINTERS

Perhaps the simplest application to consider first is the use of internal pointers, that is where a pointer refers to another record within the same file. In fact, most of the ideas we need are the same as those used for linking or chaining together data items stored in memory, and we refer you to the recent Workshops on this subject in BEEBUG Vol.7 Nos.3 to 5.

The most straightforward situation is where all the records in a file are chained together in some order. For example, consider a file of names, addresses and telephone numbers. This could be implemented as a serial file, as described in parts 3 to 5 of this series, with the records chained together in alphabetical order by name (say).

In such a case, the pointer field can be largely transparent. That is, we do not need to name a pointer field as one of the fields in each record, but just alter the coding to maintain and access records using pointers. Just add 2 to the record size (to account for the two byte pointer) when this value is calculated initially.

The File Description Record (FDR) would need to store a pointer to the first record in the chain (it might not always remain the first record), and it would also make sense to chain together all the empty records, with a further pointer in the FDR to the start of this free chain. When a file of this type is created, all the empty records would need to be chained together to initialise the file.

One advantage of this approach is that when records are deleted, the deleted record is removed from the chain of live records and added to the start of the chain of free records merely by adjusting pointers. Thus there is no need to shuffle the records around to remove any 'holes' that may appear.

On the other hand, following the chain to access records will take more time than just accessing each record in the order in which they exist in the file, because the the disc drive read/write head will generally have further to travel from one track to another. In implementing such a file structure, it might therefore be worthwhile allowing both types of access, serial and by using the pointers.

We will also find it useful to include in the FDR a note of which field (we will call it *key%*) is used as the key field for ordering purposes. This again adds flexibility, allowing any field in a file to be designated as the key field for that file (at file creation time).

Opening a file would now include reading the start of the live and free chains, and the position of the key field, as well as other information as before. Writing a new record would involve following the chain until the correct position for the new record was found, taking the first empty block from the free record chain (adjusting the free record pointers accordingly), slotting this into the live chain, and writing the new data to this record. For more information on chains and linked lists see the Workshop articles referred to above, and in particular the diagrams in the Workshop for Vol.7 No.4.

We give below the details of a procedure called PROCchain, the purpose of which is to locate a free record and incorporate this into the live chain, writing the contents of the record buffer to the new record and updating all pointers as necessary.

```
1600 DEF PROCchain(buffer%)
1610 R%=FNfree(RF%):IF NOT R% ENDPROC
1620 PROCfind_pos(key%,RL%)
1630 PROCwrite_record(R%,buffer%)
1640 BPUT#F%,R2% MOD 256
1650 BPUT#F%,R2% DIV 256
1660 PTR#F%=start%+(RS%+1)*R1%-2
1670 BPUT#F%,R% MOD 256
1680 BPUT#F%,R% DIV 256
1690 ENDPROC
```

We have assumed that the FDR contains pointers to the first live record (RL%) and the first free record (RF%), which would be read in from a file when the file was opened. Indeed, a flag could easily be incorporated in the FDR to indicate whether a file was to be accessed serially or by pointers, thus allowing the same program to handle both file types.

The routine first needs to establish that a free record is available, and if so to remove this from the free list by updating RF% to point to the next free record. We have assumed that this is performed by the function FNfree(RF%), and that the value assigned to R% is the number of the new record to be used. Provided a free record is available, this would be the original value of RF%.

The next step is to call a procedure PROCfind_pos(key%,RL%) to determine the position in the existing chain of live records for the new record. The variable key% determines which field is being used as the key for this purpose. In relation to the new record, we assume that this procedure determines values for R1%, the number of the immediately preceding record, and R2% the number of the immediately following record.

We can now call the original procedure PROCwrite_record to output the current record in memory to the new record on disc. It then remains to set the pointer of the preceding record to point to the new record, and the pointer of the new record to point to the following record.

In all of this it is worth thinking of the disc head movement that may result, and ordering the code where possible to minimise this. Thus, the procedure writes the pointer value for the new record immediately after writing the record itself, as PTR# will already be in the correct position. The pointer from the preceding record to the new one is written last.

To enter a new record from the keyboard and write it out to the file, we could use the same procedure PROCenter_record as before (see part 4), but calling PROCchain to put the new record in the file instead of PROCwrite_record.

When reading a record, we simply need to ensure that we read the pointer to the next record as well as the record itself by including the following line in the procedure PROCread_record (see BEEBUG Vol.7 No.4):

```
2555 pointer%=BGET#F%+256*BGET#F%
```

Thus to read and display all the records in a file we could write:

```
pointer%=RL%
REPEAT
PROCdisplay_record(pointer%,buffer%)
UNTIL pointer%=0
```

In many applications, it might in fact be preferable for a file to store many small chains rather than one long one, with more than one pointer per record. We will deal with the reasons for this, and with the use of external pointers next month.

This month's magazine disc/cassette contains a program demonstrating the usage of pointers. B

BEEBUG WORKSHOP

Game Strategy

David Spencer begins a short series on game playing algorithms.

One of the mystical areas of computer programming seems to be the writing of a program to play a strategy game such as chess or Connect-4. This is a great shame, because the techniques required to perform such a task are not really complicated once you understand the basic principles.

In this, and the next two Workshops, we will explain all the principles involved in writing a strategy game. To explain step by step how to write such a game would take forever, so instead we will concentrate on the theory that is needed to enable the average programmer to write a game themselves. To put the theory into practice, next month's BEEBUG will include a draughts game. The reason for choosing this over chess is that a complete chess game would be too long and involved to publish.

PLANNING A STRATEGY

The first thing to make clear is that there is no such thing as a magic formula which when given the current position of the game, throws out the best move to make. Even if there was, with a game like chess it would probably be such a complicated function that it would require a super-computer to evaluate it.

Instead, the process normally adopted for deciding the best move at any time is that of trial and error. If for a given board position, all the available moves are tried out, it is possible to see what the results are, and then make the move that leads to the best result. This is how many novices play games such as chess - they just choose the move which leads to what appears to be the best board position. On the other hand, consider how an expert decides on the best move to make. His thinking might run along the lines: "If I make this move, what will my opponent do? If he replies with that move then what can I do in reply to that?", and so on.

Whereas the beginner just considers what the position is after his move, the expert tries to predict the position after a number of moves have been made by each side. In the simplest case he might predict the move that the opponent will make in reply to his, and examine the position after that. He is however more likely to consider what the position might be in several moves time.

Clearly, the further a player can look-ahead in a game, the more likely he is to choose the best move to make, and some people go as far as saying that what makes a grandmaster is the ability to examine many moves ahead at a phenomenal speed. There is though, an element of uncertainty in this method, because the player needs to predict what move the opponent is going to make in reply to his own move. In general, you assume that the opponent will make the best move he can, but if he doesn't then the entire look-ahead process falls over. Many a chess master has lost to a novice because the novice suddenly played an unexpected, and apparently ill-advised move, throwing the expert off balance.

If this method of looking ahead is good enough for a chess grandmaster, then it is good enough for our computerised chess. Indeed, this method is used by almost all strategy game playing programs.

MOVES AND PLYS

Before looking at our playing technique from the computer's point of view, we need to clarify the meaning of two words - ply and move. In the above description the word move was used to refer to the action taken by the player, or the action taken in reply by his opponent. However, used properly, a *move* is actually the player's action together with his opponent's reply. For example, the first move of a game of chess is white's opening action and black's response to it. The correct term for half a move, in other words the action of a player moving one of his pieces, is a *ply*. Another term that could cause confusion is *position*. Throughout these articles, the word position will be used to indicate the layout of the board at a particular point in the game.

BOARD REPRESENTATION

The first thing to decide when designing a chess game is how the playing board will be represented in memory. As chess is played on an eight by eight board, the best solution is to use an eight by eight array to represent any position. For example, in Basic the statement:

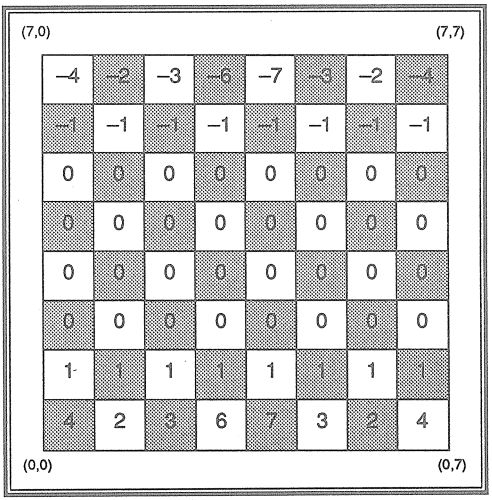
```
DIM board(7,7)
```

could be used to set up the board array. The values stored in each array element indicate the piece at the corresponding position on the board. A value of zero can indicate a blank square, while a square containing a piece could be indicated by one of the following values:

Pawn	1
Knight	2
Bishop	3
Rook	4
Rook (moved)	5
Queen	6
King	7
King (moved)	8

You will notice that rooks and kings have two different codes, one being used if the piece has not moved at all in the game, and the other for when the piece has moved. This allows the program to check easily if castling is valid, because this requires that neither the king nor the rook involved has moved in the game. The above codes can be used as they stand for white's pieces, or negated for black. So, for example, a value of -5 in the array would indicate that that particular square contained a black rook which had been moved during the game.

For our examples, we shall assume that the array is ordered in such a way that the element with subscript (0,0) represents the black square in the bottom left-hand corner of the board, and element (7,7) represents the top right-hand corner. The first number in the subscript represents the rank (row), and the second number the file (column). Therefore, element (0,7) is in the bottom right-hand corner of the board. Note that the co-ordinates are effectively in the order (y,x) and not (x,y).



The Contents of a Board Array at the Start of a Game

MOVE GENERATION

One obvious necessity in our game is a routine which, given a particular position, will produce all the moves that either player can make at that point. The routine to perform this task is called the *legal move generator*. The term 'legal' arises from the fact that moves that would put, or leave, the player in check are avoided. The input to the *legal move generator* will be the current position represented by the data

structure described above, and a flag to indicate whether the list of white or black's moves should be produced. The routine's output will be a list of all the possible moves in some form. One way to represent this would be to show each move as 'from' and 'to' coordinates. However, a more compact form would be a list that gave each piece's current position followed by a list of the positions to which it could be moved.

As an example, consider that a position is represented by a single byte, the top nibble being the rank, and the bottom nibble the file (so the square at position (7,0) would be represented by the value &70). Then, the entry in the move list for the white king's pawn could be:

&14	current position
&24	possible new positions
&34	
&FF	End of list marker for this piece

The list of all possible moves would merely be the combined lists for all individual pieces. An additional byte containing &FF could be used to indicate the end of the complete list.

Having defined the input and output of the *legal move generator*, we need to consider how it is actually going to produce the list of moves. It should be clear from the above comments that each piece can be considered separately. This reduces the task to working through each position looking for pieces of the correct colour. When one is found, that position and all the piece's possible new positions, are added to the list.

Actually producing a list of moves for a particular piece on a particular square is not too hard. For each type of piece it is simple to apply a rule to the current position to get the new position. For example, a pawn can either move one space forward keeping it in the same file, two spaces forward if it is moving from its base rank, or a space forward to an adjacent file if it taking another piece (ignoring the option of taking 'en-passent'). Similar rules can be devised for all the other pieces. Rooks, bishops and queens are special in that they are sliding pieces, and can move by a variable number of spaces. This is easily handled by continuing to apply the move rule in all the possible directions until the piece reaches the edge of the board or hits another piece.

One factor not yet dealt with by the move generator is checking to see if a particular move would leave the player in check. Perhaps the easiest way to handle this is to try out each move as it is generated, and then create a list of moves for the opponent. If one of these 'reply moves' takes the king, then the original move must have left the player in check. This method, while simple, can be very time consuming because each move has to be tried out. An alternative is to ignore illegal moves and rely on the fact that the routine that chooses the best move will reject a move that results in the loss of the king.

THE STATIC EVALUATOR

So far, we have devised a method for storing a position in memory, and a routine to work out all the possible moves that can be made from any position. Before dealing with how we decide the best move to make, we need a routine to examine a particular position, and come up with a measure of how good that position is for either player. A routine that does this is called a *Static Evaluator*. The word *Static* implies that a single, isolated, position is examined, rather than, say, also taking into account the moves made to reach that position.

In general, the score returned by the *Static Evaluator* will be a single number, which for practical purposes is best kept as an integer. It is conventional to have a score of zero indicating no advantage to either player, with a positive score representing an advantage to one player, and a negative score an advantage to the other player. The magnitude of the score indicates the advantage that the particular player has. For example, the *Static Evaluator* for a chess game could return a score with magnitude ranging from zero up to 1000. As already said, zero indicates no advantage, while a score of a 1000 means that one player is ahead by a very long way. Similarly, a score of -

1000 would imply that the other player had the large advantage.

Because the score given to a position is balanced around zero, it is normally calculated using a *static evaluation function* applied to both players in turn. This function yields for each player a number that represents his strength in the game. By applying the evaluation function to both players in turn, and subtracting the results, you end up with the desired score that is balanced around zero.

The million-dollar question is how do you quantify a player's position into a single number that realistically reflects that player's strength. Generally, there is more than one factor to take into account. In the case of chess, the number, and type, of pieces remaining affects the strength a lot, but so does the freedom that the player has to move, and how much the player is attacking his opponent. All these factors should be considered, although each has a different importance.

In chess, the number, and type, of pieces left will normally be most important in deciding a player's strength. In fact, many beginners play a game of chess by only considering their 'piece count'. The way that the *static evaluation function* copes with all these different elements is to calculate each one separately, and add them together with differing weightings to obtain a final score. In other words, the final score is given by the expression:

```
score=w1*s1 + w2*s2 + w3+s3 + .....
```

where s1, s2 etc. are the individual scores, and w1, w2 etc. are the weights applied to them.

There are now three factors to be considered when designing the *Static Evaluator:*

1. What to consider (pieces, possible moves, etc.)
2. How to give a score to each consideration.
3. What weightings to apply to each individual factor.

The first of these is usually quite easy to resolve by a bit of careful thinking. You just need to think what factors you would take into account if you were playing the game. Similarly, the second task is normally quite straightforward. For example, in a game of chess the best way to quantify the number of moves open to a player is to count the possible moves he can make. A value corresponding to the 'piece count' can be found by assigning a value to each piece type and adding these together for all the player's pieces. The value assigned to each piece reflects its importance, and there is a well defined scale that ranges from one for a pawn, to eight for a queen (see ref. 1).

The hardest factor to judge correctly is the final one in the list above - what weightings to give to each element of the evaluation. This can make the difference between a chess program beating a grandmaster and being beaten by a beginner. Consider for example, a chess game where the ability to castle was considered more important that keeping the player's queen. Such a game would be doomed to failure. Yet, a simple change in the weightings could possibly change the game into a brilliant player.

There is no easy way to choose the weighting correctly. The best approach is generally to make an educated guess, and then fine tune the values. One method of fine tuning is to get the game up and running and try to make it follow the moves of a real game that has been played in the past. Ideally, this trial game should have been between two grandmasters (perhaps in a world championship), and should have ended in a draw after at least fifty moves. You can play against the computer, and make all the moves that one of the real players made, and try to tweak the weightings so that the computer replies with the moves made by the opponent in the trial game. More details of this method are given in ref. 1.

We have covered a lot of theory this month, with no real concrete examples. This is unfortunately inevitable with an advanced and involved subject such as this. Next month I will cover the use of *game trees* and the so-called 'MiniMax' technique to decide on the best move.

Reference 1
The Chess Computer Handbook
by David Levy
Published by B.T. Batsford Ltd.

B

Beebug Education

by Mark Sealey

This month, Beebug Education considers the place of computing and information technology (IT) in the light of recent and imminent Government legislation on Education. As a foil to this, we will also look at an excellent new book by two of the most widely respected practitioners in educational computing.

THE NATIONAL CURRICULUM AND INFORMATION TECHNOLOGY

It is not appropriate here to discuss the pros and cons of the imposition of a National Curriculum on Britain's schools and colleges. What cannot be ducked, though, is some assessment of its likely impact on the use of information technology (IT) in schools. Two questions arise. What provision do the proposed changes make for work in IT, and how does Acorn equipment meet the challenge?

Two reports on curriculum areas were published during the summer: one on Mathematics, and one on Science and Technology. Each has provoked a mixed reaction, with critics pointing out that models like these - based on notional *attainment targets* (ATs) - are rather outdated.

In neither case does IT or educational use of the computer figure particularly prominently. Both are to be *core* subjects, and emphasize the use of computers as a means to understanding the real world. Simulations are also mentioned (in the Science report). There is certainly plenty of software for BBC machines here.

One of the ATs for the Technology component is *Designing and Making*. Yet if pupils who had worked on such a project subsequently wanted to write it up using a word processor, then an integrated approach would have to be adopted. The legislation hardly envisages the necessary sort of in-service training needed for teachers to work in this way. Sometimes it would be equally appropriate to record the results of an ecology survey, say, in *Masterfile* or *Quest*. But for the National Curriculum, that is another subject area, and the emphasis given by Baker's proposals to subject division militates against integration. It does not promote it.

The BBC machines with their many interfacing possibilities (User Port, Analogue to Digital converter and 1MHz bus) ought to fit nicely, for instance, into Technology AT 1: *Work with controlling real objects*. By the same token, the computer's affinities with Teletext and the many good comms packages equips it particularly well for Science AT 11 - *Information transfer*. Indeed, the report is to be applauded for stating quite emphatically that: "Children's normal work in all areas should involve where appropriate the use of IT". What is not addressed is how cash-strapped schools will deliver on this with all the major comms networks now levying a time-charge.

As for mathematics, the only real acknowledgement of the importance of microcomputers in everyday life is contained in ATs 10, 11 and 12, which concern the handling of data. Pupils' work with flowcharts and tree-diagrams is mentioned specifically - as are the collection and interpretation of data in a variety of formats. Software like *QMAP* and the mapping facilities of *FIND*, *KEY* and *DATPROBE* stand the BBC B in very good stead for such work. It is a little disturbing, though, that an appendix to the report (No. 5, page 130-1) cites one of the most heavily criticised and confusing of the early programs to attempt the marriage between crude graphics and data, *Eureka*!

There are many excellent packages published for true investigation and problem-solving work, from the MEP INSET pack, *Posing and solving problems with a* micro, *the Association of Teachers of* Mathematics' *L*, to *Droom* and *Dust* (Resource). The approach of these activities is broadly in line with the last major official examination of mathematics teaching, the 1982 Cockcroft Report. In some respects, the latest National Curriculum proposals have thrown over many of the principles which Cockcroft advocated.

Sadly, only one of the 150 odd pages of 'Mathematics 5-16' addresses the use of computers directly. Paragraphs 3.34 to 3.38 take a very narrow and at times dogmatic view of computer use, with a strong implicit secondary bias. Nothing is said to add weight to the long recognised need for integration.

Where does that leave Acorn users? In fact, they have access to machines more than adequately equipped to meet the requirements of a new curriculum. The software is here already. The peripheral support is excellent. What is most disturbing is the fact that such a state of affairs existed three or even four years ago. None of the exciting developments since (in desktop publishing, truly content-free suites, the 32 bit generation and integrated packages like the Inter series, for instance) *needs* to be around for the IT requirements of the new National Curriculum.

Nor can you legislate to have teachers equipped and prepared overnight to use these resources with the imagination that is shown by the best in the profession. Such a high standard depends on maintaining a flow of quality software. No aspect of Government intervention in IT in schools does much to support the industry that is likely to provide it.

It is a chilling thought that the power and accessibility of BBC computers may well end up being put to use in maintaining pointless attainment scores of pupils as they pass through the various inflexible hoops envisaged by the draft legislation.

TEACHER FRIENDLY - BOOK REVIEW

Title	Teacher Friendly
Authors	Chris Drage and Nick Evans
Publisher	Learning Development Aids, Freepost, Wisbech, Cambs. PE13 2BR.
Price	£14.95 (no VAT)

The contribution of Acorn's computers to educational good practice has in the past been rich and varied. A recent publication on the subject shows just how much so, and ought to be added to the library of anyone involved.

This important new publication is for all teachers and educationalists working in any way with BBC computers or IT - from nursery classes to further and higher education colleges.

Very reasonably priced, the 300 plus pages cover most aspects of the theory and practice of Acorn computers in education (including the Electron, Compact and Archimedes). It is divided into five sections. These cover the rationale of educational computing, the use and management of equipment, the software, and an outlook to the future.

Coverage of software is subdivided. The computer is thought of as a cross-curricular tool (in such subject areas as science or history), and as a teaching aid in its own right through such activities as desktop publishing and Logo.

What makes the book special, though, is its treatment of these themes *historically*. From an assessment of the importance and impact of the first Department of Industry 'Micros in Schools' scheme, via the Microelectronics in Education Programme (MEP), to the likely place in the future of 32 bit school administration packages.

Consequently, the book is not a succession of personal opinions on the merits of one program suite over another. It attempts to show how the provision of resources has evolved and benefitted from three things. The feedback from the teachers, from the many semi-professional organisations established to support IT, as well as through advances in the technology.

The sections on buggies and control devices are a good example. The result is that those not directly involved in day-to-day work with pupils, such as governors and parents, can also inform themselves of the issues. The authors are very familiar with the need to support all these processes if the world lead which Britain has established in educational computing is not to be lost.

The authors are all too aware of the pitfalls and drawbacks to the successful use of IT in schools and, by presenting an implicit vision of what can be achieved, go a long way to suggesting some of the answers. This is an excellent book, and with its attention to detail, should be required reading. B

Using Assembler (Part 5)

This month Lee Calcraft concludes his series with a look at Direct Screen Access.

In many cases, whether you are programming in Basic or assembler, there will be no need to directly access the screen. You can quite adequately display text using the PRINT statement or the OS call OSWRCH. Similarly, virtually any graphical requirements can be met using the Beeb's extensive graphics commands, while machine code programmers can use OSWRCH to perform the equivalent of VDU25, as described in parts 1 and 2 of Using Assembler. However, there are some applications where these routines are just not fast enough, and the programmer must resort to direct screen access. Almost all professional graphics-based games use direct screen access for this very reason, as do some word processors. But the price paid for the extra speed is that the program will not work across the Tube.

MODE 7

In this article, we will be taking a look at direct screen access on the Beeb; and we will begin with mode 7, because it is by far the simplest. In this mode, each of the screen's 40 by 25 character positions is represented by a single byte of RAM, and the character displayed at any position is defined by the value of the byte held at the corresponding location in screen RAM. This ranges from &7C00 to &7FFF, and if we poke the value 65 to RAM location &7C00, the letter "A" will appear at the top left-hand corner of the screen. To check this, try:

```
MODE 7
?&7C00=65
```

To make the letter "B" appear at the start of the next line, use:

```
?(&7C00+40)=66
```

This places ASCII character 66 at a position 40 bytes from the screen start. The only thing that you must watch for is that the screen has not been scrolled since the last mode change. Scrolling on the Beeb is performed in hardware, and results in an altered screen map.

THE GRAPHICS MODES

All this is easy stuff. But things get harder in the graphics modes. Here, all screens are bit-mapped. This means that one or more bits of any given byte of screen RAM specify the colour of a given pixel of the screen. There is thus no way that you can poke a value to the screen, and create an ASCII character. The best you will produce is a pattern of up to 8 pixels in size, depending on the mode in use. Try for example:

```
MODE2
?&3000=3
```

This will produce two adjacent pixels of red at the top left-hand corner of the screen. If you then add:

```
?&3001=63
```

two pixels of white will appear immediately below the red bar.

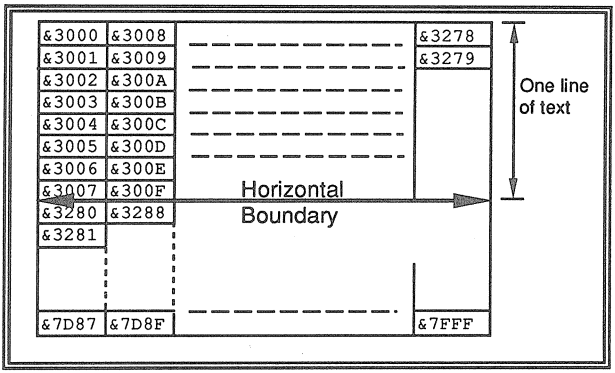


Figure 1. Mode 2 Screen Mapping

Each graphics mode uses a different bit map, and for the sake of clarity we will concentrate on mode 2. This is the only graphics mode which permits the Beeb's full range of colours. Figure 1 gives a representation of the screen

map for this mode. Each address on the map contains one byte of data, which itself contains information about two adjacent pixels. As you can see, the screen is arranged in blocks of eight vertically stacked bytes. These blocks are placed adjacent to each other in such a way that the ninth byte of the screen contains the data for pixels 3 and 4 of the very top line of the screen. This results in very fast text handling because the data for individual characters does

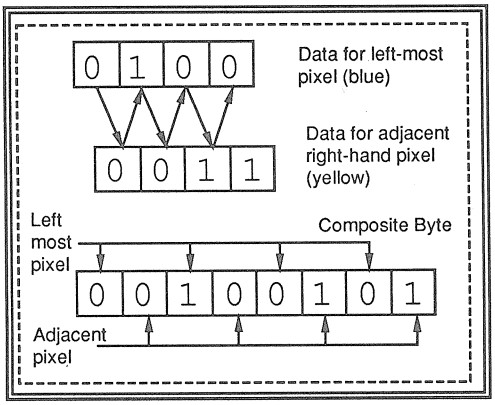


Figure 2. The Data for Adjacent pixels in Mode 2 is interleaved

not cross the horizontal screen boundaries. Plotting sprites which straddle these boundaries is a more involved affair, as we will see in a moment.

To add to this complexity, data held in each screen byte is not stored as a pair of adjacent nibbles; the data is interleaved, as indicated for mode 2 in figure 2. Only the two-colour modes are blessed with non-interleaved pixel data. The value for each of the 16 possible colours in mode 2 follows the number associated with Basic's COLOUR statement, with 1 generating red, 2 giving green, and so on. The colour value of each pair of adjacent pixels must then be combined, by taking alternate bits from the respective binary values. Thus, to place a blue pixel (value 4) next to a yellow pixel (value 3), we would use a byte value of 37 (binary 0100 interleaved with binary 0011, giving binary 00100101).

To generate a mode 2 multi-colour sprite of size 8 by 8 pixels we would need 32 bytes of data, each byte holding the merged colour values for a pair of adjacent pixels. The best way to obtain data for a given sprite is to use a sprite editor (for example, see BEEBUG Vol.6 No.8 p.23). But failing this, you can achieve reasonable results with paper and pencil. The program in listing 1 creates and plots a simple 8 by 8 sprite in mode 2, and figure 3 shows the order in which the pixel pairs are arranged. There are 32 of them, arranged in four vertical groups of 8. By arranging them in this way, we can plot the 32 byte block very quickly at 32 sequential locations in screen RAM - providing that we are not plotting across a horizontal boundary.

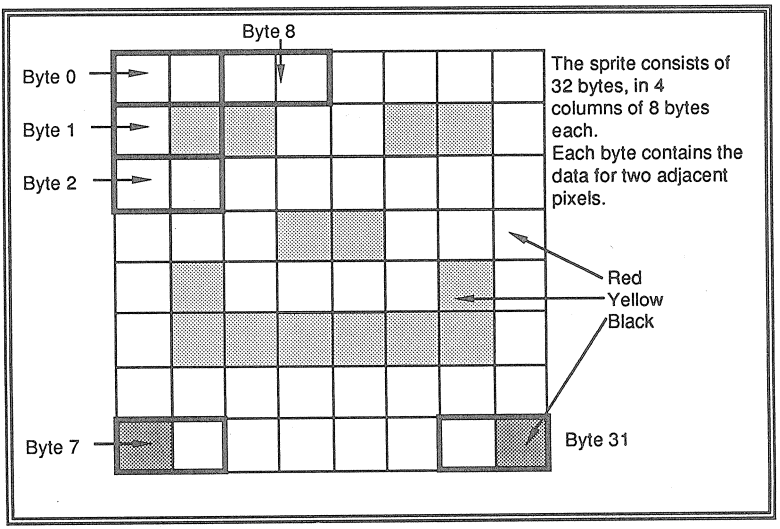


Figure 3. Constructing a Simple Mode 2 Sprite

Listing 1

```
 10 REM Mode 2 Sprite Plot
 20 REM Author Lee Calcraft
 30 REM Version B 0.3 .>Uass5-1-3
 40 MODE2
 50 :
 60 FOR A%=0 TO 31
 70 READ B%
 80 ?(&3000+A%)=B%
 90 NEXT
100 PRINTTAB(0,5)
110 DATA 3,7,3,3,7,7,3,1
120 DATA 3,11,3,7,3,15,3,3
130 DATA 3,7,3,11,3,15,3,3
140 DATA 3,11,3,3,11,11,3,2
```

To alter the screen position of the sprite by two pixels horizontally, we only need to alter the base plotting address by 8 bytes. If such an increment is acceptable, (you will find that it gives reasonably smooth movement), then horizontal movement is trivial. If you wish to move objects with single pixel increments horizontally, you will need to unpack the pairs of bytes which make up the object.

EXCLUSIVE OR PLOTTING

The way to make the sprite appear to move around the screen is to plot it at a given position, then erase it, and plot it at an adjacent position, and so on. Using conventional Beeb graphics this is usually performed with so-called Exclusive OR plotting. The reason being, that the sprite can be removed by overplotting it at the same position in Exclusive OR mode. If we want to use Exclusive OR plotting when we are directly accessing the screen, we cannot just issue a GCOL3,n. We must do it the hard way. This is not really very difficult. It just means that instead of storing a byte at a given screen location, we must first read the current byte at that location, and Exclusive OR it with the byte to be plotted, before sending the result to the screen.

The program in listing 2 puts these ideas into practice. When it is run, mode 2 will be selected, and a red and yellow sprite will repeatedly glide from left to right across the screen, descending with each pass until it reaches the bottom (but without ever straddling a horizontal boundary). The sprite itself is defined as a sequence of 32 bytes of data, as before, and read at line 660 of the program into a dimensioned area of RAM called *data*.

To display the sprite at any time, we can simply store the 32 bytes at the appropriate position in screen RAM. We can move the sprite two pixels horizontally by altering the screen address of the 32 byte block by 8 bytes in either direction. The routine which displays the sprite is listed at line 420. It uses indirect indexed addressing, with the Y register holding a value which increments from 0 to 31, as each 32-byte block is plotted. The screen address is held at &70, and the base address of the sprite at &72. The sequence of instructions:

```
LDA (scrn),Y
EOR (spr),Y
STR (scrn),Y
```

does all the work. It fetches the current byte at the screen, Exclusive ORs it with the next sprite byte, and stores the result back on the screen.

The main plotting loop is controlled by the X register, which ranges in value from 0 to 79, representing 80 consecutive horizontal positions in each left-right scan. You will see that *JSR plot* is called twice by the main body of code for each value of X. The first call plots the sprite, while the second un-plots it, after a call to OSBYTE 19 to synchronise plotting with the vertical flyback of the monitor.

Listing 2

```
 10 REM Simple Sprite Animation
 20 REM Author Lee Calcraft
 30 REM Version B 0.4 .>Uass5-2-4
 40 :
 50 MODE0
 60 DIM code &100,data 32
 70 osbyte=&FFF4:osrdch=&FFE0
 80 scrn=&70:!scrn=&3000
 90 spr=&72:!spr=data
100 :
110 PROCdata
120 PROCassem
130 MODE2
140 FOR N=0 TO 31
```

```
150 CALL code
160 NEXT
170 END
180 :
190 DEFPROCassem
200 FOR pass=0 TO 1
210 P%=code
220 [
230 OPT pass*3
240 .code
250 LDX #0
260 .loop1
270 JSR plot    \Plot sprite
280 JSR wait
290 JSR plot    \Erase sprite
300 CLC
310 LDA scrn
320 ADC #8      \Next position
330 STA scrn
340 BCC nocarry
350 INC scrn+1
360 .nocarry
370 INX
380 CPX #80     \Test end of line
390 BNE loop1
400 RTS
410 :
420 .plot
430 LDY #0
440 .sprloop
450 LDA (scrn),Y\Get screen byte
460 EOR (spr),Y \ExOr with pix data
470 STA (scrn),Y\Send to screen
480 INY
490 CPY #32     \End of sprite?
500 BNE sprloop
510 RTS
520 :
530 .wait       \FX19 frame pause
540 PHA:TXA:PHA:TYA:PHA
550 LDA #19
560 LDX #0:LDY #0
570 JSR osbyte
580 PLA:TAY:PLA:TAX:PLA
590 RTS
600 ]
610 NEXT
620 ENDPROC
630 :
640 DEFPROCdata
650 FOR A%=0 TO 31
660 READ B%
670 ?(data+A%)=B%
680 NEXT
690 ENDPROC
700 :
710 REM Data for 32 byte sprite
720 DATA 3,7,3,3,7,7,3,1
730 DATA 3,11,3,7,3,15,3,3
740 DATA 3,7,3,11,3,15,3,3
750 DATA 3,11,3,3,11,11,3,2
```

VERTICAL MOVEMENT

Because of the way in which the Beeb's graphics screens are mapped (see figure 1 for mode 2 mapping), fine vertical movement is much more tricky. You will see the problem if you try to plot the sprite from listing 1 at a base address of &3001, rather than 3000. Its upper left-hand corner will be in the correct position, but the pixel pair which should have appeared at the bottom left-hand corner will appear at the top of the sprite in the second column. To plot sprites correctly across the screen's horizontal boundaries requires more complex routines. Essentially the software must check to see whether a given byte is to be plotted across a boundary, and if so, the value &279 must be added to the screen address in modes 0, 1 and 2. When the next column of the sprite is plotted, this value must be removed again, as you will see if you examine the screen map in figure 1. This subject is usefully covered in a five part series on machine code graphics, starting in BEEBUG Vol.2 No.8.

HARDWARE SCROLLING

I want to finish the present series with a brief mention of hardware scrolling. This is a technique which allows the whole of the screen area to be scrolled horizontally or vertically with almost no time overhead. You will see the technique in use whenever you list a Basic program of any length. If the 6502 had to alter the contents of every byte of screen RAM every time that the screen scrolled, listing (and many other activities besides) would be very considerably slower.

Hardware scrolling is really a very simple process. There are two registers in the Beeb's 6845 video controller chip which determine the

address which is considered to hold the first byte of screen data. If we alter this address, then the position of the screen display will change, without a single byte of screen RAM being altered. To scroll the screen vertically, we just add (or subtract) the number of bytes per line to the screen address held in the 6845. To scroll sideways, just alter the address by a single byte.

The registers employed are R12 and R13 of the 6845, and these hold the low and high byte of (the screen address DIV 8). For example, to scroll a mode 2 screen to the left by two pixels, use:

```
start=&3008
VDU23;12,start DIV 2048;0;0;0:VDU23;13
,start MOD 2048 DIV 8;0;0;0
```

If you repeat this with *start* set to &3010, the screen will scroll by a further two pixels. To effect a vertical scroll, set *start* to &3000+&280.

When writing a hardware scroll routine based on these principles you must include a line to keep the screen address within the correct range (&3000-&7FFF for modes 0-2). When it goes outside of this range, simply add or subtract &5000 from the address, to bring it back within range. Just one more point: ideally the two registers R12 and R13 should be updated simultaneously, otherwise flicker can result. To avoid this, use machine code to perform the updating, and executé OSBYTE 19 before updating begins, to give as much time as possible before the next frame scan begins.

In most real applications of hardware scrolling, software will be required to correct the scrolled screen. Thus for example if you are scrolling a screenful of text, the program will need to replace the bottom line of the screen (wrapped around from the top of the screen after the scroll) with the new line to be displayed. Or if the screen is scrolled sideways as a backdrop to a running man (say), two adjustments will be needed. First of all, the man will probably not be required to be scrolled, so he must be rewritten at his pre-scroll position, and secondly, after a horizontal scroll, the screen wraps at the end of each line. This will need to be cleaned up after each scroll. However, the work involved in these tasks is minute compared to the effort of altering the contents of 20K of screen RAM.

As you will appreciate, hardware scrolling is a very powerful tool in the armoury of the machine code programmer. And it is hoped that these brief notes will encourage experimentation with this technique. For further information on this, and on many of the other subjects treated in this series, I can do no better than refer you to the excellent *Advanced User Guide for the BBC Micro.*

Finally I hope that this long double series on assembler programming has been of some interest, and that not too many readers will be breathing a sigh of relief now that it has finally come to an end; though I must admit to a certain sense of relief myself. B

Points Arising....Points Arising....Points Arising....Points Arising....

BEEBUG SUPER SQUEEZE (Vol.7 No.5)

An error in the design of this utility can lead to a slight problem if the program being squeezed contains embedded assembly language.

The problem manifests itself if the variable names A, X or Y are used in the main program. *Super Squeeze* will change the register names in the assembly code to the names substituted for the corresponding variables in the main program.

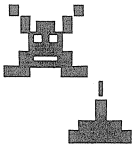
The solution to this is toavoid the use of A, X and Y as variable names if the program contains assembler code.

LABEL PROCESSOR (Vol.7 No.6)

Two errors inadvertently crept into the listing of this program. The lines affected are 1020 and 1030, and these should read:

```
1020 FOR X=0 TO 1:VDU132,157,131,141
1030 PRINT SPC(6);"BEEBUG LABEL PROCESS
OR":NEXT
```

B



Games Update

Mike Williams brings you the latest update on games for the new year.

In our previous issue we covered some of the new games available for Christmas and the New Year. However, the latest mega-game from Superior Software, *Exile*, dropped onto the editorial desk just too late to be included. From the pre-release hype, Exile is clearly intended to convey the same aura of excitement as did Elite when that was first released by Acorn. There are some similarities, the short story to set the scene and the fiendishly complicated controls once the game starts, but a game in the mould of Elite it is not.

Despite that, it is an excellent game in its own right, an action-packed adventure on the surface of an alien planet as you try to fulfil your mission to destroy the evil Triax and his phalanxes of hideous mutated maggots. It may sound laughable, but the animated and spacesuited figure is remarkably realistic, and the real cries of pain have to be heard to be believed.

Exile is a challenge that I am sure will keep the best of games players puzzled and bamboozled for quite a time; certainly I made only a meagre impact. It is not cheap, but still represents excellent value for money.

Superior has also released yet another compilation disc. Yes, you guessed it. It's ***Play it again Sam 5***. This time the barrel seems to have been scraped that bit harder to come up with the goodies. Best by far are two classics in their time, the three-dimensional *Fortress*, originally from Pace, and the irritatingly addictive *Bug Blaster* (and there were many imitations) from Alligata.

The other two games are *Imogen*, a rather delightful graphic adventure from Micro Power involving a wizard, a cat and a monkey, and lots of tantalising puzzles, and *Elixir*, a rather odd game to say the least, in which poor old Cyril tries to find the right pills in the chemist's shop to return him to his normal size. Electron owners get Micro Power's *Moonraider* as an alternative to *Fortress*.

Older Beeb owners may get twangs of nostalgia from these names from the past, but this does not seem to be one of the better Play it again Sam releases. In fact, it's rather like the curate's egg, good in parts, and a toss-up as to whether it's worth the money.

The last two games, both from Impact Software, are *Zenon*, a typical arcade-style action game for one or two players, and *Clogger*, where it is necessary to solve tests of initiative and imagination to achieve the highly desirable Master Clogger status. Zenon reminds me a bit of Stryker's Run, but is more confusing and not as good. Clogger is in the style of Repton, as you mine your way through the underground tunnels, but the final objective of locating and assembling pieces of a jigsaw seems rather weak. However, if you liked Repton then you'll probably like this. Both the Impact games are at budget prices, but even so I would have expected more for my money.

Too late for more than just a mention, Superior has released Repton Infinity (same prices as Exile), claimed as the ultimate Repton program, where you can design your own games in addition to the four provided.

PRODUCT DETAILS (*All prices include VAT)

Product	Exile
Price	£12.95 (BBC/Electron cassette) £14.95 (BBC 5.25" disc) £19.95 (Compact 3.5" disc)
Supplier	Superior Software Regent House Skinner Lane, Leeds LS7 1AX. Tel. (0532) 459453

Product	Play it Again Sam 5
Price	£9.95 (BBC/Electron dual cassette) £11.95 (BBC 5.25" disc)
Supplier	Superior Software Regent House Skinner Lane, Leeds LS7 1AX. Tel. (0532) 459453

Products	Clogger & Zenon
Price	£4.95 (disc or cassette)
Supplier	Impact Software Neepsend House, 1 Percy Street, Sheffield S3 8AU. Tel. (0742) 769950

B

POSTBAG

POSTBAG

FILE HANDLING IN BASIC I

I would like to make a comment on your series on File Handling, in particular to the limitations of OPENIN for effective file handling in Basic I. I have been told that without OPENUP, updating a file can only be done by first copying the existing contents of the file to a new file, and then leaving this open for updating. This may be acceptable for small files, but a 100K file, for example, takes nearly 15 minutes to transfer, and the duplication of files gobbles up disc space. This implies that anyone interested in making serious use of data files should swap Basic I for Basic II.

Cliff Lawrence

This is a popular misconception - there is no need to copy files in order to update a file in Basic I. OPENIN in Basic I is identical to OPENUP in Basic II. The table should clarify matters:

Token	Basic II	Basic I
&8E	OPENIN	--
&AD	OPENUP	OPENIN
&AE	OPENOUT	OPENOUT

BEEBUG Vol.7 No.2 contained an article which described the most likely changes needed to ensure Basic I compatibility, including the use of OPENIN and OPENUP. The Basic II chip is available to members from BEEBUG for £21.36 inc. p&p.

BUSINESS GRAPHICS - A PLEA

Of all the many good programs in the magazine, the one which was best for me was Business Graphics by Alan Dickinson in Vol.5 Nos.6 to 8. He really has put a massive program into the model B.

I am now following the current ASTAAD program, now updated for the Master, and my question has to be, "What are the chances of an updated Business Graphics program for the Master".

Lionel Colman

This is not the first request which we have received regarding a Master version of this program. We have spoken to Alan Dickinson on this matter but he now uses a PC, and does not have access to a Master. If any reader wishes to consider upgrading Alan's Business Graphics program to take full advantage of the Master then we would be pleased to hear from them.

A READER'S VIEW

Knowing that editors like to have comments from their readers, here are some which have been accumulating over the last year. I was pleased to see the Archimedes material moved to a new magazine. I am not particularly interested in games, and think that you print about the right amount.

Articles containing remarks like "We do not pretend that this is an easy program to follow . . ." are unhelpful. I am usually as interested in the programming technique as in the programs themselves.

You have been introducing some silly titles to articles (e.g. Now C Here, How to be a Good Mouser, Running a Temperature). In my view these are likely to be missed in a quick search through back numbers for a particular subject, and they make poor entries in an index.

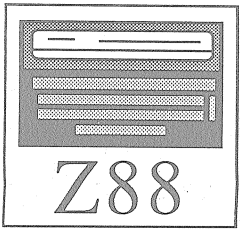
When I had a BBC B, your Basic Toolkit with its utilities for moving pieces of code, renumbering, packing etc. was highly regarded, and its re-appearance for the Master would be welcome.

Dr J.R.Barker

Readers' comments are always welcome, though we do, of course, have to balance widely differing views. On particular points, more explanation could be provided on the workings of programs, but this would use up more space reducing the number and variety of articles and programs.

We hope that the occasional amusing title will add to the enjoyment of the magazine, and indeed make the title more memorable, but we will bear Dr.Barker's comments in mind. In the annual index, titles are often re-arranged so that they appear under appropriate key words for reference, Magscan also allows keyword searching.

A new Toolkit is unlikely, but several similar utilities compatible with the Master have been published in recent issues of BEEBUG, and we may consider offering these on an EPROM as a separate magazine product. B



The Z88 Page

David Spencer reveals the mysteries of printing from Basic.

On the Z88, the printer is treated as a file, much the same as a file saved from *Basic* or *Pipedream*. This concept of treating a hardware device as a file might seem a bit strange, but it is common practice on many larger systems, and also on some micros. Obviously, the printer file is an output-only file (you cannot read text back off the printed page!).

As with any file, a channel must first be opened before anything can be sent to the printer file. This is done with a command such as:

```
ch=OPENOUT":PRT"
```

The filename *:PRT* is a special name which the Z88's operating system reserves for the printer. Before anything can be printed, the printer output must be enabled using:

```
BPUT#ch,5:BPUT#ch,ASC"["
```

Once this has been done, data is printed by sending it to that file using BPUT# or PRINT#. For example:

```
BPUT#ch,13 : REM Carriage Return
PRINT#ch,"Hello World!"
```

An important point to remember is that everything must be sent to the printer as a string. Executing a statement such as:

```
PRINT#ch,23
```

would produce garbage, because Basic will send its internal representation of the value 23 to the printer. Instead, the value must be converted to a string using the STR$ function. For example:

```
PRINT#ch,STR$(23)
```

Once all printing is complete, the printer output must be disabled, and the printer file closed. This is done using:

```
BPUT#ch,5:BPUT#ch,ASC"]"
CLOSE#ch
```

THE PRINTER FILTER

Another concept used on the Z88 which may be unfamiliar is that of the *printer filter*. This is a piece of software that 'filters' the data sent to the printer. One important task of the *printer filter* is to translate certain standard highlight codes into the correct control codes for the printer. The actual codes sent to the printer are altered using the *PrinterEd* application which is described in the Z88 user guide. Another action performed by the *printer filter* is the conversion of characters as defined with *PrinterEd*. One use of this is in producing pound signs if the printer would normally generate a '#'.

Any printable ASCII characters will pass straight through the *printer filter*, as will ASCII codes 0 and 7-13. This allows characters and cursor movement codes to be sent directly to the printer. All other codes, with the exception of ASCII code 5, are filtered out of the data and never actually reach the printer.

ASCII code 5 is different, because it notifies the *printer filter* that a special command will follow. We have already seen two examples of these commands; namely those to enable and disable printer output. The complete list of commands that can follow ASCII 5 is:

`ASC"["`	Printer on
`ASC"]"`	Printer off
`ASC"2",ASC"P",32+len`	Set page length
`ASC"2",ASC"H",32+gap`	Set microspacing
`ASC"S"`	New slot
`ASC"3",ASC"$",hi,lo`	Send code to printer
`ASC"U"`	Underline
`ASC"B"`	Bold
`ASC"X"`	Extended
`ASC"I"`	Italics
`ASC"L"`	Subscript
`ASC"R"`	Superscript
`ASC"A"`	Alternate font
`ASC"E"`	User defined

The first two of these commands were described above. They simply tell the printer filter to commence or cease sending output to the printer. The last eight (Underline onwards) are equivalent to the highlight codes available from within *Pipedream*. The first occurrence of a particular code turns the highlight on, while a second instance turns it off again. Additionally, chosen highlights are turned off automatically at a carriage return. This option is controlled using *PrinterEd*. Connected with these codes is the 'New slot' command. This is issued by

Pipedream at the end of every slot, and turns off all the highlights that would be turned off at a carriage return.

The commands to set page length and character microspacing are not really applicable to printing from Basic, and can be ignored.

The most interesting command is the one for sending an ASCII code to the printer, because this allows all the printer's effects to be controlled directly. The format of this command is somewhat more complex than that of the simple highlights. The first value sent, after the opening 5, should be ASC"3" (51 decimal. This is a count of the number of additional parameters that are to follow. Then comes the command itself, in this case ASC"$" (36 or &24). The *hi* and *lo* parameters that follow are the codes of the two characters that make up the hex value to be sent to the printer. For example, the ASCII code for ESC is &1B in hex. To send this to the printer, the following five values must be sent to the printer file:

5	Command introducer
51	Parameter count
ASC"$"	Actual command
ASC"1"	First hex digit
ASC"B"	Second hex digit

To make life easier, the following procedure sends the ASCII code given as its parameter to the printer, assuming that *ch* is the handle of the printer channel.

```
DEF PROCasc(s)
LOCAL s$
BPUT #ch,5:BPUT#ch,51:BPUT#ch,ASC"$"
s$=STR$~s:IF LEN(s$)=1 s$="0"+s$
BPUT#ch,ASCs$:BPUT#ch,ASC(MID$(s$,2))
ENDPROC
```

For example, to send the code for ESC to the printer, use:

```
PROCasc(&1B)
```

or

```
PROCasc(27)
```

All in all, while the Z88's approach to printing does allow the uniform use of highlights etc., it can cause simple printing to become rather long-winded. B

First Course: The Ins and Outs of Basic (continued from page 32)

the semi-colon at the end of line 110 to avoid the unwanted CR/LF.

In addition, TAB functions can also be included in INPUT statements (as well as in PRINT instructions), and it is therefore perfectly possible to write something like:

```
INPUT TAB(5,10)"Item: "TAB(20,10)
```

data positioning the text prompt, and the input data. Indeed, a variation of this can be placed in a loop repeatedly seeking input until some condition is satisfied, thus:

```
100 REPEAT
110 INPUT TAB(5,10)"Any more (Y/N)? "
TAB(20,10);SPC10;TAB(20,10) ans$
120 UNTIL INSTR("YyNn",ans$)
```

Using the TAB functions to specify row and column positions means that the text and any input are always displayed at exactly the same positions on the screen. This can often be much better than seeing the same message scroll down the screen when a wrong answer is given. The ten spaces, generated with SPC10, ensure that any previous unacceptable answer is wiped from the screen before new input is sought.

If you wish, you could just as easily express the loop above with separate PRINT and INPUT statements, and that enables you to take advantage of the greater output capabilities of PRINT. Thus:

```
100 REPEAT
110 PRINT TAB(5,10)"Any more (Y/N)? "
TAB(20,10)SPC10
120 INPUT TAB(20,10) ans$
130 UNTIL INSTR("YyNn",ans$)
```

When you use the TAB function to control the position of input and output on the screen, there is no essential need for semi-colons at the end of PRINT statements to avoid unwanted CR/LF sequences.

Next month, we will consider other input/output instructions, and look at the special variable @% which provides more detailed control over the formatting of numbers.

HINTS and tips

This month's hints and tips are rounded up by Lance Allison. Remember that we need all the hints we can get, and that we pay five pounds for each hint published and fifteen pounds for the star hint.

*** * * STAR HINT * * ***

RECOVERING VIEW

by Dr R. Parish

If you have ever lost text in View by pressing Break accidentally, the following little program will suit you perfectly. If you press Break and want to recover your text, type *BASIC and then CHAIN "recover". Your text will then be recovered from memory and saved into a file called RESCUED. You may then enter View and load this file to continue editing.

```
10 PRINT"Rescuing"
20 M%=PAGE+&100:m%=M%:REPEAT:IF?M%=&0Dm%
   =M%
30 M%=M%+1:UNTILM%-m%>132
40 S=OPENOUT("RESCUED")
50 FORI%=&901TOm%:BPUT#S,?I%:NEXT
60 CLOSE#S
70 PRINT"Finished"
80 END
```

SCANNING KEYS

by John Flirt

When scanning the Beeb's keyboard to check if any keys are pressed, it is normally possible to use INKEY(0). However, where speed is absolutely vital, for example in a games program, OSBYTE 122 can be used instead, giving a speed increase of about 25%. This call is used thus:

```
A%=122:key=(USR &FFF4 AND &FF00)/256
```

The only problem with this method is that the value returned is the so-called internal key number. To convert this into an ASCII code requires the use of a translation that is provided within the operating system. The address of this table can be found using:

```
A%=172:base=(USR &FFF4 AND &FFFF00)/256
```

If the variable *key* contains the internal key number of a key, then the ASCII code can be found with:

```
ascii=base?key
```

The value returned will be the code for the unshifted key, for example, a lower case letter for alphabetic keys.

DECIMAL IN ASSEMBLER

by David Spencer

One of the most tedious, but frequently used, areas of assembly language programming is writing a routine that will print out a number in decimal. The following short routine, which is easily incorporated in your own programs, will print out any sixteen bit value (0 - 65535) in decimal notation. To use the routine, two bytes of memory should be reserved, and given the label 'num'. For example:

```
num = &70
```

The value to be printed is then poked into these locations (low byte first), and the routine called with:

```
JSR nprt
```

```
1000 .nprt CLC:PHP:LDY#4:.nprt1 LDX #&30
1010 .nprt2 SEC:LDA num:SBC tenlo,Y
1020 PHA:LDA num+1:SBC tenhi,Y:BCC nprt3
1030 STA num+1:PLA:STA num:INX:BCS nprt
1040 .nprt3 PLA:TXA:CMP #&30:BEQ nprt4
1050 PLP:SEC:BCS nprt5:.nprt4 PLP:BCS
     nprt5
1060 PHP:CPY #0:BNE nprt6:PLP:.nprt5
     PHP:JSR &FFEE
1070 .nprt6 DEY:BPL nprt1:PLP:RTS
1080 .tenhi EQUB 0:EQUB 0:EQUB 0:
     EQUB 3:EQUB &27
1090 .tenlo EQUB 1:EQUB 10:EQUB 100:
     EQUB &E8:EQUB &10
```

THEN AND NOW

by John Lasruk

In Hints and Tips in BEEBUG Vol.7 No.3, we said that the keyword THEN could not be omitted from an IF statement if a pseudo-variable was being assigned. For example:

```
IF PAGE=&1900 PAGE=&E00
```

is illegal and would cause an error. However, if the THEN is replaced by a colon then the problem doesn't occur. So,

```
IF PAGE=&1900:PAGE=&E00
```

is perfectly legal, and has the same effect. B

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	£23.00
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£33.00
£20.00	Rest of Europe & Eire	£40.00
£25.00	Middle East	£44.00
£27.00	Americas & Africa	£48.00
£29.00	Elsewhere	

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING

Please add the cost of p&p as shown opposite.

BEEBUG
Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Editorial Consultant : Lee Calcraft
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Head Office Design (0782) 717161 ISSN - 0263 - 7561

Magazine Disc/Cassette

DECEMBER 1988 DISC/CASSETTE CONTENTS

I CHING - use this program to consult the ancient Chinese oracle, and find out what the year ahead has to offer.

PARTIAL RENUMBER - a most useful utility for Basic programmers to renumber selected portions of a program.

SYMMETRICAL PATTERNS - a stunning and continuously changing screen display. This is one of the best examples of this type of program which we have seen.

A MONTHLY DESK DIARY - keep track of all your appointments with this practical approach to computerised diary management.

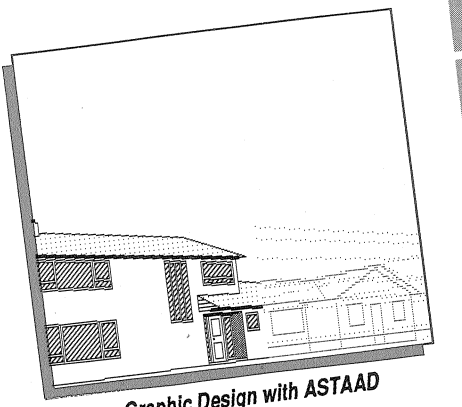
GRAPHIC DESIGN WITH ASTAAD (Part 3) - the updated version of our CAD program ASTAAD, combining parts 1 to 3 in one complete working program.

FILE HANDLING FOR ALL (Part 7) - a fully working example of a database system based on the use of internal pointers as described in part 7 of this series.

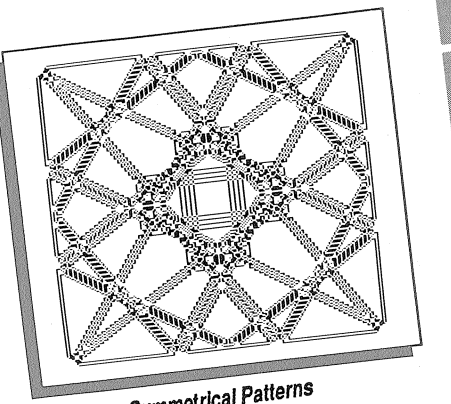
USING ASSEMBLER (Part 5) - the concluding part of this series provides both a sprite plotter and a sprite animator.

UNIVERSAL CALENDAR GENERATOR - an additional item for this month's magazine disc, this program will generate a calendar for any year you choose.

MAGSCAN - bibliography for this issue (Vol.7 No.7).



Graphic Design with ASTAAD



Symmetrical Patterns

All this for £3.50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

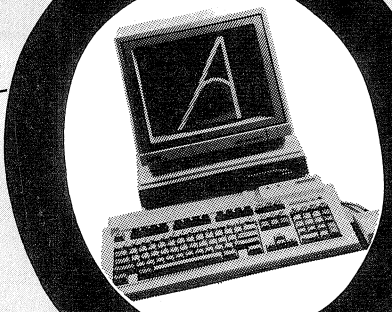
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

Archimedes

Now Available from BEEBUG

featuring

THE NEW RISC OS



0% APR

Beebug the leading specialist dealers in Archimedes offer the best deals as well as comprehensive support for you and your computer.

We are Now able to offer–

0% Finance over 12 months

£50 Voucher to spend as you wish

FREE Printer Lead, Ten 3.5" discs, lockable disc box

FREE *and* RISC OS and Clares Artisan

FREE *or* PC Emulator

In addition, purchasers of a 440 system will receive a free copy of the First Word Plus Wordprocessor.

FREE RISC OS or PC EMULATOR

Subscribe to RISC USER (UK £14.50), the Archimedes magazine, and benefit from the additional FREE offers shown above. In addition you will have free access to our expert technical support team to assist and advise you.

0% FINANCE

Now over 12 months. Please phone for an application form. Return it with the deposit and we will be able to despatch your machine within 48 hours of receipt.

	Cost	Deposit
305	801.60	66.80
305 Mono	870.60	72.55
305 Colour	1054.60	87.92
310	958.24	79.87
310 Mono	1027.00	85.62
310 Colour	1211.00	100.88
310M	1027.00	85.62
310M Mono	1096.00	91.37
310M Colour	1280.00	106.74
440	2906.10	242.23
440 Colour	3159.10	263.35

Featuring Risc OS

The exciting new multi-tasking operating system for Archimedes, RISC OS, will be available early in 1989. Purchase an Archimedes under this scheme and you will be one of the first in the country to receive RISC OS, as soon as it becomes available. Your copy will be sent to you to install, or if you prefer, we will install it for you free of charge.

24 or 36 Months Credit and Trade ins

Should you prefer, we are also able to offer credit over 24 or 36 months at 13.75% flat rate (typical APR 27.5%). We are also able to offer up to £225 for your old BBC computer. Please ask for details.

Education and Health Discounts

Attractive discounts for schools, colleges and Health Authorities. Please ask for details.

TO FIND OUT MORE
PHONE OR WRITE NOW.
TEL: 0727 40303

We offer a complete service, including Advice, Technical Support, Showroom, Mail Order and Repairs. Our showroom in St. Albans stocks everything available for the Archimedes. Call in for a demonstration.

Please indicate your requirements below.
Subscription to Risc User (£14.50 UK) ☐ Information Pack and Catalogue ☐ 0% Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ 12/36 Months Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ *Trade In BBC/Master/Compact* ☐ *Purchase 305/310/440 Base/Mono/Colour* ☐
UK Courier Delivery £7.00. Overseas please ask for a quotation. *Prices include VAT*

I enclose a cheque value £.....

Please debit my Access/Visa/Connect Card No

Expiry...../..... with £.....

Name .....

Address .....

.....

Signature .....

Beebug, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX Tel: 0727 40303

BEEBUG – The Archimedes Specialists