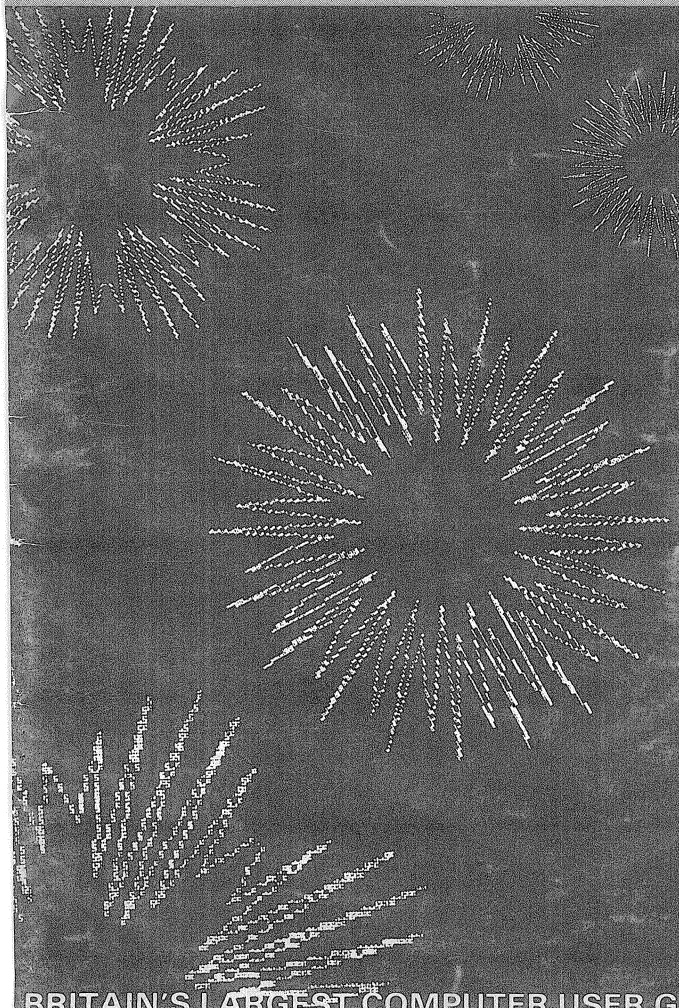
£1.20

BEEBUG

for the BBC micro

Vol 4 No 1 MAY 1985



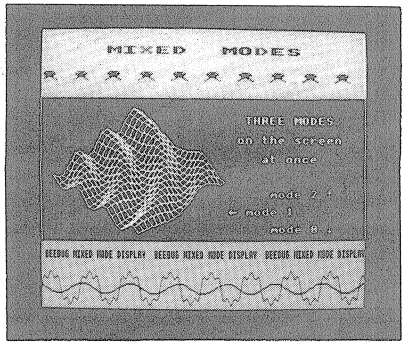
REVIEWS

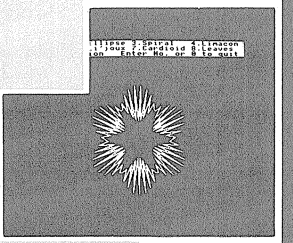
- Ultracalc 2
- Acorn Music 500
- Clef Music System
- ATPL Symphony Keyboard
- Island Logic Music System
- Utility ROMs
- Books on Graphics

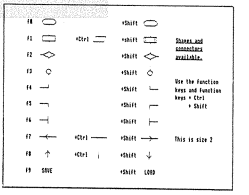
FEATURES

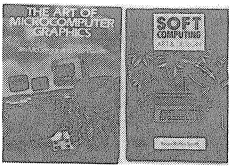
- Polar Curves
- Mixed Modes Explained
- Flowchart Generator
- Extended Assembler for Second Processors
- Adventure Games
- Making Music (Part 4)
- Free Memory Display
- Workshop on Sorting
- And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 30,000





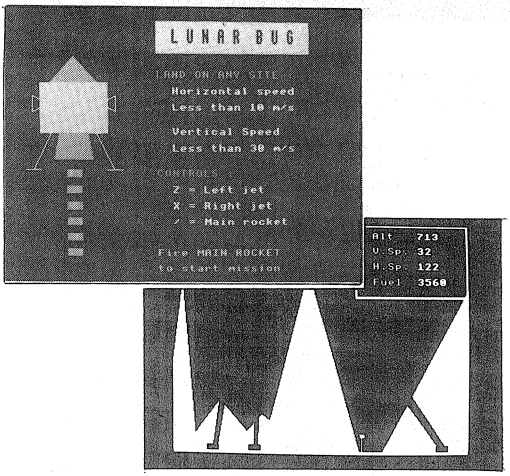


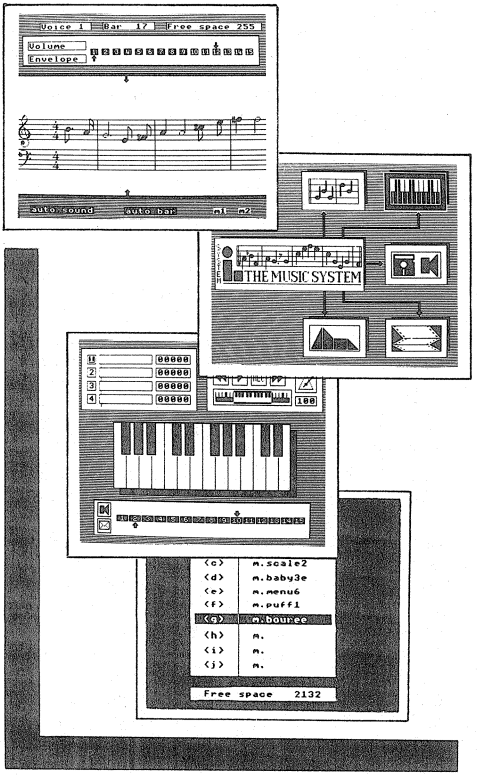


GENERAL CONTENTS

PROGRAMS

HINTS, TIPS & INFO





EDITORIAL JOTTINGS

Welcome to Volume 4 of BEEBUG. To mark the first issue of the new volume we are including a detailed and practical index to the whole of volume 3, and a voucher worth up to £3 when ordering from BEEBUGSOFT. We have also included two extra items on this month's magazine cassette. One is an extremely effective graphics display, based on the theme of an English country garden with suitable musical accompaniment. The second item is a first rate arcade game which has certainly proved quite addictive with the magazine staff. In fact, we wouldn't mind betting that anyone else would have charged you the cost of the magazine cassette for this game alone. With all the other programs from the magazine, the magazine cassette/disc provides even greater value for money this month.

Last month we ran out of space for our hint winners. They were K. Kilmoore (£10) and T.K. Cowell (£5). This month we have selected as winners the hints by E. Williams (£10) and B.R. Hill (£5). More hints and tips will always be appreciated. Remember too, that there is a special prize of £15 for any particularly outstanding hint published.

We are also revising and extending our system of testing the programs published in the magazine. In future each program will be marked with a set of symbols (icons) positively indicating the systems on which the program will work. The symbols and their meanings are as follows:-

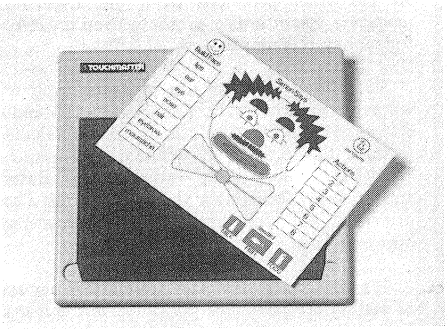
Basic I		Electron	
Basic II		Disc	
Tube		Cassette	

A symbol with a cross over will indicate that the program will not work for that system, a single line through a symbol will indicate that the program will work if modified, while an unmarked symbol indicates full working. The Electron has also been included for completeness. We hope to include these symbols within the menu on the magazine cassette/disc.

NEWS

THE MASTER TOUCH

If you're sick of struggling with the BBC micros's QWERTY keyboard, Touchmaster have the thing for you. The Touchmaster is a graphics tablet with a resolution of 256 x 256 that can be used for more than just graphics. Along with suitable overlays, Touchmaster can be used as an input device to replace the keyboard in games and educational software. The Touchmaster costs £150. Further details from Touchmaster on 0656-744770.



ALL KEYED UP

An add-on numeric keypad is available from Softlife, the maker of the excellent and cheap Softlife Eprom programmer. The keypad connects to the user port and comes with driving software on ROM for £60.25. It features all the number keys along with Return, Delete, decimal point, and other goodies. Softlife is on 0223-62117.

INSURANCE

The Micro Repair Club can offer you peace of mind for £24.95. If the guarantee on your Beeb (or any other home computer that you may own as well) has run out the Micro Repair Club offer an extended guarantee service for £24.95 for the first year and £14.95 thereafter. All repairs to your micro are entirely free while you subscribe to the club. Further details on 0990-28102.

WATFORD MOVES

Watford Electronics has moved from its tiny premises in Cardiff Road in Watford to a spanking new 9000 square foot building just down the road. The new WE address is

Watford Electronics,
250 High Street,
Watford,
WD1 2AN.

NEW BOOKS

There are a few new books of special interest to Beeb owners out recently. The Wordwise Applications Guide is concerned with the old Wordwise but will still apply in the most part to Wordwise Plus. It is written by Paul Beverley and published by Norwich Computer Services (0603-621157) for £6.50. The Hackers Handbook is a guide to that nefarious nocturnal activity for the uninitiated. Written by Hugo Cornwall (a pseudonym we are assured), it is published by Century at £4.95. Everything you always wanted to know about ROMs but were afraid to ask is not the title of Bruce Smith's latest book but could well be. Actually entitled 'The BBC ROM book', it is published by Collins at £9.95.

NEW SOFTWARE

There are several new arcade games in the offing this month. Superior Software have released a version of 'Tempest' with the approval of the games originators, Atari. 'Tempest' costs £9.95 (£11.95 on disc). From A and F comes 'Orpheus', a Frogger look-alike, for £6.90 and 'Arabian Nights' has arrived from Interceptor for £6.99. 'Combat Lynx' comes to the BBC micro courtesy of Durrel for £8.95 and on the same lines 'Laser Attack' (where do they get these clever names from?) by Viking Software costs a mere fiver. Even cheaper is 'Kissin Cousins' from English software at £4.95. Level 9 has another excellent adventure for the Beeb called 'Emerald Isle'. This will set you back £6.95. If you want to take a break from the active role two packages from Addison-Wesley - 'Tessalator' and 'Graphito' - promise to boggle your mind with graphics for £22.95 and £21.95 respectively (£29.95 and £27.95 on disc). Finally you can relax after all the action with a game of 'Whist' from Dotsoft if you have £6.50 to spare.

MUSIC THE EASY WAY

The Island Logic Music System

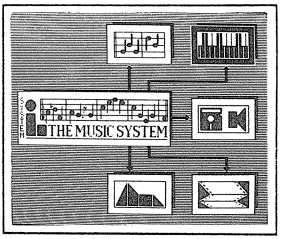
After the ballyhoo of Island Logic's outrageous advertising, Steve Ibbs, an enthusiast of synthesized music and director of his own music studio, gives a more realistic assessment.

```
Title    : Music System
Supplier : Island Logic
           c/o System, 12 Collegiate
           Crescent, Sheffield, S10 2BA.
Price    : £24.95 (disc)
           £12.95 (cassette)
```

It is a pity that the rather brash, somewhat tasteless adverts for the Island Logic Music System strike a bad note in this otherwise excellent software package for the Beeb. The style continues in the small introductory leaflet and is a discordant contrast with the superbly-written manual.

After using the system for some time, I can honestly say that it is the best music software package for the Beeb, using the internal sound chip, that I have ever seen. Icons and windows are used to great effect, and make it a joy to use. The main manual is excellent and the graphics are impressive. The system has 5 major options, the first of which enables 1-4 part songs (the 4th being percussion) to be composed, edited and played, as individual parts or combined. The screen displays the voice being edited at the time in normal stave notation, and the insertion or deletion of notes/rests is very easy. Repeats, first/second time bars, triplets, etc. are all possible, and a large number of control keys are available to speed up the process. Swapping to the other lines to check alignment is simple, and the volume and envelope can be modified for each note if desired.

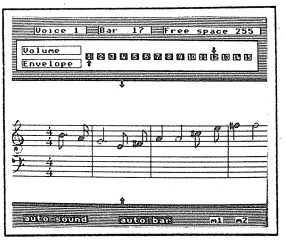
The second option converts the keyboard to a quite sophisticated musical keyboard and part of the screen shows controls similar to a tape recorder. A part can be recorded, then played back whilst the next track is added. The graphics display indicates how much 'tape' is left, and there is even a tape counter, fast forward



Music System Control Screen.

and rewind, and a moving metronome icon. The complete recording can then be saved, and loaded into both the editor for modification, and the printer for a printed score.

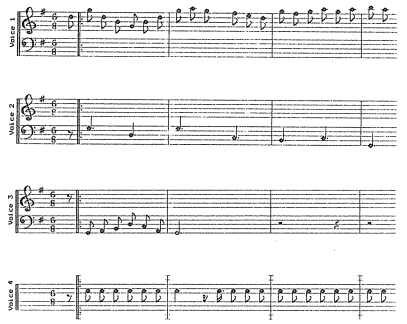
The third option allows musical files to be linked together to produce lengthy compositions. This option is only available on the disc version and enables up to 10 files to be sequenced together in up to 16 different steps.



The Editor.

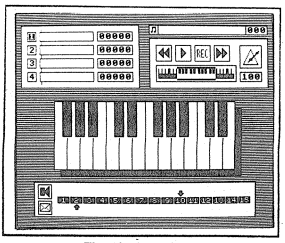
The 'linker' file can then be saved as a separate file. A minor criticism here is that it is impossible to move back from the 'sequence screen' directly to the 'file screen'.

The fourth option allows a printout of the musical files, an example of which is shown. This is of the best quality I have seen to date, including even the printout of other systems costing hundreds of pounds! Any or all of the parts can be printed in either high or low (slightly quicker) resolution mode and the parts can be

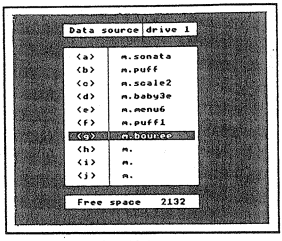


aligned or non-aligned, the latter alternative saving some paper. There was one problem: as I had set my Epson RX80 to generate a linefeed character, rather than trying to remember *FX6 every time, and the dumpout also generates one, double line-spacing occurred. There ought to be a choice available within the program.

The final option allows the pitch and envelope parameters to be loaded, altered and saved, with superb graphics. Windows and icons are again used extensively and the overall effect is to make sound creation simple. Frequency modulation is programmable, and the sounds thus created can be saved, then loaded from within the editor or keyboard options. A minor improvement would be to enable the sound and envelope graphs to be dumped on the printer.



The Keyboard.



The Linker.

The manual is very well illustrated, with summaries at the back to show all the command key functions. It would be extremely difficult to improve on this package without introducing an external keyboard or sound production hardware. It is extremely 'user-friendly', and sensible default values are included everywhere to save unnecessary typing. Excellent, but the advertising copy writer ought to be replaced.

In accordance with the editorial policy on reviews given in BEEBUG Vol. 3 No. 9, we present here the main features of MUROM, a music system produced by Beebugsoft.

MUROM

MUROM is the new self-contained 8K ROM from Beebugsoft. It comprises a full screen Music Editor and Envelope Editor, and comes with a comprehensive manual, reference card, function key strip, and demo cassette containing over 15 minutes of music.

All four music channels are displayed together allowing easy alignment of the melody and harmony, and use of Mode 7 enables well in excess of 8000 notes to be stored in memory at once. Notes may be entered by name, or in piano-keyboard style.

Ten pre-defined envelopes (*PIANO, *FLUTE, etc), and ten pre-defined sounds are provided on the Rom and may be included within your programs. (*SIREN, *PING, etc).

*PLAY is an interrupt driven command that will play music data in memory and still enable the computer to be used for any other tasks, such as printing, running another program, cataloguing a disc etc.

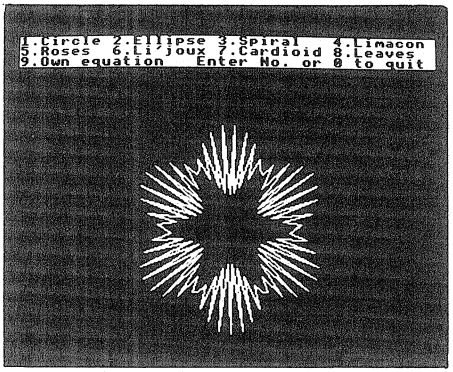
MUROM is priced at £29.00 (before members' discount). For further details please contact the Software Manager, BEEBUGSOFT, P.O. Box 50, St Albans, Herts.

POLAR CURVES EXPLAINED

Most of us have some simple understanding of cartesian co-ordinates, which are essential for exploring the graphics potential of the Beeb. However, some of the most fascinating displays are best programmed using 'polar co-ordinates'. Stuart Robinson explains what it's all about.

The major difficulty when drawing curves on the screen of the BBC micro is actually calculating the co-ordinates to join up and form the curve in an efficient manner. Some curves are ideally suited to the Cartesian (x,y) system used for the Beeb screen display but many are not.

Several curves, however, can be easily calculated if a different co-ordinate system is used - the polar co-ordinate system. Like the Cartesian system, the polar co-ordinate system uses two co-ordinates. The first is the distance from the origin (R) and the second the angle (theta) subscribed by the line joining the point to the origin (called the 'pole') and a base line (called the 'initial' line).



Just as curves may be expressed as Cartesian functions, e.g.:

```
Y = fn(X)
```

so can many curves be more simply expressed as polar functions - a description of how R changes as theta varies, e.g.:

```
R = fn(theta)
```

Many curves have very simple functions, when expressed in polar terms, that are mind bogglingly complex in the Cartesian system. Some of the more popular classroom curves are given in the table with their polar functions. Try working out the Cartesian equivalent functions, if you dare!

Curve	Polar equation
Circle	R=size
Ellipse	R=(4/(2+COS(theta)))*size
Spiral	R=theta*size
Limacon	R=(1+COS(theta))*size
Rose	R=(SIN(N*theta))*size
Cardioid	R=(1+M*COS(theta))*size
Leaves	R=(2+SIN(N*theta))*size

In each case 'size' is just a scaling factor. For functions such as the circle, theta must be measured in radians.

Of course, to use such polar functions on the Beeb, we run into the problem that Basic is only designed to understand Cartesian co-ordinates. We need a method of converting points calculated using the polar system to the Cartesian system needed to plot them.

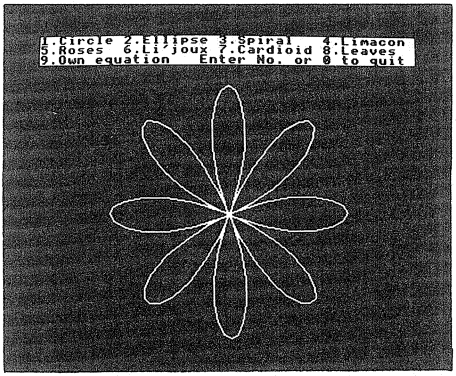
CONVERSION

The mathematical formulae which enable you (or your computer) to convert from one system to another are simple.

```
X = R*COS(theta)
Y = R*SIN(theta)
```

This is easily understood if we use a little trigonometry on the diagram.

To draw a curve using the polar co-ordinate system we simply calculate R for a number of values of theta using the polar function. These values of theta and R are then converted to X and



Y co-ordinates using the equations, above, and the points plotted in the normal way.

For a simple curve such as a circle this is perhaps trivial, but with more complex curves this apparently roundabout method actually makes life much easier. The program illustrates the technique. It will draw all of the curves in the table, including roses and leaves with different numbers of petals, and also curves known as Lissajoux figures.

The program offers a choice of eight different polar curves to be displayed, and in some cases a choice of parameters as well. The number of petals in each rose and leaf depends on the value of N% in the equation. The program prompts for a value for N%. In the equation for leaves try 3, 25 and 45 to produce different patterns. Lissajoux figures are created by using the equation for a circle, but the X and Y co-ordinates are computed using two different angles. The idea is difficult to grasp but the effect is startling. Mathematically speaking X=R*COS(theta), Y=R*SIN(phi) and the two angles, theta and phi, are said to be out of phase. You can also enter your own equation and size. For example try theta+SIN(theta) and a size of 100.

PROGRAM NOTES

The main section of the program is concerned with the selection, from a simple menu, of the curve function (in polar form) to be displayed. On selection a string (curve$) containing the function is passed to the procedure PROCdraw along with the position of the origin on the screen and limits of the co-ordinates for the most pleasing effect.

PROCdraw is the heart of the program and it performs all the real work calculating the co-ordinates and drawing the curve. The graphics origin is moved with VDU29 so that each display is centralized. The variable end% determines how many times the FOR-NEXT loop is executed and is usually set to 2.5*PI - just over a full 360 degrees. The exception is the spiral. The spiral is the only open curve drawn by the program and the greater the value of end% the longer the spiral will be. All of the other curves are closed curves - that is to say the start and finish points on the screen are identical.

For each value of theta, EVAL (in line 1540) evaluates the equation contained in curve$ and sets the result equal to R. The X and Y co-ordinates are now calculated from R and theta in line 1560. Phi is calculated as F%*theta and obviously for curves other than Lissajoux's F%=1 and so phi=theta. Finally the points are plotted in line 1570. The first point is plotted with PLOT 4 (a MOVE statement) and the rest with PLOT 5 (DRAW).

PROCdraw is a very flexible procedure but it is far from being the most efficient procedure for drawing any one of the curves in the menu. For use in your own programs you will probably want to modify it. The major reward to be gained is one of increased speed. Much is also to be gained by experimenting with the program listed, trying different equations and altering the parameters of the ones given here. The program provides an insight into the fascinating world of computer geometry and polar curves.

ITION - COMPETITION - COMPETITION - COM

When you've gained a little insight you might like to earn some cash with your polar curves. We are offering a prize of £50 for the best function to enter for option nine in this program. Try out as many different functions as you

can think of and pick your favourite. The prize will go to the one that produces the most creative and pleasing effect. Don't forget to include a value for 'size'. Send your function (one only please) to the Editorial address and mark the envelope 'Polar competition'. Entries must be in by 10th June. Good luck.

```
  10 REM Program Polar Curves
  20 REM Version B0.1
  30 REM Author  S. Robinson
  40 REM BEEBUG  May 1985
  50 REM Program subject to copyright
  60 :
  70 ON ERROR GOTO 1670
  80 :
 100 MODE 1
 110 PROCinit
 120 REPEAT
 130 PROCchoice
 140 UNTIL fini%
 150 MODE 7
 160 END
 170 :
1000 DEFPROCinit
1010 VDU28,0,2,39,0
1020 VDU24,0;0;1279;920;
1030 VDU19,1,2;0;19,3,4;0;
1040 COLOUR0:GCOL0,2
1050 COLOUR129:GCOL0,131
1060 CLS:CLG
1070 fini%=FALSE
1080 ENDPROC
1090 :
1100 DEF PROCchoice
1110 F%=1
1120 PRINT"1.Circle 2.Ellipse 3.Spiral
  4.Limacon"
1130 PRINT"5.Roses  6.Li'joux 7.Cardio
id 8.Leaves"
1140 PRINT"9.Own equation   Enter No.
or 0 to quit";
1150 REPEAT
1160 *FX15,0
1170 key$=GET$
1180 UNTIL INSTR("0123456789",key$)<>0
1190 CLS:CLG
1200 IF key$="1" THEN PROCdraw("400",6
40,460,2.5*PI)
1210 IF key$="2" THEN PROCdraw("500/(2
+COS(theta))",800,460,2.5*PI)
1220 IF key$="3" THEN PROCdraw("10*the
ta",640,460,10*PI)
1230 IF key$="4" THEN PROCdraw("200*(1
+COS(theta))",500,460,2.5*PI)
1240 IF key$="5" THEN N%=FNpetals:PROC
draw("400*COS(N%*theta)",640,460,2.5*PI)
1250 IF key$="6" THEN F=FNliss:PROCdra
w("250",640,460,2.5*PI)
1260 IF key$="7" THEN PROCdraw("100*(1
+6*COS(theta))",300,460,2.5*PI)
1270 IF key$="8" THEN N%=FNcontour:PRO
Cdraw("100*(2+SIN(N%*theta))",640,460,2
.5*PI)
1280 IF key$="9" THEN PROCown:PROCdraw
(curve$,640,460,2.5*PI)
1290 IF key$="0" THEN fini%=TRUE
1300 ENDPROC
1310 :
1320 DEF FNpetals
1330 REPEAT
1340 INPUT"Enter No. of petals and pre
ss RETURN"''"3, 4, 7, 8, 11, 12 ",N%
1350 UNTIL (N%-3)*(N%-4)*(N%-7)*(N%-8)
*(N%-11)*(N%-12)=0
1360 IF (N% AND 1) THEN =N% ELSE =N%/2
1370 :
1380 DEF FNliss
1390 REPEAT
1400 INPUT"Enter phase factor and pres
s RETURN"''"1 to 9 ",F%
1410 UNTIL F%>0 AND F%<10
1420 =F*0.2
1430 :
1440 DEF FNcontour
1450 REPEAT
1460 INPUT"Enter No. of leaves and pre
ss RETURN"''"1 to 50 ",N%
1470 UNTIL N%<51 AND N%>0
1480 =N%
1490 :
1500 DEF PROCdraw(curve$,X%,Y%,end%)
1510 CLS:CLG
1520 VDU29,X%;Y%;
1530 FOR theta=0 TO end% STEP PI/500
1540 R%=EVAL(curve$)
1550 phi=F%*theta
1560 X%=R%*COS(theta):Y%=R%*SIN(phi)
1570 PLOT 4-(theta<>0),X%,Y%
1580 NEXT
1590 ENDPROC
1600 :
1610 DEF PROCown
1620 INPUT"Equation [ fn(theta) ] "cur
ve$
1630 INPUT'"Size "R$
1640 curve$="("+curve$+")*"+R$
1650 ENDPROC
1660 :
1670 ON ERROR OFF
1680 MODE 7
1690 IF ERR<>17 REPORT:PRINT " at line
 ";ERL
1700 END
```

EXTENDED ASSEMBLER FOR THE 65C02

The 65C02 processor used in Acorn's second processor has an extended instruction set which is not supported by Basic's built in assembler. Dominique Willems shows how *the assembler can be extended to overcome this limitation.*

As Acorn launched its 65Ø2 second processor, much fuss was made about the 3 MHZ execution speed and the rather massive memory available, but what was kept in the dark, and for no obvious reason, was the fact that "65Ø2 second processor" in reality meant a brand new G65SCØ2 microprocessor. This new version of the 65Ø2 contains an enhanced instruction set which not only allows faster programs, but also provides memory saving instructions (which compact several old ones).

It is surprising to see that Acorn didn't make use of the advanced instruction set to develop their HiBasic interpreter, which instead still consists of the old 65Ø2 instruction set. It seems that even higher speeds could be obtained here, though the original timings are very satisfactory.

THE NEW INSTRUCTIONS

A brief look first at the enhanced instructions:

ORA(ZP) AND(ZP) EOR(ZP) ADC(ZP)
STA(ZP) LDA(ZP) CMP(ZP) SBC(ZP)

These do away with loading the Y-index register with zero. (ZP denotes a zero page address).

BIT IMM

Allows pre-setting of status flags using immediate mode.

BIT ZP,X BIT ABS,X

An extension of the old instruction using the X index register.

INC A DEC A

An interesting improvement on accumulator addressing - saves clearing the carry flag and adding or subtracting 1.

JMP(ABS,X)

This new instruction uses indexed absolute indirect addressing. The recommended Rockwell syntax for this is, confusingly, JMP (ABS),X and not JMP (ABS,X), but as the latter is logically correct we have used this mnemonic instead of the standard Rockwell one. The contents of the second and third bytes of the instruction are added to X, and the resulting address contains the jump address.

And now for the new commands:

BRA REL

At last an unconditional branch!

PHY PLY PHX PLX

Allows TXA:PHA:TYA:PHA to be replaced with PHX:PHY.

STZ ZP,X STZ ZP STZ ABS STZ ABS,X

STore Zero-value in memory. Replaces LDA#Ø:STA memory.

TSB ZP TSB ABS

Logically ORs memory contents with Accumulator and stores result back in memory.

TRB ZP TRB ABS

Logically ANDs memory contents with the inverse of the Accumulator (NOT A) and stores the result back in memory. Same as: LDA#value:EOR#255:AND memory:STA memory.

THE PROGRAM

Since the HiBasic Assembler doesn't include the new instructions, a method had to be found to implement them in user programs. Of course the EQU-operators could be used to put hex codes directly into memory, but this would be unprofessional and lacks clarity. The best way is via a machine code program which co-operates with the current Basic interpreter. A major advantage of Basics residing in second processors is that they exist in RAM, which provides the possibility to change values and implement JMP instructions to other machine code routines residing in user memory. This is exactly what the following program does.

After typing in the program, firstly save it as a Basic program, and then run it. If no errors occur, then type:

```
*SAVE ASSEX F686 F7FD  for Basic II
*SAVE ASSEX B67D B7FD  for HiBasic.
```

Now you are ready to use the extended assembler.

The program listed here allows all of the new instructions to be used in assembler programs in exactly the same way as for the existing ones. The routine works fully independently and needs only to be executed once by typing: *ASSEX (or *RUN ASSEX). The program will load itself at &B67D (or &F686 for Basic II users). It will then execute the first part which changes the appropriate values in the Basic interpreter area. Since this routine needs only to be executed once, HIMEM will be reset to the start of the actual extension program. In the Basic II version HIMEM will not be reset since the program will be located in the user memory area above the Basic interpreter (and not normally accessible).

The program mainly intercepts machine code routines in the Basic interpreter and redirects them to the extension routine. For example, when a PLY instruction is encountered, the JMP instruction to the error message has been changed to point to the appropriate routine.

If you want to use the extensions with Basic II active then delete line 1100 and change following the lines:

```
1020 P%=&F686
1730 ?&F6A5=&9C
1810 DATA &870D,&8767,&8712,&870E,
     &870F,&8605,&8606,&8746,&8768
1820 DATA &8769,&8775,&8713,&8714,
     &87B0,&87B1,&882C,&862B,&8623,
     &982A
1830 DATA &876A,&8832,&8782,&8A97,
     &879A,&86A8,&F7FD,&F7FE,&87CE,
     &8821
```

This extended assembler should prove most useful to machine code programmers using the 6502 second processor. Perhaps what is needed now is a corresponding disassembler. Any offers?

```
  10 REM PROGRAM EXTENDED ASSEMBLER
  20 REM VERSION B0.1
  30 REM AUTHOR  D.WILLEMS
  40 REM BEEBUG  MAY 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 MODE3
 110 PROCaddress
 120 PROCassemble
 130 END
 140 :
1000 DEFPROCassemble
1010 FORI%=0TO3STEP3
1020 P%=&B67D
1030 [OPTI%
1040 \ ONE TIME EXECUTION PROCEDURE
1050 LDA #&4C:STA A(0):STA A(1):STA A(
2)
1060 LDA #INDEX MOD 256:STA A(3):LDA #
INDEX DIV 256:STA A(4)
1070 LDA #SYNTAX MOD 256:STA A(5):LDA
#SYNTAX DIV 256:STA A(6)
1080 STA A(7)
1090 LDA #incdec MOD 256:STA A(8):LDA
#incdec DIV 256:STA A(9)
1100 LDA #LAST MOD 256:STA &6:LDA #LAS
T DIV 256:STA &7  \ RESET HIMEM
1110 LDA #&9C:STA A(10)
1120 LDA #bit MOD 256:STA A(11):LDA #b
it DIV 256:STA A(12)
1130 LDA #jump MOD 256:STA A(13):LDA #
jump DIV 256:STA A(14)
1140 .LAST RTS
1150 \ START ACTUAL EXTENSION
1160 .INDEX AND #&A6:BEQ indexerr:INC
&29:JSR A(15):LDY #2:JMP A(16)  \ INTER
CEPTION INDEX ERROR
1170 .SYNTAX LDA &3D:CMP #&41:BNE next
:LDA &3E:CMP #&A:BNE next
1180 LDA #&80:LDX #&20:JMP A(17)  \ BR
A EXTENSION
1190 .next LDA &3E:CMP #&41:BNE not1
1200 \ PHY,PLY,PHX,PLX EXTENSION
1210 LDA &3D:CMP #&19:BNE ply
1220 LDA #&5A:BNE endi
1230 .ply CMP #&99:BNE phx
1240 LDA #&7A:BNE endi
1250 .phx CMP #&18:BNE plx
1260 LDA #&DA:BNE endi
1270 .plx CMP #&98:BNE not1
1280 LDA #&FA
1290 .endi STA &29:LDY #1:JMP A(16)
1300 .syntaxerr JMP A(18)
1310 .indexerr EQUB 0:EQUB 3:EQUS"Inde
x":EQUB 0
1320 \  INC A, DEC A EXTENSION
1330 .incdec LDA &29:CMP #&C6:BNE inc
1340 LDA #&3A:BNE out
1350 .inc CMP #&E6:BNE other
```

```
1360 LDA #&1A:.out STA &29:JMP A(19)
1370 .other JSR A(20):JMP A(19)
1380 .not1 LDA &3E:CMP #&52:BNE not2
1390 LDA &3D:CMP #&62:BEQ TSB
1400 CMP #&42:BEQ TRB
1410 .not2 LDA &3E:CMP #&4E:BNE syntax
err
1420 LDA &3D:CMP #&9A:BEQ STZ
1430 JMP  A(18)
1440 .TSB LDA #0:.ret STA &29:JMP A(21
)  \ TSB EXTENSION
1450 .TRB LDA #&10:BNE ret  \ TRB EXTE
NSION
1460 \  STZ EXTENSION
1470 .STZ JSR A(22):CMP #&28:BEQ indir
err
1480 DEC &A:JSR A(28):JSR A(22)
1490 CMP #&2C:BEQ indexSTZ
1500 LDA #&64:STA &29:LDA &2B:BNE abso
lute:JMP A(24)
1510 .absolute LDA #&9C:STA &29:JMP A(
23)
1520 .indexer2 BNE indexerr
1530 .indexSTZ JSR A(22)
1540 CMP #&58
1550 BNE indexerr
1560 LDA #&74:STA &29
1570 LDA &2B:BNE absolutind
1580 JMP A(24)
1590 .absolutind LDA #&9E:STA &29
1600 JMP A(23)
1610 .indirerr EQUB 0;EQUB 6:EQUS"Indi
rect":EQUB 0
1620 \  BIT EXTENSION
1630 .bit LDA &A:STA A(25):LDA &B:STA
A(26):JSR A(22):CMP #&23:BEQ immediate
1640 LDA A(25):STA &A:LDA A(26):STA &B
1650 LDA #&20:STA &29:LDA #&18:PHA:JMP
A(27)
1660 .immediate LDA #&89:STA &29
1670 JSR &C043:JMP A(24)
1680 \  JMP EXTENSION
1690 .jump CMP #&2C:BNE indexer2:JSR A
(22):CMP #&58
1700 BNE indexer2:JSR A(22):CMP #&29:B
NE indexer2
1710 LDA #&7C:STA &29:JMP A(23)
1720 ]NEXTI%
1730 ?&B69D=&9C
1740 ENDPROC
1750 :
1760 DEFPROCaddress
1770 DIM A(28)
1780 FORS=0TO28:READ A(S):NEXT
1790 ENDPROC
1800 :
1810 DATA &BF2F,&BF89,&BF34,&BF30,&BF3
1,&BE26,&BE27,&BF68,&BF8A
1820 DATA &BF8B,&BF97,&BF35,&BF36,&BFD
2,&BFD3,&C04E,&BE4C,&BE44,&D045
1830 DATA &BF8C,&C054,&BFA4,&C2B6,&BFB
C,&BECA,&B7FD,&B7FE,&BFF0,&C043
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BEEPLESS BREAK - James Percival

Hold down Ctrl and Escape while the Break key is pressed and the Beeb won't Beep.

BASIC DIFFERENCE - Des Fisher

In Basic I the function LEN(STR$(4601/100)) will return 5. However, in Basic II this function gives 11 as the number 4601/100 has not been rounded down but is treated as 46.0100001.

USES FOR SQUARE BRACKETS - John Blackburn

A left hand square bracket followed by Return - i.e. [<Return> - is a valid Basic command and returns the current value of P% in hex.

Using CTRL-[will act just like pressing Escape.

*SPOOL USES - C.T. Marshall

*SPOOL is more versatile than you might think. For example, you could have a program that had a *SPOOL <filename> at the start, read some character definitions from DATA statements, and then output the necessary Basic statements to redefine the character. These would then be in the file ready to redefine the characters at a later stage just by *EXECing the file back in.

BUG IN EPSON MX80 - Richard Sterry

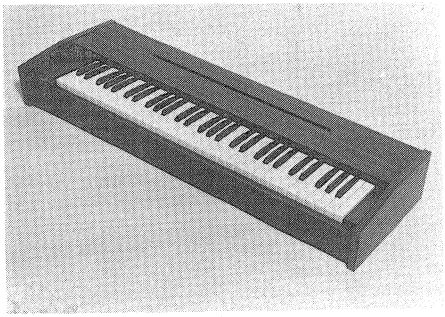
If you have an Epson MX80 type 3 (there is some variation even among the type 3s) then you will probably find that if you attempt to select linefeeds of 1/8", and have set the DIP switches to use a 'slashed' zero, then you will find that it does not respond to this. The way round this is to use a 9/72" linefeed.

MUSIC SYSTEMS FOR THE BEEB

Reviewed by Geoff Bains, Steve Ibbs and David Reed

The BBC Micro itself has much to offer the music enthusiast but it also forms an ideal base for the addition of more sophisticated hardware. We look this month at some of the delights now available to tempt the serious user.

Product : Clef Computer Music System
Supplier : Clef Computer Music
44a Bramhall Lane South,
Stockport, Cheshire, SK71AH.
061-439 3297
Price : £495



At more than the price of the Beeb, the Clef Music System has to be something special. It is. It is a complete music system with synthesizer, keyboard, and software ready to plug into your BBC micro for studio or stage work.

The normal method of sound synthesis used on cheap synthesizers is to produce a waveform rich in harmonics, like a square wave, and then filter out the unwanted overtones to produce the required sound. The Clef system tackles the problem in a different way. Sounds are created by programmable digital -- oscillators capable of producing any wave shape. 32 sound generators are available, each with its own programmable envelope and keyboard touch sensitivity. Up to 4 sound generators can be assigned to each voice, giving a minimum of 8-note polyphony available on the 61 note (5 octave) keyboard included.

Software is provided on both disc and ROM. Once the software is loaded from disc the keyboard can be played, and different preset instruments selected. One problem is that the keyboard has to be turned off before an instrument can be changed. 18 different voices are available using the keys 1-9, and Shift 1-9. I wonder why the function keys haven't been used? Nevertheless the default set of sounds is not at all bad, if a little unimaginative. Pieces can be recorded in 'real-time', and then played back instantly with all dynamic and pedal variations included. They can then be stored on disc for later recall.

Selecting 'M' displays the menu options, the first of which is the boot-up option. The second enables any of the sounds to be completely altered by changing the number of oscillators, the waveform, the fairly comprehensive envelope parameters, touch sensitivity, sustain pedal etc. Any voice with altered parameters can then be filed as a new instrument.

The third option involves creating new sets of instruments, to provide almost limitless numbers of voices, at least in theory. This sub-menu also gives a hint of further goodies to come, because selecting one option (new table creation) produces the message 'Not yet available', and a return to the menu!

The fourth option enables waveforms, the very stuff of voices, to be manipulated or new ones created, which can then be filed and loaded as new waveform sets. There are 16 waveforms stored in the ROM which cannot be altered, and a further 16 that are modifiable. In addition the 4 'primary' waves of sine, square, triangle, and sawtooth can be selected

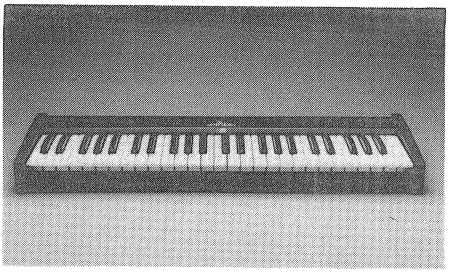
and added. Error trapping should be better here, because if a waveform is selected, stupidly, with an initial amplitude of Ø a 'division-by-zero' message appears and the program has to be re-run.

A graphics display of the waveform is produced and any harmonic up to the 25th with a relative amplitude of 1-1ØØ may be added, with the result automatically redrawn. The 'Analyser' option enables any waveform to be analysed to give a listing of the harmonic content.

The possibilities are badly limited by the software at present available. The most glaring omission is that the system does not give track-on-track recording with editing facilities, different voice for each track etc. The musical possibilities should be exciting, but aren't, because of the software limitations. In terms of hardware I expect a much better finish for a system at this price. The wood-grained effect on the cabinet looks cheap, and doesn't do the concept justice. The output is in the form of two jack sockets for connection to a stereo amp - yes, it's stereo - and the sound quality is acceptable, but doesn't match synthesizers in the same price range. There is too much output filtering, causing the sound to be rather muffled and lacking in bite.

The software flows nicely from one menu to another and a group of children soon worked it out without the benefit of the instruction notes, which are very poor. The musical examples would also benefit from being played better and more accurately. The system is an interesting development and one well worth closer investigation by those wanting to produce decent musical sounds with the computer, and who have developed beyond the internal sound chip! However, as other similar systems are bound to come on to the market, each learning from the mistakes of its predecessors, Clef will have to update and improve to stay competitive.

Product : Symphony keyboard
Supplier : ATPL
Station Rd., Clowne,
Chesterfield, S43 4AB.
0246-811585
Price : £125



The ATPL Symphony is an add-on keyboard for the BBC micro that enables you to really 'play' the internal sound chip.

The quality of this 49 note (three octave) keyboard immediately impressed me, and it makes a smart addition to the Beeb. It connects to the user port, and a disc is provided with the necessary software. The keyboard only uses the internal sound chip, and that is obviously its main disadvantage. Accepting that, it is a nice package, and well suited for both schools and home use.

Several sets of sounds are available, and many more can be created by the user. Some of the sound effects are a bit dubious, but the screen layout is clear and easy to follow. In addition ATPL provide a superb manual which explains the nature of sounds and envelopes in an easily understood way. Sounds can be saved, but compositions can't. However, ATPL says that this option is planned for release in the near future.

Sounds are held in memory in ten groups of ten sounds. A group is selected with the relevant function key with the shift key and a sound from that group called up with the function key alone. The envelope parameters that make up the sound and a few others such as the sustain option and the link-up with the noise channel, are all

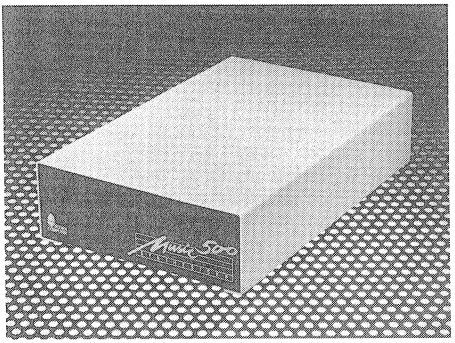
→

provided on screen. These can be easily edited using only the cursor keys.

New sounds created in this way can be saved with your own names ready to be loaded up again later. In this way you can group together all the sounds for one musical piece, call up that group before starting the piece and select the different sounds as you go.

ATPL supplies a small kit to enable the computer's audio output to be connected to a hifi system, surely the quickest and cheapest way to transform the sound from mediocre to surprisingly acceptable. A sustain pedal is another optional extra and the software is already tailored to account for this.

Product : Music 500
Supplier : Acorn Computers Ltd.
Fulborn Rd., Cherry Hinton,
Cambridge, CB1 4JN.
0223-245 200
Price : £199



The Music 500 is Acorn's latest add on for the BBC micro and possibly the most unusual. Launched at the Compec show in November last year the Music 500 is a programmable high quality music synthesizer.

Designed by Hybrid Technology Ltd, and marketed by Acorn, the Music 500 is the size of a single disc drive and has its own on-board power supply with mains switch located at the rear of the unit. It is connected to the BBC micro via a 34-way ribbon cable to the 1MHz bus. The Music 500 unit is designed to be connected directly to a stereo amplifier via a standard 5 pin DIN socket. The front panel of this synthesizer differs from most others in as much as there are no knobs to twiddle, and no music keyboard either.

The Music 500 relies upon the Beeb for all of its commands which have to be pre-programmed using its own language called Ample. The audio output is very good quality. The sounds are clear and crisp - a feature the Clef system could do well to emulate. Listening to some of the demonstration programs included on the cassette provided certainly indicates that the unit is capable of very varied sounds. Incidentally there is a tape to disc transfer program included which allows you to transfer all of the cassette's contents to disc if you wish.

There are a total of sixteen oscillators in the Music 500 each programmable in a similar way to those in the Clef system. Initially they are set up in pairs giving eight separate voices. However, different config -urations are possible such as one incredibly rich voice using all sixteen channels - something the Clef can't manage. The channels in a voice can be offset in pitch from each other, they can modulate one another using ring, frequency, or synchronization modulation to allow very complex sounds to be created. Each channel can be directed to one of seven stereo positions to add even more depth to the sound.

There are 13 programmable, and initially preset, waveforms for instant use along with 13 programmable preset envelopes. These can all be redesigned to your own specification. As well as taking preset designs, the waveforms and envelopes can be set up 'on the fly' while music is playing to give an effectively infinite variety. Waveforms are designed either in terms of harmonic content or 'graphically'. This latter method can be used along with the random function to produce pseudo noise for percussion effects. Envelopes can also be defined in two ways. Either a normal ADSR construction can be used or alternatively a more complex multi-segment construction of your own can be initiated. →

The User Guide that comes with the Music 500 is rather disappointing. The glossy cover hides pages printed with a dot matrix printer. It would seem that it was put together in somewhat of a hurry as it has some odd page numbers while others are blank.

The guide is split into a tutorial and a reference section. The tutorial part is very poor. It covers little ground in a most confusing manner. However, the reference section contains a dictionary of Ample words and that is where a many of the joys of Ample are to be found hidden.

Ample is a word based language. That is, 'programs' are user defined words that call up other user defined or predefined words. Any word can be called up into a text buffer and edited in the same way as a Basic program is edited using the cursor, Delete, and Copy keys. Advocates of Logo will recognise this programming method. In this way a typical music program might be a single word ('play' in the demos) which calls up first a series of sound set up words and then further words that comprise the actual score. This hierachical structure allows you to deal with a complex programming task a piece at a time.

Scoring in Ample is a simple matter of naming notes. If the note is to rise in pitch over the preceeding note a capital letter is used. If the tune descends, a lower case letter is used, though an octave number can be defined for each note if you prefer. In addition, the note length and starting pitches are defined numerically and bar lines can be added. Ample will, if asked, check your composition for the correct number of beats to the bar as it plays it.

The most powerful aspect of Ample is its multi-tasking ability. Separate 'players' can be scored individually and then all set playing together, in time. You can share one player's task from the keyboard, as he is playing, and change his instrument with a few deft stokes of Ample so that, say, a couple of foot is cut off his bassoon (or whatever). This means that you can experiment with the sound of an instrument 'in situ', in the middle of a piece.

Ample doesn't stop there either. The predefined words are all definable too. So that you could, for example, redefine the 'bar' word to stress the first note in every bar. The entire language is amazingly complex and versatile. It makes the software efforts of the Clef system look particularly lame.

Whilst it does take time to get used to the alien nature (at least to most Beeb users) of Ample, it is relatively easy to start producing simple music using only a smattering of the commands available. I managed my first composition (a hymn) using four part harmony in under a week.

You don't have to be a musician to appreciate the Music 500. Indeed Ample is a more than a little daunting to those not experienced in programming. However, a reasonable knowledge of music and some idea of the nature of sound is essential to make the most of this device. For the struggling artist with no liking for the computer, Acorn is soon to bring out a keyboard to complement the Music 500 and some very impressive software is promised too.

Meanwhile there is quite enough to get to grips with in the Music 500 package. For £199 it is difficult to imagine a more comprehensive musical add-on for the BBC micro.

If you are desperate for a keyboard, software is available from ATPL to interface its Symphony to Ample. This will cost you about £15 on top of the £125 for the Symphony keyboard. The software is in the form of a keyboard driver and a demo program. The demo is fairly limited. It offers only preset sounds from the keyboard and some pretty horrendous rhythm tracks. However the demo is only that - a demo. What you do with the interfacing software is really up to your imagination and your skill using Ample as a programming language. When you become proficient at using Ample and decide that you really want to play the Music 500, Symphony and Ample together provide a good vent for your creative urges.

→

NEW VERSION OF ULTRACALC

David Otley takes a fresh look at this Spreadsheet package

A new version of the Ultracalc spreadsheet has now been issued by BBC Soft which has a number of improvements. In particular it meets the two major criticisms made in my comparative review with Acornsoft's ViewSheet in BEEBUG Vol.3 No.3. Firstly, it can now operate in any screen mode allowing an 80 column display. Secondly, portions of the spreadsheet can now be sent to file and thus be incorporated into a word processing program. Ultracalc 2 now represents a highly competitive spreadsheet program that has a number of advantages over ViewSheet.

The display changes mean that mode 3 (or mode 0) can be used to see the maximum amount of the spreadsheet at a time, whilst mode 7 is still available for spreadsheets requiring the maximum amount of memory. However, great care is necessary in changing from mode 7 to other modes because, if the model is too big for the available memory in the new mode, it is completely lost. This is a serious defect in a professional spreadsheet program which should really first check itself whether the requested mode change is feasible, and not allow it to be made if there is insufficient memory available. Other improvements have been made which allow inter-column gaps to be supressed on both the screen display (except in mode 7) and on the printed output.

Output can now be sent to a file as well as direct to the printer, so that tables can be prepared for insertion into documents. This is an important feature which appears to work satisfactorily with both View and Wordwise. In addition, commands can now be sent directly to the printer to set up appropriate type and line spacing, albeit in a somewhat unfriendly manner (e.g. an Epson printer requires the sequence "/>&0F" to select condensed print!). The £ sign can also be defined so that it prints out correctly on a given printer.

A further feature is that Ultracalc 2 now automatically relocates as HICALC when used with a 6502 second processor. Use of a second processor allows the full memory to be used whatever the screen mode selected. This relocation facility gives Ultracalc 2 a worthwhile advantage over ViewSheet which requires a different chip for each system. The manual also claims that Ultracalc can be used with an Electron, provided that a ROM board to Acorn specifications is used. Finally, Ultracalc 2 also includes a brief HELP facility which displays a list of the most commonly used commands.

In conclusion, Ultracalc is now fully competitive with ViewSheet. It does not have ViewSheet's screen windows, but does have variable width columns. Although significantly slower both in recalculation times and in saving models to disc, it has a somewhat wider range of commands. In my opinion, it is easier to use because it does away with the need to refer to any function keystrip, although some users may prefer this. Unfortunately, it is rather more expensive (£80 in comparison with £60) so, for many users the choice will be finely balanced. Both programs represent good value for money and compare well with those available on other computers at several times their cost.

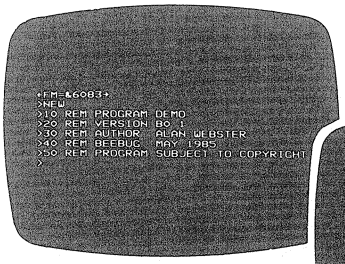
← The music systems reviewed here cover a lot of ground, and money too. If you are a performing musician then the Clef System is the only real choice despite its poor software, though with the advent of Acorn's keyboard this may well change. The ATPL Symphony provides an excellent high quality entry into performing with your Beeb. The Symphony is limited by the very nature of the Beeb's sound chip but what it does it does well. However the best value for money, and the most promising for the future, has to be Acorn's Music 500. This, as you would expect from Acorn, lays down the standards for music add-ons for the BBC micro.

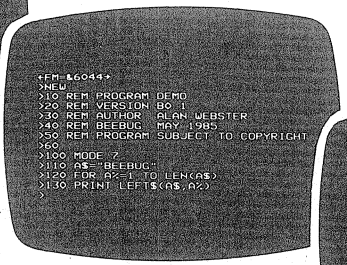
DYNAMIC FREE MEMORY DISPLAY

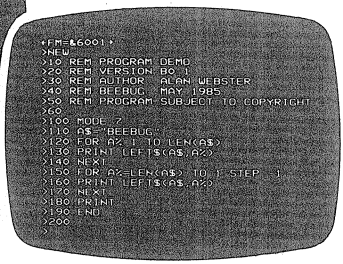
When developing a program in Basic it is often useful to know just how much memory is still available at any time. Alan Webster describes a short routine which will display this information conveniently on the screen and continuously update it for you as well.

There are often occasions when it is useful to see just how much free memory is still left in your machine, both when typing in a program and when running a program still under development.

This short utility displays the amount of free memory left at any time and updates this information continually. The routine is useful when developing programs that could run out of memory. The program displays the words 'FM=&' at the top left hand corner of the screen, and then displays the actual free memory in hex.







The routine then displays a number of spaces after the amount of free memory at the top of the screen. The cursor will now alternate between the current position in your program and the message at the top of the screen. On odd occasions, the routine may interfere with some VDU routines such as 'clear screen', but this is only a small problem which can be easily rectified by typing CLS or VDU12.

Type in the program and save it. Then run the program and, if no errors occur, press Return. The free memory should be displayed at the top of the screen. If not then check the program carefully against the printed listing.

As soon as the program is functioning correctly, you can save the machine code by typing:

*SAVE FREM 900 +110 900 900 (for disc)
*SAVE FREM B00 +110 B00 B00 (for tape)

and re-run the utility at any time by typing *RUN FREM.

PROGRAM NOTES

Most of the important lines in the program are followed by comments, but here is a brief description of each part.

Lines 1040 to 1080 Re-program the event vector to point to the start of our routine. Save the old vector for 'linking' event driven routines and set our routine to respond to the 'vertical sync' event (event number 4).

Lines 1280 to 1310 Output the free memory value in 4 byte hex.

Lines 1430 to 1480 Print out the text following the JSR. It gets the program counter and prints the text from there until it meets a NOP instruction (&EA in line 1520). It then JuMPs back to the NOP instruction. This is used as a quick way to output easily any piece of text.

```
  10 REM PROGRAM FREMEM
  20 REM VERSION B0.38
  30 REM AUTHOR  Alan Webster
  40 REM BEEBUG  MAY 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 V=&FFEE:F=&FFF4
 110 D%=10:*FX13 4
 120 PROCfilesys
 130 PROCassemble
 140 CALLbase
 150 CLS:PRINT
 160 END
 170 :
1000 DEFPROCassemble
1010 FOR A=0 TO 3 STEP 3
1020 P%=base
1030 [OPT A
1040 LDA&220:STAbuf
1050 LDA&221:STAbuf+1 ; Get old Event
vector
1060 LDA#start MOD 256:STA &220
1070 LDA#start DIV 256:STA &221 ; Our
new Event vector
1080 LDA#14:LDX#4:JSRF ; Event 4 - Ver
tical Sync.
1090 LDA#D%:STAbuf+7
1100 .start
1110 PHA:TYA:PHA:TXA:PHA
1120 LDA#&DA:LDX#0:LDY#255:JSRF ; VDU
queue empty?
1130 TXA:BEQsplit:JMPendit ; If no the
n end routine
1140 .split:LDA#117:JSRF:TXA:AND#&40 ;
 Are curor and edit cursor split?
1150 BEQdec:JMPendit:.dec:DECbuf+7
1160 BEQgo:JMPendit:.go:LDA#D%:STAbuf+7
1170 .cryon
1180 LDA&318:STAbuf+2
1190 LDA&319:STAbuf+3 ; Get Cursor pos
ition
1200 SEC:LDA&4:SBC&2:STAbuf+4
1210 LDA&5:SBC&3:STAbuf+5 ; Calculate
free memory
1220 LDA#31:JSRV:LDA#0:JSRV:JSRV ; Put
 cursor at 0,0
1230 JSRtext ; Routine to print text b
etween JSR and NOP instructions
1240 ]
1250 A$="[FM=&":$P%=A$
1260 P%=P%+LEN(A$)
1270 [OPT A:NOP
1280 LDAbuf+5:JSRshift
1290 LDAbuf+5:AND #&F:JSR disp
1300 LDAbuf+4:JSRshift
1310 LDAbuf+4:AND #&F:JSR disp ; Outpu
t free memory in Hex
1320 LDA#93:JSRV:LDA#32:JSRV:JSRV
1330 JSRV:JSRV
1340 LDA#31:JSRV:LDAbuf+2:JSRV ; Resto
re cursor
1350 LDAbuf+3:JSRV
1360 .endit:PLA:TAX:PLA:TAY:PLA:JMP (b
uf) ; Return from routine
1370 RTS
1380 .disp
1390 CLC:CMP#10:BCC num:CLC
1400 ADC#55:JSRV:RTS ; Value is A-F
1410 .num:ADC#48:JSRV ; Value is 0-9
1420 RTS
1430 .text:PLA:STA&72 ; Prints text fr
om PC until NOP
1440 PLA:STA&73:LDY#0
1450 .text2:INC&72:BNEtext3
1460 INC&73:.text3:LDA(&72),Y
1470 CMP#&EA:BEQtext4:JSRV
1480 JMPtext2:.text4:JMP(&72)
1490 .shift:AND#&F0:LSR A:LSR A ; Get
hi-byte
1500 LSR A:LSR A:JSRdisp:RTS
1510 .buf:JMP0:JMP0:JMP0 ; Quick way o
f reserving 9 bytes!
1520 ]
1530 NEXT
1540 ENDPROC
1550 :
1560 DEFPROCfilesys
1570 A%=0:Y%=0
1580 A%=(USR&FFDA)AND&FF0000 DIV&10000
1590 IF A%=0 END
1600 IF A%<3 base=&B00
1610 IF A%>2 base=&900
1620 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BASIC SPEED CONTROVERSY - Several members

Contrary to the 'Faster Basic' hint in BEEBUG Vol.3 No.6, a subroutine is not always faster than a procedure. In fact their relative speeds depend on the program composition. When a GOSUB is executed the entire program is searched, line by line, for the destination line number. The locations of procedures are stored by Basic after the first call to the procedure. Subroutines close to the start of a program will therefore be found and executed very fast. Procedures are, after their first call, always accessed in the same time regardless of their position. It is not possible to say that one method is overall faster than the other.

MORE ROMS FOR YOUR BEEB

We report this month on some of the latest ROMs that you might be tempted to purchase for your Beeb.

```
Title    : Basic Extensions
Supplier : Micro Power
Price    : £19.95 inc. VAT and p&p
Reviewer : Alan Webster
Rating   : ***
```

The Basic Extensions ROM has been on sale for about six months now, and my first impression on receiving it was one of delight.

The ROM enhances the number of direct mode commands, and also adds more instructions to Basic (instructions that can be used without using the command line interpreter). It is supplied with a 44 page A5 booklet to tell you all about the 39 new commands.

Some direct mode commands that are provided include: CONT to continue a program's execution after an error, DTOB and BTOD which convert binary to decimal and vice versa, SECURE which 'locks' your machine until the correct password is typed, and VIEW which lists a file from tape or disc without harming the program in memory.

BTOD	VIEW	GPOP
CHANGE	VERIFY	KILL
COMPACT	WILDCARD	LOOP
CONT	CASE	LPRINT
DTOB	ENDCASE	MEMSHIFT
DUMP	ENDWHILE	OTHERWISE
FIND	ENDLOOP	ORIGIN
JOIN	ENDEXIT	RUN
MERGE	ENDIF	SETCOLOUR
REPLACE	EXITIF	SETTEXT
SHIFT	ELSEIF	SETGRAPHIC
STATUS	FPOP	WHILE
SECURE	FIP	WHEN

Basic Extensions

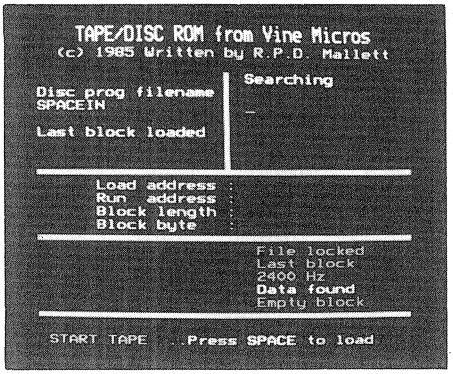
Some of the Basic language enhancements include CASE-ENDCASE, WHILE-ENDWHILE, WHEN, ENDLOOP, ENDEXIT, ENDIF and EXITIF for structured programming perfectionists, FPOP and GPOP to remove the last FOR-NEXT/RETURN address off the stack, while SETTEXT, *SETCOLOUR* and *SETGRAPHICS* are all to do with setting up windows and colours.

The operation of these new features is somewhat annoying as some of the abbreviated keywords take over from original Basic commands, which can cause frustration. Also, upon running a Micro Power games disc, we found that the software would not run properly, if at all, and had to disable the ROM beforehand.

Overall, the Basic Extensions package is worth having if you want the enhanced structured programming features, with the added bonus of various other commands for program development. The ROM is a good idea that offers some nice features, but would have benefitted from more time in the original design and planning. As it is, you will need to decide how important these extensions are for you.

```
Product : TD ROM
Supplier: Vine Micros
          Marshborough, Sandwich,
          Kent, CT13 0PG.
          0304-812276
Price   : £18.00
Reviewer: Geoff Bains
Rating  : **
```

TD ROM is an unusual device that promises to be a boon for disc-using games fans. The TD stands for 'tape to disc'. That is what this ROM is all about. It transfers your cassette programs to disc ready for near instant loading. To avoid the wrath of the software houses, TD ROM also has to be present in your machine when the game is loaded back into the machine from disc for use. In fact, Vine micros claim that clever random batch differences mean that the same TD ROM that saved the program must be the one to load it.



Legal difficulties aside, TD ROM is certainly simple to use. Typing *TD summons forth a menu giving you the choice of loading and running a program transferred previously, seeing the comprehensive on screen instructions, or transferring a new program to disc.

The transferral procedure is also easy to follow. You can either specify the number of tape files that make up the game and leave it to get on with it or opt to decide which is the last file when each file is loading.

TD ROM coped well with most games tried but has no chance with any software that alters the cassette filing system in any way. An increasing number of games are resorting to such methods of protection - Fortress, Blitzkrieg, Dune Rider, Starmaze, and Manic Minor amongst them. More worryingly the ROM seemed to have an adverse effect on the efficiency of the cassette interface. Several programs (notably Software Invasion games) that load successfully normally, refused to load under the auspices of TD ROM for transfer to disc.

The TD ROM works well within these limitations. However a more comprehensive version, though unlikely, would be preferable.

Title : Epson NLQ
Supplier : Watford Electronics
Price : £24.15 inc. VAT and p&p
Reviewer : Alan Webster
Rating : ****

Watford Electronics EPSON NLQ ROM V1.0

This is the normal EPSON dot matrix print.

As you can see, there is quite a difference between this type of printing and the NLQ printing. NLQ can also underline and enlarge.

Another feature of NLQ is: Proportional Spacing!

And you can have them altogether!

The new NLQ (Near Letter Quality) ROM is a simple and easy to use ROM which is intended to provide high quality printout (almost like a diasy-wheel) from your Epson printer. The three printers supported are the FX80, RX80 and the FX100.

To access this high quality print, you need to type two simple commands. Firstly, *NLQ80 or *NLQ100 are used to set the number of characters per line, and secondly, to activate the print routine, you use *NLQTYPE, VDU1,129 or OC129 (in Wordwise). To use the NLQ with View you must buy an NLQ driver from Watford at an additional cost of £7.50.

The printout from an Epson using this ROM is, as you can see from the example, much better than the normal Epson print. The print quality is now very close to that of the Kaga Taxan printer reviewed in BEEBUG Vol.3 No.8. The printing speed is somewhat slower than normal, but this is to be expected as each line is printed in two passes.

Overall, this is a good piece of firmware that makes good use of the Epson's printing capability to provide a performance comparable to that of more recent printers.

Product : Floppywise
Supplier: Software Services
65 South Mossley Hill Road,
Allerton, Liverpool, L19 9BG.
051-427 7894
Price : £29.95
Reviewer: Geoff Bains
Rating : ***

You'd be forgiven for thinking that there are not enough unimplemented utilities for the Beeb to fill yet another ROM. However, Software Services don't agree. Floppywise is a collection of some 14 utilities, mostly concerned with discs, and is claimed to work with any Acorn compatible DFS.

Some of the utilities have a familiar sound - FORMAT and VERIFY - and others are variations on a well known theme - MCOPY will copy more than one file from disc to disc and RCOPY will rename and copy a file all in one operation.

There are some novel ideas too. CONVERT will change a 40 track disc to 80 tracks, retaining all the data. AUTOSAVE will automatically save your program every four minutes as a backup. ASCII will display the codes for all the characters available on the Beeb. Several commands are involved with protection. You can use Floppywise to create protected discs and to backup commercial protected software.

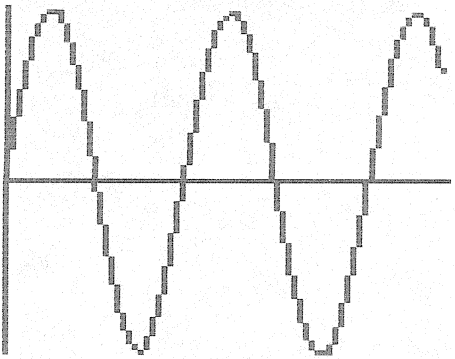
Floppywise is fully tube compatible and also adopts BEEBUG's letter prefix option to avoid *command conflict.

Floppywise is useful but not indispensable. At nearly £30 it is also pretty expensive. However, if you feel you need this sort of ROM, Software Services are offering it for £26.95 (plus £1 p&p) to Beebug members.

```
Title    : Dump Out 3
Supplier : Watford Electronics
Price    : £26.45 inc. VAT and p&p
Reviewer : Alan Webster
Rating   : **
```

Dump Out 3 is yet another screen dump ROM for the BBC which deals with all graphics modes (0,1,2,4,5,7 and 8 claims the manual), and incorporates a fast text only dump.

Dump Out 3 will work with printers from Seikosha, Epson, NEC, Shinwa, Star and Tandy and it allows you to use mode 7 as a (chunky) graphics screen, using two new OSWORD calls within the ROM. These commands are to read and plot mode 7 graphics pixels.



At this point in the manual, the reading becomes rather clouded and will, I'm sure, put some people off.

The speed of the dump is something else. It is the slowest graphics and text dump (in machine code) that I have seen to date. The whole ROM is just a scaled down version of Computer Concepts Printmaster, but with mode 7 plotting added.

If you haven't got a printer dump on ROM, then this is worth considering along with Printmaster. Check which features of each ROM suit your needs best and make your decision on that.

```
Product : Beebed
Supplier: J & O Software
          38 Hadden Way, Greenford,
          Middx., UB6 0HD.
Price   : £30.00
Reviewer: Geoff Bains
Rating  : ****
```

A major failing of the Beeb when compared to other home micros these days is its Basic editing facilities. Beebed hopes to right the balance a little with a full screen editor for BBC Basic programs.

Beebed bears a striking resemblence to Wordwise in style and use. Typing *Beebed puts your machine into edit mode. A mode 7 screen displays a page of your program which can be edited by moving the cursor to the section requiring attention and just typing in the corrected version. The function

→49

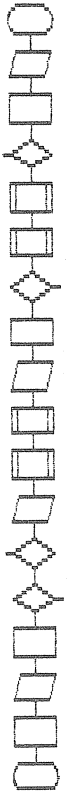
FLOWCHART GENERATOR

Flowcharts can provide invaluable aid when developing more complex programs and also provide a useful form of documentation. Nigel Balchin describes a utility which harnesses the graphics power of the Beeb to generate flowcharts on the screen for subsequent printing.

Flowcharts can be a most useful aid to the development of programs, particularly more complex ones, and they also have a role to play in documenting the structure and logic of a program once it has been fully developed and tested. You never know when you might want to modify an old program, or correct some unexpected bug, and without some kind of help it is all too easy to forget how a program functions even if totally clear at the time of writing.

That's where this utility will prove so handy. It allows a flowchart to be quickly and easily drawn on the screen including any text that you want to include as well. A flowchart can be saved to cassette or tape for future reference, and recalled at any time for further modification or for outputting to a dot matrix printer.

It would also be quite feasible to replace the flowchart symbols by others that can be laid out on a rectangular grid such as circuit diagrams. See the program notes for more detail.



USING THE FLOWCHART GENERATOR

The program is entirely in Basic and should be entered and saved to cassette or disc as usual. If you are going to run the program on a disc system (or other system with PAGE set higher than the &E00 of cassette systems) you will need to set PAGE to &1200 before loading and running the program (or use a suitable move-down routine).

GRID SIZE

The screen is divided up into a rectangular grid to assist construction of a flowchart. You can select the size of grid required by entering a whole number in the range 1 to 16 when asked. The relationship between number and size of grid is shown in the Table 1. Intermediate values will will generate proportionally equivalent grid sizes. In practice you will probably find sizes 2 or 4 the most useful.

Size	Grid dimensions
1	24 x 40
2	12 x 20
4	6 x 10
8	3 x 5
12	2 x 3
16	1 x 2

Table 1

The screen will then be cleared and a cursor (*) displayed to mark the top lefthand grid position. If you want to see the grid more clearly, Shift-3 acts as a toggle switch, drawing and removing actual grid lines on the screen. When drawing flowchart symbols and at other times the grid, if visible, will be cleared before proceeding and then restored after. As this takes a little time you are recommended to use the grid lines sparingly and remove them from the screen (using Shift-3) before selecting flowchart symbols or entering text.

DISPLAYING FLOWCHART SYMBOLS

The range of flowchart symbols built into the program is shown in Table 2 (produced using this program). The symbols are broadly arranged in pairs using a function key with or without Shift. A few symbols use the Ctrl key and a function key. To create a flowchart, simply use the cursor keys to move to the desired grid position and then select the symbol.

At any time, a previously drawn symbol can be deleted by moving to its grid position and pressing Delete. Note that two or more symbols drawn in the same position simply overlap.

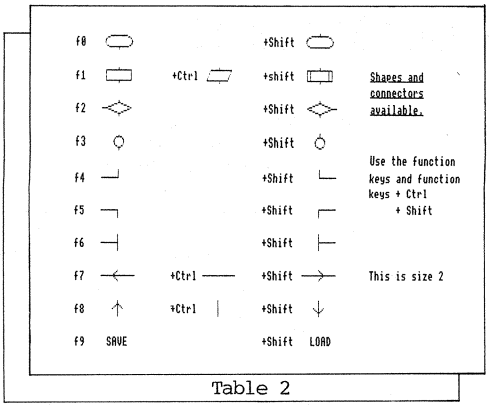


Table 2

ENTERING TEXT

You enter 'text mode' by pressing Shift-2 (i.e."). The cursor (*) is moved to the top lefthand corner of the screen and can be moved in character increments with the cursor keys. Text can be entered anywhere on the screen, and characters deleted with the Delete key. This can leave 'holes' in the flowchart symbols but the damage can be repaired by redrawing the flowchart symbol later. The problem can also be avoided by placing the cursor over the character to be deleted and retyping the character. You leave text mode by pressing either Return or Escape.

SAVING AND LOADING FLOWCHARTS

Saving and loading of flowcharts is accomplished with f9 and Shift-f9 respectively. In each case simply enter the relevant filename. Be careful as the program does not check for situations such as overwriting existing files or seeking non-existent files. The grid size is saved along with the flowchart so that when you reload a screen the grid size is automatically readjusted if necessary.

PRINTING FLOWCHARTS

Hardcopy output of the displayed flowchart is selected by pressing the Copy key. The program assumes that a suitable screen dump has previously been loaded at &D00 and that the entry point is &D02 (see line 2440). Alternatively you could replace this line by a call to a printer dump (we used Dumpmaster from BEEBUGSOFT), and you can similarly print out any flowcharts previously saved to cassette or disc. Examples of flowcharts printed in this way accompany this article.

EXIT FROM THE PROGRAM

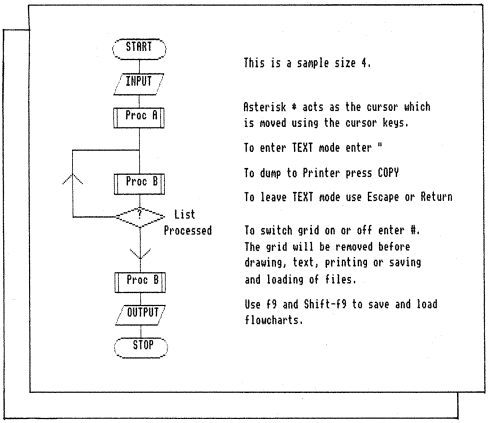
You can exit from the program by pressing Escape. This also gives you the opportunity of clearing the screen ready to draw a new flowchart. This option is also available when you select printer output.

PROGRAM NOTES

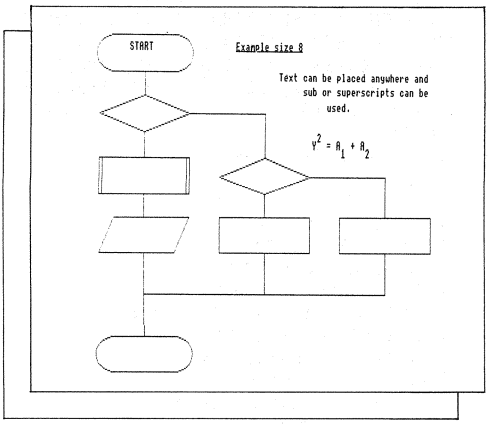
The function keys are set in lines 120 to 140 to generate ASCII codes. The Ctrl-function key combinations with values 181, 183 to 187, and 189 are currently unused and could provide additional flowchart symbols. Alternatively, as mentioned earlier, all the keys could be set to generate quite different symbols. Each function key combination is converted to a number in the range 1 to 30 used in an ON-GOSUB statement at line 1250. In turn this calls a corresponding procedure (see lines 1280 to 1510). By examining the existing procedures, and by experimenting a little, you should not find it too difficult to write some new procedures or alter existing ones.

All the procedures have names which make their functions largely self-explanatory. Note how HIMEM is adjusted at line 170 to protect the byte immediately below mode 0 screen memory, used to hold the grid size when saving and loading screens. This is handled by PROCsaveload (see line 2820 onwards) which uses the OSCLI call of &FFF7.

```
  10 REM Program FLOWCHT
  20  REM Version B2.2
  30 REM Author Nigel Balchin
  40 REM BEEBUG May 1985
  50 REM Program subject to copyright
  60 :
 100 MODE7
 110 ON ERROR GOTO 210
 120 *FX225,161
 130 *FX226,171
 140 *FX227,181
 150 VDU23,1,0;0;0;0;
 160 PROCtitle:PRINTTAB(5,22)CHR$136CH
R$129"Press the space bar to begin.":RE
PEAT:A%=GET:UNTIL A%=32
 170 MODE 0:HIMEM=HIMEM-1
 180 PROCinitialise:PROCmainloop:MODE7
:PROCresetmachine
```




```
 190 END
 200 :
 210 ON ERROR OFF:MODE 7:REPORT:PRINT"
at line ";ERL:PROCresetmachine:END
 220 :
1000 DEF PROCinitialise
1010 DIM string% 30
1020 *FX4,1
1030 *FX11,0
1040 *FX229,1
1050 GCOL4,0:GCOL4,129:COLOUR0:COLOUR1
29:CLG
1060 PROCsize:finished=FALSE:G%=FALSE
1070 ENDPROC
1080 :
1090 DEF PROCmainloop:VDU5
1100 REPEAT
1110 MOVE GDXP*HS*2,GDYP*HS:PRINT"*"
1120 A%=GET:MOVE GDXP*HS*2,GDYP*HS:PRI
NT"*"
1130 IF A%=27 THEN PROCgridtest:PROCer
rorcheck:GOTO 1260
1140 IF A%=136 THEN GDXP=GDXP-1:IF GDX
P=0THEN GDXP=XMAX
1150 IF A%=137 THEN GDXP=GDXP+1:IF GDX
P>XMAX THEN GDXP=1
1160 IF A%=138 THEN GDYP=GDYP-1:IF GDY
P=0 THEN GDYP=YMAX
1170 IF A%=139 THEN GDYP =GDYP+1:IF GD
YP>YMAX THEN GDYP=1
1180 IF A%=127 THEN PROCclear(GDXP,GDY
P)
1190 IF A%=135 THEN PROCdump
1200 IF A%=34 THEN PROCtext
1210 IF A%=35 THEN G%=NOT G%:PROCgrid
1220 IF A%<161 OR A%>190 THEN 1260
1230 XP=GDXP:YP=GDYP:A%=A%-160
1240 PROCgridtest:GCOL0,0:PROCbase(XP,
YP):VDU 29,XB-HS;YB+FS;
1250 ON A% GOSUB 1280,1300,1330,1350,1
390,1400,1470,1440,1410,1500,1290,1310,
1340,1360,1380,1370,1480,1450,1420,1510
,1520,1320,1520,1520,1520,1520,1520,146
0,1430,1520,1520:VDU29,0;0;:GCOL4,0:PRO
Cgridtest
1260 UNTIL finished
1270 ENDPROC
1280 PROCstart:RETURN
1290 PROCstop:RETURN
1300 PROCrect:RETURN
1310 PROCprocfn:RETURN
1320 PROCinout:RETURN
1330 PROCdecision("L"):RETURN
1340 PROCdecision("R"):RETURN
1350 PROCtopcon:RETURN
1360 PROCbotcon:RETURN
1370 PROCrightdown:RETURN
1380 PROCrightup:RETURN
1390 PROCleftup:RETURN
1400 PROCleftdown:RETURN
1410 PROCvertical("U"):RETURN
1420 PROCvertical("D"):RETURN
1430 PROCvertical("N"):RETURN
1440 PROChorizontal("L"):RETURN
1450 PROChorizontal("R"):RETURN
1460 PROChorizontal("N"):RETURN
1470 PROClefttee:RETURN
1480 PROCrighttee:RETURN
1490 PROCconnect:RETURN
1500 PROCsaveload("SAVE"):RETURN
1510 PROCsaveload("LOAD"):RETURN
1520 RETURN
1530 :
1540 DEF PROCclear(X,Y):PROCgridtest
1550 GCOL0,1:PROCbase(X,Y):VDU29,XB-HS
;YB+FS;:MOVE HS,NS:MOVE -HS,NS:PLOT 85,
HS,-FS:PLOT 85,-HS,-FS:GCOL4,0:VDU29,0;
0;:PROCgridtest:ENDPROC
1560 :
1570 DEF PROCtext
1580 *FX202,48
1590 *FX12,0
1600 PROCgridtest:XTS=2:YTS=(65*S)MOD
50+10+982:XT=XTS:YT=YTS
1610 REPEAT
1620 MOVE XT,YT:PRINT"*":REPEAT:A$=GET
$:UNTIL A$<>"":MOVE XT,YT:PRINT"*":A%=A
SC(A$)
1630 IF A%=136 THEN XT=XT-16:IF XT<XTS
 THEN XT=XTS
1640 IF A%=137 THEN XT=XT+16:IF XT>125
0 THEN XT=XTS:YT=YT-50:IF YT<50 THEN YT
=YTS
1650 IF A%=138 THEN YT=YT-25:IF YT<50
THEN YT=YTS:IF XT<1234 THEN XT=XT+16:IF
 XT>1250 THEN XT=XTS
1660 IF A%=139 THEN YT=YT+25:IF YT>YTS
 THEN YT=YTS
1670 IF A% =13 THEN 1710
```




```
 1680 IF A%>31 AND A%<127THEN MOVE XT,Y
T:PRINT CHR$(A%):XT=XT+16:IF XT>1250 TH
EN XT=XTS:YT=YT-50:IF YT<0 THEN YT=YTS
 1690 IF A%<>127THEN1710
 1700 MOVE XT,YT:PRINT CHR$(A%):XT=XT-1
6:IF XT<0 AND YT<=YTS-50 THEN YT=YT+50:
XT=1266 ELSE IF XT<0 THEN XT=XTS
 1710 UNTIL A%=13 OR A%=27
 1720 PROCgridtest
 1730 *FX11,0
 1740 *FX202,32
 1750 ENDPROC
 1760 :
 1770 DEF PROCgrid:FOR I%=0 TO YMAX:MOV
E30,HS*I%+10:PLOT21,XMAX*2*HS+30,HS*I%+
10:NEXT:FOR I%=0 TO XMAX:MOVE2*HS*I%+30
,10:PLOT21,2*HS*I%+30,YMAX*HS+10:NEXT
 1780 ENDPROC
 1790 DEF PROCgridtest:IF G% THEN PROCg
rid
 1800 ENDPROC
 1810 :
 1820 DEF PROCstart:PROCoval:MOVE 0,-TS
:DRAW 0,-FS:ENDPROC
 1830 :
 1840 DEF PROCstop:PROCoval:MOVE 0,TS:D
RAW 0,NS:ENDPROC
 1850 :
 1860 DEF PROCoval:MOVE -FS,-TS:DRAW FS
,-TS:PROCsemi(1):DRAW -FS,TS:PROCsemi(2
1):ENDPROC
 1870 :
 1880 DEF PROCsemi(CT)
 1890 IF CT=1 THEN XP=FS:YP=0 ELSE XP=-
FS:YP=0
 1900 FOR I=CT TO CT+19:DRAW XP+TS*SIN(
RAD(9*I)),YP-TS*COS(RAD(9*I)):NEXT
 1910 ENDPROC
 1920 :
 1930 DEF PROCrect:MOVE 0,NS:DRAW 0,TS:
DRAW -SS,TS:DRAW -SS,-TS:DRAW SS,-TS:DR
AW SS,TS:DRAW 0,TS::MOVE 0,-TS:DRAW 0,-
FS:ENDPROC
 1940 :
 1950 DEF PROCprocfn:PROCrect:MOVE 10-S
S,TS:DRAW 10-SS,-TS:MOVE SS-10,-TS:DRAW
 SS-10,TS:ENDPROC
 1960 :
 1970 DEF PROCinout:MOVE 0,NS:DRAW 0,TS
:DRAW -FS,TS:DRAW -SS,-TS:DRAW FS,-TS:D
RAW SS,TS:DRAW 0,TS:MOVE 0,-TS:DRAW 0,-
FS:ENDPROC
 1980 :
 1990 DEF PROCbase(X,Y):XB=X*HS*2+30:YB
=(Y-1)*HS+10:ENDPROC
 2000 :
 2010 DEF PROCdecision(DIR$):MOVE 0,NS:
DRAW 0,TS:DRAW-SS,0:DRAW 0,-TS:DRAW SS,
0:DRAW 0,TS:MOVE 0,-TS:DRAW 0,-FS
 2020 IF DIR$="R"THEN MOVE SS,0:DRAW HS
,0 ELSE MOVE -SS,0:DRAW -HS,0
 2030 ENDPROC
 2040 :
 2050 DEF PROCtopcon:PROCconnector:MOVE
0,-TS:DRAW 0,-FS:ENDPROC
 2060 :
 2070 DEF PROCbotcon:PROCconnector:MOVE
0,TS:DRAW 0,NS:ENDPROC
 2080 :
 2090 DEF PROCconnector:XP=0:YP=0:MOVE
XP,YP-30*S:FOR I=1 TO 40:DRAW XP+S*30*S
IN(RAD(9*I)),YP-S*30*COS(RAD(9*I)):NEXT
:ENDPROC
 2100 :
 2110 DEF PROChorizontal(dir$):MOVE -HS
,0:DRAW HS,0
 2120 IF dir$="N" THEN 2150
 2130 MOVE 0,-TS:IF dir$="R" THEN DRAW
TS,0 ELSE DRAW -TS,0
 2140 DRAW 0,TS
 2150 ENDPROC
 2160 :
 2170 DEF PROCvertical(dir$):MOVE 0,NS:
DRAW 0,-FS
 2180 IF dir$="N" THEN 2210
 2190 MOVE -TS,0:IF dir$="U" THEN DRAW
0,TS ELSE DRAW 0,-TS
 2200 DRAW TS,0
 2210 ENDPROC
 2220 :
 2230 DEF PROCleftup:MOVE -HS,0:DRAW 0,
0:DRAW 0,NS:ENDPROC
 2240 :
 2250 DEF PROCleftdown:MOVE -HS,0:DRAW
0,0:DRAW 0,-FS:ENDPROC
 2260 :
 2270 DEF PROCrightup:MOVE HS,0:DRAW 0,
0:DRAW 0,NS:ENDPROC
 2280 :
```

```
 2290 DEF PROCrightdown:MOVE HS,0:DRAW
0,0:DRAW 0,-FS:ENDPROC
 2300 :
 2310 DEF PROCrighttee:MOVE 0,NS:DRAW 0
,-FS:MOVE 0,0:DRAWHS,0:ENDPROC
 2320 :
 2330 DEF PROCleftteе:MOVE 0,NS:DRAW 0,
-FS:MOVE 0,0:DRAW -HS,0:ENDPROC
 2340 :
 2350 DEF PROCconnect:IF X<XMAX THEN MO
VE 0,-FS:DRAW HS*2,-FS:ENDPROC
 2360 :
 2370 DEF PROCwait:PRINTTAB(26,0)"Press
 space bar to continue":REPEAT:UNTIL GE
T$=" ":ENDPROC
 2380 :
 2390 DEF PROCdump
 2400 PROCgridtest
 2410 IF NOT FNyesno(300,1020,"Do you h
ave a printer ready (Y/N) ?") THEN 2440
 2420 *FX21,3
 2430 CALL&D02
 2440 PROCerrorcheck
 2450 ENDPROC
 2460 :
 2470 DEF PROCerrorcheck
 2480 finished=FNyesno(300,1020,"Have y
ou finished (Y/N) ?")
 2490 IF finished THEN 2520
 2500 IF FNyesno(300,1020,"Clear screen
 (Y/N) ?") THEN GCOL0,129:CLG:PROCsize:
GCOL4,1
 2510 PROCgridtest
 2520 ENDPROC
 2530 :
 2540 DEF FNyesno(X%,Y%,msg$)
 2550 MOVE X%,Y%:PRINT msg$;
 2560 REPEAT:A$=GET$:UNTIL INSTR("YyNn"
,A$)
 2570 MOVE X%,Y%:PRINT msg$;
 2580 IF A$="Y" OR A$="y" THEN =TRUE EL
SE =FALSE
 2590 :
 2600 DEF PROCresetmachine
 2610 *FX4,0
 2620 *FX12,0
 2630 *FX229,0
 2640 *FX225,1
 2650 *FX226,128
 2660 *FX227,144
 2670 VDU23,1,1;0;0;0;
 2680 ENDPROC
 2690 :
 2700 DEF PROCtitle:FOR I%=0 TO 1:PRINT
TAB(9,I%+5);CHR$131CHR$141;"Flowchart G
enerator":NEXT:PRINTTAB(18,9);CHR$134;"
by"
 2710 FOR I%=0 TO 1:PRINTTAB(10,I%+12);
CHR$131CHR$141;"Nigel J. Balchin":NEXT:
ENDPROC
 2720 :
 2730  DEF PROCsize:VDU4
 2740 REPEAT:CLS:PRINTTAB(5,5)"Enter si
ze please (1 TO 16) and press Return ";
:INPUTS:UNTIL S>=1 AND S<=16:S=S/4:CLS:
VDU5:PROCconstants
 2750 ENDPROC
 2760 :
 2770 DEF PROCconstants
 2780 XMAX=INT(6/S):YMAX=INT(10/S):GDXP
=1:GDYP=YMAX
 2790 HS=100*S:TS=30*S:FS=50*S:ES=80*S:
NS=49*S:SS=75*S
 2800 ENDPROC
 2810 :
 2820 DEF PROCsaveload(F$):GCOL4,0:VDU2
9,0;0;
 2830 MOVE400,1010:INPUT"Filename:" fil
e$
 2840 MOVE400,1010:PRINT"Filename:";fil
e$
 2850 IF F$="SAVE" THEN ?&2FFF=INT(4*S)
:$string%=F$+CHR$32+file$+" 2FFF,8000"
ELSE $string%=F$+CHR$32+file$
 2860 X%=string% MOD 256
 2870 Y%=string% DIV 256
 2880 CALL &FFF7
 2890 IF F$="LOAD" THEN S=?&2FFF/4:PROC
constants
 2900 ENDPROC
```

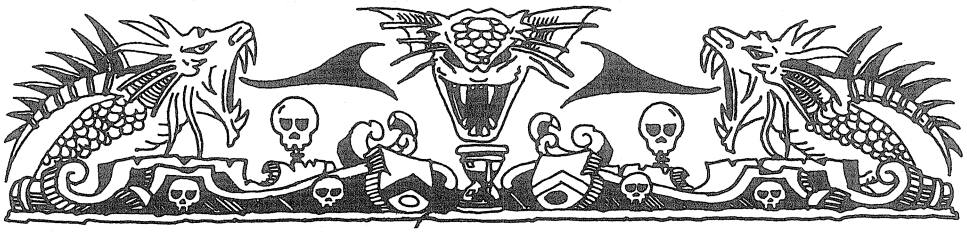
HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

QUITTING *EXEC - Bill Walker

Normally a *EXEC file that runs a series of programs in sequence, as boot files often do, will still try to run the third program (for example) even after there has been an error in the second. This can be avoided by incorporating a CLOSE#0 in any error trapping routine in each program. This will close the *EXEC file and halt the process.

LISTING Z80 UNLIST - Chang Sing Pang

Owners of the Z80 second processor who accidentally make a Z80 Basic program unlistable with the UNLIST.COM utility can restore it again by converting it to a 6502 Basic file with the DIP utility and then loading the file into the host processor with the Z80 switched off. The program is now listable.

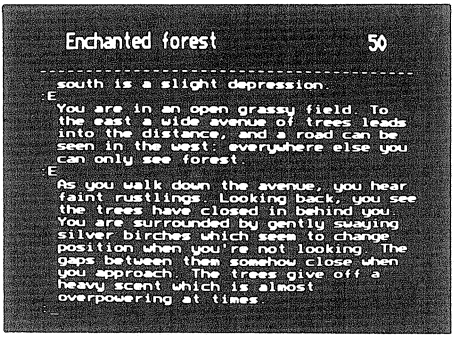


ADVENTURE GAMES

by Mitch

Acornsoft continue to disturb the peace of the BEEBUG dungeon with a new batch of games, which have caused more than a little friction between the inhabitants.

Acheton produced by Acornsoft on disc only for £17.95.



At long last a large, disc-based adventure has arrived. Cassette owners eat your heart out! This game resides on two discs which hold the program and database respectively. Locating all the data on disc enables this game to have all the subtleties and power of a mainframe adventure. Those plutocrats among you who possess 80 track drives will be pleased to note that there is a command to reconfigure the disc to this format.

The game is text only and although the instruction sheet mentions the use of colour somewhere in the game, I have not yet seen any.

The game is set in a vast cave system whose tunnels have more twists and turns than an Editor's mind! Being an adventurer of the old school, who scorns map making, I quickly became lost! The game has a 'Colossal Cave' feel to it, complete with the iron grating and black rod. The grating, however, is not all it initially seems; and, when waved, the black rod appears to have no effect. There are 350 locations to explore and 150 objects and treasures to find. If you are able to make enough progress you will be able to get through to the Master's section of the game (no I haven't!) Once inside this section there is no going back! In many of the rooms, coloured stars are to be found hanging in mid air, very reminiscent of Philosopher's Quest. I trust they are there for a reason and not for the confusion of brain-weary explorers.

I have spent almost fifteen hours on Acheton so far and have enjoyed every minute. There appears to be little restriction to your movement in this game, allowing you to wander far and wide. Vast chambers with curious rock formations, an underground harbour swarming with piranhas and a wizard's garden containing live gnomes, all blend together in this intriguing puzzle.

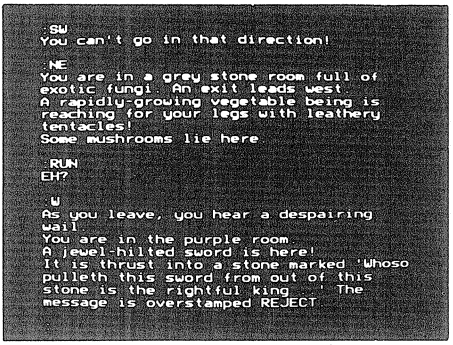
The new policy of Acornsoft is to provide a Hint and Answer envelope with all their games. This move has obvious advantages for Acornsoft but the temptation to open the envelope can be overwhelming. In addition to the envelope, this game will respond to *HELP commands with a clue number which may be used to index the hint sheet.

My one critisism is that as the room descriptions are held on disc, the drive is constantly being accessed

during the game. I am surprised that a larger batch of descriptions are not transfered to memory at the same time, which would have meant many fewer disc movements and reduced my winces considerably!

As the disc drive is materializing in more and more dungeons of late, I suspect this game will be welcomed with open arms by many wizards. I have no reservations about this one; go get it!

Quondam produced by Acornsoft on cassette for £9.95 and on disc for £14.95.



Quondam boasts on the box of being a game for 'advanced' adventurers. To prove the obvious foolishness of this claim I immediately gave the sealed 'Hints and Answers' envelope to the idiot troll and forbade him to reveal the contents until I returned victorious with the final solution.

I hate that troll! I've never realised it before but he has a very nasty smirk. Anyway, I didn't like the look of this game from the start, what kind of an adventure starts with a maze? You know I was never any good at mazes! It won't let you save when you want to, and it even sends the Mafia round if you upset it!

Having taken an hour to map the first maze I then met a very belligerent knight who prevented any further movement, thus forcing me to quit.

The game features caves, magic, dragons and the aforementioned Mafia. A friendly passing wizard has informed me that this game is, of course, a skit on the banking system (of course). You will quickly find that you may not save the game whenever you feel like it, as the gentlemen with the dark glasses and knuckledusters don't like that!

A sneaky feature of Quondam is that some commands appear to work first time, but in fact need to be repeated several times to achieve the final effect. There is also a loathsome custom official who will eat you should you attempt to pass him with anything he considers is contraband.

Of course one small peek in the envelope would be all I would need to complete the game. That's all I need, one quick peek. I hate that troll!

For those giant-killers who are currently stuck within a 'Mysterious Adventure', help is at hand. Channel 8 Software who now market this range has set up a post and telephone help service. Either send a S.A.E to Channel 8 Software, 5 Fishergate, Preston, Lancs or phone 0772-562731 for instant advice.

Remember, if you have any puffs of magic which might be of use in the writing or playing of adventures, don't keep them to yourself - I need all the help I can get!

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

COMPACT WITHOUT TEARS - Peter Sewell

You can *COMPACT a disc without losing the program in memory as follows:

```
MODE 7
PRINT (TOP DIV 256)+1,PAGE DIV 256
```

(say this gives XXX and YYY)

```
*FX 180,XXX
*COMPACT
*FX 180,YYY
```

BEEBUG Workshop

SEARCHING AND SORTING (Part 2)

By Surac

This month's Workshop continues the theme of sorting techniques with particular reference this time to sorting character strings.

Last month we looked at ways of sorting data and ended up with a couple of useful PROCedures. As I hinted, though, there are problems when it comes to sorting strings.

The snag is the profligate way that BBC Basic allows space for each variable to hold the longest string it has ever held, regardless of its present value. This approach is appallingly wasteful compared to that of other computers. Most keep string space under control and, through a process known as "garbage collection", weed out unwanted space. But on the Beeb, we can have trouble if we try to sort a string array directly, particularly if the strings are of varying size. The strings could easily rampage through memory, ending up with the dreaded "No room" error message.

To avoid the problem, sort a set of pointers to the strings, rather than the strings themselves. Use a second array, which eventually holds, in order, the indices to the strings. For example, suppose that the 38th string should be first; the first element in the pointer array would hold the value "38". Here is a modified Shell sort to put a string array into alphabetical order.

```
                POINTER SORT
10000 DEF PROCstrshell(st%,fin%)
10010 LOCAL D%,F%,I%,S%,T%
10020 FOR I%=1 TO (fin%-st%)+1:ptr%(I%)
      =I%:NEXT
10030 S%=2^INT(LOG(fin%-st%)/LOG(2))
10040 REPEAT
10050   T%=fin%-S%
10060   REPEAT
10070     F%=FALSE
10080     FOR I%=st% TO T%
10090       IF array$(ptr%(I%))>array$(
            ptr%(I%+S%)) THEN PROCswap
10100       NEXT
10110     T%=T%-1
10120     UNTIL NOT F%
10130   S%=S% DIV 2
10140   UNTIL S%=0
10150 ENDPROC
10990 :
11000 DEF PROCswap
11010 D%=ptr%(I%)
11020 ptr%(I%)=ptr%(I%+S%)
11030 ptr%(I%+S%)=D%
11040 F%=TRUE
11050 ENDPROC
```

The routine assumes that you have already DIMmed the array ptr%() to have as many elements as "array$()". Line 10020 puts the pointers into numerical order and sorting starts, using ptr%() to access the strings. Note how PROCswap only exchanges pointers and does not directly affect the strings.

The pointer approach is also useful when you sort groups of related data. For instance, a list of names and addresses can be put into order without manipulating names AND addresses.

So far, though, we have assumed that all the data is in memory. What if we need to sort a too-big-to-fit disc file? The answer is remarkably simple and obvious (when you know...).

Split the large file into smaller ones which WILL fit. Sort each small one and save it back to disc. Then, and this only works on disc-based systems, read the data from the small files in parallel. Select the largest (or smallest, depending) of the values at

```
                GIANT FILE SORT
10000 DEF PROCfilsort(srtfile$,nitems%)
10010 LOCAL F1
10020 F1=OPENIN srtfile$
10030 PROCsort(nitems% DIV 2,"D.TEMP1")
10040 PROCsort(nitems%-(nitems% DIV 2),
      "D.TEMP2")
10050 CLOSE #F1
10060 PROCmerge("D.TEMP1","D.TEMP2",srt
      file$)
10080 *DELETE D.TEMP1
10090 *DELETE D.TEMP2
10100 ENDPROC
10190 :
10200 DEF PROCsort(n%,outfil$)
10210 LOCAL i%,f2
10220 FOR i%=1 TO n%:INPUT #F1,array
      (i%):NEXT
10230 PROCshell(1,n%)
10240 f2=OPENOUT outfil$
10250 FOR i%=1 TO n%:PRINT #f2,array
      (i%):NEXT
10260 CLOSE #f2
10270 ENDPROC
10390 :
10400 DEF PROCmerge(in1$,in2$,op$)
10410 LOCAL d1,d2,f1,f2,f3
10420 f1=OPENIN in1$:f2=OPENIN in2$:
      f3=OPENOUT op$
10440 INPUT #f1,d1:INPUT #f2,d2
10450 REPEAT
10460   IF EOF #f1 THEN PROCwrapup(d1,d
        2,f2):GOTO 10500
10470   IF EOF #f2 THEN PROCwrapup(d2,d
        1,f1):GOTO 10500
10480   IF d1<d2 THEN PRINT #f3,d1:
        INPUT #f1,d1 ELSE PRINT
        #f3,d2:INPUT #f2,d2
10490   UNTIL EOF #f1 AND EOF #f2
10500 CLOSE #f1:CLOSE #f2:CLOSE #f3
10510 ENDPROC
10590 :
10600 DEF PROCwrapup(d1,d2,filno)
10610 LOCAL d1val,d2val
10620 d1val=TRUE:d2val=TRUE
10630 REPEAT
10640   IF NOT d1val AND NOT EOF #filno
        THEN REPEAT:PRINT #f3,d2:INPUT
        #filno,d2:UNTIL EOF#filno:PRINT
        #f3,d2:d2val=FALSE:GOTO 10670
10650  IF NOT d2val THEN PRINT #f3,d1:
       d1val=FALSE ELSE IF d1<=d2 AND
       d1val THEN PRINT #f3,d1:
       d1val=FALSE ELSE PRINT #f3,d2:
       IF NOT EOF #filno THEN INPUT
       #filno,d2 ELSE d2val=FALSE
10660   UNTIL NOT d1val AND NOT d2val
10670 ENDPROC
```

the start of each small file, and write it to the large file. Continue like this, taking the wanted value from whichever small file holds it, until they are all empty; the original large file is then sorted.

It's like splitting a pack of cards into 4 hands, sorting each hand, and then taking cards from each hand, in order, to end up with a sorted pack. It's usually also faster than trying to sort a single file.

PROCfilsort starts with the name of the file to be sorted and the number of items in it. It calls PROCsort twice, halving and sorting the file into D.TEMP1 and D.TEMP2. These 2 files are merged, overwriting the original file, and then deleted.

PROCsort reads half the main file into "array()", which it Shell sorts - see last month's Workshop for the code. The sorted data is then written to the temporary file. NOTE: You will have to create "array()" with:

```
DIM array(nitems% DIV 2 + 1)
```

in the main program. The "+1" allows for nitems%' being odd.

The 2 sub-files are merged by repeatedly taking the smaller value from their tops (we're sorting into ascending order) and writing it to the original file. Eventually, we get to the end of one of the temporary files while still having data in the other.

At that point, PROCwrapup simply moves the remaining data from the non-empty file to the main file. In doing so, it slots the last item from the empty sub-file into its correct place. Line 10650 does the job; it is a horrible compound IF statement, of which I am not particularly proud. It works, however, and saves a lot of space. The 2 variables "d1val" and "d2val" are flags which show when data from each file is used up.

Although I have only used 2 sub-files, the DFS allows 5 files to be open at any time. You could, therefore, use up to 4 sub-files. This would make PROCwrapup even more complex, however. What would you do if the original file needed more than 4 sub-files?

Demonstrations of both sort procedures are included on this month's magazine cassette/disc.

MAKING MUSIC ON THE BEEB (Part 4)

This month's music article from the author of "Making Music on the BBC Micro" continues to explore the musical – and not-so-musical – potential of computerised multi-part tunes.

The performance of a two-, three- or four-part piece of music on the Beeb is quite an achievement in itself. If you have been following this series you should be well on your way to programming multi-part tunes - other than Mozart's Rondo Alla Turca. As with the book, the ideas and programs presented in this series of articles are intended as a springboard for further experiments of your own. If you want to rewrite a routine or compact a piece of code then please do so. You'll learn more and get more satisfaction by performing your own experiments on the routines then by just typing in the programs and running them - although I hope you find them entertaining, too.

THE ENVELOPES AS A SYNTHESIZER

One of the simplest ways to begin your experiments is to increase the number of envelopes in a piece of music. Last month's piece only used three envelopes but if you think of each envelope as a different instrument or as a synthesizer setting then you can greatly alter the performance. As a simple example: last month's program used percussive envelopes to give a piano/harpsichord effect. If you substitute ENVELOPE2 for ENVELOPE1 and ENVELOPE3 it will give the piece a more string-like quality.

MANIPULATING THE MUSIC DATA

Because the music data is stored as a series of numbers the computer can manipulate it quite easily. Even if we can't quite call the results of these manipulations original, they will certainly give a new slant to the piece. We could begin our experiments with a simple one-part tune but as we already have a three-part tune from last month and as multi-part tunes are far more interesting to listen to than single part tunes we will jump straight in with...

THE AMAZING ONE LINE WONDER COMPOSER PROGRAM

An interesting mathematical manipulation is to turn the tune upside-down. This One Line Wonder will reverse the pitch of the notes so that high notes will be played low and low notes will be played high. Insert this line into last month's program:

```
1105 Pitch=77+77-Pitch
```

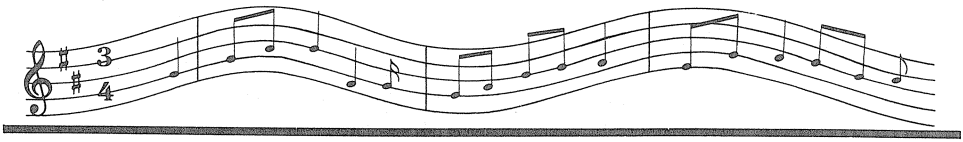
77 is the pivot point, the pitch between the highest and lowest pitches. It can be found by looking up the highest and lowest notes of the tune in the diagram in BEEBUG Vol.3 No.8 p.16 and then finding the note halfway between the two. You can make the computer do the work for you by adding these lines to last month's program:

```
   1 HiP=0:LoP=255
1101 IF Pitch>HiP HiP=Pitch
1102 IF Pitch<LoP LoP=Pitch
```

When the computer has run through the data you can print out HiP and LoP in command mode. The pivot point will be:

```
(HiP-LoP)/2+LoP
```

You will sometimes find, as in the case of Rondo, that the pivot pitch falls between two notes. The value you use does not have to be an exact note pitch and you can experiment by raising and lowering the value. The duration has



not been altered which is what makes the tune sound half-recognisable but you could apply a similar function to the duration. Try some of these expressions in line 1105:

```
Pitch=Pitch/4*3
Pitch=Pitch/4*2
Pitch=Pitch/4
Pitch=Pitch/4*5
Pitch=Pitch/4*6
Pitch=Pitch/4*7
```

If the manipulations becomes too extreme the pitch will loop over the top or under the bottom and this starts to happen in the last example. Try using algebraic expressions on the pitch values; you may discover a whole new method of composition.

BACKWARDS TUNES

With the note data arranged serially in arrays we can play a tune backwards by simply reading the data backwards. Make the following alterations to last month's program:

```
680 Ch1=C1:Ch2=C2:Ch3=C3
690 :
700 REPEAT
710 IF ADVAL(-6)>0 AND Ch1>0 Ch1=Ch1-
1:SOUNDChan1(1,Ch1)+1,Chan1(2,Ch1),Chan
1(3,Ch1),Chan1(4,Ch1)*Tempo
720 IF ADVAL(-7)>0 AND Ch2>0 Ch2=Ch2-
1:SOUNDChan2(1,Ch2)+2,Chan2(2,Ch2),Chan
2(3,Ch2),Chan2(4,Ch2)*Tempo
730 IF ADVAL(-8)>0 AND Ch3>0 Ch3=Ch3-
1:SOUNDChan3(1,Ch3)+3,Chan3(2,Ch3),Chan
3(3,Ch3),Chan3(4,Ch3)*Tempo
740 UNTIL Ch1=0 AND Ch2=0 AND Ch3=0
```

To make it sound more like a true backwards recording, change the percussive envelopes for ones with a slowish attack and fast release. You can start experimenting with these:

```
ENVELOPE1,1,0,0,0,0,0,0,2,-4,0,0,126
,0
ENVELOPE2,4,0,0,1,1,0,1,6,-32,0,0,12
6,0
```

Set ENVELOPE3 equal to ENVELOPE1. Finally, to complete the transformation, add the one line upside-down routine.

Instead of altering the program as we have been doing you could include these variations as separate procedures and present the user with a menu of 'transformations' to choose from. Once run, a tune can be played again by entering:

```
GOTO 680
```

CANDIDATES FOR CORRUPTION

Some tunes take to this sort of treatment much better than others. Rondo is basically a tune played on channel 1 with an accompaniment played on the other two channels. Bach-type pieces consisting of two or three interwoven melodic lines will tend to produce better melodic results. The transformation of other tunes can often be quite humerous. One of the most effective is the upside-down version of Sousa's Liberty Bell March better known perhaps as the Monty Python theme, which is listed in the book.

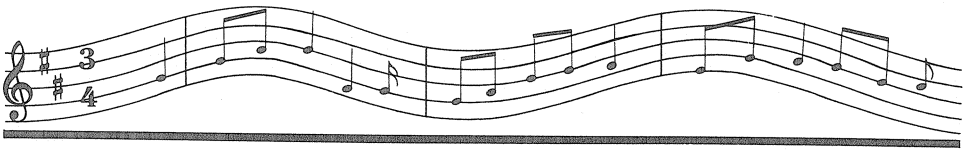
With these experiments we are just dabbling on the very fringes of computer composition but they do illustrate the power and potential of the computer in manipulating music data. Such results as we have achieved would be, if not impossible by 'hand' then certainly laborious in the extreme.

SAVING THE TUNE

If you want to play any music in a separate program, for example as background to a graphics display, you can use the program to check out the data values, i.e. see the music plays as it should, and use this SOUND-ready data in place of the note analysis routines. The data for a three-channel piece will be stored in three arrays:

```
Chan1(4,C1),Chan2(4,C2),Chan3(4,C3)
```

You can save this data to tape or disc using file handling procedures which are explained in the User Guide on page



188. You could read the saved data into the arrays thus bypassing the note analysis routines. You could expand upon this idea by creating a master program which would load and play any existing music files previously saved. You would also need to load relevant envelopes with the music data which could be done by *SAVEing and *LOADing the envelope storage area. Envelope storage begins at &8C0 and each envelope takes up &10 (16) bytes (only 13 bytes are actually used but the envelope locations are incremented in steps of &10). Having defined envelopes 1 to 4, you can save them like this:

```
*SAVE"ENV4" 8C0+40
```

They can be loaded again by:

```
*LOAD"ENV4"
```

In order to store large amounts of data, you could utilize byte arrays (using indirection operators) but I will leave this sort of development for interested readers to experiment with.

Another routine is presented here which will save a music file to tape or disc. This can be *EXECed back into the Beeb and saved as a normal program. The routine calculates the order in which note data needs to be presented to the SOUND command in order for the tune to play in sync. In other words, we can replace lines 680 to 740 with a simple read-note-and-play-it routine which takes its note information directly from DATA statements without first needing to analyse it and store it in an array.

USING THE ROUTINE

Type in the program exactly as it appears and save it. Save it also as an ASCII file by entering:

```
*SPOOL"SAVER"
LIST
*SPOOL
```

Load last month's program and add SAVER to it by entering:

```
*EXEC"SAVER"
```

This will merge the two programs. Set Tempo to the appropriate value - see line 6. Run the program and you will be prompted to insert a disc or tape. Do so and press RETURN. The tune will play, perhaps somewhat hesitantly, and a stream of data will scroll up the screen. When the cursor appears again you will have a file called TUNE on tape or disc. Delete all the lines up to 10000 by entering:

```
DELETE 1,1440
```

where 1440 is the last line of the original program. Reposition the tape if you're using tape and enter:

```
*EXEC"TUNE"
```

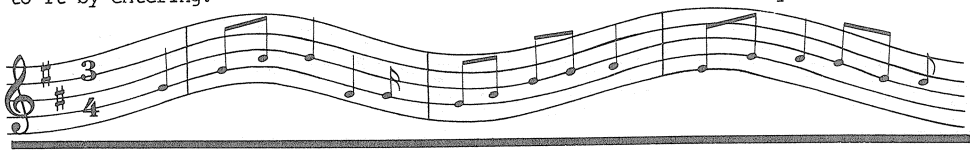
Lines of data should scroll up the screen. If you now run the program the tune should play in perfect sync. The routine beginning at line 10000 plays the data. If you substitute 'SOUND' for the string 'DATA' in lines 1113, 1118 and 1123 the *SPOOLed programs will play the tune when run without the need of lines 10000 to 10080. In this case enter NEW before *EXEC"TUNE". This is discussed further under Program Notes.

Don't forget, if you save this program to use later you will have to include envelope definitions in it somewhere.

The programs are from Making Music on the BBC Computer by Ian Waugh, published by Sunshine Books at £5.95 and used with kind permission of the publishers.

PROGRAM NOTES

The program cheats a little because instead of calculating the correct order of the data by hand - or chip - as it were, it uses the ADVAL statements to calculate the correct spacing of the notes exactly as it does when playing the tune. However, because the filing systems themselves require attention from the operating system, the ADVAL function may sometimes want



to pass notes while the operating system is not able to give them. This will happen especially if a channel is supplied with lots of short notes.

This is why Tempo needs to be adjusted. It is used to slow down the playing of a piece so the sound channels do not empty while waiting for the filing system to finish with the O.S. Tape will obviously take longer than disc but experiments have shown that Tempo values of 5 for tape and 2 for disc will work for Rondo. Other tunes may require different values. On playback, the duration should be re-adjusted as in line 10070. In the PROCSpool procedures, you can remove the variable, Tempo, altogether and control the speed of the piece as we have done in line 10070. If, however, you substitute SOUND for DATA then you should ensure that the correct Tempo value is used in the PROCSpool procedures to produce the absolute note duration required.

```
    1 REM PROGRAM 9.3
    2 REM *SPOOL Routine To Put Sound
    3 REM Data Onto TAPE or DISC
    4 REM Include These Lines in
    5 REM PROGRAM 9.2
    6 REM Tempo=2 For DISC, 5 For TAPE
    7 VDU15
    8 :
  260 Tempo=2
  694 Line=5000
  695 PRINT"INSERT DISC OR TAPE then RE
TURN"
  696 REPEAT:A=GET:UNTIL A=13
  697 *SPOOL"TUNE"
  710 IF ADVAL(-6)>0 AND Ch1<C1 Ch1=Ch1
+1:SOUNDChan1(1,Ch1)+1,Chan1(2,Ch1),Cha
n1(3,Ch1),Chan1(4,Ch1)*Tempo:PROCSpool1
  720 IF ADVAL(-7)>0 AND Ch2<C2 Ch2=Ch2
+1:SOUNDChan2(1,Ch2)+2,Chan2(2,Ch2),Cha
n2(3,Ch2),Chan2(4,Ch2)*Tempo:PROCSpool2
  730 IF ADVAL(-8)>0 AND Ch3<C3 Ch3=Ch3
+1:SOUNDChan3(1,Ch3)+3,Chan3(2,Ch3),Cha
n3(3,Ch3),Chan3(4,Ch3)*Tempo:PROCSpool3
  745 *SPOOL
 1111 :
 1112 DEF PROCSpool1
 1113 PRINT;Line;" DATA ";Chan1(1,Ch1)+
1;",";Chan1(2,Ch1);",";Chan1(3,Ch1);","
;Chan1(4,Ch1)*Tempo
 1114 Line=Line+10
 1115 ENDPROC
 1116 :
 1117 DEF PROCSpool2
 1118 PRINT;Line;" DATA ";Chan2(1,Ch2)+
2;",";Chan2(2,Ch2);",";Chan2(3,Ch2);","
;Chan2(4,Ch2)*Tempo
 1119 Line=Line+10
 1120 ENDPROC
 1121 :
 1122 DEF PROCSpool3
 1123 PRINT;Line;" DATA ";Chan3(1,Ch3)+
3;",";Chan3(2,Ch3);",";Chan3(3,Ch3);","
;Chan3(4,Ch3)*Tempo
 1124 Line=Line+10
 1125 ENDPROC
 1126 :
10000 DATA -1,-1,-1,-1
10010 :
10020 REM These Lines Play the DATA
10030 RESTORE5000
10040 REPEAT
10050 READ Chan,Env,Pitch,Dur
10055 IF Chan=-1 THEN 10080
10060 REM Set Divisor Equal to increase
      in Tempo
10070 SOUNDChan,Env,Pitch,Dur/2
10080 UNTIL Chan=-1
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BLUE SCREEN IN WORDWISE - E. Williams

Include OS"FX155,36" in Wordwise (1.2 or later) text with a green command before it and a white command or a Return afterwards and the text will be previewed with a blue background. As the preview is in mode 3 this both makes the text easier to read on a colour TV/monitor and the margin and tab settings easier to judge relative to the edge of the blue screen.

CASSETTE SOFT LABELLING - B.R. Hill

You can place a soft label of up to ten characters at any point on your cassette by typing:

*SAVE "usefulword" 0 0

This saves a very short (!) program with your label as a filename on to the cassette. This will be displayed in the normal way when you are loading a file or cataloguing the cassette.

MIXING MODES (Part 2)

Ian Hall explains some of the more advanced and interesting techniques used in his mixed mode program that we published last month.

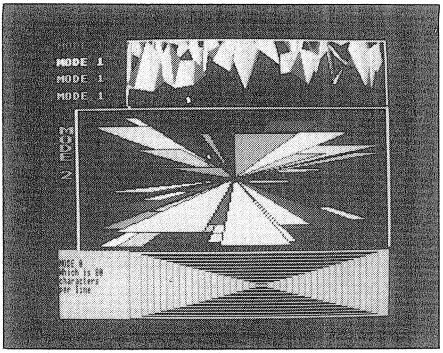
Last month's MIXMODE program is arguably the most advanced utility ever published by BEEBUG, or any other magazine for that matter. It gives your BBC micro three modes on the screen at once - 16 colours with 80 column text! This month the author describes just how it all works.

BEEB GRAPHICS

To display a screenful of text or graphics your BBC micro has to do two things. Firstly, the hardware that takes data from screen memory and actually displays it must be set up for the particular mode and, secondly, data has to be placed into the screen memory in the correct format for that mode. The bytes within the screen memory represent different things in each mode (see 'Machine Code Graphics', BEEBUG Vol.2 Nos. 8 to 10). For modes 0 and 4 one byte represents 8 pixels, for modes 1 and 5 one byte represents 4 pixels and for mode 2, one byte represents only 2 pixels.

On the hardware side, there are two devices that control the removal of data from the screen memory and convert it into a suitable form for display. These are the 6845 cathode ray tube controller (CRTC) and the video ULA. The device that does most of the work is the CRTC which is responsible for such things as producing the correct format for the display, positioning the cursor, and so on. The ULA determines such things as the relationship between logical and physical colours.

The CRTC has 18 registers and the ULA has two. The CRTC registers are set up differently for the 20K, 10K and text only modes. They are, however, the same for the all the 20K modes (0, 1 and 2). Similarly, they are the same for both 10K modes (4 and 5). The ULA registers, however, are different for each mode. To change between modes 0, 1 and 2, therefore, only the ULA registers have to be changed. The same goes for modes 4 and 5 .



To put graphics data into screen memory, the operating system relies on data stored within certain parts of memory to determine the format for the data used. The areas of memory that are used for this data (referred to as VDU variables) are between &300 and &37F, and zero page locations &D0 to &D9. They contain such data as the current mode, the current window information, cursor position, number of bytes per character and so on. Also of importance are two of the main system variables located at &248 and &249. These contain the data last sent to the video ULA register. See chapter 11 of the Advanced User Guide for further details of this.

HOW MIXMODE WORKS

To understand how the MIXMODE program of last month works, some knowledge is required of the way a TV picture is generated. Very simply, the TV picture is built up on the screen in a number of horizontal lines from top to bottom. This process, which is called a 'raster scan', takes 20ms to perform and is repeated continuously at a rate of fifty times a second. During this process the Video ULA and the CRTC are sequentially getting data from screen memory and converting it into a form to be sent along the cable to your TV.

MIXMODE operates by effectively changing the displayed mode (by altering the ULA registers) during the raster scan. By changing the displayed mode in this way, at exactly the same place in each scan, the screen appears to be displaying more than one mode. At the same time as this continual change of displayed mode is occurring, the program allows the user to select in which mode graphics data is to be placed in screen memory. The crucial factor is the timing of the displayed mode changes. This timing is undertaken using the VIA timers and interrupts.

Internally the Beeb generates an interrupt every time the raster scan starts at the top of the screen (this is called the start of vertical sync). The MIXMODE program uses this interrupt in conjunction with those generated by the timer within the User VIA.

THE VIAS

There are two VIA (versatile interface adaptor) chips in the Beeb - the internal VIA and the User VIA. Each is a device with two I/O ports, four discrete inputs which can be used for controlled interrupts to the 6502 processor, a serial register and two timers. The System VIA uses these facilities for the speech and sound system, internal hardware control, joystick fire buttons, light pen input, key pressed interrupt, analogue to digital conversion interrupt, the vertical sync interrupt from the video hardware, and the internal 100Hz clock.

Within the User VIA, only port A is used by the system, this being the printer port. Port B and all the other functions of the User VIA are free for the user.

The User VIA timer1 can be used to generate an interrupt after a set period by loading the 16 bit counter with a suitable value. The counter must be loaded with the low byte (at &FE64) followed by the high byte (at &FE65). This order ensures that when the high byte is loaded both bytes are placed into the counter from the latches at the same instant, so that you can guarantee that the counter has the desired value at the point where the high byte is loaded).

The interrupt generated when the timer reaches zero is enabled using the Interrupt Enable Register (IER) at &FE6E:

```
BIT 7 : Set or clear control bit
BIT 6 : timer1
BIT 5 : timer2
BIT 4 : CB1
BIT 3 : CB2
BIT 2 : SERIAL
BIT 1 : CA1
BIT 0 : CA2
```

The appropriate interrupt is enabled when bits 0 to 6 are set to one. These bits are set by writing a byte to the IER to set both the appropriate bit and bit 7 to one (the interrupt can be disabled by writing a one to the appropriate bit with bit 7 set to zero). In this case, to enable the timer1 interrupt, &C0 is poked to &FE6E.

The Interrupt Flag Register (IFR) has bits 0 to 6 related to the same functions as the IER. Bit 7, however, is used to indicate if an interrupt was generated by that particular VIA (ie, if any of bits 0 to 6 are set to one in both the IER and the IFR then bit 7 is set). Therefore, on receiving an interrupt, the program need only look at bit 7 first and, if set, the program can then check to see which of bits 0 to 6 have been set. In this case the check is for bit 6 (timer1) set.

THE PROGRAM

MIXMODE consists, for the main part, of the machine code program in lines 1400 to 3450. This is divided into two separate areas with lines 1400 to 2550 handling the interrupts and the control of the video hardware (which changes the displayed mode), and lines 2550 to 3450 controlling the VDU variables and the push and pull routines which switch between screen write modes.

CHANGING THE DISPLAYED MODE

The start of vertical sync interrupt is used to start the set up of timer1 in the User VIA to generate an interrupt after a period of delay1 and another after delay2. This is done

by setting the counter to free run mode (lines 1610 to 1640) and loading delay1 into the counter and delay2 into the latches (lines 2060 to 2110). In free run mode, after delay1, the counter is automatically loaded from the latches (in this case with delay2) and the count started. The values of delay1 and delay2 are such that the two interrupts occur when the raster scan is at the correct positions down the screen (as specified by the formula given last month).

At these points, the Video ULA registers are changed to display a different mode. This change of mode being performed with routines screena, screenb and screenc. The section A mode is set up on receipt of the vertical sync event, section B on receipt of the timer interrupt after delay1 and section C is initiated by the interrupt after delay2. A flag ('state') is used to identify which mode should be displayed.

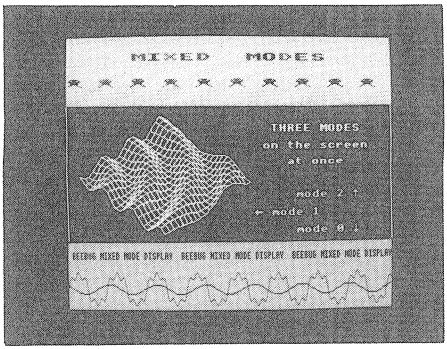
Of the two registers within the Video ULA, one controls such things as number of characters per line (memory mapped at location &FE20) and is called the Video Control Register. The second register defines the palette (the relation between logical and physical colours) and is located at &FE21. The palette control is rather complex and requires 16 values to be written to location &FE21. It is recommended that you read chapter 19 of The Advanced User Guide if you wish to know more of this.

CHANGING THE SCREEN WRITE MODE

Selection of the screen write mode is achieved by placing the VDU variables for the mode desired into the correct places in memory.

MIXMODE uses three data areas (at vduvara, vduvarb and, vduvarc) to store the VDU variables for the three modes. The data placed in the Video ULA each time the screen write mode is changed is obtained from these areas. The data structure for each of these areas is:

vduvarx	: 16 bytes to be written to the palette register of the Video ULA



vduvarx+&10	: system variable &248
vduvarx+&11	: system variable &249
duvarx+&12	: vdu variables &D0 to &D9
vduvarx+&20	: vdu variables &300 to &37F

When routines pusha, pushb or, pushc are called the vdu variables are stored away into the appropriate vduvar data area. These variables are re-established by the appropriate pull routines (called from the MIXDEMO program, for example) when it is desired to write to a particular mode. It is this action that 'fools' the Beeb into thinking it's in a certain mode when graphics or text is drawn on the screen. Hence data is placed into the screen memory in the right format for each mode area.

This combination of continually switching between displayed modes at precisely the right moment to synchronize with the raster scan and, on command, changing the Beeb's internal idea of which mode is displayed, creates the illusion of three modes on the screen at once, each separately accessible.

This is the breakdown of the assembly language section of the MIXMODE program:

LINES	: FUNCTION
1000 to 1090	: Initialise machine code locations
1130 to 1240	: Set up section modes and windows

170	: Data used to calculate values for Video ULA palette register
1430 to 1560	: Set up interrupt & event vectors
1580	: Enable vertical sync event
1600	: Disable ADC
1610 to 1640	: Set up User timer1 for free run mode
1720 to 1790	: Synchronize internal clock
1800 to 1880	: Set flash bit
1970 to 2010	: Ascertain source of interrupt
2070 to 2110	: Set up User timer1 with delay1 and delay2
2170 to 2190	: Check which mode should be selected
2340 to 2540	: Change Video ULA registers
2560 to 3440	: push and pull routines
2830 to 2980	: Save VDU variables
2990 to 3160	: Calculate Video ULA palette register values
3270 to 3440	: Restore VDU variables

Resident integers A% to H% are used to pass machine code addresses to the application program.

FURTHER NOTES

That is the basic operation of the MIXMODE program. However there are some further refinements.

CONTROL OF SYSTEM INTERRUPTS

The time taken for each graphics line to be drawn is a mere 64 micro seconds (about forty 6502 instructions). Therefore, any small delays in initiating the timer1 will mean that the place where the modes change will vary and a considerable amount of "jitter" may occur on the screen. For this reason, the system interrupts have to be controlled such that they do not affect the point at which timer1 is set up.

During normal operation, only three interrupts are occurring constantly. These are the start of vertical sync, 100Hz internal clock and, end of ADC conversion. Unfortunately, these interrupts are all running asynchronously and it is this that could cause uncertainty in the point at which the timer is set up. As mentioned in the previous article, the ADC is disabled but the same cannot be done for the internal clock as this causes the machine to lock up. Instead, the 100Hz clock is sychronized to the vertical sync which occurs at a frequency of 50Hz. This is done by resetting timer1 of the System VIA (which controls the 100Hz clock) every vertical sync such that it produces two interrupts every vertical sync (lines 1720 to 1790). This implements the 100Hz clock but the accuracy is affected slightly.

With the interrupts controlled in this way jitter is confined to one graphics line only. The effects of not controlling the interrupts can be seen by removing line 1580 and lines 1720 to 1790.

It should be noted that use of other Beeb functions which use interrupts (such as the RS423 or the speech processor) within your programs will affect this jitter.

FLASHING COLOURS

Under normal operation the least significant bit of location &248 is toggled to control the flash of logical colours 8 to 15. The Video ULA is updated constantly from this location. When running MIXMODE, the Video ULA is not updated from location &248 but from the vduvarx data areas. For this reason, the appropriate location within the data areas is updated every vertical sync (lines 1800 to 1880) and therefore, as you have seen from the demo program, full flashing colours are supported.

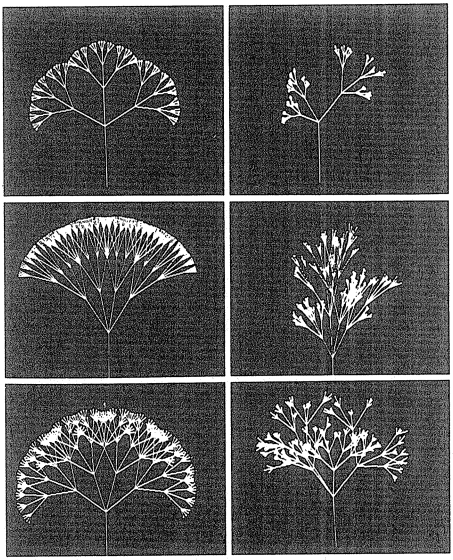
ULA REGISTER CHANGES

The Video ULA registers are changed with routines screena, screenb and screenc. Two actions are performed in these routines; one being the loading of the Video Control Register and the other the writing of the 16 values to the palette register. The order in which these two things are done produces different effects on the graphics line at the join of two modes (screenc is different to a and b). The effect will vary depending on the modes each side of the split, and some trial and error may be required to get the order right for any particular combination of modes.

EXPLORE THE WORLD OF ART AND GRAPHICS

Books about the Beeb's graphics are commonplace but the books reviewed here looked to be something rather special. Colin Cohen has been looking at them with interest and now reports.

The Art of Microcomputer Graphics by Jim McGregor and Alan Watt, Addison-Wesley, 444 pages at £14.95.



This is no beginners book, indeed even the adequate programmer not up in maths will not have an easy time of it, grasping the programs, but not the maths behind them. There are few concessions to the debutante in graphics (or in anything else) and the basic ground-work more or less stops with the explanation that the Beeb's graphic co-ordinates remain constant in all modes with the pixel (picture element) changing in size. This is certainly a very substantial compendium copiously filled with programs, diagrams and illustrations, including 16 pages of full colour plates. All aspects of computer graphics are covered here quite comprehensively, with major sections on two and three dimensional graphics.

A significant part of the book is devoted to tesselations and other repeating patterns. These are shown nested and in hierarchies, creating patterns which can be scaled, re-oriented, repeated and re-positioned. The main text is accompanied by a very substantial number of program listings to illustrate many of the concepts, and as the maths becomes more complex copious diagrams are introduced to demonstrate some of the concepts of movement, area and colour. Many of the listings and illustrations relate to Graphito, Tesselator and an earlier program, The Electronic Colouring Book [Graphito and Tesselator are two new graphics packages by Addison-Wesley which we hope to review shortly]. The book is probably definitive, it is certainly packed with ideas and information, but no-one could accuse it of being user-friendly!

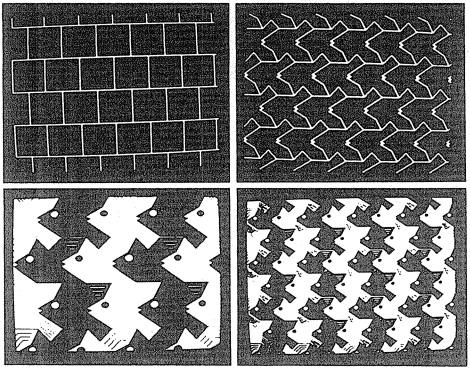
Soft Computing by Brian Reffin-Smith, Addison-Wesley, 208 pages £10.95.

If McGregor and Watt tell you how, Reffin-Smith tells you why. Tutor in Computing at the Royal College of Art and himself a maker of computer-based art and design works, he has produced a stimulating book; if you liked 'Zen and the Art of Motorcycle Maintenance', then this is for you. If on the other hand you want a down-to-earth factual introduction to computer graphics, then look elsewhere. Starting at the back, you will find a glossary, and some useful nuggets in 'Outlines for courses on the use of computers in art and design'. Working towards the front, you'll find interviews with five artists working in the field and accounts of the work of some nine more. Elsewhere there are excellent descriptions and explanations of what is actually going on - what a computer

is and does; but this is not where Reffin-Smith's heart lies. His work is an apologia for soft computing, which he sees as a qualitative activity, to do with art. To avoid its being lumped in with the trivia known as 'computer art' he has coined the word 'meta'. Soft computing is qualitative, conceptual, highly complex, multi-referential, probably political. He teases the reader: what's really going on is always in the next paragraph, the next level down. That's how it really is, with different levels of metaphor instead of one true description.

He examines our notions of what is a work of art, and of creativity, suggesting how it is possible to handle values and qualitative data with a computer. Not at all an easy book, but one which raises a number of interesting ideas.

As a production the book is disappointing: the exotic cover tantalizes like a come-on computer game package: inside the covers is grey text, interspersed with correction lines which have the weight of sub-heads, diagrams apparently off the backs of envelopes and eleven stingy



little colour reproductions, none larger than 65mm x 95mm.

All three authors are university lecturers. One used to tell children to 'Do as I say, not as I do', but in the case of these two books it may be easier to do as they do, rather than as they say, and judge the worth of the authors by the Beeb programs that they have written - McGregor and Watt for Addison-Wesley and Brian Reffin-Smith for BBC Publications.

CROSS REFERENCER UPDATE

While in use at the BEEBUG office, it became apparent that the Cross Referencer program as listed in the magazine (BEEBUG Vol.3 Vol.6) did not correctly handle all the information it should when dealing with large Basic programs. The author, Ian Gooding, has investigated the problem and come up with the following amendments to the original program (note the use of underline characters). Cassette users will also find motor control is almost essential when using this program. We hope this now clears up any problems that may have arisen in the use of the original version.

```
  270 count%=0
  370 CLOSE #0:MODE 7:PROCreport
 1430 def%=FALSE:fnc%=FALSE:quote%=FALSE:gos%=FALSE
 1680 fnc%=(?i%=&A4)
 1720 UNTIL j%>eline% OR NOT (FNletter(?j%) OR ?j%=ASC("£") OR ?j%=ASC("_") OR
FNdigit(?j%))
 1820 IF quote% OR NOT (FNletter(?i%) AND ?i%<>ASC("£") AND ?i%<>ASC("_")) THEN GOTO
1890
 1860 UNTIL i%?eline% OR NOT (FNletter(?i%) OR FNdigit(?i%) OR ?i%=ASC("£") OR
?i%=ASC("_"))
 2060 P%=lenumb:[OPT 2
 2130 ]
 3010 LOCAL p%,i%,end%,a$:p%=!!(ref%+4):a$=STRING$(6," ")
 3040 i%=2:end%=FALSE
 3770 CLOSE#0:END
```

BEGINNERS THIS WAY

INTRODUCING MACHINE CODE (Part 4)

This month Gordon Weston concludes his introduction to machine code for beginners by looking at the use of zero page and does simple arithmetic.

In the last article, I introduced a simple input routine which was capable of storing 256 input characters using the instruction STA &7E58,X where &7E58 was the start address of a block of 256 consecutive addresses and the value in X pointed to the address we wanted in that block. We can also use this type of instruction with the X or Y register to copy up to 256 characters from one part of memory to another. First enter our standard program 5:

Program 5

```
10 MODE7
20 DIM code 100
30 FOR I%=0 TO 3 STEP 3:P%=code
40 [
50 OPT I%

500 ]
510 NEXT
520 CLS:CALL start
530 END
```

Now enter these assembly lines and run the program.

Part Program 10

```
100 .start
150 LDY #0
160 .loop
170 LDA &7A00,Y
180 STA &7C00,Y
190 INY
200 BNE loop
210 RTS
```

This program will transfer 256 bytes of memory, starting from address &7A00, into mode 7 screen memory, which starts at &7C00. When the program is run nothing significant appears on the screen because we have not loaded anything into memory for the routine to copy. Type in immediate mode:

```
FOR I%=0 TO 255:I%?&7A00=65:NEXT
```

which loads the number 65 into the 256 locations in memory starting at address &7A00 (See User Guide p.409 onwards), run the program again, and a block of 256 A's should now appear at the top of the screen. If you now type:

CLS <Return>

which clears the screen and thus sets all of mode 7 screen memory to zero, and then:

CALL start <Return>

it will have the same effect as running the program because 'start' is a Basic variable set up in line 100 to contain the start address of the assembled machine.

The instruction LDA &7A00,Y has a limitation because the address &7A00 (or whatever is used) cannot be readily changed by the program. There is a more flexible instruction, which overcomes this limitation, in the form:

```
LDA (&70),Y
```

where the address within the brackets in the instruction tells the microprocessor where the address it wants is stored. The address &70 (actually &0070) is called a ZERO PAGE address because its first byte is zero. Zero page is the very first page of memory. The BBC micro operating system reserves sixteen of these valuable zero page addresses (&70 to &8F) for your own use. The instruction in the form LDA (&70),Y only works in zero page and only with the Y register.

Alter line 170 to LDA (&70),Y and add these new lines before running the program again:

```
110 LDA #00
120 STA &70
130 LDA #&7A
140 STA &71
```

These extra lines store the address &7A00 in zero page location &70. If you look at the way the address &7A00 is stored in lines 110 to 140 you will see that the least significant byte (LSB) or low byte of the address, which is &00, is loaded into address &70 and the most significant byte (MSB) or high byte of the address, which is &7A, is loaded into the address &71 which is where the microprocessor expects to find it. A memory address is always stored as two bytes with the low byte first and the high byte second. This program writes 256 A's on the screen as before, but the most important point is that by adding a few lines in the program to change the address stored in addresses &70 and &71, a new section of memory could be displayed on the screen. To make full use of this new instruction you will also have to learn to add and subtract in order to alter the address stored in addresses &70 and &71.

Adding is done in the accumulator which can only store one byte at a time. To add one two byte number to another two byte number you load the accumulator with the LSB (least significant byte) of the first number and use the instruction ADC (ADd with Carry) to add the 'LSB' of the second number. Then store the contents of the accumulator, which is the result, safely back in memory. You then do exactly the same with the MSB's (most significant bytes).

There is a slight snag which occurs when the result of two bytes added together exceeds 255, the maximum number that the accumulator can hold. When this happens a flag, called the 'Carry' flag, is set and the instruction ADC (ADd with Carry) takes this flag into account when the next two bytes are added together. For this reason you must always clear the carry flag using the instruction 'CLC' before starting an addition routine. You should be able to see the pattern of the routine in Program 11 below.

Subtraction follows the same pattern except that you set the carry flag using 'SEC' before the routine and use the instruction 'SBC' for subtracting with carry. The techniques for addition and subtraction are very much like the manual techniques we all learnt at school, with a 'carry' from tens to units and so on. In binary arithmetic, of course, a carry is always 1 or 0.

The number of instructions is now increasing and to make the part programs more compact we will now use more than one statement on a line, separated by colons, which the assembler still recognises as does Basic. As with Basic, this can make programs more difficult to read, and should not be overdone. Delete lines 110 to 210 in Program 10, enter the new lines 110 to 200 below and run the new program:

Part Program 11

```
100 .start
110 LDA #0:STA &70
120 LDA #&7C:STA &71
130 LDY #0:LDX #20
140 .loop
150 LDA #255:STA (&70),Y
160 CLC
170 LDA &70:ADC #41:STA &70
180 LDA &71:ADC #0:STA &71
190 DEX:BNE loop
200 RTS
```

This new program displays a diagonal line of squares (ASCII 255) on the mode 7 screen. Lines 110 and 120 load the first screen address in addresses &70 and &71, and the addition routine to change this screen address is in lines 160 to 180. Notice that although each line on the screen consists of 40 characters, we are adding 41 to the screen address at line 170 which gives a staircase effect when we print ASCII character 255 at line 150.

Although line 180 seems to be just adding zero to the most significant byte it is also adding in the carry flag in case it has been set. The accumulator has to be reloaded at line 150 each time because the accumulator has been used in the adding routine.

The Y register remains at zero and the X register is used as a loop counter which is rather wasteful because we can produce the same result with just the Y register by entering the lines below:

```
130 LDY#0
170 LDA &70:ADC #40:STA &70
190 INY:CPY #20:BNE loop
```

This concludes our brief introduction to machine code under the heading of 'Beginners Start Here'. If your interest has been aroused, then there are many books on this subject, such as that by Ian Birnbaum reviewed in BEEBUG Vol.3 No.6. Although this is the last article in this particular series, we shall be publishing further instructional articles on the use of machine code in future issues of BEEBUG.

MORE HASTE LESS SPEED

Reading the recent articles in BEEBUG (Vol.3 Nos.7 & 8) about indirection operators reminded me of my experience with the "?" operator. I wanted to speed up a program so I converted integer variables with values in the range 0 to 255 to use single bytes of zero page memory accessed by the "?" operator.

However, far from speeding up the program it actually slowed it down slightly. I'm at a loss to explain this; it seems to me that altering one byte of memory should be faster than altering the 4 bytes in a Basic integer variable.

Lorcan Mongey

Mr Mongey is quite right in his experiences. Using Basic's integer variables is faster than accessing single bytes with "?", certainly in our tests. We believe the explanation lies in the fact that the indirection operator, as its name implies, requires an extra layer of memory access compared to the Basic integers. Of course, using "?" with single memory bytes considerably reduces a program's memory requirements which may in itself be sufficient justification.

PUTTING THE PLUS IN WORDWISE

I swapped Wordwise for Wordwise Plus not long ago and was very pleased to read the review of Wordwise Plus in the March issue (BEEBUG Vol.3 No.9).

Please can we have some listings of programs to use in Wordwise Plus to help your many readers who do a lot of word processing.

Frederic Haas

We already have this in hand and expect to publish a number of useful segment programs in next month's issue of BEEBUG.

ELITE AT LAST

I purchased a double density disc interface with a 1.4 DDFS and disc Elite would not run. On phoning Watford I was told that I did not have the latest version of the DDFS but that with version 1.5 Elite should be OK. Send £5 etc... I did, it was and the kids were happy.

I have generally little success in transferring tape programs to disc, so decided to buy Watford Electronics' "Disc Executor". To my surprise this would not run correctly. On phoning Watford I was told that it would not work with a Watford DDFS. So be warned, there is no guarantee that a Watford DDFS will run even Watford software

R.K.Greenwood

Mr P.Christy is another member who has written regarding Elite. As we said, Elite will run with Watford DFS version 1.4 and with most versions of 1.3. This is also apparently true of the Island Logic Music System (see this issue of BEEBUG) which also employs protection. It seems you need at least version 1.5 of the Watford DDFS to run Elite.

LUNARBUG

Some of the more faithful of BEEBUG readers may remember the popular 'Lunar Lander' programs of yesteryear with their origins on mini and mainframe computers. Alan Dickinson has updated this theme with a new version to delight young and old alike.

LunarBug is a good old fashioned computer game that requires you to navigate your space craft to a safe landing on the moon's surface. The game features good graphics and provides a serious challenge to the player. You have to land your craft before the fuel runs out (there isn't a lot of it to start with) and at a certain speed. If you find that you are constantly running out of fuel then you can change the initial setting of the fuel at line 2720 to make the game easier. If you change the variable F% to 8000 then this will give you twice as much fuel.

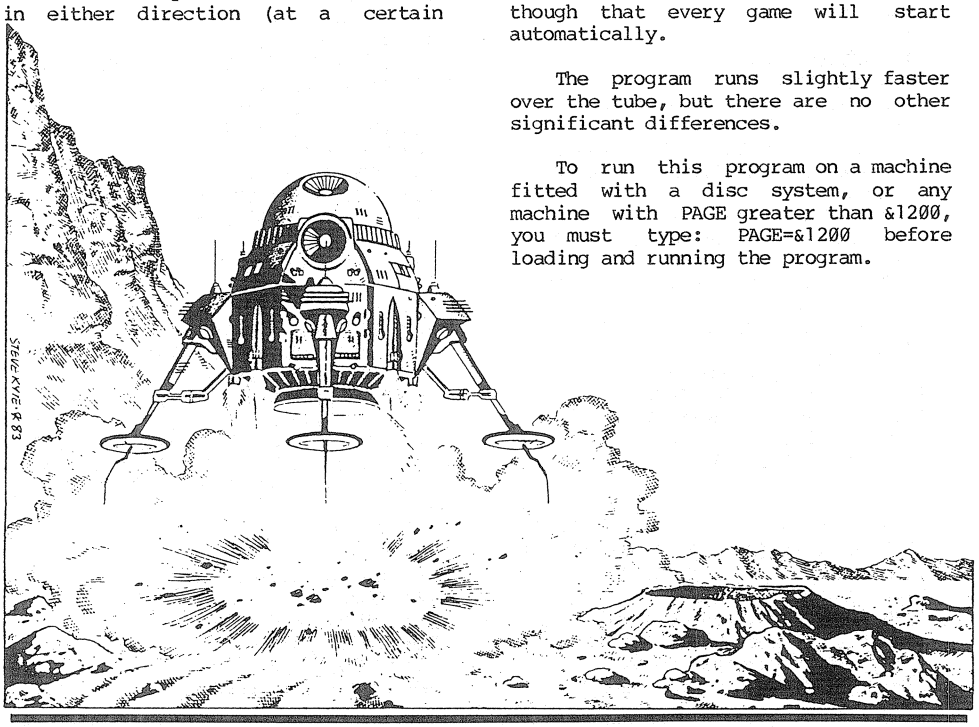
The craft has to be landed with a vertical speed of less than 30 m/s and a horizontal speed of less than 10 m/s in either direction (at a certain height these are diplayed at the bottom of the screen for greater convenience). The acceptable landing speeds can be changed by altering the two values -30 and 10 at line 3030. The keys 'Z' and 'X' fire the left jet and right jets respectively, and the '/' key fires the main rocket.

The program itself is very well structured, to make it fast, and it is well documented with plenty of remarks included within the code.

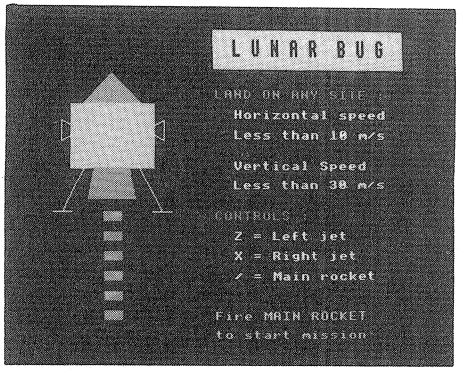
You can reduce the amount of typing needed to enter this program by omitting the program's introductory scene. To do this leave out line 170 and lines 1120 to 1600. This means though that every game will start automatically.

The program runs slightly faster over the tube, but there are no other significant differences.

To run this program on a machine fitted with a disc system, or any machine with PAGE greater than &1200, you must type: PAGE=&1200 before loading and running the program.

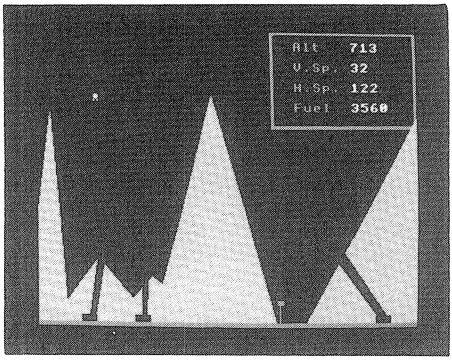


```
  10 REM PROGRAM LUNAR
  20 REM VERSION B0.2
  30 REM AUTHOR  ALAN DICKINSON
  40 REM BEEBUG  MAY 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 DIM b 9:osword=&FFF1
 110 ON ERROR MODE7:PROCabend
 120 :
 130 MODE1
 140 PROCdefines
 150 REPEAT
 160 PROCpallette
 170 PROCintro
 180 PROClandscape
 190 PROCmission
 200 TIME=0:REPEAT UNTIL TIME>333
 210 UNTIL FALSE
 220 :
1000 DEFPROCabend
1010 ON ERROR OFF
1020 *FX15
1030 REPORT:PRINT" at line ";ERL
1040 IF ERR=17 END
1050 com$="L."+STR$(ERL)+CHR$13
1060 FOR i=1 TO LEN(com$)
1070 X%=0:Y%=ASC(MID$(com$,i,1))
1080 A%=&8A:CALL &FFF4
1090 NEXT
1100 END
1110 :
1120 DEFPROCintro
1130 CLS:GCOL0,1
1140 MOVE300,500:MOVE200,400
1150 PLOT85,400,400
1160 GCOL0,2:MOVE160,400:MOVE440,400
1170 PLOT85,160,200:PLOT85,440,200
1180 GCOL0,1
1190 MOVE240,200:MOVE360,200
1200 PLOT85,220,100:PLOT85,380,100
1210 GCOL0,3
1220 MOVE 160,300:PLOT1,-32,-16:PLOT1,
0,64:PLOT1,32,-16
1230 MOVE 440,300:PLOT1,32,-16:PLOT1,0
,64:PLOT1,-32,-16
1240 MOVE208,200:DRAW132,60
1250 MOVE392,200:DRAW468,60
1260 MOVE100,60:DRAW164,60
1270 MOVE436,60:DRAW500,60
1280 VDU28,20,31,39,0
1290 COLOUR0:COLOUR130
1300 PRINTSPC(20);
1310 PROCprint2("  L U N A R  B U G  "
1320 PRINT'SPC(20):COLOUR128:COLOUR1
1330 PRINT'"LAND ON ANY SITE :"'
1340 COLOUR3
1350 PRINT"  Horizontal speed"'
1360 PRINT"  Less than 10 m/s"''
1370 PRINT"  Vertical Speed"'
1380 PRINT"  Less than 30 m/s"''
```




```
1390 COLOUR1
1400 PRINT"CONTROLS :"'
1410 COLOUR3
1420 PRINT"  Z = Left jet"'
1430 PRINT"  X = Right jet"'
1440 PRINT"  / = Main rocket"
1450 COLOUR2
1460 PRINTTAB(0,28)"Fire MAIN ROCKET"'
'"to start mission";
1470 *FX15
1480 GCOL3,3
1490 REPEAT
1500 IF INKEY-98 MOVE96,332:VDU5,249,8
,249,4:SOUND0,-10,4,1
1510 IF INKEY-67 MOVE472,332:VDU5,249,
8,249,4:SOUND0,-10,4,1
1520 UNTIL INKEY-105
1530 VDU28,0,31,19,0
1540 FOR j%=1 TO 16
1550 SOUND0,-10,4,6
1560 MOVE272,60:VDU5,249,249,4
1570 PRINTTAB(0,31);:VDU10,10
1580 NEXT
1590 ENDPROC
1600 :
1610 DEFPROCprint2(A$)
1620 LOCAL X%,Y%,A%,j%,k%
1630 A%=&A:X%=b MOD256:Y%=b DIV256
1640 FOR j%=1 TO LEN(A$)
1650 ?b=ASC(MID$(A$,j%,1))
1660 CALL osword
1670 VDU23,224,b?1,b?1,b?2,b?2,b?3,b?3
,b?4,b?4,23,225,b?5,b?5,b?6,b?6,b?7,b?7
,b?8,b?8,224,10,8,225,11
1680 NEXT
1690 ENDPROC
1700 :
1710 DEFPROCpallette
1720 VDU20
1730 VDU19,2,6,0,0,0
1740 VDU19,1,5,0,0,0
1750 VDU19,3,3,0,0,0
1760 VDU23;8202;0;0;0;
```

```
1770 ENDPROC
1780 :
1790 DEFPROCdefines
1800 VDU23,255,96,240,240,96,144,144,0
;
1810 VDU23,254,0,0,0,0,60,60,60,60
1820 VDU23,253,0,0,0,0,126,126,126,126
1830 VDU23,252,192,248,255,248,192,0,0
,0
1840 VDU23,251,0,0,24,60,126,219,0,0
1850 VDU23,250,3,7,15,7,3,0,0,0
1860 VDU23,249,85,170,85,170,85,170,85
,170
1870 ENDPROC
1880 :
1890 DEFPROClandscape
1900 VDU26:CLS
1910 :
1920 REM Console
1930 REM _______
1940 :
1950 GCOL 0,1
1960 MOVE788,718:MOVE788,1023
1970 PLOT85,1279,718:PLOT85,1279,1023
1980 GCOL0,0
1990 MOVE800,730:MOVE800,1015
2000 PLOT85,1267,730:PLOT85,1267,1015
2010 COLOUR1
2020 PRINTTAB(27,1);"Alt";TAB(27,3);"V
.Sp.";TAB(27,5);"H.Sp.";TAB(27,7);"Fuel
";
2030 :
2040 REM rough terrain
2050 REM _____________
2060 :
2070 GCOL 0,2
2080 X%=-50
2090 REPEAT
2100 x%=X%+RND(199):IF x%>650 x%=650
2110 z%=x%+RND(199):IF z%>650 z%=650
2120 Y%=RND(200)+110
2130 IF x%>450 Y%=RND(450)+400
2140 IF x%<200 Y%=RND(350)+500
2150 MOVE X%,100:MOVEx%,100
2160 PLOT 85,x%,Y%
2170 PLOT 85,z%,100
2180 X%=x%
2190 UNTIL z%=650
2200 PLOT 85,800,100
2210 :
2220 REM Beacon
2230 REM ______
2240 :
2250 MOVE 812,100:DRAW 812,160
2260 :
2270 REM Steep mountainside
2280 REM ___________________
2290 :
2300 MOVE 900,100:MOVE 1279,100
2310 PLOT 85,1279,RND(100)+700
2320 :
```




```
2330 REM Landing silos
2340 REM _____________
2350 :
2360 PROCtunnel(RND(100)+100,128,16)
2370 PROCtunnel(RND(80)+1100,-700,28)
2380 PROCtunnel(RND(150)+350,0,12)
2390 :
2400 REM Baseline
2410 REM ________
2420 :
2430 GCOL 0,1
2440 MOVE 0,100:MOVE0,88
2450 PLOT85,1279,100:PLOT85,1279,88
2460 ENDPROC
2470 :
2480 : Carve tunnels in mountain
2490 : and terminate with hangar.
2500 :
2510 DEFPROCtunnel(X%,A%,W%)
2520 GCOL 0,0
2530 MOVE X%,100
2540 MOVE X%+A%,1000
2550 PLOT 85,X%+W%,100
2560 PLOT 85,X%+A%+(3*W%),1000
2570 MOVE X%-24,100
2580 MOVE X%-24,124
2590 PLOT 85,X%+24,100
2600 PLOT 85,X%+24,124
2610 ENDPROC
2620 :
2630 : Control handed over to
2640 : pilot with module on
2650 : final approach course.
2660 :
2670 DEFPROCmission
2680 X%=5000:x%=X% DIV100
2690 Y%=80000+RND(20000):y%=Y% DIV100
2700 PROCimage
2710 DX%=50:DY%=-1
2720 F%=4000 :REM fuel
2730 B%=TRUE :REM beacon
2740 G%=-1   :REM gravity
2750 C%=FALSE:REM end condition
2760 T%=0:TIME=0
```

```
 2770 :
 2780 REPEAT
 2790 HF%=0
 2800 IFF%>0 IFINKEY-67 PROCside(-1)
 2810 IFF%>0 IFINKEY-98 PROCside(+1)
 2820 DX%=DX%+HF%
 2830 :
 2840 VF%=-1
 2850 IFF%>0 IFINKEY-105 PROCfire
 2860 DY%=DY%+VF%
 2870 :
 2880 PROCimage
 2890 X%=X%+DX%:x%=X% DIV100
 2900 Y%=Y%+DY%:y%=Y% DIV100
 2910 P%=POINT(x%,y%)
 2920 PROCimage
 2930 IFY%<25000 PRINTTAB(26,30);-DY%;"
";
 2940 :
 2950 IFX%<0 OR X%>127900 C%=TRUE
 2960 IFY%<10100 C%=TRUE
 2970 IFP%=2 C%=TRUE
 2980 IFTIME>T% PROCtimeout
 2990 UNTIL C%
 3000 :
 3010 PROCtimeout
 3020 PRINTTAB(31,30)"TIME ";T% DIV100;
" ";
 3030 IFX%<0 OR X%>127900 PROCabort ELS
E IFP%=2 PROCcrash ELSE IFDY%<-30 OR AB
S(DX%)>10 PROCcrash ELSEPROClanded
 3040 ENDPROC
 3050 :
 3060 : draw/erase module image
 3070 : at screen position x%,y%
 3080 :
 3090 DEFPROCimage
 3100 VDU25,4,x%-8;y%+20;18,4,0,5,255,4
 3110 ENDPROC
 3120 :
 3130 : fire lateral rockets
 3140 :
 3150 DEFPROCside(D%)
 3160 GCOL3,1:MOVEx%-16-16*D%,y%+24
 3170 VDU5,251+D%,8
 3180 HF%=D%:F%=F%-1:SOUND&10,-10,5,5
 3190 VDU251+D%,4
 3200 ENDPROC
 3210 :
 3220 : fire main rocket
 3230 :
 3240 DEFPROCfire
 3250 GCOL3,1:MOVEx%-16,y%:VDU5,251,8
 3260 VF%=2:F%=F%-8:SOUND&10,-10,5,4
 3270 VDU251,4
 3280 ENDPROC
 3290 :
 3300 : update console display
 3310 :
 3320 DEFPROCtimeout
 3330 T%=TIME+100
 3340 COLOUR3
 3350 SOUND 1,-12,100,1
 3360 PRINTTAB(33,1);Y% DIV100-100;"  "
;TAB(33,3);-DY%;"  ";TAB(33,5);DX%;"  "
;TAB(33,7);F%;"  ";
 3370 IFF%<1 PRINTTAB(33,7);"EMPTY";:EL
SEIFF%<500 PRINTTAB(31,30)"LOW FUEL";
 3380 COLOUR1
 3390 PRINTTAB(25,26);CHR$(254+B%);
 3400 B%=NOT B%
 3410 ENDPROC
 3420 :
 3430 : Crashes can be caused by
 3440 : hitting the mountain, or
 3450 : landing too fast.
 3460 :
 3470 DEFPROCcrash
 3480 SOUND &0010,-15,4,2
 3490 VDU19,0,12,0,0,0
 3500 VDU19,3,0,0,0,0
 3510 FOR i%=1 TO 30
 3520 MOVE x%,y%
 3530 GCOL 0,RND(4)
 3540 DRAW RND(1279),100+RND(700)
 3550 SOUND 0,-15,4+RND(2),2
 3560 NEXT i%
 3570 FOR I%=1 TO 40
 3580 SOUND 1,-15,3*I%,1
 3590 NEXT I%
 3600 PROCpallette
 3610 :
 3620 R%=RND(8)
 3630 IF R%=1 R$="No Survivors"
 3640 IF R%=2 R$="Another crater..."
 3650 IF R%=3 R$="Wreckage over 2 Km"
 3660 IF R%=4 R$="Expensive repairs"
 3670 IF R%=5 R$="OOPS........"
 3680 IF R%=6 R$="Anybody fancy some sc
rap?"
 3690 IF R%=7 R$="Did somebody sneeze?"
 3700 IF R%=8 R$="Another Monday mornin
g!"
 3710 PRINT TAB(1,30);R$;
 3720 ENDPROC
 3730 :
 3740 : Mission is aborted when
 3750 : shuttle travels out of
 3760 : side of landscape
 3770 :
 3780 DEFPROCabort
 3790 FOR j%=1 TO 10
 3800 FOR k%=1 TO 2
 3810 COLOUR k%
 3820 PRINTTAB(1,30)"Mission Aborted";
 3830 SOUND 1,-15,100+k%*32,2
 3840 FOR l%=1 TO 1000:NEXT
 3850 NEXT
 3860 NEXT
 3870 ENDPROC
```

→

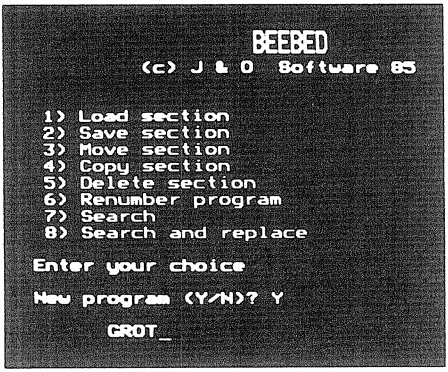
```
3880 :
3890 : Module is landed when
3900 : altitude is <101, and
3910 : touchdown speed is <30
3920 : vertically, and <10 in
3930 : either direction horiz.
3940 :
3950 DEFPROClanded
3960 R%=-DY%
3970 R$="K-E-R-T-H-U-M-P"
3980 IF R%>28 R$="C-R-U-N-C-H"
3990 IF R%<20 R$="T-H-U-D"
4000 IF R%<15 R$="B-U-M-P"
4010 IF R%<10 R$="Touchdown"
4020 IF R%<5  R$="Great Landing"
4030 PRINTTAB(1,30);R$;
4040 FOR I%=1 TO 100 STEP 4
4050 SOUND 1,-15,I%,3
4060 SOUND 2,-15,I%+20,2
4070 SOUND 3,-15,I%+32,1
4080 NEXT
4090 FOR i%=1 TO 5
4100 PRINTTAB(1,30)"REFUELLING";SPC(6)
4110 FOR j%=150 TO 200 STEP 5
4120 SOUND1,-15,j%,1
4130 NEXT
4140 PRINT TAB(1,30);SPC(10)
4150 FOR j%=200 TO 150 STEP-5
4160 SOUND1,-15,j%,1
4170 NEXT
4180 NEXT
4190 PRINT TAB(1,30)"EMERGENCY TAKEOFF"
4200 IFx%<250 idx=-1.5:dx=.5
4210 IFx%>250 ANDx%<500 idx=0:dx=.33
4220 IFx%>500 ANDx%<1000 idx=1.5:dx=.42
4230 IFx%>1000 idx=5:dx=.2
4240 :
4250 x=x%:y=y%:dy=7
4260 REPEAT
4270 PROCimage
4280 x=x-idx
4290 IF y>650 idx=idx+dx
4300 y=y+dy
4310 x%=x:y%=y:PROCimage
4320 SOUND0,-15,4,1
4330 SOUND1,-15,y% DIV9,1
4340 UNTIL x<0
4350 PRINTTAB(1,30)"MISSION ACCOMPLISH
ED"
4360 ENDPROC
```

22 ←

keys are used to good effect to scroll back and forth through your program, move directly to the top, the bottom, or a particular line, delete whole line, and so on. Beebed only actually makes the correction when Return is pressed so you can alter an entire page and then change your mind with the 'undo' key and everything is back to square one.

Pressing Escape, like Wordwise, gives you the main menu from which programs can be loaded and saved, sections of program moved around, deleted, renumbered, and searched for the occurrence of any string. These features are all thoroughly idiot proofed and have such options as remembering the last name you gave to a file with the option to use it again.

The only slight niggle with Beebed is that, despite its likeness to Wordwise, Shift with a cursor key moves



you one page up or down the program and Ctrl with a cursor key moves to the top or bottom - the exact opposite to Wordwise. Most confusing! However, this is merely a niggle that does not blemish an otherwise excellent, if a little expensive, product.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MAKING PROGRAMS RUN ON THE SECOND PROCESSOR - R.J.J. Orton

Many programs will run on the 6502 second processor despite first appearances if the value of PAGE is set to what the program expects. BEEBUGSOFT's Masterfile, for example, will run perfectly if PAGE is first set to &E00 or &1900 (not needed for Masterfile II). Although this gains little memory, the gain in speed is worthwhile.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Editor: Phyllida Vanstone
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.

Editorial Address
BEEBUG
PO BOX 50
St. Albans
Herts.

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 £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.40 6 months (5 issues) UK ONLY
£11.90 UK - 1 year (10 issues)
£18 Europe, £21 Middle East
£23 Americas & Africa, £25 Elsewhere

BACK ISSUES
(Members only)

Vol.	Single Issues	Volume sets (10 issues)
1	80p	£7
2	90p	£8
3	£1	£9
4	£1	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues &
Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks. HP10 8HQ

Hotline for queries and software orders

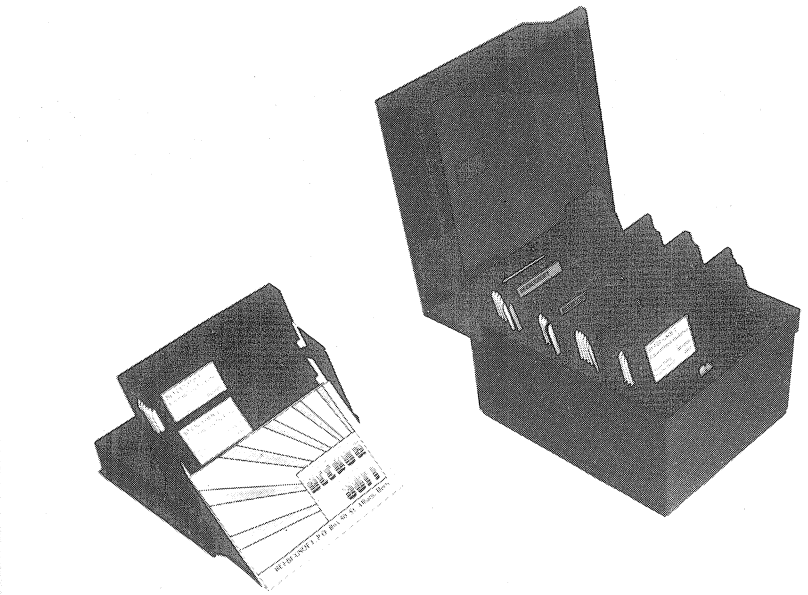
St. Albans (0727) 60263
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and
Barclaycard orders, and subscriptions
Penn (049481) 6666

If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

High Quality Low Priced Discs

Backed by The Reputation of BEEBUG



10 S/S D/D Discs – £13.90	10 D/S D/D Discs – £19.40
25 S/S D/D Discs – £33.45	25 D/S D/D Discs – £46.95
50 S/S D/D Discs – £59.30	50 D/S D/D Discs – £87.05

All Prices Include Storage Box, VAT and Delivery to Your Home (UK).

All discs are 100% individually tested, supplied with hub ring as standard, and guaranteed error free. They are ideal for use on the BBC Micro and have performed perfectly in extensive tests at BEEBUG over many months.

Orders for 25 or 50 are delivered in strong plastic storage boxes with four dividers. Orders for 10 are sent in smaller hinged plastic library cases.

We are also able to offer the empty storage container, which holds up to 50 discs for £10 including VAT and post.

Please use the order form enclosed
or order directly from:
BEEBUGSOFT, P.O. Box 109,
High Wycombe, Bucks HP10 8HQ.

BEEBUG SOFT

Magazine Cassette/Disc

MAY 1985 CASSETTE DISC CONTENTS

POLAR CURVES in 57 varieties

EXTENDED ASSEMBLER for second processors

FREE MEMORY DISPLAY for all Basic programmers

FLOWCHART GENERATOR with hardcopy output

WORKSHOP PROCEDURES - String Sort and File Sort

MAKING MUSIC - the latest additions

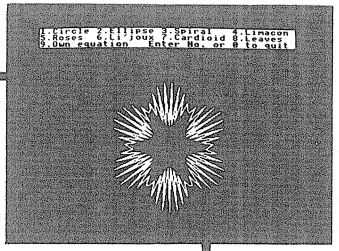
BEGINNERS EXAMPLES on using machine code

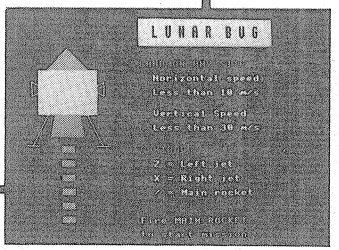
LUNAR BUG - a classic game with a new twist

EXTRA FEATURES THIS MONTH

SCRUMPY - colourful, smooth, fast-moving, addictive, action-packed - what more can we say about this first rate arcade game?

ENGLISH COUNTRY GARDEN
- A detailed and colourful display, to show off the graphics capability of the Beeb, and all with appropriate music.





All this for £3.00 (cass) £4.75 (disc) + 50p p&p.
Back issues (disc since Vol.3 No.1, cass since Vol.1 No.10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

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

Cassette subscriptions can be commuted to disc subscriptions on receipt of £1.70 per issue of the subscription left to run.

All subscriptions and individual orders to
BEEBUGSOFT, PO Box 109, High Wycombe, Bucks, HP10 8NP.

Printed in England by Staples Printers St Albans Limited at The Priory Press.

ISSN 0263-7561