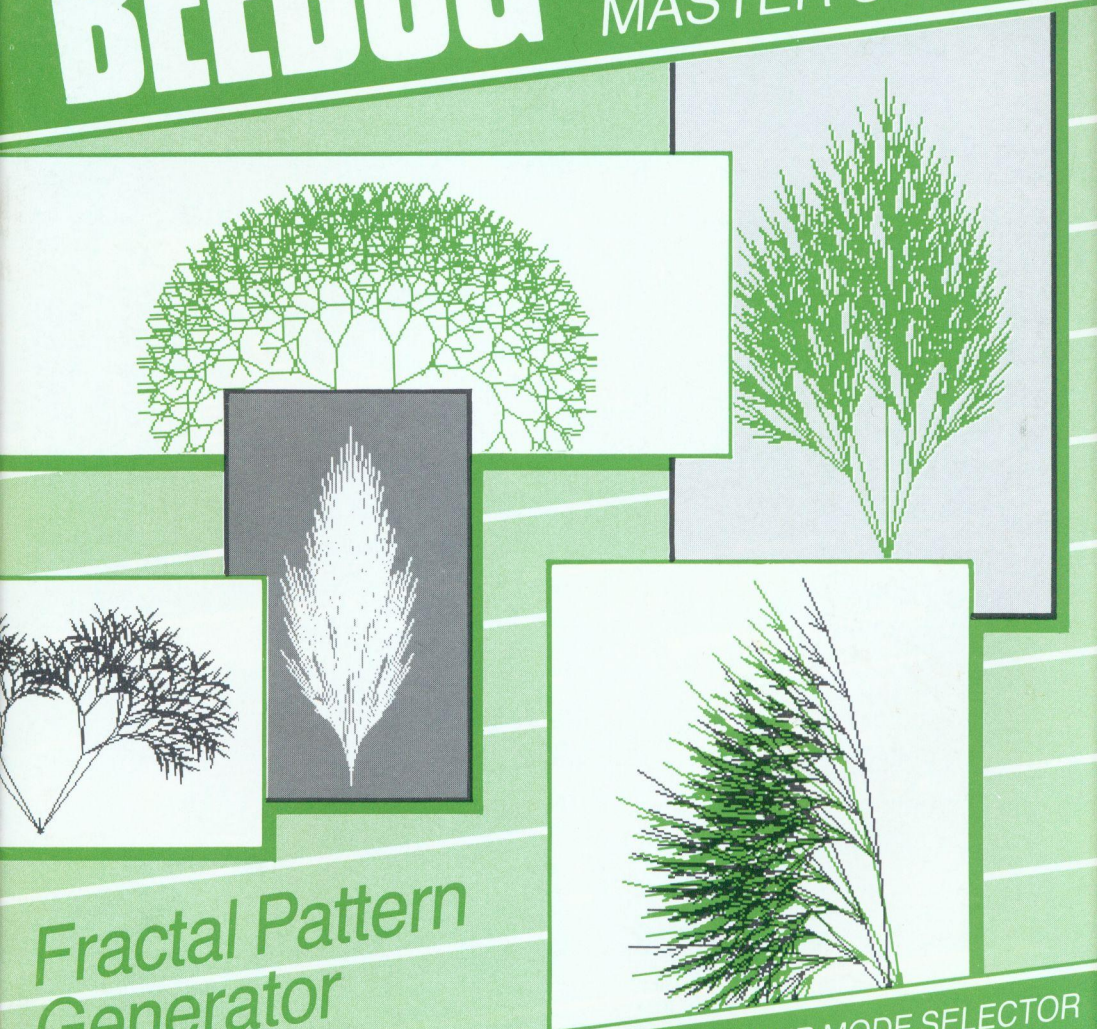
Vol.9 No.1 May 1990

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Fractal Pattern Generator

- PRACTICAL ASSEMBLER
- ADVANCED PRINTER MODE SELECTOR
- USING SIDEWAYS RAM AND ROM
- MECHANICAL VIBRATION

BEEBUG Vol.9 No.1 May 1990

FEATURES

REVIEWS

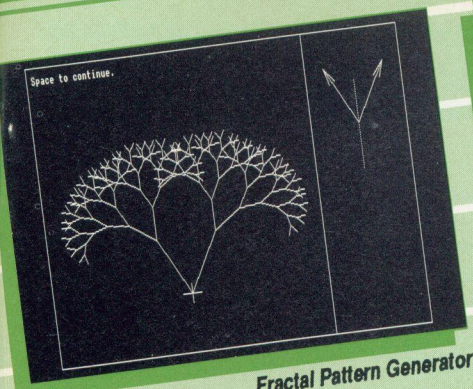
REGULAR ITEMS

HINTS & TIPS

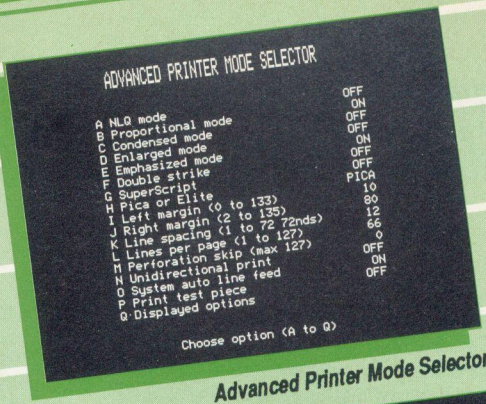
PROGRAM INFORMATION

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 |.

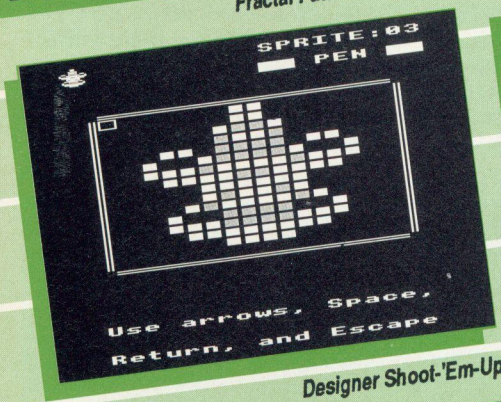
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



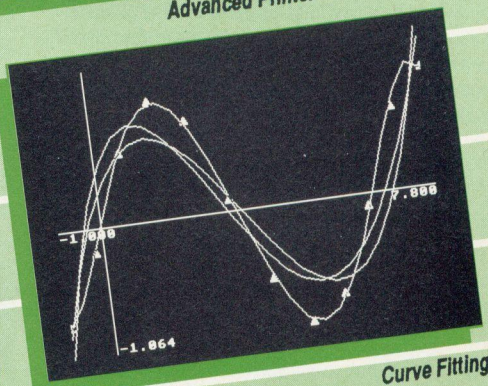
Fractal Pattern Generator



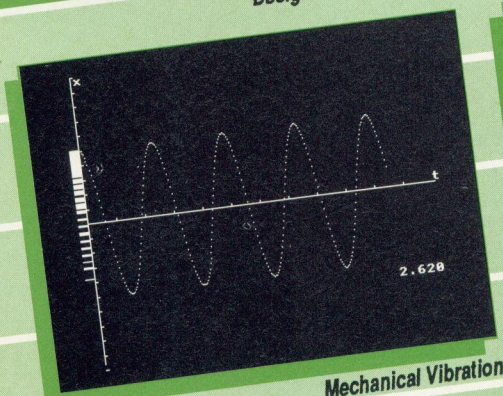
Advanced Printer Mode Selector



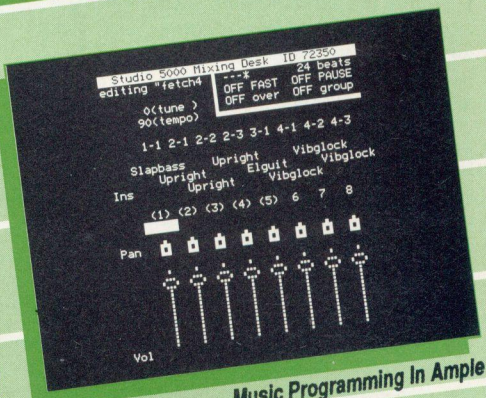
Designer Shoot-'Em-Up



Curve Fitting



Mechanical Vibration



Music Programming In Ample

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are 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

ACORN'S NEW SALES INITIATIVE

Back in the mists of time, when the BBC micro was but a fledgeling, its enormous success prompted several well-known high street stores to stock the Beeb, and indeed many other of the popular home micros of the day. Acorn also achieved a high profile in the consumer home market with some expensive advertising campaigns on television and in the national press.

Times have changed, and few are the high street stores which still stock and sell micros, and fewer still are those which sell Acorn computers. As a result, specialist dealers have become more important, able as they are to offer the consumer much more in the way of expert help and advice on potential purchases.

As a result, Acorn has perhaps come to be seen as more of a specialist educational supplier, but one which still attracts strong support from a discerning and informed user base. But the public, window shopping in their nearest town or city, are unlikely to be that aware of what Acorn still has to offer in the consumer market.

It is with interest, therefore, that we learn of a new initiative by Acorn (briefly reported last month) to put the company back into the public eye, using a new package based around the BBC A3000 microcomputer. This system will include Acorn's *1st Word Plus* word processor, *DR DOS* for PC emulation, *Genesis* a wide ranging information system, which has considerable educational potential, a supporting video, and a reprint, from the Observer newspaper, of a guide to the new schools National Curriculum for parents. The whole concept is being marketed under the name of *The Learning Curve*.

As the contents suggest, *The Learning Curve* is being strongly targeted at families with children of school age, and rests firmly on Acorn's already massive penetration of the schools market. Indeed, there is a parallel initiative which aims to market The Learning Curve through co-operation with schools.

Accompanying this package, which will be available from the beginning of April, Acorn has approved agreements with the John Lewis Partnership and with the Dixons group to promote and sell *The Learning Curve* (alongside Acorn's existing dealer network). There will also be an extensive press advertising campaign, and on TV as well (initially only TVS though). Acorn's intention is clearly to go out and recapture some of the home consumer market where its BBC micro originally enjoyed considerable vogue, and to raise the public's general awareness of Acorn's range of computers.

In this endeavour we offer Acorn our full support. However, it is reasonable to question the quality of support and advice that high street stores can offer when it comes to dealing with modern sophisticated computers. Acorn says it is well aware of this, and is undertaking a major program of training and support with the retailers involved. Let's hope it works. But our experience at BEEBUG would suggest that in many instances the specialist dealer, who trades on his reputation for advice and service, can offer a better deal. As always it is up to the consumer to decide.

NEW LOWER PRICES FOR THE ARCHIMEDES RANGE

Concurrent with the launch of The Learning Curve, Acorn has also announced new lower prices for the whole of the Archimedes range, offering savings of up to £400 on the top-end A440 compared with previous prices. Full details are included under the *News* heading. This is welcome news to Beeb owners contemplating an upgrade.

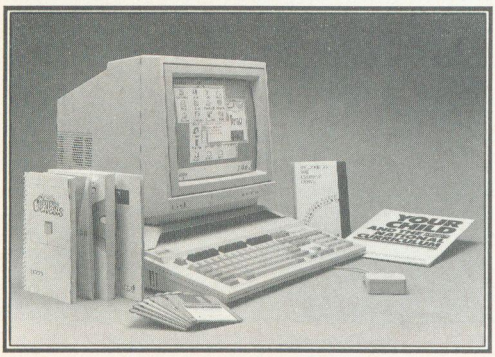
VOLUME EIGHT INDEX

As promised last month a complete index to volume 8 of BEEBUG is included with this issue of BEEBUG magazine. Likewise, the computerised bibliography for BEEBUG, Magscan, has been updated to the end of volume 8.

News News News News News News

THE LEARNING CURVE

Acorn's newly packaged A3000 system called *The Learning Curve* is now available from dealers including BEEBUG. The package includes a standard BBC A3000 32-bit micro, with manuals and applications software, plus a word processor (Acorn's recently updated 1st Word Plus with 70,000 word spelling checker), PC emulation (in the form of DR DOS from Digital Research), a comprehensive information system (Genesis from Software Solutions) including several extensible applications, supporting video and information on the National Curriculum for parents. The system costs £799 inc. VAT (but without monitor) and £1091.35 inc.VAT (with standard colour monitor). Easy payment terms are also available from BEEBUG (details on request).



The new Acorn Learning Curve

NEW LOWER PRICES ON ARCHIMEDES

All current models in Acorn's range of computers have been reduced in price (probably as a result of lower chip prices). The new prices (inc. VAT) are as shown (the old prices are in brackets):

Master 128	£458.85	(£504.85)
A3000	£688.85	(£746.35)
A410/1	£1263.85	(£1378.85)
A420/1	£1723.85	(£1953.85)
A440/1	£2413.85	(£2873.85)

These new prices took effect from the 1st April 1990. All systems are available through BEEBUG (tel. (0727) 40303) from whom full details can be obtained.

Acorn is at Fulbourn Road, Cherry Hinton Cambridge CB1 4JN, tel. (0223) 245200.

NEWS FROM TOPOLOGIKA

Topologika says that after a number of delays, its adventure game *Last Days of Doom* is now available, and costs £19.95 inc. VAT plus 50p p&p. With over 150K of text this game is available only on disc for the BBC micro, Master 128 and Compact (and also for the Electron and Archimedes range at the same price).

Topologika has also announced the availability of its 1990 catalogue with details of forthcoming releases *Astro*, *Untrivia*, the *Giant Killer Support Disc*, and for the A3000 a series of titles under the name of *Freddy Teddy*, based on educational software developed by the Northamptonshire LEA.

For more information contact Topologika at P.O.Box 39, Stilton, Peterborough PE7 3RL, or phone (0733) 244682.

GETTING PERSONAL

Apricote Studios, originators of *The Account Book* and *The Invoice Program* (see reviews in BEEBUG Vol.7 No.5 and Vol.8 No.7) has announced its latest release *Personal Accounts*. This features multiple bank accounts, including direct debits with extensive editing and searching facilities. It is compatible with all Acorn systems (including the Archimedes) and costs just £14.95 inclusive. We hope to review this in the next issue. Apricote are at 2 Purls Bridge Farm, Manea, Cambridgeshire PE15 0ND, tel.(035 478) 432.

HOOTERS

Hooters is the name of a privately published book where the content has been very largely generated by a BBC computer. This publication takes the form of a family puzzle book that combines appealing cartoons, a simple inviting appearance and an unusual competition. For every copy of the book sold, £1 is added to the prize fund which now contains in excess of £1000. The winner of the competition, which involves solving all 50 puzzles in the book, will receive the full value of the prize fund at the closing date of the competition (16th May 1991). The book costs £3.95 (£4.50 by post) and is available only from County Town Books, 7 High Street, Bedford MK40 1RN.

Fractal Pattern Generator

David Lowndes Williams describes a fascinating application of fractals to generate a variety of natural looking shapes.

WHAT IT DOES

The program listed here generates fractal patterns, particularly in the form of fractal trees. A set of vectors (lines with direction and length) is defined by the user. The vectors are then drawn recursively so that the complete set of vectors is redrawn at the end of each of the previously drawn vectors (after appropriate scaling and alignment). This process of adding vectors to the end of vectors is repeated a specified number of times. The results are interesting patterns resembling plant structures and snowflakes depending upon the set of vectors defined initially.

ENTERING THE PROGRAM

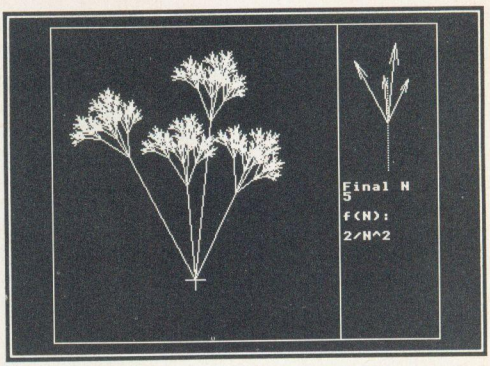
The program will work on all systems, but on a model B with DFS will need to lower the value of PAGE (set PAGE=&1700 and press Return) before loading and running the program.

HOW TO USE IT

When the program is run, you are presented with a screen which is divided into two. On the left is a cross-wire; the fractal image is constructed with this as the starting point. On the right, sitting on a vertical dotted line, is a vector (the arrow). Beneath this is some text, which, together with the vectors provides all the information necessary for the computer to construct the fractal image. Don't worry about this for the moment; the default values are sensible enough for you to use the program without needing to understand it in its entirety at this stage.

Assuming you are starting from scratch: press '1', and a fairly uninteresting curly line will be produced, starting from the cross. Note that the vector points to the right, and that the line curls to the right. The curl consists of three straight lines, and *Final N 3* is displayed on the right of the screen. Note also that the lengths of the straight lines decrease - (the actual ratio is 1:(1/2):(1/3). This is determined by the mathematical function: *f(N)=1/N*

Use the cursor keys to edit the position of the vector. Holding down Shift at the same time as operating the cursor keys makes the vector move faster. Press '1' for the fractal image to be reconstructed. (to start with, keep the vector pointing up, to the left or to the right; then try pointing it down). The patterns only really get interesting when two or more vectors are defined. To define a new vector: press 'V', and the vector on the screen will change colour. Hold down the cursor keys and a new vector will 'grow' from the base of the last one. Using the cursor keys, position this where you like and press '1' for the fractal image to be constructed.



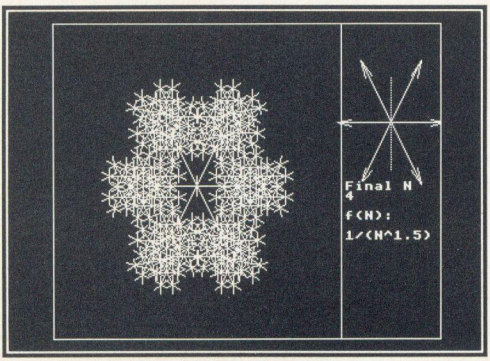
An interesting fractal 'plant'

To change between vectors use the 'A' and 'Z' keys. This enables both vectors to be altered. 'V' can be used to define up to 20 vectors, so the number of vectors is not really a limitation to what can be achieved. The 'A' and 'Z' keys step forwards and backwards through the list of vectors (in the order that they were defined).

By keeping all the vectors close together and pointing up, fractal patterns resembling plant structures tend to be produced. Placing vectors in both up and down directions, in a geometric shape, tends to produce fractal patterns resembling snowflakes.

Some patterns may need scaling and repositioning to produce an effective display. To

reposition the start of the fractal pattern press 'S'. The cross on the left side of the screen will start to flash. The cursor keys control the movement of the cross in the usual manner. Pressing 'S', '1', '2' or '3' gets you out of 'cross editing'. Pressing '1' will redraw the pattern. More complex scaling can be achieved, and this involves changing the mathematical *length* function (see later).



A fractal snowflake design

To change the number of times that the vectors are laid end to end, *Final N* must be changed. To do this press 'N'. You will then be prompted to input the *Number of iterations*; simply type in the number (then press '1' to re-display). The larger the number, the longer it will take to construct the fractal pattern. Experiment with values of 1, 2, 3 etc. in order to become familiar with its effect.

This number is the final value of N. The first set of vectors has N=1. The vectors on the end of these have N=2. The vectors on the end of these vectors have N=3 etc. If the length of these vectors is kept constant the result is a 'muddled' appearance on screen. I prefer to shorten the length of the vectors as the distance from the start (in terms of the number of vectors) increases. In other words, the multiplication factor - and hence the function f(N) - must decrease as N increases. Many possible functions exist, the simplest being f(N)=1/N. When N=1 f(N)=1, for N=2 f(N)=1/2, N=3 f(N)=1/3, N=4 f(N)=1/4 and so on. If you want the lengths of the vectors to be constant then f(N) must be constant, e.g. f(N)=1 or f(N)=2 or f(N)=0.876.

To enter f(N) press 'F', and you will then be prompted to type in the mathematical function - do so and press Return. The function must depend on N (upper case N), e.g. f(N)=2.5/(N^1.6)+(N/62.8). The *Final N* value is stored in V%(0), and you may like to include this in your expression. To scale the pattern, multiply the function by some value (e.g. f(N)=2/N will give a pattern similar to the default but twice the size).

Leaving f(N) at its default setting will still allow you to produce some impressive displays. Escape always returns you to vector editing, while Shift-Escape terminates the program completely. Pressing '2' gives you a mode 0 display (but no text is printed), while '3' produces a list of all the information needed to construct the fractal pattern.

If you want a 'snap shot' facility for a screen dump add the line:

```
305 IF ERR=17:IF INKEY-2 THEN *GDUMP
```

With this line added Ctrl-Escape will issue a *GDUMP command (change this as necessary to suit your own system).

```
   10 REM Program Fractals
   20 REM Version B1.1
   30 REM Author  David Lowndes Williams
   40 REM BEEBUG  May 1990
   50 REM Program subject to copyright
   60 :
  100 MODE1
  110 ON ERROR GOTO 310
  120 :
  130 *KEY0 *FX4,0|M*FX12,3|M
  140 *FX9,8
  150 *FX10,8
  160 *FX12,15
  170 VDU19,2,8,0,0,0
  180 :
  190 PROCInitVariables
  200 :
  210 REPEAT
  220 PROCInitScreen
  230 PROCEdit
  240 IF trap%=TRUE trap%=FALSE:VDU7:g%=
0
  250 IF g%=49 PROCfractal
  260 IF g%=50 MODE0:W%=1:PROCfractal:PR
INTTAB(1,1)"Space to continue.";:REPEAT
UNTIL GET=32:MODE1:W%=0
  270 IF g%=51 PROClist
```

```
  280 UNTIL FALSE
  290 END
  300 :
  310 IF ERR=17 AND INKEY-1 MODE7:GOTO34
0
  320 IF ERR=17 MODE1:VDU19,2,8,0,0,0:tr
ap%=FALSE:W%=0:GOTO210
  330 MODE7:REPORT:PRINT"at line ";ERL
  340 *FX4
  350 END
  360 :
 1000 DEF PROCInitVariables
 1010 DIM e%(2,20)
 1020 DIM V%(100):V%(0)=3
 1030 e%(0,0)=475:e%(1,0)=200
 1040 e%(2,0)=1:e%(0,1)=100:e%(1,1)=200
 1050 A$="1/N"
 1060 trap%=FALSE:W%=0
 1070 ENDPROC
 1080 :
 1090 DEF PROCInitScreen
 1100 LOCAL a%,b%
 1110 GCOL0,3:VDU23,1,0;0;0;0;
 1120 MOVE0,0:DRAW1279,0
 1130 DRAW1279,1023:DRAW0,1023:DRAW0,0
 1140 a%=950:MOVEa%,0:DRAWa%,1023
 1150 a%=1115:MOVEa%,550:PLOT29,a%,850
 1160 GCOL0,1
 1170 FOR b%=1TOe%(2,0)
 1180 PROCDrawVector(1115,700,e%(0,b%),e
%(1,b%))
 1190 NEXT
 1200 GCOL0,3:PROCcursor
 1210 IF W%=0 VDU28,30,30,38,15:PRINT'"F
inal N"';V%(0)''"f(N):"''A$
 1220 VDU26
 1230 ENDPROC
 1240 :
 1250 DEF PROCDrawVector(a,b,c,d)
 1260 LOCAL p,q
 1270 p=0.75:q=0.05
 1280 MOVEa,b:DRAWa+c,b+d
 1290 MOVEa+c,b+d
 1300 DRAWa+p*c-q*d,b+p*d+q*c
 1310 MOVEa+c,b+d
 1320 DRAWa+p*c+q*d,b+p*d-q*c
 1330 ENDPROC
 1340 :
 1350 DEF FNSetVectorLength(a,b,s)
 1360 LOCAL len
 1370 len=SQR(a*a+b*b)
 1380 =s/len
 1390 :
 1400 DEF PROCEdit
 1410 LOCAL flag%,a%,c%
 1420 c%=1
 1430 flag%=FALSE
 1440 GCOL0,0:PROCDrawVector(1115,700,e%
(0,c%),e%(1,c%))
 1450 GCOL3,3:PROCDrawVector(1115,700,e%
(0,c%),e%(1,c%))
 1460 *FX4,1
 1470 *FX138,0,32
 1480 REPEAT
 1490 g%=GET
 1500 PROCDrawVector(1115,700,e%(0,c%),e
%(1,c%))
 1510 IF g%=136 e%(0,c%)=e%(0,c%)-4:IF I
NKEY(-1)e%(0,c%)=e%(0,c%)-12
 1520 IF g%=137 e%(0,c%)=e%(0,c%)+4:IF I
NKEY(-1)e%(0,c%)=e%(0,c%)+12
 1530 IF g%=138 e%(1,c%)=e%(1,c%)-4:IF I
NKEY(-1)e%(1,c%)=e%(1,c%)-12
 1540 IF g%=139 e%(1,c%)=e%(1,c%)+4:IF I
NKEY(-1)e%(1,c%)=e%(1,c%)+12
 1550 g%=g%+32*(g%>96)
 1560 IF g%=65 GCOL0,1:PROCDrawVector(11
15,700,e%(0,c%),e%(1,c%)):c%=c%-1:GCOL3,
3:IF c%<>0 GCOL0,0:PROCDrawVector(1115,7
00,e%(0,c%),e%(1,c%)):GCOL3,3
 1570 IF g%=90 GCOL0,1:PROCDrawVector(11
15,700,e%(0,c%),e%(1,c%)):GCOL3,3:c%=c%+
1:GCOL0,0:PROCDrawVector(1115,700,e%(0,c
%),e%(1,c%)):GCOL3,3
 1580 IF g%=86 GCOL0,1:PROCDrawVector(11
15,700,e%(0,c%),e%(1,c%)):GCOL3,3:e%(2,0
)=e%(2,0)+1:c%=e%(2,0):IF e%(2,0)>19 e%(
2,0)=19:c%=1:GCOL0,0:PROCDrawVector(1115
,700,e%(0,c%),e%(1,c%)):GCOL3,3
 1590 IF c%<1 c%=1:GCOL0,0:PROCDrawVecto
r(1115,700,e%(0,c%),e%(1,c%)):GCOL3,3
 1600 IF c%>e%(2,0) c%=e%(2,0):GCOL0,0:P
ROCDrawVector(1115,700,e%(0,c%),e%(1,c%)
):GCOL3,3
 1610 PROCDrawVector(1115,700,e%(0,c%),e
%(1,c%))
 1620 IF g%=83 PROCEditCursor
 1630 IF g%=70 PROCformula
 1640 IF g%=78 PROCmaxN
 1650 UNTIL g%=83 OR g%=49 OR g%=50 OR g
%=51
 1660 FORa%=1TO20
 1670 e%(2,a%)=SQR(e%(0,a%)^2+e%(1,a%)^
2)
 1680 IF e%(2,a%)=0 AND a%<=e%(2,0) trap
%=TRUE
 1690 NEXT
 1700 ENDPROC
 1710 :
 1720 DEF PROCcursor
 1730 MOVEe%(0,0)-32,e%(1,0)
 1740 DRAWe%(0,0)+32,e%(1,0)
 1750 MOVEe%(0,0),e%(1,0)-32
 1760 DRAWe%(0,0),e%(1,0)+32
```

```
 1770 ENDPROC
 1780 :
 1790 DEF PROCEditCursor
 1800 GCOL0,0:PROCcursor:GCOL3,2:PROCcur
sor
 1810 REPEAT
 1820 g%=GET
 1830 PROCcursor
 1840 IF g%=136 e%(0,0)=e%(0,0)-4:IF INK
EY(-1)e%(0,0)=e%(0,0)-12
 1850 IF g%=137 e%(0,0)=e%(0,0)+4:IF INK
EY(-1)e%(0,0)=e%(0,0)+12
 1860 IF g%=138 e%(1,0)=e%(1,0)-4:IF INK
EY(-1)e%(1,0)=e%(1,0)-12
 1870 IF g%=139 e%(1,0)=e%(1,0)+4:IF INK
EY(-1)e%(1,0)=e%(1,0)+12
 1880 PROCcursor
 1890 UNTIL g%=83 OR g%=115 OR g%=49 OR
g%=50 OR g%=51
 1900 PROCcursor:GCOL0,3:PROCcursor:GCOL
3,3
 1910 ENDPROC
 1920 :
 1930 DEF PROCfractal
 1940 LOCAL safe%,A%,N,a%,b%
 1950 GCOL0,3:safe%=V%(0)
 1960 CLS:PROCInitScreen:IF g%=49 OR g%=
50 VDU24,0;0;950;1023;
 1970 FOR A%=1TO100:V%(A%)=e%(2,0):NEXT
 1980 N=V%(0)
 1990 REPEAT
 2000 PROCline
 2010 flag%=TRUE:V%(N)=V%(N)-1
 2020 FOR a%=N TO 1 STEP-1
 2030 IF V%(a%)<1 V%(a%)=e%(2,0):V%(a%-1
)=V%(a%-1)-1
 2040 NEXT a%
 2050 V%(0)=safe%
 2060 FOR b%=1TOV%(0)
 2070 IF V%(b%)<>1 flag%=FALSE
 2080 NEXT b%
 2090 UNTIL flag%=TRUE
 2100 PROCline:VDU26
 2110 ENDPROC
 2120 :
 2130 DEF PROCline
 2140 LOCAL Sx,Sy,Vx,Vy,OVx,OVy,choice,k
 2150 Sx=e%(0,0):Sy=e%(1,0):MOVE Sx,Sy
 2160 N=1:choice=V%(N)
 2170 Vx=e%(0,choice):Vy=e%(1,choice)
 2180 k=FNfunction(N):Vx=k*Vx:Vy=k*Vy
 2190 Sx=Sx+Vx:Sy=Sy+Vy
 2200 OVx=e%(0,choice)
 2210 OVy=e%(1,choice)
 2220 DRAW Sx,Sy
 2230 IF V%(0)=1 ENDPROC
 2240 FOR N=2 TO V%(0)
 2250 choice=V%(N)
 2260 PROCNewVector(OVx,OVy,e%(0,choice)
,e%(1,choice),e%(2,choice))
 2270 Vx=an:Vy=bn
 2280 OVx=Vx:OVy=Vy
 2290 k=FNfunction(N):Vx=Vx*k:Vy=Vy*k
 2300 Sx=Sx+Vx:Sy=Sy+Vy
 2310 DRAW Sx,Sy
 2320 NEXT
 2330 ENDPROC
 2340 :
 2350 DEF PROCNewVector(as,bs,ae,be,len)
 2360 LOCAL k
 2370 bn=-ae*as+be*bs
 2380 an=ae*bs+be*as
 2390 k=FNSetVectorLength(an,bn,len)
 2400 an=k*an:bn=k*bn
 2410 ENDPROC
 2420 :
 2430 DEF FNfunction(N)=EVAL(A$)
 2440 :
 2450 DEF PROCformula
 2460 PRINTTAB(2,2)"Enter function f(N).
"
 2470 PRINTTAB(2,3)":";:INPUT A$
 2480 CLS:PROCInitScreen
 2490 ENDPROC
 2500 :
 2510 DEF PROCmaxN
 2520 LOCAL N
 2530 PRINTTAB(2,2)"Enter number of iter
ations:"
 2540 PRINTTAB(2,3)":";:INPUT N
 2550 N=INT(N):IF N<0 N=-N
 2560 IF N=0 VDU7:PRINTTAB(3,3)CHR$32;CH
R$32;CHR$32:GOTO2540
 2570 V%(0)=N
 2580 CLS:PROCInitScreen
 2590 ENDPROC
 2600 :
 2610 DEF PROClist
 2620 LOCAL a%
 2630 CLS:PROCInitScreen
 2640 VDU28,1,30,28,1
 2650 PRINT'"List of vectors:"'
 2660 FOR a%=1 TO e%(2,0)
 2670 PRINT"(";e%(0,a%);",";e%(1,a%);")"
 2680 NEXT a%
 2690 PRINT'"f(N)=";A$
 2700 PRINT'"Number of iterations: ";V%(
0)
 2710 PRINT'"Start at: (";e%(0,0);",";e%
(1,0);")"
 2720 PRINT'"Press Space To Continue.";
 2730 REPEAT UNTIL GET=32
 2740 VDU26:CLS:PROCInitScreen
 2750 ENDPROC
```

The Advanced Printer Mode Selector

Use this menu-driven program by Ken Cowap to ensure your printer options are correctly set.

OVERVIEW - COMMAND INTERLOCKS

Several programs have been written to set the styles and output formats of printers; all operate by transmitting the appropriate command codes to the printer in response to selections from a menu. This sounds perfectly simple, but there are pitfalls! This program differs from others in one major aspect - it tells you what codes you shouldn't enter, and stops you trying to do it!

Where many users run into trouble is in not realising that some commands inhibit other commands, and which these are. This results in failure to obtain the desired effects and the printer is often believed to be at fault when it isn't, because inhibited commands are just ignored without any warnings or error messages. For example, in the case of the Kaga KP-810 (near Epson compatible), if you select *Near Letter Quality* (NLQ) output, you can't have Condensed, Emphasised, Double Strike, Super or Subscript, or Elite modes, and there are many more similar instances! Consulting the handbook does show the first five of the above inhibitions, but the non-availability of Elite isn't made clear. Looking up all this information is very tedious, it's not always complete, and I for one gave up long ago trying to commit it all to memory.

The solution offered by this utility is to do it all automatically; each menu option selected turns from white to green, and all the other options which have thereby been inhibited are changed to red. Any attempt to select a red menu option is ignored and a warning sounded.

FEATURES

In addition to the above, a number of other useful features have been built in. The program was designed as a speedy way of setting up the form and style of any text or listing which isn't being prepared using a word processor; these of course contain their own facilities for issuing printer commands.

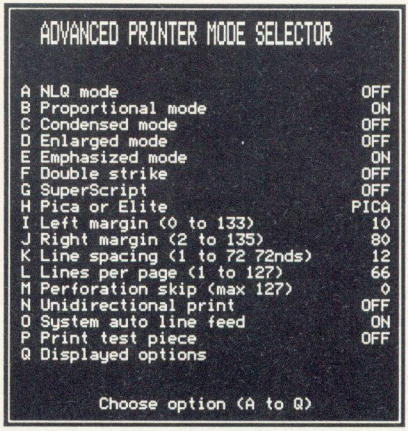
Any printer from the range of Epson compatibles offers a bewilderingly wide selection of different modes, but only a relatively small number of these would apply to a complete block of text or a listing, etc - for example one would hardly ever want the whole thing underlined! In practice the sixteen selections presented in the menu have been found to meet most requirements. The menu option *System Auto Line Feed* was included because it met a practical need, although it's not a printer control code at all - it sends *FX6,0 or *FX6,10 to the computer to switch auto-linefeed on or off respectively.

Those menu options requiring only two states, *Off* or *On*, have been made simple toggle functions - for instance a single press of the "A" key turns NLQ on, a second press turns it off again. The state of any option is shown both by change of colour and by the appropriate legend, so that there is no ambiguity with monochrome monitors. Other menu options requiring a specific (numeric) input need Return to complete the process; this input when keyed in is displayed separately on the line so that the default value remains visible until Return is pressed.

The program has been configured for a standard Epson FX-80 printer (and all those printers compatible with this), though the original was written for a Taxan Kaga KP-810, but any variations may be catered for by modifying the program data as shown below.

A short test piece has been included at option P, consisting of all the printable characters in the ASCII code range 32 to 159 inclusive plus 254 and 255. This choice of codes and the order of

printing reflects the way the Kaga codes outside the "normal" ASCII printing range of 32-127 have been allocated, plus the use of the "printable range expansion" command (code 27, 54) as they call it. Most printer character code designers seem to differ in their ideas about allocation of the supernumerary codes, even among the so-called Epson compatibles, so this routine may need to be amended for your own particular printer.



Printer mode selector menu

No commands are sent to the printer until the Q option is selected, so changes can be made to the selections until that time. A check is made to see if the printer is on line before any commands are sent; if it isn't, an alarm bell sounds and an appropriate message is displayed. After putting the printer on line, transmission of the commands then proceeds automatically; there is no need to reselect the menu options, and an acknowledgement message confirms the transmission.

Comprehensive safety nets have been devised for the menu options so that out-of-limit entries are ignored and the option remains available. Beware of any really silly entries which may corrupt the menu requiring a re-run of the program.

USING THE PROGRAM

Simply type the program in, save it and run it for the main menu display. Simple on/off options are toggled by successive presses of the related key. Options requiring specific values need Return after entering the value. If a numeric option is selected in error, simply press Return for its default value. When all selections have been completed, press 'Q' and the codes will be sent to the printer. Should the printer be off line, a warning will be given to this effect; simply switch the printer on and the codes will be sent to it without any further need for selection.

PROGRAM NOTES

Immediately before the selected command codes are sent to the printer, the program sends the codes for Printer Reset and Printer Enable to ensure that the printer is always initialised to a known state before setting up afresh. This should be borne in mind if other printer routines are used such as for downloading a font, since Printer Reset would erase such a font. So download any font after using this program.

PROCEDURES AND FUNCTIONS

PROCinit sets: cursor off; format for printing numbers; screen colour strings; and all the default option variables.

PROCcheckprinter. Looks to see if the printer is on line, by sending it a couple of null characters (line 1790) and then checking if the printer buffer was cleared (i.e. printer on line) - line 1810. If not, it does the necessary with an error message (line 1820) and an alarm sound produced by PROCalarm which simulates a bell.

PROCdel is a delay routine, length selectable for bell duration, message display etc.

PROCmenu sets up the menu display screen.

PROCtestpiece sends out the codes required to print all characters.

PROCchecka, **PROCcheckb** and **PROCcheckc** are three procedures which are used to check responses to the three different types of menu

option (interlocked on/off, numeric, and unlinked on/off).

PROCdisable is the heart of the program's unique ability to keep a proper record of the availability of all the menu options that can be "mutually destructive". It keeps a *disable* count for each of the first eight interlocked printer commands in the array ok%(). The command is only made available when the count is not above zero. The procedure also sets the colour of a menu option to white (available), green (selected) or red (disabled).

This stratagem is necessary because a particular menu option may be inhibited by more than one other menu option - thus NLQ is inhibited by any of: Condensed, Emphasised, Double Strike, Superscript, or Elite commands. For example, if NLQ had been inhibited by both Condensed and Superscript, the corresponding element ok%(1) would be incremented to 2 so that both options would have to be de-selected before NLQ could again become available; it is for this reason that the elements of the array ok%() are used as counters rather than as simple on/off toggles in the procedure.

DATA

Most of the data (the information specific to the printer in use) is contained in DATA statements at lines 2060 to 2090, 2150 to 2190, and 2220 to 2370. The sets of numbers (16 in all) in lines 2060 to 2090 are the disable masks for the first eight menu options (a 0 value decreases the count by 1, a '1' has no effect, while a '2' increases the disable count by 1).

The next data, in sets of three, give the default, minimum and maximum values for each of the sixteen options (using zero as a dummy value when necessary). Some detailed adjustments are also made here to cater for non-standard parameters (PICA/ELITE for example, instead of OFF/ON). The final set of DATA statements contain the text strings for the menu options. Note that any limits are explicitly included here.

All the information on status, availability, current, minimum, maximum, and default values for each option is stored in a number of arrays: state$(), ok%(), val%(), val1%(), val2%(), val3%(), and a$() - the latter holding the colour marking (green or blue) for each interlinked option.

CONFIGURING FOR OTHER PRINTERS

To set up the program for your particular printer, in most instances you just need to alter the DATA statements as appropriate. On a Taxan Kaga, NLQ mode is handled differently. To cope with this, change the first three values in line 2110 from 0,0,1 to 80,80,40, then delete lines 1180, 1190 and amend line 1290 to read:

```
1290 VDU1,27,1,val%(8):IF val%(8)<>77
THEN VDU1,27,1,val%(1)
```

Many 24-pin printers have fewer interlocking disabled functions; for example it is often possible to print in condensed NLQ or emphasized Elite. To redefine the disable status in the program for a particular function (function n) take the nth pair of values in the DATA statements (e.g. 11010000, 11212222 for NLQ). The first number is a set of eight flags showing how the first eight menu options are affected when that (the nth) function is switched 'on', and the second for when that function is switched 'off'. The figures used are 0 (this function is disabled by function n being 'on'), 1 (this function is unaffected), and 2 (this function is enabled by function n being 'off').

```
 10 REM Program PrintSet
 20 REM Version 1.10
 30 REM Author  Ken Cowap
 40 REM BEEBUG  May 1990
 50 REM Program subject to copyright
 60 :
100 MODE7:ON ERROR GOTO 230
110 PROCinit
120 PROCmenu
130 REPEAT
140 p%=(GET AND &DF)-64
150 IF p%>0  AND p%<9  PROCchecka(p%)
```

```
  160 IF p%>8  AND p%<14 PROCcheckb(p%)
  170 IF p%>13 AND p%<17 PROCcheckc(p%)
  180 IF p%=17 PROCdisplay
  190 UNTIL p%=17
  200 MODE7:@%=10
  210 END
  220 :
  230 MODE7:@%=10
  240 REPORT:PRINT" at line ";ERL
  250 END
  260 :
 1000 DEF PROCdisplay
 1010 REM Set parallel printer
 1020 *FX5,1
 1030 REPEAT
 1040 alarmflag%=0
 1050 REPEAT
 1060 alarmflag%=alarmflag%+1
 1070 PROCcheckprinter
 1080 UNTIL printeroff=FALSE
 1090 UNTIL alarmflag%<2
 1100 IF val%(15)=0 THEN *FX6,10
 1110 IF val%(15)=1 THEN *FX6,0
 1120 :
 1130 VDU2
 1140 REM Printer reset
 1150 VDU1,27,1,64
 1160 REM Printer enable
 1170 VDU1,17
 1180 REM Set NLQ state
 1190 VDU1,27,1,120,1,val%(1)
 1200 VDU1,27,1,112,1,val%(2)
 1210 VDU1,val%(3)
 1220 VDU1,27,1,87,1,val%(4)
 1230 VDU1,27,1,val%(5)
 1240 VDU1,27,1,val%(6)
 1250 VDU1,27,1,84-val%(7),1,0
 1260 VDU1,27,1,65,1,val%(11)
 1270 VDU1,27,1,67,1,val%(12)
 1280 VDU1,27,1,78,1,val%(13)
 1290 VDU1,27,1,val%(8)
 1300 VDU1,27,1,108,1,val%(9)
 1310 VDU1,27,1,81,1,val%(10)
 1320 VDU1,27,1,85,1,val%(14)
 1330 IF state$(16)=gn$+"ON" THEN PROCte
stpiece:VDU3:ENDPROC
 1340 VDU3
 1350 :
 1360 PRINTTAB(0,20)SPC(39);TAB(0,21)SPC
(39)
 1370 PRINTTAB(0,20)"PRINTER  MODES  SEL
ECTED"
 1380 PROCdel(15000)
 1390 ENDPROC
 1400 :
 1410 DEF PROCchecka(c%)
 1420 IF ok%(c%)>0 VDU7:ENDPROC
 1430 IF state$(c%)="OFF" OR state$(c%)=
"  PICA" state$(c%)=gn$+"ON" ELSE state$
(c%)="OFF"
 1440 IF state$(c%)="OFF" a$(c%)=wh$:val
%(c%)=val1%(c%):PROCdisable(oka$(c%)) EL
SE a$(c%)=gn$:val%(c%)=val2%(c%):PROCdis
able(okb$(c%))
 1450 IF c%=1 PROCchecka1
 1460 IF c%=8 PROCchecka8
 1470 PRINTTAB(36+3*(c%=8),3+c%)state$(c
%);TAB(2-8*(c%=8),3+c%)a$(c%)
 1480 ENDPROC
 1490 :
 1500 DEF PROCchecka1:LOCAL c1$,c2$
 1510 IF state$(1)="OFF" c1$=wh$:c2$=sta
te$(8) ELSE c1$=rd$:c2$=STRING$(6," ")
 1520 PRINTTAB(2,11)c1$TAB(33,11)c2$
 1530 ENDPROC
 1540 :
 1550 DEF PROCchecka8
 1560 IF state$(8)="OFF" state$(8)="  PI
CA" ELSE state$(8)=gn$+"ELITE"
 1570 ENDPROC
 1580 :
 1590 DEF PROCcheckb(c%)
 1600 REPEAT:PRINTTAB(31,3+c%)SPC3;f1$+c
y$
 1610 INPUTTAB(31,3+c%)val%(c%):IF val%(
c%)=0 val%(c%)=val3%(c%)
 1620 UNTIL val%(c%)>=val1%(c%) AND val%
(c%)<=val2%(c%)
 1630 IF val%(c%)=val3%(c%) j$=wh$:k$=bl
$ ELSE j$=gn$:k$=gn$
 1640 PRINTTAB(2,3+c%)j$TAB(34,3+c%)nf$+
k$TAB(31,3+c%)"    "TAB(36,3+c%)val%(c%)
 1650 ENDPROC
 1660 :
 1670 :
 1680 DEF PROCcheckc(c%)
 1690 IF state$(c%)="OFF" state$(c%)=gn$
+"ON":j$=gn$:val%(c%)=val2%(c%) ELSE sta
te$(c%)="OFF":j$=wh$:val%(c%)=val1%(c%)
 1700 PRINTTAB(36,3+c%)state$(c%)TAB(2,3
+c%)j$
 1710 ENDPROC
 1720 :
 1730 DEF PROCcheckprinter
 1740 printeroff=TRUE
 1750 VDU2,1,0,1,0,3
```

```
 1760 PROCdel(1500)
 1770 IF ADVAL(-4)=63 printeroff=FALSE:P
RINTTAB(9,23)"Choose option (A to Q)":EN
DPROC
 1780 PRINTTAB(0,23)SPC(11)rd$"PRINTER O
FF LINE"SPC(11)
 1790 IF alarmflag%=1 PROCalarm:PROCdel(
2000)
 1800 ENDPROC
 1810 :
 1820 DEF PROCalarm
 1830 ENVELOPE6,1,70,16,2,2,0,0,126,0,0,
-85,85,85
 1840 SOUND1,6,100,30
 1850 ENDPROC
 1860 :
 1870 DEF PROCdel(del%)
 1880 FOR I%=1 TO del%:NEXT I%
 1890 ENDPROC
 1900 :
 1910 DEF PROCinit:LOCAL c%
 1920 VDU23,1,0;0;0;0;
 1930 @%=&00000303
 1940 rd$=CHR$129:gn$=CHR$130:bl$=CHR$13
2:cy$=CHR$134:wh$=CHR$135:db$=CHR$141
 1950 fl$=CHR$136:nf$=CHR$137
 1960 REM Set default option variables
 1970 DIM state$(16),val%(16),val1%(16),
val2%(16),val3%(16),text$(16)
 1980 DIM ok%(8),oka$(8),okb$(8),a$(8)
 1990 FOR c%=1 TO 8
 2000 READ oka$(c%),okb$(c%):a$(c%)=wh$
 2010 NEXT
 2020 DATA 11010000,11212222,11011110,11
211112
 2030 DATA 00110110,22112112,11111111,11
111111
 2040 DATA 01011110,21211112,01111101,21
111121
 2050 DATA 01111011,21111211,00010111,22
212111
 2060 :
 2070 FOR c%=1 TO 16
 2080 READ val3%(c%),val1%(c%),val2%(c%)
 2090 state$(c%)="OFF":val%(c%)=val3%(c%
)
 2100 NEXT:state$(8)="  PICA"
 2110 DATA 0,0,1,0,0,1,18,18,15,0,0,1
 2120 DATA 70,70,69,72,72,71,0,0,1
 2130 DATA 80,80,77,0,0,133,80,2,135
 2140 DATA 12,1,72,66,1,127,0,0,127
 2150 DATA 0,0,0,0,0,0,0,0,0
 2160 FOR c%=9 TO 13:c$=STR$(val%(c%)):s
tate$(c%)=STRING$(3-LEN(c$)," ")+c$:NEXT
 2170 FOR c%=1 TO 16:READ text$(c%):NEXT
 2180 DATA "NLQ mode"
 2190 DATA "Proportional mode"
 2200 DATA "Condensed mode"
 2210 DATA "Enlarged mode"
 2220 DATA "Emphasized mode"
 2230 DATA "Double strike"
 2240 DATA "SuperScript"
 2250 DATA "Pica or Elite"
 2260 DATA "Left margin (0 to 133)"
 2270 DATA "Right margin (2 to 135)"
 2280 DATA "Line spacing (1 to 72 72nds)
"
 2290 DATA "Lines per page (1 to 127)"
 2300 DATA "Perforation skip (max 127)"
 2310 DATA "Unidirectional print"
 2320 DATA "System auto line feed"
 2330 DATA "Print test piece"
 2340 ENDPROC
 2350 :
 2360 DEF PROCmenu:LOCAL c%
 2370 PRINTTAB(1,0)cy$db$;"ADVANCED PRIN
TER MODE SELECTOR"
 2380 PRINTTAB(1,1)cy$db$;"ADVANCED PRIN
TER MODE SELECTOR"
 2390 REM:PRINTTAB(0,19)SPC40;TAB(0,24)S
PC40
 2400 FOR c%=1 TO 16
 2410 PRINTTAB(0,3+c%)cy$;CHR$(64+c%);wh
$;text$(c%)TAB(35+3*(c%=8));bl$;state$(c
%)
 2420 NEXT
 2430 PRINTTAB(0,20)cy$"Q"wh$"Displayed
options"
 2440 PRINTTAB(9,23)"Choose option (A to
 Q)"
 2450 ENDPROC
 2460 :
 2470 DEF PROCtestpiece:LOCAL x%
 2480 VDU2,1,27,1,54
 2490 FOR x%=32 TO 159:VDU1,x%;:NEXTx%
 2500 VDU1,254,1,255,1,13
 2510 ENDPROC
 2520 :
 2530 DEF PROCdisable(z$):LOCAL c%
 2540 FOR c%=1 TO 8
 2550 ok%(c%)=ok%(c%)+VAL(MID$(z$,c%,1))
-1
 2560 IF ok%(c%)>0 c$=rd$ ELSE c$=a$(c%)
 2570 PRINTTAB(2-8*(c%=8),3+c%)c$
 2580 NEXT
 2590 ENDPROC
```

Practical Assembler (Part 1)

Bernard Hill introduces the first of a new series of articles on programming in 6502 assembler.

Welcome to a new series on assembler language programming. As the title suggests, the emphasis throughout will be on the practical use of assembler, and will cover topics such as the handling of the interfaces with the operating system and filing system. The intention is that by the end of the series we shall have enough background to begin a larger project with some confidence, so on the way we shall be discussing more general aspects of program design, coding and testing which are as applicable to Basic or any other language as they are to Assembler. However, this is not intended as an introduction to assembler for complete beginners to this form of programming. For this the reader is referred to the 11-part series *Exploring Assembler by Lee Calcraft* which appeared from Vol.6 No.2 to Vol.7 No.2 - see the end of his article for a special price on these back numbers.

But before we get down to any significant detail I will attempt to answer a much more fundamental question.

WHY USE ASSEMBLER?

There's one very bad reason, but I've come across it a couple of times:

1. To make the program smaller.

On other computers it is usually true that programs are shorter when written in assembler but on the BBC the reverse is true. In fact Basic (or any interpreted language) gives you a much shorter program than assembler alone. The reason is quite simple: when running Basic you are making intensive use of the 16K Basic ROM, so your own program is much smaller as a result.

For instance, consider the addition of two four-byte integers:

```
CLC: LDX #0
 loop LDA &404,X:ADC &408,X:STA &40C,X
INX: CPX #4: BNE loop
```

This results in 17 bytes of assembled code in all (and that does not even include the assembler source at all). In Basic the equivalent is C%=A%+B%, 8 bytes in all, less than half size. So don't use assembler because your programs are too big.

The other reasons for using assembler are all valid on the BBC micro:

2. I need my program to go faster.

Of course. It is well known that assembler is many times faster than Basic, but how much faster depends on what you are doing. If you're writing a fast game then assembler may well be a must, but when waiting for input via a slow serial line an assembler program will give little improvement. And bear in mind also the possibility of using assembler inserts in a Basic program just to speed up the bits which are used intensively.

I can cite a personal example of this: some months ago I started to fulfil a long-held ambition: to write a chess program. I wrote it in Basic as the debugging is so much simpler, and I could use Basic's recursive procedures to search the move tree easily. But obviously it wasn't going to play decent chess written in Basic, so a careful examination of the code revealed two routines which were used very frequently. Consequently I re-coded the *Move Generation* and *Position Evaluation routines in assembler*, calling them from my Basic program. I'm still not sure it plays 'decent' chess, but it obviously goes faster and makes reasonable moves in a finite time! If this were a serious project then the gradual replacement of sections of code from Basic to assembler makes a very reasonable development route.

3. I want to do unusual things to the computer!

Maybe. Certain hardware features require assembler's speed to make them work fast enough, like the VIA and timers. But if you're that far into the machine then you're probably not reading this article anyway as assembler should be your natural medium.

4. I want my program to be callable without Basic.

This requirement is very common. Whether you are looking for a short routine to set your printer to bold characters (see later) or you are writing a text-finding routine (such as *FIND published in BEEBUG Vol.8 No.7) you are likely to want to call it from a word processor or database. You may even want to call it from Basic without disturbing a current Basic program (as in *FINDD, Vol.8 No.10). But there is a perennial problem with the BBC micro of where to run this code. It has to be loaded in somewhere and the memory map is very crowded. Let's look at this problem in detail.

WHERE TO PUT IT AND HOW TO CALL IT

In order to assess the various places in which you could put your code we will consider a concrete example. It's a utility which switches the printer into emphasised mode. Check with your printer manual, but on an Epson-compatible you need to send the following sequence of bytes to the printer:

```
27 69
```

This routine can, of course, be used to implement any required function on the printer - just change the codes you send and the name of the program.

This task is easy, but just to make it completely tidy we will ensure that we leave the printer in the same enabled (Ctrl-B) or disabled (Ctrl-C) state as it was before we started sending bytes to it. This status is held in bit 0 of location &D0, so in Basic our routine would be:

```
10 s=?&D0 AND 1
20 VDU 2,1,27,1,69
30 IF s=0 THEN VDU3
```

Note: the correct way to obtain the value of the printer status byte at &D0 is to use OSBYTE &75 for second-processor compatibility: but more on this next time. The same routine written in assembler appears in lines 150 to 240 of Listing 1.

So where do we put it in memory? On the Master, 512 (&200) bytes are available at address &DD00 for machine code utilities, typically loaded with a star command from disc. This is a very useful area, but does not exist on a model B or B+. With these systems we have to be more careful about the location of machine code routines. Here is an analysis of potential memory locations which we might consider:

Listing 1

```
  10 REM PROGRAM 1 Practical Assembler
  20 REM Version B1.0
  30 REM Author Bernard Hill
  40 REM BEEBUG May 1990
  50 REM Program subject to copyright
  60 :
 100 R%=&900:f$="BOLD"
 110 HIMEM=&7000:REM code placed here
 120 FOR opt=4 TO 7 STEP 3
 130 P%=R%:O%=HIMEM
 140 [OPT opt
 150 LDA &D0:AND #1:PHP
 160 LDA #2:JSR &FFEE:LDX #0
 170 .loop LDA #1:JSR &FFEE
 180 LDA bytes,X:JSR &FFEE
 190 INX:CPX #2 \ no. of bytes to send
 200 BNE loop
 210 PLP:BNE miss
 220 LDA #3:JSR &FFE3
 230 .miss RTS
 240 .bytes EQUB 27:EQUB 69
 250 ]:NEXT
 260 c$="SAVE "+f$+" "+STR$~HIMEM+" "+S
TR$~O%+" "+STR$~R%+" "+STR$~R%
 270 PRINT"*"+c$:OSCLIc$
```

Page(s)	Model B usage
0-3	Used by the OS
4-7	Current language
8	Sound buffer
9	Tape/RS423 output buffer
A	Tape/RS423 input buffer
B	Function keys
C	Expanded characters
D	Used by the DFS

Don't be tempted to use part of pages 6 and 7 - which are sometimes available in Basic - because other languages will not leave this space free. The same applies to the zero page locations &70-&8F. On a model B with disc, the best place to put machine code utilities is pages 9 and A, or page D for the cassette user. Either can expand into pages B and C but will result in the corruption of the function key and extended character set buffers if used.

If you're calling the machine code from Basic, then the infallible method is to reserve memory using the DIM statement and use that, but this is limited to assembler inserts in a Basic program.

So those are our choices on where to locate the assembled machine code, but how shall we call it up?

1. If we're calling from Basic then use CALL or USR. There'll be more on this topic in later articles.

2. We can effectively add it into the operating system.

Two commands are given by Acorn for this purpose: *CODE and *LINE. They use the user vector in &200-201 which must be set to point to the beginning of the routine, and the only difference between them is in the parameters they accept. *CODE accepts up to two decimal parameters which will be loaded into the X and Y registers before the routine starts, whereas *LINE accepts a whole string whose start address is contained in X and Y when the routine is entered. When you issue a *CODE or *LINE command, the computer executes the routine located in &200-201 with A=0 or A=1 respectively.

Unfortunately, pressing Break destroys the contents of &200-201 and we have to reload. It's quite hard to find good reasons to use *LINE or *CODE, but one excellent exception is Andrew Rowland's recent Keyword Highlighter (see BEEBUG Vol.8 No.9) which uses *LINE to enhance LIST. I refer you to that article as an example of its application.

3. We can *LOAD into &900 and use *GO 900 (Master and Compact users only).

4. Assemble into a disc image and use *filename to execute it (or */filename on cassette).

These last two methods are both available if we can arrange our assembler program to produce the disc image with load address &900 (Listing 1). The variables *locn* and *name$* set at the beginning define the name of the utility on the disc and the location at which it will load and run. Examine this program closely, and make sure you know exactly what's going on.

5. We can install it in sideways RAM (or blow an EPROM to make it permanent).

When you issue a star command to the Beeb's operating system, a sequence of events takes place. First the command is checked against the Beeb's own set (such as *KEY) and executed if found. If it is not recognised then each sideways ROM is given a chance to recognise it (such as *TYPE which the DFS ROM understands). Failing this, the command will be loaded from disc and *RUN if a file of the same name exists in the default directory or library.

While it's easy to provide a disc file of a given name, it's a relatively long job getting a sideways ROM image set up and a given star command recognised. Each ROM has a 'Service Entry' at address &8003: this means that the O.S. will enter it with a JSR &8003 so that address &8003 must contain a JMP to the address where the real routine starts. Since sideways ROMs can do all sorts of other things too, Acorn decreed that a service command (such as ours is) will have A=4 on entry and &F2, &F3 and the Y register will point to the actual characters of the command name which is to be checked (so that LDA (&F2),Y loads the first character of the command string). All the bother of checking for abbreviations (such as *DI.) or lower case (*disc) is left to the writer of the ROM.

All this is very tedious, but of course someone has already worked out how it can (fairly) easily be done. Listing 2 produces a ROM header, a table of commands in the ROM (you will want to have more than one star command accessed in one ROM), and a place for a set of assembler routines to go with each command. Look carefully at line 440 (which calls the routine identical to listing 1), noting the line 450 which re-establishes the registers before departing from the ROM. Lines 470-550 are exactly as in Listing 1, and note the very similar section of code at lines 570-600 which at the moment does nothing - it is awaiting your own routines. Other routines can be added in a similar fashion by extending the table at lines 620-630 and using the models given above. Why not start with something very simple such as writing an 'A' to the screen. The ROM also really needs a *PLAIN command to turn off the *BOLD, so why not

have a go? If you want to know more about sideways RAM and ROM there have been a number of articles in BEEBUG which deal with the structure of these in detail - see Vol.5 Nos.2-4 and 10, and Vol.6 No.1.

Listing 2

```
  10 REM Sideways RAM Header
  20 REM Version B1.0
  30 REM Author Bernard Hill
  40 REM BEEBUG May 1990
  50 REM Program subject to copyright
  60 :
 100 title$="Printer Rom"
 110 owner$="Beebug, 1990"
 120 MODE7:HIMEM=&7000
 130 FOR opt=4 TO 6 STEP 2
 140 P%=&8000:O%=HIMEM:Q%=O%
 150 [ OPT opt
 160 BRK:BRK:BRK
 170 JMP service
 180 EQUB &82
 190 EQUB copyright
 200 EQUB 1
 210 EQUS title$
 220 .copyright
 230 EQUB 0
 240 EQUS "(c) "+owner$
 250 EQUB 0
 260 .service
 270 CMP #4:BEQ service4:RTS
 280 .service4
 290 TYA : PHA : LDX #0
 300 .ncmd PLA:TAY:PHA:DEX:DEY
 310 .nch INX:INY:LDA(&F2),Y:CMP #&61
 320 BCC noc:CMP #&7B:BCS noc:AND #&DF
 330 .noc EOR cmds,X:BEQ nch:BPL on
 340 LDA (&F2),Y: BEQ do
 350 CMP #13 : BEQ do:CMP #32 : BEQ do
 360 .on CPX #0:BEQ x:LDA (&F2),Y
 370 CMP #&2E:BEQ skip
 380 .x LDA cmds,X:BMI ov:INX:JMP x
 390 .ov INX:INX:LDA cmds,X:BNE ncmd
 400 PLA:TAY:LDX &F4:LDA #4:RTS
 410 .skip INX:LDA cmds,X:BPL skip
 420 .do LDA cmds,X:PHA
 430 LDA cmds+1,X:PHA:RTS
 440 .command1 JSR bold
 450 PLA : TAY : LDX &F4 : LDA #0 :RTS
 460 \ *BOLD ROUTINE GOES HERE -----\
 470 .bold LDA &D0:AND #1:PHP
 480 LDA #2:JSR &FFEE:LDX #0
 490 .loop LDA #1:JSR &FFEE
 500 LDA bytes,X:JSR &FFEE
 510 INX:CPX #2 \ no. of bytes to send
 520 BNE loop:PLP:BNE miss
 530 LDA #3:JSR &FFEE
 540 .miss RTS
 550 .bytes EQUB 27:EQUB 69
 560 \ TO HERE --------------------/
 570 .command2 JSR ital
 580 PLA : TAY : LDX &F4 : LDA #0 :RTS
 590 .ital \ insert your own routine he
re to make italic characters
 600 RTS
 610 .cmds
 620 EQUS "BOLD":EQUW FNdec(command1)
 630 EQUS "ITAL":EQUW FNdec(command2)
 640 EQUB 0
 650 ]:NEXT
 660 c$="SAVE ROMIMAG "+STR$~Q%+" "+STR
$~O%+" 8000 8000"
 670 PRINT"*"+c$:OSCLIc$
 680 END
 690 :
1000 DEFFNdec(a)=256*((a-1) MOD 256)+(a
-1) DIV 256
```

Besides complexity, there are a number of other considerations to be made when using sideways ROM images:

1. It's slightly harder to perform fatal error handling (we'll look at this in the next article).

2. Unless you're content to use sideways RAM all the time and never blow a 'hard copy' of your image into EPROM then you have to arrange some normal RAM for the saving of any variables you may have.

But in return for these difficulties you get a massive 16K of space for your routines, and no problems about impinging on anyone else's.

That concludes this first article of our series. Next time we shall take a closer look at error handling and the interface with the operating system. I shall also be starting a *Hints and Tips* section to accompany this article.

B

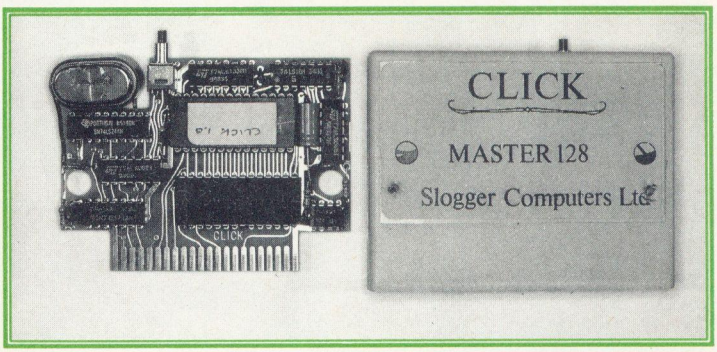
Slogger's Smart Cartridge for the Master

With the press of a button, Slogger's CLICK offers a host of DFS and ADFS utilities, a memorandum system, and more for users of the BBC Master 128. Johnathan A. Wilkinson 'presses' it into action.

Product	**CLICK**
Supplier	**Slogger Computers Slogger Business Centre, Sancreed, Newbridge, Penzance, Cornwall TR20 8QP. Tel. (0736) 810920**
Price	**£59.95 inc VAT**

Following some nine months of silence, Slogger have reappeared after their office move, and brought out CLICK. Supplied in a standard issue Acorn ROM cartridge, containing 32Kbytes of ROM, 8Kbytes of static RAM (with necessary battery-backup), and a switch on the top to activate it, CLICK offers a host of utilities.

When pressing the little black button on the top of the cartridge, teletext mode 7 is entered to present a menu screen. Once the appropriate facility has been used, the computer is restored to exactly what it was doing before (cf PMS's Genie).



The CLICK cartridge for Slogger

The following facilities are offered: Memo, ADFS, DFS, Memory Editor, and System. The required option and its sub-menus, if any, is selected by using the cursor keys to highlight the one desired, with Return to confirm.

The Memo facility is probably the most desirable facility offered. It enables various appointments to be entered, along with their date and time. At the appropriate time, CLICK activates itself and displays the message. This leaves very little excuse for missing important meetings, or events, etc. (assuming one does not forget to enter the data in the first place).

A series of utilities for use with the Advanced Disc Filing System (ADFS) is provided, namely: *Format, Verify, Sector editor,* and *Extree.* The first three need no explanation; the last, however, requires some discussion. It provides a front-end system for manoeuvring around the ADFS's hierarchical structure: the top of the screen displays, in a scrolling window format, the position of the current directory in relation to others; the centre shows the full pathname of the current directory; and the lower section the files in it. It is possible to move around the directories, and to change the attributes of any files. One slight problem encountered was that if the path name of the current directory exceeds about thirty-eight characters, the display becomes corrupt. DOS users may note the similarity of the Extree facility with the front-end systems provided by Directory Assistance, PC-Tools, etc.

A similar set of utilities is also provided for use with the Disc Filing System (DFS), save for Extree.

The Memory editor is a run-of-the-mill memory examiner, displaying the contents in both hexadecimal and ASCII formats. It is rather slow at scrolling through memory.

The final set of utilities is provided under the System menu and includes: mouse handler, CMOS clock-setting, and printer dump. The last one enables various parameters to be set, so that screens can be printed out by pressing shift in combination with the little black button.

Although CLICK offers a reasonable set of utilities, it doesn't really offer anything to shout loudly about. The use of teletext mode 7 is quite a limitation to the screen formats, which are rather plain. If one is after a set of utilities for use with DFS and ADFS, and an appointment system, a combination of ACP's Advanced Disc Toolkit (ADT) and PMS's Genie would be better suited. In all, a nice product, had it been offered a year or two ago, but already looking outdated at its launch.

B

Designer Shoot-'Em-Up (Part 1)

Design your own arcade games with these programs from Al Harwood.
Text by Alan Wrigley.

It's nice sometimes to zap the odd alien, without having to fork out for the latest arcade game. Well, now you can create your own shoot-'em-up games with this utility. It allows you to to design a number of sprites, representing you (the good guy) and the baddies, and then specify the movements your enemies will make on the screen. The game thus created will run in mode 5, giving you four colours on the screen.

Using the sprite definer program listed in this month's magazine

Some of the listings are quite long, so the article will be published in two parts. This month we will describe the sprite designer, and list the source code for the machine code program which handles all the game routines. Next month's magazine will contain the loader program for your game, and details of how to create a specification file for enemy movements, including an example which you can type in to try the program out.

SPRITE DESIGNER

Type in listing 1, save it, and then run the program. The sprites you define will be held in a file on the disc, so first of all you will be asked if you wish to start a new file or edit an existing one. Next you will be asked for the sprite number. You can define up to 22 sprites, with numbers ranging from 0-21. Sprite 0 is always the player (i.e. the good guy), sprite 1 is the missile, and the remaining sprites are the enemy, as few or as many (up to 20) as you wish. Entering -1 at this prompt will quit the editor and close the file. It is also perfectly possible to choose an existing sprite number, edit it, and then save it with a different number afterwards, as described below.

Having chosen which sprite to edit, you will now see on the screen a window 16x16 characters in size. Each character position represents a 2x2 pixel block in the completed sprite. A cursor is shown, which can be moved around the window. At the top right of the screen is displayed the current colour, and pressing the space bar puts a block of that colour on the screen at the cursor position. Pressing Return changes the current colour. The sprite is shown at the top left of the screen at its correct size, so you can see how it looks as you design it. When you are satisfied, press Escape, and then enter the sprite number with which it is to be saved. This will normally be the same number you specified at the start, unless you are producing a modified version of an existing sprite which you want to save separately.

GAME ROUTINES

The machine code routines which will control the game must be assembled using Listing 2. Type this in carefully and save it. When run, it will produce a machine code program called GAME, which must then be loaded into memory before your own game is run. Full details on how to do this will be given next month. In the meantime, you can be using the sprite definer to design some colourful sprites ready to use with your game next month.

Listing 1

```
 10 REM          >Sprite
 20 REM Program Sprite Definer
 30 REM Version B1.0
 40 REM Author  Al Harwood
 50 REM BEEBUG  May 1990
 60 REM Program subject to copyright
 70 :
100 MODE5:VDU23;8202;0;0;0;
```

```
  110 PROCinit:PROCload
  120 REPEAT PROCsprs:UNTIL S%=-1
  130 PROCsave:END
  140 :
 1000 DEF PROCinit
 1010 VDU23,128,0,126,126,126,126,126,12
6,0
 1020 VDU23,129,0,0,0,0,255,0,255,0
 1030 VDU23,130,10,10,10,10,10,10,10,10
 1040 VDU23,131,80,80,80,80,80,80,80,80
 1050 VDU23,132,0,255,0,255,0,0,0,0
 1060 DIM data &600
 1070 *FX4,1
 1080 *FX229,1
 1090 FOR A%=data TO data+&5FF STEP4
 1100 !A%=0:NEXT:ENDPROC
 1110 :
 1120 DEF PROCsprs
 1130 CLS:PRINTTAB(0,10)"WHICH SPRITE? (
0-21)"'''"Enter -1 to exit"
 1140 REPEAT INPUTTAB(0,16)""S%
 1150 UNTIL S%>-2 AND S%<22
 1160 IF S%=-1 ENDPROC
 1170 FOR A=0 TO 3:VDU19,A,0;0;:NEXT
 1180 CLS:COLOUR 1:PRINTTAB(11,0)"SPRITE
:"STRING$(2-LENSTR$S%,"0");S%
 1190 PRINTTAB(1,28)"Use arrows, Space,"
TAB(1,31)"Return, and Escape";
 1200 A%=data+64*S%
 1210 FOR X%=0 TO 3:FOR Y%=0 TO 15
 1220 ?(HIMEM+(Y%DIV8)*320+Y%MOD8+X%*8)=
?(A%+Y%+X%*16):NEXT,
 1230 VDU31,0,5:FOR Y%=1 TO 64 STEP4
 1240 VDU10,13,9,9:FOR X%=0 TO 120 STEP8
 1250 COLOUR POINT(X%,1020-Y%)
 1260 VDU128:NEXT,
 1270 COLOUR 3:PRINTTAB(2,5)STRING$(16,C
HR$129)TAB(2,22)STRING$(16,CHR$132)
 1280 FOR A=6 TO 21
 1290 VDU31,1,A,130,31,18,A,131:NEXT
 1300 FOR A=0 TO 3:VDU19,A,A;0;:NEXT
 1310 X=0:Y=1:P=0:PROCpen
 1320 REPEAT VDU17,129,17,POINT(X*8,1020
-Y*4),31,X+2,Y+5,128,17,128
 1330 G=GET:IF G=136 AND X>0 PROCcur(-1,
0)
 1340 IF G=137 AND X<15 PROCcur(1,0)
 1350 IF G=138 AND Y<16 PROCcur(0,1)
 1360 IF G=139 AND Y>1 PROCcur(0,-1)
 1370 IF G=13 PROCpen
 1380 IF G=32 GCOL 0,P:PLOT 69,X*8,1020-
Y*4
 1390 UNTIL G=27
 1400 COLOUR 3:PRINTTAB(0,24)"STORE AS W
HICH"'"SPRITE? (0-21)"
 1410 REPEAT INPUTTAB(0,26)""S%
 1420 UNTIL S%>-1 AND S%<22
 1430 A%=data+64*S%
 1440 FOR X%=0 TO 3:FOR Y%=0 TO 15
 1450 ?(A%+Y%+X%*16)=?(HIMEM+(Y%DIV8)*32
0+Y%MOD8+X%*8):NEXT,
 1460 ENDPROC
 1470 :
 1480 DEF PROCcur(x,y)
 1490 COLOUR POINT(X*8,1020-Y*4)
 1500 VDU31,X+2,Y+5,128:X=X+x:Y=Y+y
 1510 ENDPROC
 1520 :
 1530 DEF PROCpen
 1540 P=P+1:IF P>3 P=0
 1550 COLOUR P:PRINTTAB(14,2)"PEN"
 1560 COLOUR 128+P
 1570 PRINTTAB(18,2)"  "TAB(11,2)"  "
 1580 COLOUR 128:ENDPROC
 1590 :
 1600 DEF PROCload
 1610 PRINT'''"Enter filename:"
 1620 INPUTfn$:IF fn$=""ENDPROC
 1630 OSCLI("L."+fn$+" "+STR$~data)
 1640 ENDPROC
 1650 :
 1660 DEF PROCsave
 1670 PRINT'''"Enter filename:":INPUTfn$
 1680 OSCLI("S."+fn$+" "+STR$~data+"+600
 0 4000"):ENDPROC
```

Listing 2

```
   10 REM          >Code
   20 REM Program Game Routine Source
   30 REM Version B1.0
   40 REM Author  Al Harwood
   50 REM BEEBUG  May 1990
   60 REM Program subject to copyright
   70 :
  100 IF PAGE>&1200 PRINT'''"Type PAGE=&1
200 before loading"'':END
  110 code=TOP+&F00:start=&1200
  120 topscr=&3000:scrv=&3001
  130 buffer=&E00:scrhb=&58
  140 bulletX=&F00:bulletY=&1000
  150 bombX=&900:bombY=&A00
  160 osrdch=&FFE0:osasci=&FFE3
  170 osnewl=&FFE7:oswrch=&FFEE
  180 osword=&FFF1:osbyte=&FFF4
  190 oscli=&FFF7
  200 FORpass=4TO6STEP2
  210 P%=&70:O%=code
  220 [OPTpass
  230 .temp NOP:NOP:NOP
  240 .addr NOP:NOP
  250 .sprv NOP:NOP
  260 .store NOP:NOP
  270 .xlen NOP
  280 .ylen NOP
  290 .x1 NOP
```

```
 300 .x2 NOP
 310 .y1 NOP
 320 .y2 NOP
 330 .s1 NOP
 340 .s2 NOP
 350 .v NOP:NOP
 360 .x NOP
 370 .y NOP
 380 .endgame NOP
 390 .loselife NOP
 400 .screen NOP
 410 .bullet NOP
 420 .bomb NOP
 430 .xpos NOP
 440 .completed NOP
 450 .retnt NOP
 460 ]P%=start:O%=code
 470 [OPTpass
 480 .game LDA#0:JSRsetup
 490 .gameloop JSRplamv:JSRcommv
 500 JSRmvbul:JSRmvbom:JSRchhit
 510 JSRchend:JSRendli:JSRcompl
 520 LDAendgame:BEQgameloop
 530 LDAscore:STA&70:LDAscore+1
 540 STA&71:LDAscore+2:STA&72
 550 LDAscore+3:STA&73
 560 LDAscore+4:STA&74:RTS
 570 .collision:INCx1:JSRcoll:DECx1:RTS
 580 .coll \s1/2 x1/2 y1/2 setup?
 590 \ on exit if C=1 sprites collided
 600 \ x1 must equal x1+1
 610 LDYs2:LDXs1
 620 LDAx2:CMPx1:BCCjump3
 630 LDAx1:LDYs1:CLC:ADC#4
 640 LDYs2:SEC:SBC#2:CMPx2:BCSjump3a
 650 JMPjump4:.jump3
 660 ADC#4:CMPx1:BCCjump4:.jump3a
 670 LDAy2:CMPy1:BCCjump3b
 680 LDAy1:LDYs1:CLC:ADC#2
 690 LDYs2:SEC:SBC#1:CMPy2:BCCjump4
 700 LDXs1:LDYs2:SEC:RTS
 710 .jump3b ADC#2:SEC:SBC#1
 720 CMPy1:BCCjump4
 730 LDXs1:LDYs2:SEC:RTS:.jump4
 740 CLC:RTS
 750 .sprite PHA
 760 LDA#0:STAstore
 770 TXA:ASLA:ROLstore:ASLA:ROLstore
 780 ASLA:ROLstore:CLC:ADC#0
 790 CLC:ADCxtab320,Y:STAaddr
 800 LDAstore:ADC#scrhb:ADCytab320,Y
 810 STAaddr+1:PLA:TAY
 820 LDA#4:STAxlen
 830 LDA#2:STAylen
 840 LDAxspr,Y:STAsprv
 850 LDAyspr,Y:STAsprv+1
 860 .ee LDAxlen:STAtemp+2
 870 LDAaddr:AND#&F8:STAtemp:STAaddr
 880 LDAaddr+1:STAtemp+1
 890 .Sloop1 LDXylen:.Sloop2 LDY#0
 900 .Sloop3 LDA(sprv),Y
 910 EOR(addr),Y:STA(addr),Y
 920 INY:CPY#8:BNESloop3
 930 CLC:LDAaddr:ADC#&40:STAaddr
 940 LDAaddr+1:ADC#1:STAaddr+1
 950 CLC:LDAsprv:ADC#8:STAsprv
 960 LDAsprv+1:ADC#0:STAsprv+1
 970 DEX:BNESloop2
 980 CLC:LDAtemp:ADC#8:STAtemp:STAaddr
 990 LDAtemp+1:ADC#0:STAtemp+1
1000 STAaddr+1:DECtemp+2
1010 BNESloop1:RTS
1020 .xtab320 OPT FNxtab320
1030 .ytab320 OPT FNytab320
1040 .xspr OPT FNxspr
1050 .yspr OPT FNyspr
1060 .endli LDAloselife:BNEendliJ:RTS
1070 .endliJ DECloselife
1080 JSRsnd2:LDAlives:CMP#ASC"0"
1090 BNEendliJ0:INCendgame:RTS:.endliJ0
1100 DEClives:DECscreen:JMPstartnew
1110 .compl LDAcompleted:BEQcomplJ
1120 INCendgame:.complJ RTS
1130 .setup TAX:DEX:STXscreen
1140 LDA#0:STAcompleted:STAendgame
1150 STAloselife:LDA#ASC"0"
1160 STAscore:STAscore+1:STAscore+2
1170 STAscore+3:STAscore+4
1180 LDA#ASC"3":STAlives
1190 LDA#200:LDX#3:LDY#0:JSRosbyte
1200 .startnew LDA#12:JSRoswrch
1210 LDA#0:STAbullet:STAbomb
1220 LDA#10:STAxpos:JSRstatus:
1230 JSRshowman:INCscreen:LDAscreen
1240 CMPtopscr:BCCsneJ:INCcompleted
1250 RTS:.sneJ LDA#15:LDX#0:LDY#0
1260 JSRosbyte:LDA#129:LDX#100:LDY#0
1270 JSRosbyte:LDAscreen:ASLA:TAY
1280 LDAscrv,Y:STAv
1290 INY:LDAscrv,Y:STAv+1
1300 LDY#0:LDA(v),Y:JSRchcol
1310 INY:LDA(v),Y:JSRchcol:INY
1320 LDA(v),Y:JSRchcol:INY
1330 LDX#0:.cpyL LDA(v),Y:INY
1340 STAbuffer,X:INX:CMP#255:BEQcpyJa
1350 LDA(v),Y:INY:STAbuffer,X:INX
1360 LDA(v),Y:INY:STAbuffer,X:INX
1370 LDA(v),Y:INY:STAbuffer,X:INX
1380 LDA(v),Y:INY:STAbuffer,X:INX
1390 JMPcpyL:.cpyJa
1400 LDY#0:.shoL LDAbuffer,Y:CMP#255
1410 BEQshoJ:PHA:INY:LDAbuffer,Y
1420 TAX:INY:LDAbuffer,Y:INY:INY:INY
1430 STYy:TAY:PLA:JSRsprite:LDYy
1440 JMPshoL:.shoJ RTS
1450 .chcol STAtemp:LDA#19:JSRoswrch
```

```
1460 TYA:CLC:ADC#1:JSRoswrch:LDAtemp
1470 JSRoswrch:LDA#0:JSRoswrch
1480 JSRoswrch:JMPoswrch
1490 .status LDAscreen:CLC:ADC#ASC"A"
1500 STAlevel:LDX#0:.staL LDAline,X
1510 JSRoswrch:INX:CPX#22:BNEstaL:RTS
1520 .line EQUB31:EQUB0:EQUB31
1530 EQUS" SC:":.score EQUS"00000 LI:"
1540 .lives EQUS"3 LE:":.level EQUS"A"
1550 .plamv OPT FNinkey(-98):BEQplaJ
1560 LDAxpos:BEQplaJ:LDA#0:JSRmoveman
1570 .plaJ OPT FNinkey(-67):BEQplaJ0
1580 LDAxpos:CMP#33:BEQplaJ0:LDA#1
1590 JSRmoveman:.plaJ0 OPT FNinkey(-74)
1600 BEQplaJ1_:JSRfire:.plaJ1
1610 OPT FNinkey(-113):BEQplaJ1_1
1620 INCendgame:.plaJ1_1 RTS:.plaJ1_
1630 LDA#0:STAretnt:JMPplaJ1
1640 .moveman PHA:JSRshowman:PLA
1650 BNEmanJ:DECxpos:JMPshowman
1660 .manJ INCxpos
1670 .showman LDA#0:LDXxpos
1680 LDY#26:JMPsprite
1690 .fire LDAretnt:BEQretntp:RTS
1700 .retntp INCretnt:JSRsnd1
1710 LDAbullet:CMP#255:BNEfireJ:RTS
1720 .fireJ INCbullet:LDYbullet
1730 LDAxpos:STAbulletX,Y:TAX:LDA#25
1740 STAbulletY,Y:TAY:LDA#1:JMPsprite
1750 .mvbul LDAbullet:BNEbulJ:RTS
1760 .bulJ LDX#0:.bulL INX:STXx
1770 LDAbulletY,X:TAY:LDAbulletX,X:TAX
1780 LDA#1:JSRsprite:LDXx:DECbulletY,X
1790 LDAbulletY,X:BEQdelbul:TAY
1800 LDAbulletX,X:TAX:LDA#1:JSRsprite
1810 .skip LDXx:CPXbullet:BCCbulL:RTS
1820 .delbul INX:LDAbulletY,X:TAY
1830 LDAbulletX,X:DEX:STAbulletX,X:TYA
1840 STAbulletY,X:INX:CPXbullet
1850 BCCdelbul:DECbullet:JMPskip
1860 .comJJ INX:INX:INX:INX:INX:JMPcomL
1870 .commv LDA#19:JSRosbyte
1880 LDX#0:.comL LDAbuffer,X:BEQcomJJ
1890 CMP#255:BEQcomJ:PHA:INX
1900 LDAbuffer,X:STAx:INX:LDAbuffer,X
1910 TAY:INX:STXy:LDXx:PLA:JSRsprite
1920 LDXy:JSRchspr:STXx:DEX:DEX:DEX
1930 DEX:DEX:LDAbuffer,X:BEQskip0:PHA
1940 INX:LDAbuffer,X:STAy:INX
1950 LDAbuffer,X:TAY:LDXy:PLA:JSRsprite
1960 .skip0 LDXx:JMPcomL:.comJ RTS
1970 .chspr DEX:LDAbuffer,X:STAy1:DEX
1980 LDAbuffer,X:STAx1:DEX:LDAbuffer,X
1990 STAs1:INX:INX:INX:LDAbuffer,X:INX
2000 STAv:LDAbuffer,X:INX:STAv+1:LDY#0
2010 LDA(v),Y:ASLA:BCCchsJ:LSRA:STAx
2020 DEX:DEX:LDAbuffer,X:SEC:SBCx
2030 STAbuffer,X:INX:LDAbuffer,X
2040 SBC#0:STAbuffer,X:INX:RTS
2050 .chsJ LSRA:CMP#1:BNEchsJ0:DECx1
2060 JMPchsJ1:.chsJ0 CMP#2:BNEchsJ0_
2070 INCx1:JMPchsJ1:.chsJ0_ CMP#3
2080 BNEchsJ0_0:DECy1:JMPchsJ1:.chsJ0_0
2090 CMP#4:BNEchsJ0_1:INCy1:JMPchsJ1
2100 .chsJ0_1 CMP#5:BNEchsJ0_2:LDA#0
2110 STAs1:JMPchsJ1:.chsJ0_2 CMP#6
2120 BNEchsJ0_3:LDAxpos:CMPx1:BCSchsJ7
2130 DECx1:JMPchsJ8:.chsJ7:INCx1:.chsJ8
2140 LDA#26:CMPy1:BCSchsJ9:DECy1
2150 JMPchsJ1:.chsJ9 INCy1:JMPchsJ1
2160 .chsJ0_3 CMP#7:BNEchsJ1:JSRdrbom
2170 .chsJ1 DEX:DEX:DEX:DEX:DEX:LDAs1
2180 STAbuffer,X:INX:LDAx1:STAbuffer,X
2190 INX:LDAy1:STAbuffer,X:INX
2200 LDAbuffer,X:CLC:ADC#1:STAbuffer,X
2210 INX:LDAbuffer,X:ADC#0:STAbuffer,X
2220 INX:RTS
2230 .drbom TXA:PHA:LDAbomb:CMP#255
2240 BNEbomJ:RTS:.bomJ INCbomb:LDYbomb
2250 LDAx1:STAbombX,Y:TAX:LDAy1
2260 STAbombY,Y:TAY:LDA#1:JSRsprite
2270 PLA:TAX:RTS
2280 .mvbom LDAbomb:BNEmbomJ:RTS
2290 .mbomJ LDX#0:.mbomL INX:STXx
2300 LDAbombY,X:TAY:LDAbombX,X:TAX
2310 LDA#1:JSRsprite:LDXx:INCbombY,X
2320 LDAbombY,X:CMP#29:BEQdelbom:TAY
2330 LDAbombX,X:TAX:LDA#1:JSRsprite
2340 .skip1 LDXx:CPXbomb:BCCmbomL:RTS
2350 .delbom INX:LDAbombY,X:TAY
2360 LDAbombX,X:DEX:STAbombX,X:TYA
2370 STAbombY,X:INX:CPXbomb:BCCdelbom
2380 DECbomb:JMPskip1
2390 .sc100 SEC:JMPsc100_:.sc10
2400 LDAscore+3:CLC:ADC#1:CMP#ASC"9"+1
2410 BCCscJ:SBC#10:SEC:.scJ STAscore+3
2420 .sc100_ LDAscore+2:ADC#0
2430 CMP#ASC"9"+1:BCCscJ0:SBC#10:SEC
2440 .scJ0 STAscore+2:LDAscore+1:ADC#0
2450 CMP#ASC"9"+1:BCCscJ1:SBC#10:SEC
2460 .scJ1 STAscore+1:LDAscore:ADC#0
2470 CMP#ASC"9"+1:BCCscJ2:SBC#10:SEC
2480 .scJ2 STAscore+1:JMPstatus
2490 .chhJJ INY:INY:INY:INY:INY:JMPchhL
2500 .chhit LDAbullet:BEQsk:LDY#0:.chhL
2510 LDAbuffer,Y:BEQchhJJ:CMP#255
2520 BEQchhJ:STAs2:INY:LDAbuffer,Y
2530 STAx2:INY:LDAbuffer,Y:STAy2:INY
2540 INY:INY:STYy:LDX#1:.chhL0
2550 LDAbulletX,X:STAx1:LDAbulletY,X
2560 STAy1:INX:STXx:LDA#1:STAs1
2570 JSRcollision:BCCchhJ0:JSRsc10
2580 LDAs2:LDXx2:LDYy2:JSRsprite:LDYy
2590 DEY:DEY:DEY:DEY:DEY:LDA#0
2600 STAbuffer,Y:JSRsnd2:.chhJ0 LDXx
```

Continued on page 48

Using Sideways RAM and ROM

Mike Williams presents some useful sideways RAM utilities written by Jonathan Ribbens.

Last month in First Course, I described the different types of memory that can be found on a BBC micro system, and I will now deal with the more practical uses of sideways RAM and ROM. On a model B, sideways ROM/RAM is largely an optional extra, but is a built-in feature of the Master 128 and Master Compact. As a result, the latter two machines come with star commands which provide some degree of control over the use of sideways ROM/RAM. The model B has no such facilities, unless they are provided as part of the sideways RAM/ROM board upgrade by the manufacturer. In this First Course, I intend to present four utilities (all written by contributor Jonathan Ribbens) which provide equivalent facilities for the model B, though they may be used on the Master and Compact just as easily.

LISTING SIDEWAYS ROM/RAM CONTENTS

To set the scene for all machines we will start by taking a look at the use of sideways RAM/ROM on a Master series machine. On these systems the command *ROMS will show the current state of the machine. Up to 16 ROM/RAM slots (or banks) are available numbered from 1 through to F (hexadecimal I'm afraid). The *ROMS command will show which slots are configured as ROM, and which as RAM, and the names of the ROMs or ROM images, if any, occupying these slots. For example, figure 1 shows the picture for the machine on which I am writing this article (a Compact).

When you first switch on a machine, all RAM banks will appear empty, as their contents are lost when a machine is switched off. The ROMs in the list show the physical ROM chips currently installed in your machine. ROM banks with no contents shown indicate empty ROM sockets in your computer. The Compact, for example, is fitted with 4 sideways RAM banks and 3 ROM sockets (in addition to the ROM containing the OS and ADFS supplied by Acorn which occupies a single socket physically, but is spread over more than one ROM bank. On the Master 128, with its cartridge system, the user can configure, by means of links on the circuit board, the proportions of sideways RAM and sideways ROM.

```
ROM F UTILS 00
ROM E BASIC 40
ROM D Acorn ADFS 00
ROM C ?
ROM B ?
ROM A ?
ROM 9 ?
ROM 8 ?
RAM 7 VIEW 05
RAM 6 DFS 92
RAM 5 ?
RAM 4 ?
ROM 3 WORDWISE-PLUS 03
ROM 2 SPELLMASTER 00
ROM 1 ?
ROM 0 ?
```

*Figure 1. Using *ROMS on a Compact*

Listing 1

```
 10 REM Sideways-ROM Lister
 20 REM Version B1.0
 30 REM Author  Jonathan Ribbens
 40 REM BEEBUG  May 1990
 50 REM Program subject to copyright
 60 :
100 FOR p=0 TO 2 STEP 2
110 P%=&900:[OPT p
120 :
130 .rlist
140 JSR message
150 JMP listroms
160 :
170 .message
180 JSRprint:EQUS"Sideways ROM Lister"
190 EQUB13:EQUS"By Jonathan Ribbens"
200 EQUB13:EQUB13
210 EQUS"Bank  Type  Name"
220 EQUB13:BRK:RTS
230 :
240 .print
```

```
250 PLA:STA&78:PLA:STA&79:LDY#0
260 .lp1:INY:LDA(&78),Y:JSR&FFE3
270 BNElp1:TYA:CLC:ADC&78
280 STA&78:LDA&79:ADC#0:PHA
290 LDA&78:PHA:RTS
300 :
310 .hexout PHA:LSRA:LSRA:LSRA:LSRA
320 JSRhex2:PLA:AND#&F:.hex2
330 CMP#10:BCShex3:CLC:ADC#48
340 JMP&FFEE:.hex3:CLC:ADC#55
350 JMP&FFEE
360 :
370 .spaces:LDA#32:JSR&FFEE
380 DEX:BNEspaces:RTS
390 :
400 .listroms
410 LDA&F4:PHA:LDA#15:STA&70:.lp2
420 LDA#32:JSR&FFEE
430 LDA&70:STA&F4:STA&FE30:JSRhexout
440 LDX#4:JSRspaces:LDA&8006:JSRhexout
450 LDX#3:JSRspaces:LDX&70:LDA&2A1,X
460 BNEromok:JSRprint
470 EQUS"<Empty>":BRK:JMPcont
480 .romok:LDX#9:.lp3
490 LDA&8000,X:JSR&FFEE:BEQcont
500 INX:CPX#39:BNElp3:.cont
510 JSR&FFE7:DEC&70:BPLlp2:PLA:STA&F4
520 STA&FE30:RTS
530 ]:NEXT
540 *SAVE RList 900+DF 900 900
550 PRINT"Utility RList saved."
560 END
```

The first of Jonathan Ribbens' utilities is a sideways ROM/RAM lister. This is given in Listing 1. Type the program in and then save it. When you run the program, your disc drive will operate and a message will be displayed telling you that a utility called *RList* has been saved to disc. To use the utility at any time, simply type:

```
*RList
```

and provided that the disc with this utility is accessible in a drive it will load and execute giving a list of the contents of your sideways RAM/ROM.

This utility, like the *ROMS command, will list the status of all 16 ROM/RAM banks even if they are not physically fitted to your machine - hopefully, if that is the case, they will be marked as 'empty'.

LOADING A SIDEWAYS ROM IMAGE

As I explained last time, a sideways ROM image is effectively a sideways ROM program which has been saved to disc (like several of the utilities which we have published in BEEBUG magazine). To use such a program it must be loaded into a sideways RAM bank, when it will appear ready for use like any sideways ROM.

The Master and Compact have a command to do this in the form of:

```
*SRLOAD <name> <address> <bank>
```

For example, the View word processor is supplied only as a ROM image on disc with the Compact. Once the appropriate disc directory has been located, this ROM image (called *View*) could be loaded into bank 5 of sideways RAM by typing:

```
*SRLOAD View 8000 5
```

Note that the load address for sideways ROM images is always 8000 (hexadecimal).

Listing 2

```
 10 REM Sideways-RAM Loader
 20 REM Version B1.0
 30 REM Author  Jonathan Ribbens
 40 REM BEEBUG  May 1990
 50 REM Program subject to copyright
 60 :
100 FOR p=0 TO 2 STEP 2
110 P%=&900:[OPT p
120 :
130 .rload
140 JSR message
150 JSR getfilename
160 JSR getrambank
170 JSR checkram
180 JSR disctoram
190 JSR ramtoram
200 JSR initialiseram
210 JMP tidyup
220 :
230 .message
240 JSRprint:EQUS"Sideways RAM Loader"
250 EQUB13:EQUS"By Jonathan Ribbens"
260 EQUB13:EQUB13:BRK:RTS
270 :
280 .getfilename LDA#1:STA&72
290 LDA#1:LDY#0:LDX#&70:JSR&FFDA
300 JSRspaces:CMP#&D:BEQinputfilename
310 LDX#0:.lp1:LDA(&70),Y
```

```
 320 STAfilename,X:INX:INY:CMP#&D
 330 BEQefi:CMP#&20:BEQelp1:JMPlp1
 340 .elp1:LDA#0:STA&72:.efi:RTS
 350 .inputfilename
 360 JSRprint:EQUS"Enter filename:"
 370 BRK:LDA#0:LDX#ifblock MOD256
 380 LDY#ifblock DIV256:JSR&FFF1
 390 JSR&FFE7:JSRescape:RTS
 400 :
 410 .getrambank
 420 LDA&72:BNEinputrambank
 430 JSRspaces:CMP#&D:BEQinputrambank
 440 JMPasctohex:.inputrambank
 450 JSRprint:EQUS"Enter RAM bank:"
 460 BRK:LDA#0:LDX#irblock MOD256
 470 LDY#irblock DIV256:JSR&FFF1
 480 JSR&FFE7
 490 JSRescape:LDA&73:.asctohex
 500 CMP#ASC"0":BCCbadnum
 510 CMP#ASC"F"+1:BCSbadnum
 520 CMP#ASC"9"+1:BCCdectohex
 530 CMP#ASC"A":BCShextohex
 540 .badnum:BRK:BRK
 550 EQUS"Bad hex":BRK:.dectohex
 560 SEC:SBC#48:STA&73:RTS:.hextohex
 570 SEC:SBC#55:STA&73:RTS
 580 :
 590 .checkram
 600 LDX&F4:LDA&73:STA&F4:STA&FE30
 610 LDA&8123:INC&8123:CMP&8123
 620 BEQnotram:STA&8123:RTS:.notram
 630 BRK:BRK:EQUS"Bank is not RAM":BRK
 640 :
 650 .disctoram LDA#&FF
 660 LDX#drblock MOD256
 670 LDY#drblock DIV256:JMP&FFDD
 680 :
 690 .ramtoram
 700 LDX&F4:LDA&73:STA&F4:STA&FE30
 710 LDA#0:STA&74:STA&76:LDA#&80:STA&75
 720 LDA#&30:STA&77:.lp2:LDY#0:.lp3
 730 LDA(&76),Y:STA(&74),Y
 740 INY:BNElp3:INC&75:INC&77:LDA&77
 750 CMP#&70:BNElp2:STX&F4:STX&FE30
 760 RTS
 770 :
 780 .initialiseram
 790 LDX&73:LDA&3006:STA&2A1,X:RTS
 800 :
 810 .tidyup
 820 JSRprint:EQUS"Loaded.":EQUB13
 830 EQUB13:BRK:RTS
 840 :
 850 .spaces
 860 DEY:.lp4:INY:LDA(&70),Y:CMP#&20
 870 BEQlp4:RTS
 880 :
 890 .print
 900 PLA:STA&78:PLA:STA&79:LDY#0
 910 .lp5:INY:LDA(&78),Y:JSR&FFE3
 920 BNElp5:TYA:CLC:ADC&78
 930 STA&78:LDA&79:ADC#0:PHA
 940 LDA&78:PHA:RTS
 950 :
 960 .escape
 970 PHA:PHP:BIT&FF:BMIescape2
 980 PLP:PLA:RTS:.escape2:BRK
 990 BRK:EQUS"Escape - Load aborted"
1000 BRK
1010 :
1020 .ifblock
1030 EQUWfilename:EQUB20
1040 EQUB0:EQUB&FF
1050 :
1060 .irblock
1070 EQUW&73:EQUB1:EQUB0:EQUB&FF
1080 :
1090 .drblock
1100 EQUWfilename
1110 EQUD&3000:EQUD0
1120 EQUD0:EQUD0
1130 :
1140 .filename
1150 EQUS"0123456789012345678 9"
1160 :
1170 ]:NEXT
1180 *SAVE RLoad 900 AE1 900 900
1190 PRINT"Utility RLoad saved."
1200 END
```

Again, Jonathan Ribbens has provided an equivalent utility for model B users, and this is given in Listing 2. Enter, save and run this as before. This time the resultant machine code utility will be saved as *RLoad*. To use the utility type:

```
*RLoad
```

(with the disc containing RLoad in the drive) and it will then prompt for the file name of the ROM image, and secondly for the number of the RAM bank into which the ROM is to be installed. To locate a vacant RAM bank use the *ROMS command or the RList utility described before. The same command or utility can be used to confirm that the ROM image has been loaded.

Many ROMs need to be initialised before they will perform correctly. On a Master 128 this can be accomplished by pressing Ctrl-Break after loading the ROM image. On a Master Compact the same can also be achieved by appending the letter 'I' at the end of the command line, for example:

```
*SRLOAD View 8000 5 I
```

For model B users, there is a further utility which will perform the same task. This is given as Listing 3. Type it in, save it and run it, and the assembled code will be saved as RInit. Typing *RInit will initialise any and all ROM images installed.

Listing 3

```
 10 REM Sideways-ROM Initialiser
 20 REM Version B1.0
 30 REM Author  Jonathan Ribbens
 40 REM BEEBUG  May 1990
 50 REM Program subject to copyright
 60 :
100 FOR p=0 TO 2 STEP 2
110 P%=&900:[OPT p
120 :
130 .rinit
140 JSR message
150 JSR initialiseroms
160 JMP tidyup
170 :
180 .message JSRprint
190 EQUS"Sideways ROM Initialiser"
200 EQUB13:EQUS"By Jonathan Ribbens"
210 EQUB13:EQUB13:BRK:RTS
220 :
230 .print
240 PLA:STA&78:PLA:STA&79:LDY#0
250 .lp1:INY:LDA(&78),Y:JSR&FFE3
260 BNElp1:TYA:CLC:ADC&78
270 STA&78:LDA&79:ADC#0:PHA
280 LDA&78:PHA:RTS
290 :
300 .initialiseroms
310 LDA&F4:PHA:LDA#15:STA&70:.lp2
320 LDA&70:STA&F4:STA&FE30
330 LDX&8007:LDA&8000,X
340 BNEcont:LDA&8001,X
350 CMP#ASC"(":BNEcont:LDA&8002,X
360 CMP#ASC"C":BNEcont:LDA&8003,X
370 CMP#ASC")":BNEcont:LDA&8006
380 LDX&70:STA&2A1,X:.cont
390 DEC&70:BPLlp2:PLA:STA&F4
400 STA&FE30:RTS
410 :
420 .tidyup JSRprint
430 EQUS"ROMs initialised":EQUB13
440 EQUB13:BRK:RTS
450 ]:NEXT
460 *SAVE RInit 900+AF 900 900
470 PRINT"Utility RInit saved."
480 END
```

SAVING SIDEWAYS RAM IMAGES

Once again, the Master and Compact have a command *SRSAVE which allows the contents of any area of sideways RAM to be saved to disc. Thus:

```
*SRSAVE MyROM 8000+8000 5
```

will save the entire contents of RAM bank 5 in a file called *MyROM*. Don't worry about the numbers - if you need to perform this task just copy the format above with your choice of file name and relevant ROM bank.

Our final listing this month provides the same facility for model B users (in a simpler format if anything). Type in the program, save it, and then run it to create the utility RSave. You can then run this with:

```
*RSave
```

and you will be prompted for the file name to use, and the number of the ROM bank to be copied into the file.

I hope that last month's introduction and the more practical details given here will enable all users to make full use of any sideways ROM/RAM fitted to their machines. We also expect to be publishing some more sophisticated applications for sideways RAM in future issues, though these will appear as separate articles and not as part of our First Course series. Finally, my thanks to Jonathan Ribbens for the four sideways RAM utilities which I have listed here.

TECHNICAL NOTE

The four utilities listed here assemble and run in memory from address &900 onwards. This will mean that the serial port, cassette filing

system and sound buffers will be corrupted and unusable when these utilities are invoked.

Listing 4

```
 10 REM Sideways-RAM Saver
 20 REM By Jonathan Ribbens
 30 :
 40 MODE7:FORp=0TO2STEP2:P%=&900:[OPTp
 50 :
 60 .rsave
 70 JSR message
 80 JSR getfilename
 90 JSR getrambank
100 JSR ramtoram
110 JSR ramtodisc
120 JMP tidyup
130 :
140 .message
150 JSRprint:EQUS"Sideways-Ram Saver"
160 EQUB13:EQUS"By Jonathan Ribbens"
170 EQUB13:EQUB13:BRK:RTS
180 :
190 .getfilename LDA#1:STA&72
200 LDA#1:LDY#0:LDX#&70:JSR&FFDA
210 JSRspaces:CMP#&D:BEQinputfilename
220 LDX#0:.lp1:LDA(&70),Y
230 STAfilename,X:INX:INY:CMP#&D
240 BEQefi:CMP#&20:BEQelp1:JMPlp1
250 .elp1:LDA#0:STA&72:.efi:RTS
260 .inputfilename
270 JSRprint:EQUS"Enter filename:"
280 BRK:LDA#0:LDX#ifblock MOD256
290 LDY#ifblock DIV256:JSR&FFF1
300 JSR&FFE7:JSRescape:RTS
310 :
320 .getrambank
330 LDA&72:BNEinputrambank
340 JSRspaces:CMP#&D:BEQinputrambank
350 JMPasctohex:.inputrambank
360 JSRprint:EQUS"Enter ram bank:"
370 BRK:LDA#0:LDX#irblock MOD256
380 LDY#irblock DIV256:JSR&FFF1
390 JSR&FFE7
400 JSRescape:LDA&73:.asctohex
410 CMP#ASC"0":BCCbadnum
420 CMP#ASC"F"+1:BCSbadnum
430 CMP#ASC"9"+1:BCCdectohex
440 CMP#ASC"A":BCShextohex
450 .badnum:BRK:BRK
460 EQUS"Bad hex":BRK:.dectohex
470 SEC:SBC#48:STA&73:RTS:.hextohex
480 SEC:SBC#55:STA&73:RTS
490 :
500 .ramtodisc LDA#0
510 LDX#drblock MOD256
520 LDY#drblock DIV256:JMP&FFDD
530 :
540 .ramtoram
550 LDX&F4:LDA&73:STA&F4:STA&FE30
560 LDA#0:STA&74:STA&76:LDA#&80:STA&75
570 LDA#&30:STA&77:.lp2:LDY#0:.lp3
580 LDA(&74),Y:STA(&76),Y
590 INY:BNElp3:INC&75:INC&77:LDA&77
600 CMP#&70:BNElp2:STX&F4:STX&FE30
610 RTS
620 :
630 .tidyup
640 JSRprint:EQUS"Saved.":EQUB13
650 EQUB13:BRK:RTS
660 :
670 .spaces
680 DEY:.lp4:INY:LDA(&70),Y:CMP#&20
690 BEQlp4:RTS
700 :
710 .print
720 PLA:STA&78:PLA:STA&79:LDY#0
730 .lp5:INY:LDA(&78),Y:JSR&FFE3
740 BNElp5:TYA:CLC:ADC&78
750 STA&78:LDA&79:ADC#0:PHA
760 LDA&78:PHA:RTS
770 :
780 .escape
790 PHA:PHP:BIT&FF:BMIescape2
800 PLP:PLA:RTS:.escape2:BRK
810 BRK:EQUS"Escape - Save aborted"
820 BRK
830 :
840 .ifblock
850 EQUWfilename:EQUB20
860 EQUB0:EQUB&FF
870 :
880 .irblock
890 EQUW&73:EQUB1:EQUB0:EQUB&FF
900 :
910 .drblock
920 EQUWfilename
930 EQUD&8000:EQUD&8000
940 EQUD&3000:EQUD&7000
950 :
960 .filename
970 EQUS"01234567890123456789"
980 :
990 ]:NEXT
1000 *Save RSave 900+1A6 900 900
1010 PRINT"Utility RSave saved."
1020 END
```

EdiKit (Part 5)

In this final part of the series, Bill Hine explains how to add your own commands, or those taken from previously published programs, to the EdiKit ROM. As an example, he shows how to incorporate David Spencer's Partial Renumber Program (BEEBUG Vol.7 No.7) into EdiKit.

MODULAR PROGRAMS

One of the main problems in writing software for sideways ROMs is that an assembler program requires several times as much memory as the code it creates will finally occupy; typically four or five times as much. To assemble 16k bytes of code (the size of a full ROM image) would therefore require a program of up to 80k, far too big for a BBC micro or Master which, even with shadow memory, has a maximum of about 29k of user memory. One possible solution is to use 128k Basic, but even this is unlikely to be long enough for a full 16k ROM image.

The most practical alternative is to adopt a modular approach, building up the ROM from several assembler programs. However, linking the programs together can become something of a nightmare as you are bound to have to make calls from one program to another. A small alteration in one program will result in changes to the addresses of all the subsequent routines in it, so requiring changes to all the other programs which call them. Some way must be found of making each program as independent as possible of changes in the others.

One solution is to use indirect calls at unchanging addresses, and it is this method which has been adopted in EdiKit. Acorn, of course, uses the same system of indirect calls for its OS routines (OSBYTE, OSWORD etc.) to allow them to alter the code from time to time.

If you have typed in the previous programs EDIKIT2, 3 and 4, you will have noticed a number of things about them. For example, each begins at a new page of the ROM, i.e. at an exact interval of &100 bytes (see table). Although this wastes some memory, it greatly simplifies the linking together of programs, and allows for a certain amount of expansion of the code during debugging and development. The amount of memory reserved is slightly more than an exact number of pages (see the DIM expressions in line 110 of each EdiKit program) so that there is an overlap with the next program in the series.

Program	Start Adr	Size	End Adr
EDIKIT1	&8000	&C00	&8C00
EDIKIT2	&8C00	&600	&9200
EDIKIT3	&9200	&600	&9800
EDIKIT4	&9800	&600	&9E00
PartRenum	&9E00	&C00	&AA00
YourProg	&AA00	????	?????

Table of EDIKIT addresses and sizes of code.

Each program after the first contains a list of routine addresses (romexit=&8803, escexit=&8806 etc.). The assembler code in each also begins and ends with a series of JMP instructions. Those at the start, following the label *.proccalls*, represent calls to the various ROM commands. They jump either to the routines included in the program, or to another series of jumps beginning at *start+size* created by PROClinks. When the next program is run, the code it creates overlays these links with a new set of JMPs beginning at *.proccalls* in the new program. The JMPs following the label *.rtncalls* are to routines which may be of use in subsequent programs; since the JMPs following *.proccalls* will never need changing, the addresses of the jumps to routines are fixed too. They can be calculated simply by printing the value of *rtncalls* and adding 3 for each jump after the first.

Jumps to other routines can be added to the list of *rtncalls* without causing major disruption. Alterations can then be made to one program, during development or subsequently, without requiring all the others to be altered.

For example, if you refer back to EDIKIT1, you will see that the *FBASIC call is made to jump directly to &8C00 (lines 2420 to 2460). PROClinks in EDIKIT1 produces a series of JMPs starting at &8C00 to *notimp*, a simple routine printing out a "Not implemented"

message. Note that &8C00 = &8000 + &C00 = start + size. The code produced by EDIKIT2 starts at &8C00, overwriting these *JMP notimp* instructions with jumps to the real routines, in the case of *FBASIC and *FPROCFN, or to another set of *JMP notimp* instructions at &9200 (= start + size in EDIKIT2) for the remaining commands. These are set up by a version of PROClinks at the end of EDIKIT2. Calls are also set up near the start of EDIKIT2 to routines within it which may be of use to subsequent programs (lines 1590 to 1620). You will find these calls listed, together with those from EDIKIT1, in lines 1150 to 1220 of EDIKIT3. In this way calls can be made both forwards and backwards between the modular programs.

Once these principles are understood it is relatively simple to add your own code to the ROM. You must ensure that your code overlays the JMPs in PROClinks of EDIKIT4 with a jump to the code for your own commands, and passes on any unused command calls to a set of links at the end of your code. To do this, you will need to set the variable *start* to the same value as the end of the previous program. The above table will help with this. Keep a copy of the table and add the new data to it as your ROM grows.

You will also need to set *size* to an appropriate value. You can read off the length of the code from the assembler output when you run your program. Alternatively, if it is someone else's program the DIM statement at the start should tell you how much memory it uses; this figure should be rounded up to the next whole page and *size* equated to it. Don't forget to adjust the size of the DIM declaration also. This will need to be a little larger than size itself to allow for the links. They require three bytes per jump. The links themselves can be made by copying PROClinks word for word from any of the EdiKit programs; the only adjustment necessary is to the number of cycles of the FOR loop, one per jump.

There are two more important points. Firstly, you must ensure that you include romexit=&8803 and notimp=&8818 in your initial list of variables. You will need *notimp* for the call from PROClinks, of course. In addition your command must exit with *JMP romexit* instead of the normal *RTS*. This will ensure that your command returns correctly without any unbalanced PHAs or PLAs - provided of course your own code is balanced in this respect! If you wish to use any of the other EdiKit routines yourself you could also include their addresses in your initial list of variables; two useful ones might be escexit=&8806 and prnttext=&8815.

Secondly, you will need to include the name of your command at two places in EDIKIT1; in the command list (lines 1880 to 2120) and the command table (lines 2460 to 2670). You will obviously substitute your name for the first free dummy command name, currently *PCOMM*. If there are any parameters to the command you should include these in the command list which is printed out in response to a *HELP call.

To create the new version of Edirom, RUN each of the EdiKit programs in turn followed by your own program. Don't forget to save a copy of the complete ROM.

Important note - the listing for Edikit1 published in Vol.8 No.7 needs to be altered to make it work reliably. The following lines should be amended/added:

```
2240 CMP#nl:BEQ endofip2
2311 .endofip2
2312 LDA comtable,X:BEQ jmpcom2
2313 JMP nxtcomlp
2430 INY:.jmpcom2:INX:LDA comtable,X:STA hi
```

Having made these alterations, the whole of Edikit, including Parts 2,3 & 4, must then be re-run in sequence before saving the final ROM image.

INCORPORATING PARTIAL RENUMBER INTO EDIKIT

First of all, type in a copy of the Partial Renumber program from the December 1988 issue of BEEBUG Magazine (Vol.7 No.7), omitting lines 220 to 470. These are not needed because we already have our own EdiKit ROM header and command jump routines. You should also delete line 2550. Now add the additional lines given in Listing 1, which provide all the things we have discussed above. Note that size is set to &C00, the next full page above 3000 decimal which is &BB8 hex. We have used *begin* in place of *start* since David Spencer uses *start* as a variable name himself. Save the resulting program.

Continued on page 35

Mechanical Vibration

by Donald Tattersfield

INTRODUCTION

Vibration, in one form or another, concerns us all in our daily lives. When we travel along a bumpy road in a motor car this is very evident, however well-sprung the car may be. The study of vibration is a complex subject, but this program will allow you to appreciate the characteristics of a vibrating system without any knowledge of complex mathematics. It shows the results that are obtained in both graphical form (the preferred method) or in tabular format.

THE SYSTEM

As a simple example, we shall take a spring, fixed at the top, on the bottom of which we carefully place a mass. The spring will extend slowly until it comes to rest. If the mass is pulled down further from this position and released, it will execute vertical vibrations according to the characteristics of the system. In the SI system of units these are: the mass M (kg); the stiffness of the spring S (N/m), which is the force needed to extend the spring by one unit of length; the damping force D (N/m/s), which is the force resisting the motion at unit velocity; and any other force F (N), such as the impacts of the road on the car wheels in the introduction above, which may or may not depend on the time t (seconds). This system is shown in figure 1.

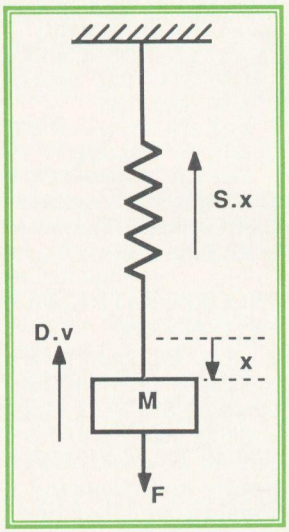


Figure 1

SOME VALUES

Before we type in the program and run it, we shall need some initial values for the characteristics of our system. After that, the opportunity is given in the program to alter one characteristic at a time so that we can see the effect on the vibration. We shall have the option to display either numerical values of x, the displacement of the mass from the mean position, for the corresponding values of t; or we may display a graph of x against t as the vibration proceeds.

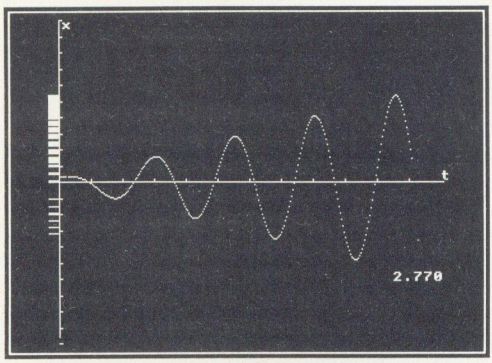
We shall start with a mass M of, say, 2kg. Let us say that this extends the spring initially by 10cm (0.1m). The gravitational force on the mass is M*g, where g=9.81m/s/s, the acceleration due to gravity. This force is about 20N. It is supported by a spring force of S*0.1N, so the stiffness of the spring is 20/0.1 or 200N/m. Now we will pull the mass downwards a further distance X0=5cm (0.05m) and let it go, so the initial velocity V0 is zero. To simplify our example still further, we shall assume D=0 and F=0. The method used in the program obliges us to use a small interval of time between the values of x displayed, so we will choose dt=0.01 seconds. The limits of displacement above and below the mean position will be about 0.05m. This has to fit into a maximum of 500 pixels vertically on the screen, so an *xscale* factor of 500/0.05=10000 seems to be reasonable. In fact, you will find that *xscale*=3000 is better. Finally, let us say that we wish to display at least 4 complete vibrations in 1200 pixels horizontally, so a *tscale* of 1200/4=300 would be sufficient. In fact, we shall use a *tscale* of 400.

RUNNING THE PROGRAM

Now that we have a reasonable set of values to enter, type in the program, save it and run. When prompted to opt for a numerical display, first reply 'N'. Then enter successively *mass*=2, *damping*=0, *spring*=200, *X0*=0.05, *V0*=0, *dt*=0.01, *xscale*=3000, *tscale*=400 and watch the mass vibrating.

MODIFYING THE CHARACTERISTICS

At any time you may press the 'Q' key to stop the display (you may also need to press Shift if you are in numeric mode). A menu will appear on the screen, displaying the values you have used. If you wish to alter one of the characteristics of the system, reply 'Y'. If you reply 'N', you will be asked if you want to repeat with the existing values, or exit. Select the parameter you wish to alter, and then type in the new value. To illustrate, select item (2) and a new value of 3 for damping. You will now have a graphical display of a damped vibration. If you select item (9) *numerical*="Y", the numerical values will be displayed instead of the graph. At any time pressing Escape will return you to the beginning of the program, where you will need to input a complete set of new values. Shift-Escape provides an immediate exit from the program.



Undamped oscillation

SPECIAL CONDITIONS

Apart from looking at the effects of altering each characteristic, four specific cases are worthy of inspection.

(1) Simple harmonic motion: D=0 and F=0. This has already been illustrated in our first example. F=0 is set automatically by line 2060 of the program.

(2) Damped motion: D*D<4*M*S, has been illustrated in our second example, with F=0 again.

(3) Critically damped motion: D*D=4*M*S, using values M=2, D=40, S=200, X0=0.05, V0=0, dt=0.01, xscale=4000, tscale=800, F=0

(4) Resonance: BEFORE running this case alter line 2060 of the program to:

```
2060 DEF FNdisturb(t)=2*COS(10*t)
```

Then enter values M=2, D=0, S=200, xo=0.01, vo=0, dt=0.01, xscale=1000, tscale=400. Here we have an external force, applied additionally, whose value varies gradually with time from 2N downwards to 2N upwards with a frequency related to SQR(S/M)=10.

GENERAL

To illustrate this program we have used specific values, and have shown how they might reasonably be obtained, but, of course, you can use your own values - a good source is a text book. If you wish to use the Imperial System of units, or any other system, you may do so without any alteration to the program, but you must be consistent throughout. The method makes only one assumption - that the acceleration of the mass does not change over a very short time interval. The simple equations for a body moving with uniform acceleration can then be applied with only a minor modification. Again, you do not need to understand any of the calculations to enjoy the program.

```
   10 REM Program Vibration
   20 REM Version B1.4
   30 REM Author  Donald Tattersfield
   40 REM BEEBUG  May 1990
   50 REM Program subject to copyright
   60 :
  100 MODE 7:ON ERROR GOTO350
  110 PROCtitle:PROCinit
  120 MODE 7:PROCmenu
  130 REPEAT
  140 IF num% MODE 7 ELSE MODE 1:VDU5
  150 VDU23,1,0;0;0;0;
  160 IF num% CLS:VDU 14:VDU130:PRINT TA
B(10);"t";TAB(25);"x" ELSE PROCaxes
  170 REM * Conditions for start *
  180 PROCstart:@%=&20309
  190 REM * Start of loop *
  200 REPEAT
  210 IF NOT num% PROCprinttime
  220 t=t+dt:sax=ax
  230 ax=(FNdisturb(t))/M-D/M*v-S/M*x
  240 dax=(ax-sax)/dt
```

```
  250 x=x+v*dt+ax*dt*dt/2
  260 v=v+ax*dt+dax*dt*dt/2
  270 IF num% PRINT'TAB(10);t;TAB(25);x
ELSE PROCbody
  280 REM * End of loop *
  290 Q$=INKEY$(0)
  300 UNTIL Q$="Q" OR Q$="q"
  310 REPEAT:MODE7:menu=FNrepeat:UNTIL m
enu<1
  320 UNTIL menu=0
  330 MODE7:@%=10:END
  340 :
  350 IF ERR=17 AND INKEY-1 MODE7:END
  360 IF ERR=17 THEN 120
  370 MODE7:REPORT:PRINT" at line ";ERL
  380 END
  390 :
 1000 DEF PROCinit
 1010 CC=180/PI:CRLF$=CHR$13+CHR$10
 1020 REM * Graph image *
 1030 VDU 23,224,128,0,0,0,0,0,0,0
 1040 REM * Mass image *
 1050 VDU 23,225,255,255,255,255,0,0,0,0
 1060 ENDPROC
 1070 :
 1080 DEF PROCtitle
 1090 VDU23,1,0;0;0;0;
 1100 t$=CHR$(146)+CHR$(57)+CHR$(185)
 1110 PRINT TAB(5,4) STRING$(10,t$)
 1120 PRINT TAB(5,16) STRING$(10,t$)
 1130 PRINT TAB(14,8) CHR$(141);CHR$(133
);"VIBRATION"
 1140 PRINT TAB(14,9) CHR$(141);CHR$(133
);"VIBRATION"
 1150 PRINT TAB(10,12) CHR$(131);"Donald
 Tattersfield"
 1160 TIME=0:REPEAT UNTIL TIME>200
 1170 ENDPROC
 1180 :
 1190 DEF PROCmenu
 1200 xscale=0:tscale=0:s$=""
 1210 VDU131:num%=FNYesNo("Do you requir
e a numerical result")
 1220 IF num% num$="Yes" ELSE num$="No"
 1230 PRINT''CHR$134"* Input system para
meters *"
 1240 VDU130:INPUT"Mass: "mass
 1250 VDU130:INPUT "Damping coefficient:
 "damping
 1260 VDU130:INPUT "Spring constant: "sp
ring
 1270 PRINT'CHR$134"* Input starting con
ditions *"
 1280 VDU130:INPUT "Initial displacement
 at t=0: "xo
 1290 VDU130:INPUT "Initial velocity at
t=0: "vo
 1300 VDU130:INPUT "Chosen time interval
 (s): "dt
 1310 IF NOT num% THEN PROCmenu1
 1320 ENDPROC
 1330 :
 1340 DEF PROCmenu1
 1350 PRINT'CHR$134"* Input scales for d
isplay *"
 1360 VDU130:INPUT "scale of x-axis: "xs
cale
 1370 VDU130:INPUT "scale of t-axis: "ts
cale
 1380 ENDPROC
 1390 :
 1400 DEF PROCaxes
 1410 GCOL 0,129:CLG
 1420 REM * Draw and label axes *
 1430 GCOL 0,0
 1440 MOVE 40,500:DRAW 1250,500
 1450 MOVE 40,20:DRAW 40,1000
 1460 MOVE 50,1000:PRINT CHR$120
 1470 MOVE 1240,535:PRINT CHR$116
 1480 FOR I=1 TO 11
 1490 MOVE 40+100*I,500:DRAW 40+100*I,51
0
 1500 NEXT I
 1510 FOR I=0 TO 1000 STEP 10
 1520 MOVE 40,5*I:DRAW 50,5*I
 1530 NEXT I
 1540 ENDPROC
 1550 :
 1560 DEF PROCstart
 1570 M=mass:D=damping:S=spring
 1580 x=xo:v=vo:t=0:menu=0
 1590 ax=0:sax=0
 1600 ENDPROC
 1610 :
 1620 DEF PROCbody
 1630 MOVE 40+tscale*t,500+xscale*x
 1640 GCOL 0,2:PRINT CHR$(224)
 1650 MOVE 10,500+xscale*x
 1660 IF v<=0 THEN GCOL 0,0 ELSE GCOL 0,
1
 1670 PRINT CHR$(225)
 1680 ENDPROC
 1690 :
 1700 DEF PROCprinttime
 1710 VDU4:COLOUR129
 1720 GCOL 0,2:PRINTTAB(34,25);t
 1730 GCOL 0,0:PRINTTAB(34,25);t
 1740 VDU 5:GCOL 0,1
 1750 ENDPROC
 1760 :
```

```
 1770 DEF FNrepeat
 1780 GCOL 0,1
 1790 VDU134:PRINT"Values used:"
 1800 VDU130:PRINT"1. mass:";SPC(10-LEN(
STR$(INT(mass))));mass
 1810 VDU130:PRINT"2. damping:";SPC(7-LE
N(STR$(INT(damping))));damping
 1820 VDU130:PRINT"3. spring:";SPC(8-LEN
(STR$(INT(spring))));spring
 1830 VDU130:PRINT"4. xo:";SPC(12-LEN(ST
R$(INT(xo))));xo
 1840 VDU130:PRINT"5. vo:";SPC(12-LEN(ST
R$(INT(vo))));vo
 1850 VDU130:PRINT"6. dt:";SPC(12-LEN(ST
R$(INT(dt))));dt
 1860 VDU130:PRINT"7. xscale:";SPC(8-LEN
(STR$(INT(xscale))));xscale
 1870 VDU130:PRINT"8. tscale:";SPC(8-LEN
(STR$(INT(tscale))));tscale
 1880 VDU130:PRINT"9. numerical:";SPC5;n
um$
 1890 IF FNYesNo(CRLF$+"Change ONE value
") THEN PRINT'"Menu selection";:REPEAT:m
enu=GET-48:UNTIL menu>0 AND menu<10 ELSE
 IF FNYesNo(CRLF$+"Repeat") THEN=-1 ELSE
=0
```

```
 1900 ON menu GOSUB 1930,1940,1950,1960,
1970,1980,1990,2000,2010
 1910 =menu
 1920 :
 1930 INPUT'"New mass: "mass:RETURN
 1940 INPUT'"New damping: "damping:RETUR
N
 1950 INPUT'"New spring: "spring:RETURN
 1960 INPUT'"New xo: "xo:RETURN
 1970 INPUT'"New vo: "vo:RETURN
 1980 INPUT'"New dt: "dt:RETURN
 1990 INPUT'"New xscale: "xscale:RETURN
 2000 INPUT'"New tscale: "tscale:RETURN
 2010 PRINT:num%=FNYesNo("New numerical"
):IF num% num$="Yes" ELSE num$="No"
 2020 IF NOT num% AND (xscale=0 OR tscal
e=0) PROCmenu1
 2030 RETURN
 2040 :
 2050 DEF FNdisturb(t)=0
 2060 :
 2070 DEF FNYesNo(text$)
 2080 PRINT text$+" (Y/N)";
 2090 REPEAT:ans$=CHR$(GET AND &DF):UNTI
L INSTR("YN",ans$)
 2100 =(ans$="Y")
```

B

EdiKit (continued from page 31)

Replace PCOMM and QCOMM in lines 2570,2580 of EDIKIT1 with RENUMBER and CHECKORDER, and in lines 2020,2030 with the full command definitions as given by David Spencer in his article (middle of the first column of page 12). You can use *RTEXT to make these substitutions.

To save time and effort we are making available a complete version of the EdiKit ROM incorporating the partial renumber facility and all the other facilities of the Basic Booster ROM which was released last year (some of which have been updated). The new ROM is available on a 16K EPROM to plug into your machine, or as a sideways ROM image on disc for loading into sideways RAM. Full details of this new BEEBUG magazine product are included elsewhere in this issue. The enhanced EdiKit ROM will be available at a special price for those who had previously purchased the Basic Booster.

```
  60 REM Modifications and additions to
  70 REM Partial Renumber by W.D.Hine
  80 REM for inclusion in EDIKIT ROM
  90 REM (c) BEEBUG May 1990
 175 romexit=&8803:notimp=&8818
 180 DIMcode&C20
 185 begin=&9E00:size=&C00
 200 P%=begin:O%=code
 220 JMP rnum:JMP chkord
 230 .rcomm JMP &AA00
 240 .scomm JMP &AA03
 250 .tcomm JMP &AA06
 260 .ucomm JMP &AA09
 270 .vcomm JMP &AA0C
 280 .wcomm JMP &AA0F
 290 .xcomm JMP &AA12
 300 .ycomm JMP &AA15
 310 .zcomm JMP &AA18
 410 .cout JMP romexit
30000 P%=begin+size:O%=code+size
30010 FOR I%=0 TO8
30020 [OPT 7:JMP notimp:]
30030 NEXT
30040 :
30050 PRINT"SAVE and LOAD ROM? Y/N"
30060 A=GET AND&DF:IFA<>ASC"Y"THENEND
30070 OSCLI"SAVE RENUMcd "+STR$~code+" "
+STR$~O%
30080 *SRLOAD RENUMcd 9E00 X
30090 *SRSAVE edirom 8000 BFFF X
30100 END
```

B

BEEBUG Education

by Mark Sealey

National Educational Resources Information Services (NERIS)
Maryland College,
26 Leighton Street,
Woburn MK17 9JD.
Tel: (0525) 290364
Mailboxes: Prestel 052525364
Campus 2000 TCD 100861
with centres in Cardiff, Glasgow and Belfast.

NERIS rates (annual):

Primary Schools	£90
Secondary Schools	£140
Colleges/LEA Centres	£140
CD-ROM	£140

with software available for both BBC and Archimedes computers.

This month Beebug Education explores one of the most exciting and important educational resources currently available for anyone working with IT or indeed in any area of education, the *National Educational Resources Information Service*, or NERIS.

NERIS is an extensive and extremely useful on-line database providing news, curriculum support material, teaching resources and guides to all areas of education in the UK. Within the last few weeks, for example, the latest National Curriculum documents on English and Technology have been made available on the system.

If you already make use of comms in education and have not yet tried accessing NERIS, it really ought to be a priority. If you and/or your institution have yet to take the plunge, then NERIS is as good a way to start as any. The scope of the database is wide indeed. At the last count, nearly 30,000 items were to be found within the system from material on Industrial Awareness, to pupil appraisal, to teaching methods, art and chemistry etc., a very wide spread of educational and learning interests indeed.

SCOPE

The initial funding of NERIS has come from the Industry and Education Unit of the Department of Trade and Industry, the body which launched and financed the micros in schools schemes. Copyright of material remains with the DES, DTI or HMSO. The essential brief has been to maintain and develop a curriculum database. This began with material on and coverage of Science, Mathematics, Geography and Social/Personal Development. However, it extends well beyond these areas now, and the aim is to provide material relating to all curricular areas.

The emphasis is on the practical. Actual work-sheets as well as graphical material (and software itself: BBC and Archimedes formats) can be down-loaded for use by teachers and lecturers; there are reviews of material, references to case studies, contact lists, other sources and curriculum development projects etc.

If material is worthy of attention but not actually available through NERIS itself, then there are details of its whereabouts, bibliographical data and often notes on its use.

NERIS also carries information on Educational broadcasts provided by the BBC and IBA. At the same time details are to be found of places to visit and educational study centres etc.

ACCESSING NERIS

Although under continual development, there are now four routes into the database, which is held on the Open University computer. All four require a computer with appropriate software, a modem and telephone line: via a Prestel Gateway, through Campus 2000 (formally TTNS), via the prestigious University/College network, JANET, and directly.

This means that, provided your call is to a local number, costs can be kept as low as possible. Access is guaranteed from 8.00 am on Mondays till 10.00 pm Fridays. The material is available by subscription on CD-ROM for machines which can use this medium; this includes both the 8 and 32 bit Acorn ranges.

NERIS also provides - on joining - software which will decode and make usable the telesoftware which can also be downloaded. You get a password and comprehensive User Guide etc., though the format of this is currently being improved and updated to coincide with new subscription arrangements which took effect at the end of March this year.

USING THE DATABASE ON LINE

Since NERIS is a videotext-style database, anyone used to Prestel will feel at home with it. There is a very slight delay while using the Gateways as the number, keyword or digit which you enter in response to any one screen or menu is collected. The menus are consistent with Prestel usage and worked faultlessly during compilation of this review. Indeed, NERIS has a very professional and robust feel to it.

SEARCHING

Assume you want to find out what is available on the topics of "Music" and the "National Curriculum". The response frame requires three criteria to be detailed. First the Interest Area, in this case Music linked by Boolean AND with "National Curriculum". A hash '#' stands for "anything", though is clearly not of much use in this field.

Next, you specify the resource medium or media - on computer, printed, broadcast, or whatever - and finally the age-range - from pre-school to post-16. At this stage, as with most others, two or three lines at the foot of the screen offer *help*, *abort* or *retry* options.

Once your selection has been accepted - and if there is an error, you will be invited to retry - you are given a resume of what was asked for together with the option to view the document titles, amend the request or terminate the search. If you proceed, just selecting the number by which the document is indexed (usually a single digit reference - four or five to a page) will lead you quickly to the material itself.

All of this is somewhat like using a library catalogue - with full publishing details - except that the actual contents ARE readily available. In fact it takes longer and sounds more complicated to describe than it is in use. Sometimes, though, your search may be more exacting.

In such cases NERIS has an extended on-line system. This deals with specific searches and is in the form of a standard five-field screen which takes you through the inquiry step by step until your search criteria are completed and as precise or as general as you wish. There is some system feedback whilst you are compiling the inquiry and wildcards (using '+') are possible. The criteria can relate to one or more of: TItle, MEdia, Age Phase, DataFile, Author and PuBlisher, where the upper case characters are used to code the search.

In practice both systems work well and are as interactive as videotext-style response frames can be. Files of information which are spread out over more than one frame, for example, follow the same practice as Teletext in having some indication of where you are in the sequence... 2/5 meaning the second page in a five page block. "More.." is always 8 (or 7 if you're at the last page).

USEFULNESS TO EDUCATIONALISTS

All this would not be much good, though, if the quality and quantity of material were not extensive, well arranged and up to date. Almost by definition no resource base can ever be "complete". NERIS is still developing; at present there are items from over a thousand sources.

There is an inevitable slant towards the National Curriculum and its structure is reflected in much of the cross referencing which the NERIS format makes possible. Experienced teachers may have their doubts about one intention of the enterprise, namely "I am working on Science Attainment Target 5 at Level 3; what resources can NERIS suggest to me?"

Nevertheless, time to most teachers these days is a most precious commodity. The grouping together of so much information in such an accessible format has to be a real selling point. At present there is nothing like it for spread and specialism alike. Indeed, any educationalist who values resources, new ideas or material or needs to have constant access to them can hardly afford to be without NERIS. Take a look! B

Music Programming In Ample (Part 2)

This month Ian Waugh discusses how to add musical expression to your compositions.

The first step in creating a piece of music in Ample is to 'put the notes in'. In fact, the sense of achievement when you've arranged your first piece is considerable. But when you play it you'll discover that there's a lot more to a musical performance than simply churning out a string of notes in the correct order. Along with your initial sense of achievement you may experience a little disappointment at the lack of musical expression. In this article we'll see how you can to add this to your music.

The reason why a string of 'straight' notes sounds flat compared with a human performance is because human musicians add many types of expression as they play. They may vary the volume, they may slur notes, add vibrato or bend the pitch of a note. Another vitally important factor is articulation. This is the small gap between notes which helps form musical phrases. Without at least some of these features music can indeed sound rather uninteresting, and give credence to the oft-heard comment that computerised music is robotic.

ARTICULATION

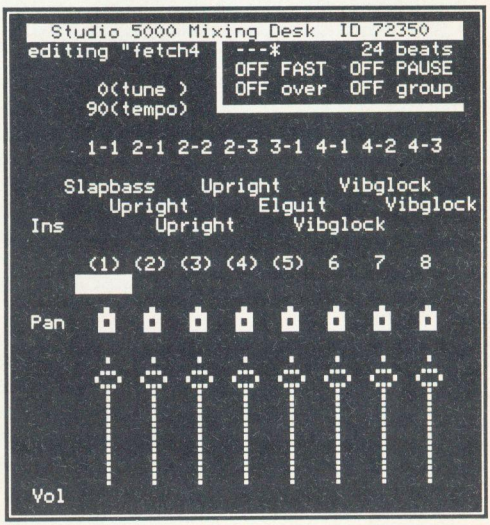
The art of articulation is the art of phrasing. If you think this is only for the musical intelligentsia try this:

```
"part1" [SCORE
8FOR(
0: 48, CCCC 24, DDDD EEEE 48, FFFF
)FOR ^
]
```

Go to the Mixing Desk and play this using different instruments. You would expect to hear four notes of each pitch but it doesn't work with Organ or Moog. This is because these instruments play at a constant volume and there is no volume change to herald the arrival of a new note. Of course, if the new note is a different pitch, then you can hear it distinctly.

This effect is mainly a result of the amplitude envelope used. Give different envelopes to the Organ instrument (use the Notepad in panel mode) and notice the difference they make to the articulation.

But, I hear you say, when an organist plays two consecutive notes of the same pitch they sound separate. That's because the organist has to physically remove his finger from the key and push it down again in order to trigger a second note. The lifting of the key causes a small gap and so the second note is heard.



Editing a mix with the Music 5000

One solution would be to put a tiny rest between each note, but this is both cumbersome and unnecessary. We can achieve the result we want in Ample by creating articulation words using *Len*. With a positive number, *Len* sets the note's gate or 'on' time, that is the time for which the note actually sounds, whatever the note's duration may be. With a negative number, *Len* sets the length of the gap between notes. If this sounds a little complex, a few examples will illustrate the principle. Define the following word:

```
"gap" [ 12 Len ]
```

and insert it in "part1" after SCORE. This will give all the notes a gate time of 12 time units. Alter the value in gap and notice the effect: raising the value increases the note's on time (but see what happens if it's longer than the actual note duration).

Len with a positive value is useful for staccato passages which, to make them easy to read, may be written using notes of different durations. As a rule of thumb, choose a *Len* value which is half that of the shortest note in the passage. Of course, altering the value of *Len* plays the notes with a greater or lesser degree of staccato.

If you don't really want a staccato effect but simply want to separate notes from one another, use a negative setting. Replace the 12 in "gap" with -4. Now all the notes will be truncated by a fixed amount which allows notes of different values to retain (most of) their 'on' time. Again, alter the value and listen to the result. To cancel the effect use a *Len* setting of 0.

SLUR

A slur is on the opposite end of the articulation spectrum to staccato. Strictly, if two notes are joined by a slur you are required to play the first note and sound the second pitch without retriggering the envelope. You cannot, however, slur notes on a keyboard as the only way to play a note is to press a key which automatically triggers the envelope (although some keyboards have a 'slur' mode they are in the minority). The nearest you can get to a keyboard slur is to play the notes smoothly, a technique known as legato. It's interesting to note that although slurs are a vital part of virtually all non-keyboard music, they are not directly supported by MIDI.

The Hybrid Music System interprets slurs in the correct musical fashion, which is generally how you'll want to use them. Notice, too, that you can slur any instrument including piano (Upright), something which is physically impossible to do on a real instrument (or over MIDI). However, using the Hybrid Music 2000 MIDI interface you can slur notes using the MIDIBEND word, a technique explained in the Music 2000 manual so we won't repeat it again here.

Substitute the following for the music line (third line down) in the "part1" definition above and play it using different instruments:

```
0: 48, C D E F F G A B C///
```

Using the Organ, the two F notes are joined together and using Upright the notes are slurred so that you can't hear the last ones. Now define this word (courtesy of Hybrid):

```
"slur"[30 ACT( 1FVAR #? IF(
1FVAR #? 5FVAR #! )IF
ACT )ACT
]
```

Change the first line of "part1" to read:

```
SCORE slur -12 Len
```

Using Upright, it now plays perfectly. The slurred notes are played smoothly (legato), and there is a short gap between the two Fs, and the B and the C, to mark the division between the phrases.

What is happening is this: the negative *Len* makes all the notes staccato, but slur negates this effect on tied notes by redefining the way the gate works. This demonstrates only one use of the powerful ACT command. A detailed look at ACT is beyond the scope of this article and interested parties are referred to the Ample Nucleus Programmer Guide.

VIBRATO

Let's now move on to something a little easier to understand and implement. Many instrumentalists introduce expression to their playing by adding vibrato or pitch bend to notes, often delayed until after the note has been produced. The Hybrid Music System has predefined vibrato, delayed vibrato and pitch bend envelopes and you can use these simply by inserting them in the music at the required places. Enter the following word and set it up to play with the Elguit instrument:

```
"part1" [SCORE
8 FOR(0:
48, C/// Bend F// Delvib -e c/-b/
Bend C/// Delvib
)FOR ^
]
```

Elguit's normal pitch envelope is Delvib but we can make it use Bend simply by writing it into the music. In fact, you can insert any envelope in the same way. Try Drop, Deepvib and Trill, for example. If you use these effects often, create short words holding the envelope names: For example:

```
"dlvb"[Delvib]
"dvb"[Deepvib]
"vb"[Vibrato]
```

ECHO

Not all pieces of music will use all available voices. I tend to think, if you have 'em you should use 'em, and a good way to use spare voices is with the *Echo* command. *Echo* takes two numbers, the first specifies the delay time between echoes, and the second the number of voices to be used.

Using the last "part1" definition, change the second line to read:

```
8 FOR(0: 6 3 Echo
```

Go to the Mixing Desk and play the piece with Elguit. The latest Mixing Desk software release allows you to add extra voices by pressing the Copy key. If yours doesn't do this contact Hybrid for details of how to get a free upgrade (it'll cost you a disc plus p&p) and enter:

```
3 VOICES Elguit
```

Now we can have fun. If you haven't previously experimented with echoes there are several things worth bearing in mind. Each echo uses a separate voice, and these can be adjusted exactly like any other voice. That is you can change their volume, pan position and the instrument they use. To change individual instruments, turn the group function off (press 'g').

An impressive use of echo is to bounce it around the stereo position. A single echo on the opposite side of the stereo field is very effective. To produce a natural echo, reduce the volume of each subsequent echo. Experiment with different delay times too. In pieces containing runs of notes of the same duration, a quiet echo falling inbetween the notes can be very effective. Echo voices can be transposed and detuned, too. Insert the following in "part1" after the *Echo* instruction:

```
1 VOICE 12 TRANS
2 VOICE -12 TRANS
```

You can use Echo to make rounds, too:

```
"part1" [SCORE
8 FOR(0: 192 3 Echo
24, C/D/ E/c/ c/D/ E/c/
E/F/ G/// e/F/ G///
GAgfe/c/ GAgfe/c/
c/g/ C/// C/g/ C///
)FOR ^
]
```

You can increase the number of echoes up to the number of available voices. Although the example may be somewhat trite, try it using Vibglock, Panflute and Upright (you may have to reduce the volume of Vibglock and Upright to balance the quieter Panflute), and pan the parts around the stereo image. The result, I'm sure you'll agree is impressive, especially for such as short program. MAKE the mix (we'll use it in a moment).

It may have occurred to you by now that if you use a delay of 0 the echoes will play at the same time as the main voice. This allows you to play the same music line on two or more voices without even creating another music part. This is particularly useful for doubling music lines to produce an orchestral effect. You may also have wondered what happens if you enter a negative delay number. Well, in true logical, Ample fashion it creates a pre-echo. That is, the echo sounds before the note! Odd? Try it.

AUTOPAN

Autopan is another very useful device which gives you a lot of result for a little effort. It directs the level control (L) to the stereo position. It takes three values - the first determines the time base, the second determines the amount of movement and the third specifies how many time units the change will take place over.

```
"pan" [IF(      % if ON turn Autopan
ON Autopan -3=L %                  ON
48, 7 32 +L     % pan full right
                % over 32 beats
)ELSE(          % else if OFF
OFF Autopan     % turn Autopan OFF
)IF             % end of if
]
```

The crux of the operation is in the third line which sets the time base to 48, pans across 7 positions and takes 32 beats to do so. It's housed in an *IF(...)ELSE(...)IF* construction so you can switch it on or off with ON or OFF commands.

Change the second line of the last "part1" definition to read:

```
8 FOR(0: ON pan 192 3 Echo
```

and insert the following just after the last music line:

```
OFF pan
```

Run this from the Mixing Desk and you'll see the voices pan across the stereo image. Lots of effect for little code.

SUSTAIN

You can mimic the effect of a piano's sustain pedal, again, by using ACT. The good news is, you don't have to know exactly what's happening in order to use the function. Here's a short program to demonstrate the effect:

```
"mem"[GVAR
]
"sus" [ACT( 5FVAR#? IF( 1 mem#+!
8 mem#? #< IF( 1 mem#! )IF
mem#? VOICE! )IF
ACT)ACT
]
 "tune" [24, 2:CagedcagA 8,CDE 24,c
agAA/
]
"part1" [tune ON sus tune ^(^^^^^^^)
]
```

Go to the Mixing Desk and enter:

```
8 VOICES Upright
```

and MAKE the mix. Now type RUN and watch and listen to what happens. The first number on the second line in *sus* (8 in the above example) determines how many voices are used. You can create a reasonable sustain using just two voices, an alternative to Echo, perhaps. Like Autopan, you can control the *sus* word with ON and OFF without knowing exactly what it does.

CHORD ACCOMPANIMENTS

If you look at a piece of modern sheet music you'll see a sequence of chords running above the melody line. Many pieces use a vamp accompaniment based on these chords made up from three or four notes. The obvious way to program this is to do so in linear fashion, but there is a more economical way. All chords of the same type differ only in pitch. That is, the intervals between the notes in C minor and B minor are exactly the same, it is only the actual pitches which are different. We can, therefore, create archetypal major, minor and seventh chords (and any other chord type or even chord progressions) and play them at the required pitch by transposing them.

Although a chord really needs three notes to give it its 'flavour' you can often manage with only two notes which, in search for even further economy, is what I've used here:

```
"maj" [%STAFF
-1:24,CG(C)e(^)G(C)|
]
"min" [%STAFF
-1:24,CG(C)-e(^)G(C)|
]
"dom7" [%STAFF
-1:24,CG(-B)e(^)G(-B)|
]
```

These have been written on the stave to make them easy to examine and alter. We create the actual chord names using the transpose word, '@', like this:

```
"Bm" [-1@ min
]
"Em" [4@ min
]
"F#7" [6@ dom7
]
"Gmaj" [-5@ maj
]
```

To create a backing track you just string the chords together:

```
"part1" [
Bm Bm F#7 F#7 Em Em F#7 F#7
Bm Bm F#7 F#7 Gmaj Gmaj F#7 Bm
]
```

To play this you'll need to reserve two voices:

```
2 VOICES Upright
```

You can create half bars and end bars, too, and, of course, you can make any alterations to the chords that you wish.

Another useful technique for use with accompaniments is to strum them. This is

Continued on page 55

512 Forum

by Robin Burton

Like last month's Forum, both topics this month stem directly from readers' letters. I don't know who was the first discoverer of the trick in the second half of the Forum, but it just goes to show what devious minds some 512 users must have!

Before that, though, a reminder of protocol and good manners seems to be about due after recent events.

LETTERS

I'm still getting a reasonable number of requests for 512 BBCBASIC 9 months or so after the original offer (total now around 350). Let me say before continuing that I'm still happy to supply this software free to BEEBUG members, or to answer queries if I'm able, but only if you obey a few simple, reasonable rules. Clearly not everyone has read the relevant back issues to find out what these are!

For anyone intending to write to me or to request a copy of BBCBASIC I'll recap. I hope the rest of you will bear with me for a moment.

I will supply a copy of Richard Russel's special 512 version of BBCBASIC to BEEBUG members free of charge, provided that you observe the conditions of the offer, which are:

1. You must supply a blank 800K formatted 5.25 inch disc.

2. It's pleasing if you enclose a letter or note to me, but in all cases you must quote your BEEBUG membership number. I do not provide this service to non-members, especially after BBC Acorn User's recent invitation to all and sundry to join in.

3. You must supply suitable self-addressed return packaging, complete with adequate postage in the form of stamps or, from overseas, International Postage reply coupons.

4. You should clearly address the outside of the package you send for my attention, and if appropriate also mark it '512 BBCBASIC'.

I apologise to the rest of you for going through all this again, but it is called for after numerous recent requests. That's the list of 'DOs', here are the 'DON'Ts'.

1. DO NOT just address your package to BEEBUG. Anything intended for me (c/oBEEBUG)or the Forum should be so marked to avoid unnecessary and time wasting extra handling.

2. DO NOT include a request for BBCBASIC with any other letter, order, query or anything else intended for other BEEBUG staff or departments (unless your letter to me is enclosed separately in its own packaging inside the first).

3. DO NOT say 'please debit my Access card/ Barclaycard/BEEBUG etc. account with the cost of postage - it can't be done. You must include suitable postage with your request.

The reason for these rules is simple - I do not work at BEEBUG's premises. I am not a BEEBUG employee, nor should BEEBUG be involved in providing the service beyond forwarding your letters to me. If you write to BEEBUG - write to BEEBUG, if you write to me - write to me care of BEEBUG. Remember the two things are NOT the same!

As I said some months ago, the vast majority of readers behave perfectly, which makes the job

easier for everyone, thank you. However, quite a number of recent requests have caused extra work, extra correspondence and delays by not following these simple rules.

I realise that some of these may have been from readers who hadn't had their 512 very long and hadn't read earlier Forums, but after this issue ignorance is no longer an excuse. If you send me a disc without suitably addressed return post and packaging I shall, from now on, assume it is a gift!

In any case I'm prompted to suggest that, if you are a new convert, either to the 512 or to BEEBUG, you would find it useful and enlightening to read the back issues. 512 Forum has been running now for nearly two years and I try not to repeat previous items if possible (unlike this one unfortunately). If you are a new member and don't have the relevant copies, back issues of BEEBUG can be easily obtained at very reasonable cost. In fact BEEBUG is the only magazine I know where back-issues cost less than new ones! How reasonable can they be?

That's that over with, and I sincerely hope for the last time. Now we can move on to more interesting things.

RUNNING MS-DOS

Yes! the heading is correct! However, you'd better read on before getting too excited. A letter containing the information on this arrived a couple of weeks ago along with a request for 512 BBCBASIC. I therefore take neither credit nor blame for the following and should tell you I haven't tried it, I merely present it in good faith as an item of general interest in more or less the form it was received.

The facts have been publicised on a bulletin board, I don't know which, but before you consider trying it, I must point out that copyright applies to the use of any version of MS-DOS. It might be alright if you personally hold a licence for MS-DOS for your own PC, but in all cases it is your responsibility to check the licence agreement. For most of you the exercise should be regarded as only of academic interest. Even so, I hope you are at least intrigued, as I was. If any of you find you can legally use a copy of MS-DOS here are the steps to take.

The first task is to create a file called 'BOOT.COM', the contents of which are an 'INT 19' instruction. If you have BBCBASIC you can easily do this by using the in-line assembler, or with any other '86 assembler, but I'll explain another way for those of you not happy with either of these methods.

First create the file itself, 'BOOT.COM' by issuing the command line instruction:

```
COPY CON BOOT.COM
```

This is the operation you have probably used before to create a batch file. As soon as the command is issued the current drive should start up and DOS will open the named file for output. You can now type text directly into the file, so this is a way of creating the file. Type a few characters into it, at least six in total but anything will do. The characters don't matter because they will be changed in a moment, but six are needed to provide room for the 'INT 19' instruction and a default stack. Finally press Ctrl-Z followed by Return, and the file will be written to and then closed. You can check that it has by issuing a 'DIR'.

Next you need to change the contents of the file using 'EDBIN'. This is the binary editor which is included on issue disc one. To edit the file ensure that EDBIN is available in a current path and load the file for editing. Some of you will know how to use EDBIN, but for the remainder all you need to do is follow the instructions. Assuming drive A: is the current drive containing EDBIN and drive B: contains the file, the command is:

```
EDBIN B:BOOT.COM
```

EDBIN will execute and confirm that the file is loaded by showing the byte count, followed by the prompt, which appears as a hyphen '-'. If you now enter an 'H' (for help) and press Return you will see all the EDBIN commands

displayed. After you've read this enter an 'E' and press Return so that you can edit the contents of the file.

Your cursor should be under the first hex byte in the file, which is to be changed to 'CD'. This is done by entering first the 'C' then the 'D', because you'll notice that bytes move in from the right as you enter them, pushing the previous contents to the left. You'll also notice that after entry, the cursor doesn't move on to the next byte automatically, you must do this yourself using the cursor right key. In the second byte, the required value is '19', so enter these two digits.

The remaining four bytes of the file should contain hex zeros; if they don't, amend them. When you've finished, leave edit-mode by pressing 'Ctrl-C'. All that now remains is to re-write the file, which is done by entering a 'W' followed by Return. No filename is needed, as EDBIN uses the original name by default. You can now leave EDBIN by entering a 'Q', at which point pressing Return will take you back to the command line.

THE BOOT DISC

The next requirement is to get hold of the copy of MS-DOS or PC-DOS that you intend to try out. This is best achieved by using your PC to prepare a new system disc by formatting it with the '\S' suffix. Again assuming drive A: is to be used for your floppy the command is:

```
FORMAT A:\S
```

This is a similar operation to producing a bootable 640K disc in the 512's 'DISK' command. The disc is first formatted, then at the end the system files (i.e. the operating system) are copied onto it.

Back on the 512 again, copy the BOOT.COM file created earlier onto the new system disc. You're now almost ready to experiment, but before you do, a couple of points for those to whom they may apply. Remember when you're treading new ground, just like trying out any new software, you can minimise risks by a couple of sensible precautions.

If you normally have any special settings, or resident utilities loaded automatically by an AUTOEXEC.BAT file it's best to reboot the 512 and run it 'bare' for this exercise. Equally, if you normally run your 512 from a Winchester it's also good advice to switch the Winchester off and boot and test from floppies until you are sure this works with no ill-effects.

After observing the warnings, all you need to do is to insert your newly created MS-DOS disc into the default drive and type:

```
BOOT
```

followed as usual by Return. The chosen version of MS-DOS should then start up successfully, indicated by a return to the familiar 'A>' prompt.

LIMITATIONS

As usual you don't get anything for nothing; what you're doing is running the MS-DOS kernel within DOS Plus, so both occupy their normal amount of memory. As my correspondent pointed out, you'd only have about 260K free in an unexpanded 512 (90K less than usual) so it isn't much use for running software.

What it might be useful for though, is installing software which normally can't be configured on the 512, but which does run once this process is complete (installation must be done under MS-DOS for some packages). If you have an expanded 512 the reduced memory won't be a problem, of course, so you may even find a few programs can be set up which don't normally run at all. Others that normally run with limitations may perform more correctly with MS-DOS running during installation, particularly if it's a later version than 2.1.

I understand that MS-DOS version 3.2 works, but only from DOS Plus 1.2, while PC-DOS 3.2 doesn't work at all. The bulletin board which originally contained the information also said that both MS and PC-DOS 2.0 work, but I can't add any comment to this.

Next month I'll include more information on some of the packages that have been tried.

Curve Fitting (Part 2)

In the second of his two part series, Sheridan Williams expands the technique of curve fitting to cater for functions other than polynomials.

Whereas a polynomial of order n can be made to fit any set of n+1 points, this does not mean that the set of points is *really* represented by that expression.

X	SIN(X)	X	SIN(X)
-1.0	-0.84	3.8	-0.61
-0.2	-0.20	4.6	-0.99
0.6	0.56	5.4	-0.77
1.4	0.99	6.2	-0.08
2.2	0.81	7.0	0.66
3.0	0.14	7.8	1.00

Table 1

Take for example the values in table 1 (which are in fact real values taken from a sine curve). Figure 1 is the plot obtained from these values using the polynomial fitting program given in part one. However, even using the polynomial of order 10 (a very complex expression), which appears to fit almost exactly, it is not possible to obtain *accurate* extrapolated values *outside* the range. Remember that the routines given here, and in part one, provide a best fit to the points provided, and do not in any way guarantee that the fit is real or accurate outside (or even between) the points.

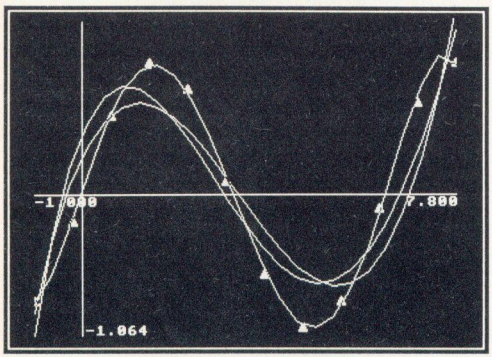


Figure 1. Polynomial fits of order 3 to 5 to a sine wave

Considering that the expression y=SIN(x) fits perfectly it would be silly to use a polynomial of high order as a substitute. So what we need is a program that would generate the function for us. Unfortunately life is not that simple. Unlike polynomials, where one simply keeps trying expressions involving increasingly higher powers of x, with functions there are an infinity of combinations. The program given this month, allows any *two* functions to be combined in an expression.

We immediately hit upon a problem - with an infinity of functions how do we select a suitable one (let alone a combination of two)?

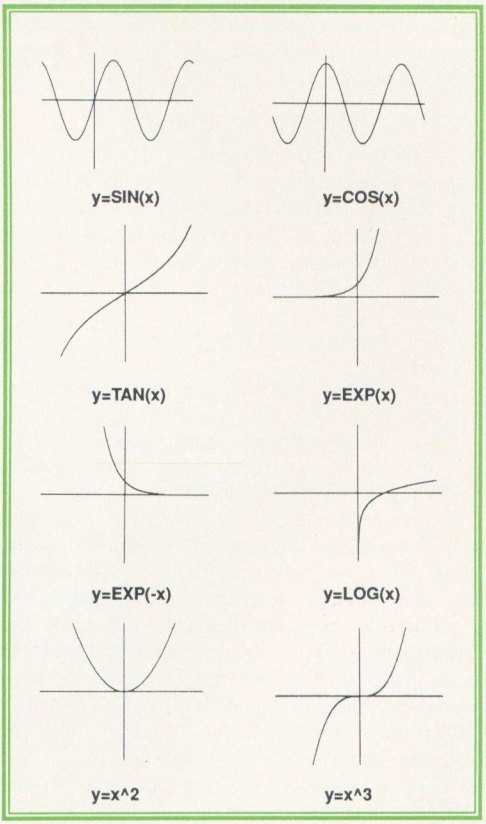


Figure 2. Standard Basic functions

The answer is to look for patterns. Figure 2 shows the shapes for the main functions provided in Basic.

EXAMPLE

Before dismantling a building structure we have taken measurements from part of it as follows:

x	f(x)	x	f(x)
-3.5	33	0.5	2.3
-2.9	18	1.0	3
-2.5	12	1.5	4.7
-1.5	5	2.0	7.5
-1.0	3	2.5	12
0	2	3.5	34

Having dismantled the structure we find that we require measurements when x=3 and when x=5. We will use the program to find the answer. We could simply try last month's polynomial routine, and would end up with a fourth order polynomial that appears to fit reasonably well. However, for reasons that will become apparent, we will try and find a function or functions that fit better.

TYPING IN THE PROGRAM

To save you a lot of unnecessary work, this month's program uses the same procedures and functions as last month's program, so assuming this was saved as "polyfit" and that you have not renumbered it, proceed as follows:

```
LOAD"polyfit"
DELETE 10,980
```

now type in this month's program

```
SAVE"ffit"
```

For convenience the magazine disc contains the whole program, together with the data file (called "caten") used in the example.

USING THE PROGRAM

The program requires data to be entered from the keyboard, or alternatively it can read ready-prepared data from a file. It will also require one or two functions to be input, but if only one function is required, press Return when prompted for the second function. Output can be to the screen or to a printer, and provided you have a screen dump ROM or utility, it can dump the resulting plot to the printer also. At present line 1420 will work with BEEBUG's own Dumpmaster ROM, or you may substitute you own command here.

To use the program in this example you would be best to enter the values into a file using a word processor, because we will be experimenting with many different functions and won't want to keep typing the values in. This month's magazine disc contains the data in a file called "caten".

Run the program, and when prompted, select input from a file. As we haven't yet decided what functions to enter press Return when each function is requested and a straight line will be fitted by default, this will allow you to see a display of the points, and assess the types of function(s) to try. Our measurements are displayed in figure 3.

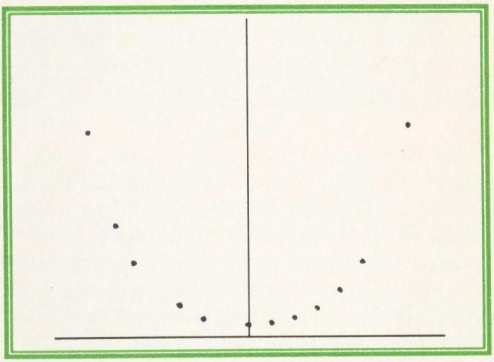


Figure 3. Display of measurements

This looks a little like $y=x^2$, or perhaps part of the curve y=COS(x) shifted vertically. Let's try each of these in turn, responding by pressing Return when the second function is requested so that we can examine one function at a time. To determine whether the fit is good we could look at the plot. However, for a better guide, study the correlation coefficient, which is a measure of the goodness of fit. A function that gives a correlation coefficient of better than 0.99 would probably suffice for most purposes, and our two trial functions give:

COS(x) corr coeff = 0.7869

x^2 corr coeff = 0.9768

neither being sufficiently encouraging. However, a combination of the two gives a correlation of 0.9901, probably sufficient for most purposes, and the calculated function is:

$y = 2.439\ COS(x) + 2.6845\ x^2$

If you recall, the task was to find values when x=3 and x=5. Plugging these into the function give values as follows:

x=3 y=21.75 x=5 y=67.80

THE REALITY

In actual fact the measurements were taken from a hanging chain, and such shapes are known as catenaries. Catenaries can be expressed in the form $y=a.e^{x}+be^{-x}$. With this knowledge we try the two functions EXP(x) and EXP(-x), and lo and behold we get a correlation of 0.9998. The function being:

$y = 1.019e^{x} + 0.993e^{-x}$

This gives values when

x=3 y=20.51 x=5 y=151.18

the latter being substantially different from that obtained with:

$y = 2.439 \cos(x) + 2.6845 x^2$

suggesting that unless you have other knowledge of the function involved, it is probably wise to assume that whatever fit you achieve is only valid within the range of initial values given. Extrapolating outside the values is increasingly dangerous the further you go.

CONCLUSION

Having drawn a graph of the points, with a little practice it is often possible to learn how to combine functions, for example adding exp(x) and exp(-x) gives a catenary, $exp(-x^2)$ gives a bell-shape known as a Normal curve.

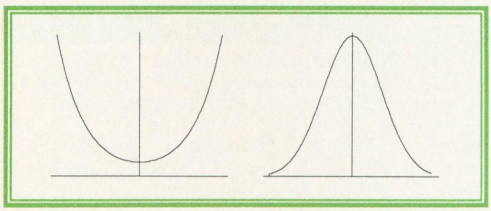


Figure 4. - $y=e^{x}+e^{-x}$ and $y=EXP(-x^2)$

UNDERSTANDING THE MATHEMATICS

The functions we wish to fit are expressed as:

$y = a.f(x) + b.g(x)$

where f(x) and g(x) represent our choice of Basic functions. In the program F1 and F1$ represent f(x), and F2 and F2$ represent g(x). The coefficients a and b can be found provided we have two equations. These equations can be derived by summing the x and y values:

$\Sigma(y.f(x)) = a.\Sigma(f(x).f(x)) + b.\Sigma(f(x).g(x))$

$\Sigma(y.g(x)) = a.\Sigma(f(x).g(x)) + b.\Sigma(g(x).g(x))$

which are evaluated in the program lines 330 to 360, using the variables:

Z is $\Sigma(y.f(x))$ T is $\Sigma(y.g(x))$
U is $\Sigma(f(x).f(x))$ V is $\Sigma(f(x).g(x))$
W is $\Sigma(g(x).g(x))$

B and A are finally evaluated in line 380.

The correlation coefficient is a measure of the "goodness of fit" and in simple terms is determined by the sum of the squares of all the distances of each point from the curve, this is then scaled to lie between 0 and 1, 0 being no fit, 1 being a perfect fit.

We apologise for omitting the last few words from the end of last month's article. This should have read:

...this is then scaled to lie between 0 and 1, 0 being no fit, 1 being a perfect fit.

```
   10 REM >Program ffit
   20 REM Version B1.6
   30 REM Author  Sheridan Williams
   40 REM BEEBUG  May 1990
   50 REM Program subject to copyright
   60 REM:
  100 n=50:DIMXX(n),YY(n),F(n),x(n),y(n)
  110 ffit=TRUE:F1$="":F2$=""
  120 max=1E38:x_min=max:x_max=-max:y_mi
n=max:y_max=-max
  130 height=991:width=1279:negligible=1
E-5:np=25
  140 MODE 4
  150 CLS:ON ERROR GOTO 660
  160 PRINT"Function fitting program"
  170 PRINT"  by Sheridan Williams"
  180 PRINT"-----------------------"
  190 PRINT"Any BASIC expressions may be
 used. The variable, however, must be x.
 Expression may be in upper or lower cas
e."
  200 PRINT"Examples of valid functions"
  210 PRINT"   sin(x)"
  220 PRINT"   2*sin(3*x)+cos(x^2)"
  230 PRINT"   EXP(-(X^2))"
  240 PRINT"   2*x^4+3+x^3-1.5*x2-100"'
  250 PRINT"Two functions are required:"
  260 INPUT"Input  first function ",F1$:
F1$=FNcase(F1$):IF F1$="" F1$="X"
  270 INPUT"Input second function ",F2$:
F2$=FNcase(F2$):IF F2$="" F2$="1"
  280 print=FNyes("Do you want printed o
utput"):PRINT
  290 IF FNyes("Read data from file") PR
OCfile ELSE PROCkey
  300 IF print PRINT"First  function is
";F1$:PRINT"Second function is ";F2$
  310 FOR I=1 TO PA:X=XX(I)
```

```
  320 ON ERROR GOTO 680
  330 F1=EVAL(F1$):F2=EVAL(F2$)
  340 ON ERROR GOTO 660
  350 U=U+F1*F1:V=V+F1*F2:W=W+F2*F2
  360 Z=Z+YY(I)*F1:T=T+YY(I)*F2
  370 NEXT I
  380 B=(V*Z-U*T)/(V*V-U*W):A=(Z-B*V)/U
  390 @%=&0608
  400 PRINT'"Best fit function is"
  410 PRINT ;A;" x ";F1$;
  420 IF B>=0 PRINT;" + ";
  430 PRINT ;B;" x ";F2$
  440 @%=&020509
  450 PRINT'TAB(5)"X";TAB(14);"Y";TAB(19
);"FITTED Y";TAB(30);"DIFF"
  460 PRINT STRING$(39,"-")
  470 FOR I=1 TO PA
  480 X=XX(I)
  490 F(I)=A*EVAL(F1$)+B*EVAL(F2$)
  500 D=YY(I)-F(I)
  510 PRINT XX(I),YY(I),F(I),D
  520 NEXT
  530 S1=0:S2=0:S3=0:S4=0:S5=0:
  540 FOR I=1 TO PA
  550 S1=S1+YY(I):S2=S2+YY(I)*YY(I)
  560 S3=S3+F(I) :S4=S4+F(I)*F(I)
  570 S5=S5+YY(I)*F(I)
  580 NEXT
  590 S6=SQR(S2/PA-S1*S1/PA/PA)
  600 S7=SQR(S4/PA-S3*S3/PA/PA)
  610 CO=S5/PA-S1*S3/PA/PA:CR=CO/S6/S7
  620 PRINT;,"Correlation coeff is ";CR
  630 VDU3:IF FNyes("Plot the data") PRO
Cfplot
  640 @%=&090A:END
  650 :
  660 @%=&090A:VDU3:CLOSE#0:IF ERR=17 EN
D
  670 REPORT:PRINT" @ line ";ERL:END
  680 VDU 3:REPORT
  690 PRINT'"An ERROR has occurred, pro
bably"
  700 PRINT"because you have input the f
unction(s)"
  710 PRINT"incorrectly. They were:-"
  720 PRINT"First  function ";F1$
  730 PRINT"Second function ";F2$
  740 PRINT"Value of x is ";X
  750 PRINT"Alternatively a value might
not exist"
  760 PRINT"for the function, eg LOG(0),
 SQR(-1) etc"
  770 PRINT;"Press 'SPACE' to re-input"
  780 REPEAT UNTIL GET=32
  790 PRINT'
  800 GOTO 150
  810 :
  820 DEF FNcase(a$):LOCAL c$,i,t$:t$=""
  830 FOR i=1 TO LENa$:c$=MID$(a$,i,1)
  840 IF c$>="a" AND c$<="z" c$=CHR$(ASC
c$ AND 223)
  850 t$=t$+c$
  860 NEXT:=t$
  870 :
```

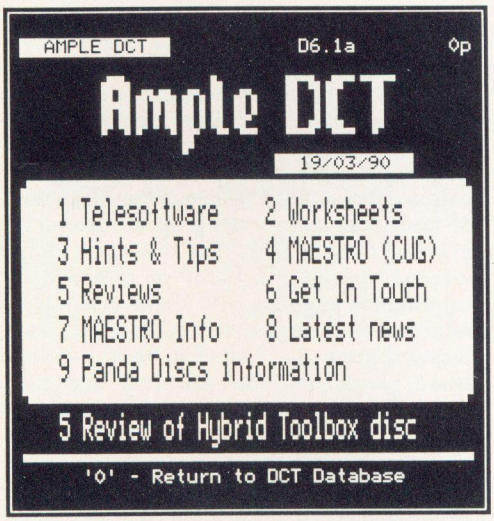
Designer Shoot-'Em-Up (continued from page 24)

```
 2610 CPXbullet:BCCchhL0:LDYy:JMPchhL
 2620 .sk .chhJ LDA#0:STAs1:LDAxpos
 2630 STAx1:LDA#26:STAy1
 2640 LDY#0:.chhL1 LDAbuffer,Y
 2650 BEQchhJJ0:CMP#255:BEQchhJ0a:STAs2
 2660 INY:LDAbuffer,Y:STAx2:INY
 2670 LDAbuffer,Y:STAy2:INY:INY:INY:STYy
 2680 JSRcollision:BCCchhJ1:INCloselife
 2690 RTS:.chhJ1 LDYy:JMPchhL1
 2700 .chhJ0a LDAbomb:BEQsk0:LDY#1
 2710 .chhL2 LDAbombX,Y:STAx2:INY
 2720 LDAbombY,Y:INY:STYy:STAy2:LDA#1
 2730 STAs2:JSRcollision:BCCchhJ2
 2740 INCloselife:RTS:.chhJ2 LDYy
 2750 CPYbomb:BCCchhL2:.sk0 RTS
 2760 .chhJJ0 INY:INY:INY:INY:INY
 2770 JMPchhL1:.chend LDY#0:.cheL
 2780 LDAbuffer,Y:BEQcheJ:CMP#255
 2790 BEQcheJ0:RTS:.cheJ INY:INY:INY
 2800 INY:INY:JMPcheL:.cheJ0 JSRsc100
 2810 JMPstartnew
 2820 .snd1 PHA:TXA:PHA:TYA:PHA
 2830 LDX#so1 MOD256:LDY#so1 DIV256
 2840 LDA#7:JSRosword:PLA:TAY:PLA:TAX
 2850 PLA:RTS:.so1 EQUW1:EQUW1:EQUW17
 2860 EQUW2:.snd2 PHA:TXA:PHA:TYA:PHA
 2870 LDX#so2 MOD256:LDY#so2 DIV256
 2880 LDA#7:JSRosword:PLA:TAY:PLA:TAX
 2890 PLA:RTS:.so2 EQUW0:EQUW2:EQUW4
 2900 EQUW3:]NEXT
 2910 PRINT'"To save code type:""*S.G
AME "+STR$~code+" "+STR$~O%+" "+STR$~gam
e+" "+STR$~start'':END
 2920 :
 2931 DEFFNinkey(A):[OPTpass
 2941 LDY#255:LDX#A+256
 2951 LDA#129:JSRosbyte:TYA:]=pass
 2961 :
 2971 DEFFNxtab320:FORA=0TO31:[OPTpass
 2981 EQUB((A*320)MOD256):]NEXT:=pass
 2991 :
 3001 DEFFNytab320:FORA=0TO31:[OPTpass
 3011 EQUB((A*320)DIV256):]NEXT:=pass
 3021 :
 3031 DEFFNxspr:FORA=0TO21:[OPTpass
 3041 EQUB((&4000+A*64)MOD256):]NEXT
 3051 =pass
 3061 :
 3071 DEFFNyspr:FORA=0TO21:[OPTpass
 3081 EQUB((&4000+A*64)DIV256):]NEXT
 3091 =pass
```

The Comms Spot

Alan Wrigley looks at some bulletin boards offering information for BBC users.

We last covered the subject of bulletin boards in BEEBUG Vol.8 No.3. Since then, of course, new boards will have appeared and old ones faded away, such is the ephemeral nature of many of them. In this article we will take a look at a few of those currently available. All these boards have been chosen because they have useful information for BBC micro users, but there are many others which also cater for Beeb users, not to mention the hundreds of boards around the country which may well contain information of general interest, even though their computing sections cover other machines. Details of times, baud rates and telephone numbers are given in our bulletin board list on page 51.



Part of the Ample DCT section on the DCT1 database

Sadly, most boards these days do in fact concentrate on other machines, such as Amigas, STs, and particularly PCs. If you have a Master 512, then this will be no problem to you, but for the rest of us it can be a frustrating exercise, especially if you are looking for downloadable BBC software.

SID

The main bulletin board for all owners of Acorn computers is, of course, SID (Acorn's own Support Information Database). If you haven't yet explored SID then you really should do so. Although there is naturally a predominance of material for Archimedes machines, the BBC is not forgotten. Acorn is, after all, still producing Master 128s, so this is still a current model.

SID offers a wealth of information, tips, contacts etc., as well as a considerable amount of downloadable software. The menu system is very clear and everything is easily available. There is a comprehensive database containing software for both BBC and Archimedes, as well as product information for both Acorn and third party products. There is also a range of documentation in the form of downloadable text files. A further section, entitled *El Sid*, offers comment and opinion, features on micro-related topics, and software reviews.

Contained within SID are a number of bulletin boards which provide information exchange on specific topics, for example hardware, software, printers, image manipulation and so on. Anyone accessing SID can leave a public message on these boards, which thus provide a useful forum for public debate, or fast access to expert help if you have a problem.

The easiest way to access SID is via the gateway which is provided from Micronet. Just log on to Micronet and type *SID to get through. Alternatively, you can dial direct to SID in Cambridge. All in all, SID is well worth a look. If anything happens in the Acorn world, then SID will know about it!

LONDON CITY MAGAZINE

The *London City Magazine* can best be described as a kind of "Lifestyle" bulletin board, but with the added advantage that it has active BBC and Archimedes user groups. The breadth of topics available dispels the myth that bulletin boards are nothing but a haven for computer freaks.

Alternative Living, for example, contains sections on Yoga, Mind & Body, Vegetarian Recipes and Aromatherapy, among others.

There are several closed user groups (CUGs), including the BBC and Arc groups already mentioned, and also GreenNet, which is an apolitical group concerned with the environment, peace, and rights. Also on the board when I logged on were erudite articles on The Holy Grail, Alternative Schooling, Music for Healing, and Nationalism.

The material to be found on LCM appears to be of a high standard, so if you want to get involved in an exchange of views on some worthwhile topics, then log on to the board and see what's on offer.

DCT

Users of the Hybrid music systems will know of the Ample DCT service on the Dudley College DCT database, which produces the excellent Panda music discs (see BEEBUG Vol.8 No.10 for a review of several of these). There are in fact two bulletin boards, DCT1 and DCT2, covering a wide range of educational topics as well as the music items.

On the music front, there are reviews of music software and hardware, information about Panda products, and music files (of non-copyright music) which can be downloaded. There is also a CUG on DCT2 called Maestro, which provides copyrighted music files. This is available for a small subscription, which goes towards the payments which must be made to the Mechanical Copyright Protection Society for the use of copyrighted music.

There is a range of educational material on both databases, including software for downloading. At the time of compiling this article, DCT2 had a distance learning project for physically handicapped people, while DCT1 had an area called *Cop Shop*, covering information and exercises on the themes of security and common sense, such as Stranger Danger, Home Watch, Self Defence etc. There was also an area containing material for children with special needs.

BEYOND BELIEF

The last board featured in this article is *Beyond Belief*, which has recently undergone an overhaul in order to improve the speed and quality of the service offered. This board features a Micro Magazine, which has software available for downloading, including some for the BBC and Archimedes. Like the DCT board, Beyond Belief caters for Hybrid users and so music files feature among those on the list.

Also on the board are an on-line adventure game, numerous contributed articles on various subjects, for example music and humour, and a wide range of chatlines covering topics such as comms, technology, role-playing games and sport. The system is continually being expanded, and future plans include software and hardware reviews, and weather satellite pictures for Archimedes users.

INTERACTION

All these bulletin boards, in common with almost all others, provide extensive facilities for chatting and messaging. Indeed, the whole concept of bulletin boards is based around interaction between the users. It is all too easy to think of a board as providing a commercial service, whereas in fact they are normally run by enthusiasts who are giving up their time for nothing in order to promote social contact between like-minded people. If you spent hours of your free time setting up an interactive system, only to find that the users simply wanted to grab everything on offer without contributing anything, you would justifiably be annoyed and might well conclude it was a waste of time. So if you log on to a bulletin board, please contribute something, even if it is only a message saying who you are and what you think of the board.

A further point to bear in mind is that a bulletin board may only have one telephone line for access. If the board is worth seeing then it is likely to be popular, so you should not spend hours browsing while others may be trying to contribute something. Most boards, in fact, appear to limit access time for each user to a specified amount within a given period.

I hope to feature more boards in a future article. If you know of any which you think should be included, please let me know! B

Bulletin Boards

The following is a selection of bulletin boards which should contain material of interest to BBC users. The numbers given for the London boards reflect the new dialling codes for London (071 for inner, 081 for outer) which come into effect on May 1st.

There are many others and we hope to give details of some more in a future issue. In the meantime, if you run or access a bulletin board which has BBC material, then please let us know the details.

Board:	**Banat Board**
Location:	London
Tel. number:	(081-783) 1151
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Beyond Belief**
Location:	Middlesbrough
Tel. number:	(0642) 787898
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**Cats Board**
Location:	Maidenhead
Tel. number:	(0628) 824852
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**CCL4**
Location:	Hull
Tel. number:	(0482) 655798
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**Chronos' Lair**
Location:	Birmingham
Tel. number:	(021-744) 5561
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**DCT1**
Location:	Dudley
Tel. number:	(0384) 239944
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**DCT2**
Location:	Dudley
Tel. number:	(0384) 238073
Speeds:	V21/23/23v
Access times:	24 hours

Board:	**Ichthus**
Location:	Reading
Tel. number:	(0734) 461466
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Kernow Opus**
Location:	Truro
Tel. number:	(0209) 821670
Speeds:	V21/23
Access times:	24 hours

Board:	**Lightfingers Place**
Location:	Bournemouth
Tel. number:	(0202) 485723
Speeds:	V21/23
Access times:	24 hours

Board:	**London City Magazine**
Location:	London
Tel. number:	(081-468) 7648
Speeds:	V21/22/22b/23/23v
Access times:	24 hours

Board:	**MABBS**
Location:	Birmingham
Tel. number:	(021-444) 8972
Speeds:	V21/23
Access times:	24 hours

Board:	**MGBBS**
Location:	Pontypridd
Tel. number:	0443 733343
Speed:	V21/23
Access times:	24 hours

Board:	**Sherwood Forest**
Location:	Nottingham
Tel. number:	(0602) 397113
Speeds:	V21/23
Access times:	24 hours

Board:	**SID**
Location:	Cambridge
Tel. number:	(0223) 243642
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**Womble Base**
Location:	London
Tel. number:	(081-979) 9606
Speeds:	V23
Access times:	18.00-23.00

More Essential Software for the 512

Bernard Hill surveys the latest offerings from Essential Software for the 512 co-processor.

Supplier	**Essential Software** **PO Box 5, Groby,** **Leicestershire LE6 0ZB.**
Prices	**Miscellaneous Disc £14.95** **Mouse driver £12.95** **Superstar alone £14.95** **GOBBC/512 alone £19.95** **Superstar + GOBBC/512 £29.90** **Screen print £14.95** **Key translator £14.95** **Screen Save £9.95** **Notepad £9.95** **(all prices inc. VAT and p&p)**

Those of you who use a PC as well as your Beeb will be familiar with the concept of a TSR. A TSR is a "Terminate and Stay Resident" program which is a feature of MS-DOS operating systems. The idea is that the program is loaded (usually doing very little else except announce itself on the screen) and then stops, but staying resident in memory. It is then activated by what has come to be known as a 'hot-key' combination: a selection of keys which are unlikely to be used elsewhere (such as Ctrl-Shift-X). When these keys are pressed and no matter what else is happening, the current computer activity is suspended and the TSR takes over. When it has finished doing whatever it was designed to do then control is passed back to the program which was running before.

TSRs are wonderful things as they are programs which are instantaneously and universally available (no matter what discs are in the drives), but they have one major drawback: they consume memory permanently, even when not running. Essential Software (whose upgraded *Ramdisc* I enthusiastically reviewed in BEEBUG Vol.8 No.5) has come up with a bunch of TSRs for the 512 co-processor, but with one highly important difference: they all reside in the BBC micro and NOT the 512 DOS co-processor, so taking up NO DOS memory at all. In order to achieve this, they all fit in the 4K of RAM left unused in the Beeb itself between the Tube code (&2F00 approximately) and the bottom of mode 3 screen memory (&4000).

The set of routines available form a suite of programs which cram an amazing number of utilities into this space. Residing in the I/O processor also gives them one other big advantage: the hotkeys which start them off cannot be confused with any DOS+ package as of course the Beeb sees the keystrokes first before passing them on to DOS+. The result is that there can be no possible key conflict (as can happen with standard TSRs). Essential has called each of its TSRs 'modules', and has bundled them all together in a very pleasing and homogeneous way. The 'look-and-feel' of the various modules is excellent and quite intuitive in their mode of operation, and there is room for them all to co-exist at once.

So what's on offer? Let's look first at the disc which contains the Print Screen utilities to see how the system fits together.

The ability (on a PC) to press the PrtScrn key and have a copy of the screen dumped to the printer is a facility I have much missed on the 512 (and even the BBC). Snatching a screenshot of a word processor or spreadsheet from the printer to use somewhere else or as reference to another program is very useful indeed, and Essential's disc caters for this superbly. At the DOS+ prompt simply type PRTSCRN 9 or PRTSCRN 24 (9-pin and 24-pin printers) to load the program from the DOS disc down into the Beeb's memory. Nothing else happens until you hotkey with Shift-Ctrl-T, when the current text screen is dumped to the printer. But make sure you're printing at 10 characters to the inch or those fancy box characters you see on the screen won't be lined up nicely because the utility uses graphics printing to emulate a full IBM-compatible printer. And if you prefer 8 lines per inch then Ctrl-Shift-X instead produces this. Simple but very useful.

Also on the disc is *GRDUMP*, which does the same for a graphics image and prints sideways in 11" x 5" in plain mode (Shift-Ctrl-P) or black-white inversion (Shift-Ctrl-I).

To assist with the overall module organisation, there is also on the disc a set of five utilities. While *GRDUMP* loads the graphics dump modules, you have to dismiss them (though I see no reason for doing so) with *TELL GRDUMP DIE*, while *TELL GRDUMP SLEEP* and *TELL GRDUMP WAKE* will suspend or re-activate the graphics dump - maybe because you've turned the printer off?

NEWKEYS GRDUMP will allow you to choose a different hot-key combination. *SAVEKEYS* and *LOADKEYS* enable the saving of these key set-ups to disc, and *STATUS* reports on the presence (awake or comatose) and hot-key triggers for the modules currently installed. *TELL, NEWKEYS, SAVEKEYS, LOADKEYS* and *STATUS* are programs which appear on each disc in the series.

One very disappointing program which came with the 512 was *STAR*. Again Essential has improved many fold on this DOS+ utility. *SUPRSTAR* is a module which when loaded is hot-keyed into action with Shift-Ctrl-* (although I used *NEWKEYS* to change this to Ctrl-Esc), and drops into a mode 7 screen straight from the application you're running. Issuing star commands is easy now, and pressing Escape takes you back to the 512, so because I can hotkey I can print a DFS catalogue of a disc for my son without ever unloading my database in the 512!

SUPRSTAR requires care, however. When in BBC mode, you mustn't do anything which might disturb the Beeb's memory: starting a language is fatal, as is *COPY, *FORM (Acorn's anyway), etc. So in fact you're very limited as to what can be accomplished but I can certainly use my sideways ROM commands which set up my printer styles (see BEEBUG Vol.5 No.3) without even leaving my DOS word processor.

Because *SUPRSTAR* is so limiting, a supplementary disc is available which contains two star commands: *GOBBC and *512. The former should be issued at the *SUPRSTAR* page, when it will save on disc all the important RAM which I mentioned earlier, and then enter Basic, giving a complete BBC environment in which to work. Of course PAGE is much higher (on a model B) at &2500, but you can do ANY operations (including word processing etc.) provided you don't press Break - if you do then of course you've lost your DOS+ session you hotkeyed from and you'll have to reboot. At the end, issue the command *512 and return to the *SUPRSTAR* screen, Escape to DOS and it's exactly as you left it. Amazing!

Another disc is available with a mouse driver. Now I'm not a great lover of mouse interfaces, but with this module installed I can scroll around my spreadsheet at lightning speed. The two buttons on the mouse are programmed to produce keyboard presses of Return and Escape (but can be altered of course), and any movement of the mouse is changed to a series of cursor key strokes. Consequently they work with any package which uses cursor key movements, and with the supplied variable sensitivity program (to change the keystrokes per inch generated) I can now play chess sat back in my armchair with my previously unused 512 mouse!

There is also a module to save DOS screens to DFS or ADFS format, and a *Miscellaneous Disc* contains the modules *COLORDEF* (to change screen colours - I hate the American spelling) and *SUSPEND* (hotkey to suspend, hotkey to resume) as well as standard DOS command-line programs to lock the computer until a password is given, together with *SOUND* and *ENVELOPE* programs to enliven your otherwise silent batch files. Also on the disc is *SELECT*, which is a program enabling you to make a choice within a batch file. Planned future hotkey modules - which Robin Burton assures me will easily be ready by the time this article is in print - include a *Notepad* (using BBC sideways RAM storage), and a key translation program which allows change of key function from anything to anything (model B users note well).

Altogether these are a valuable and useful set of additions, which I commend heartily to you. B

Disc File Identifier Updated

Jim Phillips adds some useful extensions to the Disc File Identifier, WOTAMI, published in BEEBUG Vol.8 Nos.6 & 7, to allow the program to identify files created with View Professional.

Having implemented Alan Mothersole's excellent WOTAMI program, it became apparent that there were two modifications that would enhance its performance with my discs.

On attempting to examine an ADFS disc containing some locked files, the dreaded error "Access violation at line ....." appeared. This was found to be due to the the program's use of the OPENUP command. Changing this to OPENIN proved to be a complete cure.

A number of my files have been created with View Professional and these were shown as Text with rather a strange message starting with the % symbol. Identifying this was clearly the clue to locating them and identifying them correctly.

To identify such files would be nice, but to be able to show the title in the same way that View shows the comment line would be even better. The View Professional file, in fact, starts with an options page construct which is shown as "%OP%parameter value cr". Each parameter is a two letter group and the constructs, when present, appear in the same order as the presentation on the options page. Happily the title is the first option and is denoted by the letters DE. By testing for the string "%OP%DE" the title can be located and then extracted in a similar manner to the comment line in View.

The program listed here should be typed in with the line numbers exactly as shown and both saved to disc as *WOTprf* and spooled as *TEMP* (use *SPOOL TEMP followed by LIST and then *SPOOL). The original WOTAMI program should be renamed as, say, *WOTAMI1*, and then loaded. The additions can then be appended by typing:

```
*EXEC TEMP
```

and the updated program saved as *WOTAMI*.

The main additional procedures are PROCcVprof, PROCrVprof and PROCwVprof, and a further procedure PROCcparam is employed to verify the presence of the title parameters and extract the title text.

With these additions you should find that WOTAMI will now correctly identify any View Professional files on your discs, and show the title line if present.

A further useful modification to WOTAMI is to alter two lines of the program so that, when reading file information using the WOTAMI option, the screen display is paged to prevent the information from scrolling before it can be read. To achieve this, the following two lines should be altered to:

```
6960 IF prt THEN VDU2 ELSE VDU14
7030 IF prt THEN VDU3 ELSE VDU15
```

```
   10 REM Program WOTAMI
   20 REM Version 1.2
   29 REM >>View Professional module ins
talled<<
   30 REM Author    Alan Mothersole
   35 REM Additions Jim Phillips
   40 REM BEEBUG    May 1990
   50 REM Program subject to copyright
   60 :
 1050 DIM name$(47),load(47),ex(47),leng
th(47),lock$(47),type%(47),type$(9)
 1070 FOR I%=1 TO 9:type$(I%)=z$:type$(I
%)="":NEXT
 1090 RESTORE 10000:FOR I%=1TO9:READ typ
e$(I%):NEXT
 1200 ba%=0:ro%=0:vi%=0:sh%=0:st%=0:vp%=
0:tx%=0:mc%=0:di%=0:prt=FALSE:dlen%=0:tk
%=0:free%=0
 1580 b=18
 1600 PRINTTAB(2,b)CHR$134;type$(I%);TAB
(10,b)":";CHR$131;TAB(22,b)CHR$134;type$
(I%+1);TAB(30,b)":";CHR$131;
 1625 PRINTTAB(2,b)CHR$134;type$(I%);TAB
(10,b)":";
 1630 PRINT;TAB(13,18)ba%;TAB(33,18)ro%;
TAB(13,19)vi%;TAB(33,19)sh%;TAB(13,20)st
%;TAB(33,20)vp%;TAB(13,21)tx%;TAB(33,21)
mc%;TAB(13,22)di%
 2460 IF type%(I%)=9 ENDPROC
 2480 C%=OPENIN(name$(I%))
 2570 IF type%(I%)=9 ENDPROC
 2620 IF type%(I%)=9 ENDPROC
 2630 C%=OPENIN(name$(I%))
 2740 IF type%(I%)=9 ENDPROC
 2770 IF load(I%)=0 AND ex(I%)=-1 THEN t
ype%(I%)=7:tx%=tx%+1
```

```
 2830 C%=OPENIN(name$(I%))
 2960 C%=OPENIN(name$(I%))
 3370 IF prt THEN VDU2
 3795 IF type%(I%)=0 PROCcVPROF
 3810 IF type%(I%)=0 THEN type%(I%)=8:mc
%=mc%+1
 5300 IF (USR(&FFDD) AND &FF)=2 di%=di%+
1:type%(I%)=9
 6020 C%=OPENIN(name$(I%))
 6370 C%=OPENIN(name$(I%))
 6600 C%=OPENIN(name$(I%))
 6720 IF type%(I%)<>7 ENDPROC
 6740 C%=OPENIN(name$(I%))
 6890 FOR T%=1 TO 7:READ A$
 6940 UNTIL A=97 ORA=98 ORA=112 ORA=114
ORA=116 ORA=118 ORA=120
 7005 IF A=112PROCwVprof
 7020 IF A=97 PROCwBasic:PRINT':PROCwROM
:PRINT':PROCwView:PRINT':PROCwVprof:PRIN
T':PROCwText
 7390 DEF PROCcVPROF
 7400 IF type%(I%)=9 ENDPROC
 7410 C%=OPENIN(name$(I%))
 7420 data=BGET#C%
 7430 IF data<>&25 CLOSE#C%:ENDPROC
 7440 type%(I%)=6
 7450 CLOSE#C%
 7460 vp%=vp%+1
 7470 ENDPROC
 7480 :
 7490 DEF PROCrVprof
 7500 IF type%(I%)<>6 ENDPROC
 7510 co$=""
 7520 C%=OPENIN(name$(I%))
 7530 PROCcparam
 7540 CLOSE#C%
 7550 PRINT"Filename : ";name$(I%);TAB(2
4)"Length : ";length(I%)
 7560 PRINT"TITLE :":PRINTco$:PRINT:PROC
line:PRINT
 7570 ENDPROC
 7580 :
 7590 DEF PROCcparam
 7600 LOCAL exit%:exit%=FALSE
 7610 FOR psn=1 TO 6
 7620 data=BGET#C%
 7630 IF data<> ASC(MID$("%OP%DE",psn,1)
) Exit%=TRUE:psn=6
 7640 NEXT:IF exit% ENDPROC
 7650 REPEAT:data=BGET#C%
 7660 co$=co$+CHR$(data)
 7670 UNTIL data=&0D
 7680 ENDPROC
 7690 :
 7700 DEF PROCwVprof
 7710 IF vp%=0 THEN ENDPROC
 7720 PROCline2:PRINTTAB(15)"VPROF. FILE
S"
 7730 PROCline2:PRINT
 7740 FOR I%=0 TO nof%-1:PROCrVprof:NEXT
 7750 ENDPROC
 7760 :
10000 DATA Basic,ROM,View,VSheet,VStore,
Vprof,Text,Data,<DIR>
10035 DATA (A)ll,(B)asic,(R)om,(V)iew,v(
P)rof,(T)ext,e(X)it
```

Music Programming In Ample (continued from page 41)

accomplished by including a length setting in the chord bracket like this:

```
"Bm" [-1: B(4,DFB)
]
```

If you want to experiment with different strum delay times, you can use a variable. Let's call it *st* for strum:

```
"st"[GVAR
]
```

Then include it in the chord brackets:

```
"Bm" [-1: B(st #? , DFB)
]
"Em" [-1: E(st #? , GBE)
]
"F#7" [-1: F(st #? , +ACE)
]
```

#? is used to retrieve a variable's value so:

```
 st #? NOUT
```

will print the current value of st. Here's the accompaniment section:

```
"part1" [4 VOICES Upright
K(+F+C)K 48,
Bm Bm F#7 F#7 Em Em F#7 F#7 Bm
]
```

The voices have been allocated here simply so you can type it in and RUN it. Assign a value to the variable:

```
 8 st #!
```

This is Ample's equivalent of Basic's st=8. Now type RUN. You can set different strum delays by altering the value of *st* rather than altering each of the chord words.

If you are interested in hearing just how much expression can be squeezed out of the Hybrid Music System through the use of some of these techniques, I can thoroughly recommend John Bartlett's Jazz Discs Volumes One and Two (£3.50 each) available from JB Software, 20 Crawley Avenue, Wellingborough, Northants NN8 3YH. Examination of the pieces on these discs will prove very rewarding.

In the next article we'll look at programming techniques such as designing menus, chaining programs and how to manipulate the Editors from within a program.

EDIKIT ROM

(incorporating Basic Booster)

An indispensible utility ROM for all Basic programmers for the Master, BBC B and B+ (all with sideways RAM). It contains eleven commands to help you with the editing and development of Basic programs:

***FTEXT** (find text) , ***FBASIC** (find Basic), ***FPROCFN** (find procedure/function) - three FIND commands, designed to help locate lines of programs according to their contents.

***LFROM** (list eight lines of a program), ***LPROC** (list procedure), ***LFN** (list function) - three commands, which list out significant segments of a program.

***RTEXT** (replace text) and ***RBASIC** (replace Basic) are find and replace commands directly analogous to FTEXT and FBASIC.

***SYSINF** (system information) - gives background information on the system variables and their sizes, together with the size of your program

***VARLIST** (list program variables) - lists variable names

***FKDEFS** (function key definitions) - prints out current function keys definitions.

```
>*Help Edikit

EdiKit 1.10

  FBASIC <element>
  FTEXT /<string>/
  FPROCFN
  LFROM <element>
  LPROC <procname>
  LFN <fnname>
  RBASIC <element>/<element>
  RTEXT /<string>/<string>/
  VARLST [%][[<init letter>]
  SYSINF
  FKDEFS
  RENUMBER [<first>] [<last>] [<start>] [<inc>] [F]
  CHECKORDER [<first>] [<last>]
  BRMOVE [<first>] [<last>] [<destination>]
  BRCOPY [<first>] [<last>] [<destination>]
  SQUEEZE
  SMARTNUMBER [<proc base>] [<gap>]
  BLIST <filename>
  TEXTSAVE <filename>
  TEXTLOAD <filename>
>
```

```
3570LDA #ipbuff  S256:STA pptr
3580LDA #ipbuff e256:STA pptr+1:RTS
3590.reinitbptr
3600LDY#3:LDA(bptr),Y
3610CLC:ADCbptr:STAbptr
3610CLC:ADCbptr:STAbptr
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3840LDY#1:LDA(bptr),Y:STA hi
3850INY:LDA(bptr),Y:STA lo
4090JSR inittptr:LDY#0
4110LDA(tptr),Y:BMI tokenfnd
4190CLC:ADC tptr:STA tptr
4190CLC:ADC tptr:STA tptr
4200LDA#0:ADC tptr+1:STA tptr+1
4200LDA#0:ADC tptr+1:STA tptr+1
4240.inittptr
4250LDA#ttable  S256:STA tptr
4260LDA#ttable e256:STA tptr+1:RTS
4340LDA(bptr),Y:CMP#nl:BEQ dtexit
4430.movetoknloop LDA(tptr),Y
4480INY:LDA(bptr),Y:ASL A:ASL A
```

Including the updated Basic Booster utilities:

SUPER SQUEEZE - A program compressor.

PARTIAL RENUMBER - A very useful utility which renumbers a selected block of lines.

PROGRAM LISTER - List any program direct from a file.

RESEQUENCER - Rearrange the lines in a Basic program - line numbering is automatically adjusted.

SMART RENUMBER - Renumber a program so that procedures start at a particular line number.

TEXTLOAD AND TEXTSAVE Save and load a Basic program as text.

EDIKIT (including Basic Booster) is available on:

		Members	Non-members	*Upgrade:	Members	Non-members
3.5" ADFS disc	Stock Code 1452a	£5.75	£15.00	Stock Code 1455a	£4.75	£6.33
40/80T DFS 5 .25"" disc	Stock Code 1450a	£5.75	£15.00	Stock Code 1453a	£4.75	£6.33
EPROM	Stock Code 1451a	£7.75	£18.00	Stock Code 1454a	£6.75	£9.00

*** If you have previously purchased Basic Booster, return your original disc or EPROM to obtain an upgrade at the reduced price.**

Please add 60p p&p (UK only). For overseas check the Membership page in this magazine.

Phone your order now on (0727) 40303

or send your cheque/postal order to the address below. Please quote your **name** and **membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

TO JUSTIFY OR NOT TO JUSTIFY

Mike Williams

Print formatting always seems to be something of a hit or miss affair even when you believe you understand it fully. The use of the semicolons is a case in point. Consider the following three lines:

```
100 val%=345:txt$="test"
110 PRINTTAB(0,4)"01234567890123456789
0123456789"
120 PRINTTAB(5,5)txt$TAB(15,5)val%
```

The string "test" will be left justified starting in character position 5, while the number 345 will be right justified in a field of 10 characters (the default) which commences in position 15. Line 110 serves no purpose other than to allow the alignment of the following PRINT statement to be checked precisely. However, if you replace line 120 with:

```
120PRINTTAB(5,6)txt$;TAB(15,6)val%
```

you will find that the number 345 is now left justified within the same field. It would appear that the inclusion of a semicolon *anywhere* in the PRINT statement will affect the justification of any numbers which follow. The only solution which appears to work is to avoid *any* semicolons appearing before the variable whose value is to be printed, even though this may produce a less readable line of Basic.

MORE NIBBLES TO BYTES

F.Beach

With reference to the hint in BEEBUG Vol.8 No.9 on converting the high or low nibble (4 bits) in the accumulator to a byte, there are some little known OS calls which may be useful when dealing with nibble, one of which will print the least significant nibble from the accumulator. These were fully documented in BEEBUG Vol.5 No.9, but the most useful calls are reproduced below (note that different entry points are required for different machines):

B	B+	Mast.	Comp.	Function
&F975	&F97E	-	-	Print space between nibbles
&F97A	&F983	&BDC6	&BCCC	Print accumulator in hex
&F983	&F98C	&BD75	&BCD5	Print LS nibble

Thus writing (on a Master):

```
A%=&AB
CALL &BD75
```

will result in the hex digit 'B' being printed. A rotate right (by 4 bits - ROR A four times) followed by the same call repeated would print the other nibble.

ASSEMBLY AT &8000

D.J.Halliday

When developing large machine code programs it can be useful to set the variable 'P%' to &8000 (sideways RAM) which will then leave all of user RAM free for the assembler program and variables. This is particularly useful when developing a ROM image containing a number of utilities as they can be tested as they are written.

Even if the machine code is intended to run in main memory, the extra space provided by sideways RAM may mean that you can dispense with compilers and source files. If you have Basic II or later just set O% to &8000, assemble the code, and download to P% before running (OSRDRM, an OS call documented in the Advanced User Guide is ideal for a paged ROM downloader).

Don't worry if the output to the screen looks like rubbish, the code will be correctly assembled. This output is actually a dump of the Basic ROM which the assembler assumes is the destination of the code.

HARD COPY CATALOGUE IN WORDWISE

Leslie Gardner

There is an alternative, and shorter, way to obtain a hard copy disc catalogue while using Wordwise than that published in *Hints & Tips* in BEEBUG Vol.8 No.9. From Wordwise Plus *edit* mode, press:

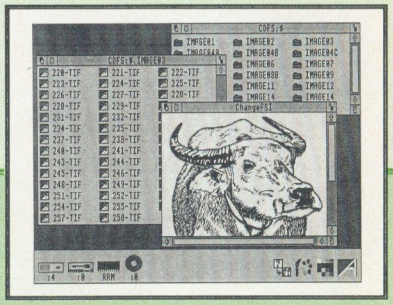
```
f1
*.
f2
```

and then use menu option 6. This will first display the disc catalogue on the screen and then print it. This works not only with the DFS but also with the ADFS, and in condensed mode the printout fits nicely into a disc envelope when trimmed.

RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.
A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.



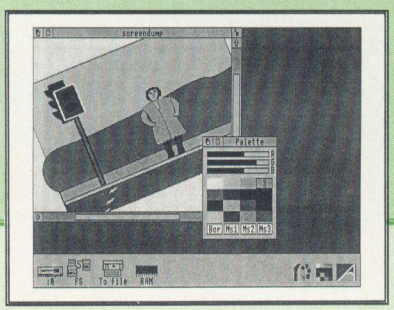
Here are just some of the topics covered in the most recent issues of RISC User:

NEXT TECHNOLOGY'S CD ROM FOR THE ARCHIMEDES
A preview of the latest innovation for the Archimedes.

BANK SWITCHING
Three programs demonstrating different uses of bank switching - animation, switching screens between alternative displays and debugging.

USING DRAW FILES IN BASIC
A library of routines to display and manipulate Draw files, and a program to demonstrate the translation, scaling and rotation of pictures.

GENESIS
A review of this impressive software package from Software Solutions for generating multi-media information systems.



ARC TO PSION
The first of two articles on using the Arc in conjunction with small portable computers.

A DESKTOP CMOS RAM EDITOR
An easy to use Desktop CMOS RAM editor.

ASSEMBLER WORKSHOP
A major series for the more advanced ARM processor programmer. The latest one offers a routine for sorting Basic arrays using ARM assembler.

MASTERING THE WIMP
A major series for beginners to the Wimp programming environment. In the latest issue - Icons: the groundwork,

INTO THE ARC
A regular series for beginners. Te latest article is - The palette and the task manager.

UNDER THE LID
A major series explaining the hardware that makes up the Archimedes.

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.10 (overseas see below).

Don't delay!
Phone your instructions now on (0727) 40303

Or, send your cheque/postal order to the address below. Please quote your **name and membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.10
Rest of Europe and Eire	£12.00
Middle East	£14.00
Americas and Africa	£15.00
Elsewhere	£17.00

RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 60263

MORE USES FOR EXTENDED CHARACTERS

I was very interested in the article and program by Lance Vick in the March issue of BEEBUG (Vol.8 No.9) on printing characters 128 to 255. I have been using these characters as additional "user defined" keys by pressing the function keys in conjunction with Shift and Ctrl. The extended characters produced will, if introduced into a program listing, give a further 20 user defined strings in the computer as follows:

```
Shift-f0 AND,   f1 DIV,   f2 EOR,   f3 MOD,
     -f4 OR,    f5 ERROR, f6 LINE,  f7 OFF,
     -f8 STEP,  f9 SPC

Ctrl -f0 PAGE,  f1 TIME,  f2 LOMEM,
     -f3 HIMEM, f4 ABS,   f5 ACS,   f6 ADVAL,
     -f7 ASC,   f8 ASN,   f9 ATN
```

You will find that, after entering your program, if it is listed, the characters will transform as shown above.

J.W.R.Eaton

PUTTING THE HEAT ON

Your news item in BEEBUG Vol.8 No.8 about D.J.Holden's *Central Heating Program* was most welcome. I have tried it out and I am most enthusiastic about it. It is simple to use, very elegant and incredibly cheap. Mr.Holden deserves credit for what he has done here.

If there are structural engineers in our membership who can offer similar programs for simple beam design etc., I for one would be most interested.

Duncan Horne

WHY NOT FLAUNT IT?

Firstly, may I thank you for maintaining the standard of BEEBUG magazine. I am particularly pleased with issues 8 & 9 of Vol.8, and more specifically with the *Master Display ROM* from Derek Gibbons. In the accompanying comment, Mr.Gibbons states "If you've got it, why not flaunt it!" I concluded that if we were to "flaunt" it, then why not also announce who owns the precious thing (as a crime prevention measure), particularly if the image could be blown onto EPROM and fitted internally, or even soldered into place.

POSTBAG

Here are the additional lines needed, with outline data which you can replace with your own name and address as required:

```
1581 LDY#255
1582 .owner
1583 INY
1584 LDA id,Y
1585 JSR osasci
1586 CMP #0
1587 BNE owner

1641 .id EQUS "OWNER:":EQUB13:EQUB13
1642 EQUS "Ivor Master":EQUB13
1643 EQUS "128 ram Bank":EQUB13
1644 EQUS "Mega":EQUB13
1645 EQUS "Hertz":EQUB13
1646 EQUS "AC1 2DC":EQUB13:EQUB13
1647 EQUS "Tel: 012-345-6789":EQUB13:EQUB13
1648 EQUB 0
```

Insert these new lines into the source listing of MDR2. The only restriction on the text is that it should not exceed 256 bytes in total.

G.W.Hetherington

CHAOS EXCITES

This is just to let you know how excited I was regarding your publication of *Order out of Chaos* in BEEBUG Vol.8 No.9. Ever since obtaining a copy of James Gleick's book I have been wondering whether it would be possible to produce a bifurcation diagram on a Beeb. Jim Vernon's simulation certainly proves that it is.

I cannot, though, really agree with the remarks you make about too much boring biographical detail in Gleick's book *CHAOS: Making a New Science*. To my way of thinking such details lighten it no end, and certainly help to flesh out the personalities involved.

Phil Dyer

PUTTING THE SQUEEZE ON

Regarding the problem with the Aces High program on a model B as a result of insufficient memory (see BEEBUG Postbag Vol.8 No.6), I have found a much easier solution which is to use the *SQUEEZE command in your BEEBUG Basic Booster ROM. This reduces the space used by the program so that it will operate in the original mode.

R.Cooksey

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, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

MIDI Music Sequencer pack for B, B+ and Master, comprises firmware ROMs, hardware interface (1 in, 4 out) and full documentation, 16 track package with 127 patterns, stave editing and printing, realtime or step-time input and full MIDI parameter editing, bargain £200. Tel. (0438) 813577.

512 upgrade for M128 complete with DOS 2.1 mouse and software £155. SMART cartridge £15, plus postage. Tel. (0778) 346287.

WANTED: Solidisk 2 meg 128k board for BBC B with documentation and software discs. Tel. (0532) 752792 eves or wk/ends.

Spectrum 48k computer with Low Profile Professional Keyboard, Interface 1 ,3 Micro-drives, Kempston Centronics Interface, E, Joystick Interface, joystick, many utility prog's, magazines, tapes, 40+ cartridges etc. Cost £450, Offers please. Tel. (0928) 566682.

Acorn Z80 second processor CP/M, handbooks and software, boxed £55, Wordwise plus and handbook £10, STL ADFS disc operating system, requires 8271 chip for DFS £15, STL 32k sideways RAM board £10, Micro Draw Graph II Pad & software boxed £15, AMX Design & Super Art mouse, software for Master only boxed £20, Viglen case and remote keyboard case for model B takes two slim disc drives £15, or offers? Tel. 01-804 7458.

WANTED: BB magazines vol. 1 issues 1,2,3,4,5 and 8. Tel. (0943) 74213.

WANTED: Epson FX80 mechanically beyond repair but electronically serviceable. Tel. (0463) 235199.

Master 128 still boxed, Philips monitor, Watford Video Digitiser (new), Intersheet, Interword and Interbase (still boxed & manuals), View, Viewsheet, Viewspell, Viewstore (all manuals), Stop Press, Super Art (mouse & manuals), Clipboard Graphic Library Vols 1&2, View Dabhand & disc, View Printer Driver Generator, View Index, View Professional, Solidisk 256k expansion board, ADFS ROM + other misc ROMs. Offers? Tel. (0293) 542288 eves & wk/ends.

View Professional £40, BEEBUG C £25, Exmon II (latest version, Master compatible) £12, Elite disc £5, or all for £60. Dumb terminal, ADM 3A Lear Siegler (mono display, keyboard, RS232) £30. Tel. 01-698 3772.

Microvitec colour monitor model 1431 (14") TTL RGB, metal cabinet, little used £100. Tel. (05395) 32657.

WANTED: ATPL board for BBC B+. **FOR SALE:** Interbase ROM in original packaging,hardly used, good condition £35, Tel. 01-555 9069.

Watford ROM/RAM board with 32k sideways RAM (not fitted) 16k static RAM and battery, inc. manual and utilities disc £25 o.n.o. Tel. (0525) 715013.

Acorn Teletext adaptor and ATS ROM £40, 2 Peartree ROM cartridges (Master) £4 each, Spectravision and Acorn joysticks £10, ROMs; Disc Doctor, Dot Print NLQ, Advanced Disc Investigator, Printmaster, Viewspell £10 each. Books for BBC B: Advanced User Guide £4, Advanced Disc User Guide £4, Advanced Basic ROM User Guide £4, Creative Assembler, Griffiths £2, Creative Graphics, Cownie £1, Assembly Language, Birnbaum £4. Software: Glentop 3D Graphics Development System £15, Viewplot £5. Tel. (0306) 889647 eves.

Master 128, twin 80T drives, Acorn Teletext adaptor, Supervision II colour monitor, Vine Overlay board all for £515. Also available 80186 board (without mouse) & Shibumi Problem Solver disc for £65, disc & ROM software including INTER range, Spellmaster, Publisher, Quest, Wapping Editor and BEEBUG Master ROM all at reasonable prices. Tel. (0452) 506361.

BBC Master Compact, mono monitor, Viewsheet, Viewstore and Publisher, DTP ROMs, utility software, games, books, magazines £350 including delivery. Tel. (0436) 71036.

M512 80186 co-processor, mouse, Gem, DOS+ £85, ADE+ 65C02 Assembler, in Master cartridge £25, Wordwise & Spellmaster in Care cartridge £30, Solidisk EPROM programmer UVIPROM 1.0 £15, Solidisk 32k SWR for BEEB £5. Tel. (0375) 858607.

BBC model B software; ROMs - Mega 3 £40, Disc Doctor, Toolkit Plus £10 each. Discs; 5.25" Interactive 3D, Paintbox II, Discmaster, Hobbit, Castle Quest, Steve Davis Snooker, £5 each. Tapes; Eric the Viking, Mini Office, Fortress, Cylon Attack, Hunchback, Frak, Pinball, Starship Command £2 each. Twin analogue joysticks £8, Books; Advanced User Guide £8, BBC Machine Code Portfolio £5. Discount for bulk. Tel. (0246) 569 406.

6502 second processor & Hi-basic ROMs, unused, boxed, unwanted gift £75 o.n.o. Intersheet unused, unwanted gift £15. Tel. 01-886 6475.

Commodore 64 plus disc drive, MPS1000 printer, software - send for list to Rowland Warburton, 124 Oaklands Avenue, Oxhey, Watford, Herts, WD1 4LQ or Tel. (0923) 34800.

BBC add-ons: 40/80 disc drives, colour monitor, printer, modem+command and some software £350 the lot - would split. Tel. 01-428 2846 eves & wk/ends.

Watford solderless ROM board, little used £22, disc drive 40/80T D/S £60, Mini Office II 5.25" disc 80T £10, BEEBUG Quickcalc 5.25" disc 40/80T £8, all genuine, boxed. Tel. (0642) 766135 eves.

Internal Master Modem c/w ROM etc. £65, Panasonic KX-P1081 £70. Tel. (0428) 713326 eves only.

WANTED URGENTLY: Boulderdash disc for BBC Master 128 any reasonable price. Tel. 061-904 0760 after 5pm.

Single Mitsubishi 80T D/S D/D (5.25") for sale in unmarked condition, asking only £60, original printmaster ROM & book, Raven 20K shadow board, c/w ROM & instructions, Assembly Language Programming (book), second edition, offers? Tel. 01-253 4399 extn. 3275 or eves (0487) 814227.

Nidd Valley Digimouse (AMX comp.) inc. Chauffeur and driver £30 o.n.o. System Delta rel. database inc manual £35 o.n.o. Dumpmaster II ROM £15 o.n.o. MOS+ disc £6 o.n.o. also available AMX 3DZicon, BEEBUG Superplot tape, Gemini Life and Business Organiser. Tel. (045527) 4018 eves.

Printer; Acorn AP-100A £40 o.n.o. Tel. (049481) 5448.

Morley 2Mb RAM disc plus Vu-Fax software, instant loading and saving - brilliant! cost £400 accept £200 o.n.o. Also Morley Advanced Teletext Adaptor £40, Red boxes starter set £50, plus more hardware, over 100 ROMs and software packages - games, music, utilities - and over 70 books. Most prices between 50p and £5! for details Tel. 091-529 4788 or send an SAE for a 6 page list to 26 Newark Drive, Whitburn, Sunderland, Tyne & Wear SR6 7DF.

WANTED: dip switch codes or complete handbook for Juki 6100 printer. Tel. (0482) 659938.

Archimedes 310 base, RISC OS and extras £600. Tel. (0271) 850355.

The BBC Micro an expert Guide, Advanced Programming Techniques for the BBC, Advanced Sideways RAM User Guide, Discovering BBC machine Code, Teach Yourself Assembler 650, 6502 Games, Programming the 6502, Guide to the BBC ROMs, 6502 Reference Guide, Assembler Routine for the 6502, Basic ROM User Guide, current value of books £85 offers around £25. Graphpad for BBC £25, All Fingers (touch typing tutor, cass.) £5, Cumana Touch Pad £15. Tel. (0442) 826079 after 7pm.

BBC Master, Pace double disc drive, Panasonic Dot matrix printer, Trolley, extras, 2 yrs old hardly used, cost £1100 accept £700 o.n.o. Tel. (0243) 374707.

WANTED: Advanced machine code techniques for the BBC micro, authors AP&DJ Stephenson, published by Granada. Tel. (0270) 625691.

Master 512 board & software, offers? or swap for Hi-res. colour monitor or twin 3.5" drive. Tel. (0707) 45761.

Hard disc for Master 20Mb £200, software, Interword, Wordwise+, View Professional, Viewstore, Masterfile etc. Tel. (0206) 47852 for complete list and prices.

14" Microvitec Cub monitor £100, Wordwise Plus ROM, reference manuals and 2 books; Wordwise Users Guide and Wordwise handbook & disc of programs £30, Sleuth ROM (for model B) £10, D/S 40T D/D powered from computer £35. Postage extra. Tel. 051-424 0160.

Archimedes 310 base, RISC OS, Watford 4-slot backplane and fan, original packing and manuals £575. Tel. 01-882 2592.

Watford Elec. 32k Shadow RAM board (cost £68) £30, also two 6264 LP-15 RAM chips (cost £4 each) £5 the pair. Tel. (0772) 635346 most eves and some wk/ends.

BBC issue 7, Watford DFS, 40/80T & 80T drives, 64k ROM/RAM, AMX mouse + Super Art, Teletext, Wordwise Plus, Spellcheck and Toolkit ROMs, Office Master and Mate, Joysticks, all leads and manuals, many games inc. Elite, Repton 3 etc. Lots of software, discs, books and magazines £450. Tel. (0483) 224113.

Master 128 £290, Master cartridge £5, Reference manuals 1&2 £6 each. All excellent condition. Tel. (0525) 210551.

The Last Ninja Karate game £5, Bullseye quiz game on cass. £2, BBC Master 512 Shareware Vol. 1 unused £20, 1987 back issues of MU, 1988 back issues of AU £10 the lot. Tel. (0326) 240734 after 6pm.

BBC B issue 7, Solidisk 128 board, 8271/1770 disc controller, 1x 40/80T drive, 1x 80T drive £300. Tel. (0384) 375180.

Acorn Master cartridges £6 each, original double Acorn joysticks £11, ACP Toolkit ROM £20, new Master Ref. manuals 1&2 £10 each, BBC Hobbit adventure game and guide £7.50. Tel. 081-989 2666.

Aries B32 Shadow RAM card (with manual) £40 inc. p&p. Tel. (0305) 772817.

Master 128, Microvitec 1441 very hi-res. monitor, Akhter twin 40/80 drives in bridge, Taxan 810 printer with RAM fitted, 64k RAM cartridge, GIS Teletext adaptor, Dumpmaster II, Master ROM, Printwise, Viewspell all manuals, books on operating system and assembly language with discs and a large amount of supporting software £740. Tel. (09274) 23659.

BEEBUG Masterfile DFS disc and manual, BEEBUG Printwise disc and manual, BEEBUG Dumpmaster ROM and manual, Dabs Hyperdriver ROM and manual, Merlin Database ROM, disc and manual, Micro Aid Cashbook disc and manual, Acornsoft Viewsheet ROM and manual. No reasonable offer refused. Tel. (0705) 371018.

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

£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

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.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
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: Alan Wrigley
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
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.

ISSN - 0263 - 7561

Printed by **Newnorth Print Limited** (0234) 41111

Magazine Disc/Cassette

MAY 1990 DISC/CASSETTE CONTENTS

FRACTAL PATTERN GENERATOR - A fascinating application for generating fractal patterns resembling various natural shapes, such as trees and snowflakes.

THE ADVANCED PRINTER MODE SELECTOR - A user friendly menu-driven program for setting printing styles, which can be adapted for any printer.

PRACTICAL ASSEMBLER (Part 2) - Two programs which demonstrate the use of assembled machine code in Basic programs.

DESIGNER SHOOT-'EM-UP (Part 1) - Two programs to help you design your own shoot-em-up type arcade games, with more to follow next month.

FIRST COURSE: Using Sideways RAM and ROM Four useful utilities for: listing sideways ROM/RAM contents, loading sideways ROM images (for the Master and the model B) and saving sideways RAM images to disc.

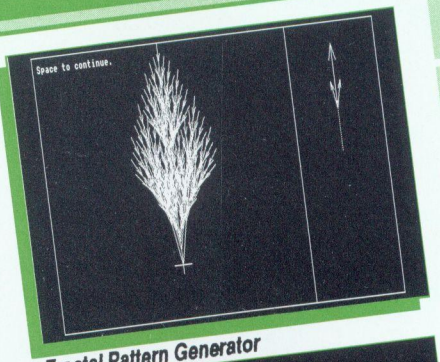
EDIKIT (Part 5) - The final program of this series which demonstrates how to add your own commands and utilities to the EDIKIT ROM.

MECHANICAL VIBRATION - An educational program demonstrating different types of vibration, both in graphical form and in tabular format.

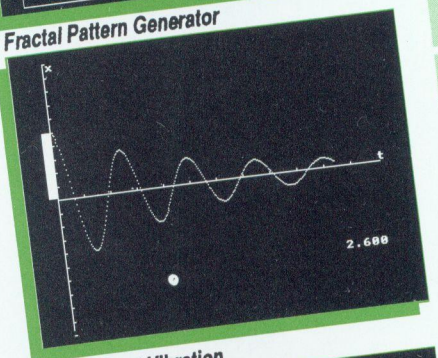
CURVE FITTING (Part 2) - Last month's program updated with additional procedures enabling a variety of functions to be selected when fitting curves to a set of points. A data file is also included.

MAGSCAN DATA - Bibliography for this issue (Vol.9 No.1).

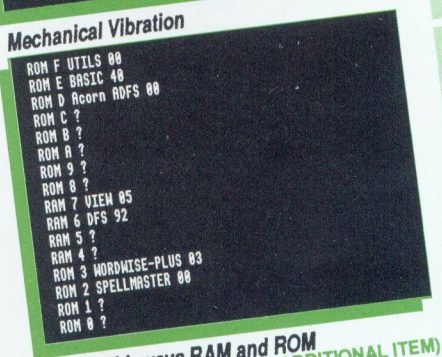
* **QUAD GAME** - An entertaining game, where you are offered different shapes and you need to make them fit.



Fractal Pattern Generator



Mechanical Vibration



Using Sideways RAM and ROM

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	
	Disc (5"or 3.5")	Cassette	Disc (5"or 3.5")	Cassette
6 months (5 issues)	£25.50	£17.00	£30.00	£20.00
12 months (10 issues)	£50.00	£33.00	£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, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

New from Acorn

THE LEARNING CURVE

THE LEARNING CURVE

NEW - FROM ACORN

It consists of:

An A3000 Computer
The new faster PC Emulator
First Word Plus v2 Word Processor
Genesis Information System
A Video explaining how to use the package
Information on the National Curriculum

Save Over £185

The free software bundled with this package is worth £300, and yet the entire system costs only £115 more than the standard A3000.

Ideal For Home & Education

This system is tailor-made for the home user with children. The A3000 and other Archimedes systems are increasingly used in schools and this all-inclusive package with the excellent Genesis software will supplement Information Technology in the National Curriculum.

Just Right for Home & Business

A Word Processor is essential in all home systems, and 1st Word Plus with its built-in spelling checker is excellent. The new PC Emulator also means that over 95% of PC software (DBase, Lotus, Wordstar etc.) will run on the A3000 straight away.

RISC User

RISC User is the world's largest Archimedes Support Group, and publishes the leading magazine devoted exclusively to the A3000 and Archimedes. An annual subscription is **£16.90 (UK)** and carries a multitude of benefits, including FREE technical support and discounts.

SPECIAL OFFERS FOR RISC USER MEMBERS

0% Finance

We are able to offer RISC User members 0% finance over 9 months, for example:

	COST	DEPOSIT	BALANCE	MONTHLY PAYMENTS
Learning Curve Computer only	£803.85	£83.85	£720	£80
Learning Curve Comp + Colour Monitor	£1021.20	£121.20	£900	£100

OR: **ARTISAN II, 10 DISCS IN A BOX & THE ARCADE GAME QUAZER**

Learning Curve Computers are in short supply - order yours now and we will supply on a first-come first-served basis.
BE ONE OF THE FIRST IN THE COUNTRY WITH THIS SUPERB SYSTEM.

ORDER FORM

Please supply: **Learning Curve Computer Only** (£803.85) ☐
Learning Curve Colour System (£1033.80) ☐
Information about the Learning Curve System ☐
A RISC User Subscription (£16.90 UK) ☐
0% Finance application form for: the Computer only ☐
the Colour System ☐

I enclose a cheque for £_____ ***(Please add £8 for courier delivery)***

Or: Please debit my Access/Visa/Connect Card No____/____/____/____/

Card Expiry Date ____/____/ Signature _____

Name _____ RISC User/BEEBUG Memb No:_____

Address _____

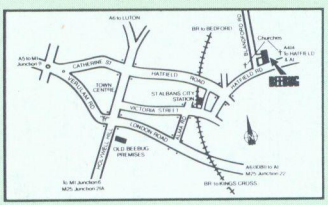
_____ Post Code _____

HOW TO FIND BEEBUG

BY CAR
St Albans is easily reached from A1, A5, A6, M1 and M25

BY TRAIN
We are 10 minutes walk from St Albans City Station on the King's Cross to Bedford line.

Office Hours 9am - 5pm Mon - Fri
Showroom Hours 9am - 5:30pm Mon - Sat
Showroom is open until 8pm on Thursdays.



If you would like a leaflet on the Learning Curve Computer System write to:
BEEBUG Ltd 117 Hatfield Road, St Albans, Herts AL1 4JS . Phone 0727 40303 FAX 0727 60263